

OCR F454 COMPUTING PROJECT

Computing Revision Game

A computational solution to help both students and teachers tackle the obstacles that stop effective revision taking place with adaptability and fun.

Candidate Name: Joe Rackham

Candidate Number: 5391

Centre: Tendring Technology College

Centre Number: 16455

Table of Contents

Problem Identification	2
Investigation and Analysis	4
Definition of the Proposed Solution	15
Design of the Solution	19
Design Investigation.....	19
Design Development.....	23
Database Design.....	30
Programming and Algorithms Design.....	34
Algorithms Flowcharts	37
Algorithms Pseudocode.....	43
Testing Strategy	59
Project Development	61
Phase 1: Scene Transition and movement.....	62
Phase 2: Database Connection	67
Phase 3: Quizzes.....	71
Phase 4: Items	82
Phase 5: Ending the Game.....	101
Phase 6: Editing the list of Questions.....	110
Phase: 7: Displaying Results	125
Creating and Implementing the Server	148
Building the Application.....	151
Robustness and Function	154
Alpha Testing.....	154
Beta Testing.....	195
Evaluation	211
Code Repository	217

Problem Identification

Introduction

Revision is problematic for both teachers and students. It is a necessary task to achieve at GCSE and A Level but teachers often find that students aren't revising for enough time or in an effective way. The head of my school's Computing Department has given me the brief for this project; to design and create an application to aid both students and staff tackle the issue of revision.

As I see it there are two main barriers to effective revision. Firstly, student engagement with conventional revision material is mostly low. Revision activities cannot hold the attention of the students for long periods of time and therefore insufficient effort is put in. Secondly, teachers are ill equipped to tailor Revision materials to their classes needs. Currently teachers rely on word of mouth from their students for information about how revision is going and what they feel should be worked in class. This method fails when students lie about their revision or feel embarrassed to say where they are struggling.

This problem is suited to being solved with a computational method. Schools have heavily restricted budgets and much of this goes towards equipment and lessons. An approach that takes the form of software wouldn't require paper to print out or extra equipment to be purchased. My Computing Department, which is representative of the situation in the vast majority of schools, is equipped with computers for the students; a computational solution would fit right into the existing environment. In addition, a computerised solution will contribute to the target held by school, as well as others, to move towards a completely paperless school.

Proposed Solution

I will aim to solve these problems by creating a revision application for Computing Students. To address the problem of Student engagement this application will take the form of a game with a changing set of questions to encourage repeated use. To address the problem of teacher interaction the application will report data on student performance back to staff members. Both the questions and student results will be stored in a central database that can be accessed by any instance of the software.

This proposed solution requires a computational solution and takes full use of the benefits available. A computerised solution will be able to store a great many more questions than a physical solution would feasibly be able to store. Furthermore, the use of databases will allow this information to be stored in an effective way and queried in a well-established way. Computers will greatly improve the scope and efficiency of the solution. Moreover, the content a solution must cover will be subject to change. Schools experience a change of qualification or exam board with relative frequency at the moment. The changing set of questions I have proposed will help deal with this issue. This system will give my solution adaptability, therefore extending the lifespan and usefulness of the system.

Client

My client, Mr G. Byford, is the Head of my Schools Computer Science Department. Mr Byford is responsible for overseeing teaching and learning in the computer science department; he has a responsibility to ensure that effective revision is taking place. Mr Byford has identified that revision is problematic at the moment and has therefore given me my brief. I believe the solution I have proposed will tackle the obstacles to revision, therefore, meeting Mr Byford's brief by improving the quality of revision taking place. Furthermore, my proposed solution will be specifically useful to Mr Byford in his role of overseeing teaching and learning by tracking student performance. It is impossible for Mr Byford to interact with every student taking computing so data is very valuable to him as a means to track students. My solution will track student performance and can therefore act as a data collection tool for Mr Byford, better equipping him to fulfil his role as a departmental head.

Description of End User

There will be two primary groups of end users for my application:

- Students using my application will be studying A-Level or GCSE Computer Science. These students will be studying multiple subjects in preparation for approximately 10 exams. Revision is essential for these students to succeed; they need quality revision materials to use. My solution meets the needs of these students by providing a tool for them to revise from. The changing question set will improve the reusability of my solution and by gamifying the quizzes I aim to improve student's engagement with the tool.
- Computer Science teachers using my application will be responsible for the learning of A-Level or GCSE students. These teachers need resources they can provide the students with and information of the students' performance which they can then use to tailor revision sessions to. My proposed solution will meet these needs by providing a heavily reusable revision tool teachers can give to their students. My solution will track student performance and provide teachers with the data they need to access how their class is tackling revision. By giving teachers the ability to change and add questions my solution will be even more useful as teachers can tailor the tool to their student's specific needs.

Investigation and Analysis

At this stage in the development of my Revision tool I need to conduct an Investigation so can make informed decisions about the evolution of my solution. My plan of research is to investigate existing solutions to similar problems, to gauge opinions from my end users with a questionnaire and to gain a better understanding of the brief through an interview with client.

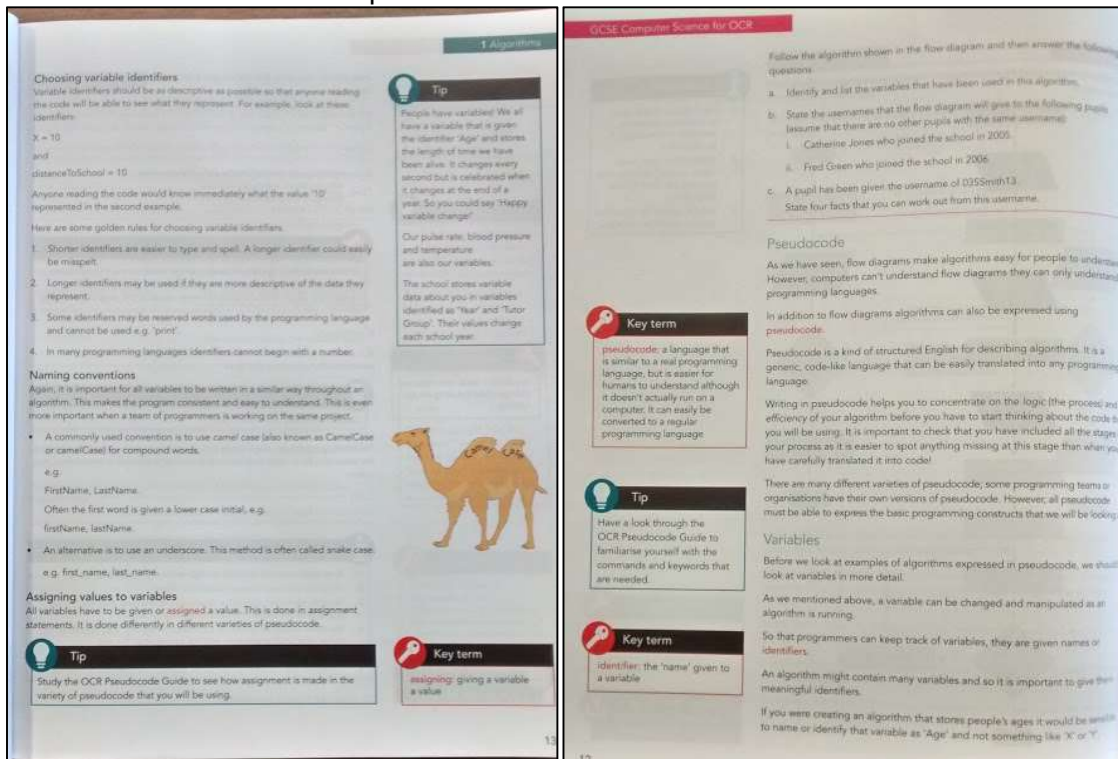
Observation of Current System

By looking at existing solutions to similar problems I am aiming to improve my proposed solution by identifying where they succeed and where they fail. I am hoping to display that current revision resources are lacking and justify the decisions I have made in my proposal. However, current revision tools are by no means useless and I aim to explore if what they do well can be applied to my own Revision tool.

Physical Revision Tools

Paper based revision tools are 'traditional' and still used extensively at my School despite their large cost. I looked at the Textbook and Revision Guide my school currently uses to instigate how useful they really are.

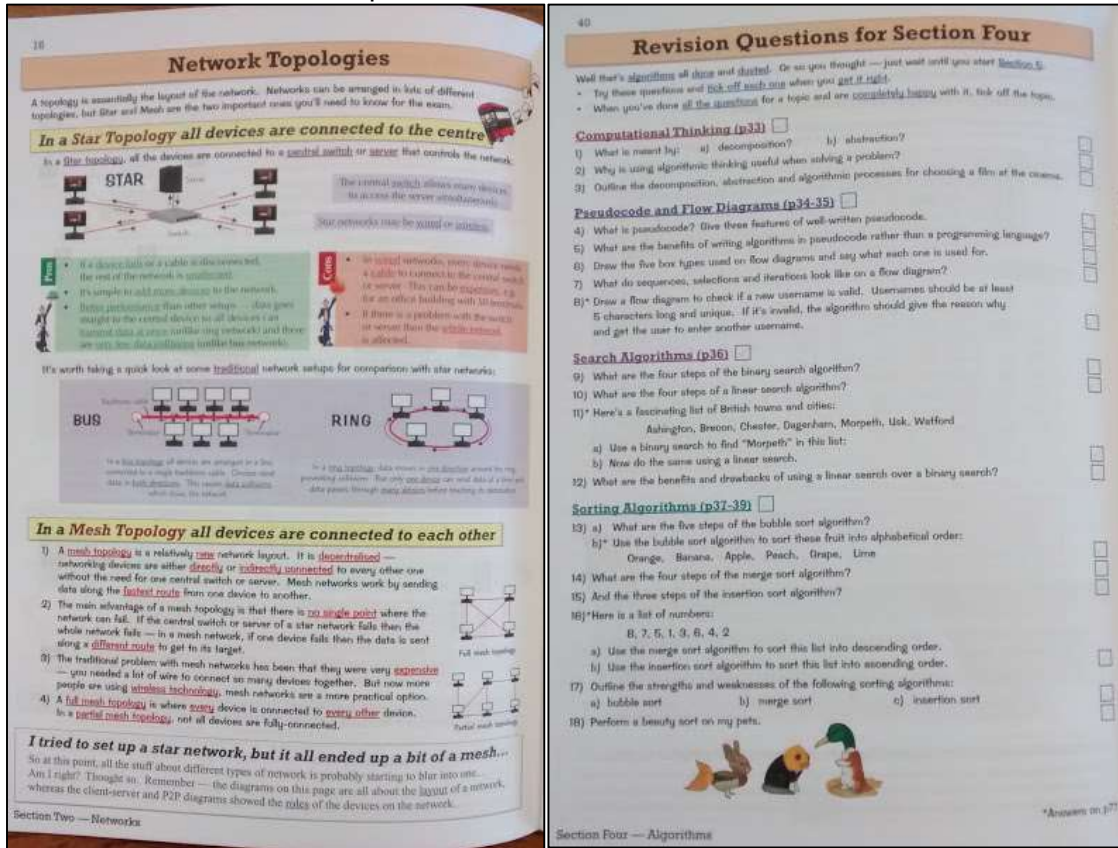
- I first looked at the Computer Science Textbook:



The textbook is made up of uninterrupted text with key points in extra textboxes in the side. This is problematic as this format might be difficult to hold students attention. In addition, only some of the important information is put aside in boxes, lots of it is in the text but not emboldened or highlighted in any way. The format of this textbook isn't very useful for revision because it is difficult to simplify the topic into the key points or test oneself on their knowledge.

The textbook caters to GCSE students on a specific specification, making it usable by only a very small group of students. The book retails at around £19, an incredible purchase if a school wishes to provide for every student.

- I also looked at the Computer Science Revision Guide:



The Revision Guide effectively divides information into short topics and uses underlining and boxing to make clear what the important information is. The use of colours and diagrams helps keep students engaged and the Revision Guide also includes jokes and has a sense of humour to appeal to its young audience.

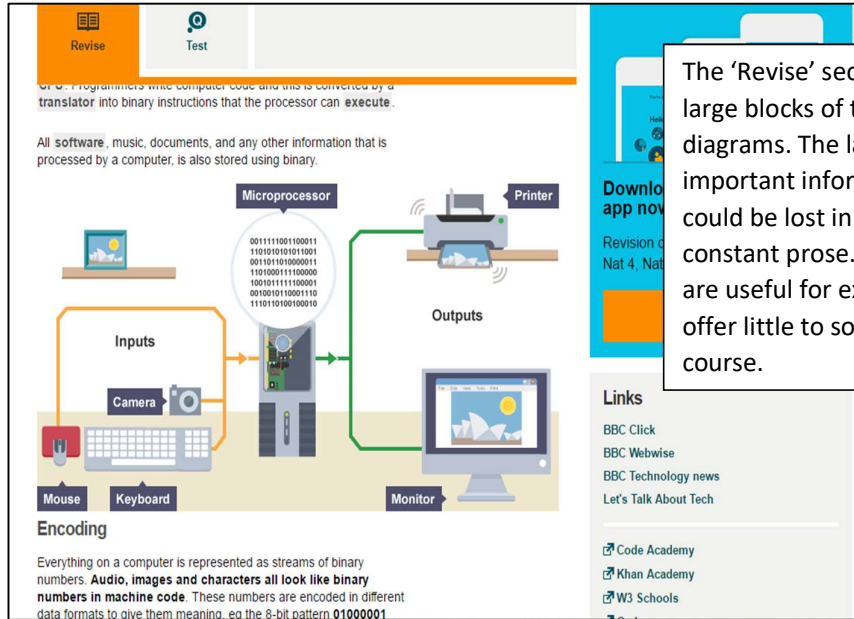
The quiz section of the book, however, only has a small amount of questions. Because the guide is paper based, the list of questions is static. This limits the usability of the Revision Guide as a way for students to test themselves.

The revision guide only targets GCSE students and only those on the OCR syllabus giving it a very narrow audience of people who could use it. Whilst it is comparatively cheaper than the revision guide, at around £6, this is still a large investment for a tool that has a very limited scope.

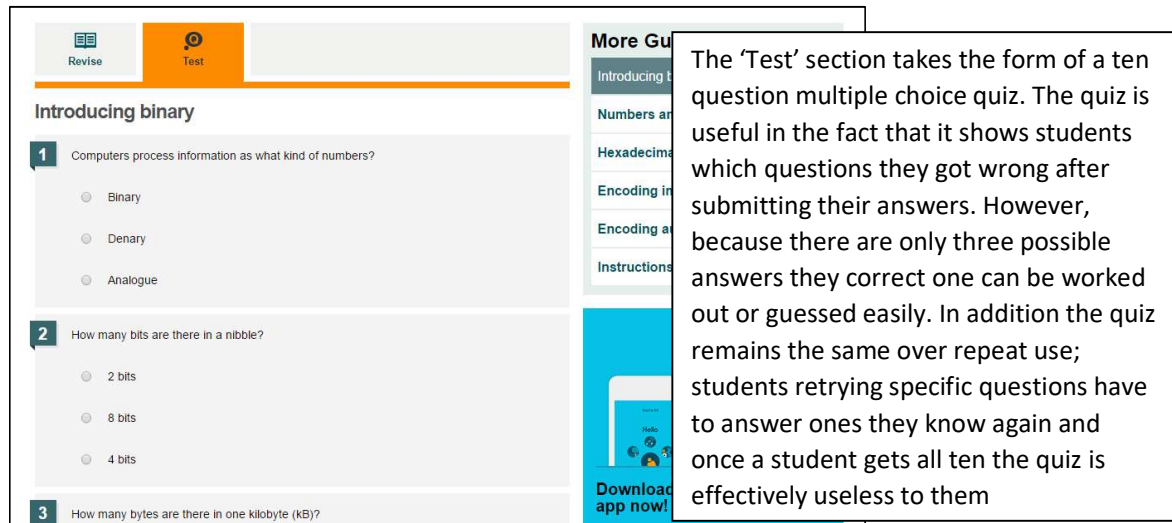
Computerised Revision Tools

As I am creating a computational solution I thought it would be prudent to investigate the computerised solutions the school currently uses.

- I first looked at BBC Bitesize. This is a popular revision website that offers both a textbook style revision guide section and quizzes.
 - I initial looked at the Revise Section on their website:



- I also looked at the Test section:



- The BBC Bitesize site and resources are free and come with the obvious immediacy of being online. However, there is no way to create an account or record scores. The system lacks any permanence or sense of progression so it is of limited usefulness to a student or teacher looking for a customised learning experience.
- I also looked at the online quiz platform for schools 'Kahoot'. This platform allows teacher to create and host quizzes for their students to take part in. This was important to look at as it has great similarities to what I want to create. Kahoot allows teacher to directly see how students

are performing against questions that they have created themselves and can change. Analysing where Kahoot succeeds and fails will allow me to improve my own application.

Which of these statements best describe a network?

58

IDEA

0 Answers

Skip

- A number of computers connected over a large geo-distance
- A number of computers connected over a small geo-distance
- A number of computers connected together
- A number of computers connected together wirelessly

The Kahoot game space presents the question and all answers on a screen, students must connect via another computer or their phone to answer. This is effective in ensuring students are engaged but means that the system cannot be used independently outside of school. The quiz uses multiple choice answers, I believe this is less effective as it doesn't reflect the type of questions students will be given in the exam. A conceit that comes from using a live interaction model is that a time limit is placed on answering questions. This could be useful in simulating a time controlled exam environment or it could pressure students into rushing and getting questions wrong. Throughout the game students are awarded points which are displayed on a leader-board. This encourages competition and I believe will help student engagement. However, points are given based on the speed a correct answer is entered, this encourages students to rush and may mean the test isn't reflective of a slow students true ability.

Kahoot is a free system and provides the tools to customise the experience. The system has accessibility, personalization and is engaging. However, Kahoot is limited by the nature in which the quizzes have to take place and the time sensitive nature of the quiz game.

Summary:

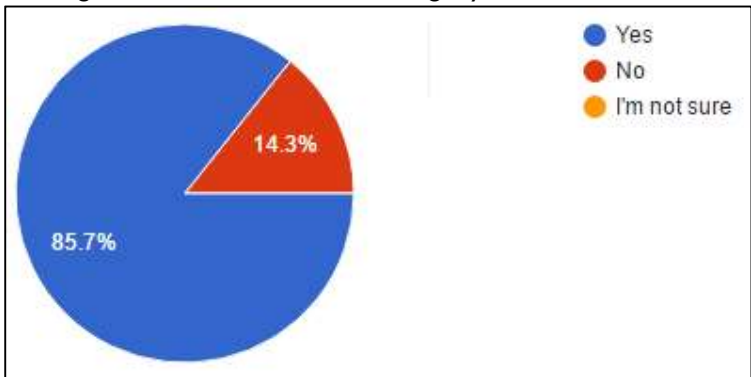
My research allowed me to explore various methods for Computing Revision currently available. By analysing where these methods succeed and fail I will be able to improve my proposed solution. I identified that student engagement was a key barrier to revision and the colourful designs of the Revision guide and Kahoot are evidence of this fact. To ensure my solution is successful I will aim to avoid the drab aesthetics of the computing textbook. A further problem that identified in current solutions is that they provide a small static list of questions. This was true for the text based solutions and BBC Bitesize; all three methods contained a small selection of questions and the Bitesize quiz failed to use its placement online to change the quiz on a second use. However, the Kahoot system implements some good ideas into its platform. I agree that the ability for teachers to set their own questions is valuable and that a scoring system helps foster competition and engagement in students. I intend to take on some of these ideas in my project by allowing staff to edit the question set and awarding student a score, but I intend to host questions of a database that can be accessed at all times so students aren't restricted by the teacher in when they can revise.

Questionnaire

After analysing current methods I want to gauge the opinions of my end users. I know what I think about revision and the currently available solutions but I need to see if the conclusions I have made are supported by Computing teachers and students. To try and get a large sample of opinions I have created two questionnaires, one for each end user, which I will use to collect data to inform my decisions.

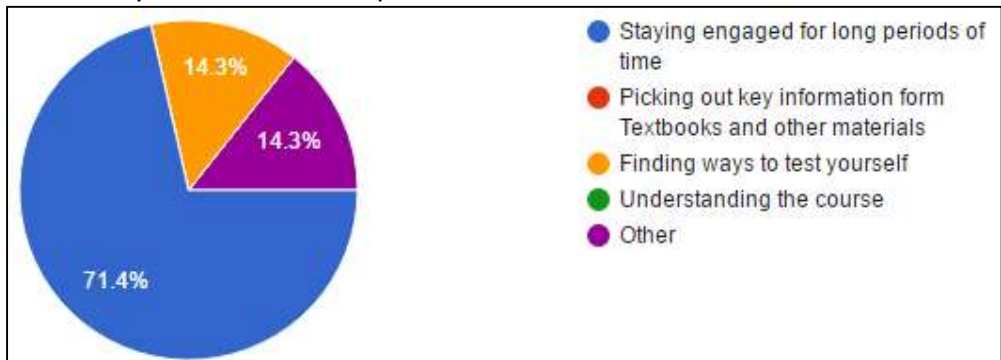
Student Questionnaire

1. Do you think a game based on Revision would get you to revise more?



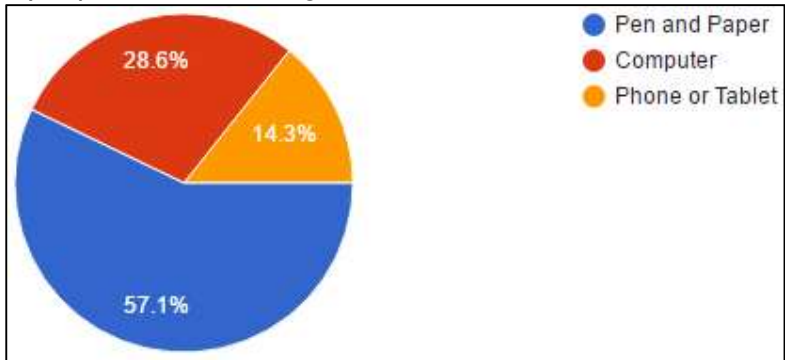
These results show that students clearly think that a tool such as the one I'm proposing would increase their revision activities. Based on these findings I believe the solution I have proposed is a suitable way to tackle the problem.

2. What do you think is the main problem with revision?



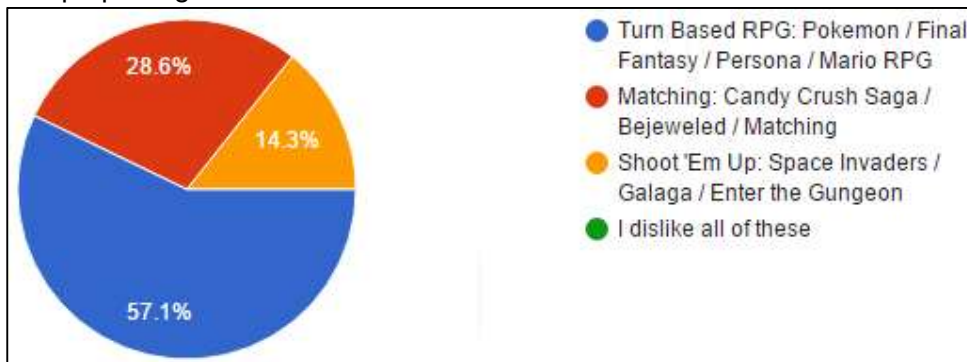
As I predicted, the largest barrier to revision for students is their engagement with the material. There is no shortage of textbooks and other revision materials, many of which are informative and can be used to test yourself, but these materials struggle to hold student attention. These results reinforce that a game would be an effective solution to the problem.

3. What do you prefer to revise using?



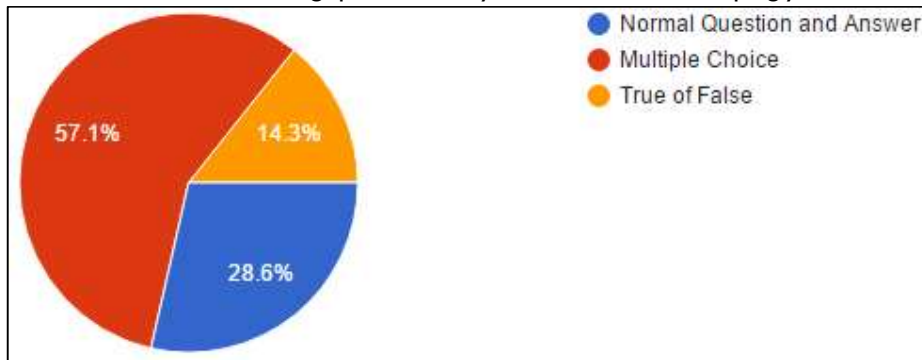
These results show that a pen and paper may be more popular with students. However, for the reasons discussed earlier I still believe that a computational solution is most appropriate; the use of a database gives my proposed solution durability that couldn't be achieved on paper. I would further suggest that the results take the form they do because students are most familiar with pen and paper materials and don't necessarily prefer them.

4. Which of the following video game genres would you prefer to make up the mechanics of the proposed game?

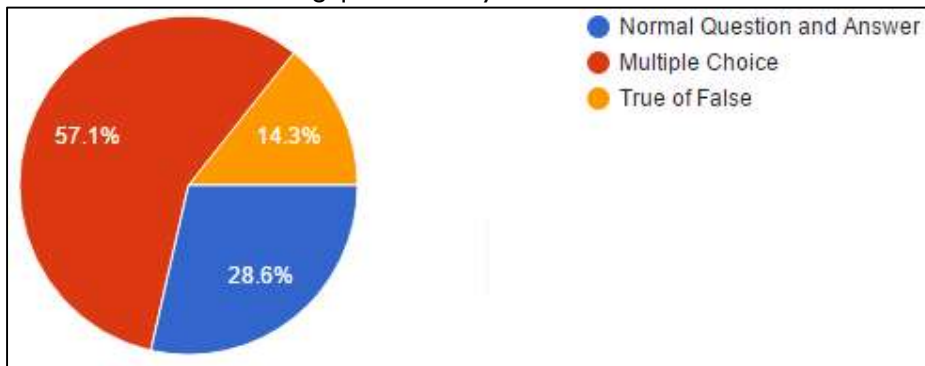


These results show a preference for turn based games in the style of Pokémon or final fantasy. I will take these games into account when I design my game's mechanics.

5. Which method of answering questions do you think is best at helping you learn the content?

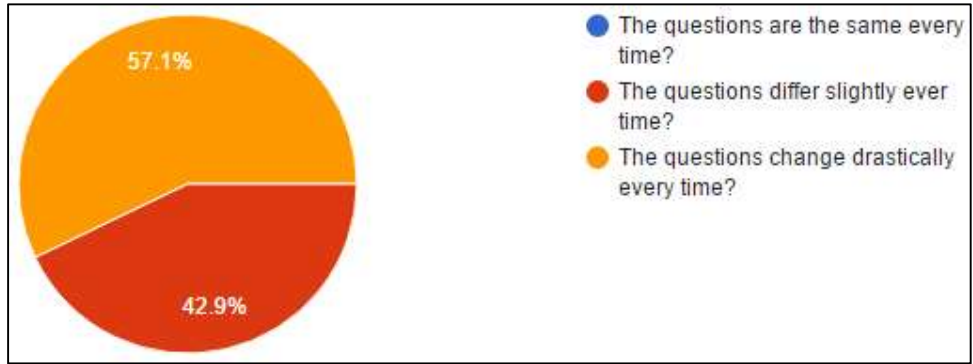


6. Which methods of answering questions do you think is easier to answer?



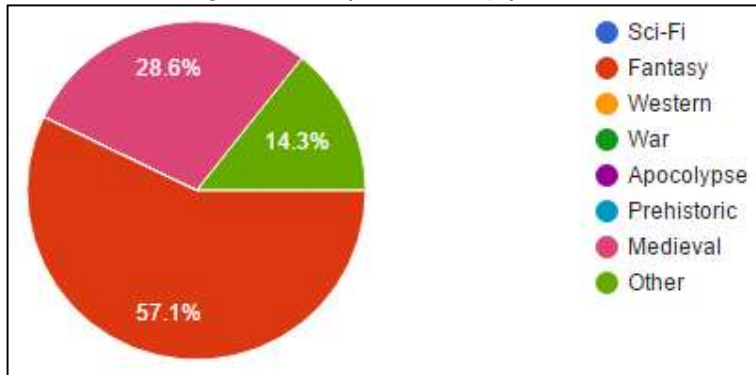
The results for which method of quizzing the students though were best and easiest were distributed in the same way. I therefore believe there may be a correlation between students saying multiple choice is the best and the fact that they find it the easiest. To be effective my system must have some level of difficulty so I intent to use a 'Normal Question and Answer' approach.

7. Which is more useful for revision, quizzes where the question list stays the same or quizzes where the question list is different on every use?



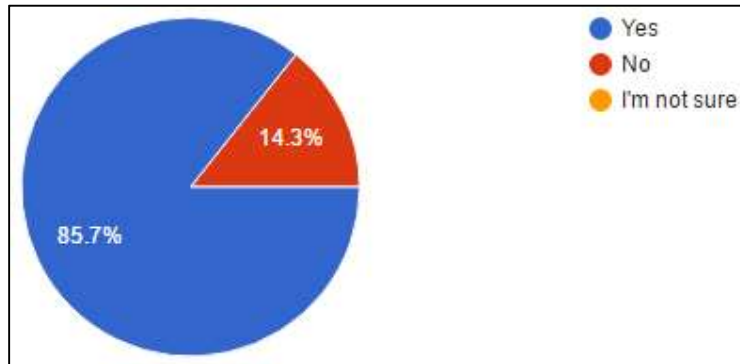
Not a single student said that they preferred static quizzes. This reinforces that my proposed solution, if it implemented a dynamic question set, would be a useful revision tool.

8. What kind of theme for a game would you most enjoy?



A Fantasy and Medieval theme is clearly the most popular option to design the game around. To maximise student engagement I intend to design the games art and assets within this genre.

9. Would you like to be able to compete with you peers by comparing some sort of score value?

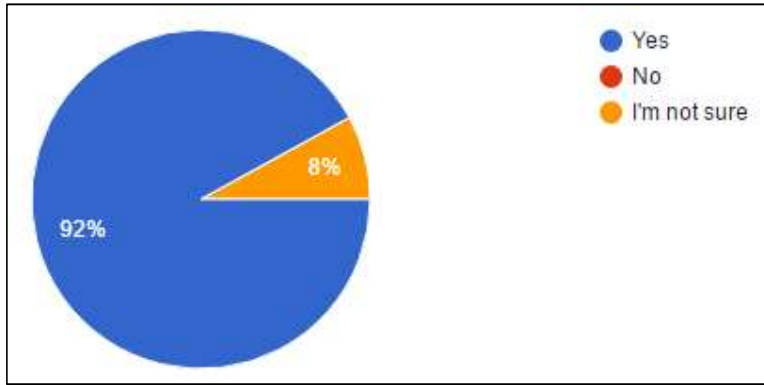


The results show clear demand for the proposed feature, in the interest of student engagement a score system should be implemented.

Staff Questionnaire

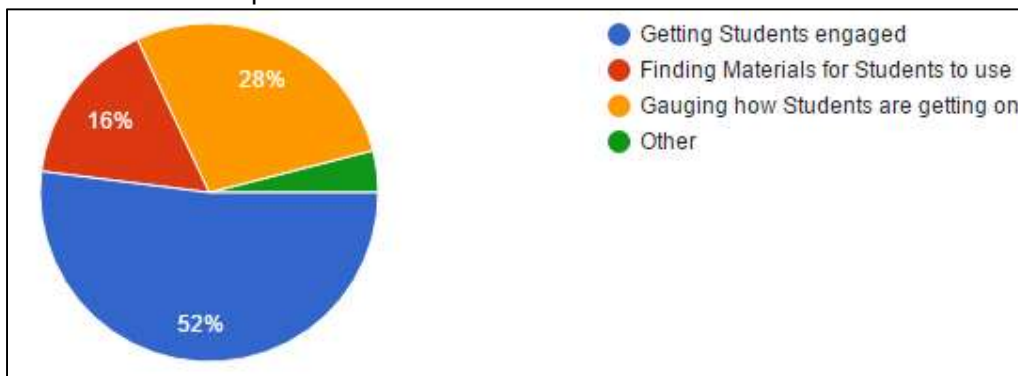
I asked Computing Teachers at from several different schools to complete a questionnaire to find the consensus on details for the Staff 'mode' of my solution. I collected 25 responses which are detailed below:

1. Do you think a game based on Revision for Computing would be useful?



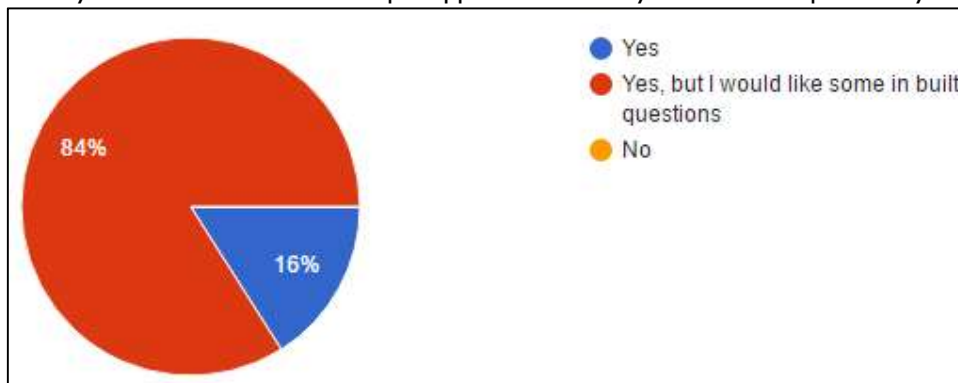
These results show there is a clear demand for my project. A vast majority think it will be useful and not a single person answered 'No.' Based on these findings I will continue working on my current project choice

2. What is the main problem with Student's Revision?



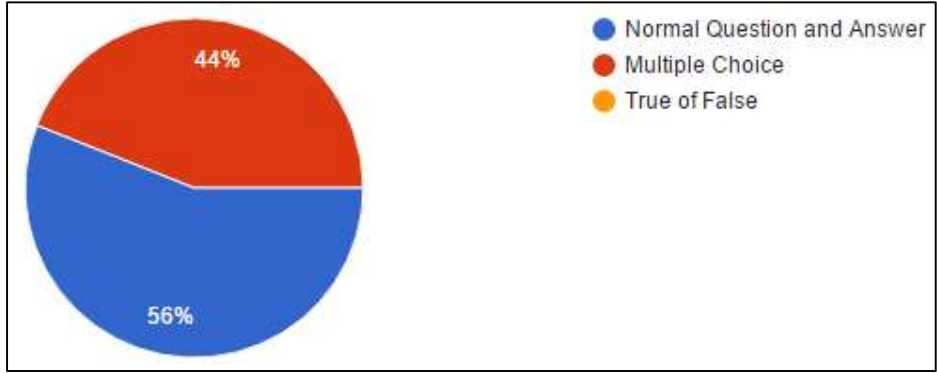
As I predicted in my analysis of the problem revision posed, the main problem is student engagement. As the results display, finding materials doesn't pose a problem for many teachers but these materials fail to get their students engaged. I am aiming to address this problem by making my project in the form of a game and will make sure I consider student enjoyment as I continue.

3. Would you find it useful to have a quiz application where you could add questions yourself?



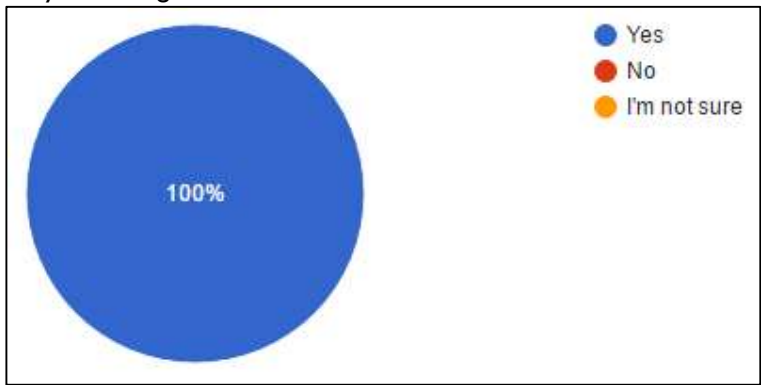
These results clearly show the ability to add questions is required for my project to be successful. They also show that there should be some in-built questions.

4. Which method of Answering Questions do you think is most effective in helping Students learn the content?



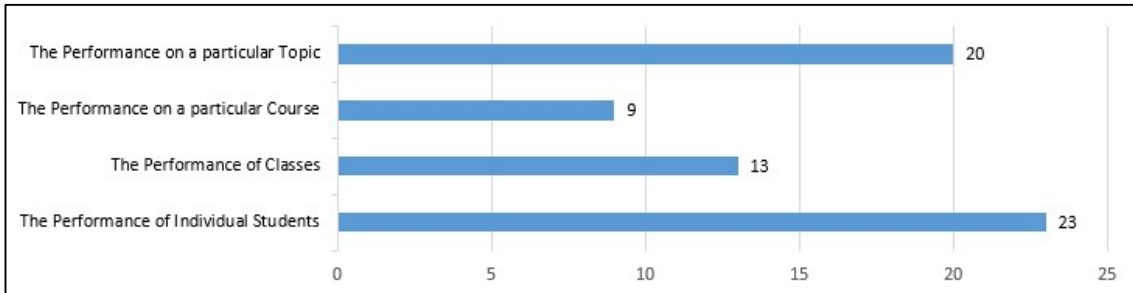
These results show that I should definitely not use a True or False system but there is no clear consensus on Multiple Choice against Normal Question and Answer.

5. Would you find it useful to be able to see the results of students; to track their progress and see how they are doing?



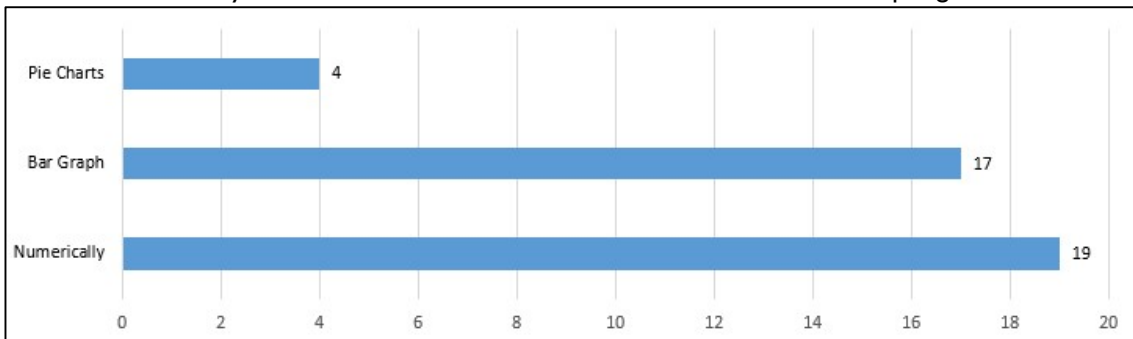
The ability to see students' progress is clearly necessary for my project to be successful.

6. Which information regarding student progress would you like to be displayed?



These results show that teachers would like to see information presented in a variety of ways but the two most important methods are by Student and by Topic.

7. How would you like to see information about students have done in a quiz game?



This data displays that it is important to represent the results numerically and graphically. The preferred form of graph is a bar chart.

Summary

The results from the student questionnaire largely agree with my suggestions. Student's answers show that engagement is the paramount barrier to revision and that there is demand for a revision tool in the form of a game. The questionnaire has shown the desire for a revision tool with a changing question set and implementing one should be a criteria for my systems success. Student's answers have informed me as to the mechanics and design of the game but I have chosen to ignore student preferences on the way questions are asked to maintain a useful level of difficulty.

The results from the staff questionnaire largely confirmed my thoughts; there was a clear demand for the solution I proposed. The results outlines that the ability to change questions and view students results would be useful and that their inclusion is necessary for my solution to succeed. The staff results have also informed me as to how student data should be arranged which will be useful later on in the project.

Interview

Because he is my primary stakeholder, I interviewed My Byford. This allowed me to get more in depth answers than what I could retrieve with a questionnaire. This was also essential as Mr Byford would have to sign off on my specification and my application would only

1. What impact does Revision have on student performance?
"When done properly, Revision has a positive effect on student's final result. High quality and high frequency Revision has been responsible for student's marks improving by 1 or 2 grades."
2. Are there any issues with Student's revision in the Computing Department?
"Absolutely, some students simply won't revise at home and this is very problematic. At the end of the course there simply isn't enough time to go over everything in depth so in school revision must be supplemented with work at home. Even when students are putting the effort in it's hard to tell whether what they're doing is effective."
3. What would you say is the main issue with Revision in the Computing Department?
"Motivation affects student across the spectrum of ability, it is by far the most significant problem with students. The resources available to students aren't 'fun' and subsequently lots of students aren't spending as much time as they need to for revision."
4. How do you currently test and track students' performance?
"When students take an exam the exam boards provide us with a complete statistical analysis, we can see how students performed by overall, by question and in relation to national averages. We receive numerical information, but also pre made bar charts. This data is amazingly helpful but we only receive this information once a formal exam has already been sat. We do our best to replicate this after mock exams but this isn't perfect; data has to be entered manually and we have to make our own graphs. In-between mock exams we're completely in the dark, we have no way to track Student's revision."
5. Do you think a quiz game based on Revision for Computing would be useful?
"I do, revising in a fun way would absolutely help with our motivation problems. A more gamified approach to revision would mean students could revise without thinking of it as an educational activity. I would be amazing if students approached revision in the same way they look at competing in video games."
6. Would you find it useful if you could track students Revision habits and test performance via a computerised system?

- “As we’ve discussed the more data we can get the better. Being able to track student’s revision at all would be useful and computerised system would be invaluable to the Computing Department.”
7. Would you find it useful to be able to edit and add to the list of questions in such a system?
“This would definitely make the system more usable. Being able to edit the question set allow us to prevent students getting bored with the tool. Furthermore, this would allow us to adapt to a change in the course specification which would greatly extend the lifespan of such a tool.”
 8. What information should be stored on students and what would be the legal implications of storing this?
“The only thing we need to know about students as individuals would be their name. As long as there isn’t any personal information stored you shouldn’t have to consider the Data Protection Act.”
 9. Can you explain how you would like details of student performance to be displayed: Numerically or Graphically? Arrange by class or by topic?
“It would be very helpful if the program you create could generate graphs in a similar way to the reports we get from exam boards. Results should be arranged by topic so we can best identify which topics to focus on.”
 10. What level of students do you think would make the most of this application: GCSE or A-Level?
“I think all students would be receptive to a tool like this, however, the group that we have the biggest problem with currently is the A-Level group. The A-Level qualification is more important to the students than the GCSE but many students struggle with the increased workload. I feel like a tool that would help students to revise vocabulary and facts would really ease the transition. The system would be more useful if it could be used by any year group but if it needs to be targeted I would recommend the A-Level group.”

Summary

My interview with Mr Byford further confirmed my thoughts towards the problem. Mr Byford was supportive of my idea and strongly advocated for the importance of engaging revision materials. The information Mr Byford provided about the way data from mock and real exams in used will be very useful in planning the way my system handles student data.

Justification for Project

The comments of my client display how pivotal revision is to the success of students during their GCSE and A-Level years. However, the comments of all the groups I worked with identified that there is a problem in the way revision takes place and a need for a revision tool that meets these issues. My investigation has reinforced that my attempts to create a revision system are justified. The most commonly identified issue is that students aren’t getting engaged with revision supports my decision to make my system in the form of a game. Furthermore, my investigation into existing materials have shown a distinct lack of ways for students to effectively test themselves; existing quizzes are static and unchanging and the answers given by staff and students shown that this is a barrier for them to be durable and useful. My investigation, therefore, displays that there is a need for a revision tool that has the adaptability of a changing question set.

Definition of Proposed Solution

After my research I felt confident to give a comprehensive definition of what I am aiming to create. In this section I will give an explanation of how I envisage my solution working and the outline a set of criteria I crucial for the solution to be successful.

Overview

A piece of software that will take the form of a revision tool for Computing Students. The software will have two 'modes,' one for students and one for teachers. The student mode will take the form of a game, students will progress through the game by answering questions taken from a larger pool of questions. The frequency a question appears will be proportional to the percentage of times it is answered correctly. The staff mode will allow staff to edit the questions in the pool, they will also be able to view the performance of students both numerically and graphically.

Success Criteria

Here I will outline the criteria my solution must meet to be deemed successful. I will compare my finished project against this criteria during my evaluation stage:

1. Criteria: My solution must ask students questions about Computer Science
Reason: My solution is a revision tool for Computer Science. I have decided to make this in the form of a quiz game which, by definition means questions must be asked. Both the student and staff questionnaires show a demand for such a tool and this solution was agreed with my client
2. Criteria: They Student facing part of my application must take the form of a game
Reason: A primary obstacle to revision that I identified was student engagement, gamifying the solution directly tackles that obstacle. Reception from my student questionnaire agrees that students would be more likely to revise with a revision game.
3. Criteria: My solution must come with enough pre-existing questions for the solution to be usable
Reason: My staff questionnaire identified that it was necessary the solution came with some 'built in' questions.
4. Criteria: My solution must allow teachers to edit, add and remove questions
Reason: A flaw observed in the current methods was that some of them catered to a specific specification. To make my solution adaptable to changes in the qualification the list of questions must be editable. The ability to change questions was also praised by my client and staff.
5. Criteria: My solution must record students' performance
Reason: I identifies that a lack of tracking was one of the main obstacles to effective revision. This was echoed in both my interview and staff questionnaire.
6. Criteria: My solution must store student' performance and questions in a central database.
Reason: Using a central database will allow changes to be made to instances of the application that are already given out, thus making the system a lot more user friendly. The amount of questions available in the currently available systems was consistently too low to allow for lots of repeated use. The central database would allow for a larger set of questions that that would be possible if questions were stored within each instance of the application.
7. Criteria: My solution must allow teacher to view students' performance both numerically and in a bar chart
Reason: My questionnaire identified that both data should be displayed both numerically and graphically. Bar charts were the most popular type of graph. In addition, the graphs provided by exam board, which Mr Byford praised for their usefulness, took the form of bar charts.

Final Specification

Here I outline my plan for the form my solution will take. The following points aren't definitive and I can deviate from them but I believe it will be useful to explain the proposed function of the system here.

Student Mode Function Specification:

- The student will control a character represented by a sprite. This character will have a health value, the game will continue as long as this value is greater than zero.
- The character will move around a map. This map will be divided into a grid and the sprite will move between grid locations.
- The character will take part in quizzes.
 - Questions will be chosen for quizzes from a larger database of questions. All questions in a particular quiz will be taken from the same topic. A question will appear more frequently if it is commonly answered incorrectly by students.
 - The quiz will be presented as a battle against an enemy character with a health value. When the student is wrong their character loses health; when they are correct the enemy loses health. The quiz will end when the enemy health value is zero or less.
- There will be items in the game:
 - Items will be positioned at random locations on the map where they can be obtained by visiting that grid location. Items will be stored in an inventory where they can be used or destroyed to free up space. There will be four different types of items:
 - Hats will be cosmetic items that can be equipped to the player. When equipped the characters sprite will change but they will have no function.
 - Tops will be cosmetic and functional items that can be equipped to the player. Tops will have a defence value that will reduce the damage done to the player by quiz enemies.
 - Weapons will be cosmetic and functional items that can be equipped to the player. Weapons will have a damage value that will increase the damage done to quiz enemies.
 - Health Items will be functional items that can be used by the player. When used they will restore some of the players health value.

Student Mode Design Specifications:

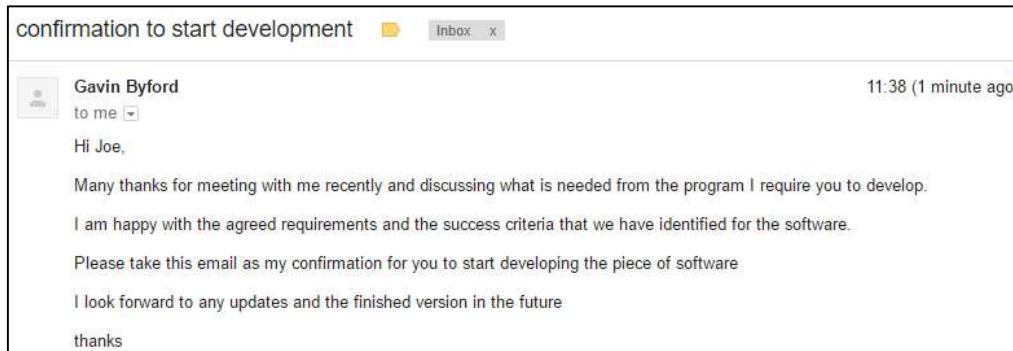
- The Student Mode will make use of bright colours
- The Student Mode will have a fantasy theme

Staff Mode Specifications:

- Teachers will be able to edit and amend to the questions database. When adding or editing questions staff will be presented with a data capture form that will check their entry against validation rules.
- Teachers will be able to view student performance:
 - Staff will be able to organise data by question, by student or by topic. They will be able to see the number of total answers and correct answers.
 - Information will also be presented graphically. The rate of correct answers in question, in students or in topics will be available presented as a bar chart.

Conformation

I presented this plan for the project to my end client. He was satisfied that the success criteria I had outlined accurately met his aims for the application and was satisfied with the specification I laid out for how I would achieve those criteria. My Byford, therefore, gave me consent to begin designing my solution.



Hardware and Software Specification

Here I outline the hardware and software I intend to use for both the development and the running of the application.

Hardware

- Development PC: In the interest of simplifying development I will do everything on my personal laptop. This will allow me to quickly move between editing the database and the game without moving between machines or resetting connections. When development is finished users will only have a packaged version of the quiz application.
- Server: I intend to use a computer connected to the school network to host my database. By using a central database the results from students using all instances of the application can be taken into account and changes made by staff can be easily distributed across all users.

Software

- Unity Game Engine: Unity is a piece of software that provides tools for creating games such as a rendering engine and a physics engine. I will use Unity to render and control the visual elements of my application.
- MonoDevelop: MonoDevelop is the Integrated Developer Environment supplied with Unity. I will use MonoDevelop to write the scripts to control the game engine. MonoDevelop and Unity works with both Java and C# but because there are more resources available in C# I will use that.
- MySQL Workbench: MySQL Workbench is an integrated developer environment for MySQL database design and maintenance. I will use this piece of software to design my database and host it for use by the application.
- Oracle Virtual Box: Virtual box is a piece of virtualisation software I intend to use of create a virtual machine from which I can host my database on the school network.

Hardware and Software Requirements

The decisions I have made about the tools I will use to develop and run my application will impose some hardware and software requirements on a user. The Unity documentation recommend Windows XP as the minimum software required to run their games. There are computers at my school that run Windows 7 or 10 but these should be easily able to run an application suitable for XP. Because my application will be dealing with large amounts of data from a database that requires processing by the application I believe it would be better to use the recommended hardware requirements for Windows XP instead of the minimum. The recommended hardware specification for XP are detailed below:

<https://unity3d.com/unity/system-requirements>

https://en.wikipedia.org/wiki/Windows_XP#System_requirements

- Pentium 300 MHz Processor or faster
- At least 128 MB of RAM

- At least 4GB of Hard Drive space available
- CD-ROM drive or compatible
- Super VGA (800 × 600) Monitor
- Sound card and Speakers or Headphones
- Keyboard and mouse or equivalent

These specifications will be easily met by every computer on the school network. Moreover, the specific computers in the computing classroom, where my application will get the majority of its use, are the most recently upgraded and are fitted with 16GB of RAM and SSDs.

Limitations

Unfortunately, circumstance means there are limitations placed on the development of this project. My solution cannot include every feature I would like it to in an ideal world. In this section I will describe the limitations placed on my project and how these manifest themselves.

Factors limiting the solution:

- **Time:** The project must be completed within one academic year and be submitted before the deadline. Two hours of lesson time are devoted to the project and the amount of time that can be devoted to it at home are restricted by other commitments.
- **Skill:** My skillset as a programmer and software developer are limited. I can and likely will have to learn some new things to create my application but there are some things I simply wouldn't be able to achieve.
- **Software and Hardware:** My tools are limited to that which I or the school possesses or can acquire for a reasonable price. Because of this some options for creating my solution aren't available to me.

Limitations of the proposed solution

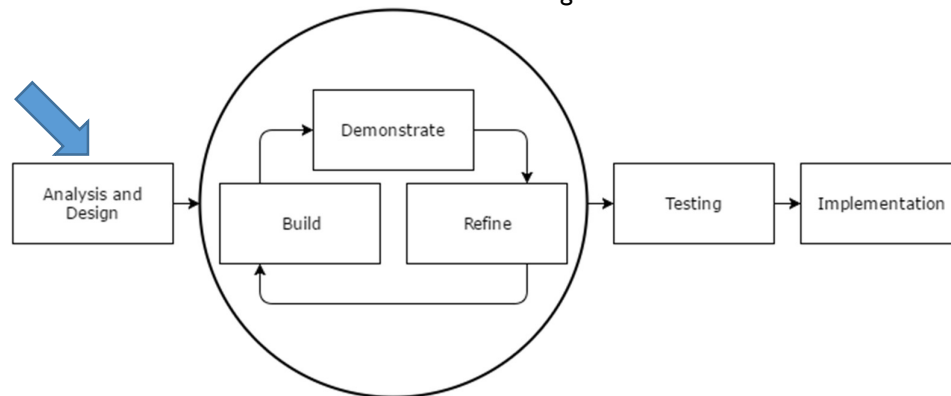
- **Simple Answers:** Because a computer has to decide if an answer is correct or not the questions will need to have simple answers. It would be impossible for the system to deal with whole sentences, but to achieve the immediacy and engagement I have aimed for answers must be handled by the application. This puts limits on the kind of question that can be answered.
- **Login:** Because no sensitive data is being stored as a part of my solution I haven't implemented a login system. Ideally this would be included but because the application doesn't legally require one I have decided to prioritise other features.
- **Printing:** My solution doesn't include any way for a teacher to print out results from the system. I have decided not to do this as the time required to implement printing can't be justified because it isn't central to the function of the solution. This limitation is acceptable as a staff member would be able to use a snipping tool or a similar program to print out the information anyway.
- **Central Database:** I have decided to store questions on a database which is an instance of the software must connect to. This is advantageous as it means a larger database of questions can be stored and older instances of the software can be kept up to date. However, this imposes the limitation that the application must be connected to the database to function. This is unfortunate but the best possible option in my opinion.
- **Game:** Because of limitations in time and software the 'game' aspect of the solution will be limited. I have decided not to include a narrative, sound effects of a game world that is more expansive than the quiz locations. I will try and make the game as fun and engaging as possible but the function of the application is prioritised over the form.
- **Art:** Because of my level of skill and the software I have available to me the artwork used to make up the game sprites will have to be simplistic. To try and mitigate this limitation I aim to use a pixelated art style.

Design of the Solution

In this section I will develop my solution to the problem. I will outline my methodology and then break down the problem into manageable tasks. First I will discuss the design of my solution with my client and develop the various GUI aspects. Then I will abstract my problem into separate task which I will create flowcharts for and subsequently write pseudocode.

Methodology

Throughout my project I will be using the Rapid Application Development methodology (RAD). The RAD model is a variation on the Waterfall model. Flaws of the waterfall model are that it takes a long time to create working code and that the project can easily deviate from the clients wished. The RAD model addresses these problems by including a prototyping loop in the project life cycle and emphasising client interaction with every successive iteration of the project. Working code is created early into development and new features are added gradually ensuring proper analysis and testing is done. I am fortunate in that I have constant access to my client so I am able to keep up with the demand for feedback. The model follows the diagram below:



I am currently in the analysis and design stage, in this stage I must analyse the problem and set out an outline of what my project will be. Only once I have a comprehensive specification will I be able to move on to the next stage of development.

Design Investigation

To ensure my design was as effective as possible I investigated software and video game UI to find out about established convention as well as what works and what doesn't.

Accessibility:

My software will likely be used by somebody who has some form of accessibility issues. I would be able to fully cater to ever disability but I should attempt to make my system as accessible as possible within reasonable limitations. I used the following articles to research UI accessibility and have outlined my findings below:

<https://developer.gnome.org/accessibility-devel-guide/stable/gad-ui-guidelines.html.en>

<https://medium.com/salesforce-ux/7-things-every-designer-needs-to-know-about-accessibility-64f105f0881b#.riq9gcwyx>

- Visual: It is important that I design my interfaces in a way that takes into account users with visibility issues. Such users may require images to be high contrast or high brightness to be able to read text and information properly. In addition, colour shouldn't be the sole way of differentiating things so that colour-blind users can still interpret the information. It would be very difficult to make the application accessible to blind users on their own. Implementing text to speech and dictation functionality is beyond the scope of what is achievable in the time frame. However, I should avoid using images as parts of questions so blind people can use the system with assistance.

- **Audio:** Some users might be hard of hearing or just have broken speakers. Knowing this I should avoid audio as the sole means to communicate information.
- **Movement:** Some users may have a disability that affects their movement or need use the game with a helper because of another disability. To make my game accessible to these people I shouldn't use time restrictions in my game to avoid alienating slower moving players.
- **Animation:** Fast flashing or jagged animations can cause agitation or seizures for users with epilepsy. If used, animation in my project it should be confined to small areas of the screen and flash in the recommended bounds of 2Hz – 55Hz.
- **Understanding:** My software should be accessible to those with mental and cognitive disabilities. One factor that makes UI far more understandable is having clearly defined areas to click. Some UIs opt for a minimalist design to reduce complexity but, in fact, click fields with clearly defined outlines help those with cognitive disabilities understand a UI. Furthermore, making use of hovering to reveal information can cause understanding issues. Hiding functionality can be confusing for all users so it should be avoided in my design.

Using this information I should be able to make decisions informed with how they will affect the ability of disabled people to use my solution. Hopefully this will lead to a system that is friendly to disabled users as much as is possible.

Perspective:

Initially games were 2D by necessity. As technology improved 3D games were made possible and are now the Industry standard. To investigate which perspective would be best for the student mode of my game I read the following articles and explored the subsequent perspective options:

http://www.gamasutra.com/view/news/112124/Analysis_The_Quandary_Of_2D_Vs_3D.php

<http://rampantgames.com/blog/?p=5934>

- **'Classic' Two Dimensional:** In the classic 2D perspective the user sees the game world from a top down or side on perspective. The world is flat and no attempt is made to give the illusion of a 3D space. This is an order of magnitude less complicated than creating a 3D game and will match the 2D design of the Staff mode. The consensus for 2D design is that it is two simple for AAA action games, but adds some novelty and nostalgia to smaller projects.



- **'Modern' and Isometric Two Dimensional:** More developed 2D games have used their improved graphical fidelity to simulate a 3D world while still using a fixed camera perspective. These are more complex to design than classic 2D but assets and textures still only need to be 2D. Again this form of design is a novelty suited to smaller projects.



- Full Three Dimensional: The most recent of the perspectives, 3D games render a fully 3D environment. This provides a more realistic experience and allows for more cinematic and complex environments but it is by far the most complicated to create. Assets must be three dimensions and maths operations must deal with depth components. 3D environments are also slow down processing and increase file size drastically.



I believe a 'Classic' Two Dimensional' perspective is best suited to the nature of my project. I am limited in manpower and time so it would be unwise to spend lots of time creating 3D assets that aren't essential to my projects success. Furthermore, my project is targeted at student and teachers, not hard-core gamers; I have therefore picked the most performant perspective tailor my design to the types of computers my end users are likely to have.

Diegesis Theory:

Diegesis theory is an approach to characterising gaming UI and is based on two primary factors; Is the component in the narrative? and Is the component in the Game World? This leads to four possible permutations each with its own benefits and drawbacks. Using the following articles I researched these four UI patterns and outlined them to advise my design:

<http://devmag.org.za/2011/02/02/video-game-user-interface-design-diegesis-theory/>

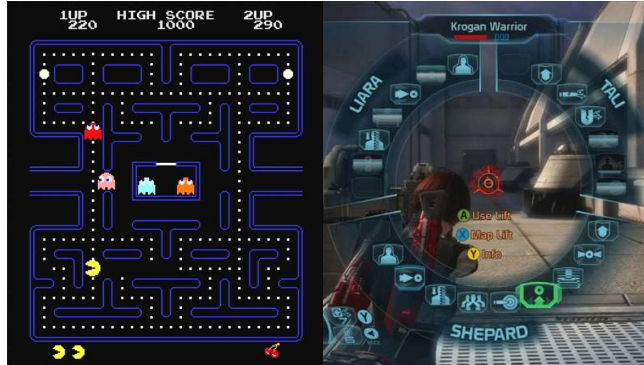
http://www.gamasutra.com/blogs/AnthonyStonehouse/20140227/211823/User_interface_design_in_video_games.php

http://www.gamasutra.com/view/feature/4286/game_ui_discoveries_what_players_php?print=1

- Diegetic Components: These components exist in both the game world and the narrative of the game. Examples of this would be if the charge of a weapon was displayed on the weapons hilt or the speedometer on a car's dashboard in a first person driving game. This type of UI design is effective in making the game more immersive on keeping the fourth wall intact. However, diegetic components can be less informative than other types and there is not much of a narrative in my game for the UI elements to fit in.



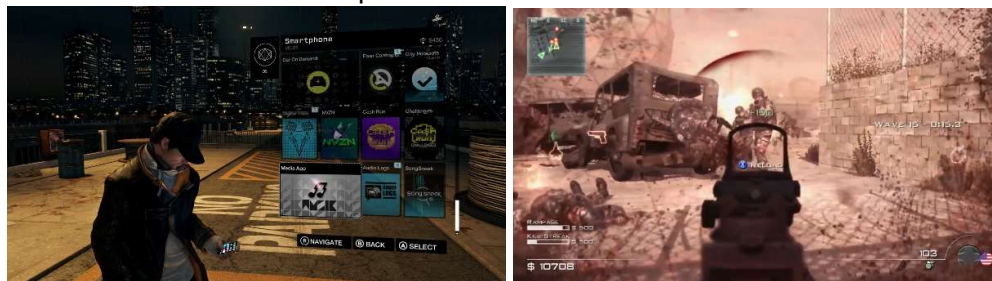
- **Non-Diegetic Components:** These components exist in neither the game space nor the narrative. Like the HUD in a traditional shooter game or a score counter in an arcade game these components provide information the character wouldn't know and present it on a 2D overlay above the game world. These components can break immersion but it is sometimes necessary to use non-diegetic components to present the user with information they need.



- **Spatial Components:** These components exist in the game world despite not fitting into the narrative. Uses of this UI design would be interaction cues on objects in the game or a colour aura around character with some special characteristic. These components break immersion over any other UI pattern as they present items in the game world which the characters are unaware of, exposing the gap between the narrative and the gamified simulation in the most blatant way.



- **Meta Components:** These components fit inside the game's narrative but exist on the 2D overlay above the game instead of in the game world. The most widespread example of this is the blood-splatter that covers the screen when the protagonist of most FPSs is hurt. Meta components are usually more legible than Diegetic elements but they are uniquely suited to First Person games. A blood splatter loses its narrative sensibleness when the player's view is not from the characters viewpoint.

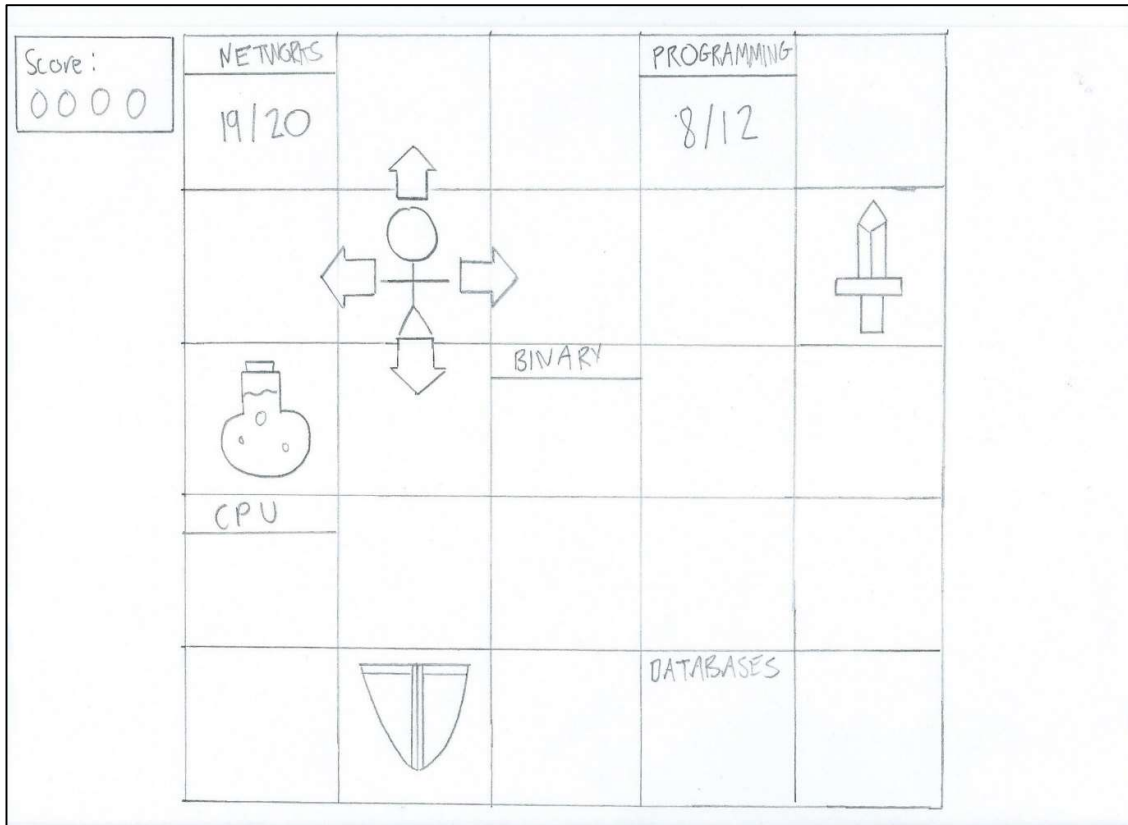


My game isn't a narrative, to base my design on the principal of getting the user immersed in the game world risks the project moving too far away from the client brief. The student mode must be a revision tool first and a game experience second. Based on this philosophy I believe I should use Non-Diegetic and Spatial components; these UI patterns were able to convey the most information to the user and are suited to a game that's in a 2D perspective.

Design Development

As part of my first interaction with my client I created a mock up design of what I thought the GUI of my application would look like. Using the conclusions from my investigation I decided to plan both the staff and student modes in a unified 'Classic' Two dimensional design. I decided to focus on the feel of the UI in this first iteration of design and spend time later ensuring my designs were accessible and followed Diegesis theory:

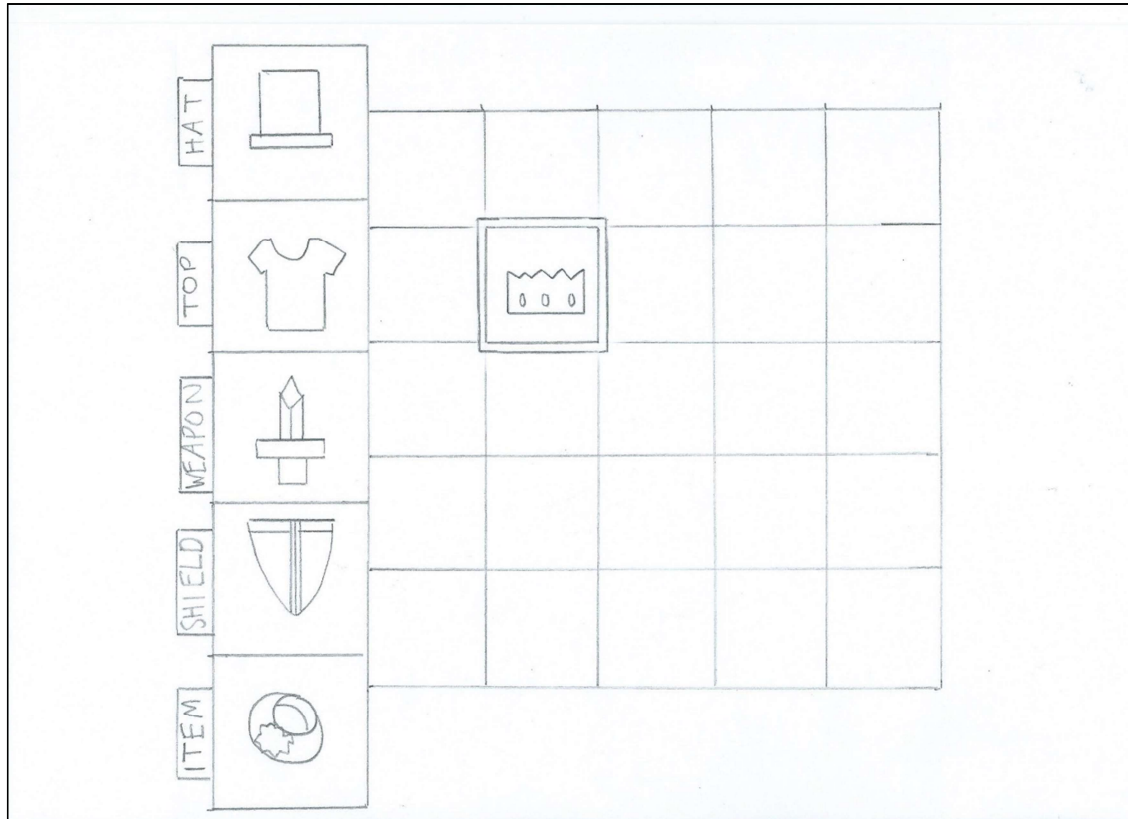
- I first showed my client an initial design for the overarching game map:



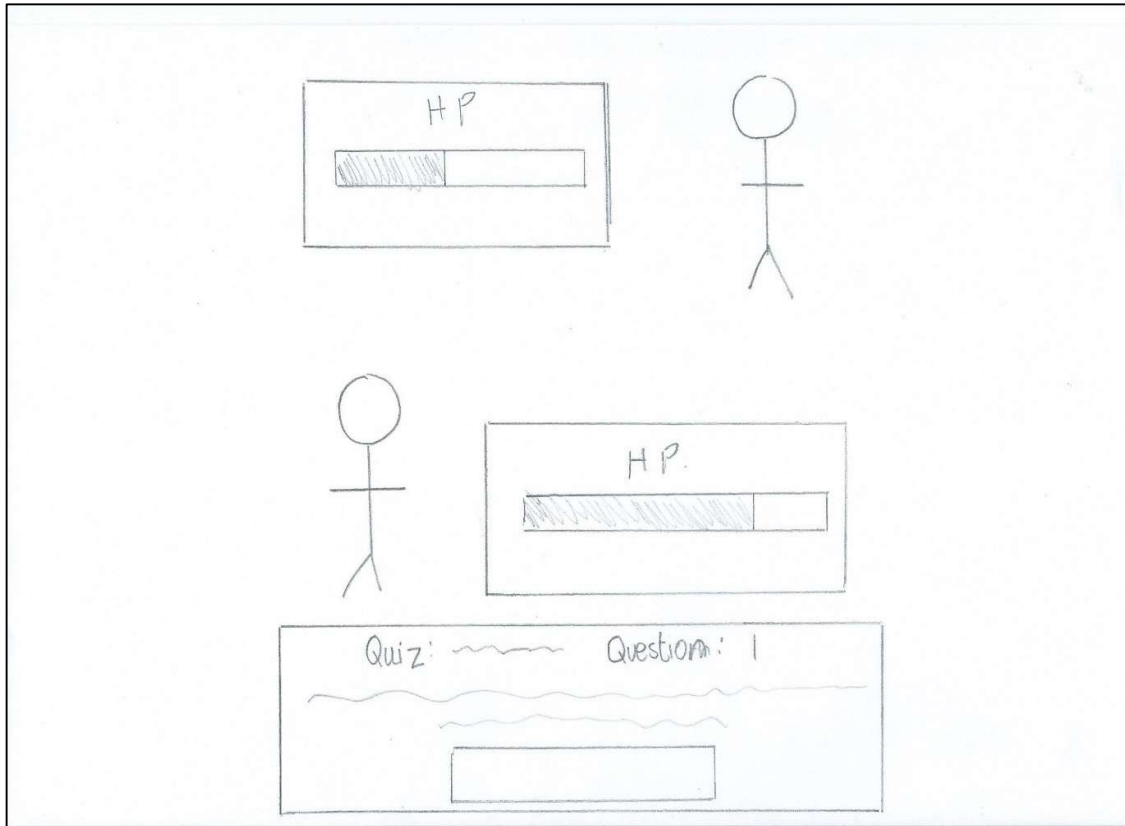
- We discussed the following issues to further my design:
 - Grid: I thought it would be useful to breakdown the game into a grid. This would allow for abstraction of the game easily when coding. My client liked this idea as he thought it would make navigation easier to understand.
 - Map: My client asked that the locations on the game 'Map' were landmarks and that the overarching navigation panel was an actual map. We discussed some ideas for landmarks that could be put on this map and I will create some designs to show the client in a subsequent meeting.
 - Colour: We decided that the game should use bright primary colour to make to game visually exciting and keep students engaged.
 - Items: The client liked the placement of collectable items on the map, he requested that the enemies also be displaced on the map
 - Statistics: My client requested that more information be displayed on the screen. He suggested that the user could see their average performance and some sort of indication of their progress. I will try and develop this further in subsequent designs.
 - Topic Headings: My client didn't like the idea of the topics for specific quizzes being displayed on the map. He informed me that, in his past experience, when students are given option of which topic they revise the often avoid the ones they are weaker on. We discussed this and came to the conclusion that the application would be

more effective if users couldn't see the topic and teachers were encouraged to emphasise a specific topic for revision by editing the questions list.

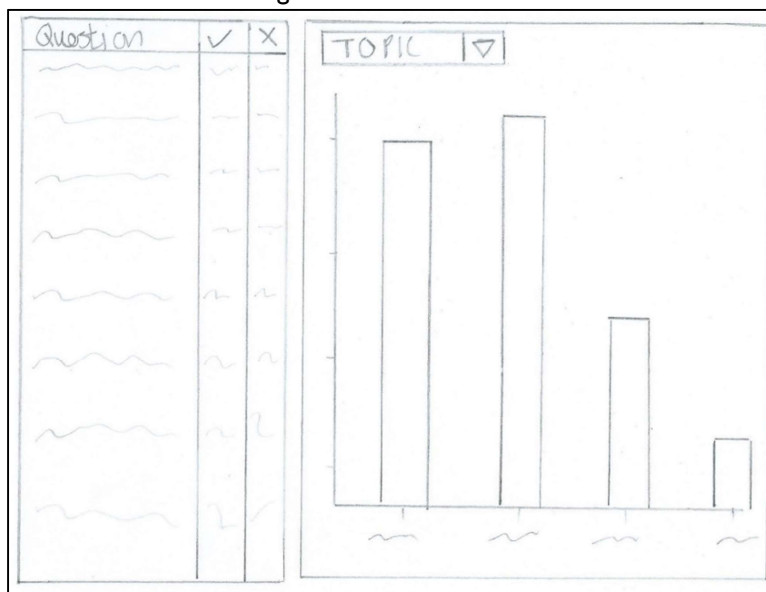
- I also showed my client an initial design for an inventory:



- There were some issues we discussed that didn't come up when looking at the map:
 - Items: The client thought there were too many different types of items and the current amount risked complicating the game and alienating some students. We decided to remove the Item / Accessory type and the Shield type leaving the Hats, Tops and Weapons. We believed this streamlines the items system and would make the game more fun.
 - Designs: My client found the two examples of hats I showed in my initial design, a top hat and a crown, funny. He suggested that continuing this comedic selection of items would make the game more fun.
- I also showed the client the design for the question and answer battle scene:



- This led to further points of discussion:
 - Validation: The inclusion of a text box led to a discussion about validation with my client. My client was worried that answers that were capitalised incorrectly would be considered incorrect. I assured the client I would use validation features to ensure that correct answers weren't considered wrong because of formatting.
 - HP: My client thought that the amount of health left should be represented numerically as well as graphically.
- I also showed the client the design for the results menu:



- This led to more discussion points:

- Question Table: My client thought that more details about a question should be displayed. We agreed that the table should display the question, the answer, the difficulty and the topic.
- Colour: I assured the client that the graph would make use of colour to distinguish the bars.
- Students: This original doesn't show any way compare different students or topics. I assured my client the final design will include the ability to compare students, topics and questions.

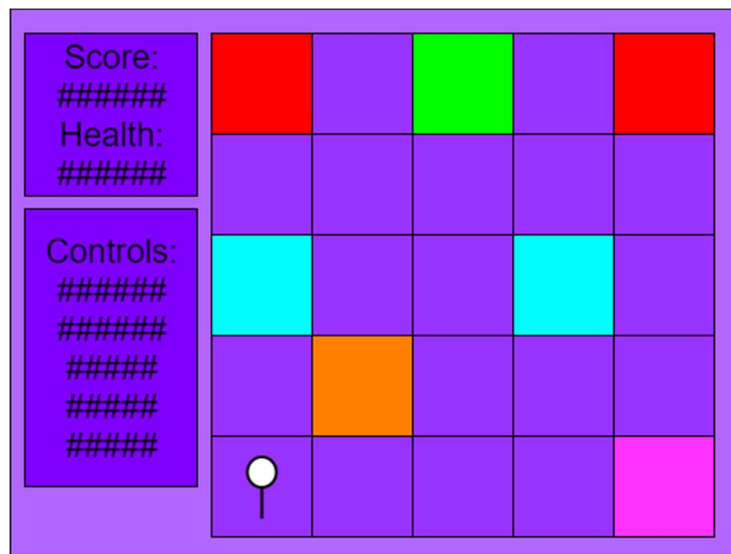
My client suggested that viewing results and editing the questions list were two separate task that wouldn't often be done together. My client asked that the final design of the screens and menus of the application incorporate a separate scene for editing, adding and deleting questions.

Final UI Design

Based on my discussions with my Client I was able to design a final version of what the User Interface in my application:

Final Map Design:

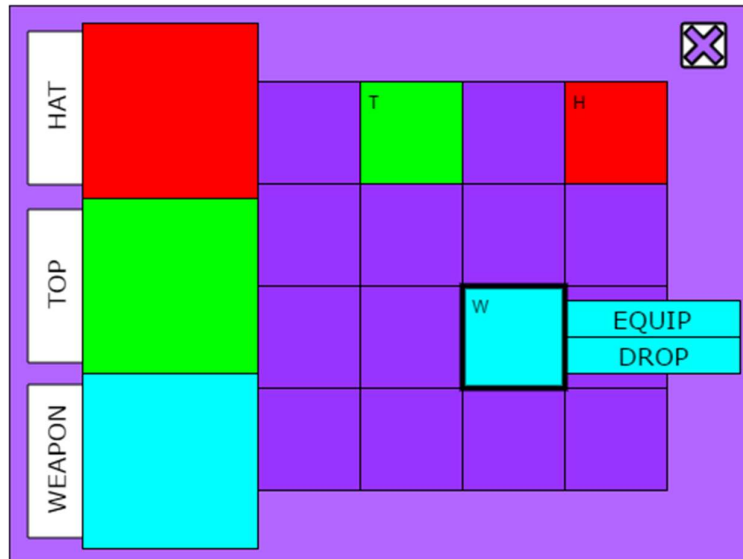
- A key point in my discussion with my client was the importance of randomising the quiz locations. In the interest of promoting this in my design, and improving the re-playability of the game I suggested that the game map be randomised every time the game played. I created a final design of what I wanted the map screen to look like as well as some examples of map landmarks, enemies and items that would be distributed around the map as well as the played.



- Bright colours are used throughout the design to promote user engagement and make the game more 'fun' to play. The use of bright colours also improved the visibility of the application and the consistent contrast between the bright designs and their black backgrounds improves the contrast to make the game more approachable to those who are colour blind.
- Both the user's total score and their health are displayed on the game map to give them an indication of their performance at every point.

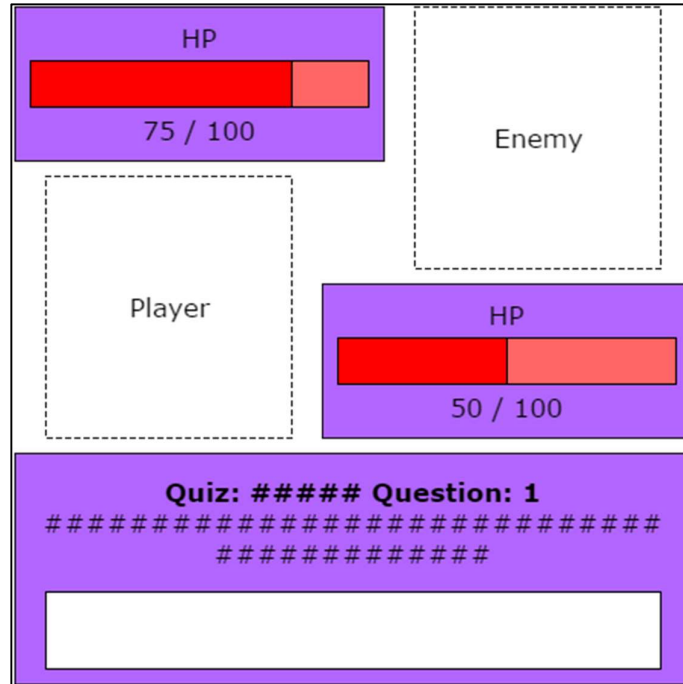
- The text displaying the topic of a specific quiz is removed. This means student cannot, as I was informed they would, avoid topics they feel they are weaker on. By forcing student to confront their subject weaknesses the application should be more effective.
- The controls of the game are displayed on the side of the screen. This should be an informative reminder during gameplay and prevent the need for teachers to give a demonstration before allowing their student to use the application.

Final Inventory Design:



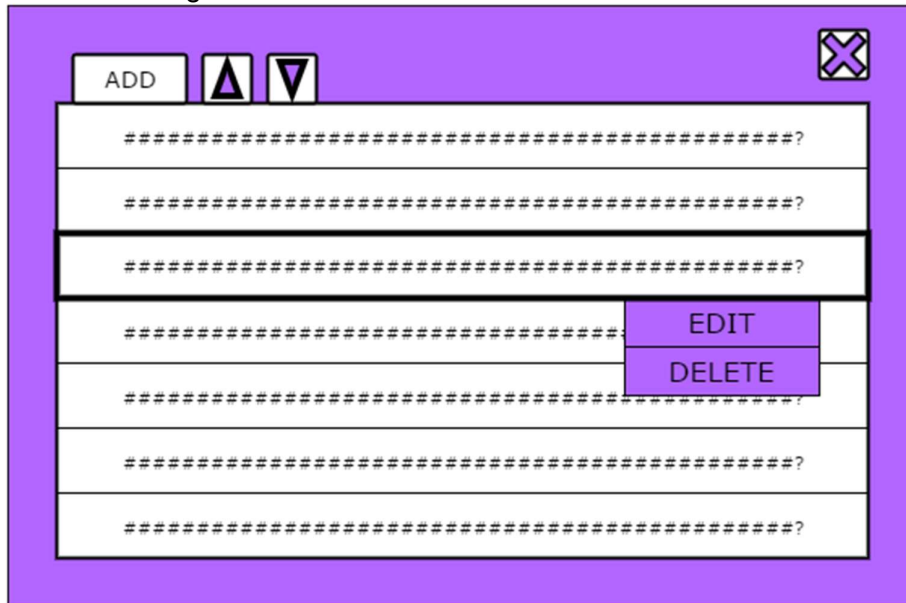
- Bright colours are used throughout the design with the aim of making the system look engaging. The use of bright colours also improves usability in brightly lit environments or by users with certain visual impairments. I initially intended to use a dark blue but changed it to the current cyan to improve contrast between the text and background.
- The equip-able item's system is streamlined into just Hat, Top and Weapon to simplify the screen and inventory system. The three colours are used to distinguish between the item types and make the design more minimalist but letters are included in the corner of any square with an item in it so that colour-blind users are still presented with this information.
- The inventory screen is separate from the main map screen. I believe this is better than a diegetic alternative as separating the two screen allows me to make both of them clearer and more informative.

Final Quiz 'Battle' Design:



- Health is displayed both graphically and numerically. This was point that was raised by my client, but the importance of displaying information in a textual way was also raised in my accessibility study.
- The dashed boxes will be filled with the player and enemy sprites in the finished game.
- The fact that the way this part of the application is inspired by the battle scenes of classic RPG's (A genre of video game) will, I believe, appeal to computing students and help to improve the overall engagement with the system.

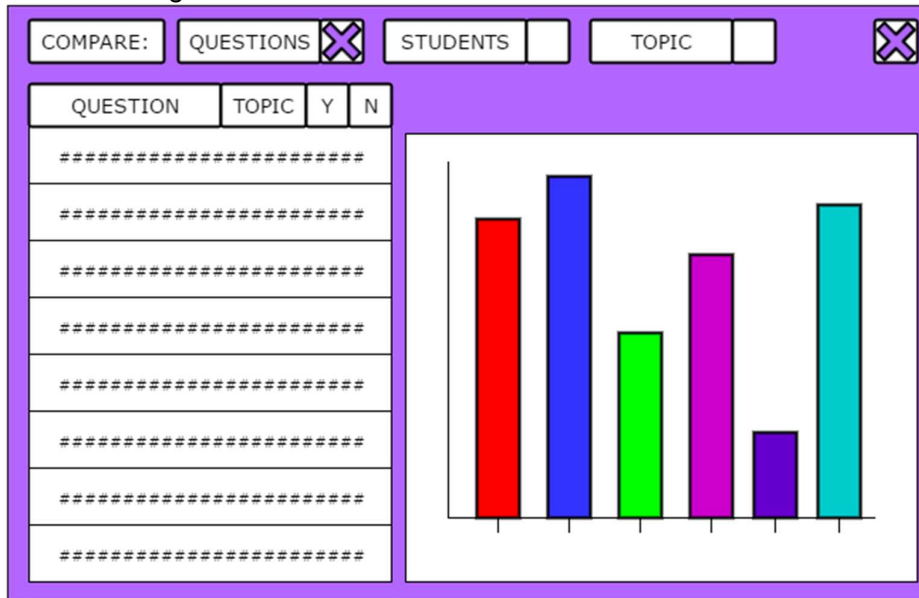
Final Question Menu Design:



- To keep text large, a small amount of questions are displayed and directional arrows are used to navigate the questions. I believed this was a better approach then attempting to display all the questions on the screen at one as this would limit the usability of the

application in its real life application because, when actually deployed, my system should be able to handle a large amount of questions that grows over the school year.

Final Results Menu Design:



- The line markers below the bars on the bar chart will be replaced with details of what the bar represents in the actual application. It is important that this information be displayed textually so the application is accessible to people with colour vision issues and to allow for black and white printouts of the charts.
- The checkboxes allow the user to switch how information is compared, this will allow the system to accommodate the full variety of filter settings my client and example end users have demonstrated a desire for.
- The use of colour in the bar chart, although it may be a fortunate side effect, is not to improve the applications aesthetics. Instead, using different colours will be useful to differentiate the different bars is important to make

I showed these final design to my client. He was happy with the aesthetics and functionality and satisfied that all the issues we discussed in the first stage of development had been accurately rectified. He therefore gave me consent to start working on the actual application.

Proposed Record, Store and File Structure

As part of my project, a lot of data will be stored. Users need to be able to access and in the case of teachers, change information on questions and items in the game. I recommend to my client that I use databases to accomplish this. Databases provide a persistent store of organised data and allow data to be interrogated easily, this should make retrieving questions and items from a specific subset easy. Databases also enforce restrictions on new data added, because teachers will be able to add questions it will be useful to be able to employ validation tools on their input. After explaining the benefits of using a database to my end client he agreed that this was the best solution and gave me permission to host the database on an unused PC connected to the Network

Database Investigation

To ensure that my database stored data in the most efficient and effective way I undertook and investigation into standards for database creation.

- Normalisation: Database Normalisation is the process of taking a flat-file, unorganized database and making into a more efficient relational database. I am going to aim to use third

normal form, this stage of normalisation requires every distinct entity in the database has its own table.

- I have decided to use 3NF because it is the highest form that I'm familiar with and offers a satisfactory level of efficiency. By giving every entity its own table 3NF is a logical and straightforward approach to database design. Furthermore, this level of normalisation reduces the possibility to data being duplicated to almost null.
- Naming Standards: Naming standards ensure the database is easy to understand, effective names are informative and in an expected standard. By looking at the following links I have created a set of standards I will use when creating my database.
 - <http://www.vertabelo.com/blog/technical-articles/naming-conventions-in-database-modeling>
 - <http://www.toadworld.com/platforms/mysql/w/wiki/6103.naming-conventions>
 - I will use the camel case method for typing and spacing names.
 - Primary keys will take the form of the table name followed by 'ID'
 - Foreign keys will take the same form as the Primary Key of their parent table but with an underscore, '_', at the start.
 - I will use singular forms of nouns used in my database. 'student' instead of 'students'.
 - I will aim to keep names short in the interest of readability.
- One database: Initially I thought it would be a good idea to use separate databases to store information on students, questions and items. However, after researching database organisation and software I have found that it would be more logical to store everything together. Using multiple databases would vastly complicate programming and make interpretation data properly impossible because of data dependencies between the different databases.

Database Design: Data Dictionary

Following my database investigation I thought it would be prudent to outline all the data my system needs to store and the features of this data. I have broken down data by entity to promote a normalised approach when I come to create the database.

Student Table:

Data	Explanation	Data Details	Example
First Name	This is needed to identify students	Data must be stored as a String Data must be in the range 1-45 characters (inclusive)	"Joe"
Last Name	This is needed to identify students	Data must be stored as a String Data must be in the range 1-45 characters (inclusive)	"Rackham"
Class	This is needed so that a teacher can view students only from a class they teach	Data must be stored as a String Data must be in the range 1-45 characters	"13D"

Class Table:

Data	Explanation	Data Details	Example
Class Name	This is needed to identify classes	Data must be stored as a String Data must be in the range 1-45 characters (inclusive)	"13D"
Exam	This is needed so that it is possible to view all the students taking a specific exam	Data must be stored as a String Data must be either 'GCSE' or 'A-Level'	"GCSE"

Exam Table:

Data	Explanation	Data Details	Example
Exam Name	This is needed to identify different exams	Data must be stored as a String Data must be either 'GCSE' or 'A-Level'	"GCSE"

Question Table:

Data	Explanation	Data Details	Example
Question Text	This is the text of the question, it is needed for the game to work	Data must be stored in a String Data must be in the range 1-100 characters (inclusive)	"What does RAM stand for?"
Answer	This is needed so an answer can be determined correct or incorrect	Data must be stored in a String Data must be in the range 1-100 characters (inclusive)	"Random Access Memory"
Topic	This is needed to arrange questions for different quizzes	Data must be stored in a String Data must be in the range 1-45 characters (inclusive)	"Binary"

Topic Table

Data	Explanation	Data Details	Example
Topic Name	Needed to associate different questions together for	Data must be stored in a String Data must be in the range 1-45 characters (inclusive)	"Binary"
Exam	Needed to determine if a topic is suitable for a student	Data must be stored as a String Data must be either 'GCSE' or 'A-Level'	"GCSE"

Answer Instance Table:

Data	Explanation	Data Details	Example
Correct	Stores if an answer to a question is correct or not	Data must be stored in a Boolean value	TRUE
Date	This is needed so a teacher can see the frequency of use and progress with a student	Data must be stored in a date time value	12:10:56 11/12/16
Student	This is needed so specific answers can be attributed to specific students	Data must be stored in a String Data must be in the range 1-45 characters (inclusive)	"Joe Rackham"
Question	This is needed to determine which question this is the answer to	Data must be stored in a String Data must be in the range 1-45 characters (inclusive)	"What does RAM stand for?"

Item Table:

Data	Explanation	Data Details	Example
Item Name	Needed to identify different items	Data must be stored in a String Data must be in the range 1-45 characters (inclusive)	"Crown"
Effect	Needed to make different items have different strengths	Data must be stored as an Integer Data must be in the range 1-10	"7"
Type	Needed to determine how the item affects the game state	Data must be stored in a String	"Hat"

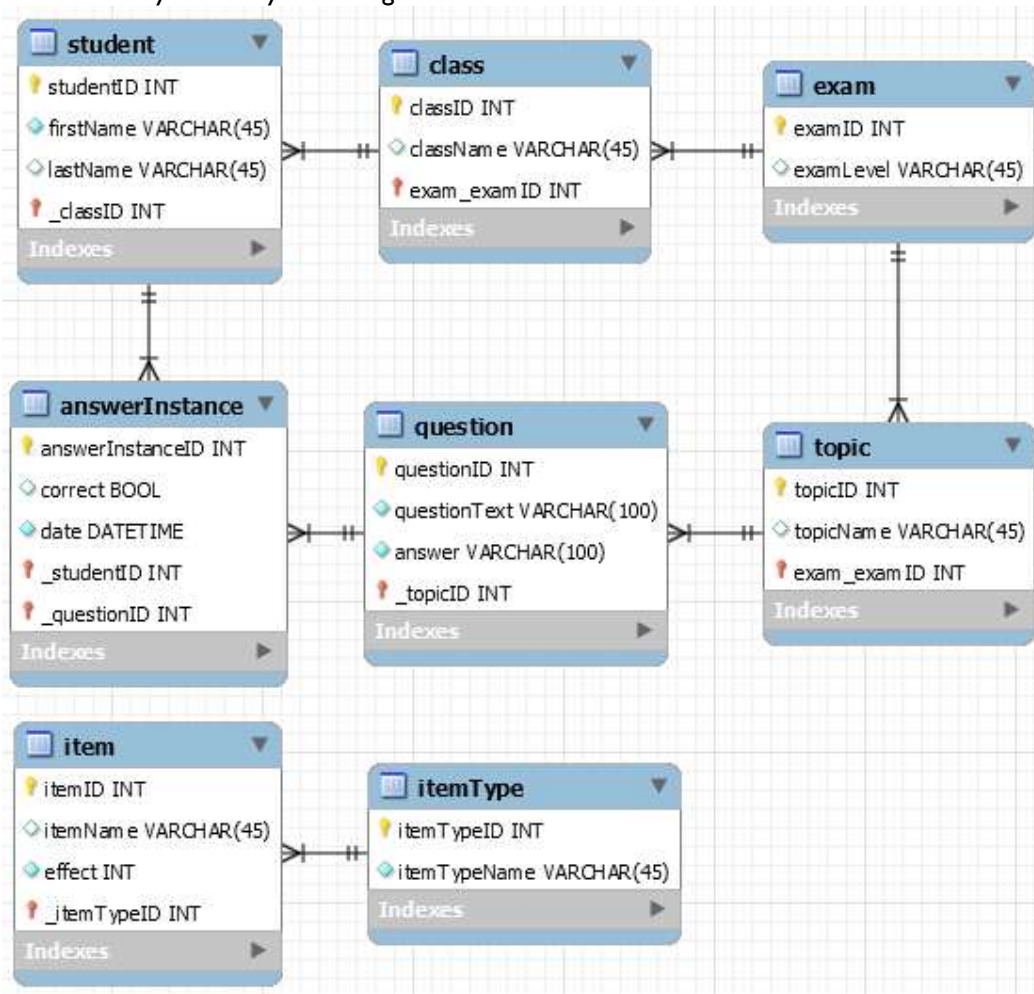
		Data must be in the range 1-45 characters (inclusive)	
--	--	---	--

Item Type Table:

Data	Explanation	Data Details	Example
Type Name	Needed to distinguish different items and the effect they have	Data must be stored in a String Data must be in the range 1-45 characters (inclusive)	“Hat”

Database Design: EER Model

Following creating my data dictionary I laid out the tables and entity relationships in the MySQL workbench model builder. By setting up the relationships between the different tables it became evident what fields needed to be users as primary and foreign keys. In the diagram below the fields at the top of each table and marked by a yellow key are primary and those at the bottom of the table and marked by a red key are foreign.



Database Design: Specification Investigation

After designing the database the need was presented to investigate the different topics that would make up a GCSE and A-Level Computing syllabus. This differs between different examination boards so I have researched several different boards and will compile the information into generic topic headings.

- <http://ocr.org.uk/Images/225975-specification-accredited-gcse-computer-science-j276.pdf>
- <http://ocr.org.uk/Images/170844-specification-accredited-a-level-gce-computer-science-h446.pdf>
- <http://filestore.aqa.org.uk/resources/computing/specifications/AQA-8520-SP-2016.PDF>
- <http://filestore.aqa.org.uk/resources/computing/specifications/AQA-7516-7517-SP-2015.PDF>
- <http://qualifications.pearson.com/content/dam/pdf/GCSE/Computer%20Science/2016/Specification%20and%20sample%20assessments/computer-science-spec-updated.pdf>

GCSE	A-Level
System's Architecture	System's Architecture
Exchanging Data (Networks, Databases etc.)	Exchanging Data (Networks, Databases etc.)
Software	Software
Ethical Cultural and Legal concerns	Ethical Cultural and Legal Concerns
Data Representation	Data Representation
Programming	Programming

I found that the topics on the GCSE and A-Level syllabus largely mimic each other, this reinforces the importance of using primary keys in my database. Questions in this system will be divided by these topics to be the focus of different quizzes.

Database Design: Data Inserts

The final part of my database design was to create the inserts. Inserts are pieces of data you can add in the design stage that will be present in the database when it is created. This is useful for implementing the basic data that will always be in the database, such as topics and exams, from the start.

itemTypeID	itemName
NULL	Hat
NULL	Top
NULL	Weapon
NULL	Usable

topicID	topicName	_examID
NULL	System's Architecture	1
NULL	Exchanging Data	1
NULL	Software	1
NULL	Ethical, Cultural and Legal Concerns	1
NULL	Data Representation	1
NULL	Programming	1
NULL	System's Architecture	2
NULL	Exchanging Data	2

Programming Investigation

To ensure that I programmed my application in an effective way I undertook an investigation into programming practices.

- Variable Naming Convention: Multiple conventions exist for the naming of variables. I decided to research what factors are considered to make a variable more useful:
[https://en.wikipedia.org/wiki/Naming_convention_\(programming\)](https://en.wikipedia.org/wiki/Naming_convention_(programming))
<http://www.makinggoodsoftware.com/2009/05/04/71-tips-for-naming-variables/>
 - Informative Naming: The research I found emphasised that variable names should be as informative as possible. Variable names such as 'n', 'value' or 'input' don't convey any information in most contexts and can make understanding the code if it is revisited for testing or debugging particularly difficult.
 - Length: Variable names need to be a suitable length to balance usefulness with the readability of the questions. Short or one character variables like 'in' or 'i' don't convey very much information and make it difficult to use a search tool to find instances of a variable.
 - Pattern: All the documents on variable naming emphasise that a consistent pattern is used. I have decided to use the 'Camel Case' convention as it is the one I'm most familiar with.

Variable naming is integral to the understand-ability of code. By using informative names of a reasonable length, all in unified pattern I will ensure that my code maintains a level of readability and will hopefully prevent problems during debugging and testing.

- Object Orientated Programming: Object Orientated Programming is a programming paradigm based on creating classes that represent real life objects. I already had an understanding of OOP from the A-Level course but I researched OOP further using the following links to gain secondary opinions on why it is effective programming practice:
<http://searchsoa.techtarget.com/definition/object-oriented-programming>
http://www.webopedia.com/TERM/O/object_oriented_programming_OOP.html
 - OOP presents a clearly defined modular structure for programming. By orientating thinking around real world objects, OOP is very understandable and easy to plan for.
 - OOP promotes the 'encapsulation' of data. Attributes of objects can only be accessed via the methods of that object and therefore it is not possible to program any incorrect changes. OOP therefore promotes data integrity.
 - Classes can 'inherit' from each other, taking on all the attributes and methods of a parent class and adding to them. This prevents the repetition of code and further contributes to the modular structure OOP naturally lends itself too.

OOP is suited to my application as Items, Students, Quizzes and Questions are all real world objects. The problem I am aiming to solve has a lot of tangible element which lend themselves to the OOP structure. My knowledge and research have displayed to me that OOP helps provide a clear plan for the structure of algorithms and results I a concise and modular end program. Based on this I will aim to incorporate OOP in my project.

- Version Control: Many professional software development firms use a version control system to simplify the process of iterating on software. These systems allow for a 'branch' to be created where the code can be modified before it is merged back into the main program. Changes are strictly recorded so a history the evolution of the project can be used. This is incredibly useful as it allows to code to be changed without the risk of breaking the main application. The change log can also be used to understand how a broken piece of code deviates from a functional version. Although I am only one person and not creating my project in a professional context I believe it would still be useful to use a version control system. The RAD Methodology I have chosen requires iteratively adding to a working piece

of code. This methodology lends itself to a version control system as changes made to the program mustn't undo the deliverables the application has already met.

<https://www.git-tower.com/learn/git/ebook/en/command-line/basics/why-use-version-control>
<https://boagworld.com/dev/to-version-control-or-not/>

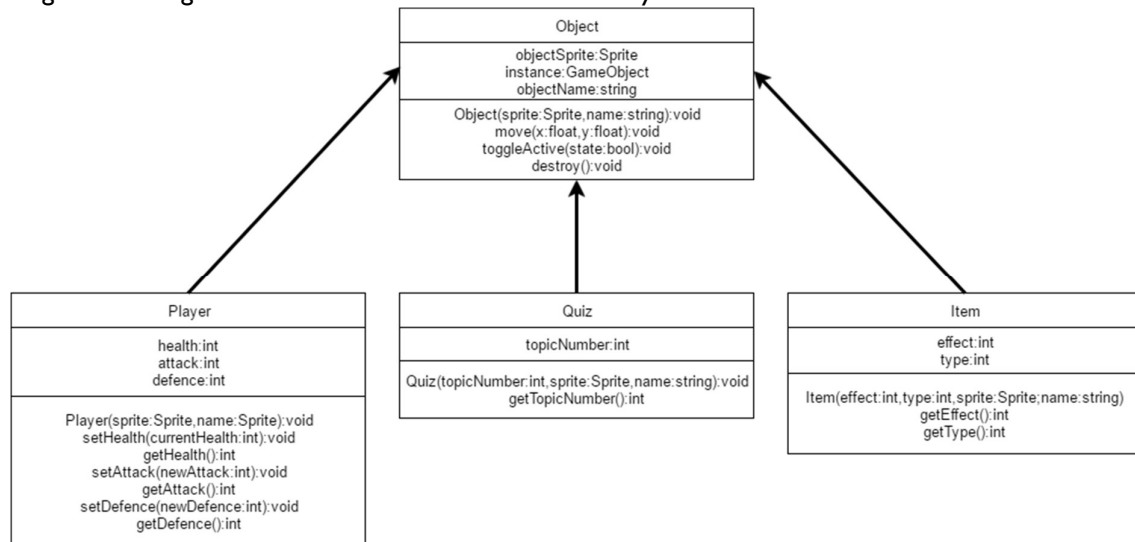
- Libraries: Libraries are complete and pre-tested code written by somebody else to perform a relatively commonplace or ubiquitous task. The use of libraries is generally encouraged as a good thing because it is good way to prevent errors and reduce the time taken to complete a project. The general opinion is that there is little value in writing code to perform a task when there is already a more fully developed and tested library to do the same job. Moreover, libraries can allow programmers to be able to interface their programs with different hardware and software without needing to know how to program for them. I fully intend to use libraries in the development of my project. Indeed, Unity comes with a standard library to interact with its game engine; attempting to build my project without using this library would be nigh on impossible to achieve with my limited experience and time. The situation with connecting to my database is similar; developing the necessary code to communicate with the database without libraries would be a project unto itself. I believe by using libraries I will massively increase the scope of what my application can do and allow me to devote a great deal more of my time to achieving mu project brief.

<https://www.quora.com/Am-I-considered-a-bad-programmer-if-I-use-a-lot-of-frameworks-and-libraries>

http://lbstanza.org/purpose_of_programming_languages.html

Programming Design: UML Class Diagram

As I had decided to adopt an object orientated approach to designing my program I used a UML diagram to design the classes that would be needed in my code.

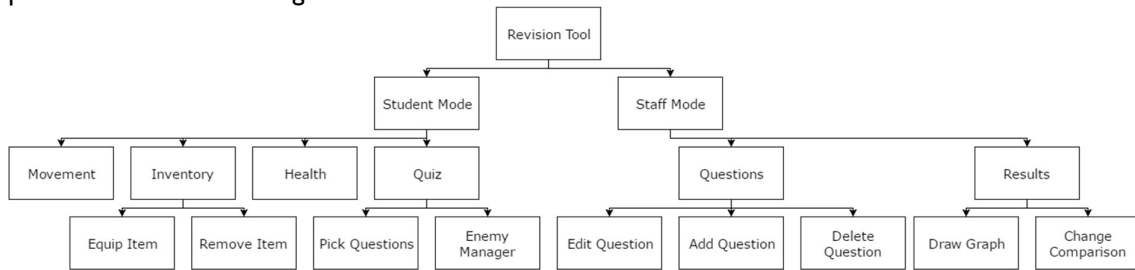


- All the different entities in my game need to be represented in the game world so I'm using a central 'Object' class with a game object and sprite attributes. The 'Object' class has methods to move and toggle the activity of the game object so it can be positioned in the right place.
- The 'Player' class is used hold the methods and attributes relating to the game object the player controls. The player is represented by a sprite in the game world so it inherits from the 'Object' class. The 'Player' class holds values for the player health, attack and defence that are used in quizzes.

- The 'Quiz' class holds methods and attributes used to handles the quizzes used in the game. The class holds an attribute that represents the topic number of a quiz. Quizzes need to be represented in the game world by an enemy so it inherits from the 'Object' class.
- The 'Item' class holds the methods and attributes relating to the items that can be collected in the game. The class inherits from 'Object' so that items can be displayed in the game world. The attributes of 'Item' class hold the type and effect of the item.

Programming Design: Abstraction

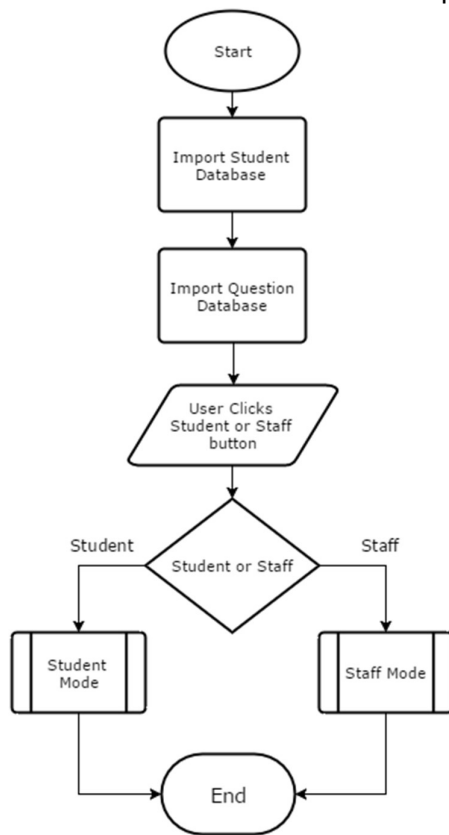
Having investigated the benefits of a modular approach I created a model of my solution broken down into smaller tasks. These smaller problems will be easier to solve and make creating a large piece of code more manageable.



The obvious distinction is that the staff and student portions of the application are separate, these parts are very different in their purpose and it made sense to consider them separately. The student facing portion of the application is then broken down into all the different mechanics of the game. The staff portion of the application is broken down into the two main problems involved with staff users. I have split the issue of dealing with questions into the three main tasks that encompasses: adding, deleting and editing. The problem of handling results is also broken down into the tasks of changing the way data is compared and graphing that data.

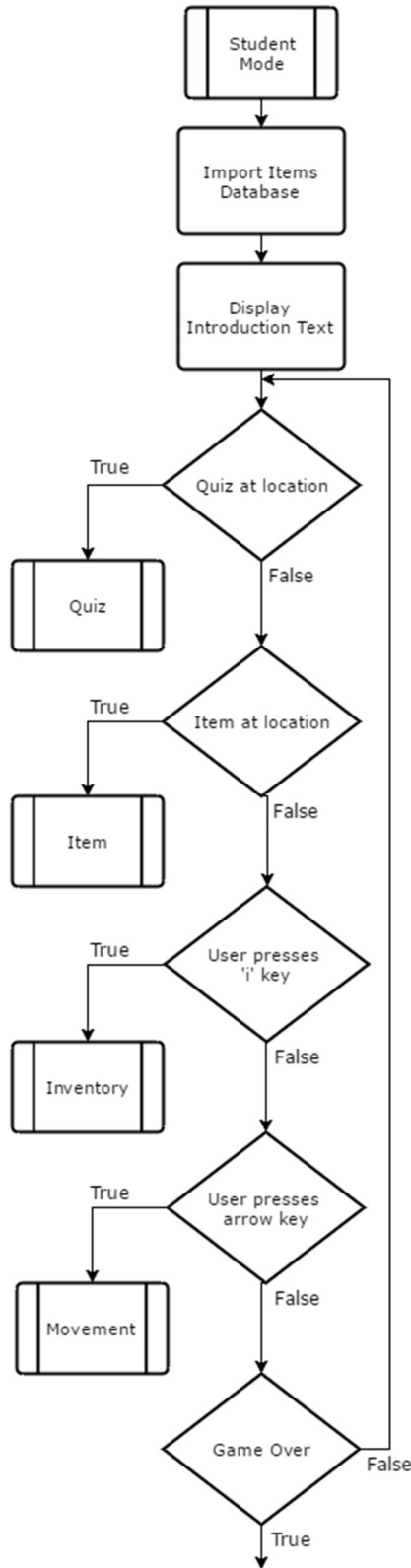
Programming Design: Algorithm Flowcharts

I then created several flowcharts to explain the program flow of my solution:



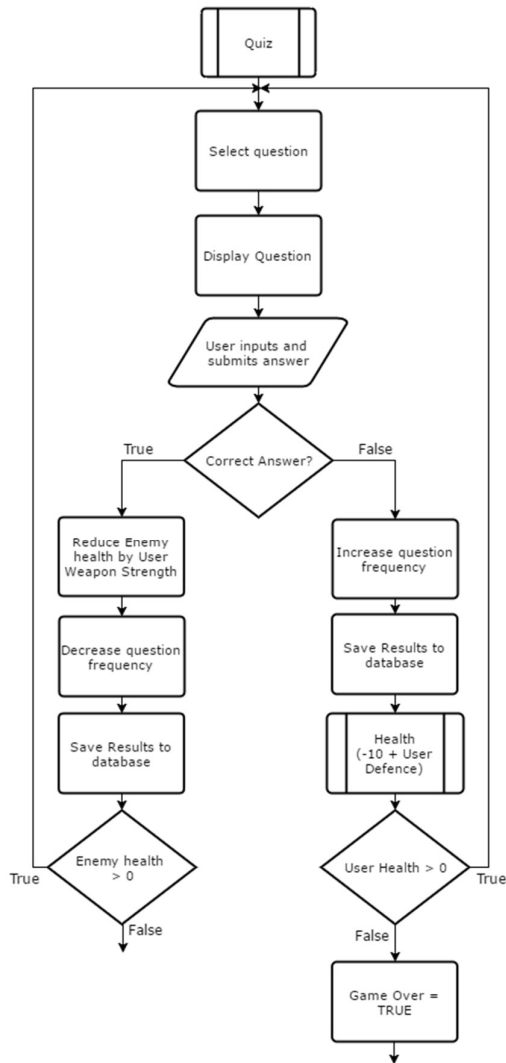
Main Program:

This flowchart outlines the program that will run when the user first starts the software. The program will first import the database of students and questions. The user will then be asked to select if they want to use the Student or Staff mode. Depending on the users' choice either the student or staff functions will be run. When the program flow returns out of these functions the program will terminate.



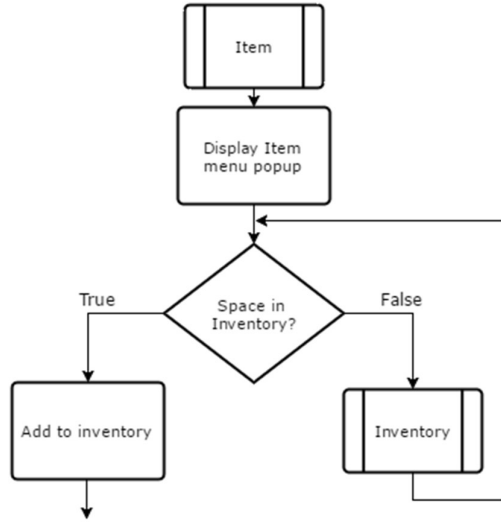
Student Mode:

This function will run when the user selects that they want to use the student portion of the program. This function will first import the student database and display the introductory test. The function will then enter a loop that will continue to iterate as long as the user health value is greater than 0. If there is a quiz at the current location the quiz function will run, if there is an item the item function will run. If the user presses the i key the Inventory function will run and if the user presses an arrow key the movement function will run.

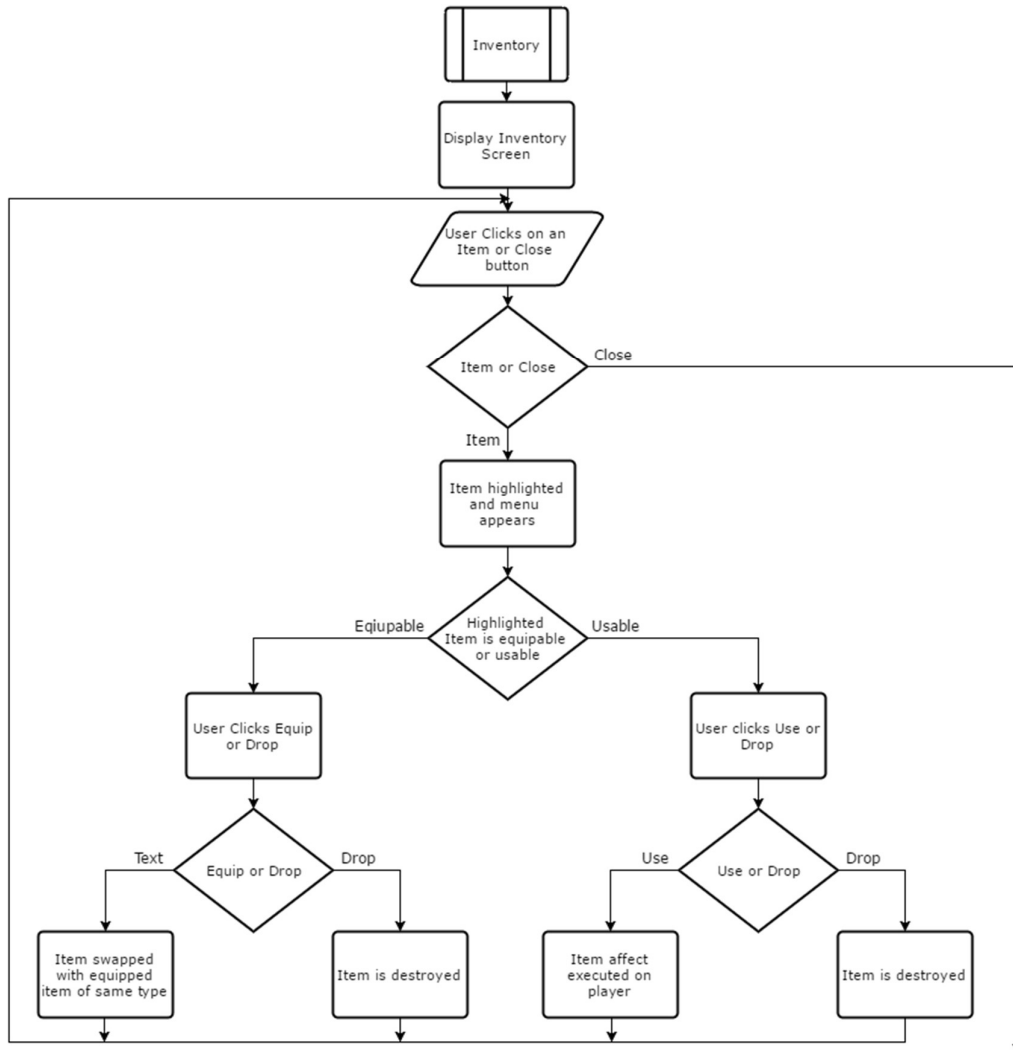


Quiz:

When this function is run the code selects an initial question from the database. The question is displayed and the user inputs an answer. If the answer is correct the enemy health is reduced by the weapon damage value; if the enemy is still alive another question is selected and the game continues. If the answer is incorrect the user health is reduced by 10 reduced by the user defence value; if the user health is 0 or lower the Game Over variable is set to true, otherwise another question is selected. When either the player or the enemy is dead the code stops looping and the function terminates.

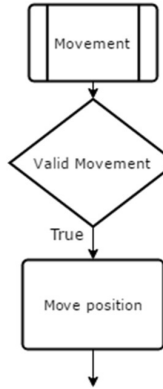


Item:
 When this function runs the item menu popup is displayed. The function checks if there is space in the inventory, if there is the item is added to the inventory. If there isn't the inventory function is run in a loop until there is space.

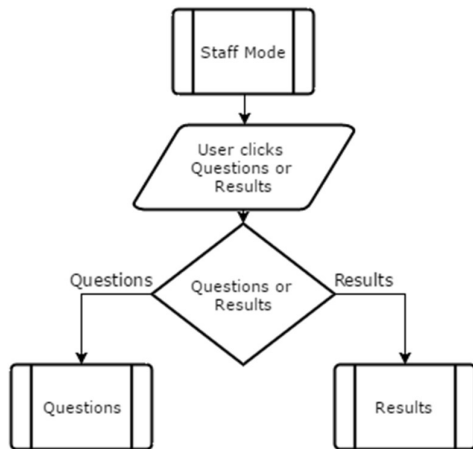


Inventory:

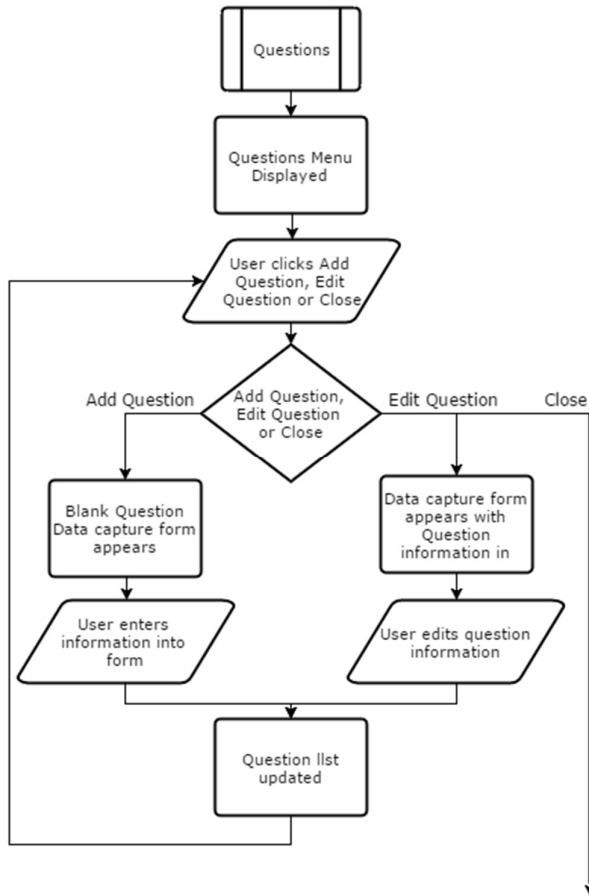
When this function is run the inventory menu is displayed, the user is then able to click an item or the close button. If the user clicks and item and it is equip-able they can click equip or destroy and the item is either swapped with the currently equipped item of the same type or destroyed. If the item is usable it is either used or destroyed. The code loops until the user clicks the close button where the function then terminates.



Movement:
When this function is run it checks that the movement the user selected is valid, if it is the movement is executed.

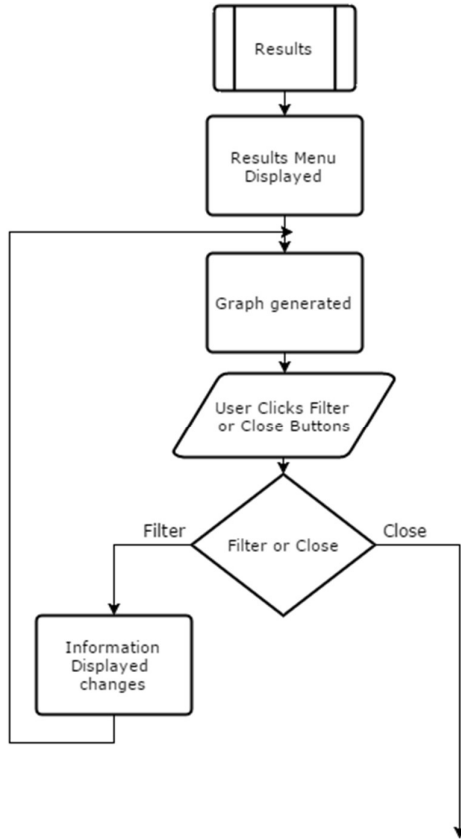


Staff Mode:
When the staff mode function is run the user either clicks the questions or results button, depending on this either the question function of the results function is run.



Questions:

When the questions function is run the questions menu is displayed. The user then clicks to add a question, on an existing question or close. If the user clicks add they fill in a black data capture form and if they click edit they adjust an existing form. The questions list is then updated. The code loops until the user clicks the close button when the function then terminates.



Results:

When this function runs the results menu is displayed. The code then begins it enters a loop where a graph is generated of the currently displayed information. When the user clicks a filter button the information displayed changes. When the user clicks close the program flow exits the loop and the function terminates.

Algorithms Design: Pseudocode

I was satisfied with the outline for the program created by my flowcharts. I then dedicated to use non-language specific pseudocode to write an initial version of the code that will make up my project. This will allow me to lay out a plan for my program without being constrained by the syntax of a specific language. To ensure I was following some form of standard for writing pseudocode I used the OCR Pseudocode guide for A Level Exams:

<http://www.ocr.org.uk/Images/260952-pseudocode-guide-teacher-guide.pdf>

Main Algorithm:

```
public StudentController studentScript  
public StaffController staffScript
```

Variables passed to the algorithm:
Instance of the script that controls the student mode
Instance of the script that controls the staff mode

```
private Boolean Up  
private Boolean Down
```

Variables local to the algorithm:
Boolean variables representing the whether the Up and Down keys are pressed

```
public procedure start()  
    LoadLevel("MainScene")  
    startCoroutine(inputLoop())  
    Up = false  
    Down = false  
Endprocedure
```

Will run when the script begins

Loads the Main scene of the application
Starts the input loop coroutine
Initialises the Boolean variables to false

```
public procedure update()  
    if keyDown()  
        if keyCode == "Up"  
            Up = true  
        elseif keyCode == "Down"  
            Down = true  
        endif  
    endif  
endprocedure
```

Runs every frame
The key pressed variables are toggled if the relevant keys are pressed

```
public IEnumerator inputLoop()  
    while true  
        if Up  
            Up = false  
            yield return startCoroutine(studentScript.gameLoop)  
        elseif Down  
            Down = false  
            yield return startCoroutine(staffScript.inputLoop)  
        endif  
    endwhile  
endIEnumerator
```

A coroutine is a procedure that allows for the use of waiting

Depending on the user input the coroutine loop for either the student or the staff algorithms

Student Mode Algorithm:

```
public string studentName
public sprite playerSprite
public array enemySprites
public array itemSprites
public QuizController quizScript
public InventoryController invScript
```

Variables passed to the Algorithm:
The name of the student user
The player sprite
Array of enemy sprites
Array of item sprites
Instance of the quiz controller script
Instance of the inventory controller script

```
private Player player
private boolean enter
private boolean space
private boolean ikey
private boolean move
private boolean gameOver
private array quizArray
private array itemArray
private array gridPosition
private array directions
```

Variables local to the algorithm
Instance of the player class
Boolean variables to be toggled when keys are pressed

Boolean variable used to end the game
Arrays representing where quizzes and items are located of the game grid
Array representing the player's position
Array representing movement directions

```
class Object
    private string name
    private gameObject instance
    private sprite objectSprite
    public procedure new(string _name, sprite _sprite)
        name = _name
        objectSprite = _sprite
        instance = new gameObject (name)
        instance.sprite = objectSprite
    endprocedure
    public procedure move(int x,int y)
        instance.position = (x,y,0)
    endprocedure
    public procedure destroy()
        destroy(instance)
    endprocedure
endclass
```

Declares a class, Object

Procedure that creates a new Object including a game object with a sprite attached

Procedure that moves the game object to a desired location

Procedure that destroys the game object from the game environment

```
class Quiz inherits Object
    private int topicNo
    public procedure new(int _topicNo, string _name, sprite _ sprite)
        super.new(_name, _sprite)
        topicNo = _topicNo
    endprocedure
    public procedure getTopicNo ()
        return topicNo
    endprocedure
endclass
```

Declares Quiz, inheriting from Object

Procedure that creates a new quiz with an enemy character and a topic

The topic number can only be accessed by methods to maintain encapsulation

```
class Player inherits Object
  private int health
  private int defence
  private int attack
  public procedure new(sprite _sprite, string _name)
    super.new(_name, _sprite)
    health = 100
    defence = 0
    attack = 0
  endprocedure
  public function getHealth ()
    return health
  endfunction
  public procedure setHealth (_health)
    health = _health
  endprocedure
  public function getAttack ()
    return attack
  endfunction
  public procedure setAttack (_attack)
    attack = _attack
  endprocedure
  public function getDefence ()
    return defence
  endfunction
  public procedure setDefence (_defence)
    defence = _defence
  endprocedure
endclass
```

Declares a class, Player, that inherits from the Object class

This new class adds health, defence and attack values to the Object class

To maintain encapsulation the attributes of the player class can only be accessed via methods

```
class Item inherits Object
  private int itemID
  private int effect
  private string type
  public procedure new(int _itemID, int _effect, string type, sprite _sprite, string _name)
    super.new(_name, _sprite)
    itemID = _itemID
    effect = _effect
    type = _type
  endprocedure
  public function getEffect ()
    return effect
  endfunction
  public function getType ()
    return type
  endfunction
end class
```

Declares Item, inheriting from Object

Procedure that constructs a new item

Methods that interact with the attributes, maintaining encapsulation

```
public procedure start()
  player = new Player(studentName, playerSprite)
  enter = false
  space = false
  ikey = false
  move = false
  gameOver = false
  quizArray = new array[5][5]
  itemArray = new array[5][5]
```

Procedure that runs when the script is first activated
A new player is created
The Boolean variables are all initialised as false

The arrays are initialised to the dimensions of the game world

```
gridPosition = {0,0}
directions = {false,false,false,false}
itemMenu.setActive(false)
int randNum
for i=0 to 4
    for j=0 to 4
        randNum = randInt(20)
        if randNum > 9 then
            Sprite enemySprite
            enemySprite = enemySprites[randNum - 10]
            quizArray[i][j] = new Quiz(randNum - 10,
                enemySprite.name, enemySprite)
        endif
    next j
next i
for i=0 to 4
    for j=0 to 4
        randNum = randInt(30)
        if randNum > 9 then
            Sprite itemSprite
            itemSprite = itemSprites[randNum - 10]
            int effect = invScript.findEffect(randNum-10)
            string type = invScript.findType(randNum-10)
            itemArray[i][j] = new Item (randNum-10, effect,
                type, itemSprite, itemSprite.name)
        endif
    next j
next i
endprocedure
```

For loop that randomly populates
the quiz Array with quizzes

For loop that randomly populated
the item Array with items

```
public procedure update()
    if keyDown() then
        if keyCode == "Return"
            enter = true
        elseif keyCode == "Space"
            space = true
        elseif keyCode == "i"
            ikey = true
        elseif keyCode == "Up"
            directions[0] = true
        elseif keyCode == "Down"
            directions[1] = true
        elseif keyCode == "Left"
            directions[2] = true
        elseif keyCode == "Right"
            directions[3] = true
        endif
    endif
endprocedure
```

Procedure that will run on each frame
keyDown() will be True when a key is pressed. These
Boolean variables will toggle when that specific key is down


```
public IEnumerator gameLoop()  
    LoadLevel ("StudentScene")  
    while gameOver  
        if quizArray[gridPosition[0]][gridPosition[1]] != 0 then  
            quizScript.playerHealth = player.getHealth()  
            quizScript.playerDefence = player.getDefence()  
            quizScript.playerAttack = player.getAttack()  
            quizScript.enemyHealth = quizArray[gridPosition[0]][gridPosition[1]].getHealth()  
            quizScript.topicNo = quizArray[gridPosition[0]][gridPosition[1]].getTopicNo()  
            yield return startCoroutine(quizScript.runQuiz())  
            LoadLevel ("StudentScene")  
            player.setHealth() = quizScript.playerHealth  
            quizArray[gridPosition[0]][gridPosition[1]] = null  
        endif  
    endwhile  
endIEnumerator
```

An Coroutine is denoted by 'IEnumerator,' a coroutine is similar to a procedure but allows for waiting and mutasking

This conditional will only be run if the current grid location is populated with a quiz. If this is true the quiz controller script will be updated with the details of this quiz and will then run it. When control of the program flow is returned to this script the Game scene will be reloaded, the player health is updated and the quiz is removed. The yield return statement declares that the main program flow will pause until the runQuiz() coroutine is finished executing

```
        if itemArray[gridPosition[0]][gridPosition[1]] != 0 then  
            yield return startCoroutine(invScript.newItem(itemArray[gridPosition[0]][gridPosition[1]]))  
            LoadLevel ("StudentScene")  
        endif  
    endwhile
```

This conditional will only run if the current grid location is populated with an item. If this is true the detail of that item are passed as a parameter of the newItem procedure in the Inventory controller script.

```
        if ikey then  
            yield return startCoroutine(invScript.inventory())  
            LoadLevel ("StudentScene")  
        endif  
        player.attack = invScript.findAttack()  
        player.defence = invScript.findDefence()  
        player.health = player.health + invScript.findHealth()  
        yield return startCoroutine(move())  
        yield return null  
    endwhile  
endIEnumerator
```

If the i key is pressed the inventory is opened

The player attributes are updated

After all check are run the move() coroutine is run

A yield return of null is used to stop a loop in a coroutine causing the program to crash

```
public IEnumerator move()  
    int x = gridPosition[0]  
    int y = gridPosition[1]  
    directions = {false,false,false,false}  
    while move == false  
        if directions[0]=true && gridPosition[1] < 4 then  
            gridPosition = {x,y+1}  
            directions[0] = false  
            move = true  
        elseif directions[1]=true && gridPosition[1] > 0 then  
            gridPosition = {x,y-1}  
            directions[1] = false  
            move = true  
        elseif directions[2]=true && gridPosition[0] > 0 then  
            gridPosition = {x-1,y}  
            directions[2] = false  
            move = true  
        elseif directions[3]=true && gridPosition[1] < 4 then  
            gridPosition = {x+1,y}  
            directions[3] = false  
            move = true  
        endif  
    endwhile  
    yield return null
```

The four directional Boolean variables are set to false

A while loop is run until the character has moved

If the user enters a valid movement the grid position is altered and the loop stops

Joe Rackham: 5391
Tendring Technology College: 16455

OCR F454 Computing Project
Computing Revision Game

```
    endwhile  
    move = false  
    player.move(gridPosition[0],gridPosition[1])  
endIEnumerator
```

The player gameObject is moved

Quiz Controller Algorithm:

```
public db database
public int playerHealth
public int playerDefence
public int playerAttack
public int enemyHealth
public int topicNo
public UI.Text pHealthText
public UI.Slider pHealthSlider
public UI.Text eHealthText
public UI.Slider eHealthSlider
public UI.Text questionText
public UI.TextField answerField
```

Variables passed to the algorithm:
Database containing stored information
Player health
Player defence, reduces damage taken by player
Player attack, increases damage done to enemy
Enemy health
The topic all questions will be taken from
UI Elements that represent the health of the player and enemy both numerically and graphically

UI text where the question is displayed
UI text field where the answer is entered

```
private array topicQuestions
private array questionArray
private file myFile
private boolean enter
```

Variables local to the algorithm:
Array of all the questions from the topic
Array holding the details of a specific question
File variable used to read and write to the database
Boolean variable used to indicated the enter key is pressed

```
public procedure start()
    myFile = openRead(database)
    topicQuestions = myFile.query("TOPIC == " + topicNo)
    topicQuestions.sort(comparator [][][3])
    gameObject player = new gameObject
endprocedure
```

Procedure that will run when the script is enabled
Opens the Database for reading and writing to
Populates the array with all the relevant questions
Sorts the questions in order of difficulty

```
public procedure update()
    pHealthText.text = playerHealth
    pHealthSlider.value = playerHealth
    eHealthText.text = enemyHealth
    eHealthSlider.value = enemyHealth
    if keyDown() && keyCode == "Return" then
        enter = true
    endif
endprocedure
```

Procedure that will run each frame
Updates the UI element so they are accurate

Turns enter to true is the enter key is pressed

```
public IEnumerator runQuiz()
    LoadLevel ("QuizScene")
    int question = 0
    string answer
    while playerHealth > 0 && enemyHealth > 0
        if question = topicQuestions.length
            question = 0
        endif
        questionArray = topicQuestions[question]
        enter = false
        while enter == false
            questionText = questionArray[0]
            yield return null
        endwhile
        answer = answerField.text;
        if answer == topicQuestions[1]
            enemyHealth = enemyHealth - 10 - playerAttack
            questionText.text = "Correct"
            questionArray[3] = questionArray[3] - 1
        else
            playerHealth = playerHealth - 10 + playerDefence
```

Loads the quiz scene
Sets the question index to 0

Loops until either the player or enemy is dead

Displays the question text

Conditional that checks if the answer is correct. If it's right the enemy health is reduced by 10 plus the attack modifier, if it's wrong the player health is reduced by 10 minus the defence modifier. Either way the 'Difficulty' value is adjusted.

Joe Rackham: 5391
Tendring Technology College: 16455

OCR F454 Computing Project
Computing Revision Game

```
        questionText.text = "Incorrect"  
        questionArray[3] = questionArray[3] + 1  
    endif  
    yield return new WaitForSeconds (2)  
    answerField.text = string.Empty  
endwhile  
endIEnumerator
```

Inventory Algorithm:

```
public db database
public array items
public array itemSprites
public UI itemMenu
```

Variables Passed to the Algorithm:
Database containing stored information
Array holding the items the player has
Array containing sprites to assign to items
UI items used to convey instructions when an item is selected

```
private array instances
private array equipIndex
private array equipObjects
private file myFile
private Boolean esc
private Boolean click
private Boolean Up
private Boolean Down
private int healthOffset
```

Variables Local the Algorithm:
Array holding all the game objects representing the items
Array holding the index of all the equipped items
Array holding the game objects for the equipped items
File used to view the database
Boolean Variables used to represent the keys pressed

Variable representing any offset to the health done during this use of the inventory

```
public procedure start()
    myFile = openRead(database)
    esc = false
    click = false
endprocedure
```

Procedure that will run when the script is first enabled
Makes the Item database available to read from
Initialises the key press variables to false

```
public procedure update()
    int identifier
    int xPosition
    int yPosition
    for i=0 to items.length()
        items[i].move (i % 5, I DIV 5)
    next i
    for i=0 to 2
        equipObjects[i] = items[equipIndex[i]]
        equipObjects[i].instantiate((0,2-i,0),Euler.identity) as gameObject
    next i
    if keyDown() then
        if keyCode == "Escape" then
            esc = true
        elseif keyCode == "Up" then
            Up = true
        elseif keyCode == "Down" then
            Down = true
        endif
    endif
    if mouseDown() then
        click = true
    endif
endprocedure
```

Procedure that will run every frame

Loop that runs through the item array and moves them to the right position on the inventory screen

Creates game object for the three equipped objects

Toggles the key Boolean variables when they are pressed

```
public IEnumerator newItem(Item newItem)
    Boolean placed = false
    while placed == false
        for j=0 to items.length()
            if items[j] != null then
                items[j] = newItem
                placed = true
                break
            endif
        next j
    end while
end IEnumerator
```

Coroutine used to add a new item to the inventory

For loop that looks for the first available space to put the item into

```
                endif
            next i
            yield return startCoroutine(inventory)
        endwhile
    endIEnumerator
```

If there is no free space the inventory is opened and will be repeatedly until space is made

```
public IEnumerator inventory()
    LoadLevel ("Inventory Scene")
    int selected
    esc = false
    click = false
    while esc == false
        if click then
            xPosition = mousePosition.x / I
            yPosition = mousePosition.y / I
            selected = x+5*y
            if items[selected] != null then
                itemMenu.SetActive(true)
                itemMenu.position = (x,y,0)
                Up = false
                Down = false
                while esc == false
                    if Up then
                        type = items[selected].type
                        effect = items[selected].effect
                        if type < 3 then
                            equipIndex[type] = selected
                        else
                            healthOffset = 0
                        endif
                    elseif Down then
                        destroy(items[selected])
                    endif
                endwhile
                esc = false
            endif
        endif
    endwhile
endIEnumerator
```

Loads the inventory scene

Loop that runs until escape is pressed

When the user clicks their mouse position is turned into an index in the inventory. If the Inventory location is populated the item menu is activated

If the user presses up they select "Equip/Use." If the item is equipable the array of equipped items is changed, if not the effect of the item is applied to the health offset

If the user presses down the select "Drop" and the item is destroyed

```
public Integer function findAttack()
    return items[equipIndex[1]].effect
endfunction
```

Functions that can be called to find the current attack, defence and health offset value to be used in the main program

```
public Integer function findDefence()
    return items[equipIndex[2]].effect
endfunction
```

```
public Integer function findHealth()
    int h = healthOffset
    healthOffset = 0
    return h
endfunction
```

The findHealth function also resets the offset to 0

Staff Mode Algorithm:

```
public QuestionController questionScript
public ResultsController resultsScript

private Boolean Up
private Boolean Down

public procedure start()
    LoadLevel("StaffScene")
    startCoroutine(inputLoop())
    Up = false
    Down = false
endprocedure

public procedure update()
    if keyDown()
        if keyCode == "Up"
            Up = true
        elseif keyCode == "Down"
            Down = true
        endif
    endif
endprocedure

public IEnumerator inputLoop()
    while true
        if Up
            Up = false
            yield return startCoroutine(questionScript.inputLoop)
        elseif Down
            Down = false
            yield return startCoroutine(resultsScript.inputLoop)
        endif
    endwhile
endIEnumerator
```

This algorithm is largely similar to the Main controller algorithm that runs first in the application. The algorithm presents the user with the Staff Scene, from this they input either up or down and coroutines from either the results or the questions algorithms are run.

Questions Algorithm:

```
public db database
public UI.table table
public UI.form captureForm
public gameObject selectBox
```

```
private file myFile
private Boolean Up
private Boolean Down
private Boolean Left
private Boolean Right
private Boolean esc
private Boolean a
private Boolean e
private Boolean d
private int filter
private array questions
private int selection
```

```
public procedure start()
    Up = false
    Down = false
    Left = false
    Right = false
    esc = false
    a = false
    e = false
    d = false
    filter = 1
    selection = 0
    refresh()
endprocedure
```

```
public procedure update()
    if keyDown() then
        if keyCode == "Up"
            Up = true
        elseif keyCode == "Down"
            Down = true
        elseif keyCode == "Left"
            Left = true
        elseif keyCode == "Right"
            Right = true
        elseif keyCode == "Escape"
            esc = true
        elseif keyCode == "A"
            a = true
        elseif keyCode == "E"
            e = true
        elseif keyCode == "D"
            d = true
        endif
    endif
    selectBox.position = (2.5,selection * table.rowWidth,0)
endprocedure
```

```
public procedure refresh()
```

Variables passed to the algorithm:
Database containing all the stored information
UI table used to display questions
UI form used to capture data
Game object used to highlight the selected question

Variables local to the algorithm:
File used to read and write from the database
Boolean variables representing which keys are pressed

Number representing the current filter setting
Array used to hold all the questions
Number representing which question is selected

Procedure run when the algorithm is first activated
Boolean key variables initialised to false

Selection variable initialised to the first question
Refresh procedure run

Procedure that runs every frame
Check which key are pressed and toggles
the variables accordingly

Moves the selection box to the right
position


```
myFile = openRead(database)
questions = myFile.all
array temp
int swaps = 0
do
    swaps = 0
    for int i=1 to questions.length()
        if questions[i][filter] < questions[i-1][filter] then
            swaps = swaps + 1
            temp = questions[i]
            questions[i] = questions[i-1]
            questions[i-1] = temp
        endif
    next i
while swaps != 0
table.display(questions)
endprocedure
```

Opens the database to be read from
Extracts all the questions

Performs a bubble sort on the questions
according to the current filter setting

Displays the questions in the table

```
public IEnumerator inputLoop()
    LoadLevel("QuestionScene")
    while (esc == false)
        if Up && selection > 0
            selection = selection - 1
        elseif Down && selection < questions.length() - 1
            selection = selection + 1
        endif
        if Right && filter < 5 then
            Up = false
            filter = filter + 1
        elseif Left && filter > 1
            Down = false
            filter = filter - 1
        endif
        if a
            yield return startCoroutine(add())
        elseif e
            yield return startCoroutine(edit())
        elseif d
            yield return startCoroutine(delete())
        endif
        refresh()
    endwhile
endIEnumerator
```

Loads the question scene
Loop until the user presses escape
The up and down keys will change the currently
selected question

The left and right keys will change the filter setting

The a key runs the coroutine to add a question

The e key runs the coroutine to edit a question

The d key runs the coroutine to delete a question

```
public IEnumerator add()
    captureForm.SetActive(true)
    captureForm.clear()
    yield return startCoroutine(captureForm.fill())
    myFile = openWrite(questionDB)
    questionDB.write(captureForm)
    captureForm.SetActive(false)
endIEnumerator
```

The capture form is displayed and cleared

The program waits until the form is submitted

The new question is written to the database and the system
is refreshed

```
public IEnumerator edit()
    captureForm.SetActive(true)
    captureForm.populate(questions[selection])
    yield return startCoroutine(captureForm.fill())
```

The capture form is displayed and populated with the
details of the selected question

The program waits until the form is submitted

```
myFile = openWrite(questionDB)  
myFile.edit(captureForm,"QID == " + selection)  
captureForm.setActive(false)  
endIEnumerator
```

The new question is written to the database and the system is refreshed

```
public IEnumerator delete()  
myFile = openWrite(questionDB)  
myFile.remove("QID == " + selection)  
endIEnumerator
```

The currently selected question is removed from the database and the system is refreshed

Results Algorithm:

```
public db database  
public UI.table table  
public UI.barChart graph
```

```
private file myFile  
private Boolean Up  
private Boolean Down  
private Boolean Left  
private Boolean Right  
private Boolean esc
```

```
public procedure start()  
    Up = false  
    Down = false  
    Left = false  
    Right = false  
    esc = false  
    refresh()  
endprocedure
```

```
public procedure update()  
    if keyDown() then  
        if keyCode == "Up"  
            Up = true  
        elseif keyCode == "Down"  
            Down = true  
        elseif keyCode == "Left"  
            Left = true  
        elseif keyCode == "Right"  
            Right = true  
        elseif keyCode == "Escape"  
            esc = true  
        elseif keyCode == "A"  
            a = true  
        elseif keyCode == "E"  
            e = true  
        elseif keyCode == "D"  
            d = true  
        endif  
    endif  
endprocedure
```

```
public procedure refresh()  
    myFile = openRead(database)  
    questions = database.all  
    table.display(questions)  
    table.setMain(filter)  
    graph.display(questions,filter)  
endprocedure
```

Variables passed to the algorithm:
Database containing all the questions
UI table used to display the questions
UI bar chart used to display and compare results

Variables local to the algorithm:
File used to read and write from the database
Boolean variables representing which keys are pressed

Procedure run when the algorithm is first activated
Boolean key variables initialised to false

Refresh procedure run

Procedure that runs every frame
Check which key are pressed and toggles the variables accordingly

Opens the database to read from
Extracts all the questions from the database
Displays the questions in the table
Sets the first column according to the current filter setting
Displays the questions in the graph comparing them in respect to the current filter setting

```
public IEnumerator inputLoop()  
    LoadLevel("ResultsScene")  
    while (esc == false)  
        if Right && filter < 2 then  
            Up = false  
            filter = filter + 1  
        elseif Left && filter > 0  
            Down = false  
            filter = filter - 1  
        endif  
        refresh()  
    endwhile  
endIEnumerator
```

Loads the results scene
Loop until the user presses escape
The left and right keys will change the filter setting

Pseudocode summary:

Writing pseudocode was incredibly helpful in allowing me to outline my thinking in code form without the obstacle of syntax. I now have a comprehensive plan to use when coding. Dealing with a database and with graphs will involve unfamiliar syntax; my pseudocode will hopefully prevent my code deviating from its purpose and ensure it meets the goals I've set out.

Some areas of my pseudocode are somewhat overly simplified. Reading from a database and creating a graph will likely not be as simplistic as the code fragments I have used in my pseudocode. I have done this because I believe it would be misguided to create my own psudeo-syntax for the purpose of planning, only to have to substitute it all for my actual code. Statements such as `graph.display()` and `captureForm.fill()` convey what I want to achieve but without a complication series of statements that I believe would be counterproductive.

Aside: Use of Global Variables

Through some online research and personal experience with programming I know the use of global variables is generally discouraged. Programmers warn that the use of global variables can propagate errors through the fact that different subroutines in different parts of the program can interfere with each other. However, I have used global variables in my planned algorithms, I intent to do so in the development of my actual application and I believe this decision is justified.

I believe the nature of my application lends itself to the use of global variables. The interference programmers talk about is something that will be useful in my case. For example the Boolean values are used to represent user input are global because I want them to be accessible in the main loop of each individual scene but I don't want the user to have to hold down the key they want to press so it is evaluated as true at a specific part of the loop. More-over, values like the player's health and their equipped items are changed in the quiz and inventory code but then need to be updated in the base student mode game loop. Using global variables prevents the need to have some way to return these values back up to the main loop. The nature of my application means that things don't need to be done in a specific order and actions in one part of the application are supposed to affect others. I therefore feel justified in my decision to use global variables:

<http://www.learncpp.com/cpp-tutorial/4-2a-why-global-variables-are-evil/>

<https://softwareengineering.stackexchange.com/questions/148108/why-is-global-state-so-evil>

<http://wiki.c2.com/?GlobalVariablesAreBad>

Testing Strategy

Testing is an imperative part of the development of my application; effective testing will ensure my application is without errors or critical flaws. Testing will be broken down into four stages: In

Development Testing, Post Development Alpha Testing and Beta Testing before finally moving on to acceptance testing.

In Development Testing

Throughout the development of my application I will need to test that new features are working properly before continuing. By creating sample users and playing the game portion of the application myself I will compare the applications behaviour with the expected result. It is important that I test both valid and invalid data to make sure my application deals with each in the appropriate way. Only testing using valid data would be insufficient as it ignores the very possible scenario of a user using the system incorrectly and the errors that would arise in that case. This phase of testing is 'White Box' testing; even if the user facing make it appear like the application is responding correctly I must also check that the code behind the user interface is producing the expected results. To this end, I will be making use of Debug statements in the code, which produce an output to the developer console, as a way of testing the status of the code at various points. Whenever an error is encountered I must pause development and attempt to fix it. When I think I have rectified the issue the test must be repeated and only when the application has passed the test can development continue.

Alpha Testing

This stage of testing will occur when I think the project is complete. I will thoroughly test every part of the software as well as how the component parts interact with each other. I will create a full set of test data to use with the program, intending to replicate real use of the system as accurately as possible. Furthermore, this test cases will be designed to test the robustness of the system. A combination of valid, invalid, boundary and erroneous data must be used to ensure the program deals with each in the appropriate way. This, again, is 'White Box' testing; It is important when determining if the system works correctly that I make sure the values and states in code are as expected instead of being satisfied with how the UI looks. This is especially important when using boundary data, the nature of which makes it prone to being treated as if it belongs to a different set.

Beta Testing

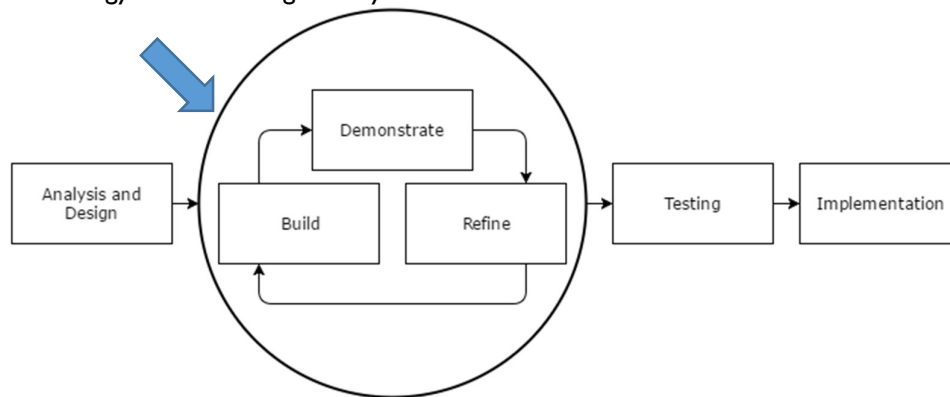
After Alpha testing is completed I will present my application to a group of end users to assess the usability of the application. I intend to ask the users to accomplish specific tasks in the application as well as allowing them to use it on their own. This will be 'Black Box' testing; the users won't have an understanding of how the code behind the application works. This will be useful in determining how easy my application is to use and how the way it functions differs from what users expect. This phase of testing will also make the application deal with lots of invalid actions and input from the users, further testing the durability of the application in a real world scenario. The experience of the users will inform me if I need to add some instructions to the game or change some of the controls. Furthermore, user feedback from Beta testing will be invaluable when it comes time to evaluate the Revision Game.

Acceptance Testing

By the time the application is in this stage of all errors should be fixed and issues with the application identified. Instead, the purpose of this phase of testing is to compare the solution I have produced with the success criteria outline at the start of development to evaluate whether I have created an effective solution to the problem. This phase of testing will involve me presenting my application to my end user so that he can also comment on how well my application meets his demand and how well he thinks it will address the issue of Revision.

Project Development

In this section I will outline how I developed my project. I aim to display the process I have taken and the strategy I have employed. I intend to implement good programming standards and follow the RAD methodology I selected to guide my software creation.

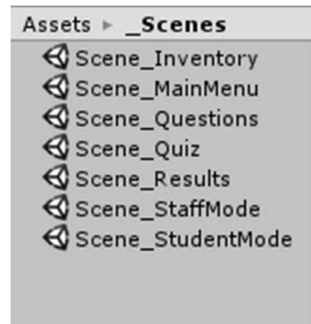


I am currently in the looping stage of development. The process I need to follow is to create iteratively add features to my code until I have achieved all my success criteria. It is important that I move from one working version of code to another so at no point is there a version of my application that doesn't function. To enforce this I will be carrying out in-development testing to make sure every new feature added is in working order before moving on the next. Furthermore, I must also present each new version of the application to my client to make sure he is satisfied with the way it works and that I am making good progress towards meeting the project brief.

By following the RAD method I hope to make sure the project stays on track. Because each phase of development will have a specific task that needs to be accomplished I shouldn't be enticed into getting distracted and working on unnecessary features. More-over, by staying in constant contact with my end user I can be confident whilst working on the project that what I'm making will meet the clients desires; any issues can be addressed before development has moved on to another part of the application.

Phase I: Scene Transition and Movement

Beginning my project I argued that the most basic function of my project was a series of scenes the user could move between, one of which took the form of a game. To meet this, most basic of deliverables I created the scenes needed in Unity and wrote code that allowed me to transition between them.



For every distinct UI screen I outlined in the Design stage I create a scene. The Unity Game engine attributes all game objects to a specific scene, this makes it easy to change the screen layout easily by switching scenes.

To each scene I attached a script to the camera that I would use to control that scene. To the Main Scene script I wrote some code to control movement between these scenes.

```
private bool Up;
private bool Down;

void Start () {
    Up = false;
    Down = false;
    StartCoroutine (inputLoop ());
}
```

The code here displays the inbuilt library subroutines, Start and Update I used in my pseudocode. As explained before, the Start procedure runs on the first frame after the script is enabled and the Update function runs every frame thereafter.

```
void Update () {
    if (Input.GetKeyDown (KeyCode.UpArrow)) {
        Up = true;
    }
    if (Input.GetKeyDown (KeyCode.DownArrow)) {
        Down = true;
    }
}
```

As planned, I used Boolean variables attributed to keyboard keys to manage user interaction.

```
public IEnumerator inputLoop() {
    while (true) {
        if (Up) {
            Up = false;
            SceneManager.LoadScene ("Scene_StudentMode", LoadSceneMode.Single);
            break;
        } else if (Down) {
            Down = false;
            SceneManager.LoadScene ("Scene_StaffMode", LoadSceneMode.Single);
            break;
        }
        yield return null;
    }
}
```

As I explained in my pseudocode coroutines are types of functions that exist in Unity, the distinction being that events in the coroutine don't have to be executed in one frame. In this piece of code I'm using a coroutine with a loop to constantly check for user input.

When the user does input a valid key the relevant scene is loaded. The 'Single' load mode means the current scene will be destroyed.

To the Student Mode Controller, the script controlling the game, I began to build up the code I would need to reach the point of a moving character. The first step in this was to create the character class.

```
class Character {
    private int health;
    private Sprite characterSprite;
    private GameObject instance;
    private Rigidbody rigidbody;
    private SpriteRenderer spriteRenderer;
    public Character(Sprite sprite, string name) {
        health = 100;
        characterSprite = sprite;
        instance = new GameObject(name);
        instance.AddComponent<SpriteRenderer> ();
        instance.AddComponent<Rigidbody> ();
        rigidbody = instance.GetComponent<Rigidbody>();
        rigidbody.isKinematic = true;
        rigidbody.useGravity = false;
        spriteRenderer = instance.GetComponent<SpriteRenderer>();
        spriteRenderer.sprite = characterSprite;
    }
    public void setHealth(int currentHealth) {
        health = currentHealth;
    }
    public int getHealth() {
        return health;
    }
    public void move(float x, float y) {
        rigidbody.position = new Vector3 (x, y, 0f);
    }
    public void deactivate() {
        instance.SetActive (false);
    }
}
```

The character class will be a template from which all enemies will be created. The class will also be inherited from by the player class.

The code was largely similar to my pseudocode but in Unity I had to manually add components to the created game object. A Rigidbody allows an object to be subject to physics and a Sprite Renderer allows a game object to take on an external PNG sprite.

I wrote a getter and setter for the health variable. Using these and making the variable private ensures to class is encapsulated.

The next logical step was to write the Player class because it inherits from the Character class, I also wrote a largely empty quiz class to work on at a later stage.

```
class Player : Character {
    private int attack;
    private int defence;
    public Player (Sprite sprite, string name) : base(sprite,name) {
        attack = 0;
        defence = 0;
    }
    public void setAttack(int newAttack) {
        attack = newAttack;
    }
    public int getAttack() {
        return attack;
    }
    public void setDefence(int newDefence) {
        defence = newDefence;
    }
    public int getDefence() {
        return defence;
    }
}
```

```
class Quiz {
    private int topicNumber;
    public Quiz (int _topicNumber) {
        topicNumber = _topicNumber;
    }
    public int getTopicNumber () {
        return topicNumber;
    }
}
```


Now that I had defined the classes needed for my student game mode I could write the start and update subroutines needed to initialise and maintain the game world. Because the Student Mode scene wouldn't be the first loaded the start subroutine only initialised the variables used in the script. The update subroutine, simply checked for key presses to adjust key variables.

The start subroutine ended with it calling a coroutine used to control the game loop.

```
public IEnumerator gameLoop () {  
    player = new Player (playerSprite, "Player");  
    gridPosition = new int[2] {0,0};  
    directions = new bool[4] { false, false, false, false };  
    int randomNumber;  
    for (int i=0;i<5;i++) {  
        for (int j=0;j<5;j++) {  
            randomNumber = Random.Range (1,12);  
            if (randomNumber > 6) {  
                quizArray [i,j] = new Quiz (randomNumber - 6);  
            }  
            randomNumber = Random.Range (0,30);  
            if (randomNumber > 9) {  
                itemArray [i,j] = randomNumber - 10;  
            }  
        }  
    }  
    quizArray [0,0] = null;  
    while (!gameOver) {  
        if (quizArray [gridPosition [0],gridPosition [1]] != null) {  
            SceneManager.LoadScene ("Scene_Quiz", LoadSceneMode.Additive);  
        } else if (itemArray [gridPosition [0],gridPosition [1]] != 0) {  
        } else if (iKey) {  
            SceneManager.LoadScene ("Scene_Inventory", LoadSceneMode.Additive);  
            iKey = false;  
        }  
        yield return StartCoroutine (mover ());  
        yield return null;  
    }  
    yield return null;  
}
```

I used a coroutine for the game loop so I could make the game engine wait for user action.

A random number generator is used to randomly populate the game world with quizzes and items, only some of the indexes are populated so some locations are blank. The origin index is also set to null so the game doesn't start with a quiz.

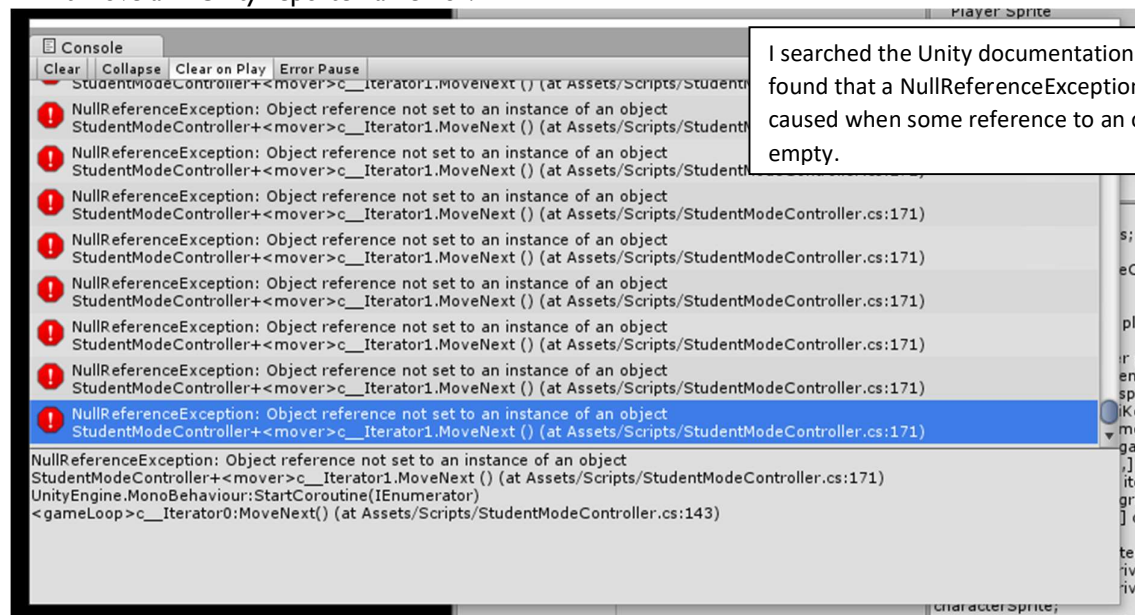
These if statements are to do with quizzes and items so I'm leaving them blank at this stage.

At this stage I had created the class the player would be created from and created the game world it would exist in. I then went on to write the script to move the player and tested my code so far in engine.

```
public IEnumerator mover() {  
    int x = gridPosition [0];  
    int y = gridPosition [1];  
    directions = new bool[4] {false,false,false,false};  
    move = false;  
    while (move == false) {  
        if (directions [0] == true && gridPosition [1] < 4) {  
            gridPosition [0] = x;  
            gridPosition [1] = y + 1;  
            move = true;  
        } else if (directions [1] == true && gridPosition [1] > 0) {  
            gridPosition [0] = x;  
            gridPosition [1] = y - 1;  
            move = true;  
        } else if (directions [2] == true && gridPosition [0] > 0) {  
            gridPosition [0] = x - 1;  
            gridPosition [1] = y;  
            move = true;  
        } else if (directions [3] == true && gridPosition [0] < 4) {  
            gridPosition [0] = x + 1;  
            gridPosition [1] = y;  
            move = true;  
        }  
        yield return null;  
    }  
    player.move(gridPosition[0],gridPosition[1]);  
    yield return null;  
}
```

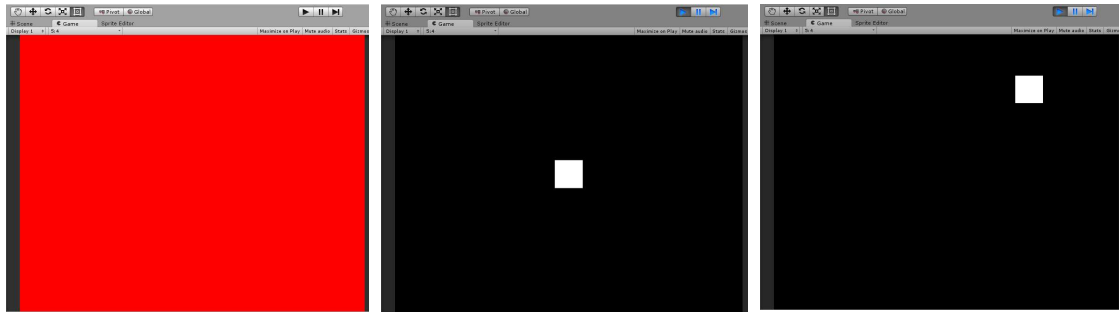
Using the coordinates stored in the grid position array and Boolean indicators in the directions array this procedure adjusts the position of the player.

This script worked to create a game object but when the mover subroutine was run the object didn't move and Unity reported an error.



With the knowledge of what the error meant from the documentation I traced the error back to a line in the game loop coroutine. I re-declared the variable when I instantiated the player, this meant the player object was limited in scope to the game loop and the version of the variable the mover could access was null. Changing this line rectified the problem.

Original Code	<code>Player player = new Player (playerSprite,"Player");</code>
Fixed Code	<code>player = new Player (playerSprite,"Player");</code>



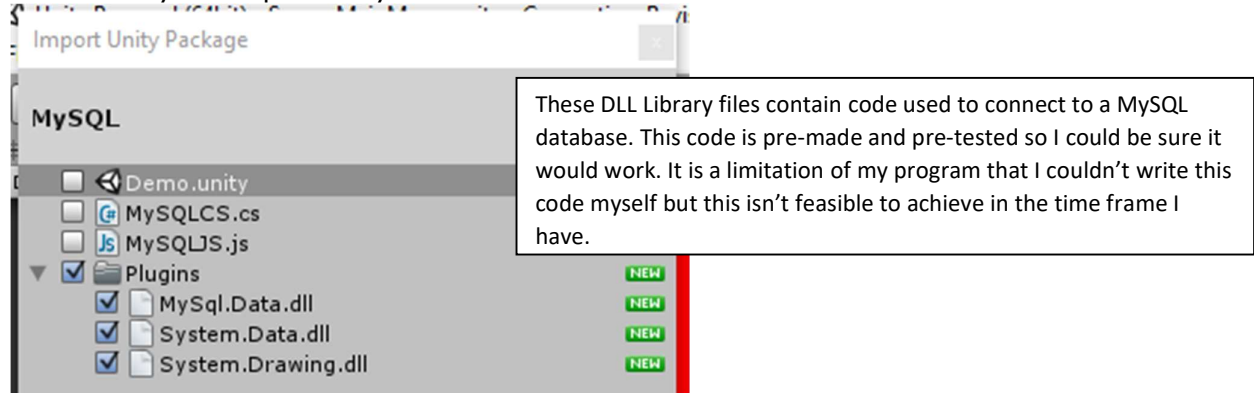
The three screen grabs above show how the code I've written allows me to move from the main scene to other scenes and how the Student Mode scene instantiates a player object the user can move. For simplicities sake I haven't yet implemented any art work but the code foundation meet the deliverables I set out to accomplish.

Client Feedback

At this stage in development there wasn't a lot visibly happening in the application for my client to comment on. Regardless, I presented the current version to My Byford. He was happy with splitting the different features of the application into different scenes and that the grid style movement pattern we discussed in the design stage of development was working as he envisioned it.

Phase 2: Database Connection

Following on from the creation of the basic functions I decided the next logical stage in development would be to create my database and write the code to connect to it. This will allow me to create the quiz and staff sides of the application. The first stage in doing this is was to import plugins used to connect my C# scripts to a MySQL database.



I then wrote a script to control the connection. The first stage in this was to import the new libraries required to interact with the database.

```
using UnityEngine;
using System;
using System.Data;
using System.Text;
using System.Collections;
using System.Collections.Generic;
using MySql.Data;
using MySql.Data.MySqlClient;
```

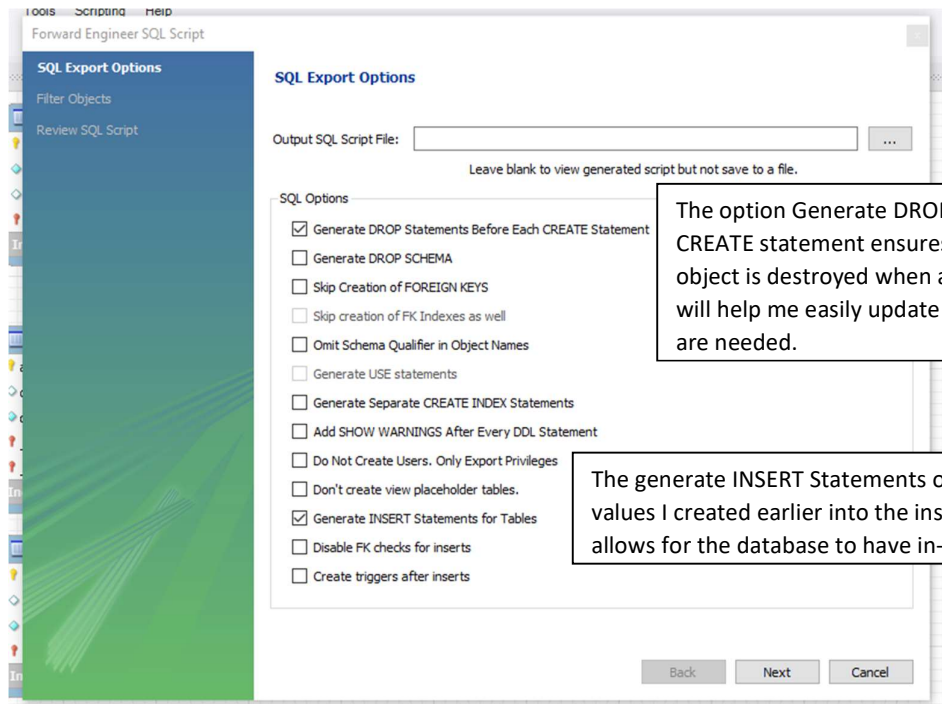
Following on from this I then wrote code that would initiate a connection to a database.

```
void Awake () {
    DontDestroyOnLoad (this.gameObject);
    connectionString = "Server="+host+";Database="+database
        +";User="+user+";Password="+password+";Pooling=true";
    try {
        connection = new MySqlConnection(connectionString);
        connection.Open();
        Debug.Log("MySQL State: "+connection.State);
    } catch (Exception e) {
        Debug.Log (e);
    }
}
```

This section collates all the information used to instantiate the connection.

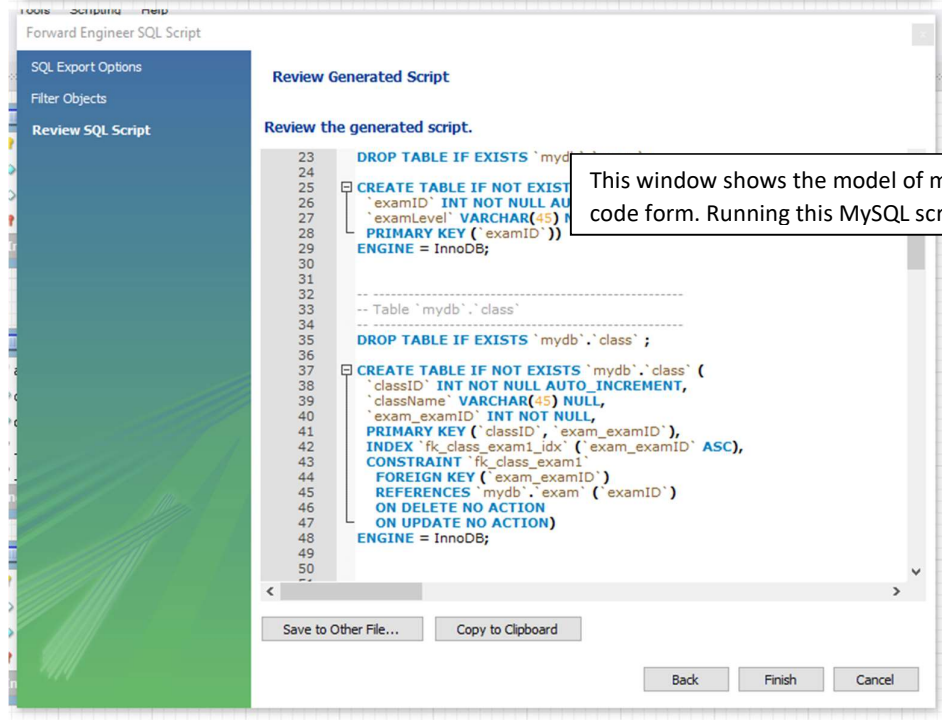
This section of the code attempts to connect to the database, using the try structure prevents the code crashing if the connection fails.

With my main application now capable of working with a database I converted the model I created in MySQL workshop to an actual database.



The option Generate DROP statements before each CREATE statement ensures that past instances of each object is destroyed when a new one is created, this will help me easily update the database if any changes are needed.

The generate INSERT Statements option inputs the insert values I created earlier into the instance of the database. This allows for the database to have in-built values.



This window shows the model of my database recreated in code form. Running this MySQL script generated the database.

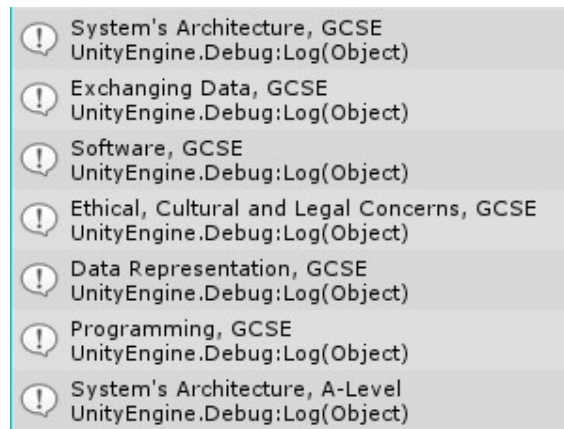
Following this I entered the details of the database and attempted to connect. The MySQL connector worked and a connection was established.



The 'Host' field holds the address where the database is located, this is my laptop at this stage but will eventually be an IP address where the database is hosted.

I then attempted to query the database to test the retrieval of data. I first edited the MySQL connector script with a query that retrieved data from the topic and exams table.

```
query = "SELECT topic.topicName, exam.examLevel FROM " +  
        "topic INNER JOIN exam ON topic._examID = exam.examID";  
command = new MySqlCommand (query, connection);  
reader = command.ExecuteReader();  
while(reader.Read()) {  
    string columnZero = reader.GetString(0);  
    string columnOne = reader.GetString(1);  
    Debug.Log(columnZero + ", " + columnOne);  
}
```



Now that I had established that my connection was capable of remotely querying my database I began to write the subroutines needed for the function of my application. I believed the most significant was the one that retrieved questions from a specific topic.

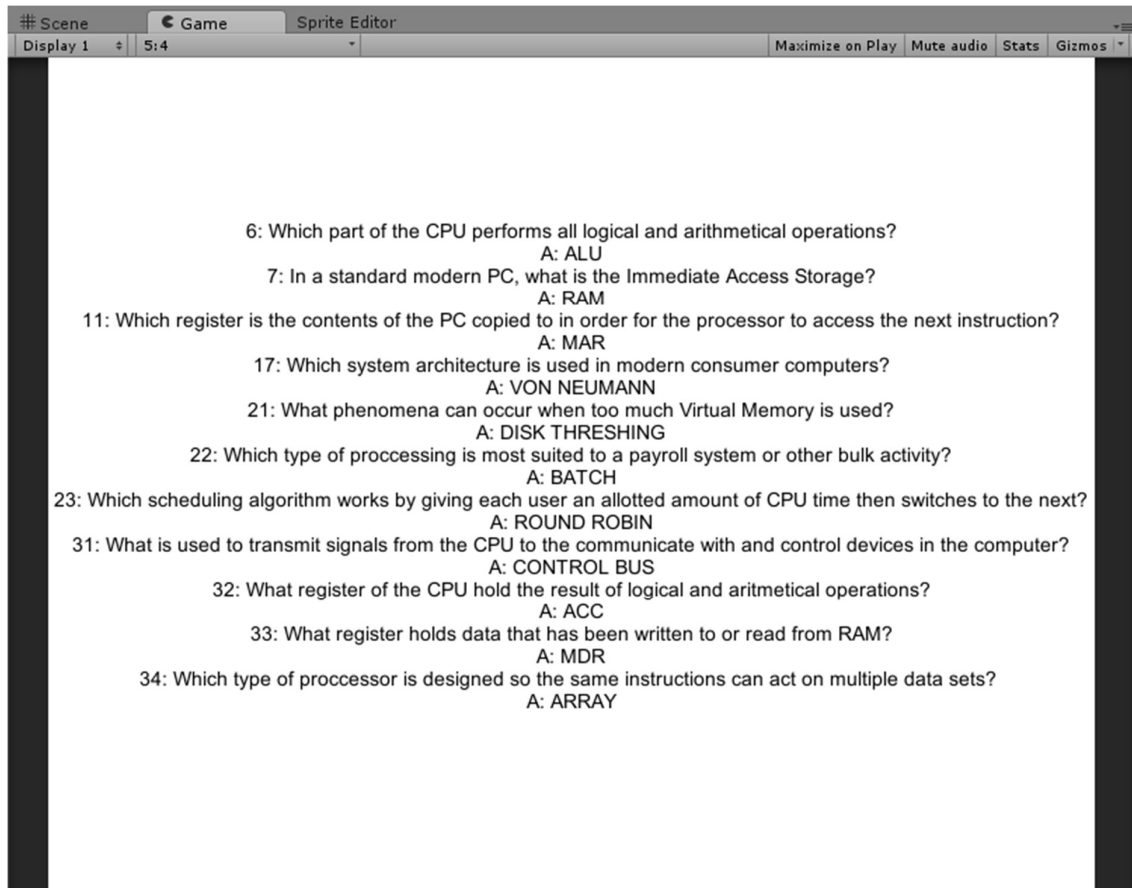
```
public string[,] FindTopicQuestions (int topicNo) {  
    query = "SELECT question.questionID, question.questionText, " +  
            "question.answer FROM question WHERE question._topicID =" + topicNo;  
    command = new MySqlCommand (query, connection);  
    reader = command.ExecuteReader();  
    int count = 0;  
    while (reader.Read ()) {  
        count++;  
    }  
    string[,] questions = new string[count,3];  
    count = 0;  
    while(reader.Read()) {  
        questions[count,0] = reader.GetString(0);  
        questions[count,1] = reader.GetString(1);  
        questions[count,2] = reader.GetString(2);  
    }  
    return questions;  
}
```

The command variable is the instruction used to manipulate the database and the reader holds the result of the query.

The function finds all the questions from a specific topic and returns them in an array made up of their ID, text and answer.

I wrote code to call this subroutine in the Quiz script and output the result to a UI text script. The code was successful.

```
questions = DatabaseScript.FindTopicQuestions (7);
questionsString = "";
Debug.Log (questions.Length);
for(int i=0;i<(questions.Length / 3);i++) {
    questionsString = questionsString + questions [i, 0] + " | ";
    questionsString = questionsString + questions [i, 1] + "\n";
    questionsString = questionsString + "A: " + questions [i, 2] + "\n";
}
text.text = questionsString;
```



After the completion of this code my application met the basic success criteria of providing computing revision questions to the user. Meeting this criteria provided a platform for me to be able to build the quiz game around.

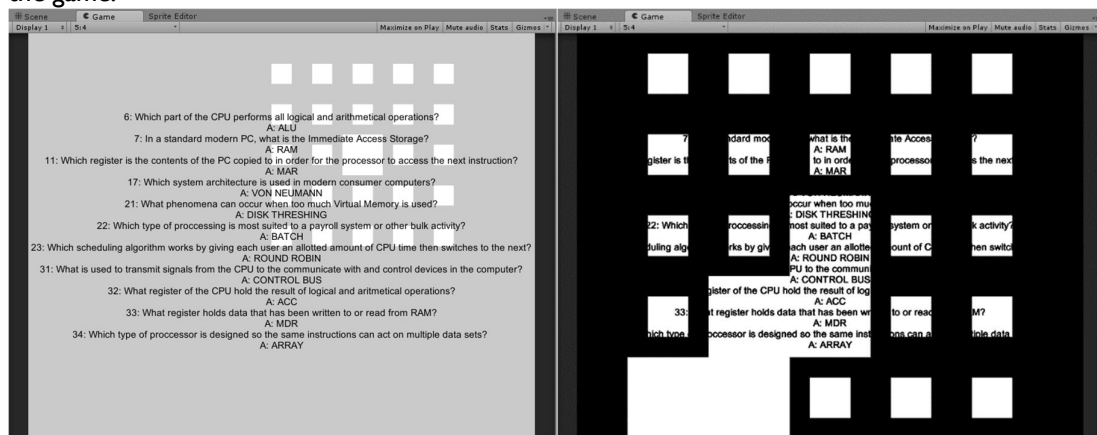
Client Feedback

I displayed to my client that the application was now able to connect to the database to present the user with a set of questions. He was satisfied with that the application showed no signs of delay or loading when connecting to the database and gave me consent to continue with development.

Phase 3: Quizzes

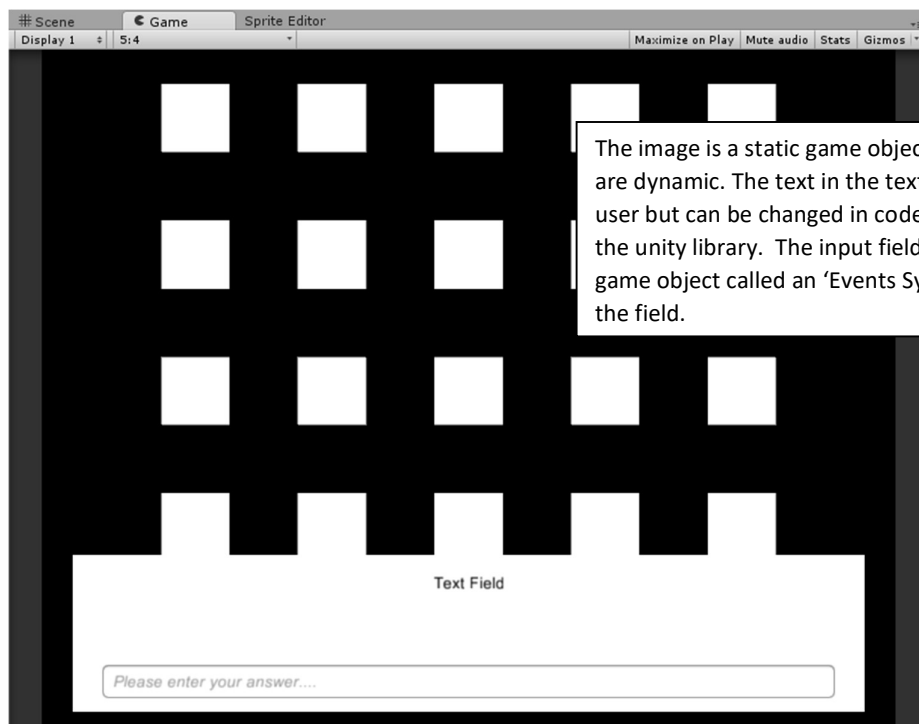
With the basic functions of my application in place I was free to develop the quiz portion of the application. Adding this functionality will make the application actually test students and is pivotal in making my project a successful revision tool.

Unfortunately my initial attempts at implementing a quiz system were met by an obstacle. When I moved between the student mode and quiz scenes game objects moved with me. In addition, when I wanted to return to the student scene it loaded a new instance of the scene and effectively restarted the game.



To deal with this problem I decided to not have separate 'Student Mode', 'Quiz' and 'Inventory' scenes and instead activate and deactivate elements in the main student mode scene as they are needed. This is not ideal as it will complicate the scene and make my system less modular, however, I am not capable of implementing any alternative where my game functions as intended.

The first step in doing this was to add UI elements to the Student Mode Scene. I initially added an input field, text box and background as these were the most basic elements needed to ask and receive answers to questions.



The next step was to write a subroutine to toggle these new UI item's as they wouldn't be needed all the time.

```
public void toggleUI (bool state) {  
    inputField.gameObject.SetActive (state);  
    questionText.gameObject.SetActive (state);  
    quizBackground.gameObject.SetActive (state);  
}
```

When an item is set to 'Not Active' it is invisible to the user and can be interacted with my user input or other game objects. This allows me to effectively hide the UI objects.

With the necessary UI elements in place I wrote the beginnings of the code that would make up the quiz subroutine. The initial version asked questions in the order they appeared in the database and told the user if they were correct or not.

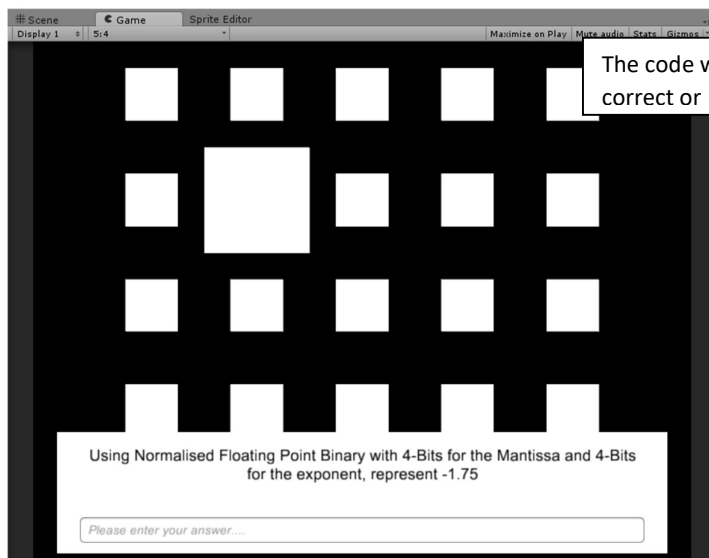
```
public IEnumerator runQuiz(int topicNumber) {  
    questions = databaseScript.findTopicQuestions (11);  
    int noQuestions = questions.Length / 3;  
    int count = 0;  
    toggleUI (true);  
    while (player.getHealth() > 0 && count < noQuestions) {  
        questionText.text = questions [count, 1];  
        enter = false;  
        while (enter == false) {  
            yield return null;  
        }  
        if (inputField.text.ToUpper() == questions [count, 2]) {  
            questionText.text = "Correct";  
        } else {  
            questionText.text = "Incorrect";  
        }  
        inputField.text = string.Empty;  
        enter = false;  
        while (enter == false) {  
            yield return null;  
        }  
        count++;  
        yield return null;  
    }  
    toggleUI (false);  
    yield return null;  
}
```

When this code was written the database wasn't fully populated with pre-built questions so the code only looked at topic 11.

At this stage the code finished looping when there were no more questions.

The 'ToUpper' command makes user answers upper case and enforces validation.

The code functioned largely as according to my pseudocode plan with minor changes necessary because of the ways unity works.



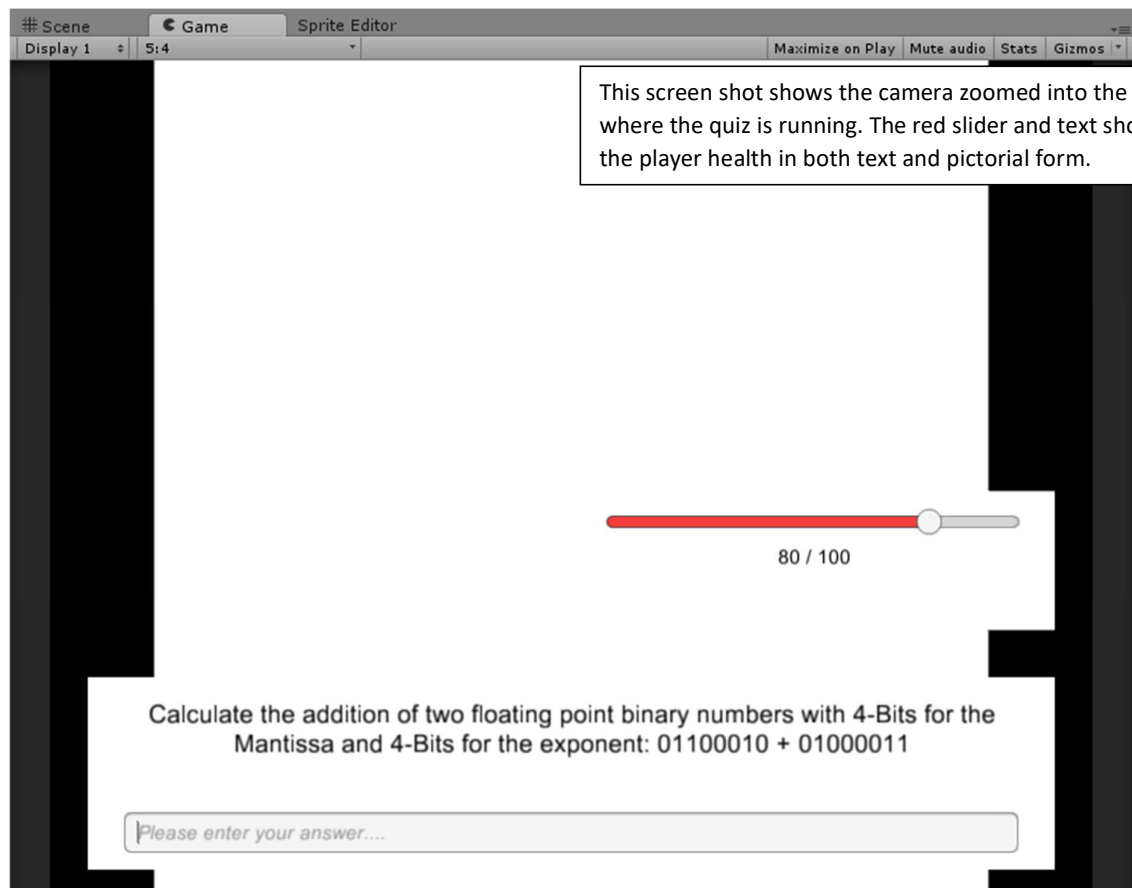
The code was able to successfully determine if an answer was correct or not and then move onto the next question.

I then added the next part of the Quiz, the player health mechanic. I changed the runQuiz subroutine to subtract health from the player when the user is incorrect and added a UI slider and text to represent this. To focus the camera on the quiz location during a quiz I also wrote zoom in and zoom out subroutines.

```
public void zoomIn(int x, int y) {  
    camera.orthographicSize = 0.5f;  
    camera.transform.position = new Vector3 (x,y,-10);  
}  
  
public void zoomOut() {  
    camera.orthographicSize = 2.5f;  
    camera.transform.position = new Vector3 (2,2,-10);  
}
```

UI elements exist on a canvas in front of the camera and therefore retain their size when the camera size changes.

The z coordinate is set to -10 so the camera can see all the game objects. If it was 0 all the game objects would be to the side of the camera and wouldn't be visible.



This screen shot shows the camera zoomed into the location where the quiz is running. The red slider and text show both the player health in both text and pictorial form.

Then, by extending the largely empty Quiz class I created earlier, I made Quiz objects inherit from the Object class so enemies could be represented by a game object in the game world.

```
class Quiz : Character {  
    private int topicNumber;  
    public Quiz (int _topicNumber, Sprite sprite, string name, Material material) : base (sprite, name, material) {  
        topicNumber = _topicNumber;  
    }  
    public int getTopicNumber () {  
        return topicNumber;  
    }  
}
```

I then gave the player and enemy game objects a material so that they could be distinguished. I stand by my decision not to include artwork for my game during the development stage, but adding colour to the character game objects was necessary in helping me understand how the game was working.

```
public Character(Sprite sprite, string name, Material material) {  
    health = 100;  
    characterSprite = sprite;  
    characterMaterial = material;  
    instance = new GameObject(name);  
    instance.AddComponent<SpriteRenderer> ();  
    instance.AddComponent<Rigidbody> ();  
    rigidbody = instance.GetComponent<Rigidbody>();  
    rigidbody.isKinematic = true;  
    rigidbody.useGravity = false;  
    spriteRenderer = instance.GetComponent<SpriteRenderer>();  
    spriteRenderer.sprite = characterSprite;  
    spriteRenderer.material = material;  
}
```

Now that the enemy game objects would actually appear on the game map I adjusted the game loop subroutine to initially put enemies on their grid location and then to move them to the side of the screen when their quiz was being run.

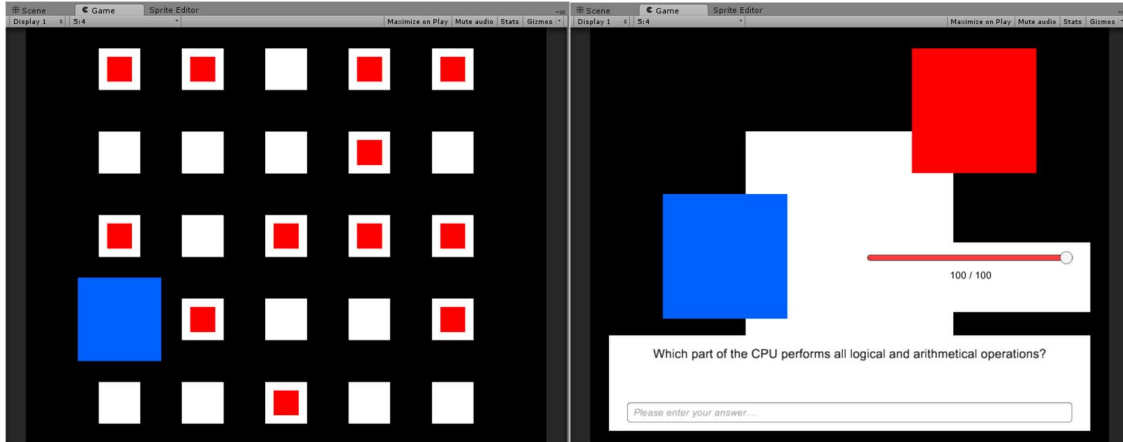
```
public IEnumerator gameLoop () {  
    player = new Player (playerSprite, "Player", playerMaterial);  
    gridPosition = new int[2] {0,0};  
    directions = new bool[4] { false, false, false, false };  
    int randomNumber;  
    for (int i=0;i<5;i++) {  
        for (int j=0;j<5;j++) {  
            randomNumber = Random.Range (1,12);  
            if (randomNumber > 6) {  
                quizArray [i, j] = new Quiz (randomNumber - 6, enemySprite, "Enemy", enemyMaterial);  
                quizArray [i, j].scaler (0.3f);  
                quizArray [i, j].move (i,j);  
            }  
            randomNumber = Random.Range (0,30);  
            if (randomNumber > 9) {  
                itemArray [i,j] = randomNumber - 10;  
            }  
        }  
    }  
    quizArray [0,0] = null;  
    while (!gameOver) {  
        int x = gridPosition [0];  
        int y = gridPosition [1];  
        if (quizArray [x,y] != null) {  
            zoomIn (x, y);  
            quizArray [x, y].scaler (0.3f);  
            quizArray [x, y].move (x+0.3f,y+0.3f);  
            player.scaler (0.3f);  
            player.move (x-0.3f,y-0.05f);  
            int topicNumber = quizArray [x, y].getTopicNumber();  
            yield return StartCoroutine (runQuiz (topicNumber));  
            zoomOut ();  
        }  
    }  
}
```

scaler() is a subroutine I added to the Character class to adjust the size of game objects. When used with the move subroutine these two lines put enemies on the map.

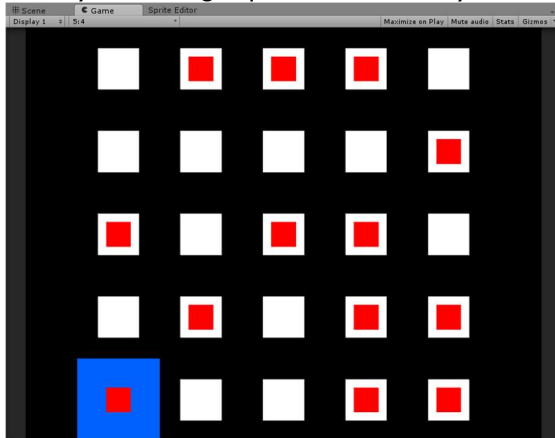
```
public void scaler(float scale) {  
    instance.transform.localScale = new Vector3 (scale, scale, 1);  
}
```

These lines arrange the player and enemy characters as they are in my design for the quizzes.

Together, these changes worked to arrange the player and enemy character as intended in my design.



However making the enemies visible exposed an error in the function of the game. Although I set the origin to always be null in the quiz array if there had been a quiz located there the enemy game object wouldn't be destroyed. To fix this problem I wrote a destroy method in the quiz class and destroyed the origin quiz when necessary.



```
public void destroy() {  
    Destroy (instance);  
}
```

```
if (quizArray [0, 0] != null) {  
    quizArray [0,0].destroy();  
    quizArray [0, 0] = null;  
}
```

Following this I added a slider and text to for the enemy. This process was largely the same as adding the player UI element.



The next stage in developing the quiz system was to store the answers in the database. However, to do this, I needed to first implement subroutines in the script to find out who the user is to associate their answers with them. The first of these subroutines checked if a player exists in the database.

```
public bool existCheck (string studentID) {  
    query = "SELECT studentID FROM student;";  
    command = new MySqlCommand (query, connection);  
    reader = command.ExecuteReader();  
    while (reader.Read ()) {  
        if (studentID == reader.GetString(0)) {  
            return true;  
        }  
    }  
    reader.Close ();  
    return false;  
}
```

I then wrote another subroutine to add a new student if they didn't already have a record in the database.

```
public void newStudent (string studentID, string firstName, string lastName, string studentClass) {  
    query = "INSERT INTO student (studentID, firstName, lastName, _classID) VALUES (" + studentID +  
        ", '" + firstName + "', '" + lastName + "', '" + studentClass + "');";  
    command = new MySqlCommand (query, connection);  
    reader = command.ExecuteReader ();  
    reader.Close ();  
    return;  
}
```

Using these two subroutines I was able to write a login procedure in the student mode script. This collected all the necessary information needed from a user and adjusted the database as needed.

```
public IEnumerator login () {
    questionText.gameObject.SetActive (true);
    inputField.gameObject.SetActive (true);
    quizBackground.gameObject.SetActive (true);
    enter = false;
    questionText.text = "Please enter you candidate number: ";
    while (enter == false) {
        yield return null;
    }
    playerID = inputField.text;
    if (databaseScript.existCheck (playerID) == false) {
        questionText.text = "You are a new user, what class are you in?: ";
        enter = false;
        while (enter == false) {
            yield return null;
        }
        string playerClass = inputField.text;
        questionText.text = "What is your first name?: ";
        enter = false;
        while (enter == false) {
            yield return null;
        }
        string playerFirstName = inputField.text;
        questionText.text = "What is your last name?: ";
        enter = false;
        while (enter == false) {
            yield return null;
        }
        string playerLastName = inputField.text;
        databaseScript.newStudent (playerID, playerFirstName, playerLastName, playerClass);
    }
    questionText.gameObject.SetActive (false);
    inputField.gameObject.SetActive (false);
    quizBackground.gameObject.SetActive (false);
}
```

The subroutine checks if the player already exists. If they do they don't need to enter any more information.

When the player is added to the database answers can now be associated with them.

Following this I could record specific instances of answers to questions in the database, associated with the student that entered them. I wrote a new subroutine in the MySQL connector script to add these answer instances and adjusted the Quiz subroutine to call it.

```
public void newAnswerInstance (string studentID, bool correct, string questionID) {
    query = "INSERT INTO answerInstance (correct, _studentID, _questionID) VALUES (" +
        correct + ", " + studentID + ", " + questionID + ")";
    command = new MySqlCommand (query, connection);
    reader = command.ExecuteReader ();
    reader.Close ();
}
```

The next stage in developing the Quiz was to make the question list reactive to student performance. The system needs to ask the most difficult questions first, as calculated by the percentage of answers to that question that are correct. The first subroutine I wrote to implement this functionality was one to calculate the difficulty in the MySQL connector script.

```
public float findDifficulty (string questionID) {  
    float correct = 0;  
    float total = 0;  
    query = "SELECT correct FROM answerinstance WHERE _questionID = "+ questionID +";";  
    command = new MySqlCommand (query, connection);  
    reader = command.ExecuteReader();  
    while (reader.Read ()) {  
        if (reader.GetString (0) == "True") {  
            correct++;  
        }  
        total++;  
    }  
    reader.Close ();  
    if (total > 0) {  
        return correct / total;  
    } else {  
        return 0;  
    }  
}
```

This subroutine takes the ID of one question at a time.

Difficulty is calculated by dividing the number of correct answers by the total number of answers. The maximum difficulty is therefore 0 and the minimum is 1. If the question has never been asked the difficulty is manually set to 0 to prevent a division error.

I then adjusted the MySQL connector subroutine so that the difficulty value was also passed into the student mode controller.

```
public string[,] findTopicQuestions (int topicNo) {  
    query = "SELECT question.questionID, question.questionText, " +  
        "question.answer FROM question WHERE question._topicID =" + topicNo;  
    command = new MySqlCommand (query, connection);  
    reader = command.ExecuteReader();  
    int count = 0;  
    while (reader.Read ()) {  
        count++;  
    }  
    string[,] questions = new string[count,4];  
    count = 0;  
    reader.Close ();  
    reader = command.ExecuteReader ();  
    while(reader.Read()) {  
        questions[count,0] = reader.GetString(0);  
        questions[count,1] = reader.GetString(1);  
        questions[count,2] = reader.GetString(2);  
        count++;  
    }  
    reader.Close ();  
    for (int i = 0; i < count; i++) {  
        questions [i, 3] = findDifficulty (questions[i,0]).ToString();  
    }  
    return questions;  
}
```

Because difficulty is calculated at this stage the difficulty of ever question is calculated even though many of them won't be used. This results in a lot of unnecessary calculation but the difficulty needs to be calculated now to prevent the order of questions changing mid-quiz and questions potentially being used twice.

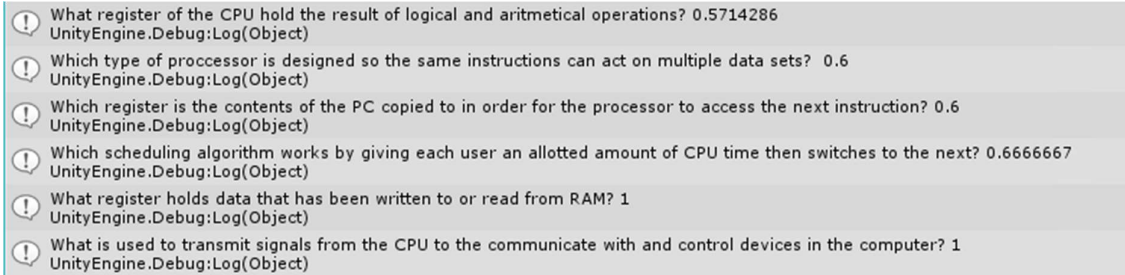
There can only be one reader per MySQL connection so the difficulty values have to be calculated after the array is populated with questions.

I then could adjust the subroutine running the quiz in the student mode controller to use these new difficulty values to adjust the order questions are asked.

```
public IEnumerator runQuiz(int topicNumber) {
    enemyHealth = 100;
    questions = databaseScript.findTopicQuestions (7);
    int noQuestions = questions.Length / 4;
    int count = 0;
    toggleUI (true);
    enemySlider.value = enemyHealth;
    enemyHealthText.text = enemyHealth.ToString() + " / 100";
    int index = 0;
    float difficulty;
    bool[] askedQuestions = new bool[noQuestions];
    while (playerHealth > 0 && count < noQuestions && enemyHealth > 0) {
        difficulty = 1;
        for (int i = 0; i < noQuestions; i++) {
            if (float.Parse(questions[i, 3]) <= difficulty && askedQuestions[i] != true) {
                Debug.Log (count);
                index = i;
                difficulty = float.Parse(questions [i, 3]);
            }
        }
        askedQuestions [index] = true;
        questionText.text = questions [index, 1];
        enter = false;
        while (enter == false) {
            yield return null;
        }
        if (inputField.text.ToUpper() == questions [index , 2]) {
            questionText.text = "Correct";
            enemyHealth = enemyHealth - 10 - playerAttack;
            enemySlider.value = enemyHealth;
            enemyHealthText.text = enemyHealth.ToString() + " / 100";
            databaseScript.newAnswerInstance (playerID, true, questions[index,0]);
        } else {
            questionText.text = "Incorrect";
            player.setHealth (playerHealth + playerDefence - 10);
            databaseScript.newAnswerInstance (playerID, false, questions[index,0]);
        }
        inputField.text = string.Empty;
        enter = false;
        while (enter == false) {
            yield return null;
        }
        count++;
        yield return null;
    }
    toggleUI (false);
    yield return null;
}
```

Questions are selected by searching for the array for the most difficult question. I've used an asked questions array so that the next most difficult question is asked on the next iteration.

To check that the order in which the questions were output was truly based on difficulty I set the quiz subroutine to output the details of each question asked to the console. The screenshot shows that the system was reacting as intended.



To develop the quiz system I had made my code only select questions from one topic to make observing changes easier. However, now that the quiz system was working as intended I changed the code to select questions properly. The first step in doing this was to write a subroutine in the MySQL connector to find if a student belongs to a GCSE or A-Level Class.

```
public bool isAlevel (string playerID) {  
    query = "SELECT student.studentID, class._examID FROM student " +  
            "INNER JOIN class ON student._classID = class.classID;";  
    command = new MySqlCommand (query, connection);  
    reader = command.ExecuteReader();  
    while (reader.Read ()) {  
        if (reader.GetString (0) == playerID) {  
            if (reader.GetString(1) == "2") {  
                reader.Close ();  
                return true;  
            }  
        }  
    }  
    reader.Close ();  
    return false;  
}
```

The subroutine uses a query with an Inner Join so it can retrieve data from the class and student tables simultaneously. Using this result it returns a Boolean value representing if the player is an A-Level Student.

The run quiz subroutine in the student mode controller could then be changed to correctly select questions.

```
if (playerIsAlevel == true) {  
    questions = databaseScript.findTopicQuestions (topicNumber + 6);  
} else {  
    questions = databaseScript.findTopicQuestions (topicNumber);  
}
```

The topics for both GCSE and A-Level are the same so retrieving questions for a A-Level student is the same as for GCSE apart from the fact that the topic ID will be 6 more.

At this stage my application now produces topic based quizzes to user that tracks their performance and adjusts based on the difficulty. My question therefore meets the most basic version of what I am intending to create. Having met this foundation I was free to add features to further gamify the quiz section to improved student enjoyment.

Client Feedback

I could then present this version of the application to my client. We discussed the state the application was in and agreed it now effectively met the success criteria to ask students questions and store their results. My Byford thought the dynamic questions list that adapted to difficulty was an effective implementation of what we discussed in the interview. After explaining why I couldn't separate the main game and quiz into separate scenes and showing Mr Byford how the quiz elements were actually displayed he permitted this as a satisfactory compromise because the use of disabling UI elements meant the system implemented was no different visually from the one we agreed on. Content that the student portion of the application had met the main educational aims of the project My Byford agreed, having seen the battle style quiz format, that I should now work on further game elements of the student mode to make the application more engaging.

Phase 4: Items

The quizzes make up the central portion of the student mode so after they were implemented I was able to add the item feature to the student mode to make the game more fun and improve student engagement.

I initially created some item names and details to enter into the database. I could then use these to test any item related code I wrote was working properly. In the interest of prioritising coding I didn't create any art for the items at the moment.

itemID	itemName	effect	_itemTypeID
NULL	Stick	1	3
NULL	Wooden Sword	2	3
NULL	Iron Sword	3	3
NULL	Steel Sword	4	3
NULL	Excalibur	10	3
NULL	Merlin's Staff	10	3
NULL	Straw Hat	1	1
NULL	Knight's Helm	5	1
NULL	Wizard's Hat	3	1
NULL	Cloth Tunic	1	2

I tried to keep the Fantasy theme in mind when I was creating the items. Hopefully this will make the game more engaging for students.

I then wrote a subroutine in the MySQL connector to retrieve this information so it could be used in the main program.

```
public string[] findItem (int itemID) {  
    string[] item = new string[3];  
    query = "SELECT * FROM item WHERE itemID =" + itemID;  
    command = new MySqlCommand (query, connection);  
    reader = command.ExecuteReader ();  
    while (reader.Read()) {  
        item [0] = reader.GetString (1);  
        item [1] = reader.GetString (2);  
        item [2] = reader.GetString (3);  
    }  
    reader.Close ();  
    return item;  
}
```

The reader should only have one entry but using the while loop prevents the code from encountering an error in the case that the data isn't returned.

With the capacity to retrieve information about items I could then write code in the student mode controller for an item class and alter the populate subroutine so items as well as quizzes are put on the map.

```
class Item : Object {  
    private int effect;  
    private int type;  
    public Item (int _effect, int _type, Sprite sprite, string  
        name, Material material) : base (sprite, name, material) {  
        effect = _effect;  
        type = _type;  
    }  
    public int getEffect () {  
        return effect;  
    }  
    public int getType () {  
        return type;  
    }  
}
```

The item class is largely the same as the Quiz class but with effect and type instead of topic number.

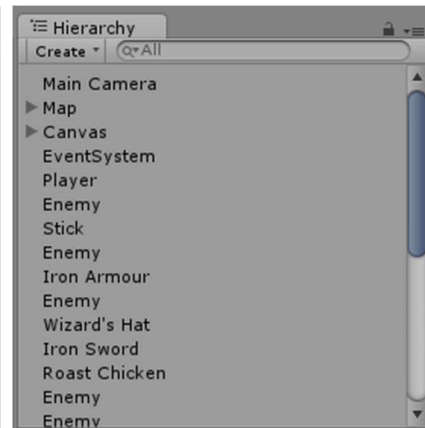
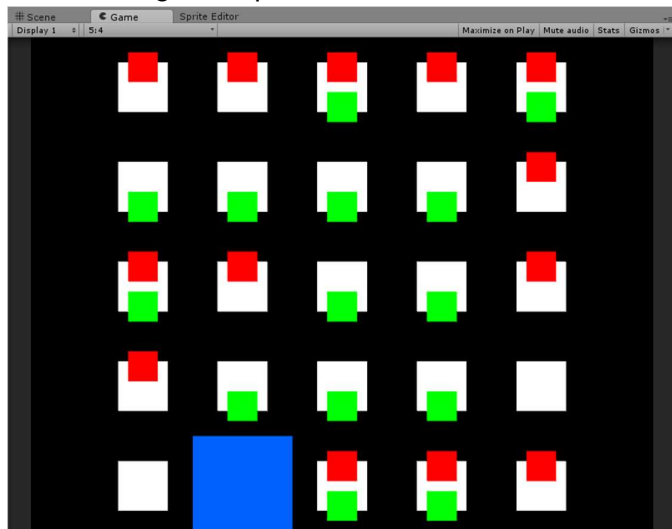
```
randomNumber = Random.Range (1,34);  
if (randomNumber > 17) {  
    string[] newItem = databaseScript.findItem (randomNumber - 17);  
    itemArray [i, j] = new Item (int.Parse(newItem[1]),  
        int.Parse(newItem[2]), itemSprite, newItem[0], itemMaterial);  
    itemArray [i, j].scaler (0.3f);  
    itemArray [i, j].move (i, j-0.2f);  
}
```

The process for population items is the same as populating quizzes.

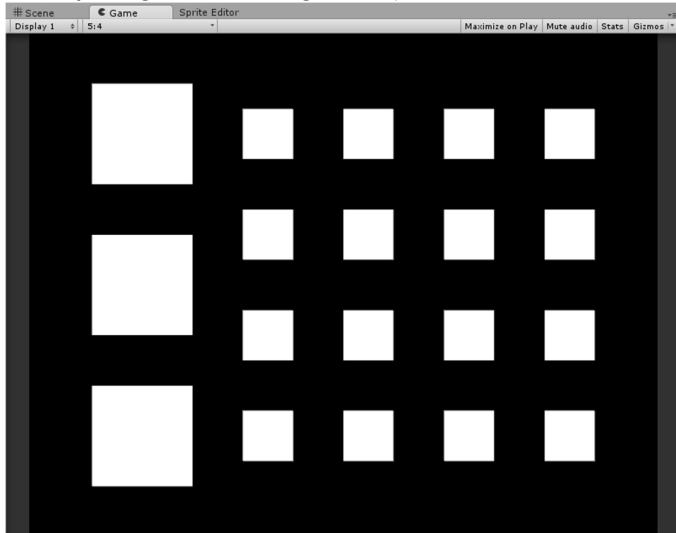
I also wrote similar code to what was written for quizzes so that items wouldn't spawn at the origin.

```
if (itemArray [x, y] != null) {  
    itemArray [x, y].destroy ();  
    itemArray [x, y] = null;  
}
```

All working together this code correctly retrieved information from the database and positioned items on the game map.



My intention was that collected items would be stored in the inventory. Similar to the problem I encountered when I was trying to implement quizzes, changing scenes to view the inventory led to a loss of data and duplicated items. I therefore decided to merge the inventory into the main student scene placing the relevant game object outside the normal view of the camera.

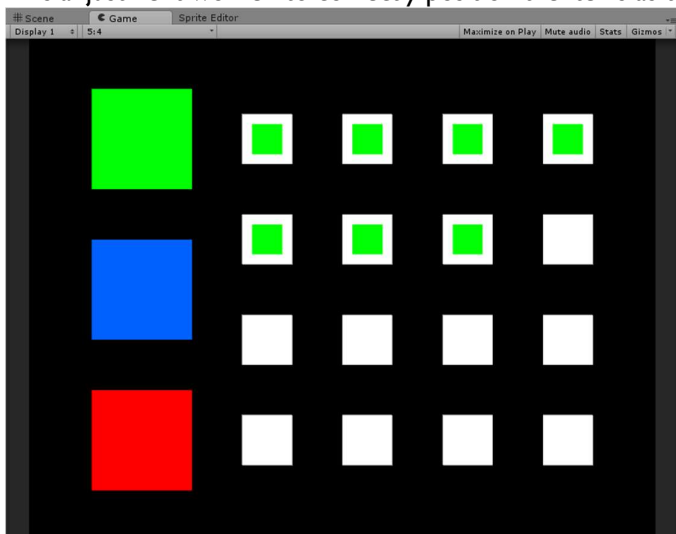


I adjusted the update subroutine so that items would be placed correctly in the inventory.

```
int count = 0;  
for (int i=0;i < inventory.Length; i++) {  
    if (inventory [i] != null) {  
        inventory [i].move (count%4 + 7.25f, 3.5f - count/4*1f);  
        count++;  
    }  
}
```

% returns the remainder from a division and / performs integer division, working together these two operators place items in a grid pattern.

This adjustment worked to correctly position the items as they were collected by the user.



To make the inventory screen accessible by the user I restructure the game loop coroutine.

```
public IEnumerator gameLoop () {  
    player = new Player (playerSprite,"Player",playerMaterial);  
    gridPosition = new int[2] {0,0};  
    directions = new bool[4] { false, false, false, false };  
    populate ();  
    yield return StartCoroutine (login());  
    while (!gameOver) {  
        int x = gridPosition [0];  
        int y = gridPosition [1];  
        if (quizArray [x,y] != null) {  
            zoomIn (x, y, 0.5f);  
            quizArray [x, y].scaler (0.3f);  
            quizArray [x, y].move (x+0.3f,y+0.3f);  
            player.scaler (0.3f);  
            player.move (x-0.3f,y-0.05f);  
            int topicNumber = quizArray [x, y].getTopicNumber();  
            yield return StartCoroutine (runQuiz (topicNumber));  
            zoomOut ();  
            quizArray [x, y].destroy ();  
            quizArray [x, y] = null;  
            player.scaler (1f);  
            player.move (x, y);  
        }  
        if (itemArray [gridPosition [0],gridPosition [1]] != null) {  
            for (int i=0; i<inventory.Length; i++) {  
                if (inventory[i] == null) {  
                    inventory [i] = itemArray [x, y];  
                    break;  
                }  
            }  
            itemArray [x, y] = null;  
        }  
        directions = new bool[4] { false, false, false, false };  
        escape = false;  
        move = false;  
        while (move == false) {  
            if (escape) {  
                zoomIn (8, 2, 2.5f);  
                yield return StartCoroutine (runInventory());  
                zoomOut ();  
                escape = false;  
            }  
            if (directions[0] || directions[1] || directions[2] || directions[3]) {  
                yield return StartCoroutine (mover ());  
            }  
            yield return null;  
        }  
    }  
    yield return null;  
}
```

Making these changes forced me to change pre-existing, pre-working code pivotal to the basic function of my system. However, because I was using a VCS I could do this with confidence that my earlier work could be easily restored.

Implementing the inventory meant I couldn't force the user to move immediately and therefore had to add this movement check.

The i key is used frequently inside quizzes so I chose to use escape to open the inventory instead.

Out of all the items in the inventory, one would be the item currently highlighted and three would be equipped. I created an array to hold the indexes of these 'important' items.

importantItems = new int[4];			
0	1	2	3
Highlighted Item	Equipped Hat	Equipped Top	Equipped Weapon

I then created an outline game object and adjusted the update subroutine so it would always follow the highlighted object.

```
int count = 0;
for (int i=0;i < inventory.Length; i++) {
    if (inventory [i] != null) {
        inventory [i].move (count%4 + 7.25f, 3.5f - count/4*1f);
        if (i == importantItems[0]) {
            itemOutline.transform.position = new Vector3 (count%4 + 7.25f, 3.5f - count/4*1f, 0f);
        }
        count++;
    }
}
```

To adjust this index I had to add to the so far blank runInventory coroutine. Largely mimicking the way the player moves, I created an inventory move coroutine and set it to be called in a loop so long as the inventory was open.

```
public IEnumerator runInventory () {
    escape = false;
    directions = new bool[4] { false, false, false, false };
    while (escape == false) {
        if (directions[0] || directions[1] || directions[2] || directions[3]) {
            move = false;
            yield return StartCoroutine (inventoryMover ());
            move = false;
            directions = new bool[4] { false, false, false, false };
        }
        yield return null;
    }
}
```

In the interest of simplicity escape both enters and exits the inventory screen.

```
public IEnumerator inventoryMover() {  
    if (directions [0] == true) {  
        int count = 0;  
        for (int i = importantItems [0]; i >= 0; i--) {  
            if (inventory [i] != null) {  
                count++;  
                if (count == 5) {  
                    importantItems [0] = i;  
                }  
            }  
        }  
    } else if (directions [1] == true) {  
        int count = 0;  
        for (int i = importantItems [0]; i < inventory.Length; i++) {  
            if (inventory [i] != null) {  
                count++;  
                if (count == 5) {  
                    importantItems [0] = i;  
                }  
            }  
        }  
    } else if (directions [2] == true && importantItems [0] > 0) {  
        int count = 0;  
        for (int i = importantItems [0]; i >= 0; i--) {  
            if (inventory [i] != null) {  
                count++;  
                if (count == 2) {  
                    importantItems [0] = i;  
                }  
            }  
        }  
    } else if (directions [3] == true) {  
        int count = 0;  
        for (int i = importantItems [0]; i < inventory.Length; i++) {  
            if (inventory [i] != null) {  
                count++;  
                if (count == 2) {  
                    importantItems [0] = i;  
                }  
            }  
        }  
    }  
    yield return null;  
}
```

The If statements aren't contained within a loop because movement inside the inventory is never necessary.

Because, once dropping items is implemented, the next item won't necessarily be held in the next index to move I have used for loops that check if there is the right number of items before or after the highlighted one for a movement to be valid.

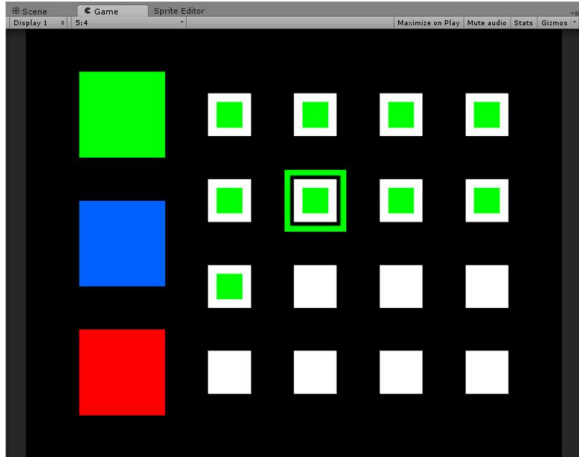
Movement up or down requires a count of 5, because rows are 4 long and the start index is counted.

	5	4	3
2	1	2	3
4	5		

Movement to the left or right only require a count of 2, representing the start and adjacent index.

	2	1	2
--	---	---	---

The combined function of the last few changes I made worked together to outline the currently highlighted index. The index responded correctly to use input to move both horizontally and vertically.



The next logical stage was to make the items equipable, usable and droppable. However, to accomplish this I thought it would be useful if items could be distinguished from each other and therefore that it was an appropriate time to create and implement the artwork for items. So that I only had to go through the process of implementing artwork once, I created all the necessary designs at the same time.

To make the best of my limited artistic skill, I used an online application called Pickxel that creates pixel based artwork. This allowed me to create designs simply and quickly so I could focus my time on the actual coding.

To add the artwork to the games I needed to the structure of the Student Mode Controller. The first step in doing this was to edit the variables passed to the script.

```
public Sprite playerSprite;  
  
public Sprite[] enemySprites;  
  
public Sprite[] itemSprites;  
public GameObject itemOutlinePrefab;
```

Items would now be distinguished by their design, there was no further need for materials to be assigned to objects.

There would now be more than one option for the sprite of an enemy or item and therefore the singular variables had to be replaced with arrays.

The 'Populate' subroutine then had to be changed so items and quizzes could be created in the correct way.

```
randomNumber = Random.Range (1,12);  
if (randomNumber > 6) {  
    int topic = randomNumber - 6;  
    randomNumber = Random.Range (0,enemySprites.Length);  
    quizArray [i, j] = new Quiz (topic, enemySprites[randomNumber], "Enemy");  
    quizArray [i, j].scaler (0.6f);  
    quizArray [i, j].move (i,j+0.2f);  
}  
randomNumber = Random.Range (1,34);  
if (randomNumber > 17) {  
    string[] newItem = databaseScript.findItem (randomNumber - 17);  
    Sprite itemSprite = null;  
    for (int k=0; k<itemSprites.Length; k++) {  
        if (itemSprites [k].name == newItem [0] + "_0") {  
            itemSprite = itemSprites [k];  
        }  
    }  
    itemArray [i, j] = new Item (int.Parse(newItem[1]), int.Parse(newItem[2]), itemSprite, newItem[0]);  
    itemArray [i, j].scaler (0.6f);  
    itemArray [i, j].move (i, j-0.2f);  
}
```

Enemies are not associated with a specific quiz so can be chosen randomly.

I changed the Object class so that it no longer had material as an attribute. This change affected all derived classes, including Quiz and Item.

Specific Items have specific sprites so this for loop is needed to find the right one.

With these changes implemented into the code, I could now add in the sprites I created to the game engine. The program used the sprites in the correct way, however, there were some problems with the way they were displayed.



Sprites were far too small, the colours in the image were not quite right and sprites were displayed slightly off centre. When I manually enlarged the sprites in the game world, they became blurry.

These problems arose because of the way my sprites were imported

The image shows the 'Inspector' window in a game engine, specifically the 'Player Import Settings' panel. The panel includes the following settings: Texture Type: Sprite (2D and UI); Sprite Mode: Single; Packing Tag: (empty); Pixels Per Unit: 100; Pivot: Center; Generate Mip Maps: checked; Filter Mode: Bilinear; Max Size: 2048; Format: Compressed. Four callout boxes with arrows point to these settings: 1. Points to the 'Open' button: 'The origin of the transform of a sprite is the centre of the image, however, my sprites were exported as 32x32 pixel squares so the centre of the sprite was outside of my design. This caused the images to appear slightly askew.' 2. Points to 'Pixels Per Unit: 100': 'The pixels per unit value controls how many pixel of an imported sprite make up 1 in-game unit. The squares that make up my map are 1 unit apart so the imported sprite would be far too small.' 3. Points to 'Filter Mode: Bilinear': 'The Bilinear filter mode blends together different colours of sprites to try and avoid blocky-ness and pixilation. This would be useful for detailed sprites, but made my pixel based sprites useless.' 4. Points to 'Format: Compressed': 'My sprites were set to the compressed import setting. This meant that, in the interest of reducing their file size, some of the information about my sprites was removed. However, some of this information was about the way the sprites were coloured; the compression resulted in the sprite appearing in incorrect colours.'

After analysing the sprite import settings I was able to rectify the issues.



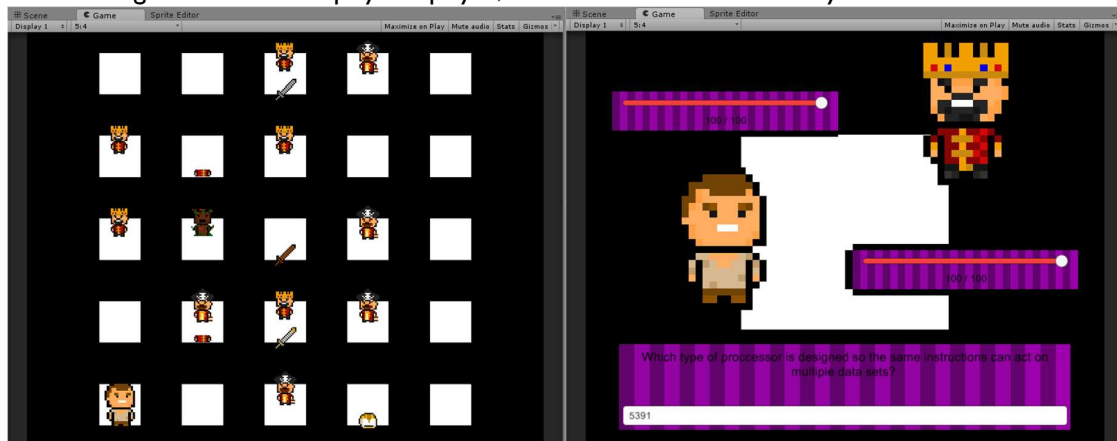
The issue of centring the sprite transform had the odd solution of setting the sprite mode to multiple. This treated the singular sprite as a sprite sheet to be partitioned. Using this feature I could clearly define the area that contained my design and simply didn't define any other sprites.

After some testing, 35 pixels per unit appeared give the best base size for sprites in the game world.

The Point filter mode was best suited to my blocky design and gave my sprite the definition they needed.

The true colour format retained the information needed to properly colour my sprites.

These changes worked to display the player, items and enemies in the way I wanted.



With all the moving items given their artwork I turned my attention to doing the same with the static game objects. Changing the background was a simple process of adding a new game object with a sprite renderer but I wanted map location to change depending on what was also at the same grid location. To map the map change I first had to pass allow my code to access it.

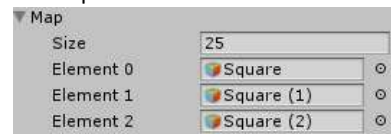
```
public Sprite playerSprite;

public Sprite[] mapSprites;
public GameObject[] map;

public Sprite[] enemySprites;

public Sprite[] itemSprites;
public GameObject itemOutlinePrefab;
```

There was no need to create new map game objects at the start of each game, so instead I passed the existing map to the script.



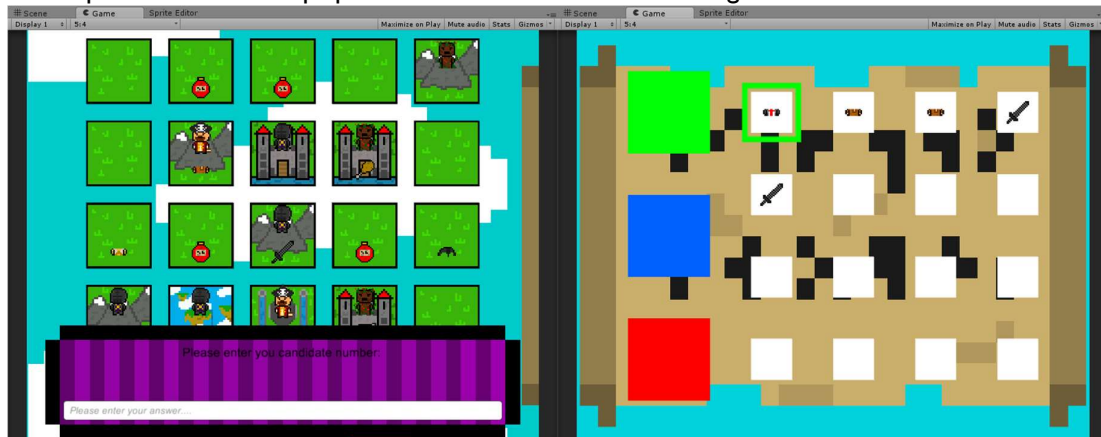
In the game inspector, I set all map locations to have a default sprite of an empty filed and filled the map sprites array with designs that had an actual location on. I then changed the populate subroutine to populate the map with these designs at the position of quizzes.

```
randomNumber = Random.Range (1,12);  
if (randomNumber > 6) {  
    int topic = randomNumber - 6;  
    randomNumber = Random.Range (0,enemySprites.Length);  
    quizArray [i, j] = new Quiz (topic, enemySprites[randomNumber], "Enemy");  
    quizArray [i, j].scaler (0.6f);  
    quizArray [i, j].move (i,j+0.2f);  
    randomNumber = Random.Range (0, mapSprites.Length);  
    SpriteRenderer sRenderer = map[i+j*5].GetComponent <SpriteRenderer> ();  
    sRenderer.sprite = mapSprites [randomNumber];  
}
```

This section of code only runs when a quiz is placed on the map.

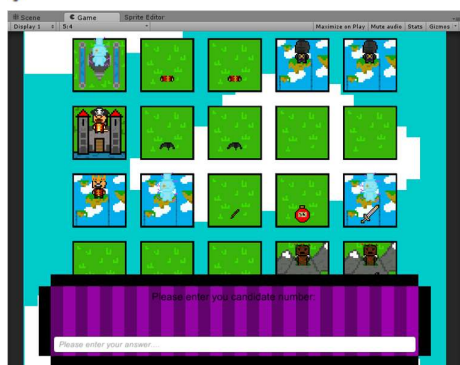
Quizzes are unrelated to map sprites, so are placed randomly.

The implementation of map sprites worked and had the following result:



There was a slight overlap between the main map and inventory screens so I had to adjust the position of the inventory elements slightly. I moved the inventory game objects one unit to right and adjusted movement positions in the code so items would still be positioned in the correct place.

```
if (inventory [i] != null) {  
    inventory [i].move (count%4 + 8.25f, 3.5f - count/4*1f);  
    if (i == importantItems[0]) {  
        itemOutline.transform.position = new Vector3 (count%4 + 8.25f, 3.5f - count/4*1f, 0f);  
    }  
    count++;  
}
```



With the artwork in place, some of the text became difficult to read. Knowing from my research that this could be solved by improving the contrast, I decided to change all the text colours to white. The white text was far more legible.

Now that I could easily distinguish between the different items in use the task of giving items some function seemed far more manageable. The first step in doing this was to write a subroutine to adjust the outlined index when an item is removed from the main inventory array.

```
public int inventoryFoucusAdjust (int current) {  
    for (int i = current; i < inventory.Length; i++) {  
        if (inventory[i] != null && i != current) {  
            return i;  
        }  
    }  
    for (int i = current; i > 0; i--) {  
        if (inventory [i] != null && i != current) {  
            return i;  
        }  
    }  
    return 0;  
}
```

When an item is removed from the inventory the 'focus' index (where the outline is positioned) needs to be changed so the system doesn't try to access an empty index.

The subroutine first looks after the targeted array to see if there is an item which will push up to fill the same space and therefore not make the outline move. If an item isn't found after, the indexes before are checked. If still no item is found the focus index is set to the blank element at index 0, much like at the start of the game.

I then used this new tool in the in the 'runInventory' coroutine to delete, or 'drop', items.

```
public IEnumerator runInventory () {  
    escape = false;  
    enter = false;  
    space = false;  
    directions = new bool[4] { false, false, false, false };  
    while (escape == false) {  
        if (directions[0] || directions[1] || directions[2] || directions[3]) {  
            move = false;  
            yield return StartCoroutine (inventoryMover ());  
            move = false;  
            directions = new bool[4] { false, false, false, false };  
        }  
        if (space) {  
            inventory [importantItems [0]].destroy ();  
            inventory [importantItems [0]] = null;  
            importantItems [0] = inventoryFoucusAdjust (importantItems [0]);  
            space = false;  
        }  
        yield return null;  
    }  
}
```

The index in the array has to be both destroyed and set to null so both the physical game object, and the object in code cease to exist.

To implement equipable items I first had to change the Player class to have the new attributed needed for displaying the hat, top and weapon.

```
class Player : Object {  
    private int health;  
    private int attack;  
    private int defence;  
    private Sprite hatSprite;  
    private Sprite topSprite;  
    private Sprite weaponSprite;  
    private SpriteRenderer hatRenderer;  
    private SpriteRenderer topRenderer;  
    private SpriteRenderer weaponRenderer;  
    private GameObject hat;  
    private GameObject top;  
    private GameObject weapon;
```

Unfortunately, one game object can't have multiple sprite renderers. This meant that the three types of equipable item require a game object, a sprite and a sprite renderer each.

I then wrote new methods to change these attributes.

```
public void setHat (Sprite sprite) {  
    hatRenderer.sprite = sprite;  
}  
public void setTop (Sprite sprite) {  
    topRenderer.sprite = sprite;  
}  
public void setWeapon (Sprite sprite) {  
    weaponRenderer.sprite = sprite;  
}
```

These methods are necessary to maintain encapsulation in the player class.

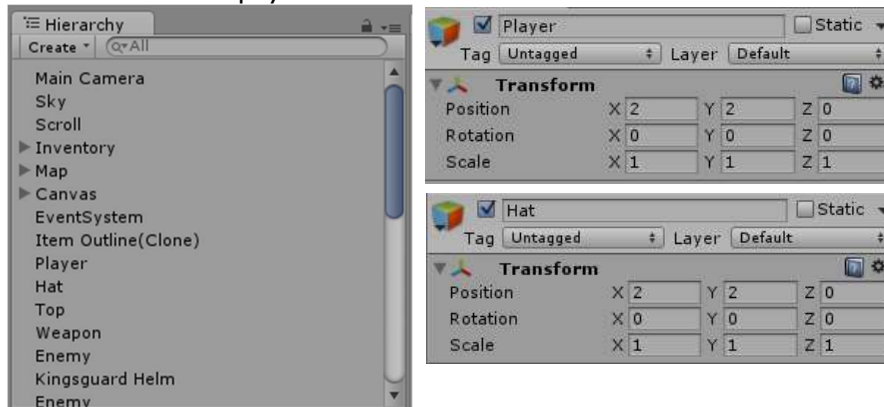
To make these new game objects move with the main player object I had to extend the move subroutine.

```
public void move (float x, float y) {  
    base.move (x,y);  
    hat.transform.position = new Vector3 (x, y, 0f);  
    top.transform.position = new Vector3 (x, y, 0f);  
    weapon.transform.position = new Vector3 (x, y, 0f);  
}
```

This line calls the original move method from the base class.

These lines of code are then run to extend the base method.

These screen shots of the game engine inspector shows the game objects were successfully created and moved with the player.



Because equipped items may return to the inventory at some point, the item object, not just the sprite needed to be stored. I created a new array, of type item, to hold what is currently equipped.

```
private Item[,] itemArray;  
private Item[] inventory;  
private GameObject itemOutline;  
private int[] importantItems;  
private Item[] equipped;
```

I then changed the update function to position these equipped items in the three boxes on the inventory screen.

```
for (int i = 0; i < equipped.Length; i++) {  
    if (equipped [i] != null) {  
        equipped [i].move (7f, 3.5f - i * 1.5f);  
    }  
}
```

With the positioning in place, I adjusted the runInventory coroutine to deal with equipping items.

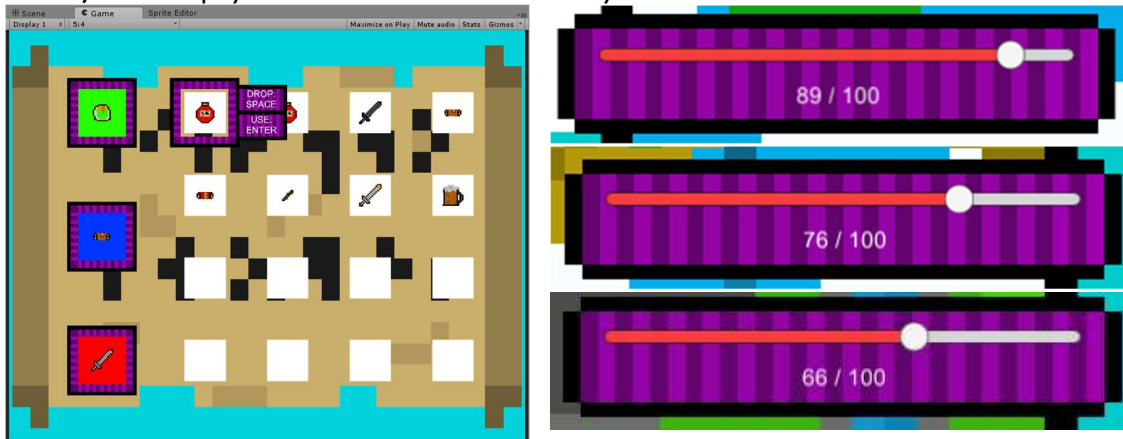
```
if (enter) {  
  if (importantItems[0] != null) {  
    if (inventory [importantItems [0]].getType () == 0) {  
      player.setHealth (playerHealth + inventory [importantItems [0]].getEffect ());  
      inventory [importantItems [0]].destroy ();  
    } else if (inventory [importantItems [0]].getType () == 1) {  
      equipped [0] = inventory [importantItems [0]];  
    } else if (inventory [importantItems [0]].getType () == 2) {  
      equipped [1] = inventory [importantItems [0]];  
      player.setDefence (inventory[importantItems[0]].getEffect());  
    } else if (inventory [importantItems [0]].getType () == 3) {  
      equipped [2] = inventory [importantItems [0]];  
      player.setAttack (inventory[importantItems[0]].getEffect());  
    }  
    inventory [importantItems [0]] = null;  
    importantItems [0] = inventoryFoucusAdjust (importantItems [0]);  
  }  
  enter = false;  
}
```

As per the instructions displayed on the inventory screen the enter button is used to equip items.

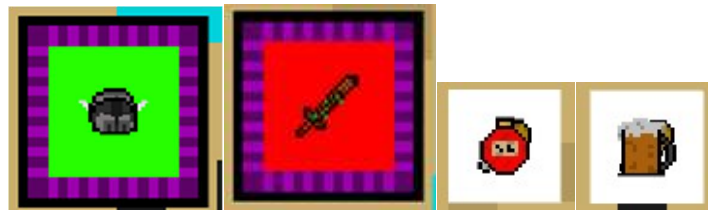
Usable game objects are destroyed but hats, tops and weapons are passed to the equipped array so they can be moved into the right position.

In all cases, an item is removed from the main inventory array. The focus adjust subroutine must be used to properly correct the outline.

This code worked to correctly put hats, tops and weapons in the correct equipped slot. The code also adjusted the players attack and defence accurately.



However, this there were problems with this solution to equipping items. Firstly usable items, such as health potions and food, weren't dealt with correctly; the player's health was unchanged and the item game object wasn't destroyed. Furthermore, this implementation didn't account for the scenario when an object was already equipped; the newly equipped items simply stacked on top of existing ones.



Upon examining the code I found that the reason usable items weren't handled correctly was the criteria of the if statement. I wrote the statement with the impression that usables were represented as type 0 whereas in reality they were type 4. Changing the criteria solved the problem and usable items worked as intended.

```

if (enter) {
    if (importantItems[0] != null) {
        if (inventory [importantItems [0]].getType () == 4) {
            player.setHealth (playerHealth + inventory [importantItems [0]].getEffect ());
            inventory [importantItems [0]].destroy ();
        } else if (inventory [importantItems [0]].getType () == 1) {
            equipped [0] = inventory [importantItems [0]];
        } else if (inventory [importantItems [0]].getType () == 2) {
            equipped [1] = inventory [importantItems [0]];
            player.setDefence (inventory[importantItems[0]].getEffect());
        } else if (inventory [importantItems [0]].getType () == 3) {
            equipped [2] = inventory [importantItems [0]];
            player.setAttack (inventory[importantItems[0]].getEffect());
        }
        inventory [importantItems [0]] = null;
        importantItems [0] = inventoryFoucusAdjust (importantItems [0]);
    }
    enter = false;
}

```

itemTypeID	itemName
1	Hat
2	Top
3	Weapon
4	Usable
NULL	NULL

I then changed update the subroutine to swap out the equipped item properly when there is already one equipped.

```

if (enter) {
    if (importantItems[0] != null) {
        if (inventory [importantItems [0]].getType () == 4) {
            player.setHealth (playerHealth + inventory [importantItems [0]].getEffect ());
            inventory [importantItems [0]].destroy ();
            inventory [importantItems [0]] = null;
        } else if (inventory [importantItems [0]].getType () == 1) {
            Item temp = equipped [0];
            equipped [0] = inventory [importantItems [0]];
            inventory[importantItems [0]] = temp;
        } else if (inventory [importantItems [0]].getType () == 2) {
            Item temp = equipped [1];
            equipped [1] = inventory [importantItems [0]];
            player.setDefence (inventory[importantItems[0]].getEffect());
            inventory [importantItems [0]] = temp;
        } else if (inventory [importantItems [0]].getType () == 3) {
            Item temp = equipped [2];
            equipped [2] = inventory [importantItems [0]];
            player.setAttack (inventory[importantItems[0]].getEffect());
            inventory [importantItems [0]] = temp;
        }
        if (importantItems != null) {
            importantItems [0] = inventoryFoucusAdjust (importantItems [0]);
        }
    }
    enter = false;
}

```

The usable items are now the only items destroyed.

When an item is equipped the index it came from is swapped with the equipped location.

If the equipped index happens to be empty the inventory index will just be set to null.

If the inventory index is set to null the outline index needs to be adjusted like if an item was deleted.

I then noticed there were some items that I had created art for and added into the game engine that never appeared in the game world. This was due to the fact that the range of item ID's I had defined in the populate array was no longer correct. To ensure this didn't happen again I wrote a subroutine in the MySQL connector script to find the number of items.


```
public int findNumberOfItems () {  
    query = "SELECT itemName FROM item;";  
    command = new MySqlCommand (query, connection);  
    reader = command.ExecuteReader();  
    int count = 0;  
    while (reader.Read ()) {  
        count++;  
    }  
    reader.Close ();  
    return count;  
}
```

This subroutine queries the database for the itemName of every item and simply increments a count for every result found.

I could then adjust the populate array to use this new subroutine.

```
int numberOfItems = databaseScript.findNumberOfItems ();  
randomNumber = Random.Range (1,numberOfItems*2);  
if (randomNumber > numberOfItems) {  
    string[] newItem = databaseScript.findItem (randomNumber - numberOfItems);  
    Sprite itemSprite = null;  
    for (int k=0; k<itemSprites.Length; k++) {  
        if (itemSprites [k].name == newItem [0] + "_0") {  
            itemSprite = itemSprites [k];  
        }  
    }  
    itemArray [i, j] = new Item (int.Parse(newItem[1]), int.Parse(newItem[2]), itemSprite, newItem[0]);  
    itemArray [i, j].scaler (0.6f);  
    itemArray [i, j].move (i, j-0.2f);  
}
```

The range is still double the number of items so the probability of an item appearing is maintained.

With this change implemented the system should react to any future changes to the database.

Indeed, the previously absent items started to appear in the game.



I then worked to make the items appear on the player game object, as if they had actually put them on. The version of the item on the player would just act as a cosmetic change, not holding any of the actual item data. To that end I added a get sprite subroutine to the Object class.

```
public void move(float x, float y) {  
    rigidbody.position = new Vector3 (x, y, 0f);  
}  
public void toggleActive(bool state) {  
    instance.SetActive (state);  
}  
public void scaler(float scale) {  
    instance.transform.localScale = new Vector3 (scale, scale, 1);  
}  
public void destroy() {  
    Destroy (instance);  
}  
public Sprite getSprite() {  
    return objectSprite;  
}
```

Although it will only be used by the Item class, this subroutine had to be written in the Object class because the sprite attribute is private and therefore can't be accessed by a method unique to the Item class.

Using this new subroutine I again changed the run inventory coroutine to adjust the sprites attached to the player.

```
if (importantItems[0] != null) {  
    if (inventory [importantItems [0]].getType () == 4) {  
        player.setHealth (playerHealth + inventory [importantItems [0]].getEffect ());  
        inventory [importantItems [0]].destroy ();  
        inventory [importantItems [0]] = null;  
    } else if (inventory [importantItems [0]].getType () == 1) {  
        Item temp = equipped [0];  
        equipped [0] = inventory [importantItems [0]];  
        inventory[importantItems [0]] = temp;  
        player.setHat (equipped[0].getSprite());  
    } else if (inventory [importantItems [0]].getType () == 2) {  
        Item temp = equipped [1];  
        equipped [1] = inventory [importantItems [0]];  
        player.setDefence (inventory[importantItems[0]].getEffect());  
        inventory [importantItems [0]] = temp;  
        player.setTop (equipped[1].getSprite());  
    } else if (inventory [importantItems [0]].getType () == 3) {  
        Item temp = equipped [2];  
        equipped [2] = inventory [importantItems [0]];  
        player.setAttack (inventory[importantItems[0]].getEffect());  
        inventory [importantItems [0]] = temp;  
        player.setWeapon (equipped [2].getSprite ());  
    }  
    if (importantItems != null) {  
        importantItems [0] = inventoryFoucusAdjust (importantItems [0]);  
    }  
}  
enter = false;
```

The player sprite only needs to be changed when an item is equipped. Changing them via the update function would accomplish the same task but be less efficient.

At no point is there a need to check to contents of the player item sprites. This allows me to write setters, but not getters, for the attributes.

The code placed the sprite on the player but there were several problems in the way they were displayed.



The first problems was that the sprites were placed behind the player. To fix this I had to create a new sorting layer.



Sorting layers control how sprites are displayed when two overlap. The sprite in the lower layer is shown and the other covered.

Items are by default put on the default layer, by putting the equipped items on a layer in front of default they will definitely be placed in front of the

By applying this sorting layer to the equipped items the first display problem was solved.

```
hat = new GameObject("Hat");  
hat.AddComponent<SpriteRenderer> ();  
hatRenderer = hat.GetComponent<SpriteRenderer>();  
hatRenderer.sortingLayerName = "Equipped Items";
```

```
top = new GameObject("Top");  
top.AddComponent<SpriteRenderer> ();  
topRenderer = top.GetComponent<SpriteRenderer>();  
topRenderer.sortingLayerName = "Equipped Items";
```

```
weapon = new GameObject("Weapon");  
weapon.AddComponent<SpriteRenderer> ();  
weaponRenderer = weapon.GetComponent<SpriteRenderer>();  
weaponRenderer.sortingLayerName = "Equipped Items";
```



The next issue was that the items were also positioned at the same coordinate as the player and therefore not in the correct place on the character's body. In examining this problem I had the idea that instead of moving each item individually, it would be easier to store all the items, with the correct transform offsets, within an empty parent game object.

```
class Player : Object {  
    private int health;  
    private int attack;  
    private int defence;  
    private Sprite hatSprite;  
    private Sprite topSprite;  
    private Sprite weaponSprite;  
    private SpriteRenderer hatRenderer;  
    private SpriteRenderer topRenderer;  
    private SpriteRenderer weaponRenderer;  
    private GameObject equipped;  
    private GameObject hat;  
    private GameObject top;  
    private GameObject weapon;  
    public Player (Sprite sprite, string name) : base(sprite, name) {  
        health = 100;  
        attack = 0;  
        defence = 0;  
        equipped = new GameObject("Equipped");  
        hat = new GameObject("Hat");  
        hat.AddComponent<SpriteRenderer> ();  
        hatRenderer = hat.GetComponent<SpriteRenderer>();  
        hatRenderer.sortingLayerName = "Equipped Items";  
        hat.transform.parent = equipped.transform;  
        hat.transform.position = new Vector3 (0, 0.11f, 0f);  
        top = new GameObject("Top");  
        top.AddComponent<SpriteRenderer> ();  
        topRenderer = top.GetComponent<SpriteRenderer>();  
        topRenderer.sortingLayerName = "Equipped Items";  
        top.transform.parent = equipped.transform;  
        top.transform.position = new Vector3 (0.015f, -0.13f, 0f);  
        weapon = new GameObject("Weapon");  
        weapon.AddComponent<SpriteRenderer> ();  
        weaponRenderer = weapon.GetComponent<SpriteRenderer>();  
        weaponRenderer.sortingLayerName = "Equipped Items";  
        weapon.transform.parent = equipped.transform;  
        weapon.transform.position = new Vector3 (0.19f, -0.1f, 0f);  
    }  
    public void move (float x, float y) {  
        base.move (x,y);  
        equipped.transform.position = new Vector3 (x,y,0f);  
    }  
    public void scaler (float scale) {  
        base.scaler (scale);  
        equipped.transform.localScale = new Vector3 (scale, scale, 1f);  
    }  
}
```

By setting the parent of the transform of the equipped items to the new 'equipped' game object their transforms can all be controlled via the parent transform. The transforms of the individual items works as an offset from the parent.

I set the offsets of all the items to correctly position them on the character safe in the knowledge it would always be accurate as long as the parent was moved to the right place.

The move and scalar functions could then be changed, now only having to alter the transform of the parent game object.

The changes made to the player class positioned the equipped items correctly on the player's body.



However, the weapon was too large and positioned behind the top and hat sprites. To fix this I had to adjust the scale and sorting order of the weapon.

```
weaponRenderer.sortingOrder = 1;  
weapon.transform.localScale = new Vector3 (0.6f,0.6f,1f);
```

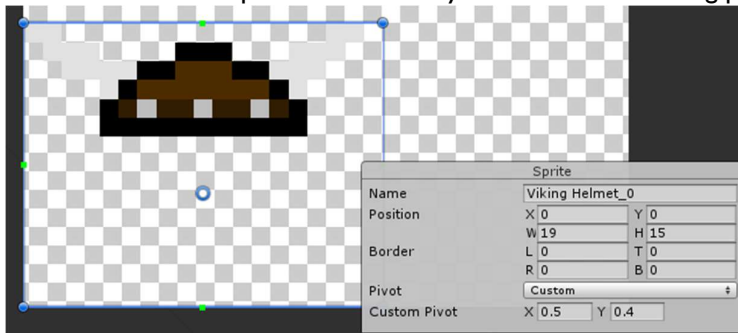
Sorting order determines the position of an item for displaying within a sorting layer. Because the hats and top don't overlap they can both remain set to 0. The weapon, however, needs to be set to 1 to put it in front. As discussed before, scale changes the size of a game object.



For the majority of cases this worked, positioning all the items correctly on the player. However, if a hat that wasn't the same shape as the player's head, as a few were, the item would look wrong.



To fix this I had to adjust the boundaries and pivot point of any irregularly shaped sprites. Some trial and error with different positions eventually resulted in them being positioned correctly.



With this final correction the items were successfully implemented. My application creates and distributes a set of items, along with the quizzes on the game world. These items can be collected, stored and equipped where they then affect the mechanics of the quizzes. With this phase finished I had implemented all the mechanics for the game and was free to go about writing the code that manages ending the game.

Client Feedback

When I presented this version of the application to my end client he agreed that the items made the game more engaging and promoted extended play. He commented that the art style we had agreed on had also been effectively implemented and he was happy I hadn't chosen a difficult art style that would have taken time away with development. Along a similar line, my client also expressed that he thought, given the application is primarily a revision, the system had enough game mechanics purely in the interest of student enjoyment. I assured him that this was the full extent of 'gamification' I intended to implement and would now be moving on to developing how the game progresses and eventually ends. We agreed that I should take care to promote extended play and competitiveness in the following phase to encourage repeated use.

Phase 5: Ending the game

At this stage the game has all the features I set out to implement and the only thing left to do to the student mode is deal with ending it.

I believe the game will work best if it takes a 'roguelike' approach to ending. Roguelike is a subgenre of gaming where the levels are procedurally generated, as they are in my game, and the game continues to infinitely provide the player with more and more levels until they fail. This type of end game is suited to my project; whilst having an end goal for student to achieve might motivate them initially, once they achieve it I believe they would lack motivation to return to the application. On the other hand a rogue like encourages replay.

Currently, one the player dies or all the quizzes are completed, there is nothing to do and the game effectively ends. To give the game the ability to carry on I modified the populate subroutine.

```
public void populate () {
    int randomNumber;
    int x = gridPosition [0];
    int y = gridPosition [1];
    SpriteRenderer sRenderer;
    mapQuizzes = 0;
    mapItems = 0;
    for (int i=0;i<5;i++) {
        for (int j=0;j<5;j++) {
            randomNumber = Random.Range (1,12);
            if (randomNumber > 6) {
                int topic = randomNumber - 6;
                randomNumber = Random.Range (0, enemySprites.Length);
                quizArray [i, j] = new Quiz (topic, enemySprites [randomNumber], "Enemy");
                quizArray [i, j].scaler (0.6f);
                quizArray [i, j].move (i, j + 0.2f);
                randomNumber = Random.Range (0, mapSprites.Length);
                sRenderer = map [i + j * 5].GetComponent <SpriteRenderer> ();
                sRenderer.sprite = mapSprites [randomNumber];
                mapQuizzes++;
            } else {
                sRenderer = map [i + j * 5].GetComponent <SpriteRenderer> ();
                sRenderer.sprite = emptySprite;
            }
        }
        int numberOfItems = databaseScript.findNumberOfItems ();
        randomNumber = Random.Range (1,numberOfItems*2);
        if (randomNumber > numberOfItems) {
            string[] newItem = databaseScript.findItem (randomNumber - numberOfItems);
            Sprite itemSprite = null;
            for (int k=0; k<itemSprites.Length; k++) {
                if (itemSprites [k].name == newItem [0] + "_0") {
                    itemSprite = itemSprites [k];
                }
            }
            itemArray [i, j] = new Item (int.Parse(newItem[1]), int.Parse(newItem[2]), itemSprite, newItem[0]);
            itemArray [i, j].scaler (0.6f);
            itemArray [i, j].move (i, j-0.2f);
            mapItems++;
        }
    }
}
if (quizArray [x, y] != null) {
    quizArray [x, y].destroy ();
    quizArray [x, y] = null;
    mapQuizzes--;
}
if (itemArray [x, y] != null) {
    itemArray [x, y].destroy ();
    itemArray [x, y] = null;
    mapItems--;
}
randomNumber = Random.Range (0, mapSprites.Length);
sRenderer = map[x + y * 5].GetComponent<SpriteRenderer>();
sRenderer.sprite = mapSprites [randomNumber];
}
```

The player's position is now calculated. This means that the subroutine can still perform its function even if the player isn't at the origin.

Whenever a quiz or an item is placed the relevant variable is incremented.

```
private bool enter;
private bool space;
private bool escape;
private bool move;
private bool gameOver;
private int[] gridPosition;
private bool[] directions;
private Camera camera;
private int mapQuizzes;
private int mapItems;
```

The subroutine is also changed so that the location where the player is located always has a location sprite with a design on it.

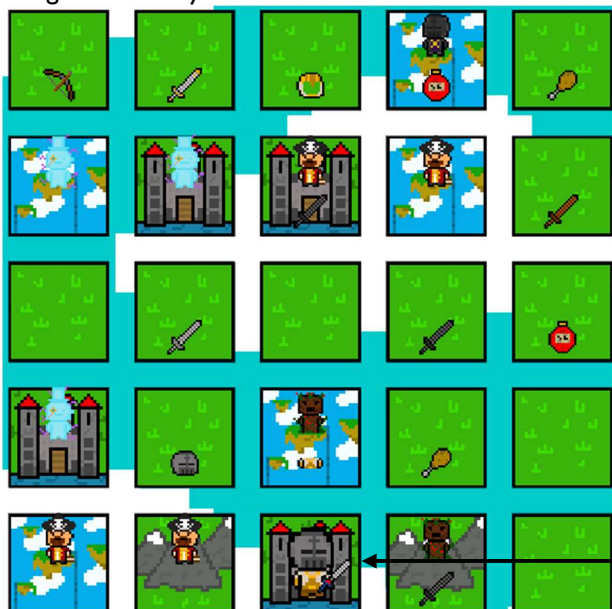
The game loop subroutine could then be adjusted so that the newly adapted populate subroutine could be used.

```
while (!gameOver) {  
    int x = gridPosition [0];  
    int y = gridPosition [1];  
    if (quizArray [x,y] != null) {  
        zoomIn (x, y, 0.375f);  
        quizArray [x, y].scaler (0.5f);  
        quizArray [x, y].move (x+0.2f,y+0.2f);  
        player.scaler (0.5f);  
        player.move (x-0.2f,y);  
        int topicNumber = quizArray [x, y].getTopicNumber();  
        yield return StartCoroutine (runQuiz (topicNumber));  
        zoomOut ();  
        quizArray [x, y].destroy ();  
        quizArray [x, y] = null;  
        player.scaler (1f);  
        player.move (x, y);  
        mapQuizzes--;  
    }  
    if (itemArray [gridPosition [0],gridPosition [1]] != null) {  
        for (int i=0; i<inventory.Length; i++) {  
            if (inventory[i] == null) {  
                inventory [i] = itemArray [x, y];  
                break;  
            }  
        }  
        itemArray [x, y] = null;  
        mapItems--;  
    }  
    if (mapQuizzes == 0 && mapItems == 0) {  
        populate ();  
    }  
}
```

When quizzes are finished and items are collected the relevant variable is decremented.

When both variables are 0, the map is empty and needs to be repopulated.

These changes worked as expected and a new map was generated when needed, effectively allowing the game to carry on forever.



This screen shot shows a newly populated map with the player at a different grid location.

I then decided to implement a score variable. I believe this will foster some competitiveness amongst students and encourage repeated play. In thinking about this, I thought it would also be a good idea to display Attack, Defence, Health and Score on the text screen. To implement this I first added some game objects into the game engine and pass them to the Student controller script.

```
public InputField inputField;  
public Text questionText;  
public Image quizBackground;  
public Slider playerSlider;  
public Image playerBackground;  
public Text playerHealthText;  
public Slider enemySlider;  
public Image enemyBackground;  
public Text enemyHealthText;  
public Image attackOverhead;  
public Image defenceOverhead;  
public Image healthOverhead;  
public Image scoreOverhead;  
public Text attackOverheadText;  
public Text defenceOverheadText;  
public Text healthOverheadText;  
public Text scoreOverheadText;
```

I then adjusted the subroutine where the UI is toggled to deal with these new UI items.

```
public void toggleUI (bool state) {  
    inputField.gameObject.SetActive (state);  
    questionText.gameObject.SetActive (state);  
    quizBackground.gameObject.SetActive (state);  
    playerBackground.gameObject.SetActive (state);  
    playerSlider.gameObject.SetActive (state);  
    playerHealthText.gameObject.SetActive (state);  
    enemyBackground.gameObject.SetActive (state);  
    enemySlider.gameObject.SetActive (state);  
    enemyHealthText.gameObject.SetActive (state);  
    attackOverhead.gameObject.SetActive (!state);  
    defenceOverhead.gameObject.SetActive (!state);  
    healthOverhead.gameObject.SetActive (!state);  
    attackOverheadText.gameObject.SetActive (!state);  
    defenceOverheadText.gameObject.SetActive (!state);  
    healthOverheadText.gameObject.SetActive (!state);  
}
```

Because the new elements aren't needed during quizzes I can simply use the '!' operator to invert the value used to control the quiz items. This ensures that both are never shown together.

I thought it would be useful to see the score in the quiz so it is never deactivated.

Next, I changed the update function to adjust the text displayed in the UI objects.

```
attackOverheadText.text = "Attack: " + playerAttack.ToString();  
defenceOverheadText.text = "Defence: " + playerDefence.ToString();  
healthOverheadText.text = "Health: " + playerHealth.ToString();  
scoreOverheadText.text = "Score: " + score;
```

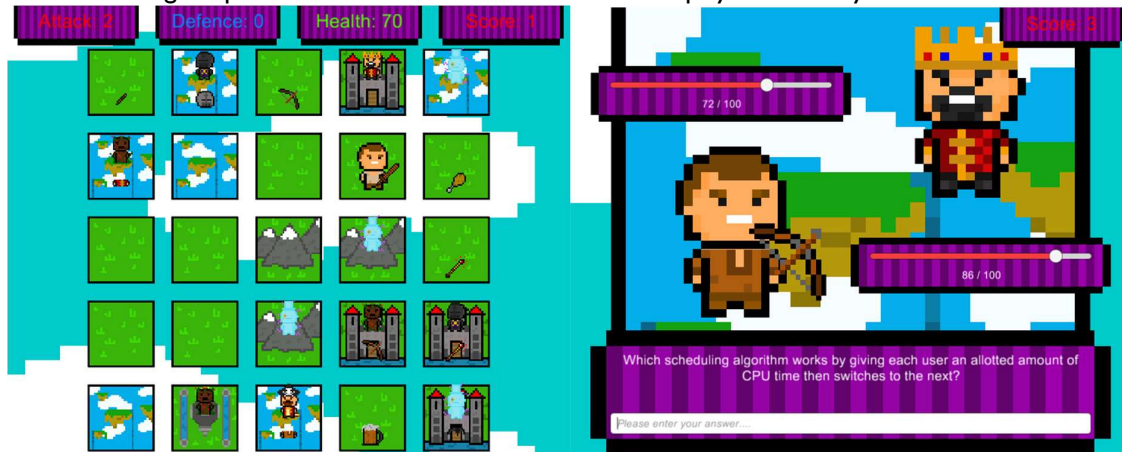

This implementation worked and the new UI elements were used correctly. However, because of the position of the camera some of the map was obscured.



To fix this I simply had to change the values in the zoom out subroutine.

```
public void zoomOut() {  
    camera.orthographicSize = 2.7f;  
    camera.transform.position = new Vector3 (2,2.23f,-10);  
}
```

With this change implemented the new UI elements were displayed correctly.



The next issue with making the game long lasting was that there was the potential for the inventory to fill. To deal with this the script needed to check if there was space in the inventory before adding a new item. The first step to accomplish this was to write a subroutine to check for space.

```
public bool inventorySpace() {  
    bool space = false;  
    for (int i = 0; i < inventory.Length; i++) {  
        if (inventory [i] == null) {  
            space = true;  
        }  
    }  
    return space;  
}
```

This subroutine will only return true if there is a space for an item to fit into the inventory.

I then added some new items into the game engine that the system could use as a popup if the inventory was full. I then passed these new game objects to the student mode controller script so I could use them in code.



```
public Image popupBackground;
public Image popupItem;
public Text popupDetails;
public Text popupText;
```

To control when these new elements were displayed I wrote a new subroutine.

```
public void togglePopupUI (bool state) {
    popupBackground.gameObject.SetActive (state);
    popupItem.gameObject.SetActive (state);
    popupDetails.gameObject.SetActive (state);
    popupText.gameObject.SetActive (state);
}
```

This subroutine mimics the toggle UI subroutine used earlier, however the appearance of the popup isn't based around quizzes and therefore the two subroutines must be separate.

Using the new UI items and the inventory space subroutine I was able to implement a system where the inventory couldn't be over filled.

```
if (itemArray [gridPosition [0],gridPosition [1]] != null) {
    while (inventorySpace () == false) {
        togglePopupUI (true);
        popupItem.sprite = itemArray [gridPosition [0], gridPosition [1]].getSprite ();
        string name = itemArray [gridPosition [0], gridPosition [1]].getName ();
        string effect = itemArray [gridPosition [0], gridPosition [1]].getEffect ().ToString();
        escape = false;
        space = false;
        while (escape == false && space == false) {
            yield return null;
        }
        if (escape) {
            zoomIn (9, 2, 2.5f);
            yield return StartCoroutine (runInventory ());
            zoomOut ();
        } else if (space) {
            itemArray [gridPosition [0], gridPosition [1]]
            itemArray [gridPosition [0], gridPosition [1]]
            break;
        }
        yield return null;
    }
    togglePopupUI (false);
    for (int i=0; i<inventory.Length; i++) {
        if (inventory[i] == null) {
            inventory [i] = itemArray [x, y];
            break;
        }
    }
    itemArray [x, y] = null;
    mapItems--;
}
```

The new elements only need to be used while the inventory has no space, the while loop enforces that.

The details of the items at the grid location that causing the overflow are displayed on the popup.

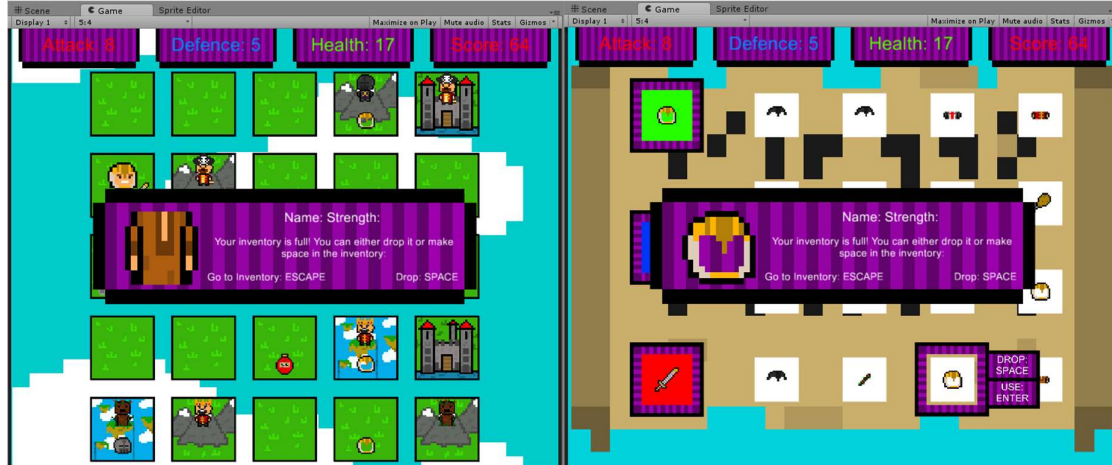
The internal while loop waits for an input.

If the user chooses to open the inventory the run inventory script is run where they can remove an item and cause inventory space to return true.

If the user chooses space the item is destroyed and disregarded and the program flow breaks from the loop.

The remaining code acts as it did before, placing the contents of the grid location in the first available space. If the item was destroyed and disregarded, the space is simply set to null and therefore still free for another item.

This implementation was successful in terms of its function, but not in terms of design. When the inventory was filled and another item was collected the popup was displayed and, correctly reacting to user input, the item could be destroyed or the inventory could be opened. The program flow stayed in the loop until the problem was resolved, but exited after. However sprite that weren't square became stretched, the popup was still displayed when the inventory was opened and I forgot to fill the item details into the name and strength text.



The problem of the stretched sprites emerged because I used a UI item in the popup instead of the sprite renderers used on the main game map. I wanted to keep using a UI element so the item appeared on the canvas and not in the game world and would therefore not be affected by any future camera changes. To fix the problem I found an option on the UI image component to preserve the aspect ratio of the sprites. This ensured sprites wouldn't be distorted.



I then adjusted a few lines from my previous changes to display the item details and hide the popup in the inventory.

```
togglePopupUI (true);
popupItem.sprite = itemArray [gridPosition [0], gridPosition [1]].getSprite ();
string name = itemArray [gridPosition [0], gridPosition [1]].getName ();
string effect = itemArray [gridPosition [0], gridPosition [1]].getEffect ().ToString ();
popupDetails.text = "Name: " + name + " Strength: " + effect;
escape = false;
space = false;
while (escape == false && space == false) {
    yield return null;
}
if (escape) {
    zoomIn (9, 2, 2.5f);
    togglePopupUI (false);
    yield return StartCoroutine (runInventory ());
    zoomOut ();
    togglePopupUI (true);
}
```

I set the popup details text to display the correct information and the toggle popup UI subroutine to hide the popup when the inventory is opened.



In testing this new functionality, I found that it was possible for the player's health to be raised over 100. This not only broke the game mechanics but caused the slider representing the players' health to fail. To fix this is simply had to adjust the set health subroutine in the player class.

```
public void setHealth(int currentHealth) {  
    health = currentHealth;  
    if (health > 100) {  
        health = 100;  
    }  
}
```

With this implemented I could now go about ending the game. This required making several structural changes to the main game loop.

```
public IEnumerator gameLoop () {  
    player = new Player (playerSprite,"Player");  
    gridPosition = new int[2] {0,0};  
    directions = new bool[4] { false, false, false, false };  
    populate ();  
    yield return StartCoroutine (login());  
    while (playerHealth > 0) {  
        directions = new bool[4] { false, false, false, false };  
        escape = false;  
        move = false;  
        while (move == false) {  
            if (escape) {  
                zoomIn (0, 2, 2.5f);  
                yield return StartCoroutine (runInventory());  
                zoomOut ();  
                escape = false;  
            }  
            if (directions[0] || directions[1] || directions[2] || directions[3]) {  
                yield return StartCoroutine (mover ());  
            }  
            yield return null;  
        }  
        int x = gridPosition [0];  
        int y = gridPosition [1];  
        if (itemArray [gridPosition [0],gridPosition [1]] != null) {  
            while (inventorySpace () == false) {  
                togglePopuUI (true);  
                popuItem.sprite = itemArray [gridPosition [0], gridPosition [1]].getSprite ();  
                string name = itemArray [gridPosition [0], gridPosition [1]].getName ();  
                string effect = itemArray [gridPosition [0], gridPosition [1]].getEffect ().ToString();  
                popuDetails.text = "Name: " + name + " Strength: " + effect;  
                escape = false;  
                space = false;  
                while (escape == false && space == false) {  
                    yield return null;  
                }  
                if (escape) {  
                    zoomIn (0, 2, 2.5f);  
                    togglePopuUI (false);  
                    yield return StartCoroutine (runInventory ());  
                    zoomOut ();  
                    togglePopuUI (true);  
                } else if (space) {  
                    itemArray [gridPosition [0], gridPosition [1]].destroy ();  
                    itemArray [gridPosition [0], gridPosition [1]] = null;  
                    break;  
                }  
                yield return null;  
            }  
            togglePopuUI (false);  
            for (int i=0; i<inventory.Length; i++) {  
                if (inventory[i] == null) {  
                    inventory [i] = itemArray [x, y];  
                    break;  
                }  
            }  
            itemArray [x, y] = null;  
            mapItems--;  
        }  
        if (quizArray [x,y] != null) {  
            zoomIn (x, y, 0.375f);  
            quizArray [x, y].scaler (0.5f);  
            quizArray [x, y].move (x*0.2f,y*0.2f);  
            player.scaler (0.5f);  
            player.move (x*0.2f,y);  
            int topicNumber = quizArray [x, y].getTopicNumber();  
            yield return StartCoroutine (runQuiz (topicNumber));  
            zoomOut ();  
            quizArray [x, y].destroy ();  
            quizArray [x, y] = null;  
            player.scaler (1f);  
            player.move (x, y);  
            mapQuizzes--;  
        }  
        if (mapQuizzes == 0 && mapItems == 0) {  
            populate ();  
        }  
    }  
    yield return null;  
}
```

The game over variable was redundant, instead I just checked that the players health is greater than 0.

I moved the section of code where the player can open the inventory or move, originally at the end of the loop, to the start. This means that once the player dies they can't make another more or continue using the inventory. Because there is never a quiz or an item at the start location this change didn't cause any issues.

I then also moved the code for handling items to before the part that checked for quizzes. This meant that the item popup wouldn't appear if the player was dead.

Now, when the player's health went below one they were unable to move or open the inventory. This only left displaying an end game message and the game would be complete.

I added some new game object which were passed to the student mode control script and wrote a new subroutine to use them to display and end game message.

```
public void endGame () {  
    endGameBackground.gameObject.SetActive (true);  
    endGameText.gameObject.SetActive (true);  
    endGameScoreText.gameObject.SetActive (true);  
    endGameScoreText.text = "Score: " + score;  
}
```

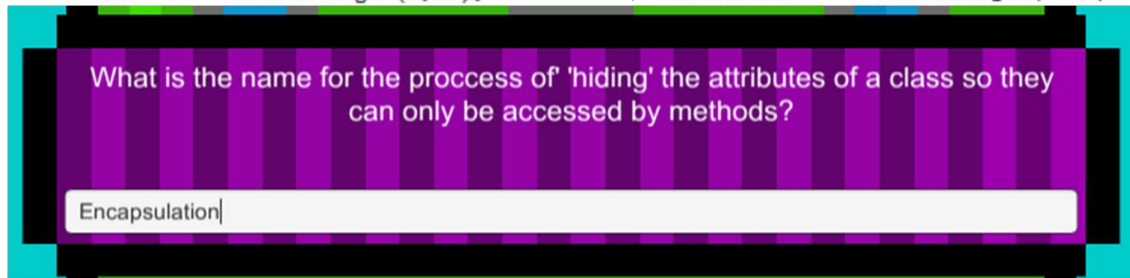
I disabled all these game objects in the game engine inspector so their active state only had to be changed once.

I placed a call for this subroutine after the while loop in the main game loop ends, this worked to display an end game message on the screen when the player died.



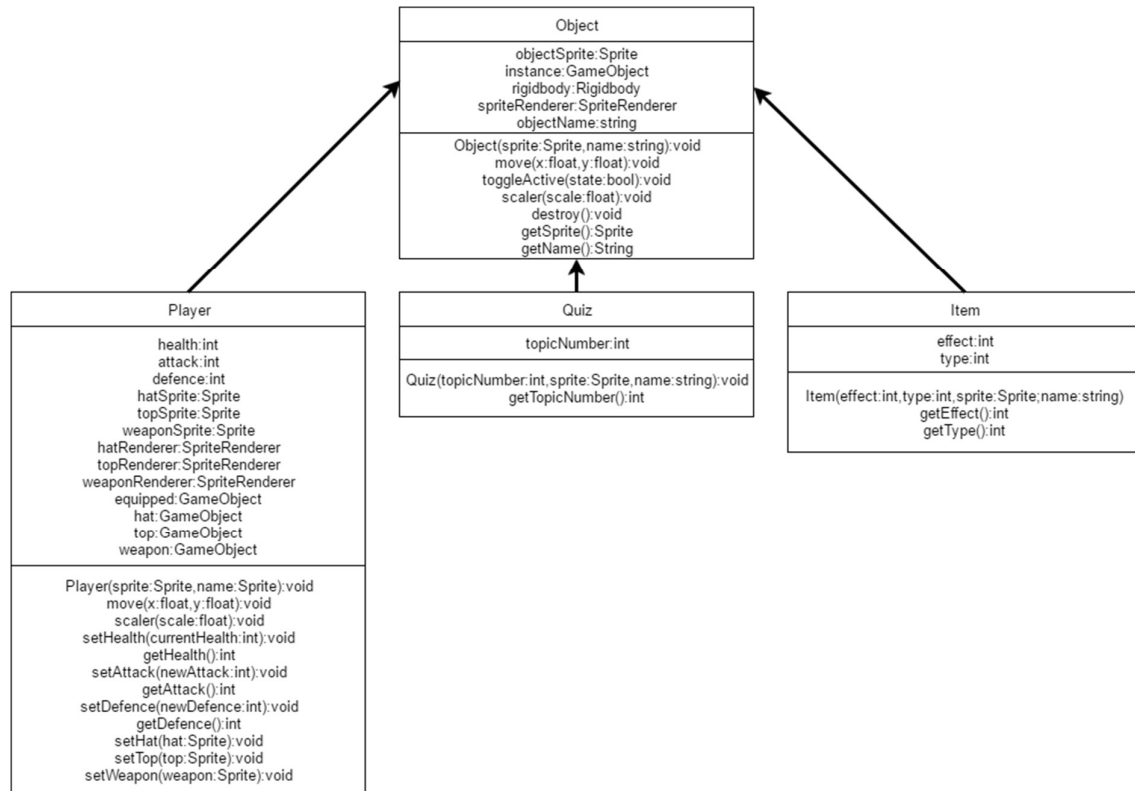
Through testing these changes I noticed that quizzes from the programming topic weren't appearing. I traced this back to a simple indexing error in the random topic generator. Changing the line resolved the problem.

```
randomNumber = Random.Range (1,12); —————> randomNumber = Random.Range (1,13);
```



With the changes made in this phase the game in my application is now easier to understand and able to prevent logic errors. A score variable is implemented to encourage re-playability and competition and the game is ended when the user loses. The student mode is now finished and I was free to move on to focus on the staff portion of the application.

Over the course of making this part of the application the reality of how classes were implemented in the system deviated from my plan. In the interest of keeping a solid understanding of my application is a continued development I remade the class diagram. Some methods appear in both a derived class and its parent because the method is expanded on in the inherited class.



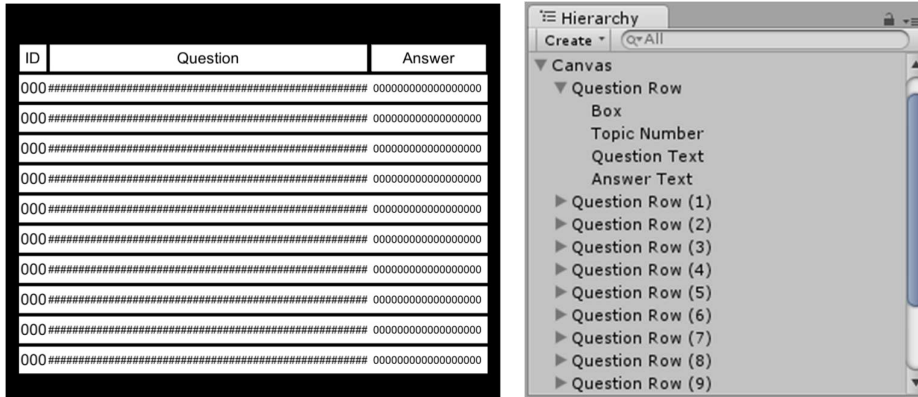
Client Feedback

With these final changes implemented my client was satisfied that I had met the deliverable requiring the student facing portion of the application to take the form of a game. My client was happy that the method of progressing and ending the game was designed to encourage repeated play and foster competitiveness. With the student facing part of the application complete, My Byford gave me his consent to move on the staff portion of the application.

Phase 6: Editing the list of questions

Of the two parts of staff portion of the application I decided to work on the scene where the questions list can be added first. This would require me to create game objects to make a table where the questions would be displayed and use the MySQL connector to edit, add and remove questions.

To display the questions I created various UI element and arranged them in a table. Each row of the table was made up of an empty game object with three text objects for children. Organising the elements in this way would make it easier in the future to adjust their position or use them in a script.



This UI design doesn't yet match up with the design I agree with the client but I didn't want to massively complicate the problems before I had any sort of solution.

I then began to write a script to control the questions scene. The first stage in doing this was to declare the public and private variables.

```
public int questionsOnScreen;  
public MySQLconnector databaseScript;  
public GameObject[] questionsDisplay;
```

I decided it was best to use the same MySQL Connector script again instead of creating a new one. Re-using it meant my program would be made up of less scripts and therefore be smaller and less complicated. Because I arranged the game objects in the way I did only the 10 parent objects needed to be directly passed to the script, not the 30 individual text boxes.

```
private bool Up;  
private bool Down;  
private bool Left;  
private bool Right;  
private string[,] questions;  
private int topicNumber;  
private int displayStartIndex;  
private Text[] topicNumberTexts;  
private Text[] questionTexts;  
private Text[] answerTexts;
```

Approaching the problems of the questions screen I decided it would be best to take from ideas I'd already used. My vision for the way it functioned was that all the questions would be stored in an array and which would be positioned to fill a grid on the screen, not dissimilar from the way the items work in the game. The directional keys would be used to navigate through the table in the same way the player or item outline is moved. In the private variables the four Boolean values will be used for movement and the string array to store the questions. The two integer variables will be used to manage which selection of questions is displayed and the three Text arrays will hold the UI Text boxes used to display questions.

With these variables in place I wrote the basic update function. I did this before writing the start function because update only needed to be very simple at this stage.

```
void Update () {
    if (Input.GetKeyDown (KeyCode.UpArrow)) {
        Up = true;
    }
    if (Input.GetKeyDown (KeyCode.DownArrow)) {
        Down = true;
    }
    if (Input.GetKeyDown (KeyCode.LeftArrow)) {
        Left = true;
    }
    if (Input.GetKeyDown (KeyCode.RightArrow)) {
        Right = true;
    }
}
```

As always, the update function is used to deal with user input.

I then began to write the start function. This required a lot of new manipulation of game objects so I only partially completed it before looking back on and analysing it.

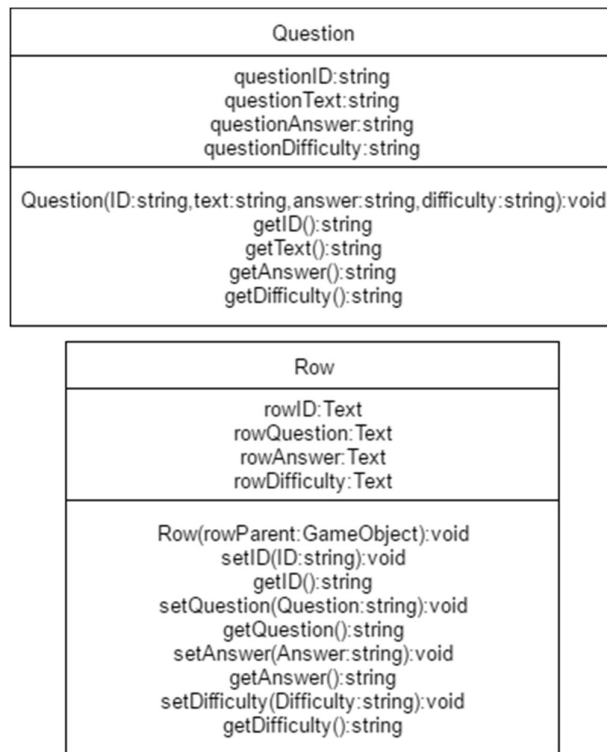
```
void Start () {
    Up = false;
    Down = false;
    Left = false;
    Right = false;
    topicNumber = 1;
    displayStartIndex = 0;
    topicNumberTexts = new Text[questionsDisplay.Length];
    questionTexts = new Text[questionsDisplay.Length];
    answerTexts = new Text[questionsDisplay.Length];
    Debug.Log ("Arrays Created");
    for (int i=0; i<questionsDisplay.Length; i++) {
        Debug.Log ("Start Loop");
        topicNumberTexts [i] = questionsDisplay [i].transform.Find ("Topic Number").GetComponent<Text> ();
        topicNumberTexts [i].text = "Hello World";
        Debug.Log ("End Loop");
    }
}
```

The for loop is to populate the text box arrays with the children of the passed game objects. Because each game object contains multiple text objects I had to use the transform.Find function and search for the separate Text game objects by name.

This code worked, correctly finding and editing the text in all the topic ID game objects.

ID	Question	Answer
Hell#####		000000000000000000
Hell#####		000000000000000000
Hell#####		000000000000000000
Hell#####		000000000000000000
Hell#####		000000000000000000
Hell#####		000000000000000000
Hell#####		000000000000000000
Hell#####		000000000000000000
Hell#####		000000000000000000
Hell#####		000000000000000000

However, at this stage I had a revelation regarding the way I was organising data in my code. Using classes worked so effectively in the student portion of the application they should be useful in this code too. To solidify my understanding of how these needed to be implemented I designed a class diagram for this script.



I first started by writing a Question class to hold data relating to specific questions.

```
class Question {
    private string questionID;
    private string questionText;
    private string questionAnswer;
    private string questionDifficulty;
    public Question (string ID, string text, string answer, string difficulty) {
        questionID = ID;
        questionText = text;
        questionAnswer = answer;
        questionDifficulty = difficulty;
    }
    public string getID() {
        return questionID;
    }
    public string getText() {
        return questionText;
    }
    public string getAnswer() {
        return questionAnswer;
    }
    public string getDifficulty() {
        return questionDifficulty;
    }
}
```

Holding all this information under one identifier was far simpler.

At this stage, the question class was very simple and didn't require much explaining.

These getter functions maintain encapsulation.

I then wrote a Row class to manage the text game objects. I didn't complete this class because I just wanted to get back to the stage I had been at without the classes.

```
class Row {
    private Text rowID;
    private Text rowQuestion;
    private Text rowAnswer;
    private Text rowDifficulty;
    public Row (GameObject rowParent) {
        rowID = rowParent.transform.Find ("Topic Number").GetComponent<Text> ();
        rowID.text = "Hello World";
    }
}
```

The code that made up the constructor was just copied from the start subroutine.

I then adjusted the start subroutine to implement these new classes. This worked to produce the same result as before, but in a more concise and future proof way.

```
void Start () {
    Up = false;
    Down = false;
    Left = false;
    Right = false;
    topicNumber = 1;
    displayStartIndex = 0;
    rows = new Row[questionsDisplay.Length];
    for (int i=0; i<questionsDisplay.Length; i++) {
        rows [i] = new Row (questionsDisplay[i]);
    }
}
```

ID	Question	Answer
Hel#####		00000000000000000000
Hel#####		00000000000000000000
Hel#####		00000000000000000000
Hel#####		00000000000000000000
Hel#####		00000000000000000000
Hel#####		00000000000000000000
Hel#####		00000000000000000000
Hel#####		00000000000000000000
Hel#####		00000000000000000000

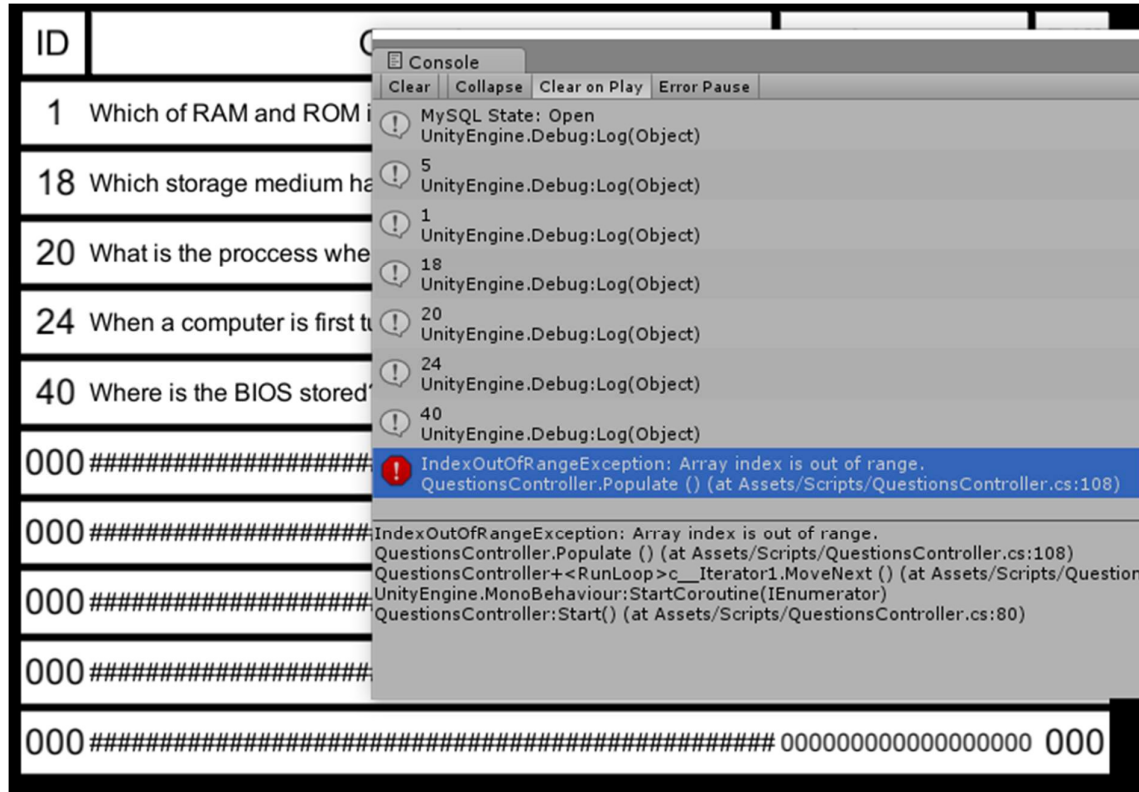
Using my new classes I added to the start subroutine to test the UI text elements.

```
void Start () {  
    Up = false;  
    Down = false;  
    Left = false;  
    Right = false;  
    topicNumber = 1;  
    displayStartIndex = 0;  
    rows = new Row[questionsDisplay.Length];  
    for (int i=0; i<questionsDisplay.Length; i++) {  
        rows [i] = new Row (questionsDisplay[i]);  
    }  
    string[,] topicQuestions = databaseScript.findTopicQuestions (7);  
    questions = new Question[topicQuestions.Length / 4];  
    Debug.Log (questions.Length);  
    for (int i=0; i<questions.Length; i++) {  
        questions [i] = new Question (topicQuestions [i, 0], topicQuestions [i, 1],  
            topicQuestions [i, 2], topicQuestions [i, 3]);  
        Debug.Log (topicQuestions[i,0]);  
    }  
    for (int i=0; i<rows.Length; i++) {  
        rows [i].setID(questions [i].getID());  
        rows [i].setQuestion(questions [i].getText());  
        rows [i].setAnswer(questions [i].getAnswer());  
        rows [i].setDifficulty(questions [i].getDifficulty());  
    }  
}
```

Because I had the most questions for it, it made sense to set the system to only look at topic 7.

The program loops through the rows and fills them with question information.

This worked when there were more questions than the amount of rows but when there were less the code returned and Index out of Bounds error.



Because I would want to populate the questions multiple times I took some of the code from the start subroutine and created a separate populate procedure. I then slightly changed the loop conditions to avoid the indexing error.

```
public void Populate () {  
    string[,] topicQuestions = databaseScript.findTopicQuestions (1);  
    questions = new Question[topicQuestions.Length / 4];  
    Debug.Log (questions.Length);  
    for (int i=0; i<questions.Length; i++) {  
        questions [i] = new Question (topicQuestions [i, 0], topicQuestions [i, 1],  
            topicQuestions [i, 2], topicQuestions [i, 3]);  
        Debug.Log (topicQuestions[i,0]);  
    }  
    for (int i=0; i<rows.Length && i < questions.Length; i++) {  
        rows [i].setID(questions [i].getID());  
        rows [i].setQuestion(questions [i].getText());  
        rows [i].setAnswer(questions [i].getAnswer());  
        rows [i].setDifficulty(questions [i].getDifficulty());  
    }  
}
```

The for loop will now stop when there are no more questions.

I then wrote a coroutine to control the input loop of the scene. I placed a call for this at the end of the Start coroutine.

```
public IEnumerator RunLoop () {  
    Populate ();  
    while (true) {  
        if (left == true || right == true) {  
            if (left == true && topicNumber > 1) {  
                topicNumber--;  
                Populate ();  
            }  
            if (right == true && topicNumber < 12) {  
                topicNumber++;  
                Populate ();  
            }  
            left = false;  
            right = false;  
        }  
        yield return null;  
    }  
    yield return null;  
}
```

I decided to use the left and right keys to control the topic. Using inequalities the code prevents the topic from leaving the range.

This code replaced the questions responding to the user input. However, when the topic was changed and the new topic had less questions the text from the previous topic.

ID	Question	Answer	Diff
2	What acts as a unique identifier in a Database?	PRIMARY KEY	0
4	Which additional piece of hardware is needed for a Star	SWITCH	0
15	The Internet is the largest example of what type of network?	WAN	0
36	Convert the hexadecimal number 3E into decimal	62	0
37	Which Logic gate takes two inputs and returns TRUE only if	AND	0
38	Which Logic gate takes two inputs and returns TRUE if atleast	OR	0
23	Which scheduling algorithm works by giving each user an	ROUND ROBIN	0.5
31	What is used to transmit signals from the CPU to the	CONTROL BUS	0.5
32	What register of the CPU hold the result of logical and	ACC	0.5
33	What register holds data that has been written to or read from	MDR	0.5

The questions in the top are related to Data whereas the questions at the bottom are related to Computer Fundamentals.

To fix this I needed to deal with every single row, not just the ones for which there were questions. Then, within the for loop, I placed an if condition. Rows with a question would be treated as before but those without are set to 'string.Empty' i.e. an empty string.

```
public void Populate () {
    string[,] topicQuestions = databaseScript.findTopicQuestions (topicNumber);
    questions = new Question[topicQuestions.Length / 4];
    Debug.Log (questions.Length);
    for (int i=0; i<questions.Length; i++) {
        questions [i] = new Question (topicQuestions [i, 0], topicQuestions [i, 1],
            topicQuestions [i, 2], topicQuestions [i, 3]);
        Debug.Log (topicQuestions[i,0]);
    }
    for (int i=0; i<rows.Length; i++) {
        if (i < questions.Length) {
            rows [i].setID (questions [i].getID ());
            rows [i].setQuestion (questions [i].getText ());
            rows [i].setAnswer (questions [i].getAnswer ());
            rows [i].setDifficulty (questions [i].getDifficulty ());
        } else {
            rows [i].setID (string.Empty);
            rows [i].setQuestion (string.Empty);
            rows [i].setAnswer (string.Empty);
            rows [i].setDifficulty (string.Empty);
        }
    }
}
```

The criterion from the for loop is effectively moved into the loop so the code can work on elements that both pass and don't pass it.

This solution worked and rows with no data to fill them were set to blank.

ID	Question	Answer	Diff	ID	Question	Answer	Diff
1	Which of RAM and ROM is volatile?	RAM	0	2	What acts as a unique identifier in a Database?	PRIMARY KEY	0
18	Which storage medium has no moving parts and is, therefore, SOLID STATE		0	4	Which additional piece of hardware is needed for a Star	SWITCH	0
20	What is the process where RAM is extended by using part of VIRTUAL MEMORY		0	15	The Internet is the largest example of what type of network?	WAN	0
24	When a computer is first turned on what is run?	BIOS	0				
40	Where is the BIOS stored?	ROM	0				

In the interest of clarity I decided to add a topic name label at the top of the screen. To do this I first had to add a subroutine to the MySQL Connector class to retrieve the topic name based on the relevant ID number.

```
public string findTopicName(int topicNumber) {
    query = "SELECT topicName FROM topic WHERE topicID = " + topicNumber + ";";
    command = new MySqlCommand (query, connection);
    reader = command.ExecuteReader();
    string topicName;
    if (topicNumber < 7) {
        topicName = "GCSE ";
    } else {
        topicName = "A-Level ";
    }
    while (reader.Read ()) {
        topicName = topicName + reader.GetString (0);
    }
    reader.Close ();
    return topicName;
}
```

This query returns the name of the topicID input.

The subroutine also adds the exam level so teachers can write questions with that in mind

By using this subroutine in the questions script, combined with some new game elements on the screen the screen now displays the topic name.

Topic Name: A-Level Software			
ID	Question	Answer	Diff
25	What piece of software allows a device to emulate another?	VIRTUAL MACHINE	0.777
28	Which type of translator translates the entirety of code in one	COMPILER	0.65
29	Which type of translator translates code line by line?	INTERPRETER	0.756
30	In Lexical Analysis keywords and symbols are replaced with	TOKENS	0.805

I then worked to implement the ability for specific questions to be selected. This required some way of identifying which question is selected and a way to scroll down when there were more questions than the number of rows.

I first added a question outline into the game engine and passed it into the script. I then changed the update function to move the outline depending on the focus row variable.

ID	Question	Answer	Diff
000	#####	00000000000000000000	000
000	#####	00000000000000000000	000
000	#####	00000000000000000000	000

```
public Image questionOutline;
questionOutline.rectTransform.localPosition = new Vector3 (0f, 170f-focusRow*45, 0f);
```

The next step was to change the run loop to handle Up and Down input. This posed an interesting challenge because not all questions would be displayed on the screen at once and the system needed to take into account the questions and rows index. The solution I found uses nested if statement which, while complicated, I believe was the best way to tackle the problem.

```
if (up == true) {
    if (focusRow > 0) {
        focusRow--;
    } else if (displayStartIndex > 0) {
        displayStartIndex--;
        Populate ();
    }
    up = false;
}
if (down == true) {
    if (focusRow < rows.Length - 1 && focusRow < questions.Length - displayStartIndex - 1) {
        focusRow++;
    } else if (displayStartIndex < questions.Length - rows.Length) {
        displayStartIndex++;
        Populate ();
    }
    down = false;
}
```

If the focus row isn't the first row the outline can simply be moved up.

If the focus row is 0 the only way Up would be a valid action is if the display start index wasn't 0.

Even if no action is completed Up still needs to be reset to stop the if statement being satisfied.

Moving the outline down is only valid if there is another row to move too and if there is a question in that row.

If the outline is at the bottom row the only way Down is valid would be if there were more questions than rows from the display start.

I then needed to change the populate subroutine to react to the display start index.

```
public void Populate () {
    string[,] topicQuestions = databaseScript.findTopicQuestions (topicNumber);
    questions = new Question[topicQuestions.Length / 4];
    for (int i=0; i<questions.Length; i++) {
        questions [i] = new Question (topicQuestions [i, 0], topicQuestions [i, 1],
            topicQuestions [i, 2], topicQuestions [i, 3]);
        Debug.Log (topicQuestions[i,0]);
    }
    for (int i=0; i<rows.Length; i++) {
        if (i + displayStartIndex < questions.Length) {
            rows [i].setID (questions [i+displayStartIndex].getID ());
            rows [i].setQuestion (questions [i+displayStartIndex].getText ());
            rows [i].setAnswer (questions [i+displayStartIndex].getAnswer ());
            rows [i].setDifficulty (questions [i+displayStartIndex].getDifficulty ());
        } else {
            rows [i].setID (string.Empty);
            rows [i].setQuestion (string.Empty);
            rows [i].setAnswer (string.Empty);
            rows [i].setDifficulty (string.Empty);
        }
    }
    focusRow = 0;
}
```

Changing this was pretty simple, I just had to use the display start variable as an offset for the questions array index.

This worked to implement 'scrolling' into the application. The system would correctly move though the full question list so all items can be displayed.

6	Which part of the CPU performs all logical and arithmetical	ALU	0.6	7	In a standard modern PC, what is the Immediate Access	RAM	0.692
7	In a standard modern PC, what is the Immediate Access	RAM	0.692	11	Which register is the contents of the PC copied to in order for	MAR	0.593
11	Which register is the contents of the PC copied to in order for	MAR	0.593	17	Which system architecture is used in modern consumer	VON NEUMANN	0.592
17	Which system architecture is used in modern consumer	VON NEUMANN	0.592	21	What phenomena can occur when too much Virtual Memory is	DISK THRESHING	0.592
21	What phenomena can occur when too much Virtual Memory is	DISK THRESHING	0.592	22	Which type of processing is most suited to a payroll system or	BATCH	0.612
22	Which type of processing is most suited to a payroll system or	BATCH	0.612	23	Which scheduling algorithm works by giving each user an	ROUND ROBIN	0.571
23	Which scheduling algorithm works by giving each user an	ROUND ROBIN	0.571	31	What is used to transmit signals from the CPU to the	CONTROL BUS	0.571
31	What is used to transmit signals from the CPU to the	CONTROL BUS	0.571	32	What register of the CPU hold the result of logical and	ACC	0.512
32	What register of the CPU hold the result of logical and	ACC	0.512	33	What register holds data that has been written to or read from	MDR	0.589
33	What register holds data that has been written to or read from	MDR	0.589	34	Which type of processor is designed so the same instructions	ARRAY	0.555

The next function I needed to add was the ability to delete questions. To do this I needed to first write a subroutine in the MySQL script to do this.

```
public void deleteQuestion(string questionID) {  
    query = "DELETE FROM question WHERE questionID = " + questionID + ";";  
    command = new MySqlCommand (query, connection);  
    reader = command.ExecuteReader();  
    reader.Close ();  
}
```

This was a pretty simplistic task. Using the unique primary key that every question has this query deletes a specific question.

I could then add another if statement to the run loop to facilitate the calling of this subroutine.

```
if (space == true) {  
    if (rows[focusRow].getID() != "") {  
        databaseScript.deleteQuestion (rows [focusRow].getID());  
        Populate ();  
    }  
    space = false;  
}
```

This code passes the ID of the question in a specific row to the delete question subroutine.

This code needs to check that the row isn't empty because it was now possible that there are no questions for a specific topic.

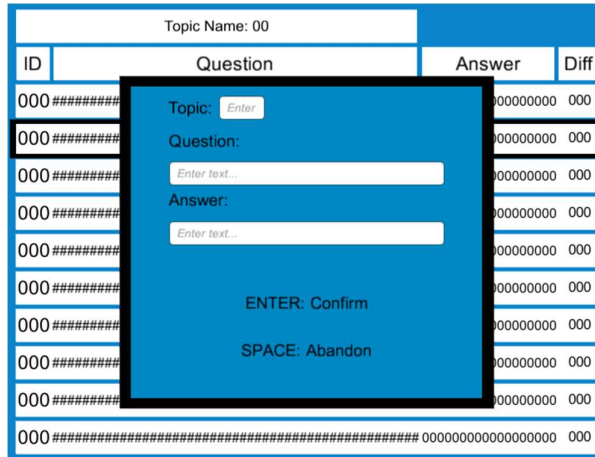
These two subroutines worked in tandem to remove questions from the database. Because of the way I had written loops throughout the previously written code, this didn't cause any errors in the student or questions scene if topics became empty.

The next step in building the question scene was to implement the ability to edit questions. Similarly to deleting questions, the first stage was to write a new subroutine in the MySQL script.

```
public void updateQuestion (string questionText, string answer, string topicID, string questionID) {  
    query = "UPDATE question SET questionText = '" + questionText + "', answer = '" + answer + "', " +  
        "_topicID = '" + topicID + "' WHERE questionID = " + questionID;  
    command = new MySqlCommand (query, connection);  
    reader = command.ExecuteReader ();  
    reader.Close ();  
}
```

Using primary keys to identify questions, this subroutine alters records in the question table based on the input passed to it.

I then created some new game objects in the game engine to allow the user to change details of a question.



With these game elements in place I could edit the questions script to use them with the MySQL subroutine.

```

if (enter == true) {
    if (rows [focusRow].getID () != "") {
        toggleInputUI (true);
        topicField.text = topicNumber.ToString();
        questionField.text = rows [focusRow].getQuestion ();
        answerField.text = rows [focusRow].getAnswer ();
        enter = false;
        space = false;
        while (true) {
            if (enter) {
                databaseScript.updateQuestion (questionField.text,
                    answerField.text, topicField.text, rows [focusRow].getID());
                break;
            }
            if (space) {
                break;
            }
            yield return null;
        }
        toggleInputUI (false);
    }
    enter = false;
    space = false;
    Populate ();
}
    
```

The code needs to check that rows aren't empty to prevent an error in the MySQL script.

The contents of the input fields are paired with the question ID and passed to the MySQL subroutine.

Pressing space instead of enter simply exits the loop. Any changes made to the input field are disregarded.

My solution successfully allowed the user to manipulate the questions table via the application. The example below shows a question that has had its answer and topic changed.

Topic Name: GCSE System's Architecture				Topic Name: GCSE Exchanging Data			
ID	Question	Answer	Diff	ID	Question	Answer	Diff
18	Which storage medium has no moving parts and is, therefore, SOLID STATE		0	18	Which storage medium has no moving parts and is, therefore, CHANGE		0

The final function the application needed was the ability to add entirely new questions. This wasn't dissimilar from earlier steps in that it first required me to edit the MySQL script.

```
public void addQuestion (string questionText, string answer, string topicID) {  
    query = "INSERT INTO question (questionText, answer, _topicID) VALUES ('"+  
        questionText+"', '"+answer+"', '"+topicID+"");  
    command = new MySqlCommand (query, connection);  
    reader = command.ExecuteReader ();  
    reader.Close ();  
}
```

This is practically identical to the other subroutines. The differences being the use of the Insert function and the lack of a question ID because a new one will be created.

It then dawned on me that using the Space button to abandon changes probably wasn't a very good idea because most questions would have a Space in them. Using find and replace I swapped Space out with references to the escape key.

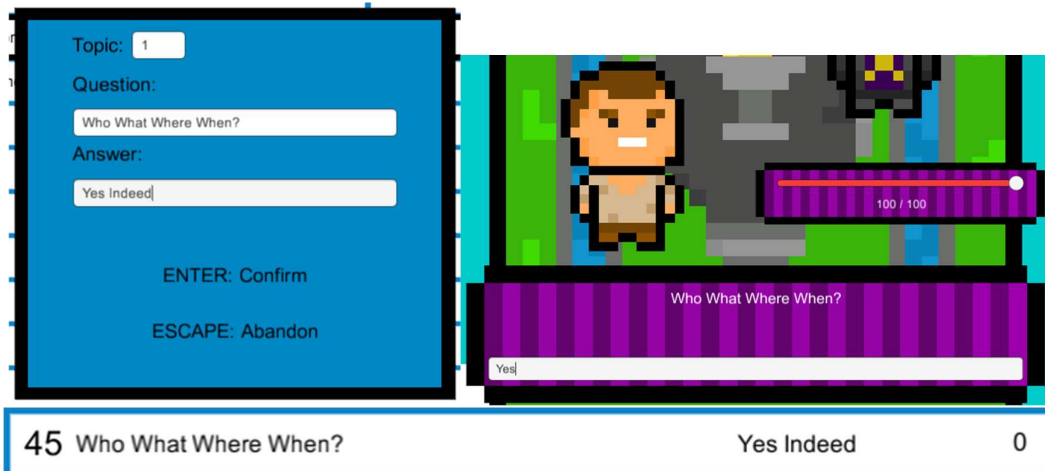
With this fix implemented I added a final if statement to the run loop coroutine to deal with adding new questions.

```
if (a) {  
    toggleInputUI (true);  
    topicField.text = string.Empty;  
    questionField.text = string.Empty;  
    answerField.text = string.Empty;  
    enter = false;  
    escape = false;  
    while (true) {  
        if (enter) {  
            databaseScript.addQuestion (questionField.text, answerField.text, topicField.text);  
            Populate ();  
            break;  
        }  
        if (escape) {  
            break;  
        }  
        yield return null;  
    }  
    toggleInputUI (false);  
    enter = false;  
    escape = false;  
    a = false;  
}
```

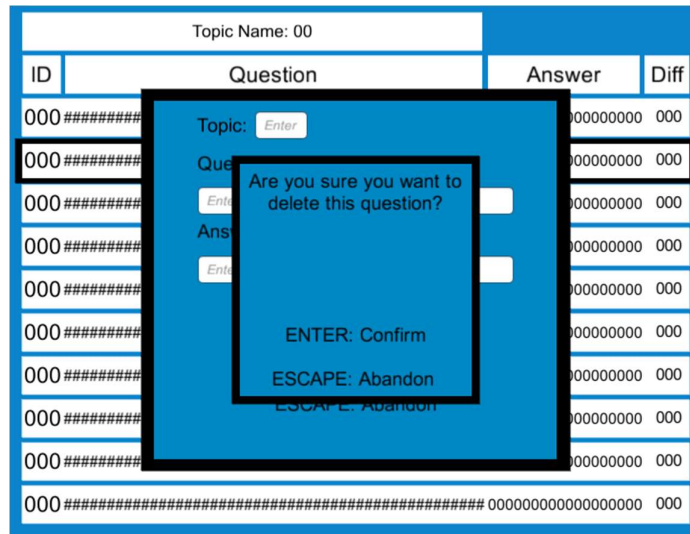
This code was largely copied from the edit if statement without the check to see if the row is empty.

I considered using the + key to add questions because it would appear in text less frequently. However I decided against it when I looked down at my own keyboard and saw there was no + key and it could only be typed using Shift =.

The following screenshots display how this solution worked to introduce new questions into the application.



Now that all the functionality was implemented I began to think about the user-friendliness of the questions scene. I decided an 'Are You Sure' popup when deleting questions would prevent accidental deletion. The first step in doing this was to create some game objects in the engine to make up the popup.



I then wrote a new subroutine in the questions script to toggle this UI.

```
public void toggleAYSUI (bool state) {
    AYSBackground.gameObject.SetActive (state);
    AYSText.gameObject.SetActive (state);
    AYSQuestion.gameObject.SetActive (state);
}
```

This is pretty much exactly the same as every other toggle subroutine.

I then wrote another subroutine to handle the function of the AYS popup.

```
public IEnumerator runAYS () {
    toggleAYSUI (true);
    enter = false;
    escape = false;
    AYSQuestion.text = rows [focusRow].getQuestion ();
    while (true) {
        if (enter == true) {
            databaseScript.deleteQuestion (rows [focusRow].getID());
            break;
        }
        if (escape == true) {
            break;
        }
        yield return null;
    }
    enter = false;
    escape = false;
    toggleAYSUI (false);
    yield return null;
}
```

The question text is displayed in the popup so users know what their deleting.

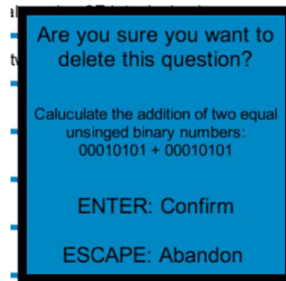
If the user presses enter the question is deleted, if they press space the deletion is deleted. This key for deletion is intentionally the opposite of what it is in the run loop to prevent questions being deleted by accidental double taps.

With this subroutine in place I could adjust the run loop to use the popup.

```
if (escape == true) {  
    if (rows[focusRow].getID() != "") {  
        yield return StartCoroutine(runAYS ());  
        Populate ();  
    }  
    escape = false;  
}
```

The bulk of the code in this branch of the if statement could be replaced with a call to the runAYS subroutine.

When implemented, this change meant that the AYS screen would popup and act as a secondary step to deleting questions.



However, through testing this feature I encountered an issue with my application. The delete question subroutine resulted in an error for any question that had answers to it. When I first built the delete functionality I had just refreshed the database and there were no records in the answer instances table so I didn't encounter this error. However, the table was now populated so the error occurred.

```
! MySQLException: Cannot delete or update a parent row: a foreign key constraint fails  
MySQL.Data.MySqlClient.MySqlStream.ReadPacket ()
```

After looking at the code involved in deleting the question and reading the details of the error I discovered that the problem was with the delete question subroutine in the MySQL script. My code resulted in an error because of the database principal of 'Referential Integrity'. This states that foreign key fields must be consistent with the primary record they reference.

```
public void deleteQuestion(string questionID) {  
    query = "DELETE FROM question WHERE questionID = " + questionID + ";";  
    command = new MySqlCommand (query, connection);  
    reader = command.ExecuteReader();  
    reader.Close ();  
}
```

The subroutine currently just deletes questions from the question table. This causes a breach of referential integrity when the question has answers that reference it.

To fix this I needed to first delete any answers to a question before the question itself. This was as simple as adding another query to the subroutine before the first.

```
public void deleteQuestion(string questionID) {  
    query = "DELETE FROM answerInstance WHERE _questionID = " + questionID + ";";  
    command = new MySqlCommand (query, connection);  
    reader = command.ExecuteReader();  
    reader.Close ();  
    query = "DELETE FROM question WHERE questionID = " + questionID + ";";  
    command = new MySqlCommand (query, connection);  
    reader = command.ExecuteReader();  
    reader.Close ();  
}
```

This change worked to delete all instances to questions and uphold referential integrity, therefore preventing this error.

With this final function implemented the Questions portion of the application was now finished. Staff users can now edit, add and remove questions thus meeting some of the most fundamental of my Staff based deliverables. I was now free to move on to developing the final scene of the application, where staff can view students results.

Client Feedback

My client was satisfied that the solution I had implemented met the success criteria we agreed on that staff should be able to edit, add and delete questions. He was satisfied with that the 'small amount of questions on screen at one time' approach we agreed on at the design stage made the information on screen a lot easier to digest. Although it wasn't a part of the agreed design when my Byford saw the 'different topics on different tabs' system for grouping questions he agreed that this made the information on screen a lot easier to process. Having received this positive feedback I was able to move on to developing the results part of the application.

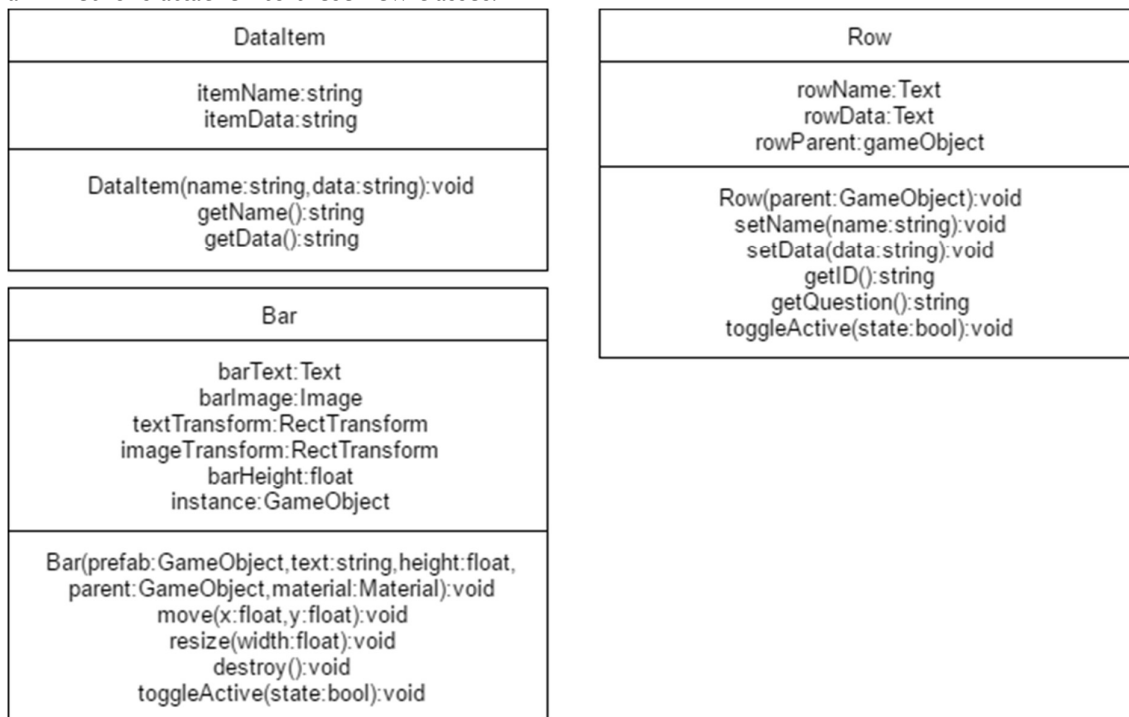
- Changing the 'Row' class to deal with there only being two fields

```
class Row {
    private Text rowName;
    private Text rowData;
    public Row (GameObject rowParent) {
        rowName = rowParent.transform.Find ("Name Text").GetComponent<Text> ();
        rowData = rowParent.transform.Find ("Data Text").GetComponent<Text> ();
    }
    public void setName (string name) {
        rowName.text = name;
    }
    public void setData (string data) {
        rowData.text = data;
    }
    public string getID () {
        return rowName.text;
    }
    public string getQuestion () {
        return rowData.text;
    }
}
```

Making this change meant that any references to the Row class in the code had to be slightly changed to account for the fact that the last two getters and setters were no longer needed.

- Removing the input popup and the related subroutines and code
- Removing the 'Are You Sure' popup and the related subroutines and code

After again implemting unexpected classes I decided to create a class diagram to plan the attributtes and methods attached to these new classes.



Making these changes meant that I had the foundation for the results scene pretty quickly. I tested the scene running the results controller script and allowing the menu to be populated with the questions information.

Topic Name: A-Level System's Architecture			
ID	Question	Answer	Diff
	Which part of the CPU performs all logical and	0.3333333	
	Which register is the contents of the PC copied	0	
	Which scheduling algorithm works by giving	0.25	
	What register of the CPU hold the result of	0	
	What register holds data that has been written to	0.25	
	Which type of processor is designed so the	0.2	
	What extra component might a Gamring PC	0.6666667	

I could now start working on making the results scene display the actual required data. The first step to in doing this was retrieving that data from the database itself. I decided to write one larger subroutine that would retrieve all the data needed for any filter setting in one lump sum as this a less processor intensive alternative to re-accessing the database on every filter change.

Because the data I needed to collect was spread across three tables (student, answerInstance and question) I looked online for the syntax involved in of using multiple Join statements in query and then tested them using MySQL Workbench:

<https://dev.mysql.com/doc/refman/5.7/en/join.html>

	correct	date	_topicID	_studentID	_classID	firstName	lastName
▶	0	2017-02-22 13:22:00	9	5391	13A	Joe	Rackham
	0	2017-02-22 13:21:59	10	5391	13A	Joe	Rackham
	0	2017-02-22 13:21:57	9	5391	13A	Joe	Rackham
	0	2017-02-22 13:21:57	9	5391	13A	Joe	Rackham
	0	2017-02-22 13:21:57	9	5391	13A	Joe	Rackham
	0	2017-02-22 13:21:56	9	5391	13A	Joe	Rackham
	0	2017-02-22 13:21:54	7	5391	13A	Joe	Rackham
	0	2017-02-22 13:21:54	7	5391	13A	Joe	Rackham
	0	2017-02-22 13:21:53	7	5391	13A	Joe	Rackham
	0	2017-02-22 13:21:52	7	5391	13A	Joe	Rackham
	0	2017-02-22 13:21:52	7	5391	13A	Joe	Rackham
	0	2017-02-22 13:21:52	7	5391	13A	Joe	Rackham
	0	2017-02-22 13:21:52	7	5391	13A	Joe	Rackham

This was the result to the query:

```
SELECT answerinstance.correct, answerinstance.date, question._topicID, answerinstance._studentID, student._classID, student.firstName, student.lastName FROM answerinstance INNER JOIN question ON answerinstance._questionID = question.questionID INNER JOIN student ON answerinstance._studentID = student.studentID;
```

The Query successful collects all the data needed for the results scene in one query output.

I could then implement the query in a new subroutine in the MySQL controller script

```
public string[,] gatherResults () {  
    // Correct, Date, Topic, student ID, Class, Name  
    query = "SELECT answerinstance.correct, answerinstance.date, question._topicID, " +  
            "answerinstance._studentID, student._classID, student.firstName, student.lastName " +  
            "FROM answerinstance INNER JOIN question ON answerinstance._questionID = " +  
            "question.questionID INNER JOIN student ON answerinstance._studentID = student.studentID;";  
    command = new MySqlCommand (query, connection);  
    reader = command.ExecuteReader();  
    int count = 0;  
    while (reader.Read ()) {  
        count++;  
    }  
    string[,] results = new string[count,6];  
    count = 0;  
    reader.Close ();  
    reader = command.ExecuteReader ();  
    while(reader.Read()) {  
        results[count,0] = reader.GetString(0);  
        results[count,1] = reader.GetString(1);  
        results[count,2] = reader.GetString(2);  
        results[count,3] = reader.GetString(3);  
        results[count,4] = reader.GetString(4);  
        results[count,5] = reader.GetString(5) + " " + reader.GetString(6);  
        count++;  
    }  
    reader.Close ();  
    return results;  
}
```

The query collects all the answers stored in the database

Student ID is the primary key for students so their name will be used for clarity for the user and not for identification. I therefore felt comfortable concatenating first and last names.

I then needed to get this data populated into the Results scene. By changing the function call to the MySQL script I could easily make results information appear in the place of Questions data.

```
results = databaseScript.gatherResults();  
dataItems = new DataItem[results.Length / 6];  
for (int i=0; i<dataItems.Length; i++) {  
    dataItems [i] = new DataItem (results [i, 1], results [i, 3]);  
}
```

-Name-	-Data-
2/14/2017 12:25:22 PM	5391
2/15/2017 1:41:05 PM	1234
2/22/2017 1:21:54 PM	5391
3/22/2017 10:44:31 AM	5169

Sorting the data into the different filters required proved to be an interesting challenge. My first attempt involved writing a 'Filter Results' subroutine that would arrange data based on three different axes. The name index would be the index in the results array used in the name column of the table (i.e. student, class). The idea was that all the different values in this index would be stored in a list together. For every value in this list the subroutine would then search for results entry's and calculate the statistic to be used in the table.

```
public string[,] FilterResults (int nameAxis, int dataAxis, int groupingAxis) {  
    filterOptions = new List<string>();  
    for (int i=0; i<results.Length/6; i++) {  
        if (filterOptions.Contains (results[i, nameAxis]) == false) {  
            filterOptions.Add (results [i, nameAxis]);  
        }  
    }  
    string[,] filteredResults = new string[filterOptions.Count,2];  
    for (int i=0; i<filteredResults.Length/2; i++) {  
        for (int j=0; j<results.Length/6; j++) {  
        }  
        //dataItems [i] = new DataItem (filterOptions [i], " ");  
    }  
    return filteredResults;  
}
```

This loop finds all the distinct values in the names index and adds them to the 'filterOptions' list.

This loop was intended to match results to their name but the solution failed to effectively materialise

This approach failed because of the fact that in some situations a student's or classes name would still be used in the table but only their results for a specific topic or on a specific day would be collected. I attempted to deal with this by introducing the grouping index but because of the fact that this index would only be needed some of the time I couldn't find a way to successfully implement this solution.

Upon reflecting on the failed solution I reached the conclusion that the problem of filtering the data was twofold.

1. Removing the irrelevant data from the data set
2. Attributing each result with the entity (student, class or date) that created it

I saw this chance to decompose the problem to be potentially useful so I wrote a subroutine to tackle problem 1 and remove irrelevant data.

```
public string[,] reduceDataSet (int axis, string value, string[,] inputData) {  
    int count = 0;  
    for (int i = 0; i < inputData.Length / 6; i++) {  
        if (inputData [i, axis] == value) {  
            count++;  
        }  
    }  
    string[,] usedResults = new string[count,6];  
    count = 0;  
    for (int i = 0; i < inputData.Length / 6; i++) {  
        if (inputData [i, axis] == value) {  
            usedResults [count,0] = inputData [i,0];  
            usedResults [count,1] = inputData [i,1];  
            usedResults [count,2] = inputData [i,2];  
            usedResults [count,3] = inputData [i,3];  
            usedResults [count,4] = inputData [i,4];  
            usedResults [count,5] = inputData [i,5];  
            count++;  
        }  
    }  
    return usedResults;  
}
```

The subroutine only needs to take two parameters, the relevant value and the index that value appears in. However, by taking input data as a parameter instead of using the global results variable the subroutine can be applied multiple times to refine the data set

This loops runs through all the elements of the results array. If they have the required value they are copied over to the used results array. At the end of looping the subroutine return an array with only the relevant results

I then moved on to solving problem 2. I adapted the foundation created for the 'FilterResults' subroutine safe in the knowledge that this wouldn't need to filter out unnecessary data.

```
public string[,] FilterResults (int nameAxis, int dataAxis, string[,] inputData) {  
    List<string> filterOptions = new List<string>();  
    for (int i=0; i<inputData.Length/6; i++) {  
        if (filterOptions.Contains (inputData[i, nameAxis]) == false) {  
            filterOptions.Add (inputData[i, nameAxis]);  
        }  
    }  
    string[,] filteredResults = new string[filterOptions.Count,2];  
    for (int i=0; i<filterOptions.Count; i++) {  
        int count = 0;  
        float total = 0f;  
        for (int j=0; j<inputData.Length/6; j++) {  
            if (inputData[j,nameAxis] == filterOptions[i]) {  
                count = count + 1;  
                if (inputData [j, dataAxis] == "True")  
                    total = total + 1f;  
            }  
        }  
        Debug.Log (filterOptions [i] + " Total: " + total + " Count: " + count);  
        filteredResults [i, 0] = filterOptions [i];  
        filteredResults [i, 1] = string.Format ("{0:N3}", total / count);  
    }  
    return filteredResults;  
}
```

The subroutine now needs to take 'inputData' as a parameter because the subroutine now needs to handle the reduced data set instead of just the results collected from the database

The loop increases the count variable for every relevant answer and the total variable for every correct answer. The value of total / count it then used as the data to be displayed

The value of 'total / count' is a float so string.Format is needed to convert it to a string

The new subroutines could be used in the following pattern at the start of the run loop to produce the displayed result.

```
public IEnumerator RunLoop () {
    string[,] data = reduceDataSet (3,"5391",results);
    data = FilterResults (2, 0, data);
    dataItems = new DataItem[data.Length/2];
    for (int i = 0; i < dataItems.Length; i++) {
        dataItems [i] = new DataItem (data [i, 0], data [i, 1]);
    }
    Populate ();
}
```

-Name-	-Data-
10	1.000
9	0.125
8	0.000
11	0.000
12	0.667

The data displayed is the total data set reduced to only those answer instances attributed to student '5391' and then filtered by topic. This is the expected result of the code I wrote into the RunLoop.

With the mechanisms in place to display data correctly I then needed to link up the game engine with the script to allow for user input. Upon coming to the problem I decided the two dropdown menu system that I had put as a place holder in the game engine was too restrictive and the headings I created weren't particular intuitive. I instead opted to try and implement the solution proposed below.

Whole: Topic:

Whole	Student: General	Topic 1
Class	Student: Topic	Topic 2
Student	Class: General	Top...
	Class: Topic	...

Class:

10A: Topic
10A: General
10B: Top...
...

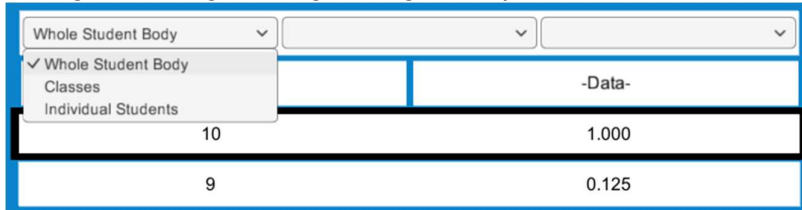
Student:

Joe Rackham 5391: General
Joe Rackham 5391: Topic
Myles ...
...

The idea behind this system was that there are three tiers of dropdown menu

- The first dropdown allows the user to select whether they want to look at the whole student base, one class or just one student
- The contents of the second dropdown depends on the contents of the first:
 - If the first dropdown holds 'Whole' the second will contain options to view individual students or classes either generally or in a specific topic
 - If the first dropdown holds 'Class' the second will contain options to view the students from every class either generally or in a specific topic
 - If the first dropdown holds 'Student' the second will contain options to view the result of individual students generally or in a specific topic
- The third dropdown will only appear if the user selects they want to see results by topic and will allow the user to choose the topic

I changed the design in the game engine and passed these now elements into the script.



	-Data-
10	1.000
9	0.125

```
public Dropdown dropdownA;  
public Dropdown dropdownB;  
public Dropdown dropdownC;
```

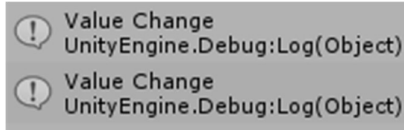
The arrangement of the dropdowns needed to change as their value changed. To implement this I looked at the Unity documentation for a way to call a subroutine when a change took place and discovered listeners. This 'listens' for a specific event on the component it is attached to and then calls a subroutine:

<https://docs.unity3d.comScriptReference/UI.InputField-onValueChanged.html>

I experimented with implementing this by adding a listener to the first dropdown that called a simple subroutine. This arrangement worked as I expected and I decided to try and use listeners to manage the dropdowns.

```
dropdownA.onValueChanged.AddListener(delegate {ValueChangeCheck ();});
```

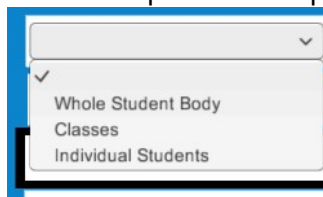
```
public void ValueChangeCheck () {  
    Debug.Log ("Value Change");  
}
```



This line adds a listener to the event 'OnValueChanged' to the first dropdown. When the listener is triggered it calls the subroutine Value Changed Check.

The listener worked and displayed the message every time the dropdown was changed.

I began to control the dropdowns by first writing a subroutine to handle a change to the value of dropdown A. Dropdowns have to have a default value, so, to allow the user to select the first option and still register a value change I added a blank option at the top of the dropdown.



I also added statements to the 'Start' subroutine setting dropdowns B and C to be an active state of false. The idea behind this was that the UI would be easier to interact with if dropdowns only appeared when needed.

```
dropdownB.gameObject.SetActive (false);  
dropdownC.gameObject.SetActive (false);
```

Part of the subroutine could then be written without changing any other part of the code:

```
public void DDChange () {  
    if (dropdownA.options.Count == 4) {  
        dropdownA.ClearOptions ();  
        dropdownA.AddOptions (new List<string> (new string[] {"Whole " +  
            "Student Body", "Classes", "Individual Students"}));  
        dropdownA.value = dropdownA.value - 1;  
    }  
    if (dropdownA.value == 0) {  
        dropdownB.gameObject.SetActive (true);  
        dropdownB.ClearOptions ();  
        dropdownB.AddOptions (new List<string> (new string[] {"Student: " +  
            "General", "Student: Topic", "Class: General", "Class: Topic"}));  
    }  
    if (dropdownA.value == 1) {  
        dropdownB.gameObject.SetActive (true);  
        dropdownB.ClearOptions ();  
    }  
}
```

This first if statement removes the blank option if needed.

For 'Whole Student Body' the options of dropdown B are always the same and can be directly placed it.

For the options 'Classes' and 'Individual Students' dropdown B would need to be populated with information from the database. This left me with two options:

1. Search through the already collected results array for distinct students and classes
2. Write subroutines in the MySQL script to find this information

I opted for option 2 because of the following reasons:

- Searching through the results wouldn't discover any students who don't have any answers associated with them, however, a teacher would still want to these students to be represented in the system so they can discover this inactivity
- Once the database is populated with a significant amount of answers searching through the results array would, in all likelihood, take longer than simply retrieving the relevant data from the database.
- Data in the database is sorted based on the primary keys, this means classes and students can be ordered without me having to implement sorting myself.

To do this I wrote a subroutine in the MySQL script for the purpose of finding the primary keys of all the records in a table.

```
public string[] findAll (string tableName) {  
    query = "SELECT * FROM " + tableName + ";";  
    command = new MySqlCommand (query, connection);  
    reader = command.ExecuteReader();  
    int count = 0;  
    while (reader.Read ()) {  
        count++;  
    }  
    string[] instances = new string[count];  
    count = 0;  
    reader.Close ();  
    reader = command.ExecuteReader ();  
    while(reader.Read()) {  
        instances[count,0] = reader.GetString(0);  
        count++;  
    }  
    reader.Close ();  
    return instances;  
}
```

This subroutine very closely mimics the pattern followed by other in the MySQL script. It reads all the records from the specified table and returns an array of their primary keys.

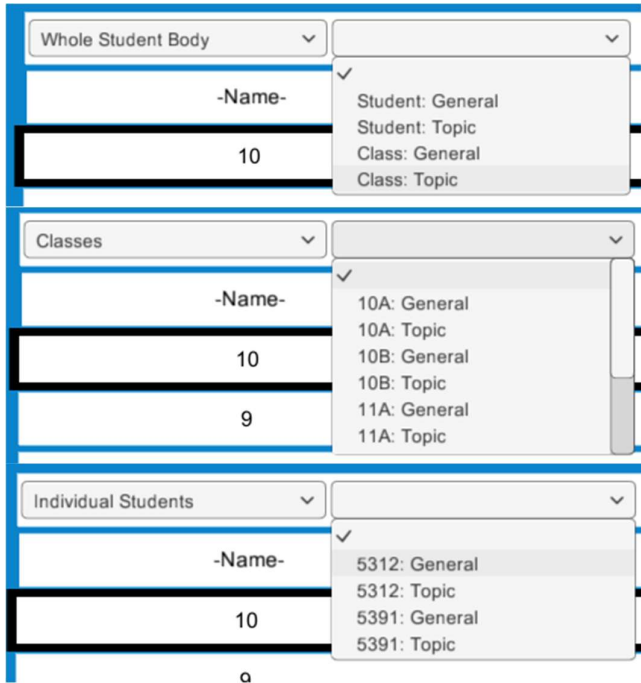
I could then use this to complete the dropdown A control subroutine.

```
public void DDACHange () {
    if (dropdownA.options.Count == 4) {
        dropdownA.ClearOptions ();
        dropdownA.AddOptions (new List<string> (new string[] {
            "Whole Student Body", "Classes", "Individual Students"}));
        dropdownA.value = dropdownA.value - 1;
    }
    dropdownB.gameObject.SetActive (true);
    dropdownB.ClearOptions ();
    if (dropdownA.value == 0) {
        dropdownB.AddOptions (new List<string> (new string[] {
            " ", "Student: General", "Student: Topic", "Class: General", "Class: Topic"}));
    }
    if (dropdownA.value == 1) {
        dropdownB.AddOptions (new List<string> (new string[] {""}));
        for (int i=0; i<classes.Length; i++) {
            dropdownB.AddOptions (new List<string> (new string[] {
                classes [i] + ": General", classes [i] + ": Topic" }));
        }
    }
    if (dropdownA.value == 2) {
        dropdownB.AddOptions (new List<string> (new string[] {""}));
        for (int i=0; i<students.Length; i++) {
            dropdownB.AddOptions (new List<string> (new string[] {
                students[i] + ": General", students [i] + ": Topic" }));
        }
    }
    dropdownB.value = 0;
}
```

These lines in the start subroutine retrieve the relevant data.
 students = databaseScript.findAll ("student");
 classes = databaseScript.findAll ("class");

The code loops through all the distinct records and adds two options to dropdown B. One for general and one for topic.

The subroutine worked and correctly populated dropdown B in the three different scenarios



Whole Student Body
 Dropdown B contains the 4 constant options

Classes
 Dropdown B contains a general and topic option for all the different classes

Individual Students
 Dropdown B contains a general and topic option for all the different students

To implement dropdown B in followed a similar pattern for dropdown A by initially trying to create the framework for the subroutine without changing any other part of the code.

```
public void DDBChange () {  
    if (dropdownB.options[0].text == "") {  
        int value = dropdownB.value;  
        dropdownB.options.RemoveAt (0);  
        dropdownB.value = value - 1;  
    }  
}
```

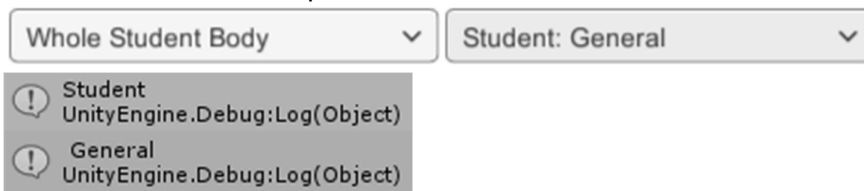
Because the list of options is variable I can't remove the blank option by resetting the options. Instead I need to remove the first index of the options list.

I then needed to split the option heading into the different pieces of data contained within.

```
string[] option = dropdownB.captionText.text.Split (':');  
for (int i = 0; i < option.Length; i++) {  
    Debug.Log (option[i]);  
}  
}
```

This line splits the option heading into an array, splitting it every time a ':' occurs.

The result of this code was too correctly split the string into the first part, which can be used to identify the specific record and the second part, which can be used to determine if the option selected is 'General' or 'Topic'.



Using this splitting I could implement the 'reduce' and 'filter' subroutines into the dropdown B code to correctly select the data items to be displayed.

```
public void DDBChange () {  
    if (dropdownB.options[0].text == "") {  
        int value = dropdownB.value;  
        dropdownB.options.RemoveAt (0);  
        dropdownB.value = value - 1;  
    }  
    string[] option = dropdownB.captionText.text.Split (':');  
    if (option [option.Length - 1] == " General") {  
        string[,] data = new string[0,0];  
        if (dropdownA.value == 0) {  
            if (option [0] == "Student") {  
                data = FilterResults (3, 0, results);  
            } else {  
                data = FilterResults (4, 0, results);  
            }  
        } else if (dropdownA.value == 1) {  
            data = reduceDataSet (4, option [0], results);  
            data = FilterResults (3, 0, data);  
        } else {  
            data = reduceDataSet (3, option [0], results);  
            data = FilterResults (1, 0, data);  
        }  
        dataItems = new DataItem[data.Length/2];  
        for (int i = 0; i < dataItems.Length; i++) {  
            dataItems [i] = new DataItem (data [i, 0], data [i, 1]);  
        }  
        Populate ();  
    }  
    else if (option [option.Length - 1] == " Topic") {  
        dropdownC.gameObject.SetActive (true);  
    }  
}
```

If dropdown A is set to 'Whole Student Body' the data set doesn't need to be reduced and can simply be filtered by student or class. However, if dropdown A has another value the data set must first be reduced. This can be easily done using the first part of the split string.

If the option selected ends in topic the system doesn't have all the required information to filter results. This needs to be handled in the subroutine controlling dropdown C.

This worked to correctly arrange data as shown in the following examples:

- | -Name- | -Data- |
|--------|--------|
| 5312 | 0.049 |
| 5391 | 0.250 |
- | -Name- | -Data- |
|--------|--------|
| 13A | 0.091 |
- | -Name- | -Data- |
|-----------------------|--------|
| 3/30/2017 11:36:38 AM | 1.000 |
| 3/30/2017 11:37:09 AM | 1.000 |

However I wasn't satisfied with the way that results were displayed for individual students. Because the 'Date' data type in MySQL records down to the minute answers recorded minutes apart would be represented as an entirely different use of the system. This meant that there was far too many different data items and the data was difficult to interpret. To rectify this I decide to alter the 'Gather Results' subroutine in the MySQL script to cut the date data short.

```
results[count,1] = reader.GetString(1).Substring(0,9);
```

This line correctly reduced the date but I then noticed that the data was in the American format.

3/27/2017	0.250
-----------	-------

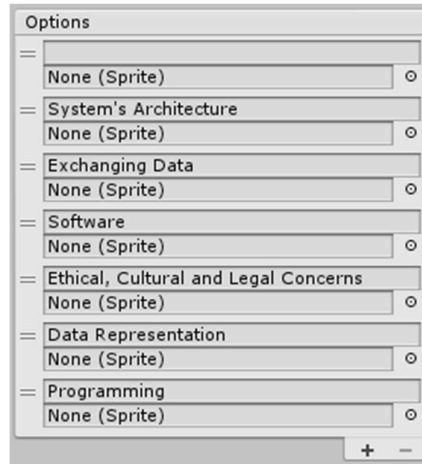
To fix this I looked at the formatting functions available in MySQL. I ended up using the DATE_FORMAT() function which allowed me to specify the way the date data was retrieved. This allowed me to arrange the date in the English format and also negated the need to cut the string in the MySQL script.

```
query = "SELECT answerinstance.correct, DATE_FORMAT(answerinstance.date, '%d/%m/%Y'), question._topicID, " +  
"answerinstance._studentID, student._classID, student.firstName, student.lastName " +  
"FROM answerinstance INNER JOIN question ON answerinstance._questionID = " +  
"question.questionID INNER JOIN student ON answerinstance._studentID = student.studentID";
```

27/03/2017	0.250
09/04/2017	1.000

This was exactly how I wanted the date to be displayed. With this change implemented all the functionality for dropdown B was working.

I could then move on to controlling dropdown C. Because dropdown C would only be used for displaying the topic options I could enter these in the game designer and not ever have to change them.



I then began to write the dropdown C subroutine.

```
public void DDCChange () {
    if (dropdownC.options[0].text == "") {
        int value = dropdownC.value;
        dropdownC.options.RemoveAt (0);
        dropdownC.value = value - 1;
    }
    string[,] dataPT1 = reduceDataSet (2,dropdownC.value.ToString(),results);
    string[,] dataPT2 = reduceDataSet (2,(dropdownC.value+6).ToString(),results);
    string[,] data = new string[(dataPT1.Length/6)+(dataPT2.Length/6),6];
    int halfway = dataPT1.Length / 6;
    for (int i = 0; i < data.Length/6; i++) {
        if (i < halfway) {
            data [i, 0] = dataPT1 [i, 0];
            data [i, 1] = dataPT1 [i, 1];
            data [i, 2] = dataPT1 [i, 2];
            data [i, 3] = dataPT1 [i, 3];
            data [i, 4] = dataPT1 [i, 4];
            data [i, 5] = dataPT1 [i, 5];
        } else {
            data [i, 0] = dataPT2 [i - halfway, 0];
            data [i, 1] = dataPT2 [i - halfway, 1];
            data [i, 2] = dataPT2 [i - halfway, 2];
            data [i, 3] = dataPT2 [i - halfway, 3];
            data [i, 4] = dataPT2 [i - halfway, 4];
            data [i, 5] = dataPT2 [i - halfway, 5];
        }
    }
    string[] option = dropdownB.captionText.text.Split (':');
    if (dropdownA.value == 0) {
        if (option [0] == "Student") {
            data = FilterResults (3, 0, data);
        } else {
            data = FilterResults (4, 0, data);
        }
    }
    } else if (dropdownA.value == 1) {
        data = reduceDataSet (4, option [0], data);
        data = FilterResults (3, 0, data);
    } else {
        data = reduceDataSet (3, option [0], data);
        data = FilterResults (1, 0, data);
    }
    }
    dataItems = new DataItem[data.Length/2];
    for (int i = 0; i < dataItems.Length; i++) {
        dataItems [i] = new DataItem (data [i, 0], data [i, 1]);
    }
    Populate ();
}
```

Similarly to dropdown B the blank first item is removed when the dropdown is used.

GCSE and A-Level versions of different topics are stored as different records in the database. Therefore, to find all the answer instances for a specific topic the code uses reduce to find the two separate groups of answers then combines them. Because the arrays used to store results are multidimensional the .Copy() procedure concatenate arrays doesn't work so the two arrays are combined manually using a loop.

With the data reduced based on the topic, it can then be further reduced using the information from dropdown B in a largely similar way the dropdown B control subroutine.

This solution compiled and ran without error, however I noticed a problem with the data displayed. The results attributed with different topics didn't seem to match up with my own experience of using the game. To test this I created a new student and answered only one question.

	answerInstanceID	correct	date	_studentID	_questionID
▶	81	1	2017-04-09 20:49:02	1234	10
	17	1	2017-03-30 11:36:38	5312	42
	18	1	2017-03-30 11:37:09	5312	8

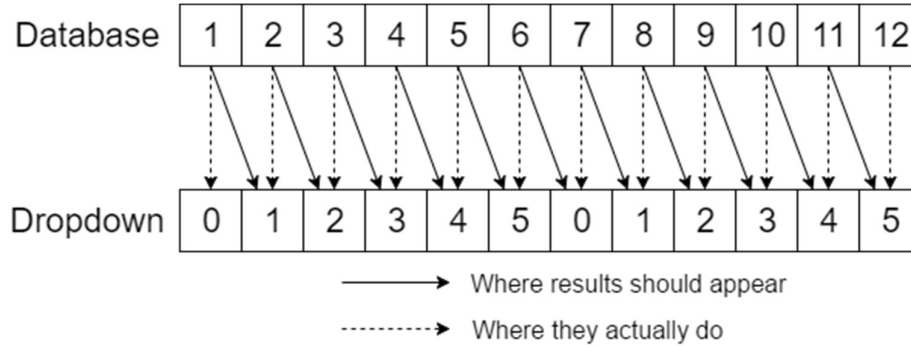
Student 1234's only answer was to question 10 which was in topic 6, 'GCSE programming'.

10	Which type of error in code causes a program t...	SYNTAX	6
----	---	--------	---

However this answer was taken into account when dropdown C was set to find results from GCSE or A-Level System's architecture.

Whole Student Body		Student: Topic		System's Architecture	
-Name-			-Data-		
1234			1.000		

To rectify this, I compared the dropdown C code with the topic table in the database and discovered that the issue was occurring because the primary keys of the topics took values between 1 and 12 which didn't properly map to the value of the dropdown which begins at 0. The diagram below illustrates this issue:



Upon changing the lines that used the dropdown value the results appeared to displayed correctly.

Whole Student Body		Student: Topic		Programming	
-Name-			-Data-		
1234			1.000		
5312			0.000		
5391			0.833		

With the third dropdown working the process of retrieving results was complete. The next step to completing the results scene was to implement graphing.

I initially looked into the idea of using charting API or asset package to create the graphs I wanted. Implementing one of these would mean that I would be able to use premade and presented code that I knew would work; I wouldn't have to 'reinvent the wheel' to solve a common problem. Several of these kinds of software exist however I wasn't able to find one suitable. Most of the available graphing systems had a cost to them and any others that I could find would be able to easily merge Unity and my existing application. I discussed this issue with my client and we reached the conclusion that the only viable solution was for me to develop the ability to create graphs myself. This places more of a burden on me as it means I can't work down a development path that would massively expedite the process of graphing, however, it is a necessary decision to ensure the application isn't missing an essential part. A positive of the fact that I can't use any external tools is that I will have a more concrete understanding of how my application works which will put me in a better position to fix any errors that occur:

<https://www.assetstore.unity3d.com/en/#!/content/11782>

<https://www.microsoft.com/en-gb/download/details.aspx?id=14422>

<https://developers.google.com/chart/>

I first created a game object in the engine to be used as a 'prefab' to instantiate the bars from.



Having enjoyed success with using a class to represent the different rows in the table I began writing a 'Bar' class to do the same for bars in the graphs.

```
class Bar {
    private Text barText;
    private Image barImage;
    private RectTransform textTransform;
    private RectTransform imageTransform;
    private float barHeight;
    private GameObject instance;
    public Bar (GameObject prefab, string text, float height) {
        instance = Instantiate(prefab);
        barText = instance.transform.Find ("Text").GetComponent<Text> ();
        textTransform = instance.transform.Find ("Text").GetComponent<RectTransform> ();
        barImage = instance.transform.Find ("Image").GetComponent<Image> ();
        imageTransform = instance.transform.Find ("Image").GetComponent<RectTransform> ();
        barHeight = height;
        barText.text = text;
    }
    public void move(float x, float y) {
        textTransform.localPosition = new Vector3 (x, y, 0f);
        imageTransform.localPosition = new Vector3 (x, y, 0f);
    }
    public void resize(float width) {
        textTransform.localScale = new Vector3 (width,barHeight*440f,1f);
        imageTransform.localScale = new Vector3 (width,barHeight*440f,1f);
    }
    public void destroy() {
        Destroy (instance);
    }
}
```

The class constructor instantiates and instance of the prefab and finds the relevant components needed for the bar's function.

The idea behind the bars is that the data value, because it can only take values between 0 and 1, is multiplied by the maximum height to the bar to make it representative of the data.

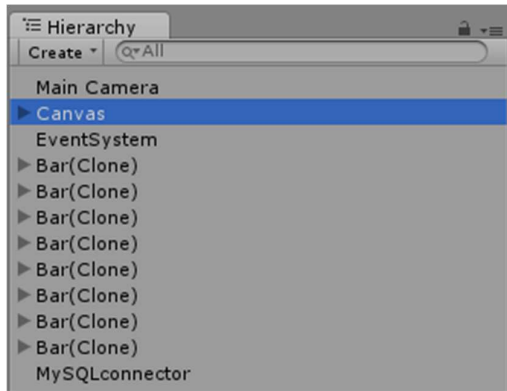
A destroy method is needed because the game object is only manipulated by the script through pass by reference and if the Bar Object is destroyed the in engine objects will persist.

I then made a temporary adjustment to the 'Populate' subroutine to instantiate some bars and test the functionality of the subroutine.

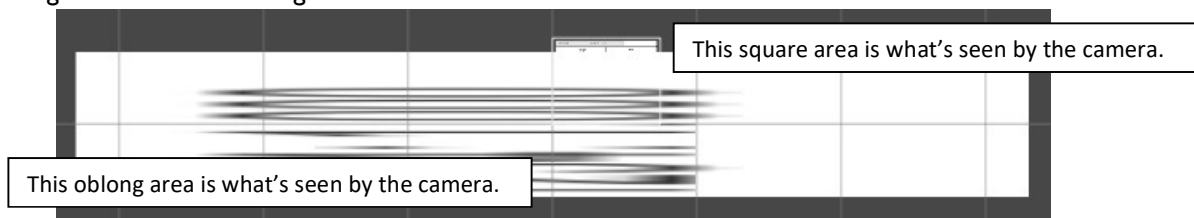
```
public void Populate () {
    for (int i=0; i<rows.Length; i++) {
        if (i + displayStartIndex < dataItems.Length) {
            rows [i].setName (dataItems [i+displayStartIndex].getName ());
            rows [i].setData (dataItems [i+displayStartIndex].getData ());
        } else {
            rows [i].setName (string.Empty);
            rows [i].setData (string.Empty);
        }
    }
    for (int i=0; i<dataItems.Length; i++) {
        Bar j = new Bar (barPrefab,dataItems[i].getName() + ": " + dataItems[i].getData(),0.75f);
        j.resize (10f);
    }
}
```

Unlike the rows there can be as many bars as needed. This loop creates a bar for each data item and uses the resize method with the intention to make the bar have a width of 10.

The result of this was only partially successful. Firstly, one Bar per data item was successfully instantiated but because they were instantiated out of the UI canvas in the game object hierarchy. This meant that although the Bars were present they wouldn't appear.



Secondly, when I manually put one of the Bars into the canvas it was massively stretched to the wrong size and in the wrong direction.



To fix the first issue I passed the Canvas as a public variable to the script. I could then set the parent of the bar instance to the canvas in the constructor to ensure the UI element would be rendered.

```
public Bar (GameObject prefab, string text, float height, GameObject parent) {  
    instance = Instantiate(prefab);  
    barText = instance.transform.Find ("Text").GetComponent<Text> ();  
    textTransform = instance.transform.Find ("Text").GetComponent<RectTransform> ();  
    barImage = instance.transform.Find ("Image").GetComponent<Image> ();  
    imageTransform = instance.transform.Find ("Image").GetComponent<RectTransform> ();  
    barHeight = height;  
    barText.text = text;  
    instance.transform.parent = parent.transform;  
}
```

The last line of the constructor makes the instance a child of the canvas.

I then realised that the sizing issue occurred because I was using the 'local scale' value, which affects the scale of the game object instead of the actual dimensions. Changing the resize subroutine to use the 'size delta' value instead gave the desired result.

```
public void resize(float width) {  
    textTransform.sizeDelta = new Vector2 (barHeight * 440f, width);  
    imageTransform.sizeDelta = new Vector2 (barHeight * 440f, width);  
}
```

To make the font size of the text scale as the bar width changed I turned on the 'Best Fit' setting in the Text component of the prefab. This setting meant that the font would be as large as possible without escaping the boundaries of the bar.

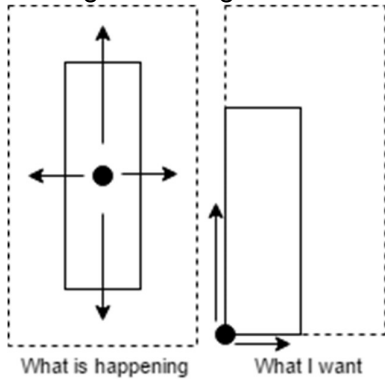


With these changes in place the bars were instantiated as intended and I could manually move them away from each other in the game engine. However, as show on the third bar, when the bar is very short the text won't appear completely and will be very small. I decided to change the line that resized the text box so the text box would always be max height. This meant that the text wouldn't necessarily fit inside the bar but all the text bars would be a uniform size.

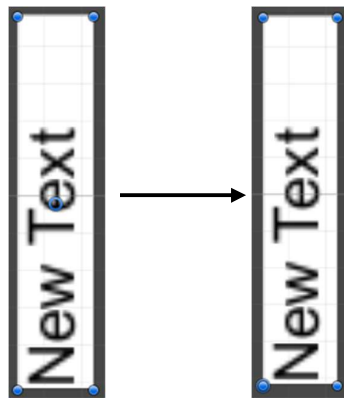
```
public void resize(float width) {
    textTransform.sizeDelta = new Vector2 (440f, width);
    imageTransform.sizeDelta = new Vector2 (barHeight * 440f, width);
}
```



This change to the procedure corrected the sizing issue but I then realised that the bars were resizing away from the centre of the image instead of from the bottom as I envisaged. The diagram below illustrates this problem:



To change this I needed to adjust the anchor of the image component to the bottom right corner. So that both would be moved to the same place I also needed to change the anchor of the text component.



I moved the anchor of both components to the bottom left corner.

This made all the components that in the Bar appeared correctly in reference to each other. (Image Rotated)



With the Bar class complete I could move on to making the graphs actually appear properly. I first created a variable to hold the bars.

```
private Row[] rows;  
private Bar[] bars;  
private int focusRow;  
private int displayStartIndex;  
private List<string> filterOptions;
```

I then split the 'Populate' subroutine into two different parts one for rows and one for graphs.

```
public void RowPopulate () {  
    for (int i=0; i<rows.Length; i++) {  
        if (i + displayStartIndex < dataItems.Length) {  
            rows [i].setName (dataItems [i+displayStartIndex].getName ());  
            rows [i].setData (dataItems [i+displayStartIndex].getData ());  
        } else {  
            rows [i].setName (string.Empty);  
            rows [i].setData (string.Empty);  
        }  
    }  
}
```

This first subroutine just has the row related code from before.

```
public void GraphPopulate () {  
    for (int i=0; i < bars.Length; i++) {  
        bars [i].destroy ();  
    }  
    bars = new Bar[dataItems.Length];  
    for (int i=0; i<dataItems.Length; i++) {  
        bars[i] = new Bar (barPrefab,dataItems[i].getName() + ": " +  
            dataItems[i].getData(),float.Parse(dataItems[i].getData()), canvas);  
        float width = 345 / dataItems.Length;  
        bars[i].resize (width);  
        bars [i].move (-340+i*width,-205);  
    }  
}
```

Because the number of bars is variable I decided it would be easier to simply destroy all the existing bars when the graph is changed.

By dividing the total width of the screen by the number of bars the subroutine calculates a width value used to resize and place the bars.

I then changed all existing references to the defunct 'Populate' subroutine to reference 'RowPopulate' and then added a function call to 'GraphPopulate' whenever the list of data items changed (In the dropdown B and C subroutines).

However when I first attempted to run this code I encountered an error.

```
! NullReferenceException: Object reference not set to an instance of an object  
ResultsController.GraphPopulate () (at Assets/Scripts/ResultsController.cs:154)
```

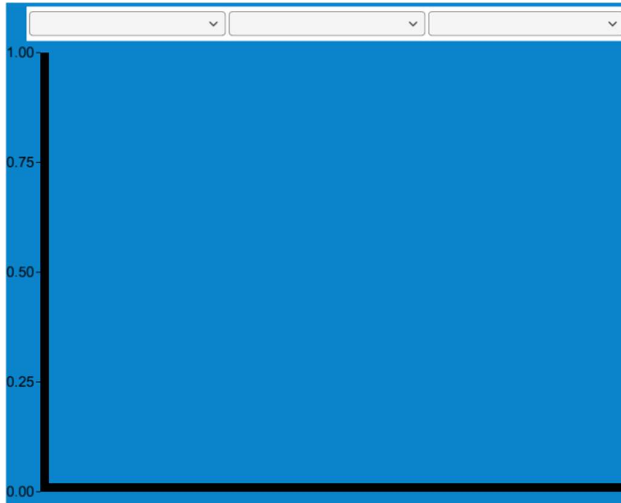
A null reference exception occurs when something is referenced in code that doesn't exist. By clicking on the error I discovered that the problematic line was the highlighted one below:

```
public void GraphPopulate () {  
    for (int i=0; i < bars.Length; i++) {  
        bars [i].destroy ();  
    }  
}
```

I deduced that the error occurred because at the start of the code the bars variable had no value assigned to it so the 'bars.Length' statement fails. To rectify this I changed the 'Start' subroutine to give bars an initial value.

```
for (int i=0; i<display.Length; i++) {  
    rows [i] = new Row (display[i]);  
}  
bars = new Bar[0];
```

This prevented the error from occurring.



I then decided to create the game objects that would make up the axes for the graph and the mechanism to move between the table and graph views. Otherwise elements would be laid on top of each other which would make it confusing to determine if the system was working.

To allow the bars and rows to be shown and hidden I added a similar 'Toggle Active' method to both classes.

```
public void toggleActive(bool state) {  
    rowParent.gameObject.SetActive (state);  
}  
public void toggleActive(bool state) {  
    instance.gameObject.SetActive (state);  
}
```

I then passed the graph game objects as a variable to the script and wrote a subroutine to display the graph.

```
public IEnumerator RunGraph() {  
    for (int i = 0; i < rows.Length; i++) {  
        rows [i].toggleActive (false);  
    }  
    for (int i = 0; i < bars.Length; i++) {  
        rows [i].toggleActive (true);  
    }  
    graph.gameObject.SetActive (true);  
    dataOutline.gameObject.SetActive (false)  
    while (enter == false) {  
        yield return null;  
    }  
    enter = false;  
    for (int i = 0; i < rows.Length; i++) {  
        rows [i].toggleActive (true);  
    }  
    for (int i = 0; i < rows.Length; i++) {  
        rows [i].toggleActive (false);  
    }  
    graph.gameObject.SetActive (false);  
    dataOutline.gameObject.SetActive (true);  
}
```

The two blocks before and after the while loop do the opposite of each other, the first loops through all bars and rows to display the graph and the second displays the table.

The graph elements and outline for data items also needed to be toggled.

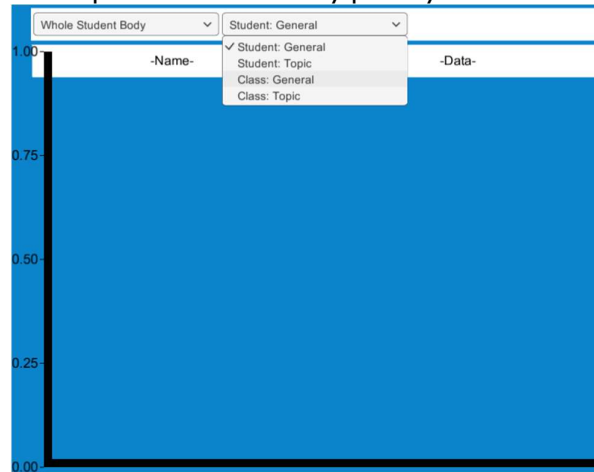
The subroutine loops, and the graph continues to be shown, until enter is pressed.

I then changed added another if statement to the 'RunLoop' subroutine to handle calling 'RunGraph'.

```
if (enter == true) {  
    enter = false;  
    yield return StartCoroutine (RunGraph());  
    yield return null;  
}
```

Enter is also used as the user input key to show the graph.

This implementation was only partially successful:



- The graph axes were shown and the outline was hidden
- All the rows were successfully hidden but the bars remained deactivated
- The Name and Data headings remained active
- Pressing enter caused the more recently used dropdown to open

I first made some changes to the 'RunGraph' subroutine to fix some of these problems.

```
public IEnumerator RunGraph() {  
    for (int i = 0; i < rows.Length; i++) {  
        rows [i].toggleActive (false);  
    }  
    for (int i = 0; i < bars.Length; i++) {  
        bars [i].toggleActive (true);  
    }  
    graph.gameObject.SetActive (true);  
    dataOutline.gameObject.SetActive (false);  
    heading.gameObject.SetActive (false);  
    while (g == false) {  
        yield return null;  
    }  
    g = false;  
    for (int i = 0; i < rows.Length; i++) {  
        rows [i].toggleActive (true);  
    }  
    for (int i = 0; i < bars.Length; i++) {  
        bars [i].toggleActive (false);  
    }  
    graph.gameObject.SetActive (false);  
    dataOutline.gameObject.SetActive (true);  
    heading.gameObject.SetActive (true);  
    up = false;  
    down = false;  
}
```

I passed the name and data heading together under a parent game object to the script to also be toggled.

Instead of enter I decided to use the 'g' key to control user input as it wouldn't have any other use on this screen.

Some of the reference to bars in the script were actually referencing the rows. These discrepancies were the reason the bars weren't being toggled properly.

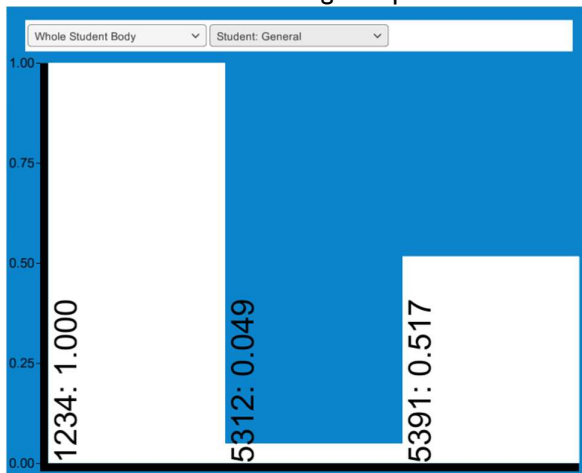
Setting up and down to false at the end of the subroutine prevents the outline moving when the table is redisplayed.

These changes resulted in all the problems described being fixed. However, now that the bars were appearing I saw that they weren't being displayed correctly.



- My understanding of how the x and y positions of the UI components was the opposite of the reality, the bars ended up all stacked on top of each other instead of side to side
- The value I used to calculate the width of bars wasn't accurate for the actual dimensions of the axes

I flipped the x and y values passed to the 'Move' method and adjusted some of the values used in the code. With these small changes in place the bars fit into the axes as intended.



To differentiate the bars I decided colour would be useful. I created some different coloured materials in the game engine to be passed to the script. Informed by my earlier research I made sure all the colours would suitably contrast with both the back ground and the text.

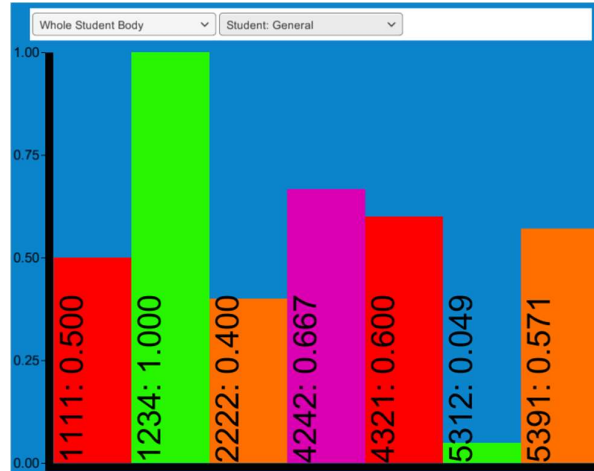


I added a material variable to the Bar class and changed the constructor call.

```
bars[i] = new Bar (barPrefab,dataItems[i].getName() + ": " +  
    dataItems[i].getData(),float.Parse(dataItems[i].getData()),  
    canvas, materials[i % materials.Length]);
```

The '%' function calculates the remainder, this ensures the materials repeat.

The result of this was coloured bars that were much easier to distinguish.



With this implemented I was satisfied with the way bar graphs were displayed and, therefore, the results section of the application was complete. Staff users can now view students' performance and usage, arranged in a variety of ways, in both numerical and graphical form.

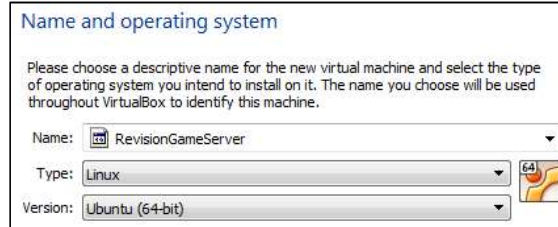
Client Feedback

When I presented this version of the application to my client he was satisfied with the way I had gone about meeting the success criteria we agreed on. My Byford believed I had met the brief that results should be displayed both graphically and numerically. He thought that the system of dropdown menus I implemented was a satisfactory solution to problem of displaying information in a variety of different ways. My Byford commented that the bar graphs looked to be particularly useful in the same way that the exam board provided bar graphs we discussed in our interview were. The conclusion of this discussion was agreement between me and Mr Byford that the development of the application was now complete as he was satisfied with the solutions I presented to all of the success criteria. My Byford therefore gave me consent to begin working on implementing the server and packaging the game before conducting alpha and beta testing before finally presenting the final version of the application to him to get his final comments before the application is implemented.

Creating and implementing the server

The next stage to completing my project was to setup the server. During the development of the application I had been connecting to the local version of the database running in MySQL workbench on my personal computer. However, as per the brief of the project the database needs to be stored on a separate machine on the school network. The way I decided to implement this was using an Ubuntu Virtual Machine.

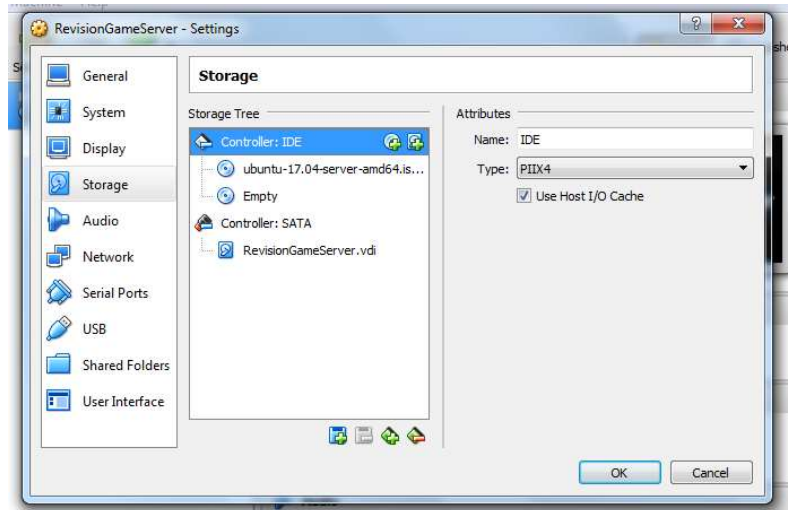
I first named and gave the type of the Operating system I wanted to emulate.



I then needed to tell the program how much RAM and Hard Disk Space could be allocated to the virtual machine. I had been given permission to host the server on an unused PC so there were no limitations of these values.

I then downloaded an Ubuntu ISO file I could install the server from. I downloaded the latest version of the ISO to make sure I had the most up to date and error free version of Ubuntu.

I could then mount the Ubuntu ISO file as a CD, from which, the Virtual Machine would boot.



I then began to configure the server settings. This involved relatively routine processed like setting my language, configuring my keyboard and setting up a login. I didn't need my server to do anything extreme or out of the ordinary so I could use the default options for partitioning the disks.

I then used the update commands to update all packages to the newest version.

1. apt-get update
2. apt-get upgrade

The next task was getting the MySQL script needed to setup an instance of my database onto the server. I achieved this using Samba, an interoperability system for transferring files between Windows and Linux systems.

1. apt-get install samba

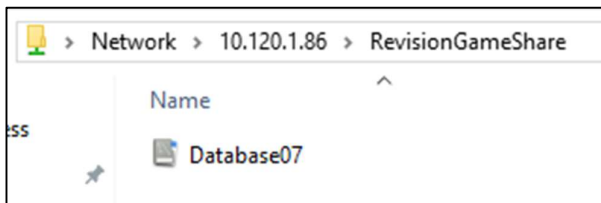
I could then create a new file share on the school network called RevisionGameShare. I edited the Samba configuration file to make this server editable by devices other than the host PC.

```
[global]
workgroup = WORKGROUP
server string = Revision Game Server %u
netbios name = ubuntu
security = user
map to guest = bad user
dns proxy = no

[RevisionGameShare]
path = /home/joe
browsable = yes
writable = yes
guest ok = yes
read only = no
force user = nobody
```

I then restarted Samba to implement these settings. With Samba in place I could put the database script into the file share so it was accessible in the virtual machine.

1. service smb restart



I could then move on to installing MySQL on the server.

1. apt-get install mysql-server

I then ran the command to secure MySQL to address some security issues with the default installation.

1. mysql_secure_installation

Through the options I was presented with when this command was run I removed the test databases installed with MySQL. I also disabled anonymous users and changed the settings so that user passwords, including the root, had to be evaluated at a minimum of medium strength. I could then login and begin interacting with the SQL client.

1. mysql -u root -p

I could then run the MySQL script from the file share.

1. mysql -u username -p ComputingRevisionGame < /home/joe/Database_07.sql

This created an instance of the database, working just as it did on MySQL workbench. The difference being that I would be able to connect to this version from anywhere on the school network.

```
Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_ComputingRevisionGame |
+-----+
| answerInstance |
| class           |
| exam           |
| item           |
| itemType       |
| question       |
| student        |
| topic          |
+-----+
8 rows in set (0.01 sec)
```

All it then took was to change the IP address field on the MySQL connector prefab in Unity to the address of the server instead of 'localhost'. With that change made any instance of the application running on a computer connected to the school network over Ethernet or Wi-Fi could connect to the database.

Building the application

Now that my application was finished I thought it would be prudent to build the executable file at this stage, before testing. I thought this would be a good idea because it will allow me to test the application in the same way the user will see it and it is relatively easy and simple to re-build an application if any changes are made once the settings are set and everything is working.

The first step to building the application was to adjust the build settings menu.

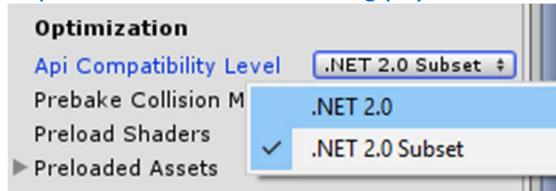
The screenshot shows the Unity Build Settings window. The 'Scenes In Build' list includes: _Scenes/Scene_MainMenu (0), _Scenes/Scene_StudentMode (1), _Scenes/Scene_StaffMode (2), _Scenes/Scene_Inventory (greyed out), _Scenes/Scene_Quiz (greyed out), _Scenes/Scene_Results, and _Scenes/Scene_Questions. The 'Platform' dropdown is set to 'PC, Mac & Linux Standalone'. The 'Target Platform' is 'Windows' and 'Architecture' is 'x86'. The 'Development Build' checkbox is checked. Callouts explain that all scenes were ticked except the greyed-out ones, and that the development build setting was activated for console access.

When I first attempted the build I encountered an error.

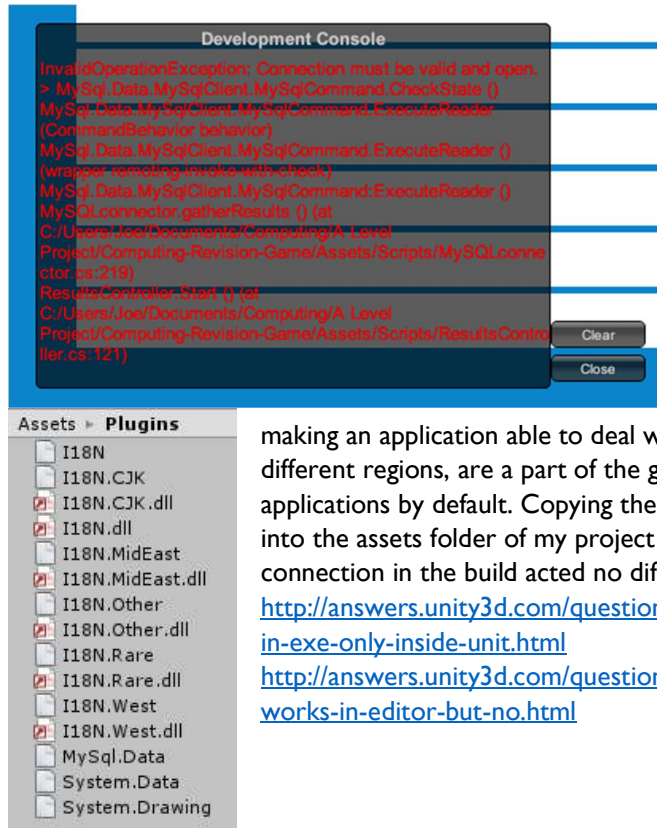
```
ArgumentException: The Assembly System.Configuration.Install is referenced by MySql.Data ('Assets/Plugins/MySql.Data.dll').  
UnityEditor.AssemblyHelper.AddReferencedAssembliesRecurse (System.String assemblyPath, System.Collections.Generic.List`1  
Error building Player: Extracting referenced dlls failed.
```

Reading the error text informed me that there was some problem with the plugin DLLs I had imported to communicate with the database. Through investigating the Unity documentation and forum I discovered that the issue was occurring because of the 'API Compatibility Level'. From what I could ascertain this setting controls the external libraries included in the build. By changing the setting to just '.NET 2.0' the build would include all the associated features of .NET and therefore the build program would recognise the MySQL DLLs.

- <http://answers.unity3d.com/questions/180736/api-compatibility-level.html>
- <http://answers.unity3d.com/questions/65003/error-building-player-extracting-referenced-dlls-f.html>



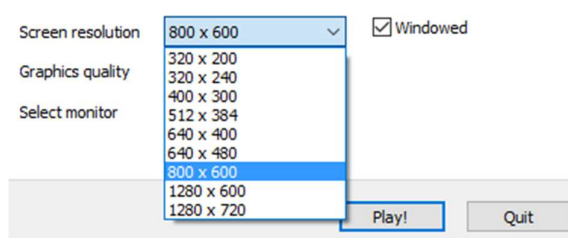
With this change in place the build program completed and the executable file was successfully generated.



However, when I ran the application became apparent through the errors in the development console that the connection with the database wasn't being established.

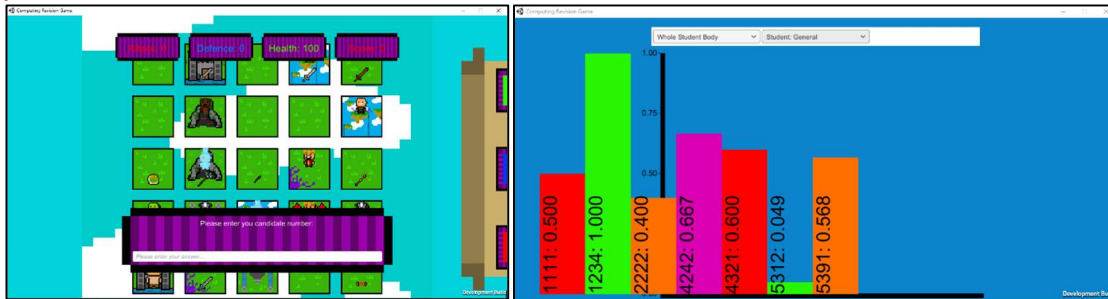
Again, I found the solution to this through the Unity forum and documentation. I discovered that the problem had arisen because I was missing some internationalisation or 'I18N' libraries. These libraries, used for

making an application able to deal with the protocols and standards of different regions, are a part of the game engine but not included in built applications by default. Copying these libraries from the Unity Program files into the assets folder of my project fixed the problem; the database connection in the build acted no different from in the game engine:
<http://answers.unity3d.com/questions/230243/mysql-commands-not-working-in-exe-only-inside-unit.html>
<http://answers.unity3d.com/questions/42955/codepage-1252-not-supported-works-in-editor-but-no.html>

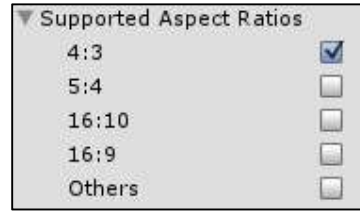
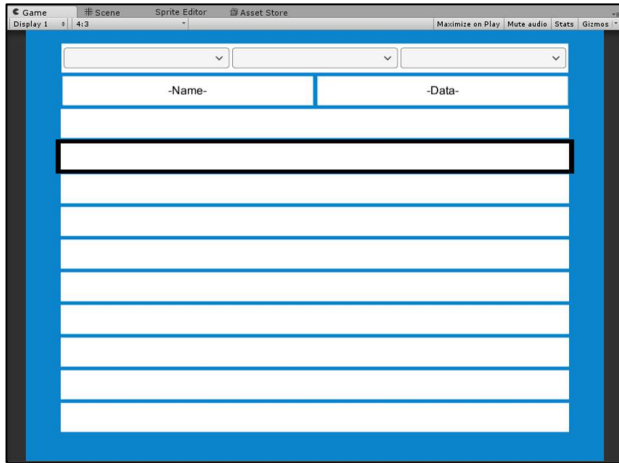


The next problem with the build was an issue with the resolution of the application. When the executable is run a menu appeared giving the user a selection of resolution options, most of which didn't match up with the way I had arranged objects in the application. This made it possible for the user to see some objects

meant to be out of scene and, more importantly, made some of the game objects appear out of place.



The first step in fixing this issue was determining which aspect would best fit my application. Testing the various options available I found that my application was quite close to a 4:3 ratio. I locked the scene view of the game engine to this ratio and set it as the only option for build resolutions in the build settings.

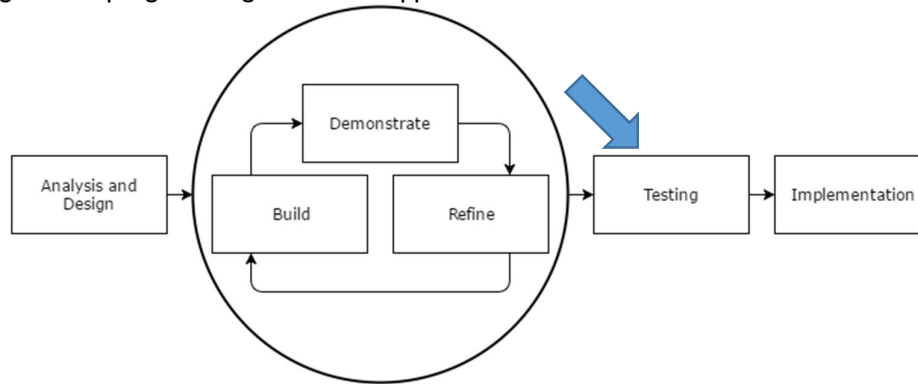


The only changes that needed to be made in the majority of scenes was a slight adjustment to the dimensions of some of the game objects with sprites. I also had to make some slight adjustments to the values used to position bars on the bar scene to make them appear correctly.

With these change in place the application would finally build correctly and I had a working version of the application to use for testing.

Robustness and Function

With the project now complete it is important that it undergoes thorough testing to ensure it works properly and meets the success criteria. As part of the process of development I had been testing new features as they were implemented and errors that caused the code to fail compilation were dealt with immediately. However, to thoroughly test my application I must also conduct 'Alpha Testing' to ensure robustness and 'Beta Testing' to gather the opinions of people without a working knowledge of the programming behind the application.

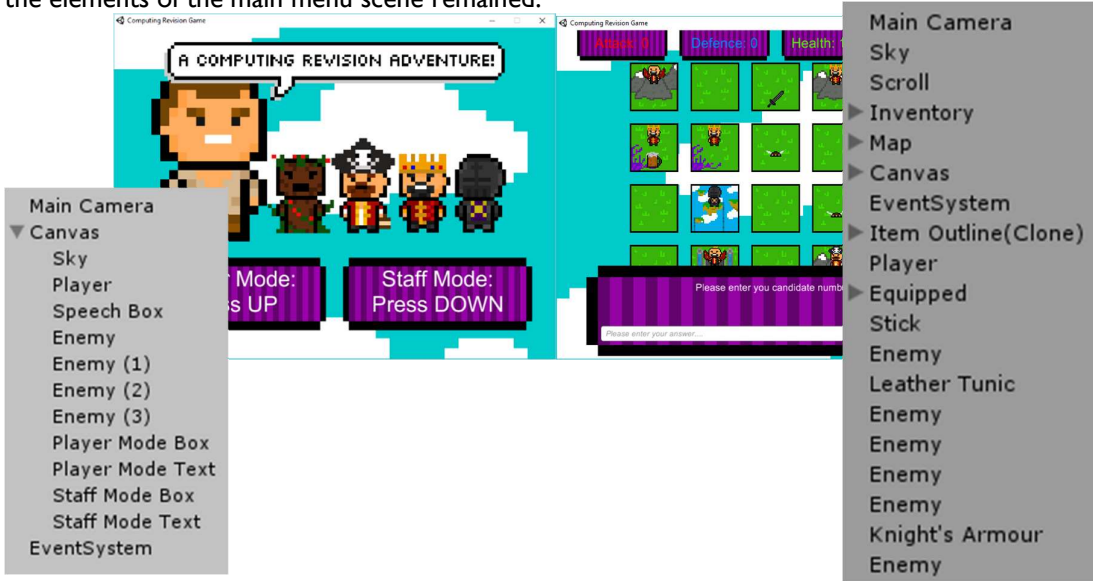



I am currently in the 'Testing' stage of development, when this stage is complete the application will be ready for implementation.

Alpha Testing

As previously explained, at this stage of testing I need to deal with the application as a finished whole. I will test every element of the project and also test that the different components don't interfere with each other. To adequately ensure the project is ready for user I must test the robustness of the application by using a mix of valid, erroneous and boundary data. I decided to frame testing around the success criteria to illustrate how the application meets the brief. Because of the in-development testing I conducted I know that the basic function of my application works soundly. This phase of testing is to ensure that the application will work all of the time, accurately handling every possible input scenario regardless of whether it 'should' occur if the user was using the application as intended.

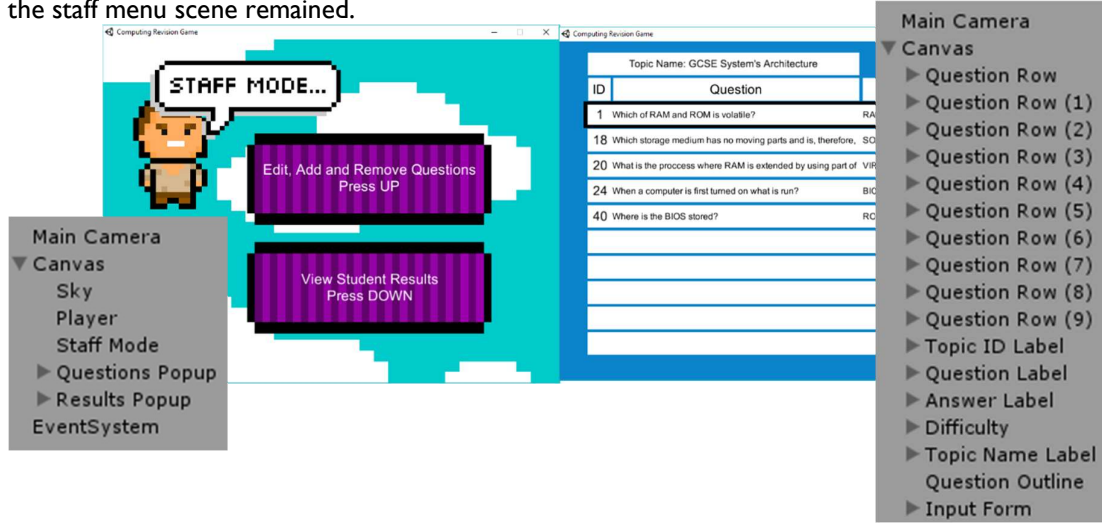
When debugging or otherwise examining the state of the code I will conduct testing in the Unity editor. Otherwise, I will use a built executable version of the application as this is what the user will interact with. After my earlier work to rectify the problems with the build version, there is no reason why the function of these two versions should differ.

General Function		
0.1: Moving from main scene to game scene		
Reason for Test	Input Data	Expected Result
Valid input test	Up key	Game scene loaded, no main menu game objects remain
<p>The text on the main scene informs the user they need to press the Up key to go to the player portion of the application. Pressing the Up key did result in the game scene being loaded, none of the elements of the main menu scene remained.</p> 		
Test Passed		
0.2: Moving from main scene to staff menu scene		
Reason for Test	Input Data	Expected Result
Valid input test	Down Key	Staff scene loaded, no main menu game objects remain
<p>The text on the main scene informs the user they need to press the Down key to go to the staff portion of the application. Pressing the Down key did result in the staff menu scene being loaded, none of the elements of the main menu scene remained.</p> 		
Test Passed		
0.3: Invalid key press on main scene		
Reason for Test	Input Data	Expected Result
Invalid input test	Left, Right, 'a', 'l', Shift, Escape	No result
<p>No other keys apart from Up and Down should elicit a response from the Main scene. None of the listed input data resulted in any change.</p>		
Test Passed		

0.4: Moving from staff menu scene to questions scene

Reason for Test	Input Data	Expected Result
Valid input test	Up Key	Questions scene loaded, no staff menu game objects persist

The text on the staff scene informs the user they need to press the Up key to go to the questions scene. Pressing the Up key did result in the questions scene being loaded, none of the elements of the staff menu scene remained.

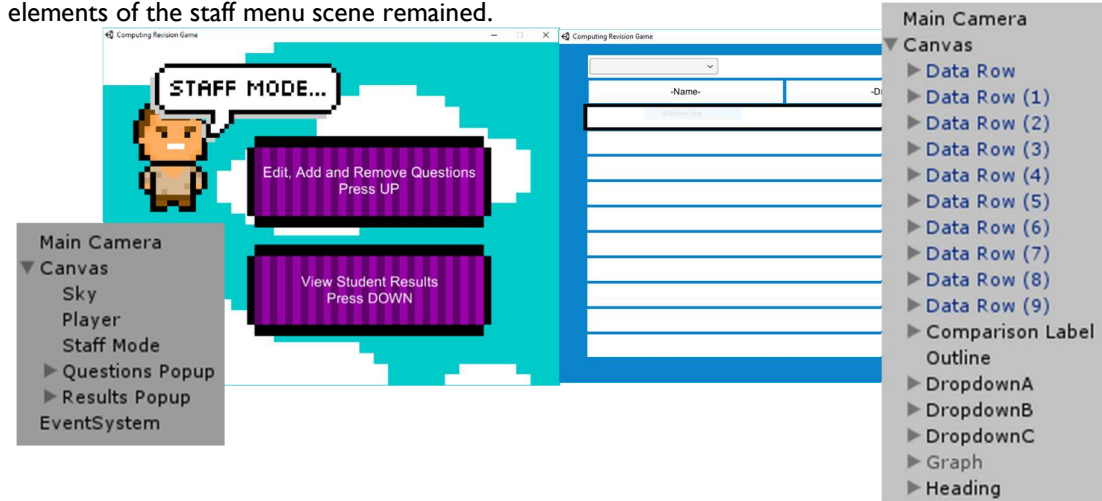


Test Passed

0.5: Moving from staff menu scene to results scene

Reason for Test	Input Data	Expected Result
Valid Input Test	Down Key	Results scene loaded, no staff menu game objects persist

The text on the staff scene informs the user they need to press the Down key to go to the results scene. Pressing the Down key did result in the results scene being loaded, none of the elements of the staff menu scene remained.



Test Passed

0.6: Invalid key press on staff scene

Reason for Test	Input Data	Expected Result
Invalid Input Test	Left, Right, 'a', 'l', Shift, Escape	No Result

No other keys apart from Up and Down should elicit a response from the staff menu scene. None of the listed input data resulted in any change.

Test Passed

Criteria 2: They Student facing part of my application must take the form of a game										
2.1: Valid Candidate Login, New Student										
Reason for Test	Input Data	Expected Result								
Valid Input Test	'5678','13A','John','Doe'	New student added to database, game begins								
<p>At the start of the game the student is asked to login. The user is asked to enter their candidate number to check if they already exist in the database. If they are a new student they should also be asked for their class and name and the details of this student should be entered as a new record to the student table of the database.</p> <p>To test this I intentionally entered a candidate number I knew didn't match with an existing record. The code correctly asked me for the other details and, by looking at the contents of the database I saw the new record had been entered.</p> <pre>1 • SELECT * FROM student;</pre> <table border="1"> <thead> <tr> <th>studentID</th> <th>firstName</th> <th>lastName</th> <th>_classID</th> </tr> </thead> <tbody> <tr> <td>5678</td> <td>Joe</td> <td>Rackham</td> <td>13A</td> </tr> </tbody> </table>			studentID	firstName	lastName	_classID	5678	Joe	Rackham	13A
studentID	firstName	lastName	_classID							
5678	Joe	Rackham	13A							
Test Passed										
2.2: Valid Candidate Login, Existing Student										
Reason for Test	Input Data	Expected Result								
Valid Input Test	'5678'	Secondary questions not asked, game begins								
<p>At the start of the game the student is asked to login. The user is asked to enter their candidate number to check if they already exist in the database. If they are an existing student the user shouldn't be asked any other questions and the game should begin.</p> <p>To test this I intentionally entered a candidate number I knew matched with an existing record. The code correctly didn't ask any further questions and the game loop began.</p>										
Test Passed										
2.3: Candidate Login, Invalid Candidate number input										
Reason for Test	Input Data	Expected Result								
Invalid Input Test	'#a!bc£4\$I'	Input is rejected								
<p>The candidate numbers used for the primary keys are four character long numerical codes. The application should reject input that doesn't take this form. The input I used was both made up of invalid characters and too long. However, this was accepted as an input and resulted in an error when the application attempted to add this as part of a record.</p> <pre>! MySQLException: You have an error in your SQL syntax; MySQL.Data.MySqlClient.MySqlStream.ReadPacket ()</pre>										
Test Failed										

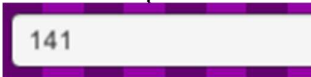
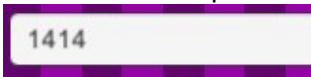
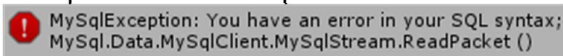
To fix this in needed to implement some validation measures into the login subroutine.

```
public IEnumerator login () {
    questionText.gameObject.SetActive (true);
    inputField.gameObject.SetActive (true);
    quizBackground.gameObject.SetActive (true);
    questionText.text = "Please enter your candidate number: ";
    inputField.characterLimit = 4;
    inputField.characterValidation = InputField.CharacterValidation.Integer;
    while (inputField.text.Length != 4) {
        enter = false;
        while (enter == false) {
            yield return null;
        }
    }
    playerID = inputField.text;
}
```

This line imposes a character limit to prevent users entering overlong candidate numbers.

This line imposes the 'Integer' validation type on input to the field to prevent any character that aren't numbers.

The while loop is implemented so the input is only accepted when the user presses enter and the input is exactly 4 characters long.

2.3a: Candidate Login, Invalid Candidate number input REPEAT		
Reason for Test	Input Data	Expected Result
Invalid Input Test	'#a bc£\$!' / '14144'	Input is rejected
Repeating the previous test with the changes implemented the input was treated as expected. The non-numerical characters never appeared and when I pressed enter the remaining characters weren't accepted as a valid input because there were only 3.		
		
Then when I typed in two more numerical characters only one appeared to maintain a length under four. This input was correctly accepted when I pressed enter.		
		
Test Passed		
2.4: Candidate Login, Invalid Class input		
Reason for Test	Input Data	Expected
Invalid Input Test	'#!13@za'	Input is rejected
The classes all take the form of three digit alpha numeric codes. The application should reject any input that doesn't take this form. The input I used was both made up of invalid characters and too long. However, this was accepted as an input and resulted in an error when the application attempted to add this as part of a record.		
		
Test Failed		

To fix this issue I needed to implement some changes to the login subroutine, but first I altered the 'existCheck' subroutine in the MySQL script to make them work with different tables.

```
public bool existCheck (string primaryKey, string tableName) {
    query = "SELECT * FROM " + tableName + ";";
    command = new MySqlCommand (query, connection);
    reader = command.ExecuteReader();
    while (reader.Read ()) {
        if (primaryKey == reader.GetString(0)) {
            reader.Close ();
            return true;
        }
    }
    reader.Close ();
    return false;
}
```

By using 'tableName' as the second parameter I changed the subroutine to be able to check the existence of records in every table.

I then changed the login subroutine to validate the class input.

```

if (databaseScript.existCheck (playerID, "student") == false) {
    questionText.text = "You are a new user, what class are you in?: ";
    inputField.characterLimit = 3;
    inputField.characterValidation = InputField.CharacterValidation.Alphanumeric;
    string playerClass = "";
    while (true) {
        while (inputField.text.Length != 3) {
            enter = false;
            while (enter == false) {
                yield return null;
            }
            playerClass = inputField.text.ToUpper();
            Debug.Log (playerClass);
        }
        if (databaseScript.existCheck(playerClass, "class")) {
            break;
        }
        yield return null;
    }
    questionText.text = "What is your first name?: ";
    inputField.characterLimit = 0;
}

```

This line sets the validation to Alphanumeric which only allows numbers and letters. These are the only characters valid for the class name

This line capitalises all letters in a string. This will mean the case the user uses won't affect if their input is valid.

This if statement means that the program will only move on and accept input if the class entered actually exists.

I then needed to add some lines to the end out the subroutine to deal with the fact I had added validation rules. Adding restrictions on input data for the main game would only stifle the kinds of questions teachers could ask

```

playerIsAlevel = databaseScript.isAlevel(playerID);
questionText.gameObject.SetActive (false);
inputField.gameObject.SetActive (false);
quizBackground.gameObject.SetActive (false);
inputField.characterValidation = InputField.CharacterValidation.None;
inputField.characterLimit = 0;

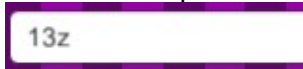
```

I removed the validation rule applied to the input field and set the character limit to 0, which actually means there is no limit.

2.4a: Candidate Login, Invalid Class input **REPEAT**

Reason for Test	Input Data	Expected
Invalid Input Test	'#!13@za' / '13a'	Input is rejected

Repeating the previous test with the changes implemented the input was treated as expected. The non-alphanumeric characters never appeared in the input field and the final character wasn't accepted as it made the length be over 3. Because the resulting string '13z' wasn't an existing class this wasn't accepted when I pressed enter.



Deleting the 'z' and entering a lowercase 'a' resulted in the input being, accurately, accepted and no error occurred when this was added as a new student.



Test Passed

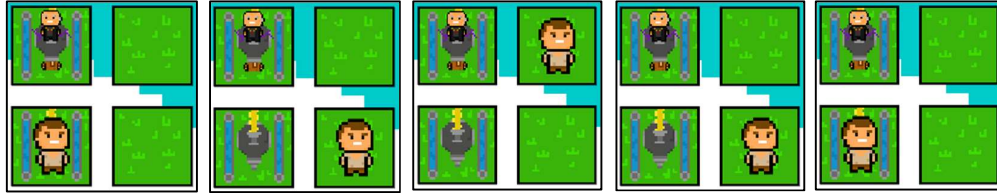
2.5: Movement on the game map in game scene

Reasons For Test	Input Data	Expected Result
Valid Input Test	Up, Down, Left, Right movement within bounds	Movement in executed gridPosition updated

The map consists of a 5x5 grid, movement to any position on this grid should be possible. If there is a valid grid position adjacent to the current position inputting the relevant directional button should cause movement to that position. The gridPosition array in the code should also be updated to reflect the change. A movement up should cause an increase in the y coordinate etc.

To test this I added a Debug statement to the student script to output the grid position to the console. Using this and the visual changes in the game engine I successfully moved to every square on the grid. The screenshots below show an extract from this process.

```
while (playerHealth > 0) {
    Debug.Log ("Grid Position X:" + gridPosition[0] + " Y:" + gridPosition[1]);
```



- ! Grid Position X:0 Y:0
UnityEngine.Debug:Log(Object)
- ! Grid Position X:1 Y:0
UnityEngine.Debug:Log(Object)
- ! Grid Position X:1 Y:1
UnityEngine.Debug:Log(Object)
- ! Grid Position X:1 Y:0
UnityEngine.Debug:Log(Object)
- ! Grid Position X:0 Y:0
UnityEngine.Debug:Log(Object)

Test Passed

2.6: Movement outside the game map on the game scene

Reasons For Test	Input Data	Expected Result
Invalid input test	Up, Down, Left, Right outside bounds	No movement executed gridPosition unchanged

Movement outside of the grid should be impossible, pressing any of the input keys to try and perform an invalid movement should therefore illicit no result. I added Debug statements to output when the directional keys were pressed and inputted invalid movements. Whenever movements weren't supposed to be executed, they weren't and the grid position didn't change until I pressed a valid key.

```
if (Input.GetKeyDown (KeyCode.UpArrow)) {
    directions [0] = true;
    Debug.Log ("Up");
}
if (Input.GetKeyDown (KeyCode.DownArrow)) {
    directions[1] = true;
    Debug.Log ("Down");
}
if (Input.GetKeyDown (KeyCode.LeftArrow)) {
    directions[2] = true;
    Debug.Log ("Left");
}
if (Input.GetKeyDown (KeyCode.RightArrow)) {
    directions[3] = true;
    Debug.Log ("Right");
}
```

- ! Grid Position X:0 Y:0
UnityEngine.Debug:Log(Object)
- ! Left
UnityEngine.Debug:Log(Object)
- ! Down
UnityEngine.Debug:Log(Object)
- ! Up
UnityEngine.Debug:Log(Object)
- ! Grid Position X:0 Y:1
UnityEngine.Debug:Log(Object)
- ! Grid Position X:4 Y:4
UnityEngine.Debug:Log(Object)
- ! Up
UnityEngine.Debug:Log(Object)
- ! Right
UnityEngine.Debug:Log(Object)
- ! Down
UnityEngine.Debug:Log(Object)
- ! Grid Position X:4 Y:3
UnityEngine.Debug:Log(Object)

Test Passed

2.7: Transition between game and inventory

Reasons For Test	Input Data	Expected Result
Valid	Escape	Camera moves to inventory, Inventory loop runs Camera moves to game, Game loop runs

At any point when the user isn't in a quiz or already in the inventory they should be able to, using the Escape key, to 'open' the inventory. This means the camera position shifts to show the inventory screen and the 'runInventory' coroutine begins in the code. If the user is in the inventory they should, also using Escape, be able to 'close' the inventory. This is represented by the camera shifting back to the game and the game loop continuing.

```
public IEnumerator runInventory () {
    Debug.Log ("Inventory Opened");
}
```

To test this I added a debug statement to output whenever the 'runInventory' coroutine was run and moved to various different grid locations attempting to validly open the inventory and then return to the game. At every one of these occasions the camera moved correctly and the debug console showed the change had been recognised in code.

```
! Grid Position X:0 Y:0
UnityEngine.Debug:Log(Object)
! Inventory Opened
UnityEngine.Debug:Log(Object)
! Grid Position X:0 Y:1
UnityEngine.Debug:Log(Object)
! Grid Position X:0 Y:2
UnityEngine.Debug:Log(Object)
! Inventory Opened
UnityEngine.Debug:Log(Object)
```

Test Passed

2.8: Attempt at transition between quiz and inventory

Reasons For Test	Input Data	Expected Result
Invalid input test	Escape	Nothing

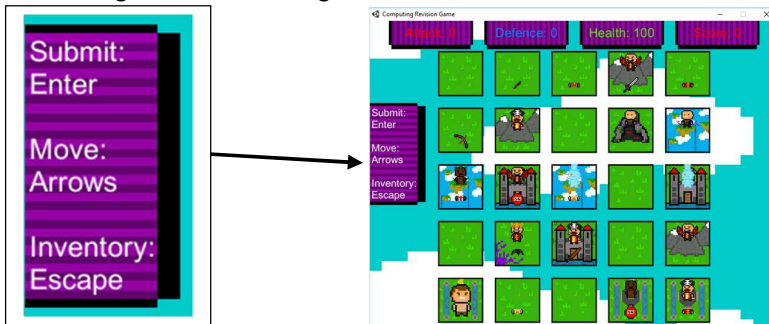
In the game, when on the game map, the Escape key is used to transition to the inventory. However, when the user is in a quiz attempts the Escape key shouldn't have any result. The inventory shouldn't open during the quiz or immediately after as a result of a press during the quiz.

```
public IEnumerator runInventory () {
    Debug.Log ("Inventory Opened");
}
```

To test this I added a debug statement to output whenever the 'runInventory' coroutine was run and pressed Escape during all the stages of the quizzes making sure to press both an even and odd number of times. Both the visual aspects of the game and the developer console indicated that the inventory hadn't been opened at any point.

Test Passed

During this test I realised that, to a potential user, the keys used to control the game may not be immediately apparent. I added a small controls box into some empty space in the game screen to make things a bit more straightforward.



2.9: Item collected, space in Inventory

Reason for Test	Input Data	Expected Result
Valid input test	Collect Kingsguard Helm Collect Tankard of Mead Collect Dragon Sword Drop Kingsguard Helm Collect Formal Doublet	Kingsguard Helm at inventory[0] Tankard of Mead at inventory[1] Dragon Sword at inventory[2] inventory[0] = null Formal Doublet at inventory[0]

If the user moves onto a grid position with an item and there is space in the inventory the item should be added into the first available space in the inventory. To test this I ran the game and,

whilst the inventory had space, collected several items. I then deleted one of these items and collected another. I added a debug statement to the 'gameLoop' coroutine at the point where new items are added to inventory to show that the right item was added at the correct position.

```
for (int i=0; i<inventory.Length; i++) {
    if (inventory[i] == null) {
        inventory [i] = itemArray [x, y];
        Debug.Log ("Index Position: " + i + " Item: " + inventory [i].getName ());
        break;
    }
}
```



! Index Position: 0 Item: Kingsguard Helm
 UnityEngine.Debug:Log(Object)
 ! Index Position: 1 Item: Tankard of Mead
 UnityEngine.Debug:Log(Object)
 ! Index Position: 2 Item: Dragon Sword
 UnityEngine.Debug:Log(Object)
 ! Index Position: 0 Item: Formal Doublet
 UnityEngine.Debug:Log(Object)

Test Passed

2.10: Item collected, no space in Inventory

Reason for Test	Input Data	Expected Result
Invalid Input Test	With Full inventory: Health Potion Collected	'Inventory Full' error popup displayed

The player inventory is limited to 16 spaces. When the inventory is full any attempts to collect another item should result in the 'Inventory full' error popup being displayed. To test this I changed a line in the 'populate' subroutine to prevent any quizzes being populated onto the game map and ran the game, collecting items until the inventory was full. I then attempted to collect another item.

```
randomNumber = /*/Random.Range (1,13);/*/ 1;|
if (randomNumber > 6) {
    int topic = randomNumber - 6;
```



As intended, the item wasn't added to the inventory, but instead the 'Inventory full' error message was displayed, informing the user of the situation.

Test Passed

2.11: Inventory Usable Item Use

Reason for Test	Input Data	Expected Result
Valid Input Test	Health: 93, (Tankard of Mead,2,4) Health: 90, (Health Potion,8,4), (Health Potion, 8,4), (Tankard of Mead,2,4) Health: 70, (Health Potion,8,4), (Chicken Leg, 3,4)	Health = 95 Health = 100 Health = 81

When the user presses enter on a selected item in the inventory the application should react differently depending on its type. Usable Health items should be destroyed and the value of their strength should be added to the player health up to and including a health value of 100. To test this functionality I run through the game several times so a variety of different items and scenario would occur. In every situation the items were dealt with correctly; the health value of the items was added to the player health until the player health exceeded 100 and the items were removed from the inventory



itemID	itemName	effect	_itemTypeID
9	Tankard of Mead	2	4
10	Chicken Leg	3	4
11	Health Potion	8	4

Test Passed

2.12: Inventory Equipable Item Equip Test

Reason for Test	Input Data	Expected Result
Valid Input Test	(Iron Helmet,3,1) (Dragon Sword,10,3) (Leather Tunic,3,2) (Mage's Staff,5,3)	Iron Helmet Equipped as Hat Dragon Sword Equipped as Weapon, Attack = 10 Leather Tunic Equipped as Top, Defence = 3 Mage's Staff Equipped as Weapon, Attack = 5, Dragon Sword Returned to Inventory

When the user presses enter on a selected item in the inventory the application should react differently depending on its type. If the item is equipable it should be made the equipped item of that type. This means it should be removed from the main inventory and placed in the equipped items box relevant to its type. A sprite of the item should be added to the player game object. If the item is a weapon the Attack value should be updated to its strength and if the item is a top the Defence value should be updated to its strength.

I tested equipping several items. In all cases the items were equipped to the correct body part and in the case of weapons and tops the player variables were adjusted. When item were already equipped in a slot they were correctly returned to the inventory.

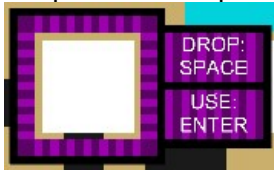
6	Knights Helm	5	1
12	Kingsguard Helm	8	1
13	Viking Helmet	2	1
14	Iron Helmet	3	1
7	Knights Armour	5	2
8	Formal Doublet	3	2
15	Kingsguard Arm...	8	2
16	Leather Tunic	3	2
16	Leather Tunic	3	2
1	Stick	1	3
2	Wooden Sword	2	3
3	Iron Sword	3	3
4	Steel Sword	4	3
5	Excalibur	8	3
17	Dragon Sword	10	3
18	Mage's Staff	5	3
19	Fire Staff	7	3
20	Wooden Bow	4	3
21	Wooden Crossbow	4	3

Test Passed

2.13: Attempting to Equip / Use with an Empty Inventory

Reason for Test	Input Data	Expected Result
Invalid input test	Enter, on Empty Inventory	Nothing

It is possible to open the inventory even if it is empty. However, pressing Enter shouldn't be accepted as valid input if there is no item to select.



To test this I repeatedly pressed Enter when the inventory was empty, in cases where it was empty because the game had only just began and when all items had been 'dropped'. In all these cases the application correctly did nothing.

Test Passed

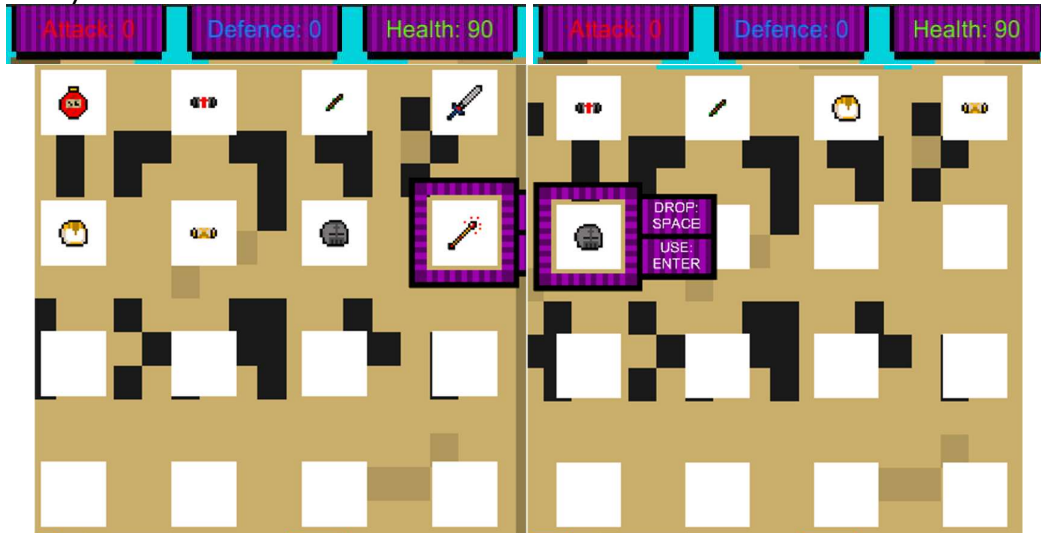
2.14: Inventory Item Drop Test

Reason for Test
Valid Input Test

Input Data	Expected Result
Inventory = (Health Potion,8,4),(Knight's Armour,5,2),(Stick,1,3),(Dragon Sword,10,3),(Kingsguard Helm,8,1),(Kingsguard Armour,8,2),(Iron Helmet,3,1),(Fire Staff,7,3),(8 Empty Slots)	Inventory = (Knight's Armour,5,2),(Stick,1,3),(Kingsguard Helm,8,1),(Kingsguard Armour,8,2),(Iron Helmet,3,1),(11 Empty Slots)
Space on: (Health Potion,8,4),),(Dragon Sword,10,3),(Fire Staff,7,3)	No result on Health, Attack or Defence.

The instructions attached to the selection outline in the inventory indicate to the user that pressing Space will drop the selected item. Therefore, pressing Space should, on all occasions where an item is selected result in that item being removed from the inventory with no result on the health, defence and attack player values.

To test this I ran the game and collected several items, one of each type and of varying strengths. I dropped some of these items and the application responded correctly. The items were removed with no impact on the Player variables and the remaining items 'moved up', filling the first available inventory slots.

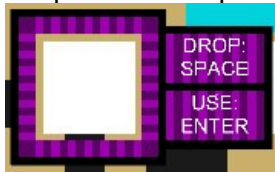


Test Passed

2.15: Attempting to Drop Item with Empty Inventory

Reason for Test	Input Data	Expected Result
Invalid input test	Space, on Empty Inventory	Nothing

It is possible to open the inventory even if it is empty. However, pressing Space shouldn't be accepted as valid input if there is no selected item.



To test this I repeatedly pressed Space when the inventory was empty, in cases where it was empty because the game had only just began and when all items had been 'dropped' or 'used'. In all these cases the application correctly did nothing.

Test Passed

Criteria 1: My solution must ask students questions about Computer Science		
1.1: Selecting and Distributing Topics		
Reason for Test	Input Data	Expected Result
Valid Input Test	(‘10A Student’,10A) (‘11B Student’,11B) (‘12A Student’,12A) (‘13A Student’,13A)	GCSE Topics GCSE Topics A-Level Topics A-Level Topics
<p>There are six topics that questions can be assigned to, each with an A-Level and GCSE variant. The application must populate the game map with quizzes from different topics. Population is random, however, all topics must have the possibility to appear. Inside quizzes the application must present users from classes in years 10 and 11 with questions from the GCSE version of the topic and students from years 12 and 13 with questions from the A-Level version.</p> <p>To test this I logged into the application as several different students, each from different classes. To test that all the topic were appearing I added a Debug statement to the ‘Populate’ subroutine to output the topic number. Logging into the application several times showed that all the different topics were appearing</p> <pre>randomNumber = Random.Range (1,13); if (randomNumber > 6) { int topic = randomNumber - 6; randomNumber = Random.Range (0, enemySprites.Length); quizArray [i, j] = new Quiz (topic, enemySprites [randomNumber], "Enemy"); Debug.Log ("Topic :" + topic);</pre>  <p>The screenshot shows a series of Unity console logs. Each log entry consists of a small icon (a circle with an exclamation mark) followed by the text 'Topic :X' and 'UnityEngine.Debug:Log(Object)', where X is a number from 1 to 6. The logs are arranged in three columns, demonstrating that all six topics were successfully generated and logged during the testing process.</p> <p>To test that the application was treating users correctly in terms of the questions it asked them I ran quizzes with the different users I had created. I added a Debug statement to the ‘runQuiz’ subroutine to output what topic primary key it was requesting questions from, on every occasion this was correct for the student logged in.</p> <p>- 10A student</p>   <p>The screenshot shows four Unity console log entries for a 10A student: 'TopicNo: 1', 'TopicNo: 3', 'TopicNo: 4', and 'TopicNo: 6', each followed by 'UnityEngine.Debug:Log(Object)'. To the right, two example quiz questions are displayed in a purple box: 'Where is the BIOS stored?' and 'Which type of error in code causes a program to fail to compile?'</p>		

- I1B Student

- ! TopicNo: 6
UnityEngine.Debug:Log(Object)
- ! TopicNo: 1
UnityEngine.Debug:Log(Object)
- ! TopicNo: 3
UnityEngine.Debug:Log(Object)
- ! TopicNo: 4
UnityEngine.Debug:Log(Object)

Which storage medium has no moving parts and is, therefore, less likely to wear?

Which Logic gate takes two inputs and returns TRUE if atleast one of them is TRUE?

- I2A Student

- ! TopicNo: 9
UnityEngine.Debug:Log(Object)
- ! TopicNo: 8
UnityEngine.Debug:Log(Object)
- ! TopicNo: 10
UnityEngine.Debug:Log(Object)
- ! TopicNo: 12
UnityEngine.Debug:Log(Object)

In Lexical Analysis keywords and symbols are replaced with what?

What is the process of analysing large amounts of data to find meaningful patterns?

- I3A Student

- ! TopicNo: 12
UnityEngine.Debug:Log(Object)
- ! TopicNo: 8
UnityEngine.Debug:Log(Object)
- ! TopicNo: 8
UnityEngine.Debug:Log(Object)
- ! TopicNo: 9
UnityEngine.Debug:Log(Object)

Which type of translator translates code line by line?

What is the name for the process of 'hiding' the attributes of a class so they can only be accessed by methods?

Test Passed

I.2: Correct Answer to Question

Reason for Test

Valid Input Test

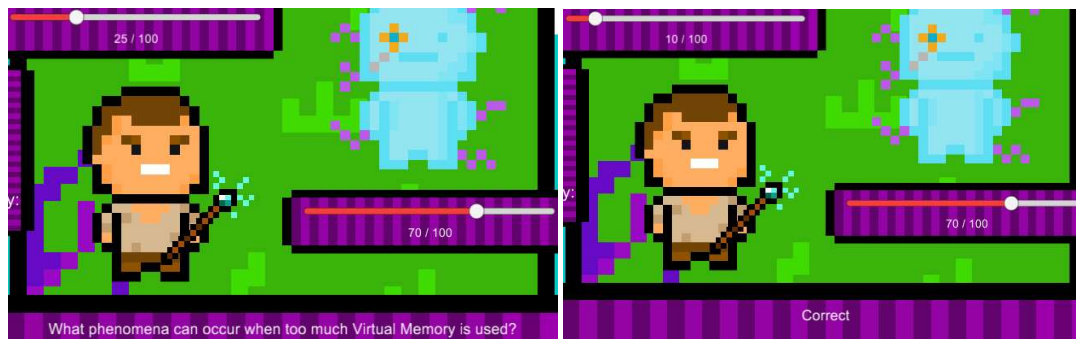
Input Data	Expected Result
playerAttack = 10, enemyHealth = 100 (41,What legislation regulates the way data must be stored?,DPA), 'DPA'	Correct message displayed enemyHeath = 80 Database: (PK,I,DATE,I I I I,41)
playerAttack = 8, enemyHealth = 82 (37,Which Logic gate takes two inputs and returns TRUE only if both of them are TRUE?,AND),'And'	Correct message displayed enemyHeath = 64 Database: (PK,I,DATE,I I I I,37)
playerAttack = 0, enemyHealth = 70 (25,What piece of software allows a device to emulate another, VIRTUAL MACHINE),'vlrTuaL MacHine'	Correct message displayed enemyHeath = 60 Database: (PK,I,DATE,3333,25)
playerAttack = 5, enemyHealth = 25, (21,What phenomena can occur when too much Virtual Memory is used?,DISK THRESHING), 'DiSk THreSHinG'	Correct message displayed enemyHeath = 10 Database: (PK,I,DATE,3333,21)

Fundamental to the success on my application is the games ability to correctly evaluate if an inputted answer is correct or not and deal with the result appropriately. If the answer inputted is correct the application should display a message indicating this and the Enemy Health value should be lowered by 10 and the player's attack value. The application should then add a record to the 'answerInstance' table of the database with the 'Correct' field set to one. The application should capitalise results before comparing them with the answer so the capitalisation of input should affect if it is correct.

To test this I ran through several quizzes answering questions correctly. I employed a variety of capitalisation patterns to test if this had an impact. On every occasion the answer to the question was evaluated to be correct, the message indicating this was displayed and the enemy health value was adjusted correctly. Furthermore, records were added to the database with the correct questionID indication the user had got the question right. The examples below display this happening in some of the test cases applied:



answerInstanceID	correct	date	_studentID	_questionID
45	1	2017-04-22 11:02:08	1111	37



answerInstanceID	correct	date	_studentID	_questionID
86	1	2017-04-22 11:14:47	3333	21

Test Passed

1.3: Incorrect Answer to Question

Reason for Test

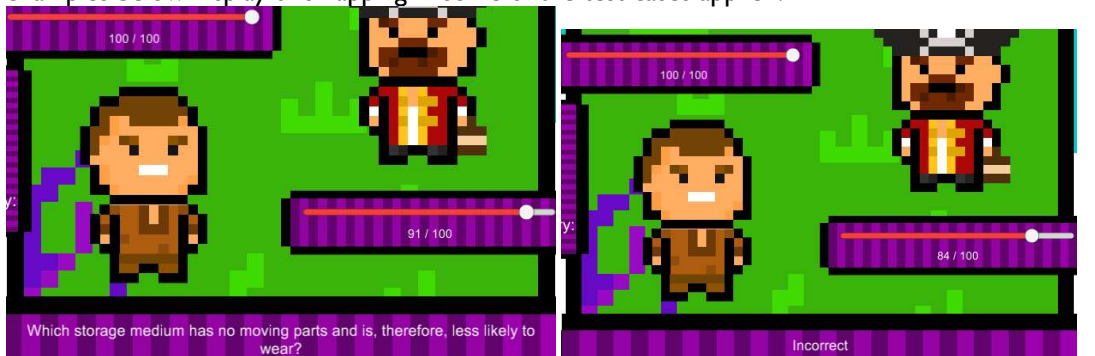
Erroneous Valid Input Test

Input Data	Expected Output
playerDefence = 8, playerHealth = 100 (10,Which type of error in code causes a program to fail to compile?,SYNTAX),'LOGIC'	Incorrect message displayed playerHeath = 98 Database: (PK,0,DATE,1111,10)
playerDefence = 3, playerHealth = 91 (18,Which storage medium has no moving parts and is, therefore, less likely to wear?,SOLID STATE),'SOLID STAND'	Incorrect message displayed playerHeath = 84 Database: (PK,0,DATE,1111,18)
playerDefence = 5, playerHealth = 65 (42, What is the process of analysing large amounts of data to find meaningful patterns?,DATA MINING),'datamining'	Incorrect message displayed playerHeath = 60 Database: (PK,0,DATE,3333,42)
playerDefence = 3, playerHealth = 77	Incorrect message displayed

(8,What type of entity relationship should be avoided?, MANY TO MANY), 'MANY TO MAN'	playerHeath = 70 Database: (PK,0,DATE,3333,8)
--	--

Fundamental to the success on my application is the games ability to correctly evaluate if an inputted answer is correct or not and deal with the result appropriately. If the answer inputted is incorrect the application should display a message indicating this and the Player Health value should be lowered by 10 minus player's defence value. The application should then add a record to the 'answerInstance' table of the database with the 'Correct' field set to zero.

To test this I ran through several quizzes answering questions incorrectly. On every occasion the answer to the question was evaluated to be incorrect, the message indicating this was displayed and the player health value was adjusted correctly. Furthermore, records were added to the database with the correct questionID indication the user had got the question wrong. The examples below display this happening in some of the test cases applied:



SOLID STAND	answerInstanceID	correct	date	_studentID	_questionID
	4	0	2017-04-22 10:27:19	1111	18



MANY TO MAN	answerInstanceID	correct	date	_studentID	_questionID
	28	0	2017-04-22 10:35:03	3333	8

Test Passed

1.4: No Answer to Question Given

Reason for Test	Input Data	Expected Output
Invalid Input Test	''	Nothing Nothing

For both logical reasons and the fact that the 'answer' field in the 'question' table is has a Not Null constraint put on it, there will never be a question to which the answer is blank. Therefore, the application shouldn't accept an answer without any actual Alphanumeric characters in it.

Column Name	Datatype	PK	NN
questionID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
questionText	VARCHAR(200)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
answer	VARCHAR(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
_topicID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
		<input type="checkbox"/>	<input type="checkbox"/>

To test this I ran a quiz and attempted to enter both nothing and an answer made up only of spaces. However, both of these were accepted as legitimate answers and resulted in the input being treated as an incorrect answer.

Incorrect

Test Failed

In an attempt to rectify this issue I enclosed the while loop that waits for user input in 'runQuiz' with another while loop to validate the input's length.

```

askedQuestions [index] = true;
questionText.text = questions [index, 1];
while (inputField.text.Length == 0) {
    enter = false;
    while (enter == false) {
        yield return null;
    }
    yield return null;
}
    
```

This internal loop waits for the user to submit their answer using Enter.

Because of this second loop the first loop will be repeated until the user submits an answer that isn't empty.

I.4a: No Answer to Question Given REPEAT		
Reason for Test	Input Data	Expected Output
Invalid Input Test	“ ”	Nothing Nothing
Repeating the test with the changes implemented only led to a partial success. Because the value in the input field is never cleared in-between login and the first quiz the first question was immediately evaluated as an incorrect answer. On subsequent questions the changes stopped the empty string being accepted but not the string made up of only spaces as these characters were counted as part of the length.		
Test Failed		

The first step in fixing this was to clear the login details from the input field. I did this by adding a statement onto the list of changes to the input field made at the end of the login subroutine that emptied the field.

```

inputField.characterValidation = InputField.CharacterValidation.None;
inputField.characterLimit = 0;
inputField.text = string.Empty;
    
```

The next step was to make the program disregard spaces when evaluating the length.

```

askedQuestions [index] = true;
questionText.text = questions [index, 1];
while (inputField.text.Replace(" ", "").Length == 0) {
    enter = false;
    while (enter == false) {
        yield return null;
    }
    yield return null;
}
    
```

The replace function replaces all instances of a space with an empty string. The result of the whole series of functions is the number of non-space characters.

I.4b: No Answer to Question Given REPEAT		
Reason for Test	Input Data	Expected Output
Invalid Input Test	“ ”	Nothing Nothing

Repeating the test with these further changes implemented resulted in a success. Neither the empty answer nor the answer made up only of spaces were accepted as valid input. Only when actual characters were entered did the quiz progress.

Test Passed

Criteria 4: My solution must allow teachers to edit, add and remove questions

4.1: Topic Tab Navigation in the Questions menu

Reason for Test	Input Data	Expected Output
Valid Input Test	topicNumber = 3, Right topicNumber = 9, Left topicNumber = 2, Left topicNumber = 11, Right	topicNumber = 4 topicNumber = topicNumber = 1 topicNumber = 12

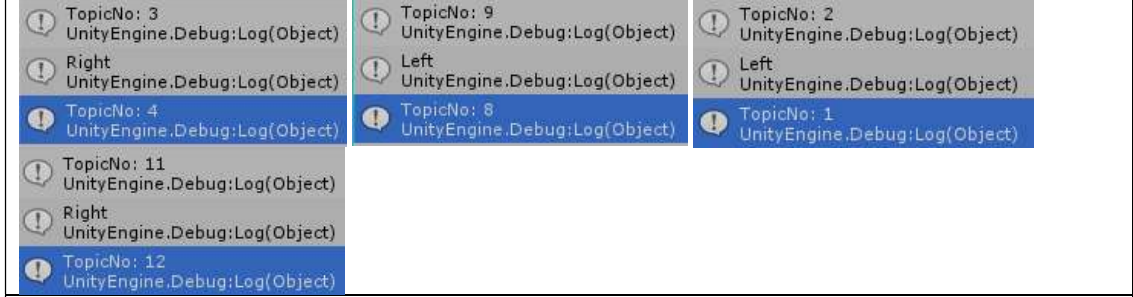
The total bank of questions is distributed between six topics, each with a GCSE and A-Level variant. These 12 categories are each represented by a tab in the questions scene. The Left and Right arrow keys should, if there is an adjacent topic to move to, result in a transition to the tab of the topic with a topic ID one lower or one higher respectively. This means that the topicNumber variable is changed and questions from that topic are displayed.

To test this I added a debug statement to the end of main while loop of the 'RunLoop' subroutine to output the Topic number at every point, and to the 'Update' subroutine to output whenever Up or down was pressed. I made sure that I moved to the first and final topic to make sure that input was still handled correctly at the boundary of the valid region.

```

if (Input.GetKeyDown (KeyCode.LeftArrow)) {
    left = true;
    Debug.Log ("Left");
}
if (Input.GetKeyDown (KeyCode.RightArrow)) {
    right = true;
    Debug.Log ("Right");
}
Debug.Log ("TopicNo: " + topicNumber);
    
```

From the result of these Debug statements and the visual changes on screen I could see the application was responding to user input correctly.



Test Passed

4.2: Invalid Topic Tab Navigation in the Questions menu

Reason for Test	Input Data	Expected Output
Invalid Input Test	topicNumber = 1, Left topicNumber = 12, Right	Nothing Nothing

The total bank of questions is distributed between six topics, each with a GCSE and A-Level variant. These 12 categories are each represented by a tab in the questions scene. The left and right keys are used to navigate between these tabs but movement outside these 12 tabs should be impossible and attempting this action should elicit no result.

Again I used debug statements in the 'RunLoop' and 'Update' subroutines to see how the topicNumber variables changed.

```

if (Input.GetKeyDown (KeyCode.LeftArrow)) {
    left = true;
    Debug.Log ("Left");
}
if (Input.GetKeyDown (KeyCode.RightArrow)) {
    right = true;
    Debug.Log ("Right");
}
Debug.Log ("TopicNo: " + topicNumber);

```

The results attempting the test cases displays how attempting to move to a topic above 12 or below 1 was accurately not accepted as valid input.

Test Passed

4.3: Deleting Questions

Reason for Test

Valid Input Test

Input Data	Expected Output
(1,Which of RAM and ROM is volatile?)	Record 1 Deleted from 'question' All records where _questionID = 1 Deleted from 'answerInstance'
(35,How many bits are in a Megabyte?)	Record 35 Deleted from 'question' All records where _questionID = 35 Deleted from 'answerInstance'
(42,What is the process of analysing large amounts of data to find meaningful patterns?)	Record 42 Deleted from 'question' All records where _questionID = 42 Deleted from 'answerInstance'

As part of my brief, Staff users need to be able to edit the questions list and this encompasses the ability to delete questions. By pressing Escape with any question selected, the user should cause the 'Are You Sure' Popup to appear. Pressing Enter with this popup displayed should then result in the application sending a query to the database deleting that question. Any answer to this question stored in the database should also be deleted to maintain the referential integrity of the database.

To test this I opened the Questions scene and attempted to delete several of the questions. As the screenshots from the database show, on every occasion this resulted in the questions being removed from the database along with any answers to that question.

questionID	questionText	answer	_topicID
1	Which of RAM and ROM is volatile?	RAM	1
2	What acts as a unique identifier in a Database?	PRIMARY KEY	2

answerInstanceID	correct	date	_studentID	_questionID
5	0	2017-04-22 10:27:20	1111	1
14	1	2017-04-22 10:28:30	1111	1
65	0	2017-04-22 11:08:51	4444	5

questionID	questionText	answer	_topicID
2	What acts as a unique identifier in a Database?	PRIMARY KEY	2

answerInstanceID	correct	date	_studentID	_questionID
65	0	2017-04-22 11:08:51	4444	5

34	Which type of processor is designed so the same instr...	ARRAY	7	
35	How many bits are in a Megabyte?	8388608	5	
36	Convert the hexadecimal number 3E into decimal	62	5	
52	1	2017-04-22 11:07:21	3333	34
130	1	2017-04-22 15:51:13	1111	35
108	0	2017-04-22 13:38:19	1111	36
129	0	2017-04-22 15:50:55	1111	36
34	Which type of processor is designed so the same instr...	ARRAY	7	
36	Convert the hexadecimal number 3E into decimal	62	5	
52	1	2017-04-22 11:07:21	3333	34
108	0	2017-04-22 13:38:19	1111	36
41	What legislation regulates the way data must be stored?	DPA	4	
42	What is the process of analysing large amounts of dat...	DATA MINING	10	
43	What extra component might a Gamring PC have that ...	GPU	7	
66	1	2017-04-22 11:09:06	4444	42
90	0	2017-04-22 11:41:26	3333	42
104	0	2017-04-22 12:50:26	3333	42
105	0	2017-04-22 12:53:01	3333	42
51	1	2017-04-22 11:07:18	3333	43
41	What legislation regulates the way data must be stored?	DPA	4	
43	What extra component might a Gamring PC have that ...	GPU	7	
128	1	2017-04-22 15:50:23	1111	41
51	1	2017-04-22 11:07:18	3333	43

Test Passed

In the process of conducting this test I realised that the controls to delete, add and edit questions wasn't necessarily intuitive and decided to add a controls information box to the scene.

33 What register holds data that has been written to or read from MDR 1

Change Topic:Left/Right Change Question:Up/Down Edit:Enter Add:a Delete:Escape

There was no reason why this text couldn't appear all the time so I didn't need to adjust any code to include it.

4.4: Attempting to Delete with no Question Selected		
Reason For Test	Input Data	Expected Result
Invalid Input Test	Escape, on Empty Questions List	Nothing
It is entirely possible to be on the tab for a specific topic even when the topic has no questions. In this scenario the questions outline, although immovable, still appears on the screen; it is entirely possible that a user could press Escape even though there is no question to delete. In this scenario the application shouldn't accept this as a valid input and should do nothing.		

Topic Name: GCSE Ethical, Cultural and Legal Concerns			
ID	Question	Answer	Diff
To test this I deleted all the questions from a particular topic and then pressed Escape again. The application correctly did nothing with this input.			
Test Passed			
4.5: Editing a question			
Reason For Test			
Valid Input Test			
Input Data		Expected Result	
(8,What type of entity relationship should be avoided?,MANY TO MANY)		(8,What type of entity relationship should be avoided?,MANY TO MANY)	
(1, Which storage medium has no moving parts and is, therefore, less likely to wear?,SOLID STATE) Topic = 4		(4, Which storage medium has no moving parts and is, therefore, less likely to wear?,SOLID STATE)	
(9, Which type of translator translates code line by line?,INTERPRETER) Question = 'Question'		(9, Question,INTERPRETER)	
(6, Which type of error in code causes a program to fail to compile?,SYNTAX) Answer = 'Answer'		(6, Which type of error in code causes a program to fail to compile?,ANSWER)	
(5, Convert the hexadecimal number 3E into decimal,62) Topic = 10, Question = 'What is 2+2?', Answer = '4'		(10, What is 2+2?,4)	
As part of my brief, Staff users need to be able to edit the questions list and this encompasses the ability to edit questions without having to delete and re-add them with different details. By pressing Enter with any question selected, the user should cause the 'Question Details' Popup to appear populated with the details of the question. Pressing Enter should result in the record for that question being updated with the value of the fields.			
To test this I opened the Questions scene and attempted to edit several of the questions. I made sure to test changing all the different fields of the question as well as all the fields and no fields. The first three and last test case worked as expected. The relevant fields were updated and this change was evident in both the application and the database.			
Topic Name: A-Level Exchanging Data			
ID	Question	Answer	Diff
8	What type of entity relationship should be avoided?	MANY TO MANY	0.714
8	What type of entity relationship should be avoided?	MANY TO MANY	8
Topic Name: A-Level Exchanging Data			
ID	Question	Answer	Diff
8	What type of entity relationship should be avoided?	MANY TO MANY	0.714
8	What type of entity relationship should be avoided?	MANY TO MANY	8

Topic Name: GCSE System's Architecture			
ID	Question	Answer	Diff
18	Which storage medium has no moving parts and is, therefore,	SOLID STATE	0
18	Which storage medium has no moving parts and is, the...	SOLID STATE	1
Topic Name: GCSE Ethical, Cultural and Legal Concerns			
ID	Question	Answer	Diff
18	Which storage medium has no moving parts and is, therefore,	SOLID STATE	0
18	Which storage medium has no moving parts and is, the...	SOLID STATE	4
29	Which type of translator translates code line by line?	INTERPRETER	0.571
29	Which type of translator translates code line by line?	INTERPRETER	9
29	Question	INTERPRETER	0.571
29	Question	INTERPRETER	9
Topic Name: GCSE Data Representation			
ID	Question	Answer	Diff
36	Convert the hexadecimal number 3E into decimal	62	0
36	Convert the hexadecimal number 3E into decimal	62	5
Topic Name: A-Level Ethical, Cultural and Legal Concerns			
ID	Question	Answer	Diff
36	What is 2+2?	4	0
36	What is 2+2?	4	10
10	Which type of error in code causes a program to fail to	SYNTAX	0.571
10	Which type of error in code causes a program to fail to...	SYNTAX	6
10	Which type of error in code causes a program to fail to	Answer	0.571
10	Which type of error in code causes a program to fail to...	Answer	6
Test Failed			

However there was an issue with the fourth test case. The text entered wasn't capitalised before it was put into the database meaning a student would never be able to get it correct.

In order to address this issue I added a statement into the loop that runs whilst the 'Question Details' popup is open to constantly capitalise the content of the answer field.

```

if (enter == true) {
    if (rows [focusRow].getID () != "") {
        toggleInputUI (true);
        topicField.text = topicNumber.ToString();
        questionField.text = rows [focusRow].getQuestion ();
        answerField.text = rows [focusRow].getAnswer ();
        enter = false;
        escape = false;
        while (true) {
            answerField.text = answerField.text.ToUpper ();
        }
    }
}

```

This loop runs until the user submits their changes to the question

This line ensures that when the user submits their input the answer field will be capitalised

4.5a: Editing a Question REPEAT	
Reason For Test	
Valid Input Test	
Input Data	Expected Result
(6, Which type of error in code causes a program to fail to compile?,SYNTAX) Answer = 'Answer'	(6, Which type of error in code causes a program to fail to compile?,ANSWER)
I reloaded the database and repeated the failed test case. With the changes implemented the application worked correctly and the answer field was capitalised.	
10 Which type of error in code causes a program to fail to	SYNTAX 0.571
10 Which type of error in code causes a program to fail to...	SYNTAX 6
10 Which type of error in code causes a program to fail to	ANSWER 0.571
10 Which type of error in code causes a program to fail to...	ANSWER 6

Test Passed

4.6: Editing a with invalid values Question	
Reason for Test	
Invalid Input Test	
Input Data	Expected Result
(1, Which storage medium has no moving parts and is, therefore, less likely to wear?,SOLID STATE) Topic = 13	Input not accepted, change to question not attempted
(1, Which storage medium has no moving parts and is, therefore, less likely to wear?,SOLID STATE) Topic =	Input not accepted, change to question not attempted
(9, Which type of translator translates code line by line?,INTERPRETER) Question = ''	Input not accepted, change to question not attempted
(9, Which type of translator translates code line by line?,INTERPRETER) Answer = ''	Input not accepted, change to question not attempted

(5, Convert the hexadecimal number 3E into decimal,62) Question = [String over 200 characters], Answer = [String over 100 characters]	Input not accepted, change to question not attempted
---	--

With the aim of not stifling the users creativity my application doesn't require questions and answers be made up of specific kinds of characters. However there are still ways In which the data entered into the 'Questions Details' popup can be invalid. The topic number entered must be between 1 and 12 (Inclusive) to represent a valid topic, Questions and answers must contain actual Alphanumeric characters to make sense to the user and the make the field not null, and the Question and Answer fields must not exceed 200 and 100 characters because of the database constraints.

I created a test case for each of these scenarios to evaluate if the application responded correctly by rejecting this input. Attempting to change the topic outside the bounds was not handled appropriately. Topic values of 13 or noting were accepted as valid input and led to a foreign key error when this change was applied to the database.

Topic: 13 Topic: Enter

Question: Question:

Which storage medium has no moving parts and Which storage medium has no moving parts and

Answer: Answer:

SOLID STATE SOLID STATE

! MySQLException: Cannot add or update a child row: a foreign key constraint fails MySQL.Data.MySqlClient.MySqlStream.ReadPacket ()

The test cases where the questions or answers field contained no actual characters was also mistreated. The input was accepted as a valid question and the record was updated in the system.

Topic: 9

Question:

Enter text...

Answer:

INTERPRETER

29	INTERPRETER	0
----	-------------	---

In addition, the test case where over-long strings were entered was also accepted as valid input, this led to a length error when the application attempted to add this as a record in the Database

Topic: 5

Question:

With the aim of not stifling the users creativity my

Answer:

ECAUSE OF THE DATABASE CONSTRAINTS. |

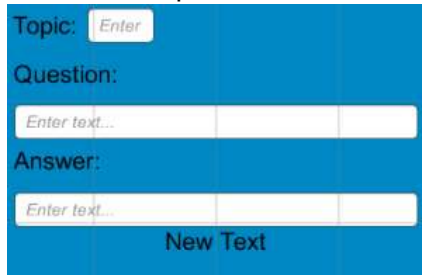
! MySQLException: Data too long for column 'questionText' at row 1 MySQL.Data.MySqlClient.MySqlStream.ReadPacket ()

Test Failed

The first step in rectifying this was to change the validation rules on the input fields:

- I changed the character limit on the topic field to 2 to reduce the range of possible values that could be entered.
- I changed the validation rule on the topic field to only accept integer numbers to ensure there would be no letters in field.
- I changed the character limit on the question field to 200 characters.
- I changed the character limit on the answer field to 100 characters.

I then added an error message text box to the input field to be used in displaying what was wrong with a user's input.

A screenshot of a web form titled "New Text" on a blue background. It features three input fields: "Topic:" with a small "Enter" button, "Question:" with a text box containing "Enter text...", and "Answer:" with a text box containing "Enter text...".

I passed this new text field to the script and adjusted the enter branch of the main program loop.

```
while (true) {
    answerField.text = answerField.text.ToUpper ();
    if (enter) {
        if (topicField.text.Length == 0) {
            errorMessageText.text = "Please enter a topic";
        } else if (int.Parse (topicField.text) < 1 || int.Parse (topicField.text) > 12) {
            errorMessageText.text = "Invalid Topic Number: Enter a topic between 1 and 12";
        } else if (questionField.text.Replace (" ", "").Length == 0) {
            errorMessageText.text = "No question entered";
        } else if (answerField.text.Replace (" ", "").Length == 0) {
            errorMessageText.text = "No answer entered";
        } else {
            databaseScript.updateQuestion (questionField.text, answerField.text,
                topicField.text, rows [focusRow].getID());
            Populate ();
            break;
        }
    }
    enter = false;
}
```

The if statement works as follows to catch and inform the user of error:

- The first statement evaluates to true if the topic field is empty and informs the user the question needs a topic.
- Because of the character validation applied I know the contents of the topic field will only be numbers. The second statement can therefore convert the contents to an integer and use inequalities to work out if it is inside the range. If not an error message is displayed.
- The third and fourth statements use the Replace (" ", "") function to find the number of alphanumeric characters in the questions and answer fields. An error message is displayed if this length is 0.
- Only if all previous statements have been evaluated to false will the application attempt to change the question in the database.

4.6a: Editing a with invalid values Question REPEAT	
Reason for Test	
Invalid Input Test	
Input Data	Expected Result
(1, Which storage medium has no moving parts and is, therefore, less likely to wear?,SOLID STATE) Topic = 13	Input not accepted, change to question not attempted, Error message displayed
(1, Which storage medium has no moving parts and is, therefore, less likely to wear?,SOLID STATE) Topic =	Input not accepted, change to question not attempted, Error message displayed
(9, Which type of translator translates code line by line?,INTERPRETER) Question = ''	Input not accepted, change to question not attempted, Error message displayed
(9, Which type of translator translates code line by line?,INTERPRETER) Answer = ''	Input not accepted, change to question not attempted, Error message displayed
(5, Convert the hexadecimal number 3E into decimal,62) Question = [String over 200 characters], Answer = [String over 100 characters]	Only first 200 and 100 characters respectively accepted, Change executed

Up repeating this test with the same test cases the input was treated as expeted. The invalid topic and empty fields cases were not accepted and their respective error messages were displayed.

In the case of the overlong input the fields were cut off after they reached their character limit so the changes could be submitted to the database without resulting in an error.

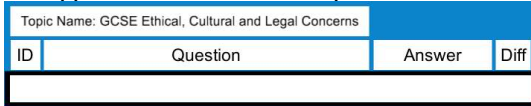
<p>Topic: <input type="text" value="5"/></p> <p>Question:</p> <p><input type="text" value="With the aim of not stifling the users creativity my"/></p> <p>Answer:</p> <p><input type="text" value="WITH THE AIM OF NOT STIFLING THE USERS"/></p>
Test Passed

Knowing that the input popup would also be used for adding questions and that similar validation problems would occur, I copied over a lot of the changes made the editing portion of the main loop.

```

errorMessageText.text = "";
while (true) {
    answerField.text = answerField.text.ToUpper ();
    if (enter) {
        if (topicField.text.Length == 0) {
            errorMessageText.text = "Please enter a topic";
        } else if (int.Parse (topicField.text) < 1 || int.Parse (topicField.text) > 12) {
            errorMessageText.text = "Invalid Topic Number: Enter a topic between 1 and 12";
        } else if (questionField.text.Replace (" ", "").Length == 0) {
            errorMessageText.text = "No question entered";
        } else if (answerField.text.Replace (" ", "").Length == 0) {
            errorMessageText.text = "No answer entered";
        } else {
            databaseScript.addQuestion (questionField.text, answerField.text, topicField.text);
            Populate ();
            break;
        }
        enter = false;
    }
}

```

4.7: Attempting to Edit with no Question Selected		
Reason for Test	Input Data	Expected Output
Invalid Input Test	Enter, with no Question selected	Nothing
<p>It is entirely possible to be on the tab for a specific topic even when the topic has no questions. In this scenario the questions outline, although immovable, still appears on the screen; it is entirely possible that a user could press Enter even though there is no question to edit. In this scenario the application shouldn't accept this as a valid input and should do nothing.</p> 		
To test this I deleted all the questions from a particular topic and then pressed Enter. The application correctly did nothing with this input.		
Test Passed		

4.8: Adding a Question	
Reason for Test	
Valid Input Test	
Input Data	Expected Output
Topic = 1, Question = 'Who?', Answer = 'What'	New Question (PK,Who,WHAT,1)
Topic = 12, Question = '#@//', Answer = '#@//'	New Question (PK,#@//,#@//,12)

Topic = 5, Question = '(2+2)*3', Answer = '12'		New Question (PK,(2+2)*3,12,5)	
<p>As part of my brief, Staff users need to be able to edit the questions list and this encompasses the ability to add new questions. By pressing 'a' at any point, the user should cause the 'Question Details' Popup appearing. Pressing Enter should then result in a new question being added to the database with the same values as those entered into the popup.</p> <p>To test this I opened the Questions scene and attempted to add several new questions. I made sure to test a variety of topics and use all the different kinds of characters that should be accepted as part of the question and answer. On every occasion the application worked perfectly and the new questions were successfully added.</p>			
<p>Topic: 1</p> <p>Question: Who?</p> <p>Answer: WHAT</p>			
45 Who?		WHAT 0	
<p>Topic: 12</p> <p>Question: #@//</p> <p>Answer: #@//</p>			
46 #@//		#@// 0	
<p>Topic: 5</p> <p>Question: (2+2)*3</p> <p>Answer: 12</p>			
47 (2+2)*3		12 0	
45	Who?	WHAT	1
46	#@//	#@//	12
47	(2+2)*3	12	5
Test Passed			
4.9: Adding with Invalid Values			
Reason for Test			
Invalid Input Test			
Input Data		Expected Result	
Topic = 13, Question = 'Who?', Answer = 'What'		Input not accepted, creating question not attempted, Error message displayed	
Topic = , Question = 'Who?' Answer = 'What'		Input not accepted, creating question not attempted, Error message displayed	

Topic = 1, Question = '', Answer = 'What'	Input not accepted, creating question not attempted, Error message displayed
Topic = 1, Question = 'Who?', Answer = ''	Input not accepted, creating question not attempted, Error message displayed
Topic = 1, Question = [String over 200 characters], Answer = [String over 100 characters]	Only first 200 and 100 characters respectively accepted, New Question (PK,[200],[100],1)

With the aim of not stifling the users creativity my application doesn't require questions and answers be made up of specific kinds of characters. However there are still ways in which the data entered into the 'Questions Details' popup can be invalid. The topic number entered must be between 1 and 12 (Inclusive) to represent a valid topic, Questions and answers must contain actual Alphanumeric characters to make sense to the user and the make the field not null, and the Question and Answer fields must not exceed 200 and 100 characters because of the database constraints.

I created a test case for each of these scenarios to and the application responded correctly. In the case of a topic outside the range the program displayed the correct error message.

Topic: 13
Question:
Who?
Answer:
WHAT
Invalid Topic Number: Enter a topic between 1 and 12

This was also the case for any test where one of the fields was empty.

Topic: Enter
Question:
Who?
Answer:
WHAT
Please enter a topic

Topic: 1
Question:
Enter text...
Answer:
WHAT
No question entered

Topic: 1
Question:
Who?
Answer:
Enter text...
No answer entered

In the case where the input to the question and answer fields was too long the application correctly only accepted the first 200 and 100 characters and the question was added.

Test Passed

Criteria 7: My solution must allow teacher to view students' performance both numerically and in a bar chart

Testing this portion of the application required me to generate some test data. To test this in a useful way this pool of data needed to be very large. There are around 150 Computing students at my school and, with consistent use over the school year, this will amount to a large amount of answers and individual data items that the results tab needs to sort through. It was therefore imperative that I test the results scene with an amount of data that mimics the real life use of the application to make sure it would function correctly when implemented.

This amount of data would have taken an unreasonable amount of time to generate manually so I looked to find a tool that would automate the process. I found a website called 'generateData.com' that performed exactly the task I wanted. The website asked me to enter the details of my tables and, with this information, generated MySQL insert statements.

<http://www.generatedata.com/#generator>

The first step was to generate a pool of students based on the rules of my database.

Table Column	Data Type	Examples	Options
studentID	Auto-increment	1, 2, 3, 4, 5, 6...	Start at: 1000 Increment: 1 Placeholder string:
firstName	Names	Alex (any gender)	Name
lastName	Names	Smith (surname)	Surname
classID	Custom List	Please Select	<input checked="" type="radio"/> Exactly 1 <input type="radio"/> At Most 1 Enter values separated by 10A 10B 11A 11B 12A 13A

- studentID: Set to increment up from 1000 with a step of 1. This will produce valid candidate numbers as long as less than 9000 records are generated.
- firstName: Set to pick a random value from a list of typical first names
- lastName: Set to a random value from a list of typical surnames
- classID: Set to randomly pick from the class primary keys. _classID wasn't accepted as a valid field name for this will need to be changed before the script is run.

This generated an output made up of insert statements for the student class

```
INSERT INTO `student` (`studentID`,`firstName`,`lastName`,`classID`) VALUES (1000,'Wayne','Freeman','11A'),(1001,'Griffith','Barnes','13A'),(1004,'Lyle','Combs','11B'),(1005,'Kelly','Bruce','10A'),(1006,'Arthur','Martin','11B'),(1007,'Reese','Ford','10A'),(1008,'Ian','Manning','12A')
INSERT INTO `student` (`studentID`,`firstName`,`lastName`,`classID`) VALUES (1010,'Jerry','Mann','10B'),(1011,'Marsden','Bender','10A'),(1014,'Nathan','O'Neill','11B'),(1015,'Murphy','Smith','10A'),(1016,'Ethan','Shannon','12A'),(1017,'Carter','Hayes','12A'),(1018,'Chester','St
INSERT INTO `student` (`studentID`,`firstName`,`lastName`,`classID`) VALUES (1020,'Addison','Kerr','12A'),(1021,'Wyatt','Mack','11A'),(1022,'1025','Philip','Mitchell','12A'),(1026,'Christian','Chambers','11B'),(1027,'Marshall','Cochran','12A'),(1028,'Theodore','Sullivan','13A'),(1029
```

The site would only generate 100 statements at a time so I ran in twice, taking care to adjust the range of the primary key generator. Running these statements populated the student table as expected.

studentID	firstName	lastName	_classID
1000	Malcolm	Burch	12A
1001	Kimberly	Hooper	11B
1002	Wayne	Fowler	12A
1003	Sean	Wheeler	10A
1004	Cleo	Osborne	10A
1005	Aiko	Mercer	10B
1006	Channing	Terrell	10B
1007	Vladimir	Nichols	10B
1008	Heidi	George	11B
1009	Ian	Puckett	10A
1010	Katelyn	Holder	10A

I then setup the website to generate records for the 'answerInstance' table.

Table Column	Data Type	Examples	Options
correct	Number Range	No examples available.	Between 0 and 1
studentID	Number Range	No examples available.	Between 1000 and 1199
questionID	Number Range	No examples available.	Between 1 and 43

- correct: Set to take on values between 0 and 1
- studentID; Set to take values between 1000 and 1199, the ID's of students in the database
- questionID: Set to take values between 1 and 43, the ID's of questions in the database
- The other fields in the table are set to be auto filled, I adjusted my computers clock to differ the dates of records

This generated insert statements in exactly the way I wanted.

```
INSERT INTO `answerInstance` (`correct`, `studentID`, `questionID`) VALUES (0,1111,27),(1,1101,42),(0,1021,32),(1,1139,3)
INSERT INTO `answerInstance` (`correct`, `studentID`, `questionID`) VALUES (0,1093,20),(1,1146,29),(0,1066,19),(0,1130,3)
INSERT INTO `answerInstance` (`correct`, `studentID`, `questionID`) VALUES (1,1056,8),(1,1126,3),(1,1182,11),(1,1103,43)
```

I ran the website several times and ended up with statements for 3000 answers. Running these statements as a MySQL script worked to populate the database with a suitable data set. The only change I needed to make was to disable the option that limited the amount of results SELECT would return in MySQL.

answerInstanceID	correct	date	_studentID	_questionID
1	0	2017-04-23 14:21:24	1153	40
2	1	2017-04-23 14:21:24	1073	32
3	0	2017-04-23 14:21:24	1117	8
4	0	2017-04-23 14:21:24	1191	16
5	0	2017-04-23 14:21:24	1049	19
6	1	2017-04-23 14:21:24	1022	17

SELECT Query Results

Limit Rows

Limit Rows Count:

To test the calculations and graphing performed by my application I will be using Google Sheets and using formulas to perform the same actions on the same data. The results from these two sources being the same will indicate that my application is working correctly.

A	B	C	D	E
Correct	Date	Topic	Student ID	Class
1	23/04/2017		1028	10B
1	23/04/2017		1120	11A
0	23/04/2017		1067	13A
1	23/04/2017		1004	10A
0	23/04/2017		1078	12A
0	23/04/2017		1062	10A
0	23/04/2017		1191	13A
0	23/04/2017		1137	10B
1	23/04/2017		1124	11B

7.1: Whole Student Body / Student / General

Reason For Test	Input Data
Valid Input Test	Whole Student Body / Student / General

Through the process of investigation and the development of the application I decided one of the ways the data should be displayed is the whole student base compared against each other. To generate the expected result for this I created a pivot table in the Google Sheet with the following specifications:

Rows - Add field

Group by: Student ID ×

Order: Ascending ▾

Sort by: Student ID ▾

Show totals

Columns - Add field

Values - Add field

Display: Correct ×

Summarise by: AVERAGE ▾

Filter - Add field

- The Row setting means that results will be grouped together by Student, fulfilling the second part of the results grouping.

- The lack of a columns setting means the only value that will be shown is the total. This is suitable because the grouping setting is to look at results generally not by topic

-The values setting is set to calculate an average from the correct field. This is what the application does and will be the value I will compare.

-I adjusted the filter to show the same students as in the application.

The results given by the application and the sheets table consistently matched with each other meaning that the results scene had calculated its values correctly.

Whole Student Body ▾
Student: General ▾

-Name-	-Data-		A	B
1028	1.000			
1120	0.571	1	1004	1
1067	0.571	2	1028	1
1004	1.000	3	1067	0.5714285714
		4	1120	0.5714285714

Test Passed

7.2: Whole Student Body / Student / Topic

Reason For Test	Input Data
Valid Input Test	Whole Student Body / Student / System's Architecture Whole Student Body / Student / Software Whole Student Body / Student / Data Representation

A way data should be displayed in the table is the whole student body's performance in a particular topic. To evaluate this in Google Sheets I set the following specifications.

Rows - Add field

Group by: Student ID ×

Order: Ascending ▾

Sort by: Student ID ▾

Show totals

Columns - Add field

Group by: Topic ×

Order: Ascending ▾

Sort by: Topic ▾

Show totals

Values - Add field

Display: Correct ×

Summarise by: AVERAGE ▾

Filter - Add field

- The Row setting means that results will be grouped together by Student, fulfilling the second part of the results grouping.

- The columns setting means the values will be broken down by topic, fulfilling the third part of the results grouping.

-The values setting is set to calculate an average from the correct field. This is what the application does and will be the value I will compare.

-I adjusted the filter setting so it would show the same students and topics as in the application.

Because of the way I generated test data it was possible for students to have answers to both the GCSE and the A-Level version of a topic. This didn't interfere with the applications function it just meant that results for a specific topic occurred across two different columns in the Sheets document. The results given by the application and the sheets table consistently matched with each other meaning that the results scene had calculated its values correctly.

Whole Student Body ▾ Student: Topic ▾ System's Architecture ▾

			A	B	C	D
1124	1.000	1		1	7	Grand Total
1184	0.000	2	1124	1	1	1
1152	0.500	3	1152	0.5		0.5
1178	0.000	4	1178	0	0	0
		5	1184	0		0

Whole Student Body ▾ Student: Topic ▾ Software ▾

			A	B	C	D
1157	0.333	1		3	9	Grand Total
1029	1.000	2	1029	1		1
1192	1.000	3	1084	0		0
1084	0.000	4	1157	0	0.5	0.3333333333
		5	1192	1		1

Whole Student Body ▾ Student: Topic ▾ Data Representation ▾

1015	0.667		A	B	C	D
1003	0.500	1		5	11	Grand Total
1186	1.000	2	1003	1	0	0.5
1199	0.500	3	1015	1	0	0.666666667
		4	1186	1		1
		5	1199	0.5		0.5

Test Passed

7.3: Whole Student Body / Class / General

Reason For Test Input Data

Valid Input Test Whole Student Body / Class / General

A way data should be displayed in the table is all the classes generally. To evaluate this in Google Sheets I set the following specifications.

Rows - Add field

Group by: Class

Order: Ascending

Sort by: Class

Show totals

Columns - Add field

Values - Add field

Display: Correct

Summarise by: AVERAGE

Filter - Add field

- The Row setting means that results will be grouped together by Class, fulfilling the second part of the results grouping.

- The lack of a columns setting means the only value that will be shown is the total. This is suitable because the grouping setting is to look at results generally not by topic

-The values setting is set to calculate an average from the correct field. This is what the application does and will be the value I will compare.

-I adjusted the filter setting to match so I could compare so the table would show the same students and topics as in the application.

The results given by the application and the sheets table consistently matched with each other meaning that the results scene had calculated its values correctly.

Whole Student Body Class: General

10B	0.489		A	B
11A	0.476	1	10A	0.5092592593
13A	0.396	2	10B	0.4891304348
10A	0.509	3	11A	0.4761904762
12A	0.477	4	11B	0.5666666667
11B	0.567	5	12A	0.476744186
		6	13A	0.3956043956

Test Passed

7.4: Whole Student Body / Class / Topic

Reason for Test Input Data

Valid Input Test Whole Student Body / Class / System's Architecture
Whole Student Body / Class / Software
Whole Student Body / Class / Data Representation

A way data should be displayed in the table is classes' performance in a particular topic. To evaluate this in Google Sheets I set the following specifications:

Rows - Add field

Group by: Class ✕

Order: Ascending ▾

Sort by: Class ▾

Show totals

Columns - Add field

Group by: Topic ✕

Order: Ascending ▾

Sort by: Topic ▾

Show totals

Values - Add field

Display: Correct ✕

Summarise by: AVERAGE ▾

Filter - Add field

- The Row setting means that results will be grouped together by class, fulfilling the second part of the results grouping.

- The columns setting means the values will be broken down by topic, fulfilling the third part of the results grouping.

-The values setting is set to calculate an average from the correct field. This is what the application does and will be the value I will compare.

-I adjusted the filter setting so it would show the same topics as in the application.

Because of the way I generated test data it was possible for students to have answers to both the GCSE and the A-Level version of a topic. This didn't interfere with the applications function it just meant that results for a specific topic occurred across two different columns in the Sheets document. The results given by the application and the sheets table consistently matched with each other meaning that the results scene had calculated its values correctly.

Whole Student Body ▾ Class: Topic ▾ System's Architecture ▾

10B	0.450					
11A	0.391					
13A	0.294					
10A	0.489					
12A	0.441					
11B	0.810					
		A	B	C	D	
		1	1	7	Grand Total	
		2	10A	0.4444444444	0.5	0.4888888889
		3	10B	0.4117647059	0.4782608696	0.45
		4	11A	0.4285714286	0.375	0.3913043478
		5	11B	0.75	0.8235294118	0.8095238095
		6	12A	0.4285714286	0.4444444444	0.4411764706
		7	13A	0.2222222222	0.32	0.2941176471

Whole Student Body ▾ Class: Topic ▾ Software ▾

11B	0.462					
13A	0.471					
12A	0.556					
10A	0.647					
10B	0.667					
11A	0.778					
		A	B	C	D	
		1	3	9	Grand Total	
		2	10A	0.5714285714	0.7	0.6470588235
		3	10B	0.7142857143	0.625	0.6666666667
		4	11A	0.75	0.8	0.7777777778
		5	11B	0.5714285714	0.3333333333	0.4615384615
		6	12A	0.5	0.6	0.5555555556
		7	13A	0.25	0.6666666667	0.4705882353

Whole Student Body ▾ Class: Topic ▾ Data Representation ▾

10B	0.579				
13A	0.476				
11A	0.467	1		5	11
10A	0.565	2	10A	0.5454545455	0.5833333333
12A	0.500	3	10B	0.5714285714	0.6
11B	0.500	4	11A	0.5	0.4285714286
		5	11B	0.4444444444	0.6
		6	12A	0.375	0.6666666667
		7	13A	0.4705882353	0.5
					0.4761904762

Test Passed

7.5: Classes / Class / General

Reason for Test	Input Data
Valid Input Test	Class / 10A / General Class / 11B / General Class / 12A / General

A way data should be displayed in the table is the general performance of students in a particular class. To evaluate this in Google Sheets I set the following specifications:

Rows - Add field

Group by: Student ID

Order: Ascending

Sort by: Student ID

Show totals

Columns - Add field

Values - Add field

Display: Correct

Summarise by: AVERAGE

Filter - Add field

Filter: Class

Show: 1 items

- The Row setting means that results will be grouped together by student as they are in the application.
- The lack of a columns setting means the only value that will be shown is the total. This is suitable because the grouping setting is to look at results generally not by topic
- The values setting is set to calculate an average from the correct field. This is what the application does and will be the value I will compare.
- I adjusted the filter setting to change which class was displayed.

The results given by the application and the sheets table consistently matched with each other meaning that the results scene had calculated its values correctly.

Classes
10A: General

1149	0.333			
1163	1.000	1	1087	0.6666666667
1129	0.500	2	1129	0.5
		3	1149	0.3333333333
1087	0.667	4	1163	1

Classes
10B: General

			A	B
1028	1.000			
1137	0.250	1	1028	1
1152	0.333	2	1118	1
		3	1137	0.25
1118	1.000	4	1152	0.3333333333

			A	B
1189	0.667			
1084	0.167	1	1084	0.1666666667
1185	0.667	2	1182	0.3333333333
		3	1185	0.6666666667
1182	0.333	4	1189	0.6666666667

Test Passed

7.6: Classes / Class / Topic

Reason for Test	Input Data
Valid Input Test	Class / 10A / Exchanging Data Class / 10A / Ethical, Cultural and Legal Concerns Class / 10A / Programming

A way data should be displayed in the table is the performance of students in a particular class in a particular topic. To evaluate this in Google Sheets I set the following specifications:

Rows - Add field

Group by: Student ID ×

Order: Ascending ▾

Sort by: Student ID ▾

Show totals

Columns - Add field

Group by: Topic ×

Order: Ascending ▾

Sort by: Topic ▾

Show totals

Values - Add field

Display: Correct ×

Summarise by: AVERAGE ▾

Filter - Add field

- The Row setting means that results will be grouped together by student as they are in the application.

- The columns setting means the values will be broken down by topic, fulfilling the third part of the results grouping.

- The values setting is set to calculate an average from the correct field. This is what the application does and will be the value I will compare.

- I adjusted the filter so it would show the same topics and students as the application.

Because of the way I generated test data it was possible for students to have answers to both the GCSE and the A-Level version of a topic. This didn't interfere with the applications function it just meant that results for a specific topic occurred across two different columns in the Sheets document. The results given by the application and the sheets table consistently matched with each other meaning that the results scene had calculated its values correctly.

Classes	10A: Topic	Exchanging Data
---------	------------	-----------------

1075	0.500		A	B	C	
		1			2	Grand Total
1093	1.000	2	1016	0		0
1016	0.000	3	1075	0.5		0.5
		4	1093	1		1
1112	1.000	5	1112	1		1
<div style="display: flex; justify-content: space-between; margin-top: 10px;"> Classes <input type="text" value="Classes"/> 10A: Topic <input type="text" value="10A: Topic"/> Ethical, Cultural and Legal <input type="text" value="Ethical, Cultural and Legal"/> </div>						
1090	0.000		A	B	C	D
		1		4	10	Grand Total
1030	0.000	2	1030	0		0
1058	1.000	3	1058	1		1
		4	1090	0		0
1116	0.000	5	1116		0	0
<div style="display: flex; justify-content: space-between; margin-top: 10px;"> Classes <input type="text" value="Classes"/> 10A: Topic <input type="text" value="10A: Topic"/> Programming <input type="text" value="Programming"/> </div>						
1114	1.000		A	B	C	D
		1		6	12	Grand Total
1119	0.000	2	1010		0	0
1075	0.000	3	1075	0		0
		4	1114	1	1	1
1010	0.000	5	1119	0		0
Test Passed						
7.7: Individual Students						
Reason For Test	Input Data					
Valid Input Test	Individual Students / 1000 / General Individual Students / 1168 / System's Architecture Individual Students / 1193 / Data Representation					
A way data should be displayed in the table is the performance of students on a particular day, both generally and by topic. To evaluate this in Google Sheets I set the following specifications:						

Rows - Add field

Group by: Date ✕

Order: Ascending ▾

Sort by: Date ▾

Show totals

Columns - Add field

Group by: Topic ✕

Order: Ascending ▾

Sort by: Topic ▾

Show totals

Values - Add field

Display: Correct ✕

Summarise by: AVERAGE ▾

Filter - Add field

Filter: Topic ✕

Show: 2 items ▾

Filter: Student ID ✕

Show: 4 items ▾

- The Row setting means that results will be grouped together by data as per the first part of the grouping settings.

- The columns setting means the values will be broken down by topic, however show totals is ticked, this means the third part of the grouping settings can be fulfilled

- The values setting is set to calculate an average from the correct field. This is what the application does and will be the value I will compare.

- I adjusted the filter setting to change which student and topic was displayed.

Because of the way I generated test data it was possible for students to have answers to both the GCSE and the A-Level version of a topic. This didn't interfere with the applications function it just meant that results for a specific topic occurred across two different columns in the Sheets document. The results given by the application and the sheets table consistently matched with each other meaning that the results scene had calculated its values correctly.

Individual Students ▾

1000: General ▾

Date	Value	ID	A	B
23/04/2017	1.000			
26/04/2017	1.000	1	23/04/2017	1
27/04/2017	1.000	2	24/04/2017	1
24/04/2017	1.000	3	26/04/2017	1
		4	27/04/2017	1

Individual Students ▾

1168: Topic ▾

System's Architecture ▾

Date	Value	ID	A	B
23/04/2017	1.000			
24/04/2017	1.000	1	23/04/2017	1
27/04/2017	1.000	2	24/04/2017	1
		3	27/04/2017	1

Individual Students ▾

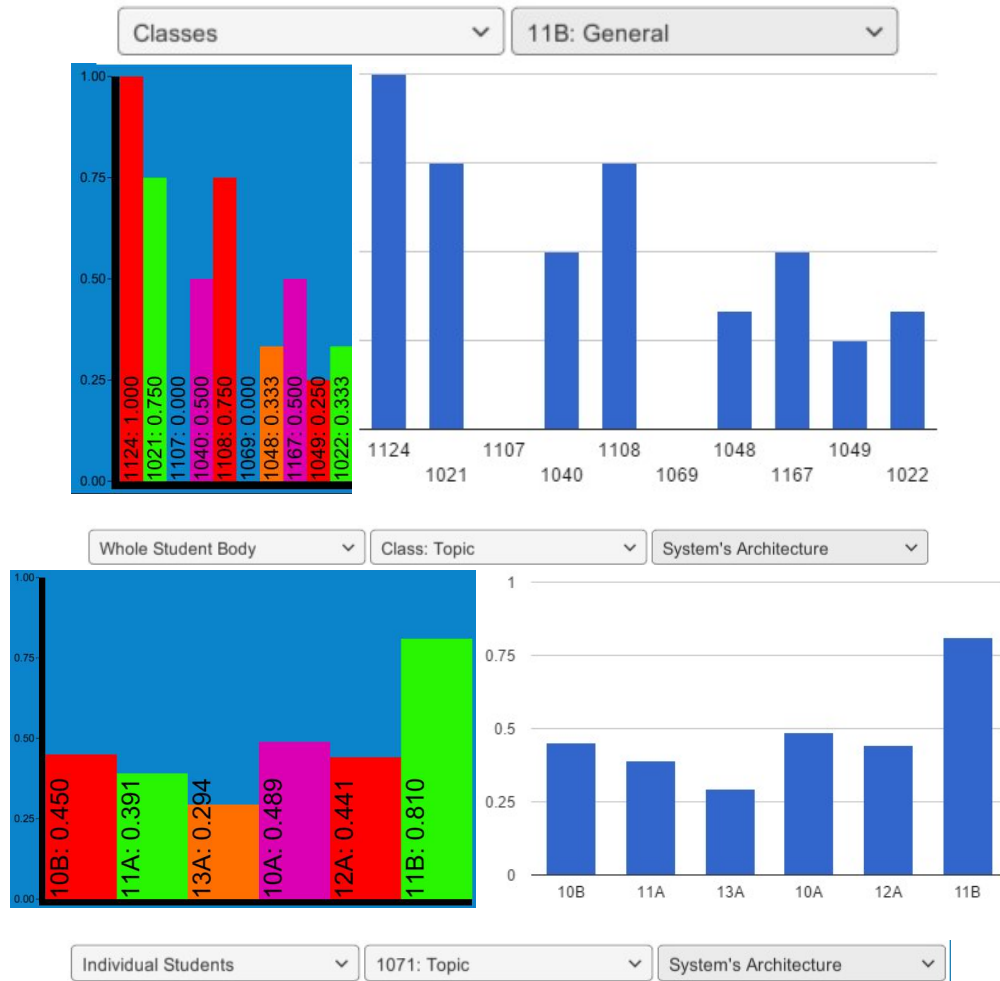
1193: Topic ▾

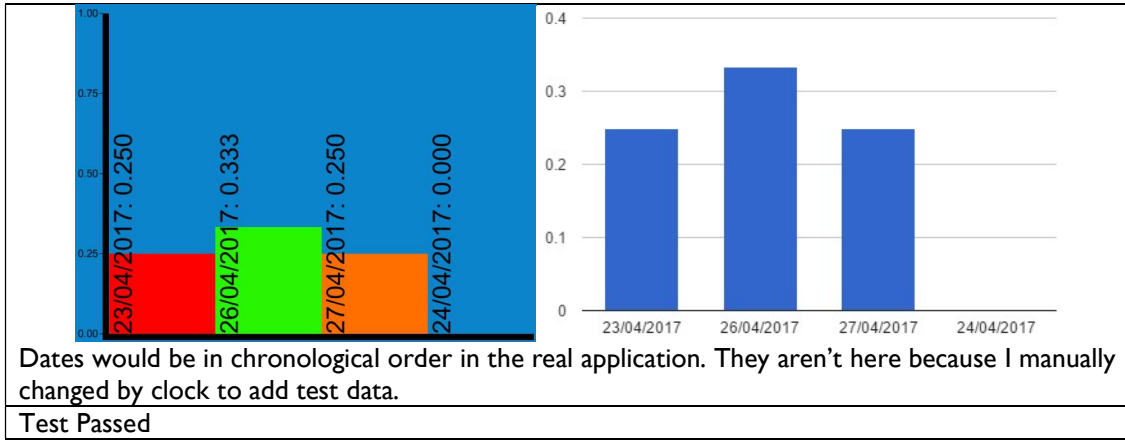
Data Representation ▾

23/04/2017	0.000		A	B
24/04/2017	0.000	1	23/04/2017	0
		2	24/04/2017	0
27/04/2017	0.000	3	27/04/2017	0

Test Passed	
7.8: Graphing	
Reason for Test	Input Data
Valid Input Test	Classes / IIB / General Whole Student Body / Class / System's Architecture Individual Student / 1071 / System's Architecture

As a part of my project brief, student results need to be displayed in bar graph form as well as graphically. The charting part of the application simply takes the values in the table and plots them so the actual calculation of values doesn't need to be tested. Instead I need to compare the graph produces will graph made from the same information in Google Sheets to see if the application is plotting results correctly. The screenshots of graphs shown below display how graphing was consistently correct





During this test I decided the Results scene would be a lot more intuitive if users were told how to graph. I added a text box with this information on to the scene for this purpose. I made the box a child of the Headings game object so it wouldn't be in the way when the graph appeared.



7.9: Extreme Graphing		
Reason for Test	Input Data	Expected Result
Extreme Input Test	0 Records 200 Records	Graph screen loaded, empty graph displayed Graph screen loaded, full graph displayed

When faced with an extreme amount of records the application should react as it would in a normal case. To test this I attempted to create a graph with both no records and 200 records needing to be displayed.

Firstly I created a new student record in the database with no answers attributed to it. When I attempted to graph the results of this student the application didn't encounter an error and continued to display an empty graph.

I then used the test data generate statements to get make it so that there were 200 students. When I attempted to graph the bar chart was successfully generated. However, the text on the chart very small and would need zooming in to read. The application performed the desired task perfectly but the end result was restricted by the way the charting works. If I had unlimited time to complete this project I could attempt to fix this issue. However I do not and because this issue doesn't prevent the application working it must be left as a limitation of the project.

Test Passed

With this final test complete the whole of my application is tested and I can speak with certainty that my application is fully functional and has the robustness to deal with real users and the volume of

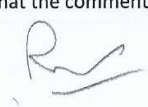
users my application will deal with in its real life implementation. I am now confident in deeming my application User-Ready and can now move on to presenting it to actual End-Users and my Client for their evaluation.

Beta Testing

To properly evaluate my application it was important that I present it to actual end users to gauge their unbiased opinion of my work. I created some Beta Testing forms that I asked users to fill out as they were using the application. There were two version of this form to accommodate for the two different groups of end users. The forms also began with a list of tasks before allowing the users to use the application as they pleased. This ensured that I would be able to collect user opinions on the full breadth of the project features.

Students: I first presented the application to some potential student end users:

- First Student User:

Student User Beta Testing Form	
Name: <u>Ronan W</u>	Date: <u>26/04/2017</u>
I consent that the comments given are my own and are given honestly	
Signature: 	
Please attempt the following tasks, comment on how easy it was for you to achieve this task and how straight forward you thought the process was.	
Get to the student portion of the application and login to the system	
<u>Straight forward,</u>	
Move around the map to a quiz	
<u>The map is very easy to look at and use</u>	
Take the Quiz	
<u>The quiz is good, but the answers need to be precise. Apart from that, really enjoyed the battle focus!</u>	
Move around the map to an item	
<u>Very simple, icons are clear</u>	
Equip / Use an item you have collected	
<u>Inventory screen works like a charm and is easy to use</u>	
Delete / Drop and item you have collected	
<u>Concise and easy to do</u>	

My first user student user expressed largely positive things about the function of the application. The only issue he raised was that user input needs to be a complete match to the answer stored in the database to be evaluated as correct. He identified that this didn't allow for 'typos' and misspellings.

Please Now play the game as you like for as long as you feel you need to properly evaluate the application's effectiveness. Please then answer the following questions.
Was the application generally simple or complicated to use
Very simple,
Did you enjoy using this revision more or less than conventional methods
I liked the fact it was a game, rewarding you for doing well in the quiz
Do you think the way questions are asked in the application is effective or not effective at helping you revise the computing content
Yes, but typos are possible which is annoying
Could you see yourself using this application as a part of your revision
Yes
Do you encounter of what you would perceive to be an error or bug with the application
No, nothing of the sort
Given unlimited time and unlimited and unlimited resources, could you suggest any improvements to the application
Typos wont happen, more content like sprites and characters.

The User expressed that the portion of the application he interacted with was simple to use and that he encountered no issues or errors. The only suggestion he made for hoe the game could be improved is making the answers more flexible to typing errors.

- Second Student User:

Student User Beta Testing Form	
Name: Josh R 10AJS	Date: 25/04/17
I consent that the comments given are my own and are given honestly	
Signature: <i>[Signature]</i>	
Please attempt the following tasks, comment on how easy it was for you to achieve this task and how straight forward you thought the process was.	
Get to the student portion of the application and login to the system?	
It was very easy to complete this as instructions were displayed to get to the menu and the login was as straight forward as any	
Move around the map to a quiz?	
Was slightly confusing as to what was what on the map screen however anyone could figure it out.	
Take the Quiz?	
Easy to use however annoying how I would have to put "Driver" instead of "Drivers". Maybe make it multiple choice?	
Move around the map to an item?	
Easy - no faults	
Equip / Use an item you have collected?	
Fairly straight forward, not very clear on what the items do.	
Delete / Drop an item you have collected?	
You literally have to press space which is very easy.	

The second student user also commented that the tasks were relatively easy to accomplish but that he would like it if the answers were more flexible to different phrasings and spellings. This user also said that the way the map and items system worked was not immediately apparent, although he could "figure it out" in the time he spent with the application.

Please Now play the game as you like for as long as you feel you need to properly evaluate the application's effectiveness. Please then answer the following questions.
Was the application generally simple or complicated to use?
It was very simple considering it was made from scratch with no animation
Did you enjoy using this revision more or less than conventional methods?
More because it gave me a sense of accomplishment.
Do you think the way questions are asked in the application is effective or not effective at helping you revise the computing content?
They are effective however answering was had to be very specific which was annoying.
Could you see yourself using this application as a part of your revision?
Yes
Do you encounter of what you would perceive to be an error or bug with the application?
No
Given unlimited time and unlimited and unlimited resources, could you suggest any improvements to the application?
Multiple choice questions (or some over variation), animations and sound to make the item system more clear.

The second student user also commented that the system was simple to use and said it provided a "Sense of encouragement" that encouraged play. The user commented that he didn't find any errors or issue with the application but suggested adding some form of validation to accept correct but not accurate answers as well as some more on screen information to make the mechanics of the game clearer.

- Third Student User:

Please attempt the following tasks, comment on how easy it was for you to achieve this task and how straight forward you thought the process was.
Get to the student portion of the application and login to the system?
It was relatively easy, although it could have had examples for each input.
Move around the map to a quiz?
It was straight forward although there could have been on screen input feedback.
Take the Quiz?
Inputting answers was easy and straight forward.
Move around the map to an item?
This was clear and straight forward.
Equip / Use an item you have collected?
The color coordination made this section straight forward, although there could be more of a distinction between equipped items.
Delete / Drop an item you have collected?
It would be the same as for equipping an item, so easy.

The third user described most of the tasks as “straightforward” to accomplish. The user did suggest that there could be more information displayed on screen to explain how the different mechanics worked.

Please Now play the game as you like for as long as you feel you need to properly evaluate the application's effectiveness. Please then answer the following questions.
Was the application generally simple or complicated to use?
it was generally simple to use, altho some sections were more complicated.
Did you enjoy using this revision more or less than conventional methods?
I thought that it was good for being neutral of some information.
Do you think the way questions are asked in the application is effective or not effective at helping you revise the computing content?
I think that it is effective for the written information in the exam, such as the essay questions.
Could you see yourself using this application as a part of your revision?
Yes
Do you encounter of what you would perceive to be an error or bug with the application?
Yes No, it was perfect
Given unlimited time and unlimited and unlimited resources, could you suggest any improvements to the application?
Yeah, maybe it could be more similar to games such as pokémon and could have 3D graphics.

The third student user commented that he thought the application would be able to become an effective part of his revision. The user also commented that he didn't experience any issues or errors. As for future developments, the user suggested that I could implement more complex '3D graphics' to improve the game aspects of the application.

- First Staff User:

Staff User Beta Testing Form	
Name: <u>MARC SNELLING</u>	Date: <u>26/04/17</u>
I consent that the comments given are my own and are given honestly	
Signature: <u>M Snelling</u>	
Please attempt the following tasks, comment on how easy it was for you to achieve this task and how straight forward you thought the process was.	
Get to the staff portion of the application	
EXCEEDINGLY EASY TO ACCESS, CLEAR SIMPLE INSTRUCTIONS GIVEN TO PRESS ↓ KEY TO ACCESS	
Add a new question to the topic 'GCSE System's Architecture'	
AGAIN SIMPLE INSTRUCTIONS FOR ARROW KEYS, EASY TO FOLLOW, I ENTERED THE QUESTION AND ANSWER AFTER SUCCESSFUL MENU NAVIGATION. WHEN ASKED TO ENTER THE TOPIC I HIT A SNAG, I TRIED TO ENTER THE STRING "MEMORY" BUT THE PROGRAM ONLY ACCEPTS INTEGER INPUT FOR TOPICS.	
Delete the Question "Convert the hexadecimal number 3E into decimal"	
EASY TO NAVIGATE TO AND DELETE THE QUESTION, I HAD TO CHANGE TOPIC TO SEE THE HEX QUESTION BUT THIS WAS EASILY UNDERSTOOD FROM READING THE INSTRUCTIONS	
Edit the question with a primary key '36', give it a new topic number, question and answer	
EASY TO LOCATE 36 ENTERING INFORMATION INTO THE TEXT BOXES AND FOLLOWING THE MENU WAS SIMPLE.	
Find the most difficult question in the topic 'A-Level System's architecture'	
QUESTIONS ARE RANKED BY DIFFICULTY, I SURMISED THE HIGHER THE NUMBER THE MORE DIFFICULT THE QUESTION. WHAT IF SOMEONE ASSUMED IT WAS THE OTHER WAY AROUND?	
Find the result for class '10A' compared in terms of their performance in the topic 'Programming'	
UNCLEAR HOW TO RETURN TO MAIN MENU, ONCE I WAS HERE 3 SIMPLE DROP DOWN MENUS ALLOW ME TO SELECT "CLASSES" "10A BY TOPIC" "PROGRAMMING" TO SEE THE RESULT	
Create a graph of the performance over time of for the student with a candidate number '1111'	
AGAIN 3 SIMPLE DROP DOWNS ALLOW ME TO SELECT "INDIVIDUAL STUDENTS" "1111" AND PRESS A KEY TO GENERATE A GRAPH VERY EASY TO USE	

The first staff user worked through the first set of tasks in the questions scene easily and without issue. The user also comment that the first few results tasks were straightforward but expressed that he thought it was awkward that the user needed to restart the application to access a different scene.


Find results for all the different classes compared generally, graph this data
THIS WAS EASY TO DO, THE SYSTEM HAD A DROP DOWN OPTIONS AND A BUTTON PRESS, RESULTS WERE THEN DISPLAYED AS A GRAPH
Please Now play the use the application as you like for as long as you feel you need to properly evaluate its effectiveness. Please then answer the following questions.
Was the application generally simple or complicated to use
THE APPLICATION IS VERY EASY TO NAVIGATE AND SIMPLE TO UNDERSTAND. 9/10 USABILITY
Do you think the application would be useful in evaluating student performance
DEFINITELY, RESULTS ARE TRACKED AS STUDENTS HAVE THEIR OWN DATABASE ENTRY, RESULTS ARE ATTACHED TO THIS RECORD. THIS WILL BE INVALUABLE, ESPECIALLY SINCE DATA CAN BE GRAPHED AND SEEN
ATA GLANCE.
Could you see yourself using this application as a part of your revision
YES DEFINITELY, I WISH I HAD SOMETHING LIKE THIS TO USE AT GCSE.
Did you encounter of what you would perceive to be an error or bug with the application
NO ERRORS OR BUGS, ONE MIS CONCEPTION ON MY PART ON DIFFICULTY NUMBERING (LOWER OR HIGHER MORE DIFFICULT?)
Given unlimited time and unlimited and unlimited resources, could you suggest any improvements to the application
GRAPHICAL IMPROVEMENTS, MORE MODIFIERS, LEG ARMOUR, VAMBRACES, SPELLS, COINS FOR CORRECT ANSWERS, PURCHASE ABLE ITEMS, LEVELLING / XP SYSTEM. MORE MAPS AND BOSSES, DIFFERENT SCENARIOS BASED ON PLAYER CHOICE IE I.E. DIFFERENT STORY LINES, STORYLINE

The first staff user commented that the he thought the application would be something that he could use with his classes. The user commented that the application had no errors or bugs and gave said has “9/10 usability”. In terms of things the user would like to see improved he suggested that the difficulty value in the questions window wasn’t immediately clear. Other than that the only suggestions given were ones that involved adding more features to the game.

I will now login to and show you the student facing portion of the application. Please use this part of the application as much as you like before commenting on how useful you think it will be for students.
Do you think this tool will improve students engagement in revision
YES I THINK MAKING REVISION INTO A FUN COMPETITIVE GAME WILL INCREASE STUDENT ENGAGEMENT AND RETENTION OF INFORMATION. IT WILL BE LESS BORING AND MAKE REVISION MORE PALATABLE.
Do you think the way questions are asked will be effective or not effective in helping students revise
THE QUESTIONS ARE LAID OUT IN A SIMPLE QUESTION/ANSWER FORMAT, THIS WILL BE EFFECTIVE AS SHORT, SHARP QUESTIONS ARE EASIER TO TAKE THAN READING SCENARIOS OR BLOCKS OF TEXT
Could you see yourself using this application as a part of your revision plan
YES DEFINITELY, THIS WOULD MAKE A GREAT REVISION RESOURCE.

When presented with the student facing part of the application the first staff user said it will “Increase student engagement”. The user called the application a “Great Revision resource” and expressed that he thought the “Short, sharp” question and answer format simplified the application.

- Second Staff User:

Staff User Beta Testing Form	
Name: <u>MARIE GIRDLESTONE</u>	Date: <u>26/04/17</u>
I consent that the comments given are my own and are given honestly	
Signature: 	
Please attempt the following tasks, comment on how easy it was for you to achieve this task and how straight forward you thought the process was.	
Get to the staff portion of the application	
They could be a bit clearer	
Add a new question to the topic 'GCSE System's Architecture'	
Make it clear which topic number relates to the topic of choice	
Delete the Question "Convert the hexadecimal number 3E into decimal"	
Use same button for controls, eg. Delete button to delete	
Edit the question with a primary key '36', give it a new topic number, question and answer	
Crashed. Edited fine.	
Find the most difficult question in the topic 'A-Level System's architecture'	
Make it clear with regards to difficulty value	
Find the result for class '10A' compared in terms of their performance in the topic 'Programming'	
Make navigation easier - going back to menus Found easily	
Create a graph of the performance over time of for the student with a candidate number '1111'	
Make it clear that 1 = 100% on the axis.	

The second staff user accomplished all of the tasks but made comments on how these processes could be improved on her testing form. These comments largely suggested ways in which I could improve the games clarity to make it easier to understand to a new user. Unfortunately during this Beta test we encountered an error. The user wanted to add a question to the database that included an apostrophe. This resulted in a MySQL syntax error as the apostrophe is the symbol used in MySQL to open and close strings.

```
! MySQLException: You have an error in your SQL syntax;
MySQL.Data.MySqlClient.MySqlCommand.ExecuteReader ()
```

To rectify this issue I needed to 'escape' apostrophe characters before the system attempts to add them to the database. I could do this easily using the replace command.

```
databaseScript.updateQuestion (questionField.text.Replace("'", "'"),
    answerField.text.Replace("'", "'"), topicField.text, rows [focusRow].getID());
```

```
databaseScript.addQuestion (questionField.text.Replace("'", "'"),
    answerField.text.Replace("'", "'"), topicField.text);
populate ();
```

This meant that every apostrophe character would be prefaced by another apostrophe and wouldn't cause a syntax error. To make sure this change was working properly I repeated earlier tests with new test cases.

4.5b: Editing a Question REPEAT	
Reason For Test	
Valid Input Test	
Input Data	Expected Result
(6, Which type of error in code causes a program to fail to compile?,SYNTAX) Question = 'Which type of error in code causes it's compilation to fail?'	(6, Which type of error in code causes it's compilation to fail?SYNTAX)
I reloaded the database and repeated the test, attempting to add made the question details contain an apostrophe. With the changes implemented the application worked correctly and the input was allowed.	
10 Which type of error in code causes a program to fail to	SYNTAX 0.571
10 Which type of error in code causes a program to fail to...	SYNTAX 6
10 Which type of error in code causes it's compilation to fail?	SYNTAX 0.444
10 Which type of error in code causes it's compilation to fail?	SYNTAX 6

4.8a: Adding a Question Repeat	
Reason for Test	
Valid Input Test	
Input Data	Expected Output
Topic = I, Question = 'Joe's game's items', Answer = 'Joe's game's items'	New Question (PK, Joe's game's items, Joe's game's items,I)
This test also resulted in a question with apostrophe's being accepted.	
49 Joe's game's items	JOE'S GAME'S ITEMS 0
49 Joe's game's items	JOE'S GAME'S I... 1
Test Passed	

This issue highlights the importance of beta testing. The second staff user attempted to use the system in a way I hadn't thought of and led to a solvable issue with the program being exposed. It was imperative that I carry out this 'Black box' testing by putting user's unfamiliar with the program in a position to use it so these kinds of situations could arise.

With this change implemented Beta testing continued.


Find results for all the different classes compared generally, graph this data
G button missing from screen. label
Please Now play the use the application as you like for as long as you feel you need to properly evaluate its effectiveness. Please then answer the following questions.
Was the application generally simple or complicated to use
It was ok once you know the controls, but it's not intuitive for a beginner.
Do you think the application would be useful in evaluating student performance
Yes it would be very useful
Could you see yourself using this application as a part of your revision
Yes, it would be very handy for my classes and see what needs to be worked on.
Did you encounter of what you would perceive to be an error or bug with the application
System crashed once. Clearer labels for the controls.
Given unlimited time and unlimited and unlimited resources, could you suggest any improvements to the application
Comparisons limited to year groups. Breakdown of individual results - specific question results, would be useful - print outs?

The second staff user comment that they thought the staff portion of the application was would be 'Very handy' at exposing areas of weakness and that it would be useful revision tool. The user commented that the control scheme for the staff portion of the application wasn't particular intuitive but worked fine after she was used to it. The user suggested that, to improve the application information should be broken down in more ways and make the way to control the application

I will now login to and show you the student facing portion of the application. Please use this part of the application as much as you like before commenting on how useful you think it will be for students.
Do you think this tool will improve students engagement in revision
Yes, they would enjoy it, especially if they were into RPGs.
Do you think the way questions are asked will be effective or not effective in helping students revise
They would be effective, does it allow for typos?
Could you see yourself using this application as a part of your revision plan
For upper sets, it would appeal to them

The second staff user commented that they thought that the system would be useful for engaging student in revision and that they would use it in their upper sets. This user also made a comment about the fact that the system doesn't accommodate typos.

- Third Staff User:

Staff User Beta Testing Form	
Name: C McWilliam	Date: 26/4/17
I consent that the comments given are my own and are given honestly	
Signature: 	
Please attempt the following tasks, comment on how easy it was for you to achieve this task and how straight forward you thought the process was.	
Get to the staff portion of the application	
Add a new question to the topic 'GCSE System's Architecture'	
Topic Number rather than name.	
Delete the Question "Convert the hexadecimal number 3E into decimal"	
Straight forward	
Edit the question with a primary key '36', give it a new topic number, question and answer	
Easy to edit	
Find the most difficult question in the topic 'A-Level System's architecture'	
Easy to find, could be highlighted	
Find the result for class '10A' compared in terms of their performance in the topic 'Programming'	
No back button to leave question editor	
Create a graph of the performance over time of for the student with a candidate number '1111'	
Refreshes from class to student. → If scrolled down no No alert.	

The third staff users comments regarding the first set of tasks was that they were largely straightforward. The user was able to achieve all of the tasks set independently. However the user did comment that the some changes could be made to make the system easier to use such as adding a back button to return to the staff menu and resetting the outlined row to the start of the table when the data set is changed.

Find results for all the different classes compared generally, graph this data
The backwards from initial graph 'G' could need explaining
Please Now play the use the application as you like for as long as you feel you need to properly evaluate its effectiveness. Please then answer the following questions.
Was the application generally simple or complicated to use
Generally simple, missing some instructions for basic functionality.
Do you think the application would be useful in evaluating student performance
Definitely
Could you see yourself using this application as a part of your revision
Yes.
Did you encounter of what you would perceive to be an error or bug with the application
No
Given unlimited time and unlimited and unlimited resources, could you suggest any improvements to the application
a few more text instructions around some areas

The user commented that the application was 'Generally simple' to use they could 'Definitely' see themselves using it as a way to view student's results. Their suggested improvements revolved around adding more instructions to make it easier to use and automatically shifting back to the table view when the data set was changed to prevent looking at an old graph.

I will now login to and show you the student facing portion of the application. Please use this part of the application as much as you like before commenting on how useful you think it will be for students.
Do you think this tool will improve students engagement in revision
Definitely will with the Competition element
Do you think the way questions are asked will be effective or not effective in helping students revise
Yes, effective
Could you see yourself using this application as a part of your revision plan
Yes

When presented with the student facing portion of the application the third staff user answered that she that it would “definitely” improve student engagement and that she could see herself using the tool in her revision plans.

I believe Beta testing was generally successful. All the users were able to navigate the application and accomplish all of the set tasks. Moreover, every user said that they thought my application was effective; students said they thought it would become a useful tool in their revision and teachers said they thought the application presented them with a useful way to view their students’ performance that they could see themselves using in their class rooms. Besides one error that was easily rectified, all users found my application to be working fine and even the most critical comments on the way the system is controlled and interacted with said that they didn’t pose a problem one the user had some experience with the way the application worked.

Performing ‘Black Box’ testing and putting the application in the hands of people with no understanding of how it was meant to work was also useful in comprehending the limitations of my application. Things that were obvious to me, as the developer, weren’t as clear to users. Some users recommended that more on-screen instructions would be necessary to make the application immediately accessible to new users. In addition, many of the users suggested ways of making the application more streamlined by implementing features like adding a button to facilitate moving between the staff scenes. This information will be invaluable as I begin to evaluate my project.

Evaluation

With the project development now complete, the application tested for robustness and function and my work presented to real end users I have now reached the stage where I must retrospectively evaluate the success of my project. I will first compare the finished version of my project to the success criteria me and my client agreed on before finally presenting the finished solution to the client for him to comment on how successfully he thinks I have met the brief he laid out. I will then discuss the future of the application; how maintainable my solution will be as it is continued to be used and how I could potentially develop the solution further in the future.

Acceptance Testing

At this stage of development I know my application is working and robust and I have received feedback from some end users. I must now conduct 'Acceptance Testing' to evaluate how well my application meets the brief set out for it and the success criteria I defined with my client. I will then present the application to my client so he can comment on how satisfied he is with the project and how effective he thinks it will be when deployed in the school.

1. Criteria: My solution must ask students questions about Computer Science:
Evaluation: My application does indeed provided its student users with quizzes all themed around the topics of Computer Science. Students are presented with one question at a time and informed whether they are correct. I believe the application does this effectively by incorporating a questions list that changes to constantly ask students the questions they and their peers have struggled with most. The feedback from all of my users suggests that this will be a valuable tool in the process of revision. As tested previously my application is consistent in correctly identifying right answers and will refuse to accept an empty answer field as valid input.
Limitations: A limitation of the way that my application asks students questions is that answers given must match exactly with the answers in the database. This means questions where there are multiple answers can't be used and answers could be falsely evaluated as incorrect because of different phrasing. The users that experienced the problem commented that they thought this made the process too strict. However I believe this is an acceptable limitation of the project. Students displayed in the questionnaire that they thought multiple choice questions were easy and these kinds of questions are not used in exams so I feel justified in the question, answer system I chose to implement. Furthermore, accommodating multiple answers would require me adding further tables to the database which would further complicate the problem. Both groups of end users said they though the application would be very useful and aid in revision, so I am satisfied with the system implemented in the limited time frame.
2. Criteria: They Student facing part of my application must take the form of a game
Evaluation: The quizzes in the application are framed around a game that takes inspiration from the classic RPG's. I believe the application makes use of gamification effectively. The students I sent my questionnaire to largely said they believed framing revision around a game would make it more engaging and, informed by my research, I made use of bright colours and a fantasy art style to promote student enjoyment. The student end users I tested the application with all spoke of how the enjoyed playing the game which speaks to the game's success considering the application is primarily an educational tool. Moreover, the teachers all commented that they thought the gamified elements would improve their students' enjoyment and engagement with revision. However, I don't believe the game elements take away from the systems effectiveness for revision. The weapons and tops allow the user to encounter more questions before the games end and the score variable encourage repeated play. Thorough testing of the game has shown it will effectively operate through extended use and continue to generate new maps and quizzes infinitely if needed.

Limitations: A limitation of the game part of the application is the lack of progress as the user plays the game. The pool of items doesn't change and the enemies don't get and different or indeed, harder as the user makes their way through subsequent map generations. This could lead to a sense of stagnation and dwindling student engagement over repeated use. However, the end users that were given the application all spoke positively of the application after the limited time they spent with it and said it was more engaging than a quiz in a textbook or other conventional methods. Given the time constraints placed on the project I'm satisfied that it addresses the issue of student engagement to the extent it does.

3. Criteria: My solution must come with enough pre-existing questions for the solution to be usable

Evaluation: Through the necessary input needed for in development testing my application accumulated 43 different questions, where every topic has at least one question to it. This was sufficient enough for end users to interact with the system during beta testing so I am satisfied that including this number in implemented version of the application will be enough for students and staff to gain an understanding of how the application works before adding questions of their own.

4. Criteria: My solution must allow teachers to edit, add and remove questions

Evaluation: Emphasised in importance by both my client and potential end users my application does indeed allow teachers to edit the questions list as they see fit. The questions scene displays the relative difficulties of questions to inform teachers in their decisions and, through, thorough testing I know the scene effectively validates user input to maintain the function of the application. Staff users tasked with changing the questions list were all able to accomplish their set tasks on their own and found themselves unable to make and invalid changes that would compromise the database.

Limitations: The questions scene is partially limited by the way in which it handles assigning questions to different topics. The staff users that interacted with my application consistently said that having to know the topic number wasn't a very intuitive way of assigning questions. I believe this is a justifiable limitation because all my users were able to navigate the system after some initial trepidation but I would definitely look at this as key areas to improve in potential future development. A further limitation is the lack of security on the questions tab. There is no login required to access the staff portion of the application which puts more of a burden of staff to make sure their students aren't using the system improperly. This limitation is not severe because the staff portion of the application doesn't hold any secure or private data it just might manifest as a potential annoyance during use. This is a feature that I believe would be valuable to develop in the future.

5. Criteria: My solution must record students' performance

Evaluation: It is evident from the testing that has taken place and the prior discussion that the application records the results of every question ever asked in the system. Testing has shown that the record of student performance is consistently correct and durable for the large amount of data the application will have to deal with when it is implemented.

6. Criteria: My solution must store student' performance and questions in a central database.

Evaluation: Again, the fact that this criteria is met is clear from the testing and prior discussion. I believe my application makes effective use of a database by using it in conjunction with the Ubuntu server. This means that all the results and questions are accumulated in one place so changes to the questions set are immediately put in place on every instance of the application. Moreover, user results are accumulated in one place so teachers can have a whole class working on the application at once and be able to easily view their results grouped together. The testing that took place with end users showed that the system was capable of working with the large data set that would accumulate on the

database and the staff end users all commented that they thought the application would be a useful tool for comparing students results and tracking their progress.

Limitations: A limitation of the current system is that students and staff must be on a computer connected the school network to access the database and be able to use the application. This means, for staff, that they can't view student's results outside of work hours and, for students, they can only revise in this specific way during in computing lessons or free periods. This limits the usability of my application, however, I believe this is an acceptable limitation. My application is meant to only be a revision tool and by no means meant to be a complete substitute for all forms of revision. Restricting the times in which students can use the application isn't necessarily a bad thing as it encourages using a variety of tools. Furthermore, not keeping an actual store of the data as a part of the application prevents the file size of the application growing and means that question details that are inputted incorrectly and then rectified won't persist in versions of the application.

7. Criteria: My solution must allow teacher to view students' performance both numerically and in a bar chart

Evaluation: My application successfully displays student's results in both a numerical table format and in bar graphs. . I believe my application does this effectively by comparing results in the three ways most popular in the staff questionnaire, by class, by student and by topic. Testing has shown that the application will consistently calculate results correctly and plot results on graphs accurately. The staff end users that were presented with the project commented that they thought the application would. Be a useful tool in analysing students' results. Furthermore, testing using a large data set allows me to be confident in saying that the application will be capable of dealing with the large volume of answers is meant to encounter when applied in real life.

Limitations: Despite the effort put into it, I still believe the system of dropdowns is cumbersome and the experience of end users displays it's not particularly intuitive. However, for the large variety of different setting my application needs to facilitate I am satisfied with this result. The current system is fully functioning and end users were all able to find the desired data eventually. A further limitation is fact that, for very large sets of results, the bar graphs become unreadable without zooming in. This is an unfortunate result of the way that I decided to display the text on the graphs. If I had unlimited time for my project I would attempt to rectify this issue by redesigning the way that text is displayed. However, there are time constraints placed on the project and I am satisfied with the current system that is perfectly usable in all but extreme cases and appeared to have the potentially be very useful in the opinion of the end users presented with the project.

Usability

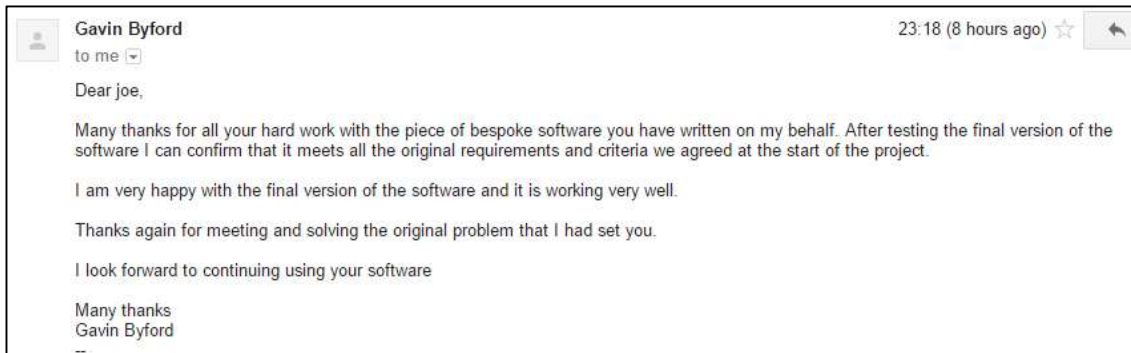
The average user of the application won't have a comprehensive working knowledge of how it works so it is important that I discuss the usability of my application. In basic terms my application has shown itself to be thoroughly usable. I set the Beta testing users a series of tasks to accomplish without my assistance and they were consistently met. Through testing I have shown that my application is durable in the face of invalid data input and improper use.

However, when considering the simplicity of my system I received some mixed comments. Students largely commented that the process of playing the game was simple but some teachers had suggestion for the way the staff portion of the application could work better. The need to know the different topic numbers to add questions and the lack of a way to transition between the Questions and Results scenes emerged as consistent issues as well as comments that some more informational text would be useful in understanding how the application works. These opinions have informed me that a perhaps the way I chose to organise information wasn't necessarily the most universal and

that investing in more presenting information to the user in a clearer and more simplistic way would be a good idea for future development. That being said, the comments given by my end users by no means suggested the application was unusable. The more critical opinions suggested that the application simply took some time to get used to and one staff end user said the application had a “9/10” usability despite these issues.

Presentation to Client

With the finished version of the application now available and shown to work on the school system, I showed the Revision Game system to Mr Byford to gain his comment on the finished product and for him to, if he was satisfied with what I had developed, to agree that the project was finished and the brief had been met. After a discussion about the project he send me the following email to articulate what we discussed.



As the screenshot below displays, Mr Byford was very happy with the solution I presented him with. He agreed with my evaluation that the success criteria had been met and that I presented him with an appropriate solution to the problem. My client expressed that he would continue to use my software to as a Revision tool.

Maintainability of the solution

My solution was designed from the group up with maintainability as a key aim. We made the decision to make a changing question set and with the knowledge in mind that exam boards and qualification specifications change. To that end teachers can edit the set of questions independently to keep the system up to date. By using a central database all these changes will be immediately distributed across all instances of the application meaning there is no reason a student should even have to answer an out of date question. Moreover, the tables and menus in the staff scenes are designed to accommodate for an infinite list of questions by being able not making the questions or results set fit inside a finite grid. Furthermore, as new students arrive in the school they need only to login to the system for the first time before they are fully represented in the application.

In addition, I have adopted good programming standards to ensure that the application can be altered if any changes need to be made in the future. I have made sure to use logical and informative names and because of the UML Class diagrams and Database Entity Relationship diagrams I created I have a concrete plan of how the different elements of the application all fit together and interact with each other.

One possible issue with the continued maintainability of the application is that there is no way to change the class of a student as they move through the school years or remove them entirely when they leave. This means that these records will have to be manually edited or the database will need to be refreshed at the end of every year. This will be an inconvenience as the application is used over repeated years but should only be an annual occurrence that won't occur when anybody is actually trying to use the system. A further potential maintainability issue is that the system doesn't

accommodate for any changes to the topics or class structure over subsequent years of use. In the case of the topics; they were aggregated from looking at the course specification from several different examination boards and purposefully made general. This may be a satisfactory solution to accommodate exam board changes but would cause problems if a new field topic areas in computing should arise. In the case of classes the application wouldn't mimic the real life class structure if it were to change, however, compromises like grouping real life classes together into one larger class on the application wouldn't its function or preventing from working, I would just mean teachers may need to filter the results the system gives them to look at the results of solely their own students.

Generally I think my application is largely maintainable. The way the system is designed allows staff members to change the contents of questions table of the database as the please and new students can be added easily. Considering the application on an annual structure may led to some maintainability issue as the values taken to be constant by the application change, but none of these issues would stop the application from actually functioning. If I, or somebody else, were to need to change the programming or want to develop the application further, the use of good programming standards coupled with thorough planning and diagram creating will ensure the code understandable and changed should be able to be made without interfering with the existing function.

Future Development

No application is ever perfect and I haven't created a system without the potential for improvement. Discussion with my end users has highlighted some ways I could improve the general function as well as some potential new features I could develop in the future.

- **General Function Changes:** There were some key points of contention in the way my application worked that I would be the first thing I would try to change if I were to revisit development. I would first add a way to move between the Scene and Results Tab without having to restart the system. I would also change the topic field in the Questions scene to the more elegant solution of a dropdown menu to remove the need to know student classes. In addition to this I would make smaller changes like simply making the outline return to the top of the table when results are changed in the Results scene. I would also redesign the way text is presented on the graphs to prevent it becoming too small to read. Informed by the experience of my Beta users I would hope to make the application more usable.
- **Help Screens:** A consistent suggestion throughout Beta testing was that my application could be made clearer or more simplistic by adding some more information to explain how works. In the future, I think it would be a valuable decision to add a help screen popup to each scene that explained its purpose and how to use it.
- **Game sound and animation:** Beta users suggested that adding sound or animation to the game scene would have a two-pronged effect. Firstly it would make the game more immersive and enjoyable but it would also allow me to make it clearer when something had happened and emphasises specific moments and actions to the user. I definitely believe adding some form of sound or animation to the game scene would improve its ability to communicate with the user.
- **Flexibility of answers:** A consistent issue with both student and staff end users is the way my application imposes the strict restriction that answers need to be exactly correct. In future development I would look for a way to re-design my database so it accommodated questions having multiple answers as well as a way to redesign the validation applied to answers entered to permit spelling mistakes.

Final Comments

Over the course of this project I am happy with what has been achieved. I have created a revision tool that my end users and client agree will be useful in combatting the problem of revision. My application improves student engagement by gamifying the process and provides a robust platform to analyse student performance and activity. As I have discussed previously there are limitations to my project that prevent it from being a complete replacement for all revision methods. However, my aim was to create a fun tool to aid both students and teachers tackle revision and contribute something to the overall process of study. The comments of my end users and the approval of my client lead me to believe that brief has been met.

Code Repository

For reference purposed here is the final working version of my code.

MainMenuController:

```
using UnityEngine;
using UnityEngine.SceneManagement;
using System.Collections;

public class MainMenuController : MonoBehaviour {

    private bool up;
    private bool down;

    //Used for initialisation
    void Start () {
        up = false;
        down = false;
        StartCoroutine (inputLoop ());
    }

    //Runs every frame
    void Update () {
        if (Input.GetKeyDown (KeyCode.UpArrow)) {
            up = true;
        }
        if (Input.GetKeyDown (KeyCode.DownArrow)) {
            down = true;
        }
        //Sets boolean values to true when keys are pressed
    }

    //Loops forever one initialised
    public IEnumerator inputLoop() {
        while (true) {
            //Loads scenes depending on user input
            if (up) {
                up = false;
                SceneManager.LoadScene ("Scene_StudentMode", LoadSceneMode.Single);
                break;
            } else if (down) {
                down = false;
                SceneManager.LoadScene ("Scene_StaffMode", LoadSceneMode.Single);
                break;
            }
            yield return null;
        }
    }
}
```

StudentModeController:

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
using UnityEngine.SceneManagement;

public class StudentModeController : MonoBehaviour {

    public Sprite playerSprite; // Passed to the Script: Player

    public Sprite[] mapSprites; // Passed to the Script: Game Control
    public Sprite emptySprite;
    public GameObject[] map;

    public Sprite[] enemySprites; // Passed to the Script: Quizzes

    public Sprite[] itemSprites; // Passed to the Script: Items
    public GameObject itemOutlinePrefab;

    public MySQLconnector databaseScript; // Passed to the Script: Database

    public InputField inputField; // Passed to the Script: UI
    public Text questionText;
    public Image quizBackground;
    public Slider playerSlider;
    public Image playerBackground;
    public Text playerHealthText;
    public Slider enemySlider;
    public Image enemyBackground;
    public Text enemyHealthText;
    public Image attackOverhead;
    public Image defenceOverhead;
    public Image healthOverhead;
    public Image scoreOverhead;
    public Text attackOverheadText;
    public Text defenceOverheadText;
    public Text healthOverheadText;
    public Text scoreOverheadText;
    public Image popupBackground;
    public Image popupItem;
    public Text popupDetails;
    public Text popupText;
    public Image endGameBackground;
    public Text endGameText;
    public Text endGameScoreText;
    public GameObject controls;

    private Player player; // Created by the script: Player
    private string playerID;
    private bool playerIsALevel;
    private int playerHealth;
    private int playerAttack;
    private int playerDefence;
    private int score;

    private bool enter; // Created by the script: Game control
    private bool space;
    private bool escape;
    private bool move;
    private int[] gridPosition;
    private bool[] directions;
    private Camera camera;
    private int mapQuizzes;
    private int mapItems;

    private Quiz[,] quizArray; // Created by the script: Quizzes
    private string[,] questions;
    private int enemyHealth;

    private Item[,] itemArray; // Created by the script: Items
    private Item[] inventory;
    private GameObject itemOutline;
    private int[] importantItems;
    private Item[] equipped;
```

```
//Main class for the game that the player, quizzes and items derive from
class Object {
    private Sprite objectSprite;
    private GameObject instance;
    private Rigidbody rigidbody;
    private SpriteRenderer spriteRenderer;
    private string objectName;
    public Object (Sprite sprite, string name) { //Constructor for new instance
        objectName = name;
        objectSprite = sprite;
        instance = new GameObject(objectName);
        instance.AddComponent<SpriteRenderer> ();
        instance.AddComponent<Rigidbody> ();
        rigidbody = instance.GetComponent<Rigidbody>();
        rigidbody.isKinematic = true;
        rigidbody.useGravity = false;
        spriteRenderer = instance.GetComponent<SpriteRenderer>();
        spriteRenderer.sprite = objectSprite;
    }
    public void move(float x, float y) { //Moves the position of the object
        rigidbody.position = new Vector3 (x, y, 0f);
    }
    public void toggleActive(bool state) { //Toggles the active state of the object
        instance.SetActive (state);
    }
    public void scaler(float scale) { //Scales the object
        instance.transform.localScale = new Vector3 (scale, scale, 1);
    }
    public void destroy() { //Destroys the object
        Destroy (instance);
    }
    public Sprite getSprite() { //Returns object sprite
        return objectSprite;
    }
    public string getName() { //Returns object name
        return objectName;
    }
}

//Player class inheriting from 'Object'
class Player : Object {
    private int health;
    private int attack;
    private int defence;
    private Sprite hatSprite;
    private Sprite topSprite;
    private Sprite weaponSprite;
    private SpriteRenderer hatRenderer;
    private SpriteRenderer topRenderer;
    private SpriteRenderer weaponRenderer;
    private GameObject equipped;
    private GameObject hat;
    private GameObject top;
    private GameObject weapon;
    public Player (Sprite sprite, string name) : base(sprite, name) { //Constructor for player
        health = 100;
        attack = 0;
        defence = 0;
        equipped = new GameObject("Equipped");
        hat = new GameObject("Hat");
        hat.AddComponent<SpriteRenderer> ();
        hatRenderer = hat.GetComponent<SpriteRenderer>();
        hatRenderer.sortingLayerName = "Equipped Items";
        hat.transform.parent = equipped.transform;
        hat.transform.position = new Vector3 (0, 0.11f, 0f);
        top = new GameObject("Top");
        top.AddComponent<SpriteRenderer> ();
        topRenderer = top.GetComponent<SpriteRenderer>();
        topRenderer.sortingLayerName = "Equipped Items";
        top.transform.parent = equipped.transform;
        top.transform.position = new Vector3 (0.015f, -0.13f, 0f);
        weapon = new GameObject("Weapon");
        weapon.AddComponent<SpriteRenderer> ();
    }
}
```

```
        weaponRenderer = weapon.GetComponent<SpriteRenderer>();
        weaponRenderer.sortingLayerName = "Equipped Items";
        weaponRenderer.sortingOrder = 1;
        weapon.transform.localScale = new Vector3 (0.6f,0.6f,1f);
        weapon.transform.parent = equipped.transform;
        weapon.transform.position = new Vector3 (0.19f, -0.1f, 0f);
    }
    public void move (float x, float y) { //Moved the player
        base.move (x,y);
        equipped.transform.position = new Vector3 (x,y,0f);
    }
    public void scaler (float scale) { //Scales the player
        base.scaler (scale);
        equipped.transform.localScale = new Vector3 (scale, scale, 1f);
    }
    //Setters and Getters for Health, Defence
    public void setHealth(int currentHealth) {
        health = currentHealth;
        if (health > 100) {
            health = 100;
        }
    }
    public int getHealth() {
        return health;
    }
    public void setAttack(int newAttack) {
        attack = newAttack;
    }
    public int getAttack() {
        return attack;
    }
    public void setDefence(int newDefence) {
        defence = newDefence;
    }
    public int getDefence() {
        return defence;
    }
    //Setters Hat, Top and Weapon
    public void setHat (Sprite sprite) {
        hatRenderer.sprite = sprite;
    }
    public void setTop (Sprite sprite) {
        topRenderer.sprite = sprite;
    }
    public void setWeapon (Sprite sprite) {
        weaponRenderer.sprite = sprite;
    }
}

//Quiz class inheriting from object
class Quiz : Object {
    private int topicNumber;
    public Quiz (int _topicNumber, Sprite sprite, string name) : base (sprite, name) { //Constructor for new quiz
        topicNumber = _topicNumber;
    }
    public int getTopicNumber () { //Returns topic number
        return topicNumber;
    }
}

//Item class inheriting from object
class Item : Object {
    private int effect;
    private int type;
    public Item (int _effect, int _type, Sprite sprite, string name) : base (sprite, name) { //Constructor for new item
        effect = _effect;
        type = _type;
    }
    public int getEffect () { //Returns effect
        return effect;
    }
    public int getType () { //Returns type
```

```
        return type;
    }
}

//Used for initialisation
void Start () {
    score = 0;
    // Initialise: Game control
    enter = false;
    space = false;
    escape = false;
    move = false;
    gridPosition = new int[2];
    directions = new bool[4];
    camera = this.GetComponent<Camera> ();
    toggleUI (false);
    togglePopupUI (false);
    // Initialise: Quizzes
    quizArray = new Quiz[5,5];
    // Initialise: Items
    itemArray = new Item[5,5];
    inventory = new Item[16];
    importantItems = new int[4];
    importantItems [0] = 0;
    equipped = new Item[3];
    itemOutline = Instantiate (itemOutlinePrefab, new Vector3 (8.25f, 3.5f, 0f), Quaternion.identity) as GameObject;

    StartCoroutine (gameLoop());
}

//Runs every frame
void Update () {
    //Handles player input
    if (Input.GetKeyDown (KeyCode.Return)) {
        enter = true;
    }
    if (Input.GetKeyDown (KeyCode.Space)) {
        space = true;
    }
    if (Input.GetKeyDown (KeyCode.Escape)) {
        escape = true;
    }
    if (Input.GetKeyDown (KeyCode.UpArrow)) {
        directions [0] = true;
    }
    if (Input.GetKeyDown (KeyCode.DownArrow)) {
        directions[1] = true;
    }
    if (Input.GetKeyDown (KeyCode.LeftArrow)) {
        directions[2] = true;
    }
    if (Input.GetKeyDown (KeyCode.RightArrow)) {
        directions[3] = true;
    }
    //Upates UI elements
    playerHealth = player.getHealth();
    playerAttack = player.getAttack ();
    playerDefence = player.getDefence ();
    playerHealthText.text = playerHealth.ToString() + " / 100";
    playerSlider.value = playerHealth;
    int count = 0;
    for (int i=0;i < inventory.Length; i++) {
        if (inventory [i] != null) {
            inventory [i].move (count%4 + 8.25f, 3.5f - count/4*1f);
            if (i == importantItems[0]) {
                itemOutline.transform.position = new Vector3 (count%4 + 8.25f, 3.5f - count/4*1f,
0f);
            }
            count++;
        }
    }
    for (int i = 0; i < equipped.Length; i++) {
        if (equipped [i] != null) {
```

```
        equipped [i].move (7f, 3.5f - i * 1.5f);
    }
}
attackOverheadText.text = "Attack: " + playerAttack.ToString();
defenceOverheadText.text = "Defence: " + playerDefence.ToString();
healthOverheadText.text = "Health: " + playerHealth.ToString();
scoreOverheadText.text = "Score: " + score;
}

//Main game Loop
public IEnumerator gameLoop () {
    player = new Player (playerSprite, "Player");
    gridPosition = new int[2] {0,0};
    directions = new bool[4] { false, false, false, false };
    populate ();
    yield return StartCoroutine (login());
    //Loops while the player is alive
    while (playerHealth > 0) {
        directions = new bool[4] { false, false, false, false };
        escape = false;
        move = false;
        //Waits for movement from the user
        while (move == false) {
            if (escape) {
                zoomIn (8.95f, 2, 2.5f);
                yield return StartCoroutine (runInventory());
                zoomOut ();
                escape = false;
            }
            if (directions[0] || directions[1] || directions[2] || directions[3]) {
                yield return StartCoroutine (mover ());
            }
            yield return null;
        }
        int x = gridPosition [0];
        int y = gridPosition [1];
        //Handles an item at gridposition
        if (itemArray [gridPosition [0],gridPosition [1]] != null) {
            //Handles a full inventory
            while (inventorySpace () == false) {
                togglePopupUI (true);
                popupItem.sprite = itemArray [gridPosition [0], gridPosition [1]].getSprite ();
                string name = itemArray [gridPosition [0], gridPosition [1]].getName ();
                string effect = itemArray [gridPosition [0], gridPosition [1]].getEffect ().ToStri
ng();

                popupDetails.text = "Name: " + name + "   Strength: " + effect;
                escape = false;
                space = false;
                while (escape == false && space == false) {
                    yield return null;
                }
                if (escape) {
                    zoomIn (8.95f, 2, 2.5f);
                    togglePopupUI (false);
                    yield return StartCoroutine (runInventory ());
                    zoomOut ();
                    togglePopupUI (true);
                } else if (space) {
                    itemArray [gridPosition [0], gridPosition [1]].destroy ();
                    itemArray [gridPosition [0], gridPosition [1]] = null;
                    break;
                }
                yield return null;
            }
            togglePopupUI (false);
            for (int i=0; i<inventory.Length; i++) {
                if (inventory[i] == null) {
                    inventory [i] = itemArray [x, y];
                    break;
                }
            }
            itemArray [x, y] = null;
            mapItems--;
        }
    }
}
```

```
//Handles a quiz at map position
if (quizArray [x,y] != null) {
    zoomIn (x, y, 0.375f);
    quizArray [x, y].scaler (0.5f);
    quizArray [x, y].move (x+0.2f,y+0.2f);
    player.scaler (0.5f);
    player.move (x-0.2f,y);
    int topicNumber = quizArray [x, y].getTopicNumber();
    yield return StartCoroutine (runQuiz (topicNumber));
    zoomOut ();
    quizArray [x, y].destroy ();
    quizArray [x, y] = null;
    player.scaler (1f);
    player.move (x, y);
    mapQuizzes--;
}
if (mapQuizzes == 0 && mapItems == 0) {
    populate ();
}
}
endGame ();
yield return null;
}

//Ends the game
public void endGame () {
    endGameBackground.gameObject.SetActive (true);
    endGameText.gameObject.SetActive (true);
    endGameScoreText.gameObject.SetActive (true);
    endGameScoreText.text = "Score: " + score;
}

// Facilitates Logging in
public IEnumerator login () {
    questionText.gameObject.SetActive (true);
    inputField.gameObject.SetActive (true);
    quizBackground.gameObject.SetActive (true);
    questionText.text = "Please enter your candidate number: ";
    inputField.characterLimit = 4;
    inputField.characterValidation = InputField.CharacterValidation.Integer;
    while (inputField.text.Length != 4) {
        enter = false;
        while (enter == false) {
            yield return null;
        }
        yield return null;
    }
}
playerID = inputField.text;
if (databaseScript.existCheck (playerID, "student") == false) {
    questionText.text = "You are a new user, what class are you in?: ";
    inputField.characterLimit = 3;
    inputField.characterValidation = InputField.CharacterValidation.Alphanumeric;
    inputField.text = "";
    string playerClass = "";
    while (true) {
        while (inputField.text.Length != 3) {
            enter = false;
            while (enter == false) {
                yield return null;
            }
        }
        playerClass = inputField.text.ToUpper();
    }
    if (databaseScript.existCheck(playerClass, "class")) {
        break;
    }
    yield return null;
}
questionText.text = "What is your first name?: ";
inputField.characterLimit = 0;
inputField.text = "";
enter = false;
while (enter == false) {
    yield return null;
}
}
```



```
        string playerFirstName = inputField.text;
        playerFirstName.ToUpper ();
        questionText.text = "What is your last name?: ";
        inputField.text = "";
        enter = false;
        while (enter == false) {
            yield return null;
        }
        string playerLastName = inputField.text;
        playerLastName.ToUpper ();
        databaseScript.newStudent (playerID, playerFirstName, playerLastName, playerClass);
    }
    playerIsAlevel = databaseScript.isAlevel(playerID);
    questionText.gameObject.SetActive (false);
    inputField.gameObject.SetActive (false);
    quizBackground.gameObject.SetActive (false);
    inputField.characterValidation = InputField.CharacterValidation.None;
    inputField.characterLimit = 0;
    inputField.text = string.Empty;
}

//Populates the game map
public void populate () {
    int randomNumber;
    int x = gridPosition [0];
    int y = gridPosition [1];
    SpriteRenderer sRenderer;
    mapQuizzes = 0;
    mapItems = 0;
    for (int i=0;i<5;i++) {
        for (int j=0;j<5;j++) {
            randomNumber = Random.Range (1,13);
            if (randomNumber > 6) {
                int topic = randomNumber - 6;
                randomNumber = Random.Range (0, enemySprites.Length);
                quizArray [i, j] = new Quiz (topic, enemySprites [randomNumber], "Enemy");
                Debug.Log ("Topic :" + topic);
                quizArray [i, j].scaler (0.6f);
                quizArray [i, j].move (i, j + 0.2f);
                randomNumber = Random.Range (0, mapSprites.Length);
                sRenderer = map [i + j * 5].GetComponent <SpriteRenderer> ();
                sRenderer.sprite = mapSprites [randomNumber];
                mapQuizzes++;
            } else {
                sRenderer = map [i + j * 5].GetComponent <SpriteRenderer> ();
                sRenderer.sprite = emptySprite;
            }
            int numberOfItems = databaseScript.findNumberOfItems ();
            randomNumber = Random.Range (1,numberOfItems*2);
            if (randomNumber > numberOfItems) {
                string[] newItem = databaseScript.findItem (randomNumber - numberOfItems);
                Sprite itemSprite = null;
                for (int k=0; k<itemSprites.Length; k++) {
                    if (itemSprites [k].name == newItem [0] + "_0") {
                        itemSprite = itemSprites [k];
                    }
                }
                itemArray [i, j] = new Item (int.Parse(newItem[1]), int.Parse(newItem[2]), itemSpr
ite, newItem[0]);
                itemArray [i, j].scaler (0.6f);
                itemArray [i, j].move (i, j-0.2f);
                mapItems++;
            }
        }
    }
}
if (quizArray [x, y] != null) {
    quizArray [x, y].destroy ();
    quizArray [x, y] = null;
    mapQuizzes--;
}
if (itemArray [x, y] != null) {
    itemArray [x, y].destroy ();
    itemArray [x, y] = null;
    mapItems--;
}
```

```
    }
    randomNumber = Random.Range (0, mapSprites.Length);
    sRenderer = map[x + y * 5].GetComponent<SpriteRenderer>();
    sRenderer.sprite = mapSprites [randomNumber];
}

//Handles player movement
public IEnumerator mover() {
    int x = gridPosition [0];
    int y = gridPosition [1];
    while (move == false) {
        if (directions [0] == true && gridPosition [1] < 4) {
            gridPosition [0] = x;
            gridPosition [1] = y + 1;
            move = true;
        } else if (directions [1] == true && gridPosition [1] > 0) {
            gridPosition [0] = x;
            gridPosition [1] = y - 1;
            move = true;
        } else if (directions [2] == true && gridPosition [0] > 0) {
            gridPosition [0] = x - 1;
            gridPosition [1] = y;
            move = true;
        } else if (directions [3] == true && gridPosition [0] < 4) {
            gridPosition [0] = x + 1;
            gridPosition [1] = y;
            move = true;
        }
        yield return null;
    }
    player.move(gridPosition[0],gridPosition[1]);
    yield return null;
}

//Toggles UI between the game map and quizzes
public void toggleUI (bool state) {
    inputField.gameObject.SetActive (state);
    questionText.gameObject.SetActive (state);
    quizBackground.gameObject.SetActive (state);
    playerBackground.gameObject.SetActive (state);
    playerSilder.gameObject.SetActive (state);
    playerHealthText.gameObject.SetActive (state);
    enemyBackground.gameObject.SetActive (state);
    enemySilder.gameObject.SetActive (state);
    enemyHealthText.gameObject.SetActive (state);
    attackOverhead.gameObject.SetActive (!state);
    defenceOverhead.gameObject.SetActive (!state);
    healthOverhead.gameObject.SetActive (!state);
    attackOverheadText.gameObject.SetActive (!state);
    defenceOverheadText.gameObject.SetActive (!state);
    healthOverheadText.gameObject.SetActive (!state);
}

//Toggles the item popup UI
public void togglePopupUI (bool state) {
    popupBackground.gameObject.SetActive (state);
    popupItem.gameObject.SetActive (state);
    popupDetails.gameObject.SetActive (state);
    popupText.gameObject.SetActive (state);
}

//Zooms in the Camera
public void zoomIn(float x, float y, float size) {
    camera.orthographicSize = size;
    camera.transform.position = new Vector3 (x, y, -10);
}

//Zooms out the camera to its original position
public void zoomOut() {
    camera.orthographicSize = 2.7f;
    camera.transform.position = new Vector3 (2,2.23f,-10);
}

// Runs Quizzes
```

```
public IEnumerator runQuiz(int topicNumber) {
    enemyHealth = 100;
    if (playerIsALevel == true) {
        questions = databaseScript.findTopicQuestions (topicNumber + 6);
        Debug.Log ("TopicNo: " + (topicNumber + 6));
    } else {
        questions = databaseScript.findTopicQuestions (topicNumber);
        Debug.Log ("TopicNo: " + topicNumber);
    }
    int noQuestions = questions.Length / 4;
    int count = 0;
    toggleUI (true);
    enemySlider.value = enemyHealth;
    enemyHealthText.text = enemyHealth.ToString() + " / 100";
    int index = 0;
    float difficulty;
    bool[] askedQuestions = new bool[noQuestions];
    while (playerHealth > 0 && count < noQuestions && enemyHealth > 0) {
        difficulty = 1;
        for (int i = 0; i < noQuestions; i++) {
            if (float.Parse(questions[i, 3]) <= difficulty && askedQuestions[i] != true) {
                index = i;
                difficulty = float.Parse(questions [i, 3]);
            }
        }
        askedQuestions [index] = true;
        questionText.text = questions [index, 1];
        while (inputField.text.Replace(" ", "").Length == 0) {
            enter = false;
            while (enter == false) {
                yield return null;
            }
            yield return null;
        }
        if (inputField.text.ToUpper() == questions [index , 2]) {
            //Debug.Log(inputField.text.ToUpper() + " " + questions[index,2]);
            questionText.text = "Correct";
            enemyHealth = enemyHealth - 10 - playerAttack;
            enemySlider.value = enemyHealth;
            enemyHealthText.text = enemyHealth.ToString() + " / 100";
            databaseScript.newAnswerInstance (playerID, true, questions[index,0]);
            score++;
        } else {
            questionText.text = "Incorrect";
            player.setHealth (playerHealth + playerDefence - 10);
            databaseScript.newAnswerInstance (playerID, false, questions[index,0]);
        }
        inputField.text = string.Empty;
        enter = false;
        while (enter == false) {
            yield return null;
        }
        count++;
        yield return null;
    }
    toggleUI (false);
    yield return null;
}

//Runs the inventory
public IEnumerator runInventory () {
    escape = false;
    enter = false;
    space = false;
    directions = new bool[4] { false, false, false, false };
    while (escape == false) {
        if (directions[0] || directions[1] || directions[2] || directions[3]) {
            move = false;
            yield return StartCoroutine (inventoryMover ());
            move = false;
            directions = new bool[4] { false, false, false, false };
        }
        if (space) {
            if (inventory[importantItems [0]] != null) {
```

```
        inventory [importantItems [0]].destroy ();
        inventory [importantItems [0]] = null;
        importantItems [0] = inventoryFoccusAdjust (importantItems [0]);
    }
    space = false;
}
if (enter) {
    if (inventory[importantItems[0]] != null) {
        if (inventory [importantItems [0]].getType () == 4) {
            player.setHealth (playerHealth + inventory [importantItems [0]].getEffect ());
            inventory [importantItems [0]].destroy ();
            inventory [importantItems [0]] = null;
        } else if (inventory [importantItems [0]].getType () == 1) {
            Item temp = equipped [0];
            equipped [0] = inventory [importantItems [0]];
            inventory[importantItems [0]] = temp;
            player.setHat (equipped[0].getSprite());
        } else if (inventory [importantItems [0]].getType () == 2) {
            Item temp = equipped [1];
            equipped [1] = inventory [importantItems [0]];
            player.setDefence (inventory[importantItems[0]].getEffect());
            inventory [importantItems [0]] = temp;
            player.setTop (equipped[1].getSprite());
        } else if (inventory [importantItems [0]].getType () == 3) {
            Item temp = equipped [2];
            equipped [2] = inventory [importantItems [0]];
            player.setAttack (inventory[importantItems[0]].getEffect());
            inventory [importantItems [0]] = temp;
            player.setWeapon (equipped [2].getSprite ());
        }
        if (importantItems != null) {
            importantItems [0] = inventoryFoccusAdjust (importantItems [0]);
        }
    }
    enter = false;
}
yield return null;
}
}

//Adjusted the item focussed on in the inventroy
public int inventoryFoccusAdjust (int current) {
    for (int i = current; i < inventory.Length; i++) {
        if (inventory[i] != null && i != current) {
            return i;
        }
    }
    for (int i = current; i > 0; i--) {
        if (inventory [i] != null && i != current) {
            return i;
        }
    }
    return 0;
}

//Checks if there is space in the inventroy
public bool inventorySpace() {
    bool space = false;
    for (int i = 0; i < inventory.Length; i++) {
        if (inventory [i] == null) {
            space = true;
        }
    }
    return space;
}

//Handles player movement in the inventory
public IEnumerator inventoryMover() {
    if (directions [0] == true) {
        int count = 0;
        for (int i = importantItems [0]; i >= 0; i--) {
            if (inventory [i] != null) {
                count++;
                if (count == 5) {
```

```
                importantItems [0] = i;
            }
        }
    }
} else if (directions [1] == true) {
    int count = 0;
    for (int i = importantItems [0]; i < inventory.Length; i++) {
        if (inventory [i] != null) {
            count++;
            if (count == 5) {
                importantItems [0] = i;
            }
        }
    }
} else if (directions [2] == true && importantItems [0] > 0) {
    int count = 0;
    for (int i = importantItems [0]; i >= 0; i--) {
        if (inventory [i] != null) {
            count++;
            if (count == 2) {
                importantItems [0] = i;
            }
        }
    }
} else if (directions [3] == true) {
    int count = 0;
    for (int i = importantItems [0]; i < inventory.Length; i++) {
        if (inventory [i] != null) {
            count++;
            if (count == 2) {
                importantItems [0] = i;
            }
        }
    }
}
}
yield return null;
}
}
```

Staff Mode Controller

```
using UnityEngine;
using UnityEngine.SceneManagement;
using System.Collections;

public class StaffModeController : MonoBehaviour {

    private bool up;
    private bool down;

    //Used for initialisation
    void Start () {
        up = false;
        down = false;
        StartCoroutine (inputLoop ());
    }

    //Runs every frame
    void Update () {
        if (Input.GetKeyDown (KeyCode.UpArrow)) {
            up = true;
        }
        if (Input.GetKeyDown (KeyCode.DownArrow)) {
            down = true;
        }
    }

    //Runs continually when the scene is initialised
    public IEnumerator inputLoop() {
        while (true) {
            //Loads scenes depending on user input
            if (up) {
                up = false;
                SceneManager.LoadScene ("Scene_Questions", LoadSceneMode.Single);
            }
        }
    }
}
```

```
        break;
    } else if (down) {
        down = false;
        SceneManager.LoadScene ("Scene_Results", LoadSceneMode.Single);
        break;
    }
    yield return null;
}
}
```

QuestionsController

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;

public class QuestionsController : MonoBehaviour {

    public MySQLconnector databaseScript; //Passed to Script: General
    public GameObject[] questionsDisplay;
    public Text topicNumberText;
    public Image questionOutline;

    public Image inputFormBackground; //Passed to the script: Additional UI
    public InputField topicField;
    public Text topicText;
    public InputField questionField;
    public Text questionText;
    public InputField answerField;
    public Text answerText;
    public Text confirmText;
    public Text abandonText;
    public Text errorMessageText;
    public Image AYSBackground;
    public Text AYSText;
    public Text AYSQuestion;

    private bool up; //Private: Game Control
    private bool down;
    private bool left;
    private bool right;
    private bool enter;
    private bool escape;
    private bool a;
    private Question[] questions;
    private int topicNumber;

    private Row[] rows; //Private: Display
    private int focusRow;
    private int displayStartIndex;

    //Class used to hold question information
    class Question {
        private string questionID;
        private string questionText;
        private string questionAnswer;
        private string questionDifficulty;
        public Question (string ID, string text, string answer, string difficulty) { //Constructor
            questionID = ID;
            questionText = text;
            questionAnswer = answer;
            questionDifficulty = difficulty;
        }
        //Return methods for all attributtes
        public string getID() {
            return questionID;
        }
        public string getText() {
            return questionText;
        }
        public string getAnswer() {
            return questionAnswer;
        }
    }
}
```

```
        public string getDifficulty() {
            return questionDifficulty;
        }
    }

    //Class used for rows on display
    class Row {
        private Text rowID;
        private Text rowQuestion;
        private Text rowAnswer;
        private Text rowDifficulty;
        public Row (GameObject rowParent) { //Constructor
            rowID = rowParent.transform.Find ("ID Text").GetComponent<Text> ();
            rowQuestion = rowParent.transform.Find ("Question Text").GetComponent<Text> ();
            rowAnswer = rowParent.transform.Find ("Answer Text").GetComponent<Text> ();
            rowDifficulty = rowParent.transform.Find ("Difficulty").GetComponent<Text> ();
        }
        // Getters and setters for attributtes
        public void setID (string ID) {
            rowID.text = ID;
        }
        public void setQuestion (string question) {
            rowQuestion.text = question;
        }
        public void setAnswer (string answer) {
            rowAnswer.text = answer;
        }
        public void setDifficulty (string difficulty) {
            rowDifficulty.text = difficulty;
        }
        public string getID () {
            return rowID.text;
        }
        public string getQuestion () {
            return rowQuestion.text;
        }
        public string getAnswer () {
            return rowAnswer.text;
        }
    }

    //Used for initialisation
    void Start () {
        up = false;
        down = false;
        left = false;
        right = false;
        enter = false;
        escape = false;
        a = false;
        topicNumber = 1;
        displayStartIndex = 0;
        rows = new Row[questionsDisplay.Length];
        focusRow = 0;
        for (int i=0; i<questionsDisplay.Length; i++) {
            rows [i] = new Row (questionsDisplay[i]);
        }
        toggleInputUI (false);
        toggleAYSUI (false);
        StartCoroutine (runLoop());
    }

    //Runs every frame
    void Update () {
        // Toggles user input variables
        if (Input.GetKeyDown (KeyCode.UpArrow)) {
            up = true;
            Debug.Log ("Up");
        }
        if (Input.GetKeyDown (KeyCode.DownArrow)) {
            down = true;
            Debug.Log ("Down");
        }
        if (Input.GetKeyDown (KeyCode.LeftArrow)) {
```

```
        left = true;
        Debug.Log ("Left");
    }
    if (Input.GetKeyDown (KeyCode.RightArrow)) {
        right = true;
        Debug.Log ("Right");
    }
    if (Input.GetKeyDown (KeyCode.Return)) {
        enter = true;
    }
    if (Input.GetKeyDown (KeyCode.Escape)) {
        escape = true;
    }
    if (Input.GetKeyDown (KeyCode.A)) {
        a = true;
    }
    topicNumberText.text = "Topic Name: " + databaseScript.findTopicName(topicNumber);
    questionOutline.rectTransform.localPosition = new Vector3 (0f, 170f-focusRow*45, 0f);
}

// Populates the rows on the screen with questions
public void populate () {
    string[,] topicQuestions = databaseScript.findTopicQuestions (topicNumber);
    questions = new Question[topicQuestions.Length / 4];
    for (int i=0; i<questions.Length; i++) {
        questions [i] = new Question (topicQuestions [i, 0], topicQuestions [i, 1],
            topicQuestions [i, 2], topicQuestions [i, 3]);
    }
    for (int i=0; i<rows.Length; i++) {
        if (i + displayStartIndex < questions.Length) {
            rows [i].setID (questions [i+displayStartIndex].getID ());
            rows [i].setQuestion (questions [i+displayStartIndex].getText ());
            rows [i].setAnswer (questions [i+displayStartIndex].getAnswer ());
            rows [i].setDifficulty (questions [i+displayStartIndex].getDifficulty ());
        } else {
            rows [i].setID (string.Empty);
            rows [i].setQuestion (string.Empty);
            rows [i].setAnswer (string.Empty);
            rows [i].setDifficulty (string.Empty);
        }
    }
}

// Main run Loops that run forever
public IEnumerator runLoop () {
    populate ();
    while (true) {
        // Handles navigation
        if (left == true || right == true) {
            if (left == true && topicNumber > 1) {
                topicNumber--;
                displayStartIndex = 0;
                focusRow = 0;
                populate ();
            }
            if (right == true && topicNumber < 12) {
                topicNumber++;
                displayStartIndex = 0;
                focusRow = 0;
                populate ();
            }
            left = false;
            right = false;
            Debug.Log ("TopicNo: " + topicNumber);
        }
        if (up == true) {
            if (focusRow > 0) {
                focusRow--;
            } else if (displayStartIndex > 0) {
                displayStartIndex--;
                populate ();
            }
        }
        up = false;
    }
}
```



```
        if (down == true) {
            if (focusRow < rows.Length - 1 && focusRow < questions.Length - displayStartIndex - 1)
            {
                focusRow++;
            } else if (displayStartIndex < questions.Length - rows.Length) {
                displayStartIndex++;
                populate ();
            }
            down = false;
        }
        // Handles deletion
        if (escape == true) {
            if (rows[focusRow].getID() != "") {
                yield return StartCoroutine(runAYS ());
                populate ();
            }
            escape = false;
        }
        // Handles editing questions
        if (enter == true) {
            if (rows [focusRow].getID () != "") {
                toggleInputUI (true);
                topicField.text = topicNumber.ToString();
                questionField.text = rows [focusRow].getQuestion ();
                answerField.text = rows [focusRow].getAnswer ();
                errorMessageText.text = "";
                enter = false;
                escape = false;
                while (true) {
                    answerField.text = answerField.text.ToUpper ();
                    if (enter) {
                        if (topicField.text.Length == 0) {
                            errorMessageText.text = "Please enter a topic";
                        } else if (int.Parse (topicField.text) < 1 || int.Parse (topicField.text)
> 12) {
                            errorMessageText.text = "Invalid Topic Number: Enter a topic between 1
and 12";
                        } else if (questionField.text.Replace (" ", "").Length == 0) {
                            errorMessageText.text = "No question entered";
                        } else if (answerField.text.Replace (" ", "").Length == 0) {
                            errorMessageText.text = "No answer entered";
                        } else {
                            databaseScript.updateQuestion (questionField.text.Replace(" ", ""),
                                answerField.text.Replace(" ", ""), topicField.text, rows [focusRo
w].getID());
                        }
                    }
                    populate ();
                    break;
                }
                enter = false;
            }
            if (escape) {
                break;
            }
            yield return null;
        }
        toggleInputUI (false);
    }
    enter = false;
    escape = false;
    a = false;
}
// Handles adding a new question
if (a) {
    toggleInputUI (true);
    topicField.text = string.Empty;
    questionField.text = string.Empty;
    answerField.text = string.Empty;
    enter = false;
    escape = false;
    errorMessageText.text = "";
    while (true) {
        answerField.text = answerField.text.ToUpper ();
        if (enter) {
            if (topicField.text.Length == 0) {
```

```
        errorMessageText.text = "Please enter a topic";
    } else if (int.Parse (topicField.text) < 1 || int.Parse (topicField.text) > 12
) {
        errorMessageText.text = "Invalid Topic Number: Enter a topic between 1 and
12";
    } else if (questionField.text.Replace (" ", "").Length == 0) {
        errorMessageText.text = "No question entered";
    } else if (answerField.text.Replace (" ", "").Length == 0) {
        errorMessageText.text = "No answer entered";
    } else {
        databaseScript.addQuestion (questionField.text.Replace("'", ""),
        answerField.text.Replace("'", ""), topicField.text);
        populate ();
        break;
    }
    enter = false;
}
if (escape) {
    break;
}
yield return null;
}
toggleInputUI (false);
enter = false;
escape = false;
a = false;
}
yield return null;
}
yield return null;
}

// Runs the are you sure popup
public IEnumerator runAYS () {
    toggleAYSUI (true);
    enter = false;
    escape = false;
    AYSQuestion.text = rows [focusRow].getQuestion ();
    while (true) {
        up = false;
        down = false;
        left = false;
        right = false;
        if (enter == true) {
            databaseScript.deleteQuestion (rows [focusRow].getID());
            break;
        }
        if (escape == true) {
            break;
        }
        yield return null;
    }
    enter = false;
    escape = false;
    toggleAYSUI (false);
    yield return null;
}

// Toggle the input field
public void toggleInputUI (bool state) {
    inputFormBackground.gameObject.SetActive(state);
    topicField.gameObject.SetActive(state);
    topicText.gameObject.SetActive(state);
    topicField.text = string.Empty;
    questionField.gameObject.SetActive(state);
    questionText.gameObject.SetActive(state);
    questionField.text = string.Empty;
    answerField.gameObject.SetActive(state);
    answerText.gameObject.SetActive(state);
    answerField.text = string.Empty;
    confirmText.gameObject.SetActive(state);
    abandonText.gameObject.SetActive(state);
    errorMessageText.gameObject.SetActive (state);
}
}
```

```
// Toggles the Are you sure UI elements
public void toggleAYSUI (bool state) {
    AYSBackground.gameObject.SetActive (state);
    AYSText.gameObject.SetActive (state);
    AYSQuestion.gameObject.SetActive (state);
}
}
```

ResultsController

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
using System.Collections.Generic;

public class ResultsController : MonoBehaviour {

    public MySQLconnector databaseScript; //Passed to the script: General
    public GameObject[] display;
    public Image dataOutline;
    public Dropdown dropdownA;
    public Dropdown dropdownB;
    public Dropdown dropdownC;
    public GameObject barPrefab;
    public GameObject canvas;
    public GameObject graph;
    public GameObject heading;
    public Material[] materials;

    private bool up; //Private: General
    private bool down;
    private bool g;
    private DataItem[] dataItems;
    private string[,] results;
    private string[] classes;
    private string[] students;

    private Row[] rows; // Private: Display
    private Bar[] bars;
    private int focusRow;
    private int displayStartIndex;
    private List<string> filterOptions;

    // Class used to store data items from
    class DataItem {
        private string itemName;
        private string itemData;
        public DataItem (string name, string data) { //Class constructor
            itemName = name;
            itemData = data;
        }
        // Getters for attributes
        public string getName() {
            return itemName;
        }
        public string getData() {
            return itemData;
        }
    }

    // Class used to represent the rows on the display
    class Row {
        private Text rowName;
        private Text rowData;
        private GameObject rowParent;
        public Row (GameObject parent) { // Class constructor
            rowName = parent.transform.Find ("Name Text").GetComponent<Text> ();
            rowData = parent.transform.Find ("Data Text").GetComponent<Text> ();
            rowParent = parent;
        }
        // Getters and Setters for rowName and rowData
        public void setName (string name) {
            rowName.text = name;
        }
    }
}
```

```
    }
    public void setData (string data) {
        rowData.text = data;
    }
    public string getID () {
        return rowName.text;
    }
    public string getQuestion () {
        return rowData.text;
    }
    // Toggles the active state of the row
    public void toggleActive(bool state) {
        rowParent.gameObject.SetActive (state);
    }
}

// Class representing the bars on the graph
class Bar {
    private Text barText;
    private Image barImage;
    private RectTransform textTransform;
    private RectTransform imageTransform;
    private float barHeight;
    private GameObject instance;
    //Class constructor
    public Bar (GameObject prefab, string text, float height, GameObject parent, Material material
) {
        instance = Instantiate(prefab) as GameObject;
        barText = instance.transform.Find ("Text").GetComponent<Text> ();
        textTransform = instance.transform.Find ("Text").GetComponent<RectTransform> ();
        barImage = instance.transform.Find ("Image").GetComponent<Image> ();
        barImage.material = material;
        imageTransform = instance.transform.Find ("Image").GetComponent<RectTransform> ();
        barHeight = height;
        barText.text = text;
        instance.transform.parent = parent.transform;
    }
    public void move(float x, float y) { // Moves the bar
        textTransform.localPosition = new Vector3 (x, y, 0f);
        imageTransform.localPosition = new Vector3 (x, y, 0f);
    }
    public void resize(float width) { // Resizes the bar
        textTransform.sizeDelta = new Vector2 (510f, width);
        imageTransform.sizeDelta = new Vector2 (barHeight * 510f, width);
    }
    public void destroy() { // Destroys the bar
        Destroy (instance);
    }
    public void toggleActive(bool state) { // Toggles if the bar is active
        instance.gameObject.SetActive (state);
    }
}

// Used for initialisation
void Start () {
    up = false;
    down = false;
    g = false;
    displayStartIndex = 0;
    rows = new Row[display.Length];
    focusRow = 0;
    for (int i=0; i<display.Length; i++) {
        rows [i] = new Row (display[i]);
    }
    bars = new Bar[0];
    dataItems = new DataItem[0];
    results = databaseScript.gatherResults();
    dropdownA.onValueChanged.AddListener(delegate {DDAChange ();});
    dropdownB.onValueChanged.AddListener(delegate {ddbChange ();});
    dropdownC.onValueChanged.AddListener(delegate {DDCChange ();});
    dropdownB.gameObject.SetActive (false);
    dropdownC.gameObject.SetActive (false);
    students = databaseScript.findAll ("student");
```

```
        classes = databaseScript.findAll ("class");
        StartCoroutine (RunLoop());
    }

    //Runs on each frame
    void Update () {
        if (Input.GetKeyDown (KeyCode.UpArrow)) {
            up = true;
        }
        if (Input.GetKeyDown (KeyCode.DownArrow)) {
            down = true;
        }
        if (Input.GetKeyDown (KeyCode.G)) {
            g = true;
        }
        dataOutline.rectTransform.localPosition = new Vector3 (0f, 170f-focusRow*45, 0f);
    }

    // Populates the rows with result information
    public void rowPopulate () {
        for (int i=0; i<rows.Length; i++) {
            if (i + displayStartIndex < dataItems.Length) {
                rows [i].setName (dataItems [i+displayStartIndex].getName ());
                rows [i].setData (dataItems [i+displayStartIndex].getData ());
            } else {
                rows [i].setName (string.Empty);
                rows [i].setData (string.Empty);
            }
        }
    }

    // Populates the graph with bars
    public void graphPopulate () {
        for (int i=0; i < bars.Length; i++) {
            bars [i].destroy ();
        }
        bars = new Bar[dataItems.Length];
        for (int i=0; i<dataItems.Length; i++) {
            bars[i] = new Bar (barPrefab,dataItems[i].getName() + ": " +
                dataItems[i].getData(),float.Parse(dataItems[i].getData()),
                canvas, materials[i % materials.Length]);
            float width = 677f / dataItems.Length;
            bars[i].resize (width);
            bars [i].move (15f,-80.5f-i*width);
            bars [i].toggleActive (false);
        }
    }

    // Reduces the set of results based on a specific value
    public string[,] reduceDataSet (int axis, string value, string[,] inputData) {
        int count = 0;
        for (int i = 0; i < inputData.Length / 6; i++) {
            if (inputData [i, axis] == value) {
                count++;
            }
        }
        string[,] usedResults = new string[count,6];
        count = 0;
        for (int i = 0; i < inputData.Length / 6; i++) {
            if (inputData [i, axis] == value) {
                usedResults [count,0] = inputData [i,0];
                usedResults [count,1] = inputData [i,1];
                usedResults [count,2] = inputData [i,2];
                usedResults [count,3] = inputData [i,3];
                usedResults [count,4] = inputData [i,4];
                usedResults [count,5] = inputData [i,5];
                count++;
            }
        }
        return usedResults;
    }

    // Groups results together with a specific entity
    public string[,] filterResults (int nameAxis, int dataAxis, string[,] inputData) {
```

```
List<string> filterOptions = new List<string>();
for (int i=0; i<inputData.Length/6; i++) {
    if (filterOptions.Contains (inputData[i, nameAxis]) == false) {
        filterOptions.Add (inputData[i, nameAxis]);
    }
}
string[,] filteredResults = new string[filterOptions.Count,2];
for (int i=0; i<filterOptions.Count; i++) {
    int count = 0;
    float total = 0f;
    for (int j=0; j<inputData.Length/6; j++) {
        if (inputData[j,nameAxis] == filterOptions[i]) {
            count = count + 1;
            if (inputData [j, dataAxis] == "True") {
                total = total + 1f;
            }
        }
    }
    filteredResults [i, 0] = filterOptions [i];
    filteredResults [i, 1] = string.Format ("{0:N3}", total / count);
}
return filteredResults;
}

// Main run Loops that Loops as long as the scene is open
public IEnumerator RunLoop () {
    while (true) {
        // Handles user input
        if (up == true) {
            if (focusRow > 0) {
                focusRow--;
            } else if (displayStartIndex > 0) {
                displayStartIndex--;
                rowPopulate ();
            }
            up = false;
        }
        if (down == true) {
            if (focusRow < rows.Length - 1 && focusRow < dataItems.Length - displayStartIndex - 1) {
                focusRow++;
            } else if (displayStartIndex < dataItems.Length - rows.Length) {
                displayStartIndex++;
                rowPopulate ();
            }
            down = false;
        }
        if (g == true) {
            g = false;
            yield return StartCoroutine (runGraph());
            yield return null;
        }
        yield return null;
    }
}

// Generates a graph
public IEnumerator runGraph() {
    for (int i = 0; i < rows.Length; i++) {
        rows [i].toggleActive (false);
    }
    for (int i = 0; i < bars.Length; i++) {
        bars [i].toggleActive (true);
    }
    graph.gameObject.SetActive (true);
    dataOutline.gameObject.SetActive (false);
    heading.gameObject.SetActive (false);
    while (g == false) {
        yield return null;
    }
    g = false;
    for (int i = 0; i < rows.Length; i++) {
        rows [i].toggleActive (true);
    }
}
```

```
    }
    for (int i = 0; i < bars.Length; i++) {
        bars [i].toggleActive (false);
    }
    graph.gameObject.SetActive (false);
    dataOutline.gameObject.SetActive (true);
    heading.gameObject.SetActive (true);
    up = false;
    down = false;
}

// Manages a change in the value of Dropdown A
public void DDChange () {
    if (dropdownA.options.Count == 4) {
        dropdownA.ClearOptions ();
        dropdownA.AddOptions (new List<string> (new string[] {"Whole Student Body", "Classes", "Individual Students"}));
        dropdownA.value = dropdownA.value - 1;
    }
    dropdownB.gameObject.SetActive (true);
    dropdownB.ClearOptions ();
    dropdownB.AddOptions (new List<string> (new string[] {""}));
    if (dropdownA.value == 0) {
        dropdownB.AddOptions (new List<string> (new string[] {"Student: General", "Student: Topic", "Class: General", "Class: Topic"}));
    }
    else if (dropdownA.value == 1) {
        for (int i=0; i<classes.Length; i++) {
            dropdownB.AddOptions (new List<string> (new string[] {classes [i] + ": General", classes [i] + ": Topic" }));
        }
    }
    else if (dropdownA.value == 2) {
        for (int i=0; i<students.Length; i++) {
            dropdownB.AddOptions (new List<string> (new string[] {students[i] + ": General", students [i] + ": Topic" }));
        }
    }
    dropdownB.value = 0;
}

// // Manages a change in the value of Dropdown B
public void DDBChange () {
    if (dropdownB.options[0].text == "" ) {
        int value = dropdownB.value;
        dropdownB.options.RemoveAt (0);
        dropdownB.value = value - 1;
    }
    string[] option = dropdownB.captionText.text.Split (':');
    if (option [option.Length - 1] == " General") {
        string[,] data = new string[0,0];
        if (dropdownA.value == 0) {
            if (option [0] == "Student") {
                data = filterResults (3, 0, results);
            } else {
                data = filterResults (4, 0, results);
            }
        }
        else if (dropdownA.value == 1) {
            data = reduceDataSet (4, option [0], results);
            data = filterResults (3, 0, data);
        }
        else {
            data = reduceDataSet (3, option [0], results);
            data = filterResults (1, 0, data);
        }
    }
    dataItems = new DataItem[data.Length/2];
    for (int i = 0; i < dataItems.Length; i++) {
        dataItems [i] = new DataItem (data [i, 0], data [i, 1]);
    }
    rowPopulate ();
    graphPopulate ();
}
else if (option [option.Length - 1] == " Topic") {
    dropdownC.gameObject.SetActive (true);
}
}
```

```
}  
  
// // Manages a change in the value of Dropdown C  
public void DDCChange () {  
    if (dropdownC.options[0].text == "") {  
        int value = dropdownC.value;  
        dropdownC.options.RemoveAt (0);  
        dropdownC.value = value - 1;  
    }  
    string[,] dataPT1 = reduceDataSet (2,(dropdownC.value+1).ToString(),results);  
    string[,] dataPT2 = reduceDataSet (2,(dropdownC.value+7).ToString(),results);  
    string[,] data = new string[(dataPT1.Length/6)+(dataPT2.Length/6),6];  
    int halfway = dataPT1.Length / 6;  
    for (int i = 0; i < data.Length/6; i++) {  
        if (i < halfway) {  
            data [i, 0] = dataPT1 [i, 0];  
            data [i, 1] = dataPT1 [i, 1];  
            data [i, 2] = dataPT1 [i, 2];  
            data [i, 3] = dataPT1 [i, 3];  
            data [i, 4] = dataPT1 [i, 4];  
            data [i, 5] = dataPT1 [i, 5];  
        } else {  
            data [i, 0] = dataPT2 [i - halfway, 0];  
            data [i, 1] = dataPT2 [i - halfway, 1];  
            data [i, 2] = dataPT2 [i - halfway, 2];  
            data [i, 3] = dataPT2 [i - halfway, 3];  
            data [i, 4] = dataPT2 [i - halfway, 4];  
            data [i, 5] = dataPT2 [i - halfway, 5];  
        }  
    }  
    string[] option = dropdownB.captionText.text.Split (':');  
    if (dropdownA.value == 0) {  
        if (option [0] == "Student") {  
            data = filterResults (3, 0, data);  
        } else {  
            data = filterResults (4, 0, data);  
        }  
    } else if (dropdownA.value == 1) {  
        data = reduceDataSet (4, option [0], data);  
        data = filterResults (3, 0, data);  
    } else {  
        data = reduceDataSet (3, option [0], data);  
        data = filterResults (1, 0, data);  
    }  
    dataItems = new DataItem[data.Length/2];  
    for (int i = 0; i < dataItems.Length; i++) {  
        dataItems [i] = new DataItem (data [i, 0], data [i, 1]);  
    }  
    rowPopulate ();  
    graphPopulate ();  
}  
}
```

MySQLconnector

```
using UnityEngine;  
using System;  
using System.Text;  
using System.Collections;  
using System.Collections.Generic;  
using MySql.Data;  
using MySql.Data.MySqlClient;  
  
public class MySQLconnector : MonoBehaviour {  
  
    public string database; //Passed to the script: Database details  
    public string host;  
    public string user;  
    public string password;  
  
    private string connectionString; // Private: Connection variables  
    private MySqlConnection connection;  
    private string query;  
    private MySqlCommand command;
```



```
private MySqlDataReader reader;

// Runs when the game object first appears
void Awake () {
    DontDestroyOnLoad (this.gameObject);
    connectionString = "Server="+host+";Database="+database
        +";User="+user+";Password="+password+";Pooling=true";
    try {
        connection = new MySqlConnection(connectionString);
        connection.Open();
        Debug.Log("MySQL State: "+connection.State);
    } catch (Exception e) {
        Debug.Log (e);
    }
}

// Closes the connection when the application is closed
void OnApplicationQuit () {
    if (connection != null) {
        Debug.Log (connection.State);
        connection.Close();
        Debug.Log (connection.State);
        Debug.Log ("My SQL Connection Closed");
    }
}

// Finds the details of an item
public string[] findItem (int itemID) {
    string[] item = new string[3];
    query = "SELECT * FROM item WHERE itemID =" + itemID;
    command = new MySqlCommand (query, connection);
    reader = command.ExecuteReader ();
    while (reader.Read()) {
        item [0] = reader.GetString (1);
        item [1] = reader.GetString (2);
        item [2] = reader.GetString (3);
    }
    reader.Close ();
    return item;
}

// Finds the questions for a specific topic
public string[,] findTopicQuestions (int topicNo) {
    query = "SELECT question.questionID, question.questionText, " +
        "question.answer FROM question WHERE question._topicID =" + topicNo;
    command = new MySqlCommand (query, connection);
    reader = command.ExecuteReader();
    int count = 0;
    while (reader.Read ()) {
        count++;
    }
    string[,] questions = new string[count,4];
    count = 0;
    reader.Close ();
    reader = command.ExecuteReader ();
    while(reader.Read()) {
        questions[count,0] = reader.GetString(0);
        questions[count,1] = reader.GetString(1);
        questions[count,2] = reader.GetString(2);
        count++;
    }
    reader.Close ();
    for (int i = 0; i < count; i++) {
        questions [i, 3] = findDifficulty (questions[i,0]).ToString();
    }
    return questions;
}

// Checks if a specific records exists
public bool existCheck (string primaryKey, string tableName) {
    query = "SELECT * FROM " + tableName + ";";
    command = new MySqlCommand (query, connection);
    reader = command.ExecuteReader();
    while (reader.Read ()) {
```

```
        if (primaryKey == reader.GetString(0)) {
            reader.Close ();
            return true;
        }
    }
    reader.Close ();
    return false;
}

// Adds a new student
public void newStudent (string studentID, string firstName, string lastName, string studentClass)
{
    query = "INSERT INTO student (studentID, firstName, lastName, _classID) VALUES (" + studentID
    + ", '" + firstName + "', '" + lastName + "', '" + studentClass + "')";
    command = new MySqlCommand (query, connection);
    reader = command.ExecuteReader ();
    reader.Close ();
}

// Adds a new answer instance
public void newAnswerInstance (string studentID, bool correct, string questionID) {
    query = "INSERT INTO answerInstance (correct, _studentID, _questionID) VALUES (" + correct + ",
    + studentID + ", " + questionID + ")";
    command = new MySqlCommand (query, connection);
    reader = command.ExecuteReader ();
    reader.Close ();
}

// Finds the difficulty of a specific question
public float findDifficulty (string questionID) {
    float correct = 0;
    float total = 0;
    query = "SELECT correct FROM answerInstance WHERE _questionID = " + questionID + ";";
    command = new MySqlCommand (query, connection);
    reader = command.ExecuteReader();
    while (reader.Read ()) {
        if (reader.GetString (0) == "True") {
            correct++;
        }
        total++;
    }
    reader.Close ();
    if (total > 0) {
        return correct / total;
    } else {
        return 0;
    }
}

// Checks if a student is A-Level or GCSE
public bool isAlevel (string playerID) {
    query = "SELECT student.studentID, class._examID FROM student INNER JOIN class ON student._classID = class.classID";
    command = new MySqlCommand (query, connection);
    reader = command.ExecuteReader();
    while (reader.Read ()) {
        if (reader.GetString (0) == playerID) {
            if (reader.GetString(1) == "2") {
                reader.Close ();
                return true;
            }
        }
    }
    reader.Close ();
    return false;
}

// Finds the number of items
public int findNumberOfItems () {
    query = "SELECT itemName FROM item";
    command = new MySqlCommand (query, connection);
    reader = command.ExecuteReader();
    int count = 0;
    while (reader.Read ()) {
```

```
        count++;
    }
    reader.Close ();
    return count;
}

// Finds the name of a specific topic
public string findTopicName(int topicNumber) {
    query = "SELECT topicName FROM topic WHERE topicID = " + topicNumber + ";";
    command = new MySqlCommand (query, connection);
    reader = command.ExecuteReader();
    string topicName;
    if (topicNumber < 7) {
        topicName = "GCSE ";
    } else {
        topicName = "A-Level ";
    }
    while (reader.Read ()) {
        topicName = topicName + reader.GetString (0);
    }
    reader.Close ();
    return topicName;
}

// Deletes a question
public void deleteQuestion(string questionID) {
    query = "DELETE FROM answerInstance WHERE _questionID = " + questionID + ";";
    command = new MySqlCommand (query, connection);
    reader = command.ExecuteReader();
    reader.Close ();
    query = "DELETE FROM question WHERE questionID = " + questionID + ";";
    command = new MySqlCommand (query, connection);
    reader = command.ExecuteReader();
    reader.Close ();
}

// Updates a question
public void updateQuestion (string questionText, string answer, string topicID, string questionID
) {
    query = "UPDATE question SET questionText = '" + questionText + "', answer = '" + answer + "', _t
opicID = '" + topicID + "' WHERE questionID = " + questionID;
    command = new MySqlCommand (query, connection);
    reader = command.ExecuteReader ();
    reader.Close ();
}

// Adds a question
public void addQuestion (string questionText, string answer, string topicID) {
    query = "INSERT INTO question (questionText, answer, _topicID) VALUES ('"+questionText+"', '"+a
nswer+"', '"+topicID+"');";
    command = new MySqlCommand (query, connection);
    reader = command.ExecuteReader ();
    reader.Close ();
}

// Finds all the results
public string[,] gatherResults () {
    // Correct, Date, Topic, student ID, Class, Name
    query = "SELECT answerInstance.correct, DATE_FORMAT(answerInstance.date, '%d/%m/%Y'), question
._topicID, " +
        "answerInstance._studentID, student._classID, student.firstName, student.lastName " +
        "FROM answerInstance INNER JOIN question ON answerInstance._questionID = " +
        "question.questionID INNER JOIN student ON answerInstance._studentID = student.studentID;"
;
    command = new MySqlCommand (query, connection);
    reader = command.ExecuteReader();
    int count = 0;
    while (reader.Read ()) {
        count++;
    }
    string[,] results = new string[count,6];
    count = 0;
    reader.Close ();
    reader = command.ExecuteReader ();
}
```

```
        while(reader.Read()) {
            results[count,0] = reader.GetString(0);
            results[count,1] = reader.GetString(1);
            results[count,2] = reader.GetString(2);
            results[count,3] = reader.GetString(3);
            results[count,4] = reader.GetString(4);
            results[count,5] = reader.GetString(5) + " " + reader.GetString(6);
            count++;
        }
        reader.Close ();
        return results;
    }

    // Finds the primary keys of all the records in a particular table
    public string[] findAll (string tableName) {
        query = "SELECT * FROM " + tableName + ";";
        command = new MySqlCommand (query, connection);
        reader = command.ExecuteReader();
        int count = 0;
        while (reader.Read ()) {
            count++;
        }
        string[] instances = new string[count];
        count = 0;
        reader.Close ();
        reader = command.ExecuteReader ();
        while(reader.Read()) {
            instances[count] = reader.GetString(0);
            count++;
        }
        reader.Close ();
        return instances;
    }
}
```