Final Report

Game With A Purpose

Author: *Bernice Thomas*
Project supervisor: *Irena Spasic*
Project Moderator: *Wendy K Ivins*

CM3203
One Semester Individual Project
40 credits

Abstract

A game with a purpose is a method used to encourage humans to do a task that is difficult for a computer to accomplish by presenting it as a game. This report will discuss the planning, implementation, evaluation and conclusion of a game with a purpose involving synonyms, antonyms and hypernyms of words. The project itself will focus on producing both an enticing, addictive game which encourages users to play and a useful database of words collected for the client.

Acknowledgements

I would like to thank my supervisor, Irena Spasic for her advice and support during this project.

Table of Contents

Table of Figures

1 Introduction

As the world continuously turns to a more online world, the task putting all the data from the real world online is becoming increasingly important. Obtaining data from humans is a strenuous and time consuming chore. One of the most challenging aspects of gathering data is finding people willing to spend their time giving it. To solve this issue, for this project I will produce a Game With a Purpose(GWAP) which will automate data collection. The idea behind the game is to use the computational power of humans to perform a task that computers cannot do (originally, image recognition) by packaging the task as a game. [1]

People spend billions of hours a year playing online computer games [2], to utilise this time my game will collect data from game players. The game I will be creating chooses random adjectives or nouns and asks users to provide synonyms, antonyms or hypernyms for the word. Players will be randomly matched with other players online and will compete against one another. Once both players guess the same word, the game moves on to the next word. There will also be a single player version, using WordNet as a knowledgebase allowing users to test their own knowledge of the English language.

To ensure the system is a success, there are two main outputs of the project that will be considered; the game and the data. The game itself should be fun and interactive, different game elements will be added to ensure this. The user should be able to create an account and login, allowing them to keep their high score. Another element that I will add to the game is a leader board. By adding a leader board the game it will become more competitive, encouraging people to play for longer, thus giving more data. By making the game as accessible as possible, more players can play the game meaning more data will be collected. Therefore, it is important to ensure that the website is responsive when accessing it on a mobile or tablet device. The other aspect of the system that must be considered is the data collection. The data should be stored in a meaningful way to ensure it is useful. Due to the high volumes of data being stored, it is important to store it as efficiently as possible, this would help to reduce hardware costs for the final system.

2 Background

I will now be discussing the background of the project including different games GWAP that have previously been developed. I will also be discussing other methods that have been employed instead of GWAP and the reasoning behind them. By researching this area, I will have a better understanding of what makes a more successful system.

2.1 History of games with a purpose

A GWAP is a human-based computation method of outsourcing tasks that may be difficult for computers to solve but much easier for humans by employing gamification. Gamification utilises game elements in a task, making them seem more enjoyable and entertaining. This section of the report will discuss different types of GWAP that have already been developed. I will also inform the reader of the benefits of each system and the issues that arose during their development.

2.1.1 ESP Game

ESP game was developed by Luis Von Ahn in the year 2003 [3] and was the first known GWAP. The game was very similar to the system I will be developing, it featured images that the user had to describe without using the given taboo words. Players were matched with one another and played together to try a both guess the same word. The game made searching for images using words much easier as it produced an extensive database of words correlating to images [4]. This was especially useful in the early 2000's when image recognition was far less advanced as it is now, meaning the tasks were far more difficult for a computer to solve, compared to a person. Below is a screenshot of the game in 2006.

Figure 1 Screenshot of ESP Game [5]

During a Google TechTalk in July 26, 2006 [6] , Luis Von Ahn discussed the issues that arose during the introduction of the game. Some of these issues could possibly occur with my proposed system. One issue was with people cheating by guessing the same, irrelevant words for each image they were shown. To avoid this, Luis incorporated test images into the game. If the user played well with the test images, they assumed their data for the proceeding images was 'good'. Another method used to clamp down on cheaters was to only input a word into the database once it had been matched by 'n' pairs. Louis gave the equation during his lecture which can be seen below. The equation shows that as we increase 'n', the probability of the label not being corrupt is exponential to x.

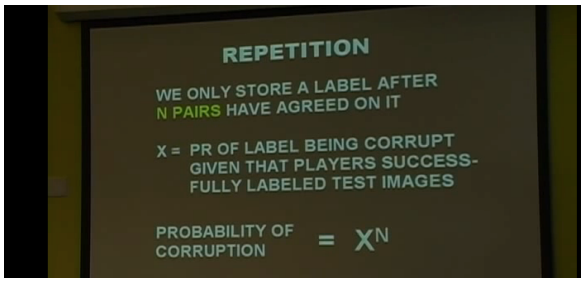

Figure 2 Screenshot from Louis Von Ahn techtalk

This method of cheating could occur in my game. I will avoid this by implementing WordNet to check if a match is valid by and adding in limits to words so they must be guessed a certain number of times before they are declared a valid match. This will help to reduce the chances of any irrelevant words being included in the database.

Increasing the number of games users play is vital to ensure enough data is collected to have a sufficient database. According to Luis Von Ahn during his TechTalk, when players were told that their partner has entered a guess, they played for 4% longer. This will be an important aspect to consider when producing on my GWAP as I should be taking every measure possible to increase playing time of players.

The ESP game became so popular that during its peak, there were over 75 thousand players, with 15 million agreements. One of the drawback of the system according to Louis Von Ahn during his TechTalk is that the players playing the ESP game were bias as only 'certain' people would play the game, rather than the general population. This may skew the results collected by the game, compared to results if you were to ask the general population. Although this may be an issue, there is no real way to fix it.

Google relaunched a version of ESP game in 2011, however it wasn't a game, it was simply an image labelling website called crowdsource [7]. Thanks to the advancing capabilities of artificial intelligence, GWAP are less vital as they were in the 2000's. For instance, Google photos uses image recognition to enable you to search for photos by the elements that are in the picture. This method is much faster than using people to manually describe each picture and far less time consuming. Although this is the case, it may not always be correct and to create the technology in the first place, Google used a combination of both human data and Artificial intelligence. This may be the more future proof method, making both technologies work together rather than just picking one or another. This method is utilized by Google as data collected by its image labelling website is used to help train their AI to recognize the images [8].

2.1.2 Other games with purposes

Along with the ESP game there are many other types of games with a purpose that accomplish different tasks. Another type of GWAP is 'Smorball' and 'Beanstalk', these games, both produced by the same developers asks players to type words they see from images which are scanned pages in the Biodiversity Heritage Library [9]. This results in historical books being transcribed online. Due to the various fonts used in historical literature, it is very difficult for image recognition to reliably read the text, therefore it is helped by the players of the games. One of the elements I particularly appreciate from both examples is the graphics utilized in the games which help it feel more 'game like'.
Nanocrafter is a game developed by the University of Washington [10]. This game uses humans to help to decipher DNA which is difficult for computers to do. Puzzles are shown to players featuring different DNA structures [11], the better the players are, the higher the levels they can progress to, giving them more complex DNA structures. The game utilizes the player's creativity to build systems, allowing computers to focus more on the areas highlighted by humans [12]. This is another prime example of combining the efforts of both humans and machines.

2.2 WordNet

WordNet is a large lexical database of English [13] that was produced by Princetown university in mid-1980s [14] .
WordNet is open source [15] meaning it is available to use for my project without having to ask for permissions. I will be using WordNet as the base for my database, it will be used to make guesses when playing against a computer and it will also be used to help validate if a match is correct. One of the major benefits of implementing WordNet is that they have many easy to use APIs available online, making the system simpler to implement. In the future, the system will not need WordNet to function as it will have more data than WordNet in the databases. However, it is vital to employ WordNet whilst the database for the word game is being produced by the players.

3 Specification & Design

This section of the report will give sensible design solutions to the problems I have been given. I will state what functions I hope the system will have at the end of the project, along with the mechanisms of how they will work. For each design decision I make, I will justify them using data I have found from reliable sources. When developing my project, I have decided to utilise the waterfall model software development process which was First introduced by Dr Winston W. Royce in a paper published in 1970 [16]. I have chosen this method due to its clearly defined stages, compared to the agile development method which can become long and complicated. As my project has a limited timeframe, I think it is more suitable to follow the waterfall methodology, the structure of which can be seen below.

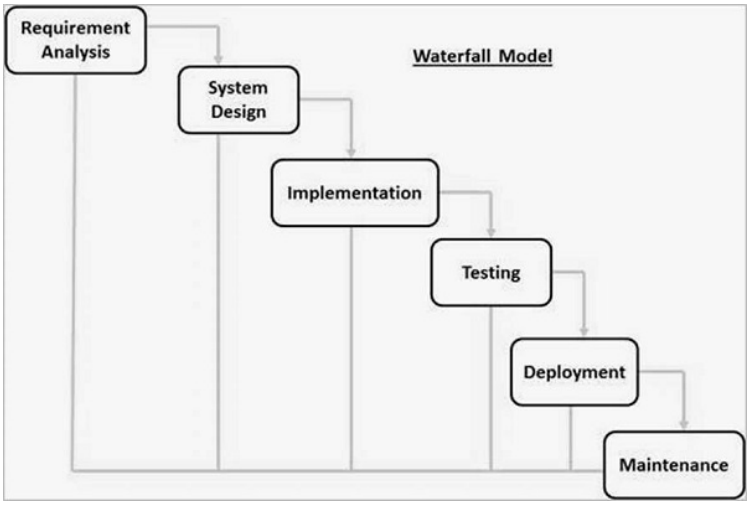

Figure 3: Waterfall Model [17]

3.1 Specification

The specification section of this report will discuss the outcomes I hope to produce from the system. Not only will they act as an aid during the implementation phase, but they will also help me cipher weather or not this project has been a success during the testing of the game.

3.1.1 Requirements

I have divided my requirements into two sections; Functional and non-functional. Functional requirements are specific behaviours that the system should have whereas Non-functional requirements are more design focussed and judge the operation of a system.

Functional requirements

1. Players must be able to create an account
2. Players must be able to play against other players online
3. Players must be able to play against the computer
4. Players must be able to pass on a word if they are stuck
5. There must be a time limit on each game
6. The game must be playable on at least the top 4 most used browsers according to NetMarketShare [18]
7. The game must be playable on mobile phone and tablet devices
8. Players should be able to view a leader board
9. Players could be able to change the difficulty when playing against a computer
10. Players could be able to play as a guest

Non-functional requirements

1. There must be an appropriate colour scheme to allow for colour blindness, therefore colours red and green and the colours blue and yellow cannot be layered on top of one another
2. The game must take no longer than 4 seconds to load
3. The game should be reliable, it should work 99% of the time
4. The text of the game should be big enough to read, all font on the game must be over 15pt
5. It should take no longer than 90 seconds for an average player to sign up to the game
6. Over 50% of players should want to play the game again after playing once

3.1.2 Test Cases

Below are a series of test cases that I have produced which will help me deem weather the project has been a success. Another benefit of producing test cases was that it made me realise areas of the system that I had not yet properly realised such as the fact that along with logging in, I will have to create a mechanism for users to actually create an account.

Test Case ID: 1		**Test Purpose:** Sign Up	
Environment: OS X Google Chrome			
Precondition:			
Test Case Steps: 3			
Step No	Procedure	Response	Pass/Fail
1	Player goes to the website	The first page is loaded	
2	Player clicks sign up	The sign up page is loaded	
3.i	Player enters an email address that is already being used	Comment next to the email address comes up informing the user that the email address is already in use.	
3.ii	Player enters username that is already being used.	Comment next to the username informs the user that the username is already taken.	
3.iii	Passwords enters are not the same	Comment next to the password appears informing the user that the two passwords entered do not match.	
3.iiii	All fields entered by the user are valid.	Sign Up is complete and the user is logged in to the game with their new account	
Comments:			

Test Case ID: 2		**Test Purpose:** Log In	
Environment: OS X Google Chrome			
Precondition: Player has already signed up			
Test Case Steps: 3			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	Player goes to the website.	The first page is loaded	
2	Player clicks Log In.	The Log in page is loaded	
3.i	Player enters incorrect username and correct password and clicks sign in.	Message box appears informing the user that the username is not recognised.	
3.ii	Player enters incorrect username and incorrect password and clicks sign in.	Message box appears informing the user that the username is not recognised.	
3.iii	Player enters correct email address and incorrect password and clicks sign in.	Message box appears informing the user that the password is incorrect.	
3.iiii	Player enters correct email address and correct password and clicks sign in.	Player is taken to the logged in page.	
Comments:			

Test Case ID: 3		**Test Purpose:** Play Game against person	
Environment: OS X Google Chrome			
Precondition: Player is logged in			
Test Case Steps:			
Step No	Procedure	Response	Pass/Fail
1	Select Classic on the drop down mode list.	Classic is selected	
2	Click P v P	The system will take the player to the loading room and will wait until a player is found to play with. The system will then count down 3 seconds until the game starts.	
3	Enter a word in the text box that is corresponding to the word given and press enter.	If the two players match a correct word, the game will move on to the next word, otherwise the player will be prompted to continue entering words	
4	Player continues to play the game for 90 seconds		
5	After 90 seconds the game will end	The system will inform the player of their score and the player will be asked if they want to play again.	
Comments:			

Test Case ID: 4		**Test Purpose:** View High Scores	
Environment: OS X Google Chrome			
Precondition: Player is not logged in			
Test Case Steps:			
Step No	Procedure	Response	Pass/Fail
1	Player goes to the website.	The first page is loaded	
2	Player selects view leader board	The leader board page is loaded, the player can scroll down through the high scores.	
Comments:			

3.1.3 Use case

Below is the use case for the system I will be producing. The use case shows all the functions that should be available to the different types of users. I created the use case using Creately which is an online tool. I used this software instead of simply using Microsoft Words SmartArt features due to its increased design flexibility. Creately offers a much larger database of shapes and allows you to freely position them wherever you like on the page. Creately also offers easy access to the designs online allowing me to make changes to the designs on any device, without having to install a software by simply clicking on a link: https://tinyurl.com/wordifyusecase

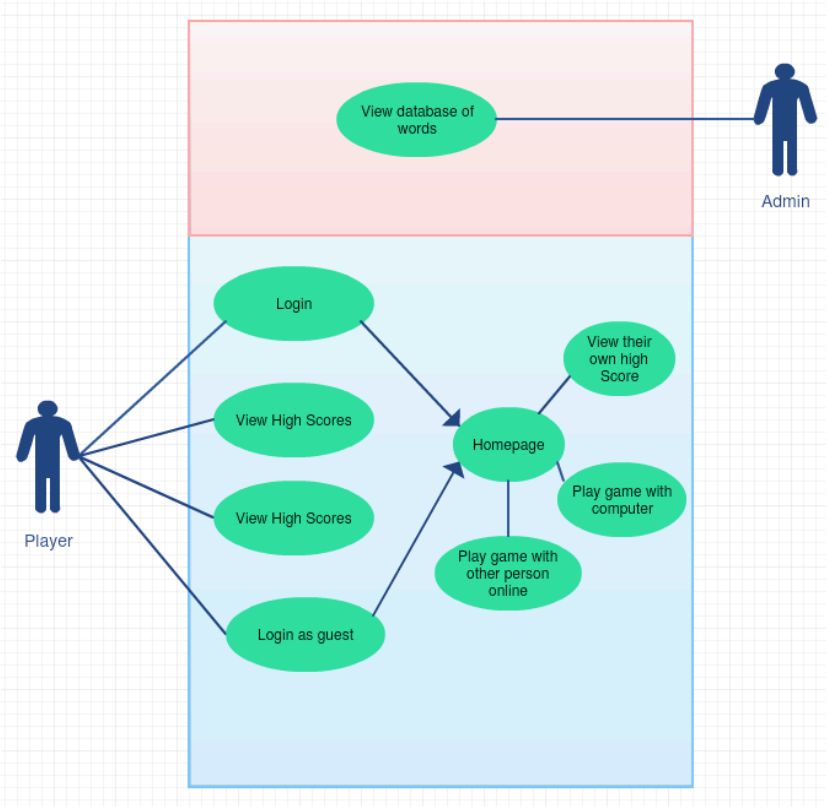

Figure 4 Use Case diagram, made on Creately

3.2 Design

I will now discuss the design of my system. My GWAP will follow the client-server model, meaning that all communication is between the client and my server. By doing this the security of the game is easier to control compared to the Peer-to-peer model which makes cheating very hard to prevent as there is no central server. One of the drawbacks of the client-server model is that the system will rely heavily on the stability of the server. Therefore, I will have to ensure that the server used for my system is as reliable as possible and can handle large amounts of requests.

3.2.1 User Interface design

As my game will be online, I will be utilising three main languages for the front end; CSS, JavaScript and Pug (formerly known as Jade). When choosing the code to use for developing the front end, I researched many different options. My initial idea was to code the front end in pure HTML, however it is not a dynamic language, making the website much more difficult to be responsive. HTML does not support the use of templates which makes it more time consuming to develop the website. I therefore decided to use a templating language, I decided upon implementing Pug, a template engine for JavaScript that acts as a middle man between the front and back end, making injecting data into the HTML easy by using simple commands. A similar alternative to Pug is Handlebars which I also have considered. The main aspect of Handlebars that I particularly appreciated is how it clearly separates the code from the HTML. Although I did like the clarity, I found much more documentation online for Pug. I also much preferred the clarity of the syntax used in Pug. Pug is much easier to read and maintain than pure HTML as it forces you to indent. It is also much more succinct, it doesn't need you to constantly open and close elements, simple indentation is enough. Below is a comparison between a simple HTML/handlebars webpage and it being coded in Pug.

HTML/Handlebars

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Jade vs HTML</title>
    <script type="text/javascript">
      function someFunction(){
        alert("hello");
      }
    </script>
  </head>
  <body>
    <h1>Which is better Jade or Handlebars?</h1>
    <div id="div_tag_id" class="div_tag_class">
      <p>Testing the two template engines</p>
      <p>
        Jade uses indentation and code minimal,
        Handlebars looks like normal HTML along with
        its curly brackets for expressions {{}}
      </p>
    </div>
  </body>
</html>
```

Figure 6: HTML/Handlebars example

Pug

```
doctype html
html(lang='en')
    head
        title Jade vs HTML
        script(type='text/javascript').
            function someFunction(){
            alert("hello");
            }
    body
        h1 Which is better Jade or Handlebars?
        #div_tag_id.div_tag_class
            p Testing the two template engines
            p
                | Jade uses indentation and keeps code minimal,
                | Handlebars looks like normal HTML along with
                | its curly brackets for expressions {{}}
```

Figure 5 Pug Example

JavaScript will be used for much of the backend of the system, however it will also be utilised on the front end, making elements in the webpage more responsive. I chose to use JavaScript as I have previous experience with utilising, therefore I was aware of the various functions that it offers which would require for my system.
I will use JQuery, a cross-platform library with JavaScript, to make the webpages more reactive. I will use it when validating data before submitting forms. The checks will ensure that the password and confirm passwords match, by keeping the this on the client side requests aren't being wasted by the servers checking this.

I want to ensure that the website is playable on mobile devices along with computers. One of the methods I will use to ensure this will be within the CSS. I will keep any sizes in percentages, so that they adapt with the screen size, whilst keeping the overall appearance of the game consistent which is important as people can easily transfer from playing from PC to mobile and still feel familiar with the game. I will use Bootstrap as the CSS which is a free to use front end framework [19]. It is responsive and professional looking which is highly beneficial as I have a limited amount of time to programme, it will help to speed up the process as it helps with the design.

3.2.2 Wireframe designs

Below are the designs I have produced for the front end of my system using Moqups. My interactive designs can be viewed by visiting: https://tinyurl.com/moqupsdesigns . I am using bootstrap for the base of the CSS in my website and Moqups offers an array of bootstrap elements making it look as accurate to the finished product as possible. This feature is not available in Balsamiq which is the other website that I considered when producing my products wireframes as I had previous experience using it. It is important to use bright colours to keep people engaged and excited, therefore I used orange shades as the main colour. I have decided to name the game 'Wordify'. I have chosen this name as it is simple and helps to describe to the users what the game does. Wordifys font that I have chosen is 'Luckiest Guy' which is not a default HTML standard font, therefore I will import it from google fonts. Although this was an initial concern, it turned out much more reliable than I had initially anticipated, the fronts compatibility can be seen below. Google Fonts are released under open source licenses [20]. This means that they can be used in any non-commercial or commercial project meaning I do not have to ask for permissions to use them.

Google Fonts Compatibility [21]

The Google Fonts API is compatible with the following browsers:

- Google Chrome: version 4.249.4+
- Mozilla Firefox: version: 3.5+
- Apple Safari: version 3.1+
- Opera: version 10.5+
- Microsoft Internet Explorer: version 6+

The Google Fonts API works reliably on the vast majority of modern mobile operating systems, including Android 2.2+ and iOS 4.2+ (iPhone, iPad, iPod). Support for earlier iOS versions is limited.

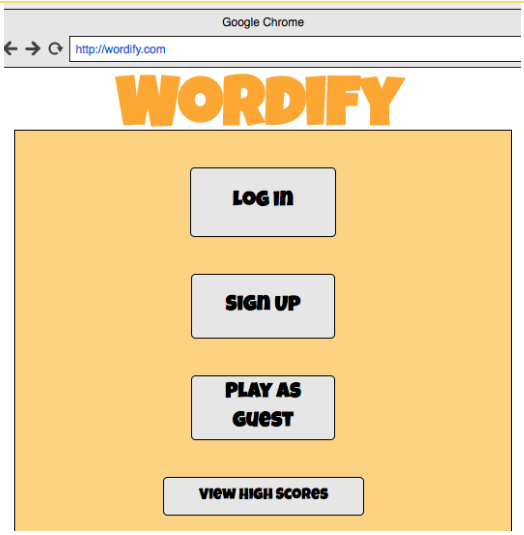

Figure 7 First Page created on Moqups

First Page

This is the page that the user is shown when they first visit the website. The font size on all the buttons on this page is 22pt and the font is luckiest guy. This font will be used throughout the entire game. I chose this font because it is a fun friendly looking font which makes the game seem more exciting.

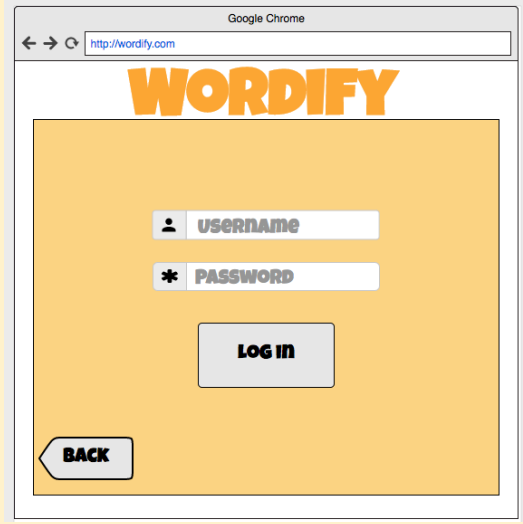

Figure 8 Log In page created on Moqups

Log In

When the user clicks login they are taken to the log in page. All the font on this page is 22pt. When the user clicks login they are taken to the logged in page and if they click back they are taken to the First Page.

Figure 9 Sign Up page created on Moqups

Sign Up Page

When the user clicks signup from the first page they are taken here where they enter their credentials, and click sign up. Once the user clicks sign up they are logged in and taken to the logged in page.

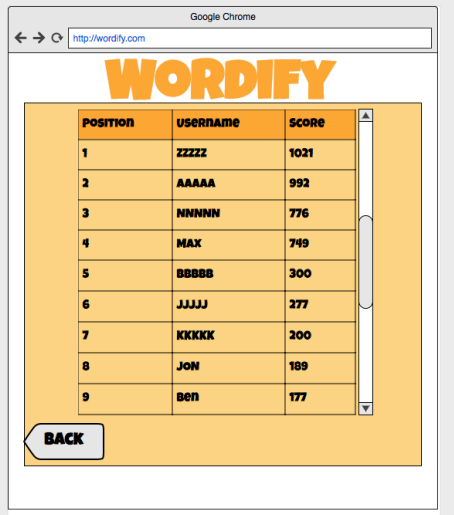

Figure 10 Not logged in leader board page created on Moqups

Not logged in leader board

When the user goes to the leader board from the first page they are shown this page. The font size in the table is 22pt and the font size on the button is 25pt.

Figure 11 Rules Page created on Moqups

Rules

Clicking on the question mark in the corner of the webpage shows the rules of how to play the game.

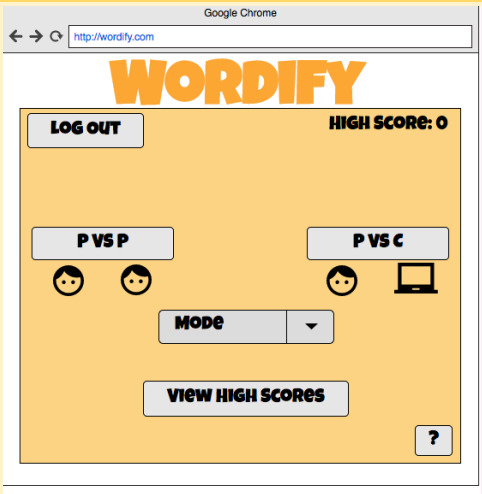

Figure 12 Logged in page created on Moqups

Logged in

Once the user has logged in or clicked play as guest, they are taken here. This is where the user can choose what type of game they play. They can either play against another player or against the computer. They can they choose the mode of the game by using the dropdown list.

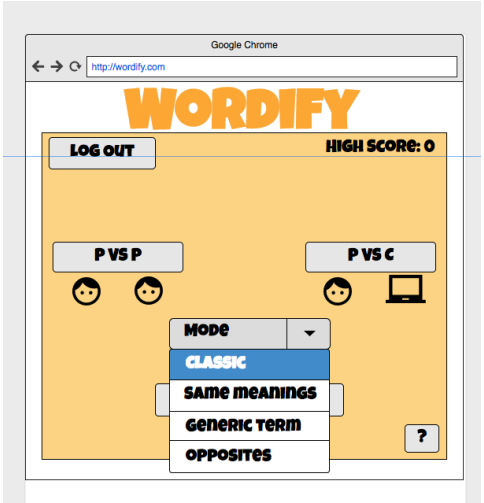

Figure 13 Selected dropdown list page created on Moqups

Selected dropdown list

The options the user can choose from on the dropdown list is either classic, same meanings, generic terms or opposites. I have used these words instead of synonyms antonyms and hypernyms because they are easier to understand, this is especially important for people who are trying to learn a new language as they may not be familiar with those lesser common words.

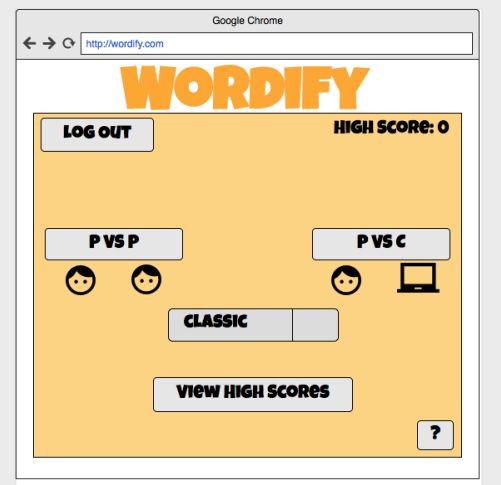

Figure 14 Classic selected page created on Moqups

Classic selected

The classic game mode is a combination of all the modes; synonyms, antonyms and hypernyms.

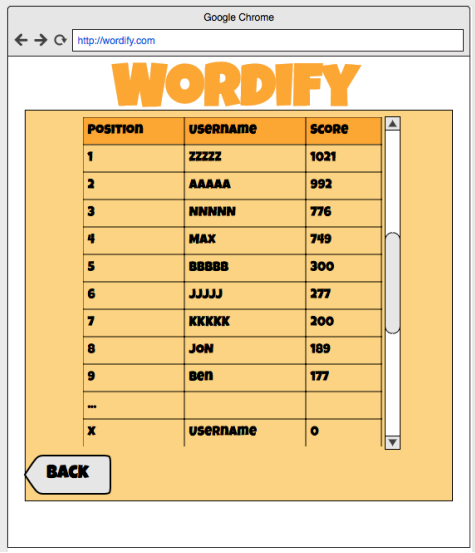

Figure 15 Logged in leader board page created on Moqups

Logged in leader board

Once logged in, the user can go to the leader board and see what position they are in on it. This element is exciting for the user as they can see their position go up as they increase their high score.

Figure 16 Searching for players page created on Moqups

Searching for players

If the user selects p v p they are taken to this page where players are matched with other players that are waiting for games. If they are waiting for too long the user can click on the back button and they are taken back to the logged in page.

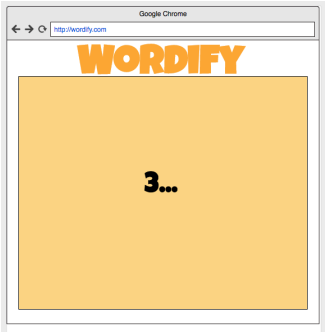

Figure 17 Countdown page created on Moqups '3...'

Countdown

If the player selects the p v c mode or once they have found a partner on p v p page they are taken to the countdown which gives the user 3 seconds to prepare before the game starts. This important as the players must be ready before the game starts.

Figure 18 Countdown page created on Moqups '2...'

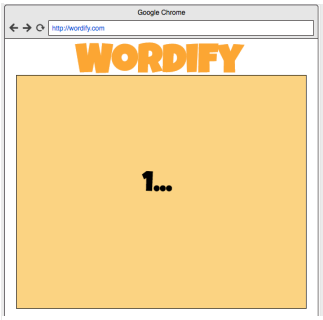

Figure 19 Countdown page created on Moqups '1...'

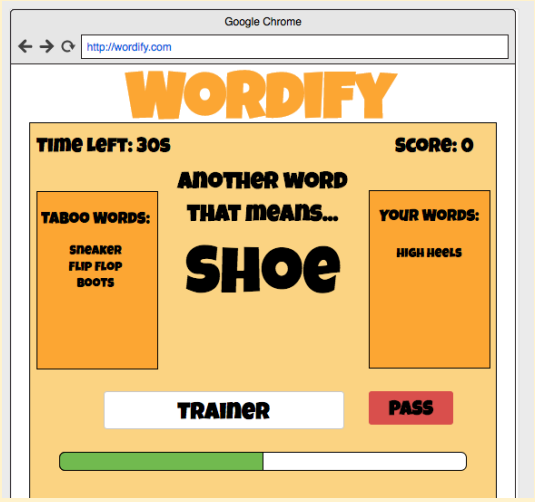

Figure 18 Synonym game page created on Moqups

Synonym game

When the user is asked to give a synonym they are asked to give 'Another word that means…'. If they fill the green bar below which takes 14 questions, they get 1000 bonus points.

By clicking pass, players can move on to the next word. The taboo words are shown are words that players are not allowed to use. When a user enters a word it appears in the 'Your Words' list.

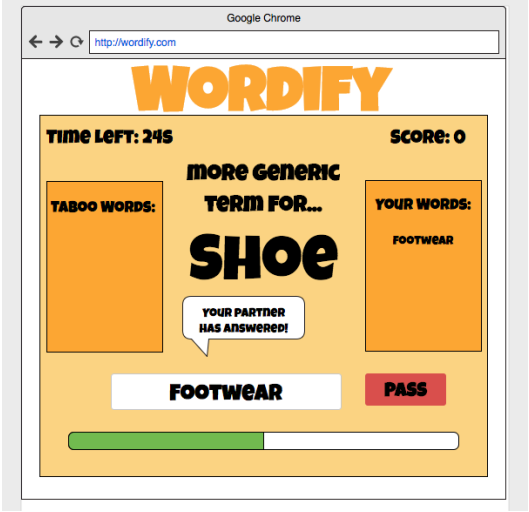

Figure 19 Hypernym page showing player has answered created on Moqups

Hypernyms

When the user is asked to give a hypernym, the game asks for a 'more generic term for' the word. Another element in this page is the message box informing the player that their teammate has entered an answer already. This element was used in Von Ahns game with a purpose and resulted in a 4% increase in game play [6].

Figure 20 Antonym game page created on Moqups

Antonyms

When the user is asked to give an antonym, they are asked to give the 'opposite of' the given word.

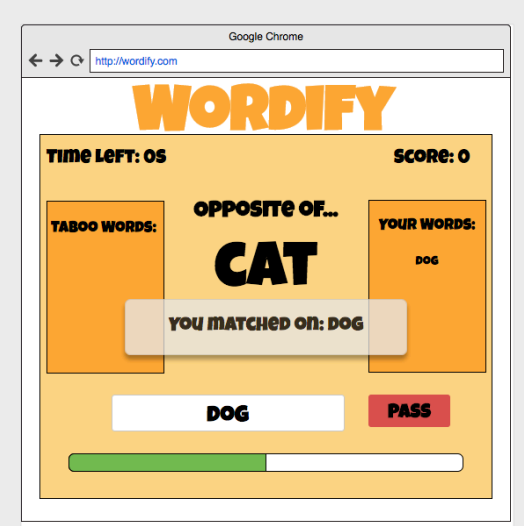

Figure 21 matched word game page created on Moqups

Matched Answer

This is what will be shown when the pair make a match. The message box will appear for 1 second and then the game will move on to the next word.

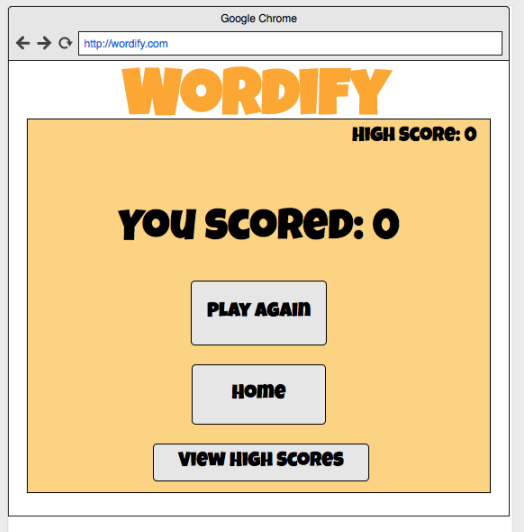

Figure 22 End game page created on Moqups

End of Game

At the end of the game, the user will be shows their high score and will be given the options to either play again, return to home which is the logged in page or view the high scores.

3.2.3 Backend design

I explored a variety of options when considering what language to use for the back end of my project. One of my initial ideas was to code the backend in Python as I am familiar with Python and packages built for WordNet were supported in Python [22]. The other option I considered was to implement the backend in JavaScript. JavaScript is a great language for web based projects because it can be efficient to code the front-end and back-end in the same language, as well as it being lightweight and supportive for data-intensive applications. Below is a comparison between Python and NodeJS. I used this table to help visualise the benefits of both languages and come to a decision over which code to utilise for my project.

Language	Speed	Functionality	Code
	Using the testing from Yan Cui, Principal Engineer at DAZN, I can compare the speeds of both languages from a variety of tests. [23]		
Python	Okay	Good - Python offered a suitable package for WordNet that offered all the functionality needed.	Great – Python is renowned for its codes clarity and simplicity [24], its syntax forces you to keep it neat and tidy.
NodeJS	Great	Great – Along with a suitable WordNet package, packages could be used for API routing and socket connections.	Okay- There is no strict clean coding rules when using NodeJS.

Figure 23 Comparison between Python and NodeJS

After researching the two languages, I have decided that the best option for this project is NodeJS. Another benefit of implementing JavaScript is its popularity. JavaScript is one of the most popular languages used [25] on the internet today. NodeJS is a 'a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine).' [26] This means that there should be plenty of online documentation to aid me during the development of the system.

I will be using Express, a NodeJS web application framework used for designing and building web APIs. Express allows me to receive HTTP requests at certain paths and process these requests to perform different actions and respond with different content. Request parameters and session cookies can be sent to my server which Express can process, allowing for tailored responses unique to different users. Middleware in Express can be used to authorize requests before processing them, for example checking if the user is authenticated before allowing access to a particular API. Express allows for different formatted responses, for example, plain-text, JSON and HTML - this allows me to create powerful APIs for my web application.

Navigation

When the user navigates through the website, I want to keep the address bar the same as in my opinion it looks more 'game like' , I also wanted to stop the page from completely reloading every time the user does an action. In order to do this, I will utilise Ajax requests which will allow me to send and receive data from the server asynchronously and update the page without inferring with the display. When navigating, an Ajax request will be made to receive a formatted version of the page that is customised based on the session information retrieved from that user. I will directly render pages dynamically using Express with my Pug templates. This method offers allot of flexibility that you wouldn't have get with static HTML. This is vital for a system like mine where the game requires a lot of different changing varying data.

Sockets

I will be implementing sockets into my game to speed up the communication between the client and the server. I have decided to use sockets as the speed at which messages can be sent between client and server are significantly faster than ajax requests [27]. This was vital for game play as every second counts when the user sends a guess to the server as the player has a finite amount of time when playing a game. When deciding upon what library to use, I considered two options; SockJS and Socket.io. Although SockJS is an attractive library which works on all modern browsers and in environments which don't support the WebSocket protocol [28] it is relatively young compared to socket.io and has far less documentation online. As this is my first time implementing sockets, I have decided to utilise socket.io due to its extensive documentation and online community. I will use Socket.IO which is a JavaScript library for real time web applications sockets into the game. By applying Socket.IO, implementing sockets into my game is sped up. This is vital for a project like this where timings are limited. When a user logs in to the game, a socket connection will be made. According to its website socket.io is "One of the most powerful JavaScript frameworks on GitHub, and most depended-upon npm module." [29] .

Coding designs

Below are basic designs of some of the most prominent functions that will be utilised in my GWAP. By creating the basic structure of the functions, implementing the system should be much faster.

Sign Up/Log in:

When the user logs in or signs up I wanted to ensure it is as secure as possible. I will be implementing it with PassportJS which is an easy to use, flexible authentication middleware for NodeJS. I also researched express-authentication which is an 'Un-opinionated authentication for express; an alternative to passport.' [30]. Although I liked this package, there is little documentation online about implementing the library which may make debugging issues much more difficult. I was also aware that it hadn't been updated very recently which means it may not be compatible on the latest version of NodeJS which I will be using, or it may contain bugs I may find when implementing it. There is an abundance of documentation online for passportJS [31] which is one of the reasons why I have chosen to implement this library [32]. One of the benefits of using passport is that it supports a wide range of authentication platforms, it supports local authentication and it is also compatible with Google and Facebook, which I could incorporate into the game at a later date. This save time for users when they sign up or log in as they would simply have to click log in with Facebook, meaning we may have more people doing it as it is much less hassle. This also means I could potentially incorporate a 'friends' system into the game, or use their data in other ways. I have utilised the passport 'local-login' feature which allows a customisable authentication system suitable for Wordify that allows users to sign up and authenticate with a username and password. [33]

When signing up, various checks will be made before the new account will is created. On the client side, JavaScript and JQuery will be used to check that the password and confirm passwords match they will also check that all other fields have been filled in. Its beneficial to do this on the client side as it saves the server having to do this which takes up more time and puts needless strain on the server. On the server side, the server will check to see if the username or email address already exists, if it doesn't then it will create a new user and hash the password using bcrypt. It is important that the password is never stored as plaintext on the server and a very secure hash function, such as bcrypt, is used so that if the database is ever compromised the credentials of the users are never fully revealed. I have chosen to use bcrypt because it is a highly-recommended hashing function used by major websites, such as Twitter. If the username does already exist the user will not successfully sign up and will be informed using an information window.

Matchmaking

When a player clicks on the P vs P button the system will begin matchmaking them with an opponent. To match the players together the 'findRoom' function is called. The function checks to see if there are any rooms of the same game mode that they have selected with only one player in them. If there are, they will join the available room and the server will tell their opponent that another player has joined. Once two players are in the same room, they will both save each others userIDs which will be used to send messages to one another using the socket.io emit functions. The countdown to the game will then begin. If there are no available rooms, the server will create a new document in the room schema with the clients ID and will send the client to the waiting room where they will wait until an opponent has joined the game. When two players join a room, they have 60 seconds to play as many games as possible. Each game consists of a keyword and taboo words if there are any. Once two players match on a correct word, they move on to another game.

```
Function joingame(gametype){
	If(availableroom(gametype)){
		Join(room)
		Start(game)
	}else{
		createnewroom(gametype)
	}
}
```

Room Countdown

There are two countdowns that will be used for each room. The first countdown will be to countdown 6 seconds until the first game begins and the second timer will countdown 60 seconds from when the first game begins. Both timers will use the same countdown function which will use JavaScript.
The countdown will use a JavaScript function 'setInterval' which calls a function at specified intervals until certain conditions are met. The function will minus one from the 'timer' every second and will display the seconds left in the specified element ID. Once the current date is greater than the timerdate, either the 'clearinterval' and 'endGame ' or 'startRoom' functions are called, depending upon what countdown is being shown.

Create Game

There will be a function called getRandomWordFromKeywordsSchema() which will be used to generate a new game and select a random word for it. This function will be called after two players match on a correct word, if a player passes on a word or just at the beginning when the whole game starts. The function will return the new keyword, its taboo words which are word that cannot be used to create a match and will also update the current score.

Check guesses

The check Guess function will go through several checks each time a user submits a guess. The checks will be done as shown below in the specific order. It's important to make the checking as efficient as possible as this function will be called each time a player makes a guess. Along with checking if the two words entered by payers actually match, we also have to check for incorrect matched to prevent cheating in the game. One of the steps I have taken to ensure the function is as light possible for the server is that I have left utilising WordNet to the last step. This is because searching WordNet is expensive as it is such an extensive database of words. As of December 2017, there are 147,278 words inside the database [34]. Therefore, it will be used as a last option when checking if the word match is correct.

1. Add the guess to the games guesses field.
2. Has the opponent already guessed this word?
    a. Yes: Move on to number 3
    b. End check
3. Is the match a taboo word?
    a. Yes: End check
    b. No: Move on to number 4
4. Have the words matched over the correct match level
    a. Yes: It's a Correct match!
    b. No: Move on to number 5
5. Is the match correct with WordNet?
    a. Yes: Its correct Match!
    b. No: End Check

WordNet

WordNet is a free to use [35] large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept [36]. One of the most notable aspects of WordNet is its extensive knowledge structure of relational words. Below is a visualisation of the WordNet structure, produced by Dr Jaap Kamps.

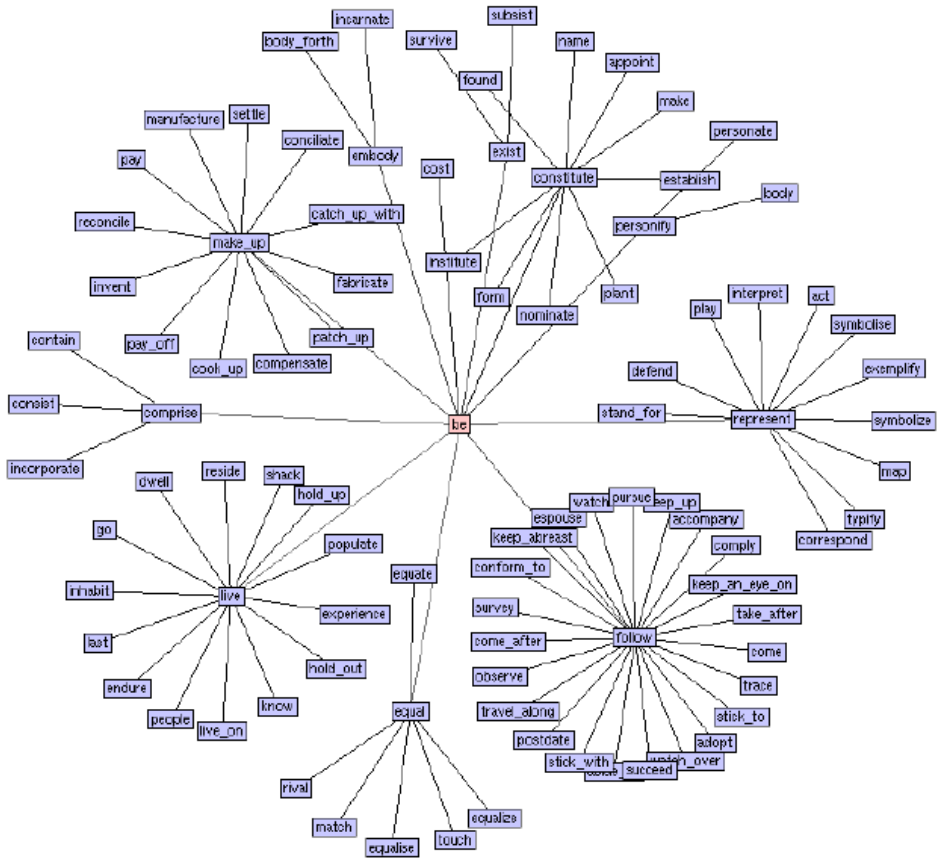

Figure 24 The WordNet database from the vista point of verb 'be' and Maximum Path Length of 2. [37]

I will be using an API to search WordNets database, this will help save a lot of time during the implementation of my system, which is needed for this project with such a tight time schedule. The package that I have chosen is 'WordNet-Magic' [38] .

This can be implemented directly into my project by installing it as a Node package. WordNet-Magic can be easily utilised in node.js, which is one of the many reasons why I have chosen this language.

3.2.4 Database design

When designing my databases structure, I looked at all the possible routes and came to a decision based upon the pros and cons of each option. I initially expected that I would use MySQL for my databases because it is generally the default database for web-based applications [39]. However, upon further research I discovered the benefits of implementing a MongoDB system. It is schema free, giving my system the freedom to adapt over time [40]. For example, if I want to add a new game mode to the game it would be easier to implement it in MongoDB over MySQL by defining new models rather than creating a new table. I also chose MongoDB as it is very compatible with node.js which is the language I am using for the backend of my system. This is thanks to mongoose which is a MongoDB object modelling tool. Another benefit of MongoDB over MySQL is speed. Speed is vital in a game like Wordify where there is a timer, meaning every second counts when evaluating the players guesses. According to MongoDB, their customers see performances improve by up to 6X when moving from MySQL to MongoDB [41] [42].
Below are the designs I have created for my databases. It is important to design the databases thoroughly before implementing them as it speeds up the making process and ensures that there are no unexpected issues that may arise in the future. The game I am making is very memory exhaustive as all the matches made need to be recorded. Therefore, it was essential that I made the system as efficient as possible. I needed to ensure that all the data that is stored was really needed and that there was not any unessential data being sorted which uses up precious memory.

Users schema

This schema will store all the users basic information including email address, username, password (hashed using bcrypt) and their high score.

Username	Password	Email	High Score
Text, maxlength=20	Text, maxlength=32	Text, maxlength=34	Number, min=0

Below is a basic structure of my users model, in NodeJS using mongoose

```
const userSchema    = mongoose.Schema({
   username     : { type: String, maxlength: 20 },
   password     : { type: String, maxlength: 32 },
   email        : { type: String, maxlength: 34 },
   highscore    : { type: Number, min: 0 },
});
```

Room schema

This schema will store all the rooms that are created for the games. A room is created when a player searches for a room and cannot find one with only 1 client in it that hasn't started yet.

Clients	Mode	CreatedAt	StartedAt	EndedAt
User Reference Array	String	Timestamp	Timestamp	Timestamp

Below is a basic structure of my model, in NodeJS using mongoose

```
const roomSchema    = mongoose.Schema({
   clients      : [ { type: mongoose.Schema.Types.ObjectId, ref: 'User' } ],
   gamemode     : { type: String },
   createdAt    : { type: Date },
   startedAt    : { type: Date },
   endedAt      : { type: Date },
});
```

Game schema

This schema will store all the individual games. A game is a round within a room. Each game document will store the keyword and the guesses that have been entered by the clients in the corresponding room.

Room	**Keyword**	**Relation**	**Guesses- Array**	
			Player	Guess
Room Reference	String	String	User Reference	String

Below is a basic structure of my keywords model, in NodeJS using mongoose

```
const gameSchema    = mongoose.Schema({
   room         : { type: mongoose.Schema.Types.ObjectId, ref: 'Room' },
   keyword      : { type: String },
   relation    : { type: String },
   guesses      : [{
                    player: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
                    guess:  { type: String },
               }]
});
```

Matched words schema

This schema will store the keywords with their matched words, their relation (synonym, antonym or hypernym) and the number of times they have matched.

Keyword	**Matched Word**	**Relation**	**Number of matches**
Text, maxlength=20	Text, maxlength=20	Text, maxlength=1	Number, min=0

```
const matchedWordsSchema    = mongoose.Schema({
   keyword    : { type: String, maxlength: 20 },
   matchedWord    : { type: String, maxlength: 20 },
   relation    : { type: String, maxlength: 1 },
   numberOfMatches    : { type: number, min: 0 },
});
```

Entity Relationship Diagram

I have created the entity relationship designs using Creately which is a free software that can be used online, making it easy to make edits to the diagram. By creating the diagram, I have been able to visualise the system and make sure there are no areas that I have missed out on before moving on to the implementation of the system. The diagram should also make producing the actual system much easier as when checking for errors as I can clearly see the structure of the Databases.

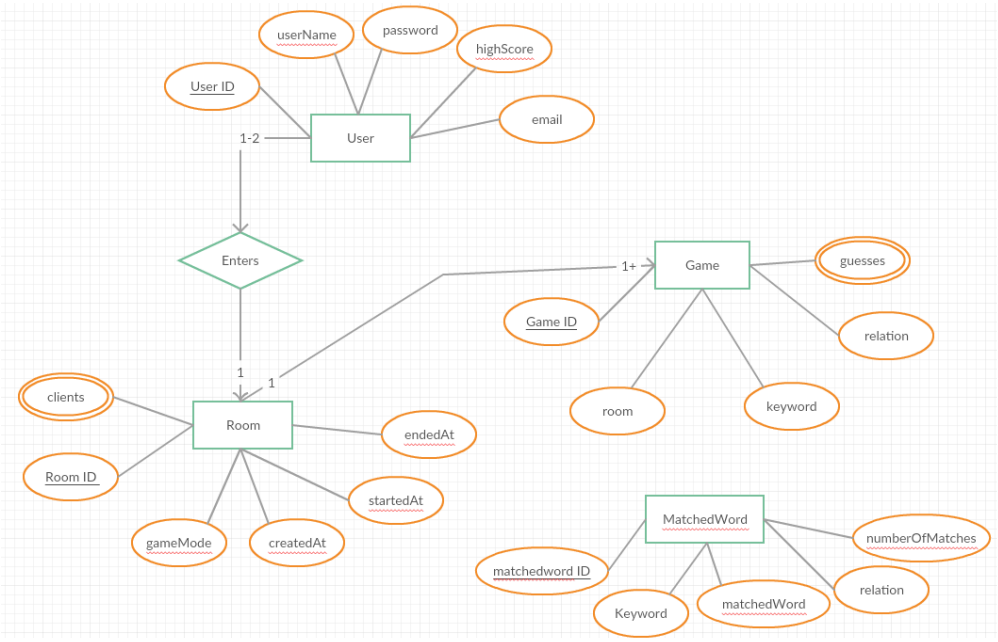

Figure 25 Entity Relationship Diagram for Wordify

4 Implementation

I am now on the third stage of the waterfall model, the implementation of the system. To ensure a successful system, I will attempt to follow the designs I have as closely as possible. I will now discuss the stages of the implementation of my game, examining the different issues and decisions I faced. I will also inform the reader of how I overcame them, justifying my decisions with reliable sources.

One of the first decisions I had to make when implementing the system was what text editor to use. When considering what text editor to use for the project I assumed I would use sublime text editor as it is my usual go to editor. I like its simplicity and its extensive library of importable packages. However, for this larger project, I decided to research alternatives and discovered Visual Studio Code. I hadn't used Visual Studio Code before, but from my researched it seemed most suitable for my requirements. One benefit the text editor offers is its Git integration [43] which would save me much time when making changes to the system as I simply had to click a button and my project would be committed to GitHub. I also liked the in-editor debugger [44] which made debugging much simpler.

I utilised GitHub in many ways throughout this project, its main benefit was the fact that you can track the changes made in files [45], meaning if there was an error, you can find where you added it and revert to the original code. Another way I benefitted GitHub was backing up any of my work. This was vital as if any of the files on my computer were lost or corrupted I always knew I had a backup on my GitHub respiratory, giving me peace of mind. Another notable benefit that GitHub will give me during the project is accessibility to my code, wherever I was I could always access it, this proved useful over Easter when I am away from home as I can still access my project.

When developing the website, I used my local server to host the site. This made making changes easy as I only needed to compile and run it straight from my local machine. I decided to do this rather than using a 3rd party host such as Heroku because it was much quicker to compile and host when running and editing code. I could see the results much quicker. It was also much easier as I could use debugging tools and view compilation messages and debugging outputs.

4.1 Frontend

The first stage of the implementation was to produce the front end. I found developing the web pages in pug very easy and hassle free thanks to its simple syntax. I initially produced a container which is the base of all the other pages which can be seen below.

```
html
  head
    title Wordify
    header
        h1(class='text-center', style="font-family: Luckiest Guy; color:#ffa834; font-size:500%")
Wordify

    include includes/css/core.pug

  body(style="font-family: Luckiest Guy")
    div.container(style='background: #ffd783; max-width:550px; border: 3px solid #000000;
height:500px;' id='container')
        block container

    include includes/js/core.pug
```

Due to the limited time I had for the project, I decided to keep the game consistent regardless of the devices used. Instead of making the website completely responsive, I used percentages instead of sizes, to that the game fitted each screen size whilst keeping its original proportions.

4.2 Backend

I will now discuss the implementation of the backend of my system. To help understand the structure of the system I have produced descriptions of each file in my project which can be seen below.

GWAP/ :

/app :

/models :

/game.js : The game schema for MongoDB

/index.js : This is the entry file that initialises all models

/Keywords.js : The Keywords schema for MongoDB

/matchedWords.js : The matched words schema for MongoDB

/room.js : The room schema for MongoDB

/user.js : The user schema for MongoDB

/public : All the files publicly available

/js :

/main.js : This is the main JavaScript file used by the front-end client

/routes :

/index.js : This file contains all the routing for the application and handles all server requests

/sockets :

/index.js : This file handles all socket connections and socket requests made by the client

/views : Stores all of the templates used as the views of the game

/includes :

/css :

/core.pug : A core pug file which initialises all css files for the application

/js :

/core.pug : A core pug file which initialises all JavaScript files for the application

/blank.pug : This is the basic structure of the website, every other page is contained by blank.pug

/countdown.pug : The countdown page before a game starts

/endgame.pug : The view shown once a game has ended

/game.pug : The view shown during a game

/home.pug : The page shown when you click back if you click log in or sign up or view high scores, It is the same as /index.pug apart from it not extending blank.pug (otherwise you have two blank.pug views inside one another .).

/index.pug : The page shown when you visit the game, not logged in.

/leaderboard.pug : The page that displays the high scores

/loggedIn.pug : The homepage, shown when the user logs in

/login.pug : The log in page

/rules.pug : This page shows the rules for playing the game

/searching.pug : This is the page that is shown to the players when they are searching for an available room

/signUp.pug : The sign up page

/config :

/database.js : This file contains the MongoDB connection configuration information

/passport.js : This file contains the PassportJS functions which handle authentication and registration

/index.js : This is the main entry point for the application which starts the server, it also initialises the database, socket handler, routing and other requirements for the server to run.

4.2.1 Challenges faced & points of interest

For most of the back-end implementation I followed my initial designs closely. However, there were some adaptations that I made to the project for various reasons. I will now discuss notable challenges I faced whilst working on the backend of the system, along with the solutions I devised and the reasoning's behind them.

Sockets

Below are some examples of how I implemented sockets into Wordify using the JavaScript library Socket.IO. I found implementing socket.io into the project simple thanks to the libraries easy to use functions.

Client Side examples

```
function createSocket() {

    socket = io();
    socket.on('tabooWords', function(taboowords){});

    socket.emit('pass',{ game: gameId, room: roomId }
```

Server Side Examples

```
module.exports = function(io) {
    io.on('connection', function (socket) {
        console.log("Connected");

        if(socket.request.session.passport.user) {
            socket.userId = socket.request.session.passport.user;
            console.log("userid: "+ socket.userId);

            socket.on('findGame', function (data, t) {});
            socket.opponent = clientId;
            client.opponent = socket.id;
            client.emit('joined', t, room.id);
//tells the players we have joined and also sends t, the start time for the game.
            socket.emit('joined', t, room.id);
```

Viewing the High Scores

To view high scores, I utilised Express to create a 'highscores' API route, when requested it returns an array of the top highscores retrieved from my database. The client side code then loops through the high scores, adding a new row to the high scores table.

Client side:

```
$.ajax({
        url: '/highscores',
        type: 'GET',
        success: function(data){
            top10scores=data.highscores;
            for(var i = 0; i < top10scores.length; i++){
                var table = document.getElementById("highScoreTable");
                var row = table.insertRow(i+1);
                var position = row.insertCell(0);
                var username = row.insertCell(1);
                var tableplayerscore=row.insertCell(2);
                // Add some text to the new cells:
                position.innerHTML = i+1;
                username.innerHTML = top10scores[i].username;
                tableplayerscore.innerHTML=top10scores[i].highscore;
            }
        }
    });
}
```

Server side:

```
    router.get('/highscores', function(req, res){
      Users.find({}).sort({highscore: -1}).select('username highscore -_id').limit(10).exec(
        function(err, topUsers) {
          return res.json({highscores:topUsers});
        }
      );

    });
```

P vs P

P vs P involves playing a game of Wordify against another human player online. I will now discuss any functions that I had not planned during the design section or ones that I have changed due to unforeseeable reasons.

Game Selection

Once a game has started, a word must be chosen by the server and sent out to the players. This is what the getRandomWordFromKeywordsSchema() function does. It checks to see if the mode is random or not. If it is, the server selects a random mode and then continues. The system then selects a word from the database with the same relation as the mode and sends it to the clients via the socket connection. Along with this, the function completed many other tasks that I had initially not considered, including updating the score, collecting the taboo words, and sending the information to the user about the previous game, if there was one, such has their opponent passed or what word they had matched on.

By including taboo words in the game gives an added challenge to the player. It also helps to flesh out the data collected by the game as it forces players to think of the less obvious answers rather than the already well documented answers in the system. When a new game is started. The list of taboo words is produced by the server. It comprises of any matched words to the current keyword that have been matched over 20 times. (This number can be changed as the game scales up).

```
function getRandomWordFromKeywordsSchema(mode, roomId, roomEndedAt, score, previousgametype ,guess){
    console.log("Game mode: "+ mode);
//if the mode is random, choose a random mode
    if(mode=="R"){
        var modes = ['S', 'A', 'H'];
        mode=modes[Math.floor(Math.random() * 3)];
    }
    Keywords.count({mode:mode}).exec(function (err, count) {
    // get the number of keywords and get a random number between 0 and the count
        var random = Math.floor(Math.random() * count)
    // query all keywords, fetch  the 'random'th keyword
        Keywords.findOne({mode:mode}).skip(random).exec(
            function (err, keyword) {
                var newgame = new Game({room:roomId, keyword:keyword.keyword, relation:mode});
                newgame.save(function (err) {
                    if (err) return console.error(err);
                });

                var gameWord = keyword.keyword.toLowerCase();
                var client = io.sockets.connected[socket.opponent];
                console.log(previousgametype);
                if(previousgametype=="newgame"){
                    var firstWord= true;
                }else{
                    var firstWord=false;
                }
                socket.emit('newword', newgame.keyword, newgame.relation, newgame.id, roomEndedAt, firstWord);
                client.emit('newword', newgame.keyword, newgame.relation, newgame.id, roomEndedAt, firstWord);
//tells the players that there is a new word, what the word is and its mode.
                console.log("Create Game: " + newgame.keyword);
                socket.emit('scoreUpdate', score);
                client.emit('scoreUpdate', score);
                MatchedWords.find(
                    { keyword:keyword.keyword, relation:mode, numberOfMatches:{ $gt: taboowordlimit } }).exec(function(err,
matchedwords) {
                    if(matchedwords){
                        var taboowords=[];
                        if(matchedwords.length>0){
                            for(var i = 0; i < matchedwords.length; i++){
                                amatchedworddocument=matchedwords[i]
                                taboowords.push(amatchedworddocument.matchedWord)
                            }
                            socket.emit('tabooWords', taboowords);
                            client.emit('tabooWords', taboowords);
                        }else{ console.log("there are no taboo words!")}
                    }else{ console.log("there are no taboo words!")}
                });
                //if players passed on the previous word, tell them
                if(previousgametype=="pass"){
                    socket.emit('youPassed');
                    client.emit('opponentPassed');
//tells the players if there has been a match on the previous word, and the word the match was on
                }else if(previousgametype=="guess"){
                    socket.emit('match', guess);
                    client.emit('match', guess);
                }
            }
        );
    });
}
```

Quitting a game

It was important to give the players a means of quitting their game whilst in play. This is because if they were paying against another person online who wasn't playing properly, they won't want to waste their time playing a pointless game. When the user quits the game, they leave the game and their opponent is notified that their opponent has left the game. They are then also subsequently also pulled out the game. To do this I simply call the endgame function early and send both players to the logged in page.

Passing on a word

If players are stuck on a word they can pass on it by clicking the pass button which simply tells the server to get a new keyword and take away 50 points from the players. When a player Clicks on the pass button, they emit "pass" via the socket connection with the server which calls the following function:

```
socket.on("pass", function(data){
    var gameId=data.game;
    var roomId=data.room;
    Room.findById(roomId, function(err, room) {
        var roommode=room.gamemode;
        var currentScore=room.score;
        var newScore=currentScore-50;
        room.score=newScore
        var opponent=room.opponent;
        room.save(function (err) {
            if (err) return console.error(err);
        });
        if(opponent.toString()=="5ac6638f0501800014622195"){
            console.log("the player passed against computer")
            getRandomWordFromKeywordsSchemaVSComputer(roommode, roomId, room.endedAt, newScore,
"pass")
        }else{
            var client = io.sockets.connected[socket.opponent];
            getRandomWordFromKeywordsSchema(roommode, roomId, room.endedAt, newScore, "pass");
        }
    });
});
```

P vs C

Users also have the choice to play against their computer instead of another player online. My allowing players to play against a bot, even if there is no one searching for a game at a given time, a player can still play a game. This is one of the steps I have taken to ensure that the game is as accessible as possible. I have created a basic model which in the future can be tweaked to make it play more realistic. The computer should also become more realistic as more data is gathered by players.

During the planning phase of the project. I hadn't put much thought into how I would construct the bot player. The solution I realised was to have a 'bot' account, the same as if you were to create a normal user account. By doing this it made implementing the P vs C much quicker as much of the P vs P functions only had to be slightly adapted to accommodate for the new game mode.

Starting a game

Starting a P VS C game is a much simpler version of creating a P VS P game. It creates a game with the player and the computer account, and sets the start time and end time for the game. Once room is created, the client is sent the room id and the time that the game starts. The server waits 7.7 seconds until it calls the 'getRandomWordFromKeywordsSchemaVSComputer()' function which gets the first keyword of the game.

```
socket.on('computerGame', function (data, t) {
    gameMode=data.mode;
    var t = new Date();
    var startTime=new Date();
    var endTime=new Date();
    var compID=mongoose.Types.ObjectId("5ac6638f0501800014622195");
    var endTime=endTime.setSeconds(startTime.getSeconds()+69);
    t.setSeconds(t.getSeconds() + 7);
    var noDate = new Date(0);
    var newroom = new Room({ gamemode: data.mode, opponent : compID, host : socket.userId, createdAt:
startTime, startedAt: t, endedAt: endTime, score: 0 });
            newroom.save(function (err) {
                if (err) return console.error(err);
            });
    socket.emit('joined', t, newroom.id);
    setTimeout(function(){getRandomWordFromKeywordsSchemaVSComputer(data.mode, newroom.id,
newroom.endedAt, newroom.score, "newgame")},7700);
});
```

Computer guesses

Every 2.5 seconds there is a 60% chance that the computer will make a guess. I initially set the frequency to 4 seconds, however the games were very slow. By adjusting the intervals to 2.5 the guesses were being made at a more human like rate. A guess is called by utilising the 'setinterval' function with an interval of 2500 milliseconds between each recall. As more people play the game, I plan to adjust the figures to try get the best computer player which plays as similar to a normal human player as possible.
Guesses made by the computer are chosen by searching for the matched words in the matched words database and selecting one at a random that has the same keyword and relation as the current word. If there are no words that have been matched with the given keyword, the computer will pass. Every time a guess is made, the same 'checkGuess' function is called as if it was a normal human player to verify if there is a match and if the match was valid.

Software versions

Below is a table of all the packages that are used in my system, along with the versions, and their purpose for being implemented into the system.

Package type	Version	Where it was used
bcrypt-nodejs	0.0.3	Used for encrypting users' passwords before storing them in the database.
body-parser	^1.18.2	Parses all the requests made to the server used with express.
connect	^3.6.6	A HTTP framework for NodeJS to handle requests, sessions and cookies.
connect-flash	^0.1.1	Used with passport to send messages to the client such as log in failed.
connect-mongo	^2.0.1	Used to make the connection to the MongoDB server.
cookie	^0.3.1	Used by passport to check if a user is already logged in by checking their cookies.
express	^4.16.2	A web application framework for NodeJS used for routing.
express-session	^1.15.6	Used to create and handle web sessions when using Express.
express-ws	^3.0.0	A web socket endpoint for Express applications used for routing of sockets.
http	0.0.0	Used to create a HTTP server for the NodeJS application.
mongoose	^5.0.3	Used to bridge the MongoDB database and NodeJS.
passport	^0.4.0	An authentication library for NodeJS.
passport-local	^1.0.0	Allows local authentication and registration within the application.
pug	^2.0.0-rc.4	A template engine for NodeJS used to create web templates for the front-end.
session.io	^1.0.0	Used for socket authentication and session information between socket connections.
socket.io	^2.1.0	A socket handling module for NodeJS used for socket connections and messaging.
wordnetjs	^0.3.0	Used for searching WordNet database when verifying if a match is correct.

Figure 26 Table displaying the package versions used in the system

4.2.2 Database

I used mongo as my database program which gives me more freedom with changes and functions as its schema free. This allowed me to make changes to the structure of the databases without having to completely restart the databases. This was crucial this project as I was constantly making changes thought the project due to new issues that I had not thought about during the design stage.
When working on the backed, there were some aspects of the database that I hadn't considered. One aspect was with the roomSchema. I decided that it was easier to have a host and an opponent, instead of a list of the users in the room. The host is the userID of the player that originally created the room and the opponent is the userID of the player that joined the room. By doing this it was easier to check if there was an issue during the matchmaking process as I clearly know which user was the original member of the room and which user was the joining member. I also had realised that in the database design I had not recorded the rooms score which was essential to keep saved during the game therefore, I added a score element to the schema.

```
const roomSchema    = mongoose.Schema({
   //this is the room schema like the database
   host         : { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
   opponent     : { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
   gamemode     : { type: String },
   createdAt    : { type: Date },
   startedAt    : { type: Date },
   endedAt      : { type: Date },
   score        : { type: Number},
});
```

I also created another database which I hadn't originally planned to make. This database stored all the possible keywords that can be used in the game, along with the game modes that they were suitable for.

```
const keywordsSchema    = mongoose.Schema({
   //this is the keywords schema like the database
   keyword    : { type: String, maxlength: 20 },
   mode       : {type:String}
});
```

According to the Data Protection Act 1998, data must be 'kept safe and secure' [46]. If someone could access the data, Wordifys reputation could be severely damaged. The school has its own mongo servers which are members of ISF. This is important as the database will be storing some sensitive information such as users passwords (although encrypted using bcrypt) and their email addresses. Not only is it important to keep the users data safe but It is also important to keep the word data collected secure as this is the most valuable part of the game.

4.3 Utilising the WordNet database

During the planning process of the project, I intended to use the package 'WordNetMagic' to search the WordNet database. I initially implemented 'WordNetMagic' which is a node.js module for working with Princeton's WordNet lexical database for the English language. However, the package was far too slow when running it in the game, meaning it took far too long to see if a matched word pair was actually correct. I then researched alternative WordNet APIs to find a more suitable library for my project. Not only did I struggle with finding a system which is fast enough but I also struggled with finding a system that can retrieve the antonyms of words. The package I found was called 'wordnet' [47]. 'wordnet' was much faster at gathering data for words during the game, however it did not have the capabilities to retrieve antonyms from the database. I finally found a package called 'wordnetjs' which is a build of WordNet in JSON [48]. It was fast and effective at finding synonyms and hypernyms. The package can also quickly find antonyms, but it rarely has more than 3 or 4 antonyms for a word. In the future, I may decide to change the package I have implemented to one which is more effective at retrieving suitable antonyms.
Finding the synonyms, antonyms and hypernyms for each word using 'wordnetjs' was very simple thanks to the functions the library offers.
When searching for synonyms, you can find both close words which are closely related and far words which are less closely related to the given word.

Finding Synonyms

```
var synonyms = wn.synonyms(gameWord);
                    for(var i=0; i<synonyms.length; i++){
                        var closeSynonyms = synonyms[i].close;
                        var farSynonyms = synonyms[i].far;

                        for(var j=0; j<closeSynonyms.length; j++){
                            allSynonyms.add(closeSynonyms[j]);
                        }
                        for(var j=0; j<farSynonyms.length; j++){
                            allSynonyms.add(farSynonyms[j]);
                        }
                    }
```

Finding Antonyms

When searching for antonyms for a word, the function first searches for the synonyms and then proceeds to find antonyms for both the gameWord and its close synonyms. By using this method, more antonyms can be found, making the WordNet search more effective.

```
var antonyms = wn.antonyms(gameWord);
                    for(var i=0; i<antonyms.length; i++){
                        var words = antonyms[i].words;

                        for(var j=0; j<allSynonyms.length; j++){
                            var closeSynonym = allSynonyms[j];
                            var closeAntonyms = wn.antonyms(closeSynonym.toLowerCase());
                            for(var k=0; k<closeAntonyms.length; k++){
                                var closeWords = closeAntonyms[k].words;

                                for(var l=0; l<closeWords.length; l++){
                                    allAntonyms.add(closeWords[l]);
                                }
                            }
                        }
                        for(var j=0; j<words.length; j++){
                            allAntonyms.add(words[j]);
```

Finding Hypernyms

Hypernyms are found by simply searching the word and looking up the categories it belongs to. If the category is of type “noun” it also finds all words relating to it and adds them to a list. By using this method hypernyms can be found for different words.

```
var hypernyms=wn.lookup(gameWord);
                    for(var i=0; i<hypernyms.length; i++){
                        var category = hypernyms[i].syntactic_category;
                        if(category == "Noun"){
                            var relations = hypernyms[i].relationships.type_of;
                            for(var j=0; j<relations.length; j++){
                                var relationId = relations[j];
                                var relationWord = wn.lookup(relationId);
                                for(var k=0; k<relationWord.length; k++){
                                    var hypernymWords = relationWord[k].words;
                                    for(var l=0; l<hypernymWords.length; l++){
                                        allHypernyms.add(hypernymWords[l]);
                                    }
                                }
                            }
                        }
                    }
```

4.4 Hosting

I decided to use Heroku to host my game as it was a free hosting service which will make my game available to the public. This will make testing much easier than if I used the school’s server to host as each tester would need Cardiff University credentials to access the website. One benefit of Heroku is its servers’ reliability, in the last 60 days (as of 15th April 2018) its uptime is 99.999408% [49]. In order to use Heroku I needed to create a 'package.json’ file which included all of the npm packages that I used for my website. I created this by typing into the terminal where my index file was ‘npm install’. After pushing the final version of my game to my GitHub repository I linked the repository to my Heroku account. Another benefit of Heroku is that as it is linked to my GitHub, any changes that I push through to the master will automatically update to the website. I created a file called ‘procfile’ which is used by Heroku to know what commands should be first called when starting the application [50]. Below is the contents of the file

```
web: node index.js
```

I have planned to use the school’s version of MongoDB for my database, however it can only be accessed on campus. Therefore, during the construction of my system, I utilised mLab which is a free to use cloud-hosted MongoDB service. Another benefit of mLab is that Heroku offer it as an add on [51] meaning the databases can be easily viewed when logged into Heroku via an easy access link. To connect to the database, I simply used the following command in my server side index file:

```
mongoose.connect('mongodb://admin:wowow1234@ds119449.mlab.com:19449/heroku_xfw6k262');
```

5 Results and Evaluation

I am now on the final stage of the waterfall methodology during this project. I will be examining the results of my project and evaluating its success. Once I finished the initial implementation of my system, I showed it to my supervisor and she suggested a few improvements which I acted upon. The first improvement I was given was to make it clearer to players when they match on a word and what word they've match on. I added a new element to the game schema which was the 'correctMatch' which records what word the players matched on. If the player passed, it is saved as false. Then, at the end of the game the players are shown a scrollable table featuring all the words they had in the game, along with what they matched on. Below is the newly adapted game schema along with the matched words table.

```
const mongoose  = require('mongoose');//requires the module mongoose

// game model
const gameSchema    = mongoose.Schema({
   room         : { type: mongoose.Schema.Types.ObjectId, ref: 'Room' },
   keyword      : { type: String },
   relation     : { type: String },
   correctMatch       : { type: String, default: false},
   guesses      : [{
                    player: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
                    guess:  { type: String },
                }]
});
```

Figure 27 Scrollable matched words table show to players at the end of their game

I also made the information text that appears when players match clearer by over doubling the size of the text and by increasing the time it is shown by 200 milliseconds; the improvements can be seen below.

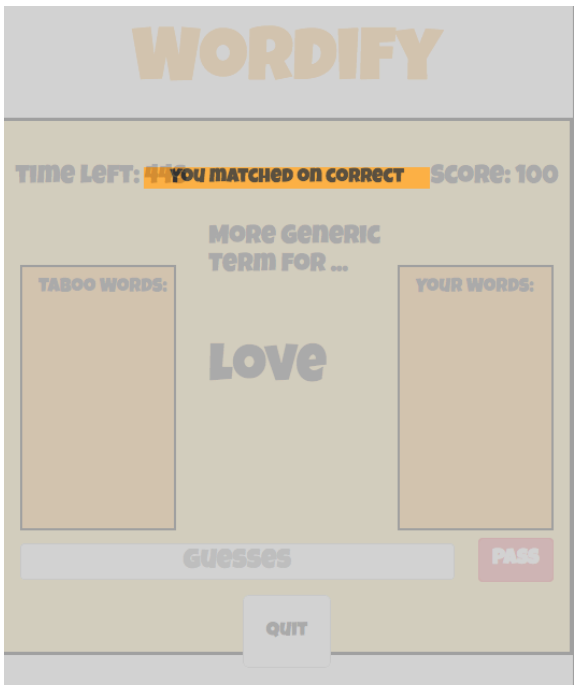

Figure 28 Wordify information Text before improvement

Figure 29 Wordify information text after improvement

5.1 Testing

I will now conduct various tests on my completed game which will help me deem whether my project has been a success. The testing will also help me to discover any last-minute bugs that could be corrected before presenting my GWAP to the client. I would also like to see if there are any areas that the test participants think could be improved in future versions of the game. There are two types of testing that I will be conducting; Usability and functionality testing. I have decided to conduct these two testing methods because together they cover all of the areas of my system that need to be evaluated. Functionality testing involves verifying that the game complies with the functional requirements that I developed during the design process of the report. Unlike functionality testing, usability testing takes the design principles into concern, it will examine the games appearance, accessibility and overall flow. My usability testing will focus on how well the customer can use the product to complete the given tasks. I will be conducting functionality testing before usability as testing on a website that has functionality issues will only uncover functional problems. This may also cause participants to be frustrated if struggling with bugs when trying to complete a task which may undermine my usability results.

5.1.1 Functionality Testing

I will now begin my functionality testing to deem whether my game delivered upon my functional requirements that I produced in the planning stage of the project. I will be conducting multiple tests to prove that I have complied with all my functional requirements. One type of functionality testing that I will be conducting is compatibility testing. Compatibility testing involves testing the system on different types of hardware, operating systems and applications. As my requirements specify that 'The game must be playable on at least the top 4 most used browsers according to NetMarketShare [12]' and 'the game must be playable on mobile phone and tablet devices', I considered this test an important one to conduct. I will also be completing my test cases that I produced at the start of the project to see if the game features the functions that I had originally planned it to have. My test cases should help me to the deduce results of requirements 1, 2 5 and 8.

Compatibility testing

As I want my game to be playable by as many people as possible it is important to ensure that the website functions on various devices. In requirement number 6, I stated that my game must function on at least the top 4 browsers on market share (as of 4th April 1017): Chrome, Internet Explorer, Firefox, Edge. I also stated in requirement number 7 that the game must be playable on mobile and tablet devices.

By having a more accessible game, more people can play which means more data can be gathered. Below are the results of my computer browsers compatibility testing. Not only should the games all function on each browser but they should also be consistent with appearance where possible. This is important as you want the users to be as familiar with the game as possible when accessing it on different platforms.

Browser: Device	Testing results	Comments
Chrome: Mac	Pass	Game worked as expected, no issues.
Internet Explorer: Windows PC	Pass	Game worked as expected, no issues.
Firefox: Windows PC	Pass	Game worked as expected, no issues.
Edge: Windows PC	Pass	Game worked as expected, no issues.
Safari: Mac	Pass	Game worked as expected, no issues. However, the dropdown box looked different compared to the other browsers. There are ways to overcome this by overriding the style.
Safari: Mobile	Pass	The game worked as expected, however the game was a little more awkward to play, it wasn't responsive and it was awkward as they keyboard took up much of the limited screen space on the mobile device.

Figure 30 Compatibility testing results

As all the compatibility testing passed, I can assume that I have passed requirements number 6 and 7, however there are still improvements that could be made, especially with regards to mobile and tablet devices compatibility. I will discuss these improvements in chapter 6 of the report.

Test Cases

During the planning phase of this project I produced a series of test cases which described how the system should function. I will now check each test case and decide whether they have passed.

Results

Test Case ID: 1		**Test Purpose:** Sign Up	
Environment: OS X Google Chrome			
Precondition:			
Test Case Steps: 3			
Step No	Procedure	Response	Pass/Fail
1	Player goes to the website	The first page is loaded	Pass
2	Player clicks sign up	The sign up page is loaded	Pass
3.i	Player enters an email address that is already being used	Comment next to the email address comes up informing the user that the email address is already in use.	Pass
3.ii	Player enters username that is already being used.	Comment next to the username informs the user that the username is already taken.	Pass
3.iii	Passwords enters are not the same	Comment next to the password appears informing the user that the two passwords entered do not match.	Pass
3.iiii	All fields entered by the user are valid.	Sign Up is complete and the user is logged in to the game with their new account	Pass
Comments:			

Test Case ID: 2		**Test Purpose:** Log In	
Environment: OS X Google Chrome			
Precondition: Player has already signed up			
Test Case Steps: 3			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	Player goes to the website.	The first page is loaded	Pass
2	Player clicks Log In.	The Log in page is loaded	Pass
3.i	Player enters incorrect username and correct password and clicks sign in.	Message box appears informing the user that the username is not recognised.	Pass
3.ii	Player enters incorrect username and incorrect password and clicks sign in.	Message box appears informing the user that the username is not recognised.	Pass
3.iii	Player enters correct email address and incorrect password and clicks sign in.	Message box appears informing the user that the password is incorrect.	Pass
3.iiii	Player enters correct email address and correct password and clicks sign in.	Player is taken to the logged in page.	Pass
Comments:			

Test Case ID: 3		**Test Purpose:** Play Game against person	
Environment: OS X Google Chrome			
Precondition: Player is logged in			
Test Case Steps:			
Step No	Procedure	Response	Pass/Fail
1	Select Classic on the drop down mode list.	Classic is selected	Pass
2	Click P v P	The system will take the player to the loading room and will wait until a player is found to play with. The system will then count down 3 seconds until the game starts.	Pass
3	Enter a word in the text box that is corresponding to the word given and press enter.	If the two players match a correct word, the game will move on to the next word, otherwise the player will be prompted to continue entering words	Pass
4	Player continues to play the game for 90 seconds	The game lasts 60 seconds	Fail

5	After 90 seconds the game will end	The system will inform the player of their score and the player will be asked if they want to play again.	Pass
Comments: The game worked as described, the only difference was that I had changed the length of the game form 90 seconds to 60 seconds.			

Test Case ID: 4		**Test Purpose:** View High Scores	
Environment: OS X Google Chrome			
Precondition: Player is not logged in			
Test Case Steps:			
Step No	Procedure	Response	Pass/Fail
1	Player goes to the website.	The first page is loaded	Pass
2	Player selects view leader board	The leader board page is loaded, the player can scroll down through the high scores.	Pass
Comments:			

In my opinion, the test case results prove that the system delivers upon what I had planned. Although it does mostly function as I had planned, there ae some minor changes that have caused some fails within the tests. For example, in test case 3 Steps number 4 and 5 failed because the game lasts 60 seconds instead of the original 90 that I had planned. This was changed because upon reflection I thought that 90 seconds was too long. Therefore, although it was marked down as a failure, it should be seen as an improvement to the system.

Functional Requirements testing

I will now check each of my functional requirements I stated during the implementation stage of the project to see if I have fulfilled them. I will use a combination of my functionality testing results to cipher weather each requirement has passed.

Requirement No	**Pass/Fail**	**Comments**
1	Pass	When visiting the website, players can click sign up and can create an account. Test case No 1
2	Pass	In P Vs P mode players play against other players online. Test Case No 3
3	Pass	In P Vs c mode players play against a bot online.
4	Pass	Players can pass on a word if they are stuck during a game
5	Pass	Each game lasts 60 seconds, Test Case No 3
6	Pass	You can play on any of the top browsers: Compatibility Testing
7	Pass	You can play a game on mobile and tablet devices: Compatibility Testing
8	Pass	Players can view the leader board on the homepage, after a game and before they sign in. Test Case No 4
9	Fail	Players cannot change the computers difficulty, but this could be easily incorporated in the future
10	Fail	Players cannot play as guest, but this could be easily incorporated in the future

Figure 31 Functional requirements testing results

As you can see, my system passed 8 out of the 10 functional requirements. I am pleased with this result as the final 2 requirements were 'could' requirements which I could not comply with due to time constraints.

5.1.2 Ethical Approval

I applied for ethical approval as my usability testing that I will be conducting involves human participation and collecting data. The approval certificate can be found under Appendix A. I produced a briefing information sheet which informed the participant of what they will be asked to do during the testing, this can be found under Appendix B. This helped me to plan my tests and overall sped up the process as it forced me to be organised.

5.1.3 Usability Testing

I will be conducting usability testing to evaluate the user's opinion of my product. By conducting a variation of tests, I hope to get a more rounded idea of how successful my system is. When deciding upon what testing methods to conduct, I considered the benefits and drawbacks of each. One of the testing methods that I have researched is A/B testing [52]. A/B testing involves showing test participants two versions of a product and seeing which version they prefer. This method could be done to see what designs of the game were best preferred. Although I did like the idea behind this method, I decided against it as I have a very limited amount of time and so can't produce many different design versions of my game.

One testing method I will be using to collect usability data is by using a questionnaire. I have chosen this method as it will give me the freedom to find out whatever I want to know from testing participants as I can design the questionnaire. One of the key benefits of using a questionnaire to gather the results is that the actual users or customers of the product will be giving the feedback compared to traditional testing methods that may be managed by the developer of the system. This therefore removes any bias by collecting feedback direct from actual users. I have decided to split my questionnaire up into multiple sections to help it flow better for the participant. I have decided to conduct System Usability Scale (SUS) questions in one of the sections of my questionnaire. SUS is a reliable tool for measuring the usability of a product [53]. One of the main benefits of using this tool is how widely used it is, allowing me to compare my results against others who have completed it. I adapted the questions to suit my game and ensure that it was clear as to what the questions were asking.

Another method of usability testing that I will be conducting is data analysis. This testing method will be used to ensure that the data is actually being collected and to decide if the data that the game produces is sufficient for the client. I will be conducting this method of testing after the questionnaire so that there is some data in the system to examine.

Questionnaire

I will now discuss the steps taken to conduct the questionnaire testing.

Preparations

During the usability testing, there will be 30 Keywords in the database that will be randomly selected. In the future, there would be many more words but for the purposes of this test I deemed 30 as a sufficient number as it is not too few for there to be continuous repetition of words in a game but also not too many so that players will never see the same word more than once.

Before testing, I must acquire 'base data' for my game. I need base data so that some of the most obvious words that are not on WordNet are considered correct. Therefore, I found 6 volunteers who agreed to play the game for 5 minutes against one another. I created sheets which I handed out to each participant informing them of what I would like them to do (figure 34). These games gave Wordify enough data to consider some words correct that previously weren't. For instance, when using the word knife, the hypernym cutlery is not considered correct by WordNet. After data collecting it was considered correct as it had been matched over 10 times. The base data also gave Wordify some 'Taboo' words, making the game more challenging for players. Before the participants began playing Wordify, I timed them to see how long it took them to create an account on the game. This was important to find this out as one of my requirements was that 'it should take no longer than 90 seconds to sign up'. I didn't want to make it too obvious to the participants that I was testing them so I made sure I was discrete as I didn't want to artificially effect their timings.

GWAP data collection

Hello, thank you for agreeing to take part in my Game With a Purpose data collection stage of my testing. For my game to be successful I need some data to be gathered. Please follow the following instructions, if you have any issues, simple ask myself, Bernice Thomas any questions.

1. Create an account:
    a. Please Create a Wordify account, use a fake email address, username and password which I will give to you on a slip and click 'Sign Up'.
2. Now click P vs P and wait until you get matched with another player.
    a. If you have any issues, let me know.
3. Enter as many guesses and possible for what word.
    a. if after 10 seconds you get no matches, click pass and you will move on to the next word.
4. Please play 5 games
5. Thankyou the data collection is over!
    a. Any questions? Email me : bernicethomas@icloud.com

Figure 34 Data Collection Sheet

Results

The first results (below) show how long it took each participant to sign up. It was important that it didn't take too long for users to sign up as they may not bother if it is a big inconvenience. On average it took players 33 seconds to sign up. This therefore means that we have passed Non-functional requirement number 5 which states that it must take users not longer than 90 seconds to sign up.

Participant	**Sign Up time (Seconds)**
1	28
2	40
3	34
4	30
5	33
Average	33

Figure 32 The time it took participants to sign up on Wordify

Once I had finished collecting base data I produced a test plan to ensure my usability testing was organised. It was especially important for my project as I needed the participants to be playing against each other at the same time. This test plan was submitted to the school's ethical board to get ethical approval.

When designing the questionnaire, I followed Question and Questionnaire Design 'Conventional Wisdom' which describes methods that should be adopted for a successful survey design [54]. By following the handbook, I hope to get more accurate results from my usability tests.

The handbook points out that 'Questions on the same topic should be grouped together', therefore I broke up the questions into 3 sections. Not only does this make it much easier for myself when planning what questions to ask, but it also makes it much easier for the participants when filling in the questionnaire as it makes it as clear as possible what I am asking them about. Section 1 will discuss the game play itself, this will help me to determine if I have met some of the non-functional

requirements. Section 2 will be SUS testing and finally, section 3 will ask the participant of any issues or improvements they had for the system.

For question number 3 in section 3 of the survey I used filter questions which is one of the elements that are required in the 'Conventional Wisdom'. I asked the participants if they had any issues with the game before asking them to describe the issues they faced during the game. This is important as users may get annoyed if they are being asked questions that are not relevant to them.

When creating my questionnaire, although I wanted to collect as much data as possible I was aware that people are giving their time voluntarily. Therefore, I ensured that every question I asked was as valuable as possible. I mostly used scale questions where the participants were asked how much they agree or disagree with a statement on a scale of 1 to 5. I chose a 5-point scale because according to the Question and Questionnaire Design 'Conventional Wisdom' scales of 5 had the highest ease of use and I wanted to make my scales as easy to use as possible [55]. By using scale questions, the results will be much easier to evaluate for myself whilst also being quicker and easier for participants to complete.

I used Google forms to produce the questionnaires because it was quick and easy compared to manually creating a survey using Microsoft word. Another drawback of using a word document as the survey is that I would have to physically collect the data whereas google presents the results of the survey as a spreadsheet which makes analysing the results much quicker. Using Google forms also makes distributing the questionnaire out much simpler as I only have to send a link to the participants. Another tool I especially liked was its error handing and answer validation which help to reduce the number of human errors when filling in the questionnaire. For example, in the image below, the answer validation ensures that the participant can only select one option. If they do not it will inform them that they are only required to choose one option from the list. You can also indicate if a question is required to be answered or not, this proved useful for my filter questions as sometimes an answer was required and other times it wasn't.

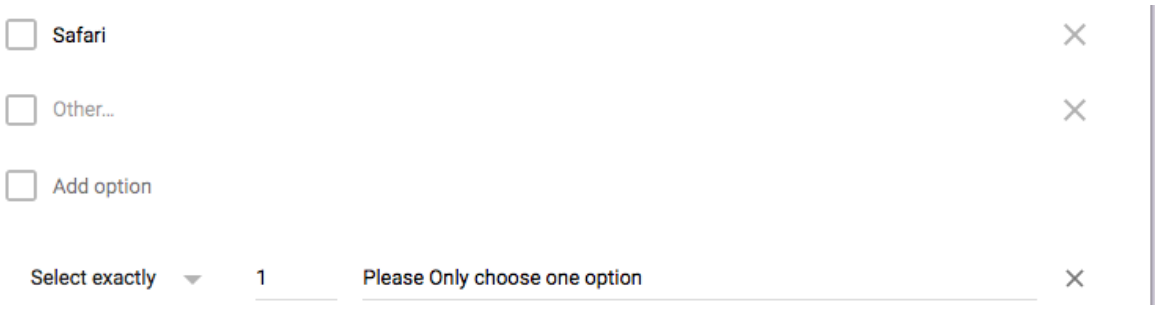

Figure 33 Answer validation example

I recruited 5 volunteers to take part in my questionnaire, all of which had never seen my Wordify game before. I chose to use 5 participants as It is argued that 5 participants would reveal 80% of usability problems, this can be proved in the graph

below from Jakob Nielsen and Thomas K. Landauers research in 1993 [56]. Having 5 participants worked out well logistically as I could have two P vs P games and one P vs C game running at the same time.

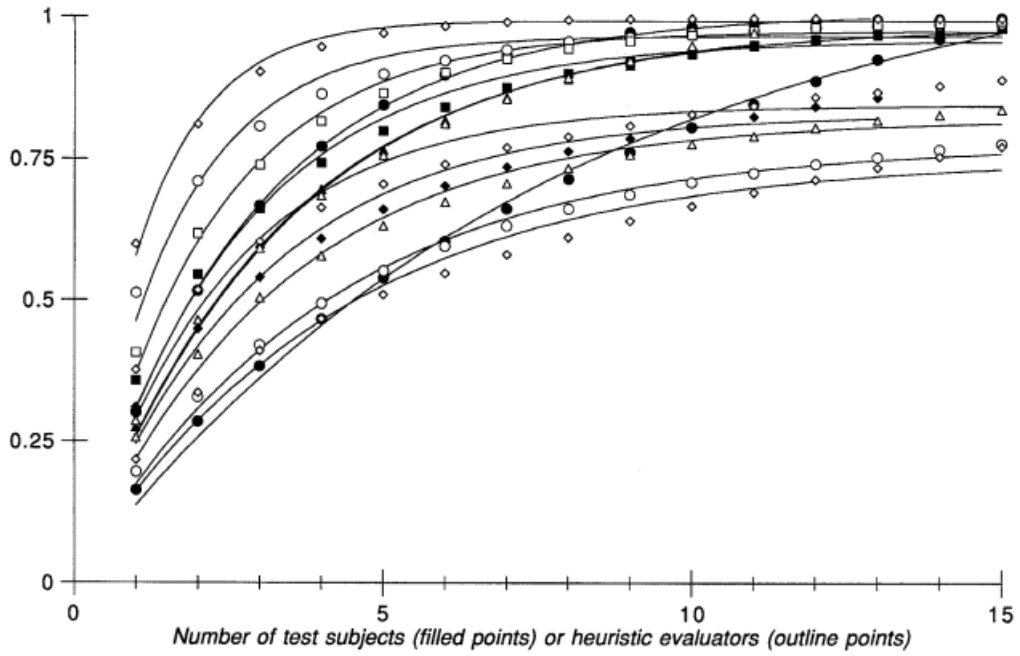

Figure 1 *Proportion of usability problems found with increasing numbers of subjects or evaluators for the interfaces in Table 1. The markers indicate the actual values from the studies and the lines indicate the fitted curves according to (EQ 1). The values from the various studies have been normalized to proportions rather than absolute number of problems to allow comparisons in a single figure.*

Figure 34 Graph showing the relationship between number of testing participants and the number of problems found [57]

You can view my completed questionnaire using the following link: https://tinyurl.com/WordifyQuestionnaire

Questionnaire Testing

During testing, players were asked to play Wordify for 5 minutes, playing a mixture of P vs P games and P vs C games. The participants used a variety of browsers which helps to provide a more accurate reflection of the real-world users. Along with the participants not being allowed to have seen the game before, other eligibility factors for my testing were that they had to be over the age of 16 and finally had to have basic computer skills.
During testing, I oversaw the operations and ensured that people were not having any difficulties with matchmaking when searching for opponents. I wanted to make sure that the participants were aware that they could ask any questions if needs be throughout the process so I constantly reminded them in all the documentation they were given throughout the process. After completing the testing, the participants were asked to fill in the questionnaire that I had created using google forms (https://tinyurl.com/WordifyQuestionnaire). None of my testing participants had any questions after the testing was completed which was a positive indication for the results to come.

I will now discuss the results of each section of my questionnaire from my usability survey. Google forms generated a spreadsheet of the results, saving me valuable time collecting the data. For each question, I have calculated a mean score from all the participants results. A score above 3.5 indicates that the participants tended to agree with the given statement. Any scores below 2.5 indicated that participants tended to disagree with the statements and any scores in-between I regarded as neutral.

Section 1: The Game:

The game section of the questionnaire focusses on the design and functionality of Wordify. The design of the game was important as it helps to keep players engaged. This section of my questionnaire will help me to establish if the system has met some of the requirements that I produced at the start of the project. Question 4 will help me to confirm requirement "The text of the game should be big enough to read".

I wanted to see If they found it exciting to play and if they would want to play it again. It is important for the game to score big on them wanting to play again as the more games that are played means more data that is collected, thus improving the database of words.

The table below shows the results from section 1 of my questionnaire.

I liked the overall design of the website	3	4	5	4	4	Average(mean) 4.2
Navigating through the website is easy	3	4	4	5	5	4.2
I like the games colour scheme	4	5	4	3	3	3.8
The font was clear to read throughout the website	4	5	5	5	5	4.8
Playing the game was exciting	4	5	4	4	4	4.2
I would play Wordify again	5	5	5	5	4	4.8
I think the length of a game (60 seconds) is too short	4	2	3	3	4	3.2

Figure 35 Questionnaire results from section 1

Overall I am pleased with the results of this section, on average people agreed with a score of 4.2 that they found the game exciting and 4.8 to that they would play the game again. These two aspects of the game are vital to have as many people playing the game as possible to collect as much data as possible. The games colour scheme and design scored slightly lower with the participants with an overall score of 3.8 and 4 respectively. However, my client wasn't too concerned with the actual design of the website, and in the future, they can be easily improved. Although this is the case, I still regard these scores as a pass as they are above 3.5, confirming the non-functional requirement 1. I found the results regarding the length of the game interesting as it scored 3.2 which Is pretty much on the fence, in the future I would like to test the game with different time lengths and see which is the most favourable with the participants.

Section 2: SUS Testing:

Section 2 of the questionnaire was used for the SUS testing. This was one of the most important sections of the testing process as it gives a broad overview of the success of the project. Below are the results, along with the mean scores from each question.

I think that I would like to play Wordify frequently	5	5	3	3	4	**Average(mean)** 4
I found Wordify unnecessarily complex	3	1	3	1	1	1.8
Wordify was easy to play	4	4	3	5	5	4.2
I would need the support of a technical person to play the game	1	1	1	1	1	1
The various functions in the game were well integrated	4	4	4	5	4	4.2
There was too much inconsistency in the game	2	2	1	1	1	1.4
Most people would learn to use the game very quickly	4	4	4	5	5	4.4
I found the game very awkward to use	2	2	3	1	1	1.8
I felt very confident playing Wordify	4	5	3	5	4	4.2
I needed to learn allot of things before I could get going with playing Wordify	1	1	2	1	2	1.4

Figure 36 Questionnaire results from section 2

To calculate my websites SUS score I conducted the following calculations to the mean scores from all the participants:

- For each of the odd numbered questions, subtract 1 from the score.
- For each of the even numbered questions, subtract their value from 5.
- Take these new values which you have found, and add up the total score. Then multiply this by 2.5.

The average SUS score is around 68 [58], anything above this score would be deemed an acceptable score. My testing achieved a score of 84. A score above 80.3 is deemed an A grade, therefore I am very pleased with the outcome of the SUS testing. When examining the results there are no obvious outliers that indicate there were issues in their fields.

Section 3: Issues/ Improvements

The final section of my questionnaire enabled me to catch any final bugs that may have been present in the game. Users were prompted to give any issues they faced during testing along with any future improvements that could be made to the game.

I'd rather not have to log in to play the game	Did you come across any issues/bugs during testing?	If so, please explain...	Are there any improvements that you think could be made to the game?	Any Further comments
5	Yes	Log out not visible at all times.	Help button in context. Not having to chose an option before matching another player. Perhaps, the players could be matched and then decide the mode they would like to play in.	
2	Yes	Changing the play type and playing against computer displays a score of 0 right away instead of showing the game.	Bigger screen size	
4	No		notification/alert with information about the game/tutorial	
5	No		Not really	Seemed like a pretty solid program
2	No		You shouldn't have to click the text box to type an answer each game	
3.6	40%			

Figure 37 Questionnaire results from section 3

After conducting the tests, I could fix some of the errors people faced when testing. For example, I enabled the textbox without having to click it by simply inserting the following code in the input element:

```
    onFocus="this.select()"autofocus="autofocus"
```

I was unable to replicate the error mentioned by tester number 2, this highlighted a flaw in my testing method. In the future, I would have liked to have acquired all participants contact details so that I was able to further discuss any issues they faced during testing.

I also wanted to see if people would have preferred to not have to log in to play the game. It may be needless hassle or it could be a way to show off to your friend your high score. The results indicated that people would rather not have to log in. If I had more time I would have liked to have added the play as guest option to the game, however due to time restrictions I was not able to.

5.1.4 Data Analysis

I will now analyse the data collected during the testing and base data stages. Not only will I be able to verify that the game is working properly, I may also be able to see what the most popular antonyms for certain words are. From testing and base data, my system gathered 72 different matched words and a total of 219 matches were recorded. As you can see in the table below, there are stronger correlations with some matches and weaker correlations with others by how many matches were made. In my opinion, these results can help indicate that the system is a success and the games potential if there were more players in the future.

keyword	matchedWord	relation	numberOfMatches
tight	loose	A	8
funny	odd	S	1
fat	big	S	1
fat	huge	S	1
fat	obese	S	2
alcohol	drink	H	9
wrong	incorrect	S	4
correct	incorrect	A	4
love	feeling	H	6
love	emotion	H	6
tea	drink	H	11
right	left	A	8
hard	soft	A	11
cold	freezing	S	1
cold	icey	S	1
cold	chilly	S	2
orange	fruit	H	12
orange	colour	H	10
orange	color	H	5
orange	food	H	8
wrong	correct	A	5
fat	wide	S	1
fat	large	S	1
fat	chubby	S	2
boy	man	H	13

Figure 41 Table of Matched words collected during testing

Non-functional Requirements

I will now be verifying if my system complies with the non-functional requirements that I produced during the planning process of the system. The results of my testing can be seen below.

Requirement No	Pass/Fail	Comments
1	Pass	From my questionnaire the colour scheme scored 3.8, this indicates that people tended to agree that they liked it. Also, there are no red and green or blue and yellow colours overlapping.
2	Pass	A game takes less than 1 second to load.
3	Pass	None of the test participants during usability testing had an issue with loading the game.
4	Pass	All the fonts are over 15pt and from my testing people strongly agreed (4.8/5) that the font was clear to read throughout the website.
5	Pass	We timed our participates when they signed up to the game during testing and it took on average 33 seconds to sign up
6	Pass	From our questionnaire I discovered that people strongly agreed (4.8/5) that they would play Wordify again.

Figure 38 Non-functional requirements testing results

I am really happy with the results from the non-functional requirements testing. My system has passed all the requirements that I stated during planning. I was particularly happy with the results from requirement number 5 as it took participants over half the amount of time to sign up than I had expected. Although this is a positive, it may indicate that my requirements were too easy and I was setting my targets too low.

6 Reflection & Future Work

As the project comes to an end, I will now reflect upon my results and discuss any future work I would like to add to the system at a later date.

6.1 Reflection

I will now reflect upon the results of my testing and discuss the effectiveness and validity of the techniques I adopted. Overall I am pleased with the outcome of the testing. I researched multiple techniques before choosing the methods I adopted and I believe I have completed enough tests to get a rounded view of the system I have produced

If I was to conduct testing again, I would have liked to have asked participants what device and browser they were using to test Wordify on. This would have made interpreting the results much easier as some issues may only exist or may become more prominent on certain platforms. This would have proved particularly useful during section 3 of my questionnaire where participant number 2 had an issue with the game which I could not replicate. Another solution to this may have been to collect contact information from each participant as I would be able to ask them further questions about their issue after the testing. However, this would reduce the anonymity of the testing with thus may affect some of the participant's honesty during the questionnaire.

Another improvement that could have been made to my usability testing is to increase the number of participants. Although Jakob Nielsen and Thomas K. Landauers theory stated that 5 participants revealed 80% of a systems faults [57]. By increasing the number of participants to 10 you would further increase the figure to around 90% which can be seen on figure 37. By increasing the number of test participants, the integrity of my usability testing would be strengthened.

One aspect of my testing that I am particularly proud of is the use of google forms during my usability testing. I was able to split the questionnaire into sections, add instructions before each section and even validate each participant's answers to ensure there were no human errors. In future projects, I will definitely be utilising this tool once again.

6.2 Future Work

Due to the projects time constraints, it was obvious that I wouldn't be able to create the prefect system. Although I am proud of the outcome, there are some aspects that I would like to add to or improve in the future.

6.2.1 Game Play

One improvement to the game play that I wold like to add in the future is even more game modes. One mode in particular that I would like to add is words that rhyme with other words. By adding more game modes into the system, Wordify becomes more exciting and challenging, hopefully bringing in more players to the game.

When playing P vs C games, there are many improvements that could be made to make it more realistic and exciting. One improvement that could be made is with the way the computer chooses what guesses it will make. The Luis Von Ahn's ESP computer player uses pre-recorded games [6] so that people are actually playing against other people, just at different times (not live). Although this method would make it more realistic for the opponent, there may be a more beneficial way of choosing the guesses that would help to further improve the database, for example choosing lesser matches words.

Changing the games difficulty is another feature I would like to add into the game in the future. This could be done in numerous ways. One method that could be used is increasing the number of taboo words. If there are more taboo words, the game is more challenging as players must think of new words. Another method that could be adopted is to increase the number of matches that a word must be regarded as a correct match if it isn't in the WordNet database. By doing this only the most closely related words would be regarded as a match, thus increasing the games difficulty.

One notable issue with the game that I have not yet discussed is that currently, if a player quits, it pulls the opponent out of the game. The ESP game by Louis Von Ahn switched to a computer player automatically if a player quits [6]. This could be a beneficial addition to the game in the future as it would stop players from getting frustrated if they are playing well and then their opponent quits the game.

During the usability testing, participants commented upon the number of duplicate words that featured in a game. I would like to stop duplicate words in games, although in the future when there are more words on the system it will become less often. A simple check could be implemented into the game when creating a new word that verifies that the word has not already appeared in the current game the player is playing in. This would help to make the game more exciting for players.

One final game play improvement that I will discuss involves improving the matchmaking process. The suggestion was made by one of the testing participants to not have to choose a game mode straight away when searching for an opponent, instead choosing after the player's match. This would be a helpful addition to the game, especially during the early stages of the game as there will be lower levels of traffic on the game, helping to make matchmaking much faster. Although this could help matchmaking, I am unsure as to how the two players would be able to come to an agreement over what mode of game they play.

6.2.2 Design

The overall design of the game was not a priority for my client. Therefore, I didn't want to spend too much time designing the system. If I had more time I would like to improve the appearance of the game as it is one of the most important aspects when attempting to draw more players into the game. If the game doesn't look exciting and innovative, less people would want to play the game.

Another design feature that I would like to put more work into is to make the website more mobile and tablet friendly. I had initially hoped to have a fully responsive website that adjusts to the device, however due to time restrictions I simply ensured that the game fitted all devices screen sizes. According to statcounter, mobile and tablet devices account for 48.25% of internet market share in the UK from April 2017 till April 2018 [59]. This highlights the importance of mobile and tablet compatibility.

6.2.3 Performance

Currently, although I have taken some steps to ensure my system is efficient, the performance of the game was not a massive issue as there is very little traffic on the website. In the future, I would take some additional steps to improve the performance of my game. One improvement I may make would be to upgrade my hosting plan with Heroku which is currently on the free service. By doing so, Heroku offers better scalability, storage and memory; this allows the website to handle a greater number of users and improves performance for the game. Another performance enhancing technique that I could adopt is to provide a greater number of servers to support more users connecting from different locations and offer a greater experience when playing the game.

6.2.4 Additional Features

An additional feature that would have to be added to the website at a later date is a forgotten password option when logging in. When a player clicks the forgotten password button on the login page, they should be prompted to give their username or email address. The system should then send an automated email to the corresponding email address with instructions upon how to change their accounts password.

One of the comments made during the testing was to include the log out button on every page of the website. This would be a simple addition programmatically and would help to improve the usability of the website.

My project demonstrates the capabilities of GWAP when applying them to words, but Wordifys game structure could be easily adapted for many other GWAP. One example could be producing family trees of famous people or historical figures. Players could be prompted to give the names of well-known figures mothers, fathers, brothers or sisters, producing a family tree which could be utilised by informative websites such as Wikipedia.

7 Conclusions

From my research into the utilisation of GWAP I have discovered their importance even in today's society. In the age of improving AI there are still tasks that machines simply cannot conduct as effectively as humans due to our abundance of creativity – and machines lack of it.

I will now conclude my findings and highlight the importance of my project. The aim of my project was to implement a GWAP similar to the ESP game, but instead of images players would focus on words and be asked to provide their synonyms, antonyms and hypernyms. Although there is still room for improvement, I believe I have met the goals of my project.
My testing has proved that I able to produce a functional GWAP that efficiently stores the data which can be used for various purposes online. Not only this but I have also produced an exciting game which users enjoyed playing and said that they would continue to play in the future.

8 Reflection on Learning

Not only am I proud of the results of the project I have produced but I am also proud of the progress and development I have personally made throughout this project. By completing a project entirely by myself I have learned to develop many new skills that I had previously not used. I will now reflect upon what I have learned during this project and discuss what aspects I believe I could further improve upon.

I was keen to incorporate many new languages and frameworks into my system that I had not used before to expand my plethora of knowledge and improve my coding skills. Although it meant that it initially took a little longer as I had to learn the new languages, the new languages I did learn helped to speed up production. For example, I had never used a template engine like pug before, but by learning this language I have been able to produce a dynamic website. I used various documentation online to learn to code with pug and to debug errors I faced. Another software I have learnt during this project is sockets. I was keen to implement sockets into my game as I was aware of the performance benefits they offered. Although this was the case, learning how to apply them was not as simple as it initially seemed. I was glad that I chose the package with the most online documentation, as without this I would have struggled to utilise the technologies.

I am proud of my time management skills during this project. I made sure that I stuck to my Gantt Chart as closely as possible, which is not something that I have done during previous projects I have worked on. In the future projects, I would focus on creating a comprehensive, well thought Gantt Chart because this project has made me realise how useful they can be if you properly follow them. If this is not done it can lead to a last-minute rush to complete tasks, reducing the quality of the work produced and can also cause unnecessary stress

My frontend development skills could still be improved, as my supervisor wasn't too concerned by the overall look of the front end, I didn't spend much time researching the best ways to improve the usability of a website. If I was to work on this project again I would spend more time on this try to improve my web design skills and include graphical objects into the game.

Table of abbreviations

GWAP - Game With a Purpose
ISF - Information Security Forum
API - Application programming interface
AI -Artificial Intelligence

Appendices

Appendix A: Ethical Approval Document

APPROVED
Approval ID: COMSC/Ethics/2018/013

School of Computer Science & Informatics

Ethical Approval Request Form
Form valid until 21st March 2018

Instructions

Do not use this form if your research is with the NHS or NHS-linked: please refer instead to the NHS Local Research Ethics Committee.

Do not use this form if your research involves adults who do not have the capacity to consent. Such projects have to be submitted to the National Research Ethics Service (NRES) system: http://nres.nhs.uk/

Please carefully review:

- School Research Ethics documentation
- Data management, collecting personal data, data protection act requirements
- Information Security Framework
- Research Integrity and Governance
- Research Ethics

Please complete the Research Integrity Online Training Programme (Staff link, Student link) prior to submitting this form.

Please complete this form at least **2 weeks** before starting your data collection/human involvement activities and send to comsc-ethics@cardiff.ac.uk along with **all** the relevant attachments:

- Full Project plan/proposal
- Participant Information Form, either:
  - hard copy, e.g briefing and debriefing (if appropriate)
  - online equivalent
- Consent Form or online equivalent (or justification as to why this is not possible)
- Certificate(s) of completion of the Research Integrity Online Training Programme (RIOTP) for all staff associated with a project (and students if applicable).
- (If applicable) Details concerning external funding
- (If an extension is requested) Provide a list of motivations and list of amendments to any previous approvals

Submissions will be reviewed at the next COMSC Research Ethics Group meeting held approximately fortnightly.

Page 1 of 11

Page 1 of 26

Appendix B Briefing Sheet

Game with a purpose Information Sheet

Introduction

You are being invited to take part in a research study. Before you decide it is important for you to understand why the research is being done and what it will involve. Please take the time to read the following information carefully and discuss it with others if you wish. Ask if there is anything that is not clear or if you would like more information. Take time to decide whether or not you wish to take part. Thank you for reading this.

What is the purpose of the study?

The purpose of this study is to get a collection of words that people believe are synonyms, antonyms or hypernyms of other words through the means of a game. It is also to find out how easy the game was to play.

How is the study structured?

Once you have agreed to take part in the study and unless you choose to withdraw at any point, you will be asked to meet up with myself, Bernice Thomas to conduct the research.
In the briefing session, the purpose and workings of this study will be explained to you. You will also be provided with the opportunity to ask questions regarding this study.

During the experiment, you will be asked to play the game on your own computer or one provided (whatever your preference) for 10 minutes. I will then ask you to fill in a questionnaire about how user friendly you found the game to be. You can ask me any questions you have about the study throughout the session.

During the debriefing session, you will be able to ask any questions you have about the research study you have taken part in, and inform me of any issues you had with the study.

Who is organising and funding the research?

The study is organised by researchers from Cardiff University, Bernice Thomas.

Why have I been chosen?

As a volunteer, you have responded to our request for participants to take part via word of mouth.

Do I have to take part?

It is up to you to decide whether or not you want to take part in the study. If you do decide to take part you will be given this information sheet to keep and be asked to sign a consent form. If you decide to take part you are still free to withdraw at any time and without giving a reason.

How will the data be collected and stored?

All information that is collected about you during this research will be kept strictly confidential. The information will be collected by the game will be sent across a secure connection to the database. The database will be kept secure and will be accessible by myself and my supervisor.
We may share the data we collect with researchers at other institutions, but any information that leaves Cardiff University will have your personal details removed. In any sort of output we might publish, we will not include information that will make it possible for other people to know your name or identify you in any way.

References

[1] Wikipedia, “ESP game,” [Online]. Available: https://en.wikipedia.org/wiki/ESP_game.

[2] J. McGonigal, “We spend 3 billion hours a week as a planet playing videogames. Is it worth it? How could it be MORE worth it?,” 15 February 2011. [Online]. Available: https://www.ted.com/conversations/44/we_spend_3_billion_hours_a_wee.html. [Accessed 8 May 2018].

[3] A. Saini, “Solving the web's image problem,” BBC, 14 May 2008. [Online]. Available: http://news.bbc.co.uk/1/hi/technology/7395751.stm. [Accessed 23 February 2018].

[4] T. O'Reilly, “Google Image Labeler, the ESP Game, and Human-Computer Symbiosis,” 3 september 2006. [Online]. Available: http://radar.oreilly.com/2006/09/google-image-labeler-the-esp-g.html. [Accessed 14 March 2018].

[5] Dondona, “ESP Game,” [Online]. Available: https://dodona.ugent.be/en/exercises/1678755178/. [Accessed 24 February 2018].

[6] L. v. Ahn, “YouTube,” 26 July 2006. [Online]. Available: https://www.youtube.com/watch?v=tx082gDwGcM. [Accessed 9 February 2018].

[7] Google, “Crowdsource home,” [Online]. Available: https://crowdsource.google.com/home. [Accessed 2 March 2018].

[8] “Image Labeler Help,” [Online]. Available: https://support.google.com/imagelabeler/answer/7085943?hl=en&ref_topic=7085456. [Accessed 18 February 2018].

[9] T. ROSE-SANDLER, “Smorball and Beanstalk: Games that aren’t just fun to play but help science too,” 28 August 2015. [Online]. Available: https://blog.biodiversitylibrary.org/2015/08/smorball-and-beanstalk-games-that-arent-just-fun-to-play-but-help-science-too.html. [Accessed 1 March 2018].

[10] “About,” NANO Crafter , [Online]. Available: http://nanocrafter.org/about. [Accessed 4 March 2018].

[11] Nanocrafter, “About,” [Online]. Available: http://nanocrafter.org/about. [Accessed 21 March 2018].

[12] NanoCrafter, “Science,” [Online]. Available: http://nanocrafter.org/science. [Accessed 3 March 2018].

[13] WordNet, “WordNet,” [Online]. Available: https://wordnet.princeton.edu/wordnet/. [Accessed 18 February 2018].

[14] WordNet, “News,” 2012. [Online]. Available: https://wordnet.princeton.edu/news-0. [Accessed 29 March 2018].

[15] [Online]. Available: (https://wordnet.princeton.edu/wordnet/license/ .

[16] A. Powell-Morse, “Waterfall Model: What Is It and When Should You Use It?,” 8 December 2016. [Online]. Available: https://airbrake.io/blog/sdlc/waterfall-model. [Accessed 19 February 2018].

[17] Tutorials Point, “SDLC - Waterfall Model,” [Online]. Available: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm. [Accessed 26 February 2018].

[18] NetMarketShare, “Browser Market Share,” 2018. [Online]. Available: https://netmarketshare.com/browser-market-share.aspx?options=%7B%22filter%22%3A%7B%22%24and%22%3A%5B%7B%22deviceType%22%3A%7B%22%24in%22%3A%5B%22Desktop%2Flaptop%22%5D%7D%7D%5D%7D%2C%22dateLabel%22%3A%22Trend%22%2C%22attributes%22%3A%22share%22%2C%22group%22%3A%22browser%22%2C%22sort%22%3A%7B%22share%22%3A-1%7D%2C%22id%22%3A%22browsersDesktop%22%2C%22dateInterval%22%3A%22Monthly%22%2C%22dateStart%22%3A%222017-02%22%2C%22dateEnd%22%3A%222018-01%22%2C%22segments%22%3A%22-1000%22%7D. [Accessed 3 February 2018].

[19] Bootstrap, “Introduction,” [Online]. Available: https://getbootstrap.com/docs/4.0/getting-started/introduction/. [Accessed 3 March 2018].

[20] Google, “Google Fonts,” [Online]. Available: https://developers.google.com/fonts/. [Accessed 13 April 2018].

[21] “Google Fonts FAQ,” Google, 9 April 2018. [Online]. Available: https://developers.google.com/fonts/faq . [Accessed 13 April 2018].

[22] NLTK, “WordNet Interface,” [Online]. Available: http://www.nltk.org/howto/wordnet.html. [Accessed 13 March 2018].

[23] Y. Cui, “Comparing AWS Lambda performance when using Node.js, Java, C# or Python,” 2 April 2017. [Online]. Available: https://read.acloud.guru/comparing-aws-lambda-performance-when-using-node-js-java-c-or-python-281bef2c740f. [Accessed 3 May 2018].

[24] N. Tollervey, “5 reasons why Python is a popular teaching language,” 10 April 2015. [Online]. Available: http://radar.oreilly.com/2015/04/five-reasons-why-python-is-a-popular-teaching-language.html. [Accessed 28 March 2018].

[25] “TIOBE Index for April 2018,” April 2018. [Online]. Available: https://www.tiobe.com/tiobe-index/. [Accessed 14 April 2018].

[26] “tutorials point,” [Online]. Available: https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm. [Accessed 12 April 2018].

[27] CUBRID community, “A Node.js speed dilemma: AJAX or Socket.IO?,” 14 July 2017. [Online]. Available: https://www.cubrid.org/blog/nodejs-speed-dilemma-ajax-or-socket-io. [Accessed 23 March 2018].

[28] SockJS , “SockJS,” 2 May 2018. [Online]. Available: https://github.com/sockjs/sockjs-client. [Accessed 6 May 2018].

[29] Socket.IO, “Socket.IO,” 2018. [Online]. Available: https://socket.io/ . [Accessed 2 May 2018].

[30] express-authentication , “express-authentication,” 2016. [Online]. Available: https://www.npmjs.com/package/express-authentication. [Accessed 29 March 2018].

[31] PassportJS, “Documentation,” [Online]. Available: http://www.passportjs.org/docs/. [Accessed 18 March 2018].

[32] PassportJS, “Documentation,” [Online]. Available: http://www.passportjs.org/docs/. [Accessed 9 May 2018].

[33] PassportJS, “Log In,” [Online]. Available: http://www.passportjs.org/docs/login/ . [Accessed 22 April 2018].

[34] Wordnet, “Stats,” [Online]. Available: https://web.archive.org/web/20171216124924/https://wordnet.princeton.edu/wordnet/man/wnstats.7WN.html. [Accessed 16 December 2017].

[35] WordNet, “Licence and Commercial Use,” [Online]. Available: https://wordnet.princeton.edu/license-and-commercial-use. [Accessed 11 03 2018].

[36] WordNet, “What is Wordnet,” [Online]. Available: https://wordnet.princeton.edu/. [Accessed 11 03 2018].

[37] J. Kamps, “Visualizing WordNet Structure,” 1 October 2014. [Online]. Available: https://www.researchgate.net/publication/228729013_Visualizing_WordNet_structure. [Accessed 17 March 2018].

[38] “Wordnet-magic,” 2015. [Online]. Available: https://www.npmjs.com/package/wordnet-magic. [Accessed 17 May 2018].

[39] Oracle, “Top 10 reasons to choose MySQL for Web-based Applications,” August 2011. [Online]. Available: http://www.oracle.com/us/products/mysql/mysql-wp-top10-webbased-apps-461054.pdf. [Accessed May 1 2018].

[40] MongoDB, “MongoDB Architecture,” 2018. [Online]. Available: https://www.mongodb.com/mongodb-architecture. [Accessed 1 May 2018].

[41] MongoDB, “MongoDB and MySQL Compared,” 2018. [Online]. Available: https://www.mongodb.com/compare/mongodb-mysql. [Accessed 27 April 2018].

[42] “70 MongoDB Interview Questions and Answers,” 3 May 2016. [Online]. Available: https://www.datasciencecentral.com/profiles/blogs/70-mongodb-interview-questions-and-answers. [Accessed 27 April 2018].

[43] Microsoft, “Using Version Control in VS Code,” 3 May 2018. [Online]. Available: https://code.visualstudio.com/docs/editor/versioncontrol. [Accessed 6 May 2018].

[44] Microsoft, “Debugging,” 5 May 2018. [Online]. Available: https://code.visualstudio.com/docs/editor/debugging. [Accessed 6 May 2018].

[45] GitHub, “Tracing Changes in a file,” [Online]. Available: https://help.github.com/articles/tracing-changes-in-a-file/. [Accessed 8 April 2018].

[46] “Data Protection,” [Online]. Available: https://www.gov.uk/data-protection. [Accessed 28 April 2018].

[47] wordnet, “wordnet,” 2014. [Online]. Available: https://www.npmjs.com/package/wordnet. [Accessed 1 May 2018].

[48] spencermountain, “wordnetjs,” npm, 2016. [Online]. Available: https://www.npmjs.com/package/wordnetjs. [Accessed 12 April 2018].

[49] Heroku, “Heroku status,” Heroku, 15 April 2018. [Online]. Available: https://status.heroku.com/. [Accessed 15 April 2018].

[50] Heroku, “Process Types and the Procfile,” 15 March 2018. [Online]. Available: https://devcenter.heroku.com/articles/procfile. [Accessed 2 April 2018].

[51] Heroku, “mLab MongoDB,” [Online]. Available: https://elements.heroku.com/addons/mongolab. [Accessed 28 March 2018].

[52] VWO, “A/B Testing,” [Online]. Available: https://vwo.com/ab-testing/. [Accessed 2018 April 9].

[53] usability.gov, “System Usability Scale (SUS),” Usability.gov, [Online]. Available: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html. [Accessed 20 April 2018].

[54] J. A. Krosnick and S. Presser, “Question and Questionnaire Design,” 15 February 2009. [Online]. Available: https://web.stanford.edu/dept/communication/faculty/krosnick/docs/2009/2009_handbook_krosnick.pdf. [Accessed 5 April 2017].

[55] C. C. Preston and A. M. Colman, “Optimal number of response categories in rating scales: reliability, validity, discriminating power, and respondent preferences,” 14 09 1999. [Online]. Available: https://www2.le.ac.uk/departments/npb/people/amc/articles-pdfs/optinumb.pdf. [Accessed 14 04 2018].

[56] J. Nielsen and T. K. Landauer, “A Mathematical Model of the Finding of Usability Problems,” 24-29 April 1993. [Online]. Available: http://peres.rihmlab.org/Classes/PSYC6419seminar/p206-Five%20Users%20nielsen.pdf. [Accessed 5 April 2017].

[57] N. Jakob and K. L. Thomas, 24-29 April 1993. [Online]. Available: http://peres.rihmlab.org/Classes/PSYC6419seminar/p206-Five%20Users%20nielsen.pdf. [Accessed 23 April 2018].

[58] J. Sauro, “Measuring Usability With The System Usability Scale (SUS),” measuringu, 2 February 2011. [Online]. Available: https://measuringu.com/sus/. [Accessed 28 April 2018].

[59] statcounter, “Desktop vs Mobile vs Tablet Market Share United Kingdom,” 30 April 2018. [Online]. Available: http://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/united-kingdom. [Accessed 9 May 2018].

[60] MongoDB, “MongoDB and MySQL Compared,” [Online]. Available: https://www.mongodb.com/compare/mongodb-mysql. [Accessed 31 January 2018].

[61] SPOJ, “ESP game (series 07),” 2015. [Online]. Available: http://www.spoj.com/GUGC2015/problems/PROG0233/. [Accessed 7 February 2018].