The name of your project

CS39440 Major Project Report

Author: Your Name ([rhs24@aber.ac.uk](mailto:rhs24@aber.ac.uk))

Supervisor: Dr Edel Sherratt (eds@aber.ac.uk)

19th February 2019

Version 1.0 (Draft)

This report is submitted as partial fulfilment of a BSc degree in  
Computer Science (G400)

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Wales, UK

Declaration of originality

I confirm that:

* This submission is my own work, except where clearly indicated.
* I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
* I have read the regulations on Unacceptable Academic Practice from the University’s Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
* In submitting this work, I understand and agree to abide by the University’s regulations governing these issues.

Name Rhodri Smith

Date 22/04/2020

Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name Rhodri Smith

Date 22/04/2020

Acknowledgements

I am grateful to friend and family for mental and financial support, not just within this project but through the 3 years of university so far. I am also grateful to the tutors and

I’d like to thank…

Abstract

This document will contain the design and implementation of a web based game that interacts differently based on the type of user. The project is aimed at primary school students and is designed to allow a student to play a simple game, where a score is recorded and uploaded to a database. The students and their score will then be visible to a staff member, wherethey will be able to change the student’s difficulty for the game. A guest account was also necessary for a someone to be able to test the site. The website contains a login screen for the user and will require the user to either sing in or be a guest to visit other pages.

The project was designed with the basic web languages and within an IDE called Atom[7]. The languages used where front end languages and backend languages.

Contents

[1. Background, Analysis & Process 7](#_Toc39538220)

[1.1. Background 7](#_Toc39538221)

[1.1.1. Languages 7](#_Toc39538222)

[1.1.2. The Game 7](#_Toc39538223)

[1.1.3. The Table 7](#_Toc39538224)

[1.1.4. Security 7](#_Toc39538225)

[1.1.5. Database 8](#_Toc39538226)

[1.1.6. Design ideas 8](#_Toc39538227)

[1.2. Analysis 8](#_Toc39538228)

[1.2.1. Aims and objectives 8](#_Toc39538229)

[1.2.2. Issues and Approach 9](#_Toc39538230)

[1.2.3. Other sites 11](#_Toc39538231)

[1.3. Process 11](#_Toc39538232)

[2. Design 12](#_Toc39538233)

[2.1. Overall Architecture 12](#_Toc39538234)

[2.1.1. UML and other diagrams 12](#_Toc39538235)

[2.2. Detailed Design 13](#_Toc39538236)

[2.2.1. Login page 13](#_Toc39538237)

[2.2.2. Game Page 14](#_Toc39538238)

[2.2.3. Staff and Admin Page 16](#_Toc39538239)

[2.3. Other Relevant Sections 17](#_Toc39538240)

[2.3.1. Security 17](#_Toc39538241)

[3. Implementation 18](#_Toc39538242)

[3.1.1. The Table 18](#_Toc39538243)

[4. Testing 20](#_Toc39538244)

[4.1. Overall Approach to Testing 20](#_Toc39538245)

[4.2. Automated Testing 20](#_Toc39538246)

[4.2.1. Unit Tests 20](#_Toc39538247)

[4.2.2. User Interface Testing 20](#_Toc39538248)

[4.2.3. Stress Testing 20](#_Toc39538249)

[4.2.4. Other Types of Testing 20](#_Toc39538250)

[4.3. Integration Testing 20](#_Toc39538251)

[4.4. User Testing 20](#_Toc39538252)

[5. Critical Evaluation 21](#_Toc39538253)

[6. Annotated Bibliography 22](#_Toc39538254)

[7. Appendices 23](#_Toc39538255)

[A. Third-Party Code and Libraries 24](#_Toc39538256)

[B. Ethics Submission 25](#_Toc39538257)

[C. Code Samples 25](#_Toc39538258)

# Background, Analysis & Process

## Background

This section will talk about the background research that was done to complete the project. This section will talk about the different sections that needed to be looked into to keep the project up to standard. This section mostly talks about the research that was done before the project was started.

### Languages

Researching the different languages was a major section of research, without this the project would not have been able to be completed. This was mainly looking at the way the languages should be set out, e.g. should the languages be in separate files. The research also took into consideration how the files should be set out, e.g. for the database connection it should not be stored within the server, another would be to have a main CSS file that would store data for items that are on every page, e.g. the footer. Node.js was another language that was investigate briefly for the project

### The Game

Creating the game was a large area of research, how and what sort of game was. During this phase of the project a few different areas to do this was investigated. One area was creating a canvas game within JavaScript. Another method was to use a framework that uses a canvas and WebGL. This would have been phaser [1], which is free to use software that works within the browser. The final way that was investigated was to use just basic HTML to create basic games. Using just html would lower the amount of animation that could be done within the game, however.

The canvas element that could have been used for the game would have allowed a dynamic 2d shapes. It is not designed specifically around creating games however, it is possible to make them within the canvas.

Research was also looked into how the questions and answered would be saved on the page, would they be stored on the database and pulled into an array for the JavaScript to run, or would the page randomly create the questions. Storing the questions would require a table to store them all and a way to upload new questions. Randomly making them would not require this, but would require the creation of random questions and answers that are appropriate, e.g. incorrect answers need to be realistic to the real answer.

A multi canvas approach was looked into for this project, since detection of the canvas itself is easily possible and could allow for some sort of visually stimulating affect for when the user clicks on an answer. Issues again with this however is scalability for different resolutions

### The Table

Creating the table was another section that was investigated. During this time DataTables [2] was one table system that was looked into. DataTables [2] allow a more advanced way to create and interact with tables, this includes a search feature and a feature that added more than one page that a user can view. Data Tables [2] is a free to use plugin that uses jQuery JavaScript to create events within the website. DataTables [2] would bring an advanced feel to the website, that allowed complexes tools to be accessed with ease. DataTables [2] requires a table to be created and sort of acts as an advanced skin for the table, e.g. without DataTables [2], the table will just look very basic and would not contain items such as a search feature or a page feature.

Research was also done into the different styles a table can be, this research was done from the non DataTables [2] point of view, as with the script, it would not be as necessary.

### Security

Security was another item that needed to be researched, this is to stop anything happening that should not. The research was done into looking not only security of user’s data, but also into the security of the website, e.g. can a user just break the website by entering the wrong data or entering data where they should not. The security is paramount to the website not being taken down or data being stolen. A section of research within security investigated the login area of the project and investigated PHP commands that would do thing such as hashing passwords. Hashing passwords is a secure way to store the passwords. It changes the passwords from plane text and changes it into a bunch of random text. This means that if someone gets access into the database, the passwords are still security for the users.

Research was also done into exploring the different types of security issues a website could have e.g. SQL injections, and the ways a developer could stop these from happening. If a site gets SQL injected, someone could easily steal data from the database which they should not be able to access.

### Database

Research into the database that was storing the information was a crucial part for the project, as it would need to put into detail how the data would be stored, what level of normalisation would be used and the formats that the data would be stored as, e.g. text or int. The type of database was also investigated and the two main types that where investigated was a PostgreSQL and MySQL. These are both popular database management systems and both are free to use (MySQL under the terms of the ‘GNU General Public License [3]’). PostgreSQL is more advanced with its datatypes, allowing support of arrays a user defined types while MySQL only allows standard types. Some of the detail that was taken into consideration was from a website: PostgreSQLTutorial.com [4]. This website took into consideration the pros and cons of PostgreSQL and MySQL. The database research also required investigation into how a group of users (school) data would be stored. This meant looking into should they have an entire database to themselves or only a table.

Looking at using the database long term, using command line was an option however, research was done into using other methods one method was the phpMyAdmin [5] tool. This tool allows for easy maintenance of use of the database without having to have the knowledge of the command line commands. It also allows a user to control the website from anywhere, whereas to do this otherwise a user would need to use something such as TeamViewer [6] to control the database off site. TeamViewer lets a user to connect to a device and control it from anywhere, e.g. at home.

### Design ideas

Before starting the project, a few design ideas where looked into, like how the user would get from one screen to another. This would range from the colours the website will have, to how the user would navigate the website. Would the user be able to navigate the website themselves or would they only be allowed onto certain pages that they are taken too?

The colours that the website will be was also investigated, e.g. what colours go well together and what don’t work well together. If the website had too many bright colours, the users eyes might get tired of looking at the page after a while.

The way the page is set out was also took into consideration, as if everything was on one side of the page. It would look rather unprofessional. Flexbox was one tools that was researched, it is built into CSS and allows a user to easily set out a website through a flexible box like setup. It work by using dividers to dynamically set different sizes for other dividers within the site. A site that gave good information for the Flexbox tool was called CSS-Tricks [12], and it gave a good amount of information about the tool. The site also contains a nice set of images that easily described how each element within it would change the page, e.g. ‘flex-direction: column’ on the ‘body’ tag would set everything in the ‘body’ tag of the webpage to be in the center of the screen and it would line up the dividers into a column.

## Analysis

This section will talk about the different approaches and objectives of the project. Without this section, the reader would not be able to understand what the aims of the project would be. There are three sections to this heading, the first being the aims and objectives of the site. This section should give a clear explanation of what needed to be done to complete the project

The second section are the issues and approach for the site. This section looked at the different issues that may be encountered within the creation of the site. While also talking about what way the website was going to be constructed, e.g. how will the game be made.

The third section talks about the different websites that where investigated. This section talks about the different pages and their positives and negatives. This section will help understand the good things and the bad things that could go into a website.

### Aims and objectives

For the project, there were a few main aims and objectives that needed to be completed to fulfil the project. Other smaller aims where also made, however if these did not get completed, the main areas of the project would still be completed, and the project would run as needed. These aims where to have a game that would upload a user’s (student) score to a database, an area that another user (teacher) can view the score. The teacher would also need to be able to increase the difficulty for the students game. A login screen was also necessary for the project and an admin section. There would also need to be an option for a user to connect to the website through a guest account that would not save any details

One of the main aims and objectives that where created was to create a simple game that would track a user’s score and then upload it to a database. This Game would need to be a basic game that’s aimed towards young people that are primary school age.

Another main objective was to complete a login screen for the user. This would allow the user to login only if they could. The page would have to take 4 different types of accounts to 2 different places. The first being a guest account and the student account, that needed to the game page of the website. Other 2 accounts are the

### Issues and Approach

#### Languages

During the time researching the project, a few issues came to light. One being that php and JavaScript did not work very well together, or in some areas not at all. This being an issue when needing to pass variables such as difficulty to the PHP functions, e.g. changing the user’s difficulty. To get around this issue, the website used forms and the submit button, as the php could read the data that was entered into the forms and work through it that way. Another way this could have been done would be through cookies and changing data within it and then getting the PHP to read that file for the needed data.

As for communication languages, only English was used within the website.

#### Multi-Page Site

Another issue that came up was how would the website remember the users details once they go onto the next page. While researching, 2 ways came up more common that not on how to do this. Cookies and sessions, cookies storing the information locally and sessions storing them on the server side. For the project, sessions where used as no long-term data needed to be stored. The data that was passed from one page to another was just the user’s username and table name (school), if the user is an admin, another session is sent that tells the website that they are an admin.

#### The Game

When creating the games for the website, research was done into what type of game should be created and how. Within the project, the HTML, non-canvas approach was taken, this was partly to do with the webpage being able to change size based on monitor resolution. This approach was take as it would be easily reusable and would be easy for the user to play/understand. The main issue with a canvas approach, is that because the canvas is not designed for game or really even dynamically changing items, e.g. items moving on the canvas. This does not mean its not possible however it is mainly designed to allow more graphically enhanced items to be created. Another issue with it is that to detect clicks, the coordinates of the clickable items must be imputed into the code. This meant that a non canvas approach felt a lot more appropriate for the needed game and design.

#### The Table

The approach for the table that was used within the project was to use the DataTables [2] as it gives a more advanced while still being easy to use for the user and developer. DataTables [2] uses jQuery and JavaScript which is easy to work with and allows events such as onclick events for when the user clicks on specific rows or columns. DataTables[2] also gives a good professional look to the table, while also giving it some ease of life features, e.g. a user can choose how many items they can see at once on the table. This approach allowed a much more professional feel to the website while being mostly simple to use.

#### Security

Security is another issue that needed to be investigated and taken into consideration when creating the website. Website require a lot of security to stop users from either stealing information or breaking the website.

#### Add New Students

The Website will not allow new students to be created through the site. This is partly to do with it not feeling appropriate for a new user to be added, as they would need to be added through the staff members section, which should not be able to access more than the users details and allow them to change the difficulty. The admin is also notable to add a user to the database through the website, however if they had access to the database, they would be able to add a user from there. A school would need to register their students and staff members when singing up to the website. A school will then need to contact the company to be able to access the database and add a student though the database. This was partly to do with the ease of adding someone through the phpMyAdmin [5]. Which also lowers the risk of errors getting into the database, e.g. duplicates. This does not mean however, the users details cannot be changed, data such as the users name and certain other details can be changed through the website however, so an error such as a misspelling exists, this can be changed by one of the admins within the website

#### Login Page

The login page will need at least 3 boxes, for the school name, username and password. These are necessary for a user to login and without these the login page will not work. However, within the project there are 4 boxes, this is to do with the full admin accounts, these account need to connect to a separate database, which would be what the first box is used for, however, for the user to connect to the correct school, they will need to enter the school they want to connect to in that box. There were a few choices when doing this however, the first choice did not work correctly. As the submit button is a submit button on the form, it either refreshes the page or takes the user to another page. The original plan was to have a pop up box that appears and the admin to type in the school there, however, since this would not be in the same form, and the submit button changes the page, the code for the pop up would run and then instantly disappear as the user gets taken to another page. This gave two options, option 1 was to add a pop up box on the school side for the admin to type in the school they want to connect to. This box would only show up for admin accounts, however this lead to issues with having two different accounts of admin accounts. The second option, was to add the fourth box, this does make the login page a little bit more untidy and complex to not admin views, however it was needed.

#### The Admin

There are two types of admin within the webpage, one being a school admin, this would be a single account given to the school, this would be given as it means if the school has simple issues, they will be able to correct the issue themselves. There is another type of admin account, these accounts will not be given to schools and will be used by the developers to view specific schools data from the website itself, this could allow them to view the website as a staff member. This could be useful if a school is having an issue or has errors, the developers will be able to see the issue with their own eyes. This could be an issue that seems fine on the database, but on the website, it is not showing up correctly. The admin accounts the school will receive, will login the same way as any other account, however the other admin account will use the bottom box, which is the school the user wants to connect too.

### Other sites

Some of the research that was done before the web application was started was to look at similar websites and try to look at the pros and cons of the websites. This meant that the project would be able to use the advantages of those websites while not having their disadvantages. An example of some pages, where that they did not store a user’s score for long term viewing while others forced the users to login to view the pages.

Kahoot [9] is a site similar to the one that is being developed however, one of the downsides to Kahoot [9] is that it does not keep a record of its users full data on the site for other individuals to use. It does store the data per game, however if a school wanted to use Kahoot [9], they would have to look at every game they have posted to see the results. There is no one page where they can view everything for their students. This is partly to-do with the site not forcing a game player to sing up. The user just follows a link and is then taken to the correct game., they then get a nickname, which they choose themselves or get randomly (depending on what the creator has set). The user then just goes through the question one by one until it has been completed. The site is very nice to look at and playing the games are rather enjoyable. The site also allows the user to create new games, by setting their own choice of question and answers, the site can be used for almost any quiz-based activity.

## Process

Since the start of the project there has been a set out workload for each week, this originally was set out in a basic format of just what should be done each week, this changed however to follow a proper methodology quickly. The original file would have followed a waterfall system however, after consideration a variation of the scrum methodology was taken up. The original format that was looked into was the waterfall system, however this was too basic for the project as it does not allow flexibility for the project if something had changed.

The scrum methodology work by having sprints, which basically splits the project up into different sections, these are called sprints. Sprints usually last for a minimum of a week and at the start of each sprint, and each day, there will be a meeting between all the members that are going to be working on the project. This meeting will outline the different criteria of the project, while also talk about the issues at hand and how long each section will take to complete. During this the members that will be working on the project will talk about how they will be able to complete the project and discus if they have had any issue with the previous sprint. Once the sprint has finished, that section get reviewed by the team to make sure that everything is working correctly.

The methodology of scrum cannot be followed truly as to truly follow it; you must be in team of people which was not possible with the project. This meant that the sections such as having meetings at the start of the week or day was not possible. Instead of having these meetings, it was decided that for the first day of every week, a session of brainstorming was done to find or think of different ways of completing the tasks that needed to be done that week. At the end of each sprit, the program went through a basic testing process to see if it all worked correctly.

GitHub [11] was also used within the project, this allowed for version control to be used within the website. GitHub [11] also stores the users files online, meaning that if something happens to the user’s computer, as long as they have uploaded their files, they should be fine to easily re download the old backup. GitHub [11] also allows the user to create a local repository, this linked with the GitHub [11] desktop application can allow the user to upload and download new changes to their work anytime. The site also knows what files are new and what are old, meaning that if you have older files than what is stored on its repository, it wont upload the old files. This allows the website to work very well when someone is swapping between more than one machine.

Atom [7] allows GitHub [11] to be directly integrated into it, allowing the user to create push and pull requests at anytime. If you make a change within the code, the application will know you have made a change and put the file into an ‘Unstaged’ section for the user to prepare for an upload to the repository if they want too. By having these two integrated together, it became very simple to make sure all the data was backed up.

# Design

## Overall Architecture

#### Layout

The layout for the website was to set everything to be in the centre of the screen, nothing on the left or right side, this is partly to-do with the website not needing any navigation by the user. Having everything centred makes the website look more professional but also makes the website change size based on the resolution of the monitor. The login page is not affected by the display changes however. The website uses a tool called flex-box within CSS. It allows a website to be set out in particular ways that give them professional looks. The flex box tool was used on all the sites within their CSS files.

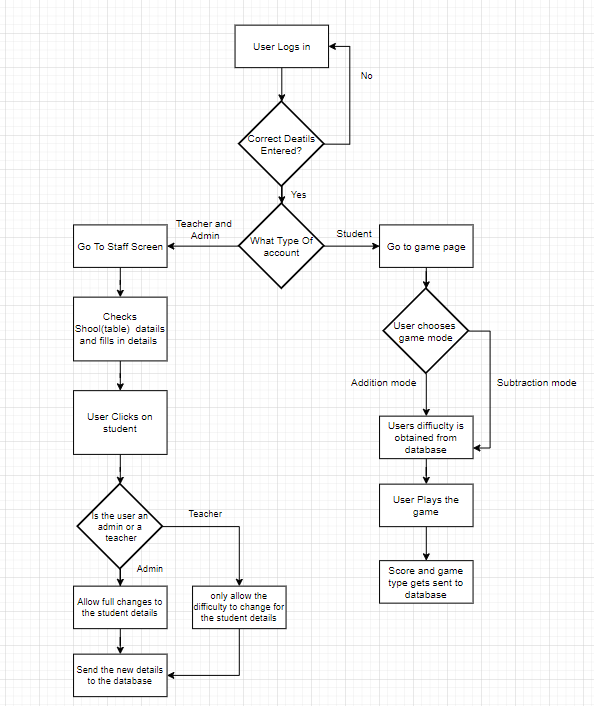
### UML and other diagrams

To create these diagrams, a website called draw.io [13] was used. This is a free to use site that allows the user to easily create different diagrams, these range from UML diagrams to flowcharts. This site allowed the diagrams to be shown bellow to be easy to view while still showing easy connections between each sections. With the diagrams however, there will be a paragraph or two to explain what is going on. The user should be able to follow the diagrams without the paragraphs however, detail was decided to be added, to explain every section in detail.

The diagram bellow shows the basic flow for the website, from what the user does to what the website also does backend. This is just a basic diagram to easily show the reader the basics of the website. As the reader can see the user will start at the top on the login screen. Once the user logs in, the system will then check if the password is correct, if the password is correct, the website will then check what account type the user is.

If the user is an admin or staff member, the diagram would go down the left branch. The left side shows the user being taken to the staff screen. This system then checks the detail of that user to fill in the table with appropriate details, e.g. students. The next box shows what happens when a user clicks on one of the table rows. This shows what a user can do based on what account type they are, if they are a staff account, all they can change is the difficulty of their students, however, if they are an admin account, they can change all the details that are shown on the table. The last box on this side shows the data that the user has entered, changing the clicked-on item in the database, not just the table. This is just a basic insight on how the staff and admin pages work, this is not a detailed section for the reader to view.

The right-hand side of the diagram shows where the user would be taken if they where the student or guest profiles. The first box within this branch of the diagram shows the user being taken to the game page. This is the page that contains the game that the user will be playing. The user can then choose the game mode that they would like to play the game on, this is a diamond as there are multiple options that change how the game can run, however still all lead to the same outcome for the rest of it. The next box then shows the system getting the users difficulty, this gets the difficulty from the online database and is not something that the user sets when starting the game. The next box shows the user playing the game. The users score will change based on the amount of correct answers they get e.g. a correct answer gives +1 to their score. The score will always start at 0. The next box and last box show’s the users score being sent to the database. This then would show up on the staff pages table as a new score. This is also a very basic flow for the website.



## Detailed Design

### Login page

The login page’s design is designed to look simply, allowing people to easily go onto the site without any issues. The design makes it, so the user only has to enter their details in and are then taken to the correct place. The user must enter their details into 3 boxes, one being the school box, this would be the table name within the code. The second two input boxes are for the username and password of the users, this gets used within the database to search for the user. The third box would only need to be filled in by and admin, and would just be the school name that they wish to connect too

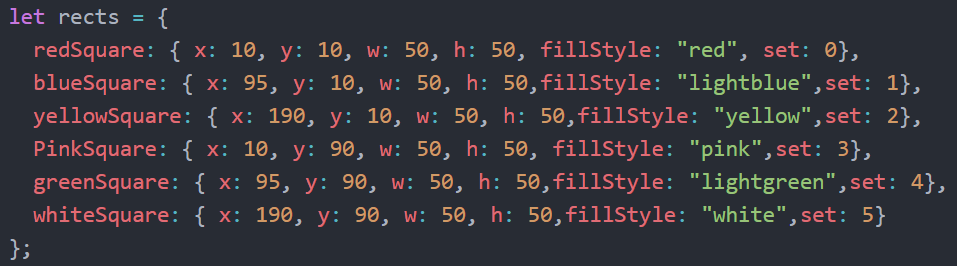
The page uses HTML forms and PHP to complete the login and take the user to the correct page. The pages that the user will be sent to will be dependent on the type of user, a student and guest will be taken to the game page, while the admin and the staff accounts will be taken to the staff page. This page must be the first page that a user sees as without this page, the user would break the other pages as their details would not be set for the pages to run off, e.g. username

### Game Page

The Game screen was also created in a way that the user can easily use and get through. It works by requiring the user to click on the ‘click me’ button and then after that the user just clicks on one of the coloured boxes to answer. The game will then automatically submit the score for the user. The game for the game page will be in the centre of the screen, with the buttons that the user can click on being just under them. The question will come up above the boxes and the answers will show up in the boxes themselves. Making it easier for the user to know what box to click on to give their answer. The text within the boxes and the boxes will be reasonably sized for the user to see the answer. Without the JavaScript, the page is very basic looking, this is due to the squares that the user would click on, are created within the script. By creating the squares in the script, the developer could easily change how many squares appear on the screen at once, e.g. right now there are 6 squares, however it could easily be changed to 4, either by removing 2 lines of code in the array that stores the squares details, or by adding an if statement and 2 arrays, each running based on what the users difficulty is set too.

The users score is sent to the database through a form. The user will not see this form as never shows up on screen and the form is automatically filled in and submitted. This was done so that someone cannot accidently not submit their score and close the page.

The game works by giving the player a completely random question, this is not a question that is stored anywhere. This lowers the chance of a use getting a repeat question and does not require a user to put every possible question they want into a database. The game is based off users clicking on 6 coloured squares, this square is not coded onto the HTML itself, they are instead added after through JavaScript on the load of the page. The data for the squares are stored within an array in the JavaScript. This array stores the coordinates for the squares, the hight, width and colour of the squares. This allows the user to easily add more or remove the number of squares when needed. This could also be used to change the number of squares based on difficulty; however, the project does no incorporate this at the time of this report.



Since the squares are not created within the HTML and the JavaScript must create them instead, a function was required that would allow the creation and put them into the correct place. A for each loop was also required to run, as it would run for the amount of items in the array, and set the details of the squares to be the necessary details. The function was called DivMaker, since it was creating the squares for the game. The ‘currentDiv variable gets the ‘JavaGame’ divider and the code bellow inserts it so that the squares are all put into the ‘JavaGame’ division tag.

A switch() statement is used for the chosen game mode to be used, the switch statement checks the ‘gameMode’ variable to see what it is set too. Within the switch() statement there are if statements for the difficulty the user has been given, this changes a variable called ‘randomNumberDifficulty’, this will change from 10, if the difficulty is 1, too 100 if the difficulty is 2. This then would change the questions from being single digit questions, e.g. 1+2, to double digits, e.g. 25+33. There is a break for this loop that runs when the user get to question 10. This statement changes the question section to show the users answer instead of a new score and removes all of the squares on the website.



Since the game makes random question, a random number generator was needed, this was the Math.random (), this creates a random integer, however, the issue with this is that it contains numbers after the decimal point. Which is why within the code, the Math.floor(Math.random() \* randomNumberDifficulty); was used. This created a random number, which is then multiplied by the randomNumberDifficulty variable. The Math.floor() then forces the number to be a full number and not have a decimal point behind it. The Math.floor() methods where used multiple times within the project, one to create the question, then one to create the wrong answers. The wrong answers however are in a ‘for each()’ loop, this puts the wrong answer into every box. The system then puts the correct answer into one of the squares.

To make sure the correct answer does not come up more than once, the text within the squares are all checked and if they are the same as the answer, they get changed to be a new number. If more than one correct answer showed up, the only one that would have been accepted is the one the system set to be correct. This would be annoying to a user if they clicked the correct answer, but because there were two squares with the same text, they get it wrong. The function that stops this duplication is called ‘OnClickLoop(){}’, and it also adds an event listener onto the squares. This means that when a user clicks on one of the question, the AnswerChecker() function runs.

As the game is now, there are only an addition and subtraction option. This is something that can change in the future. Same with the difficultly, there are only options for difficulty 1 and 2. The subtraction and addition section works the same way, however the subtraction randomiser keeps the number smaller than the answer, which the addition makes the answers bigger than the question.

When the user clicks on a square, the AnswerChecker(){} function runs, this checks to see if the answer the user clicks on was correct. It does this by checking the name of the rectangle, and if it equals the one that holds the ‘correctAnswerSquare variable then the system will run the ScoreUpdate(){} function, which adds to the users score and then redoes the ‘RandomQuestionGenerator(){}’ function to create a new question. The ‘correctAnswerSquare’ is set when the squares are given their answers, the ‘correctAnswerSquare’ variable will be set to the square that contains the correct answers name. If the user got the question wrong, there score does not increase and a new question is given to them. The question number is then also updated to be higher.

Once the user has completed all of the 10 questions that they would have been answering. The system gets rid of all the squares on the page, it then creates a for and fills it in correctly. The for is then automatically submitted and sent to the PHP code to update the database with the users new score.

### Staff and Admin Page

The staff and admin page contain a table within the centre, this table contains all the students that are within the table. The staff member or admin can edit the details within the table by clicking on the user’s row. By clicking on a student within the table, a form pops up and the user can change the details of the student they clicked on. If they are not an admin, they can only change the user’s difficulty. The admins can change all the details that are shown within the table. The admin can also remove any users that are on the table.

The user changes the datils through a form that then sends the users details into a php function that then changes the database. The remove button also uses php to remove the user from the database. Both the ‘submit’ and ‘remove’ buttons are submit buttons for the form, they both also use a hidden column within the table. The hidden column is the users number, this is something that a staff member would not necessarily need to see and could just lead to confusion, this cannot also be changed and is used to change the users details or remove the user from the database table.

A php function is also used to fill in the table, it does this by searching through the database and then uses the ‘echo’ command to place the text into the html code. The echo command places the <tr> and <td> tags into the page for the number of items there are in the database, it does this with a while loop. These tags are the basic HTML table tags that can be used anytime within a table within HTML. It puts the students onto the table through a while loop that runs for the amount of students that are returned from the database and with the correct details for that student.

The system stops anyone who is not an admin from entering the wrong information by making them readOnly. This is something within JavaScript that makes the text boxes not editable. A PHP function is used to check if the $\_SESSION[“Admin”] session does not equal the text admin. This would only equal admin if the user has logged in as an admin.

### Logout code

The logout file is a very small file, all the file does is destroys the current active session, and send the user back to the login page, allowing them to login again if necessary. This is needed for security within the website as without this, someone could be able to access someone else’s account if the sessions have not ended yet. Every page has a logout button that logs the user out in the footer of the page.

### Database connection

The code for the database connection is just a very basic connection. It has 2 functions, the first being to open up a connection to the database ‘OpenCon()’. It has 4 variables set, the first being the host address, which is just "db.dcs.aber.ac.uk". Which sets the Aberystwyth database server as being the host for the connection. The second two are the username and password, which are my details to connect to the database. The fourth and final sets the database for which the system will connect too. This is the ‘cs39440\_19\_20\_rhs24’. The system also has a ‘die’ section and a ‘connect\_error’, this means that if the connection does not happen, it will give an error message to the user to say why the website could not connect to the database.

The file also has a second function, ‘CloseCon($conn)’, which closes the connection to the database, once the system no longer needs its connection, is closes the connection. The database connection file is access by all the pages within the website to get information from the database.

## Other Relevant Sections

### Security

Each page, other than the login, must have had the user to login to access them. If does this by checking if a session is running. Which is done by checking for a session that stores the user’s username if it is empty. It will take the user back to the login page using the header() function, this function allows the developer to send people to other URLs. This header is partly needed as without it, pages such as the staff page would not show anything as it uses the user’s details that they logged in with to get the details for the table.

# Implementation

### The Table

DataTables [2] is a third party library that was used within the code. This was mainly done to create a simple and easy to use table, which would be complex. Without data tables, issues with bigger tables would come into effect, e.g. if the school had a lot of users, the table may require the user to scroll down a vast amount to get to a specific user. This isn’t really acceptable for a website that other businesses will use. DataTables [2] only gives a table its foundation, e.g. search bars and pages, however, it still needs a basic HTML table to run first before it can initiate, otherwise it just would not work. Without DataTables, to create a similar sort of system would be incredibly complex and would take a vast amount of time, since it adds so much to a website in looks as well as more usability.

At the start of the project any admins would have not just needed to login, but would have then been given a pop-up box to put the school that they wish to connect to into, this became quite difficult with the way the login page was created. Since the login page had a submit button, anything that happened after that click either refreshed the page or took the user to the next page. This got to the point where it was decided to add the fourth login box. The other option was to take the admin to another separate page where there would be a form with one box, that then sets the school they want to access as. The second approach was not take up, this is because it would have changed the original structure of the website. This meant that the first approach was taken, keeping the original structure of the site and still keeping the website easy to use.

PHP was much more difficult than expected, and not because of the language but because of the running the website. For the site to be tested, the developer needed to connect to the Aberystwyth VPN, send the files over and then, through Putty [8] do a ‘fixwebperms’ to set the correct file permissions. This meant that anytime development needed to be done, it had to be on the Aberystwyth VPN. This slowed down the testing for the website massively.

Another issue with the PHP, is that getting errors from it where sometimes very annoying. Such as if you had the wrong table name, there will be no error originally and the developer will have to put something into the code to get the error to show up, e.g. ‘mysqli\_error($conn);’ the ‘$conn’ being the variable for the connection to the database. This did not make it any more difficult however, it just became quite annoying during the testing phase.

#### Different questions

Different types of question where originally going to be implemented onto the site. Spelling questions was one of these types, being a user would get a word that had a letter missing, and then they would have to click on the correct letter that would correspond with the letter that is missing. This was not implemented due to a couple of reasons, one being that the maths questions do not get the data from the database and instead it creates a completely random question. This meant that if the spelling question was created, the code would need to be split up into two different scripts realistically, otherwise the file would get very large and very difficult to read very quickly. This could be done in the same way the addition and subtraction game mode are set, e.g. through a pop-up box. Different scripts will then run based off what the user has chosen to play.

The second reason for the spelling to not be included is that on top of having the maths section to-do, is that the spelling section would be quite complex. The system would have to read the size of the word, it would then need to choose a random letter within that words size. It would then need to remove the letter and replace it with something else, e.g. ‘\_’. The system would then have to be remembered what letter it took out and store it. After this the system would need to choose three other random letters to put into the squares to be incorrect answers. To get a random letter, the JavaScript would need to have an array filled with every letter possible, and then choose one of those letters to input onto it. This could have also been taken further with the difficult part, where two letters where missing and there where two letters in each box. Another part of this would be that all the words would have to be stored somewhere, would they be stored online in a database and then brough down into an array, or would there just be an array already filled with just random words. Another way to get the random words could be to install a third-party framework onto the editor and use that, an example of one of these frameworks would be the random-words [10] framework, this framework generates words to use as sample text and allows the user to get completely random words to be generated.

Since the size of adding the spelling section would be enormous and complex, it was decided that it would not be added at this point in the project and would be added later date.

# Testing

## Overall Approach to Testing

The original approach to testing was going to be to get real people, of the age that the project is aimed for to test it. This would allow the developer to highlight issues that they may not have thought of before. The testers may have also been able to say about things that could have been improved, e.g. the look of the website. A small amount of testing would have been done by the developer however, this would be to test the security of the site and to find any issues that a normal user might not find, but someone trying to break the site would. However due to issues out of the control of the developers, this testing was not allowed to be done. Mainly because access to the younger people become impossible.

This meant that the way the testing needing to be done needed to change. Instead of using younger people, this would be manual testing and user testing but not for the target audience. This meant that the testing will not give as good information as it would if someone from the correct audience group was doing it, e.g. they may not like something that someone of the target audience might like.

This version of the user testing meant that older, more knowledgeable users tested the site. However, they were asked to not find security flaws as their main purpose was to find issues within the game and site itself. Their task was not to try to SQL inject the website and break the site that way. The people that would now be testing the site would be mostly made up of people that are doing a computer science course, with a few less knowledgeable users also in the mix. The non-computer science users where brought in to try to give different points of views. However, all would be much older than the target audience, 20 years old and above.

Automated testing was chosen not to be done within the website, this was down to a lot of the unit testing software that would have been used, would only work for one language, e.g. JavaScript and not PHP. Meaning that more than one third party software was needed to be used. During the research process no real built in unit testing systems where found that could have been used within the project.

## Automated Testing

### Unit Tests

### User Interface Testing

### Stress Testing

### Other Types of Testing

## Integration Testing

## User Testing

# Critical Evaluation

# Annotated Bibliography

[1] photonstorm.com, P., 2020. Phaser - A Fast, Fun And Free Open Source HTML5 Game Framework. [online] Phaser.io. Available at: <https://phaser.io/> [Accessed 28 April 2020].

[2] Datatables.net. 2020. Datatables | Table Plug-In For Jquery. [online] Available at: <https://datatables.net/> [Accessed 28 April 2020].

[3] Gnu.org. 2020. The GNU Operating System And The Free Software Movement. [online] Available at: <https://www.gnu.org/> [Accessed 29 April 2020].

[4] Postgresqltutorial.com. 2020. PostgresqlVs. Mysql: A Comprehensive Comparison. [online] Available at: <https://www.postgresqltutorial.com/postgresql-vs-mysql/> [Accessed 30 April 2020].

[5] contributors, p., 2020. Phpmyadmin. [online] phpMyAdmin. Available at: <https://www.phpmyadmin.net/> [Accessed 30 April 2020].

[6] TeamViewer. 2020. Teamviewer – Remote Support, Remote Access, Service Desk, Online Collaboration And Meetings. [online] Available at: <https://www.teamviewer.com/en/> [Accessed 30 April 2020].

[7] Atom. 2020. A Hackable Text Editor For The 21St Century. [online] Available at: <https://atom.io/> [Accessed 3 May 2020].

[8] Putty.org. 2020. Download Putty - A Free SSH And Telnet Client For Windows. [online] Available at: <https://www.putty.org/> [Accessed 5 May 2020].

[9] Kahoot!. 2020. Kahoot! | Learning Games | Make Learning Awesome!. [online] Available at: <https://kahoot.com/> [Accessed 5 May 2020].

[10] npm. 2020. *Random-Words*. [online] Available at: <https://www.npmjs.com/package/random-words> [Accessed 5 May 2020].

[11] "Build software better, together", GitHub, 2020. [Online]. Available: https://github.com/. [Accessed: 06- May- 2020].

[12] "A Complete Guide to Flexbox | CSS-Tricks", CSS-Tricks, 2020. [Online]. Available: https://css-tricks.com/snippets/css/a-guide-to-flexbox/. [Accessed: 06- May- 2020].

[13] "Flowchart Maker & Online Diagram Software", *App.diagrams.net*, 2020. [Online]. Available: https://app.diagrams.net/. [Accessed: 06- May- 2020].

# Appendices

* 1. Third-Party Code and Libraries
  2. Ethics Submission

* 1. Code Samples