The Development of a Quizzing Application

Team 8: Renata Jeny, Mohammad Nadat, Dominika Piosik, Amy Gourlay, Sebastian Thomas, Mandar Tamhane, Kyle Leggate, Arnav Dhawan, Steven Marshall, Yousaf Nasir
AC41004 Industrial Team Project
B.Sc. Hons Applied Computing / Computing Science
University of Dundee, 2020
Supervisors: Dr Brian Plüss & Mr Hasith Nandadasa

Abstract - *At its most basic level a quiz is a form of mind sport or game in which players attempt to answer questions correctly. It is a game to test a player's knowledge about certain subjects. Modern society leads quite a busy life. With quiz games becoming ever more popular, it's often difficult to arrange a game for a time that suits everyone.*

The aim of this three-week project was to develop a quiz game that didn't have any security implications. An Agile approach was used to this project. The team first came up with a design for the game, which was then followed by a prototype. After this user testing occurred and the game was then evaluated before becoming a fully developed game.

In essence, this document aims to outline and describe the development process of a web-based quiz game and how the final game was completed. This includes the background research, specification, design, implementation, user testing and evaluation required to make it.

1 Introduction

Before the nation went into lockdown in March, spending time with family and friends was very different. Whether weekends were spent watching football, going to the pub or going for walks, suddenly none of these options were viable. However, many people quickly found comfort in the ever-popular weekly Zoom quiz [1]. Although it wasn't the platform's primary use, it became the norm for the nation. This created a gap in the market for a more refined, unique and suitable way of quizzing with friends and relatives, which was the intention of this project.

The project proposal was flexible in nature, this meant that there was total control of which languages, technologies and methodologies to adopt when completing the project. Additionally there were several personal objectives which the team set out that contributed towards making this quizzing application, they were as follows: Create a web-based quizzing application for friends or family virtually, require no accounts or logins, have a lifeline when users are stuck, a scoring system, a leader board, and a large and diverse set of questions. It was also decided to carry out this project using an Agile approach [2], with the development split into 3 teams: Back-end, Front-end, and a Database team.

2 Background

Due to the open-ended nature of the project, relevant background research had to be conducted in order to gain further understanding of the project scope. This involved performing a competitor analysis [See Appendix A] to help identify possible requirements and conducting research for User Interface design. Additionally, time was also spent exploring and comparing different frameworks which would be essential to the composition of the project.

2.1 QuizUp

QuizUp [3] is a trivia game similar to Trivial Pursuit and is available on Android and IOS. The user is able to pick a topic out of thousands available then can either challenge a random player from around the world or one of their friends. In a game, there's six rounds of questions followed by a bonus question. More points are awarded for the player who gives the correct answer in the shortest time possible.

One of the key features of QuizUp is that when playing with a friend, the player can either challenge them in real time, or they can choose to play the game at a later point [4]. With everyone having busy lives this gives a lot of flexibility in the game; friends can play together but they don't necessarily have to play at the same time. This is something the team found unique and were keen to implement something similar.

2.2 Kahoot

Kahoot [5] is a game-based learning platform where users can play and generate games. The games take the form of multiple-choice quizzes that can be accessed via a web browser or through the Kahoot mobile application available on both IOS and Android.

As Kahoot is targeted towards an educational setting, some distinguishing aspects of this application were drawn up. Much like QuizUp, gameplay does not have to take place at the same time, since players can opt to play at their own convenient time. Another feature of Kahoot which sets it apart from its competitors such as 'QuizUp'

and ‘QuizWitz’ [6] is the fact that users are not required to login when joining a lobby. All that is required is the game pin, thus allowing users to start playing a game relatively quickly as opposed to other similar multiplayer quiz websites.

Kahoot utilises a clean and simple User Interface, helping provide a clear navigation throughout the website. This was an important aspect which was noted by the front-end team as it correlated with the team’s own ambitions of creating a straightforward yet attractive design. Although the core features of Kahoot are to allow players to create and join a lobby to start playing a quiz, there are also a number of additional features which supplemented the range of offerings provided by Kahoot. Also it allows users to create their own quizzes and then being able to publish them to other users in either a public or private setting [7]. This was found to not only be an exciting addition, but also an extremely effective method of engaging with its users and enhancing the customisability of the quizzes.

2.3 What makes a good online game?

There are many aspects that contribute to making a good and successful online game. Research was conducted into this area by members of the project team in order to identify trends. This research led the team to classify exactly what makes a good online game.

It can be noted how a growing popularity of multiplayer games has created a consumer demand for originality among the multitude of games available on the market. This has highlighted the specific need of a game to have unique elements in some form or another, in order to attract potential users. Thus, originality [8] was classified to be a key aspect of creating an engaging and fun game.

An online game should also aim to be re-playable as this helps ensure that players are returning to play more [9]. Therefore, the game should maintain a difficulty balance in order to ensure a challenging yet exciting gameplay. This will also ensure that players are able to feel excited each time they play the game through the stimulating experience, thus mitigating the possibility of players getting bored quickly.

Additionally, a key factor when discussing the refinement of a game is the controls in place for bugs and potential cheating [10]. Players want to play a game that is fundamentally reliable and robust. Therefore, it is vitally important that controls are in place which prevent cheating from occurring as this could drive users away.

2.4 Target Audience

The project’s target audience is the interests and demographics of the majority of its users. Although it would have been good to reach out to as many people possible, it’s more important that it directly got to potential consumers as they are the people who are more likely to play the game. At the end of 2019 there were approximately 2.5 billion gamers in the world [11]. That figure was up 500 million from four years ago, and this is expected to increase by 100 million by the end of 2020. This resulted in the team taking time to carefully choose the target audience for the game.

One typical way developers determine their target audience is by assuming that they are similar people to themselves. However, this isn’t necessarily always true, plus, it is quite a narrow-minded approach. Instead, a data driven approach was adopted.

The team started with trying to research the project’s biggest 3 competitors as the data from this would have been one of the most effective ways to produce a good estimate of who the project’s target audience would be. The competitors the team chose were Kahoot!, QuizUp, and Trivia Crack. An Excel spreadsheet was then produced with these games to include their demographics and user’s interests. However, this was unsuccessful as the means of getting user statistics data were no longer free. Until a couple years ago it was possible to get demographics from Facebook Audience Insights and Google AdWords, but these services now need to be paid for in order to use them.

This lead to an alternative strategy for finding the target audience. In order to break the project down, it was firstly decided if the game emphasised cognitive skills, which are the core mechanics of how humans learn, and retain focused, and problem-solve, or if it mainly emphasised sensorimotor skills, which is information we receive from our bodies through our sensory systems like touch (aiming and reactions). Due to the project being a quiz/trivia game it was decided that although it had sensorimotor aspects in terms of the user receiving a higher score if they answered a question quicker, it leaned more towards teaching cognitive skills due to that fact that the game’s priority is to quiz the player.

Secondly it was decided if a single-dimensional (one layer) or multi-dimensional (two or more layers) game. As it was a quiz that solely focuses on answering a correct question, the team found that it was a single-dimensional-game. With knowing that the team’s game mainly focused on cognitive skills and was single dimensional, it was concluded that the target audience would be casual gamers. These gamers tend to have even gender split of players between 18 and 35 and will usually play the game infrequently and in short bursts. A study from Global Web Index showed that 77% of millenials and 57% of the Gen Z population who are gaming more due to Covid-19 stated they will keep gaming after the pandemic. This is important due to the fact that even before the pandemic the number of casual gamers has been increasingly rising, this will begin to soften the more hardcore and midcore gamers allowing an even more diverse gaming community [12].

2.5 Website Design

As the project was aimed towards building a Web-Based Application, extensive research was carried out to investigate the key components of good website design [13], as well as looking into examples of website designs that were to be avoided. This direct comparison enabled the front-end team to recognize elements of successful websites. Hence, allowing them to begin devising a meticulous approach to the overall design.

Although designs are subjective in nature, the team was able to form a general picture of the desired outlook of the project. Through the research that was performed into looking at different websites, elements of website design were identified which the team collectively deemed to be best avoided.

The 'Ken's Quiz Site' [14] was disclosed as part of the website and user interface design research. It was highlighted by the team to have a variety of negative features in terms of web accessibility. The homepage, immediately registered as being cluttered with numerous links to different quizzes [See Figure 1]. Although the links were separated into categories, the images accompanying the category were deemed to be inconsistent and difficult to discern what the category was about. Hence it was necessary to read the title to find out what the quiz category was. This meant that the images acted more of a distraction, rather than supplementing the overall interface. Additionally, the colour scheme applied did not seem to be the most appealing for users. Also the colour contrast ratio was judged to be too low since it did not meet the WCAG guidelines [15]. The research conducted on this website, made the front-end team ensure that the finished product incorporated a pleasing colour scheme and to necessitate a clear navigation for the website.

Figure 1 – Homepage screen of 'Ken's Quiz Site'

2.6 Technologies Used

2.6.1 HTML

HTML (HyperText Markup Language) [16] is fundamental to every webpage – acting as the core. It enables the creation and structuring of sections, paragraphs, headings and links for webpages and applications. Being the main markup language of the web, it runs natively in every browser. Hence, it provides the basic structure of websites, which is enhanced by the incorporation of CSS and JavaScript.

2.6.2 CSS

CSS (Cascading Style Sheets) is used to define styles for webpages [17]. This includes the design, layout, colours and fonts. CSS aids web developers to create a uniform and consistent look across several pages of a website. As HTML is the foundation of the website, CSS deals with the presentation. Therefore, once a style is defined within a Cascading Style Sheet, it can be used by any page that references the CSS file. Additionally, CSS can also be used to define the cell padding of tables cells and colours of a tables border – thus providing more control to the developers on how the Web pages look.

2.6.3 JavaScript

JavaScript is a programming language which is commonly used in web development. As it is a client-side scripting language – the source code is processed by the client side as opposed to the web server. This means that JavaScript functions can be executed after a webpage has loaded without communicating with the server. Hence, it can be noted how JavaScript is used to control the behaviour of different elements, as it is responsible for interactivity of a site, due to most of the dynamic behaviour on a web page owing to JavaScript.

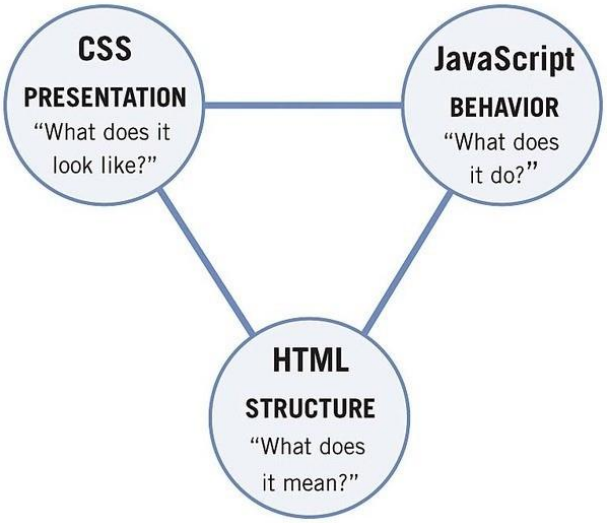

Figure 2 – Diagram of the relationship of HTML, CSS and JavaScript [18].

2.6.4 Visual Studio Code

Visual Studio Code [19] is an Integrated Development Environment developed by Microsoft for Windows, macOS and Linux. It aims to provide developers with all the necessary tools in place to build applications with support for operations such as debugging, syntax highlighting and code refactoring. This IDE was selected over NetBeans [20] primarily due to the development team having previous experience with Visual Studio Code, enabling them to start implementing code efficiently. Additionally, the fact that Visual Studio Code is lightweight and versatile – providing support for extensions – as well as being rated highly for web

development, was a major contributing factor to the final decision of opting with this IDE.

2.6.5 Vue.js

Vue.js [21] was selected to be utilized for the front-end due to its simplicity compared to alternatives. As Vue does not require advanced developer tooling to get started like in React [22] and Angular [23] it is relatively easy to set up. This was a key consideration when deciding which framework to choose, due to the time constraints of the project.

2.6.6 Nuxt.js

Nuxt.js [24] builds on top of Vue.js. It is a higher-level framework as it simplifies the development of Vue.js applications. It allows creation of universal Vue applications which means applications that are rendered on the server. Additionally, Nuxt.js allows the pre-rendering of views on the server as well as the fact that it grants configuration via file and folder structure. Lastly, Nuxt.js comes with Babel which is a JavaScript compiler which handles latest JavaScript versions such as ES6/ES7.

2.6.7 Framework Alternatives

The two main alternatives that were considered were React and Angular. Vue was chosen over React because it has more features that could be advantageous to the project. As Vue incorporates built in features such as state management and routing the team agreed was necessary for the project.

Angular was not selected as it used TypeScript that may have proved to be more difficult to learn for members of the development team. Additionally, as Angular does not run in the browser; Vue was unanimously agreed to be utilized for the completion of this project.

3 Specification

After background research was conducted and different components were researched and understood, the product specification was produced. In order to reflect user testing feedback, the original specification was revised throughout the project lifecycle. A product backlog [See Appendix B] was created to ensure that all the components from the specification would be implemented. Agile methods were also used during the development of the project.

3.1 Requirements

Prior to commencing development, detailed planning sessions took place to specify the desired set of features. The requirements were based on the Client Brief and the development team drafted a set of User Stories [See Appendix C] and considered relative importance of each User Story. These were then ranked in terms of priority and the estimated effort needed to deliver each feature. That served as the backbone for the Product Backlog created in the form of an Excel spreadsheet. Over the course of the project changes to both the User Stories and the Product Backlog were made.

3.2 Methodology

Before development began, the team discussed methodologies suitable for a highly dynamic environment that would allow for swift reaction to changing requirements, stakeholder engagement, and provide opportunities to receive rapid feedback. The Agile approach with scrum framework was selected as it was determined to best serve the needs of the project, employ a wide range of skills of the cross-functional team, and aid self-management with strict responsibility for assigned tasks. Moreover, Agile is generally known for improving team productivity and satisfaction [25].

The team assumed a blended approach, incorporating User-Centred Design into the development process. Even though Agile and UCD seem distant, with seemingly incompatible allocation of resources, the main similarity is that the two methodologies are both user and customer focused [26]. Applying the two techniques allowed for better understanding of the problem, rapid testing, and helped mitigate project risk, so that the team was able to deliver a usable product that meets the needs of the client and the end user.

Considering time constraints, the team agreed on 1-week long sprints with a planning session at the start of each sprint and sprint retrospective meeting, both led by the team leader who acted as product owner. At each planning session a to-do list was drafted and the items allocated for each sprint were then put into sprint backlogs [See Appendix D]. Weekly meetings with the client served as sprint reviews and helped reflect on the progress made in the previous week, ensuring the product meets the needs of the stakeholders. Daily scrum meetings were helpful in progress evaluation and issue identification, and all milestones were noted in the backlog.

3.3 Project Management

During the initial planning meeting, the team drafted a general timeline based on the client brief that offered flexibility. The first week was dedicated to setting up rules for teamwork, research, deciding on the details and refining the plan. The second and third week were meant to be the implementation stage, where the product started to take shape, with the basic application version produced by the end of week 2 and week 3 being adding more advanced features that make the gameplay more entertaining. This approach worked well, allowing the team to enter each sprint with a clear vision and set of goals to accomplish by each review. The team followed the initial plan over the three weeks, with small changes to accommodate the users' needs and the feedback from the client.

At the start the team appointed a project manager and a scrum master as permanent positions to manage task allocation and documentation throughout the project lifecycle. The team of 10 was initially split into three subgroups focusing on researching and setting up the foundation for three aspects of the project: frontend (4 people), backend (2 people) and API (4 people), with team leaders appointed for each of the subgroups. This structure was ideal for the first week, however week 2 brought new challenges and restructuration were necessary. From week 2 onwards, the team was divided into two groups with their main areas of focus being software development and report writing & documentation. 6 people worked solely on refining the mock-up, setting up the database and API, integrating new features into the application. The remaining 4 people concentrated on the report, user testing and evaluation, and introducing changes to user stories and the product backlog based on client feedback. This allowed the team to maximise their potential, with clear task allocation that played to every members' strengths.

3.4 Ethical Approval

The team agreed that to adhere to user-centered design practices, user testing was required to thoroughly evaluate the web application at the prototyping stage as well as the final evaluation stage. The user interview script consisted of two parts, both in the form of a video call. Firstly, the participants received a set of instructions based on the created user stories, and the researcher observed the participants using the application on the user's own device through screen sharing. The second part was an interview, with the researcher asking open-ended questions, taking notes of the participants' answers.

As the project falls into the low-risk category, the group application submitted by Dr Plüss on behalf of AC41004 proved to be sufficient for the user involvement planned.

3.5 Slack

Due to the team having a lot of members, an application called Slack was used [27]. A Slack workspace is made up of channels, where team members can communicate and work together. Each subgroup had a channel dedicated to their area, this allowed easy, effective and simplified communication in the subgroups. A channel with every team member was also created to allow each subgroup to communicate, or any general issues in the project to be discussed. This was also used to arrange meetings.

3.6 RACI Matrix

Due to the complexity of the project where many tasks were dependent on other tasks, an RACI Chart [See Appendix E] was adopted in order to better aid the team. As an RACI Matrix helps to highlight who exactly is responsible for which precise task [See Figure 3]. This enabled the team to identify which member was responsible, accountable, consulted and informed for a particular assignment [28]. Thus, allowing communication to be more streamlined during the project, as members knew exactly who they needed to speak to. Additionally, the risk of conflict for task ownership was also mitigated using the RACI Chart, as task roles were clarified, hence eliminating potential confusion.

Team Key:	Front-end Team	Report Team	Back-end Team	Database-Team					
RACI CHART WEEK 3									
Project task or deliverable	Renata Jeny (PM)	Amy Gourlay	Mandar Tamhane	Sebastian Thomas	Mo Nadat	Yousaf Nasir	Dominika Piosik	Steven Marshall	Kyle Leggate	Arnav Dhawan
Create game lifelines	C	C	C	A & R	C	I	I	I	C	C
Program the sounds	C	C	C	A & R	C	I	I	I	C	C
Work on Game Timer	C	C	C	C	A & R	I	I	I	C	C
Finish the Scoring System	C	A & R	C	C	C	I	I	I	C	C
Scoreboard Sorting	C	C	A & R	C	C	I	I	I	C	C
Finish Game	A & R	R	R	R	R	I	C	I	R	R
Unit Testing	A & R	R	R	R	R	I	I	I	C	C
Code Review	A & R	R	R	R	R	I	I	I	R	R
Report Write-Up	C	I	I	C	I	A & R	R	R	C	I
Documentation	A & R	I	C	I	I	R	R	I	I	I
User Testing	R	I	I	I	I	C	A & R	C	I	I
API code	C	I	I	I	I	I	I	I	A & R	R
Database Restructuring	C	I	I	I	I	I	I	I	A & R	R

Figure 3 – RACI Matrix Chart used for Week 3

4 Design

4.1 User Centred Design

The team wanted to focus on a good user experience being the core of the product. All the design decisions made were to ensure that the user is at the centre of the process. Having a user centred approach [29] to the designs also allowed the team to pick up certain design principles widely found in other digital products.

Familiarity [30] is a good UX strategy and that is exactly what the team wanted to incorporate. It was important that users would not be spending time figuring out what they should be doing. There could have been a lot of design choices that would make the game a lot more sophisticated and interesting but figuring out how to play it may have increased unnecessary complexity. Therefore, a decision was made from the start of the design process to have a lightweight, simple interface without any over-the-top gimmicks.

4.2 Accessibility

The whole product was designed with accessibility being a huge part of the design choices. The project team did not want to create a separate option or mode for better accessibility. Instead, it was paramount that the product be accessible out of the box. Due to this, designs were made catering for keyboard input, creating affordances for colour-blindness, and testing extensively for colour contrasts. Unfortunately, due to some restrictions in the front-end framework, it was not possible to provide for keyboard feedback and colour-blind affordances in the final prototype.

4.3 Miro Board

A Miro Board [See Appendix F] was used during Sprint 1 and 2 by all members of the team. It was primarily utilised for brainstorming initial ideas and game features, which then transitioned into creating the low and high-

fidelity prototypes. This enabled the team to collaborate extensively whilst still being able to retain all the necessary information in one place. Additionally, the Miro board was also further used as a communication tool, where each team member could use it to collate ideas and then showcase them in the board [See Figure 4]

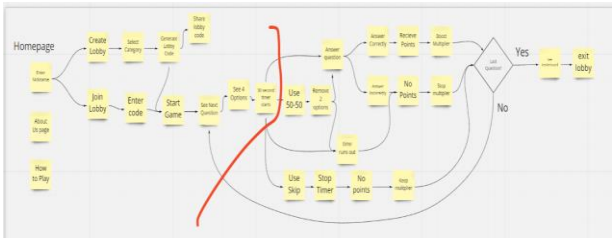

Figure 4 – User-Flow chart created on Miro Board

4.4 Branding

The team collectively came up with the Quizz.io brand and documented the entire process on the Miro board. After the initial discussion, each member contributed in suggesting a colour palette for the product. The team then voted on a selection of colours that they felt would best represent the needs of the client, as well as, the feel of the game that the team was trying to achieve. Once the colours were selected, a similar process was carried out for the font.

4.5 Prototyping

4.5.1 Low Fidelity Prototyping

In order to initiate a visual representation from the brainstorming session; paper prototypes were developed by the front-end team [See Figure 5]. This was done collaboratively. Each member of the team sketched a design that they felt would the most suited for the game. The instruction was to incorporate as many features in as much detail you can within a specified time. Once the time was over, each member uploaded their sketches to this Miro board and talked through the best features of their sketches. At the end, the team picked out the best features from all the sketches and came up with a final paper prototype.

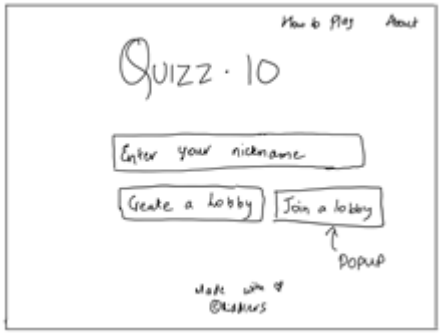

Figure 5 – Paper Prototype of Home Screen

Although these prototypes were extremely basic, the sketches were able to showcase the overall design and possible layout that each member had in mind. One key layout feature that was showcased in one of the designs was to have the Timer in the middle of the screen between the four answer buttons. This was judged to be not only visually appealing, but also created space on the screen for other features such as the help option buttons. Although, the paper prototypes were a fast and efficient method of visualising the design of the product, they lacked the ability to communicate interaction and sounds. Hence, a second stage of prototyping was required to take the initial mock-ups a step further in development.

4.5.2 High Fidelity Prototyping

High Fidelity prototypes are usually created when a solid understanding is established of the design, as they closely resemble the appearance and functionality of the actual finished product [31]. They also act as an effective method of gaining meaningful feedback from users due to the interactivity which can be incorporated into the prototype. Taking this onboard, the front-end team were able to expand on the initial sketches and swiftly moved onto produce high fidelity prototypes [See Figure 6] using Adobe XD [32]. These aimed to provide a more refined design with the inclusion of interaction and sounds enabling a more polished prototype. Hence, this allowed the project team to clearly visualise the design in order to better aid them in planning and being able to efficiently identify key design requirements. The prototypes were specifically paramount during the implementation stage since they enabled the development team to have more focused goal, due to the prototype being utilised as a reference when the front end was being designed. Site navigation was also simulated in Adobe XD, which highlighted the importance of consistency between the different web pages. Thus, helping create a relatively smooth transition from the design stage to the implementation stage.

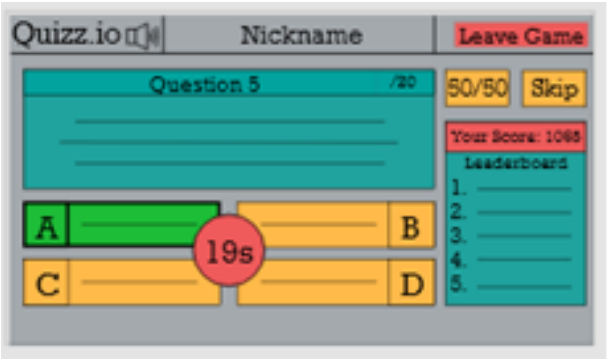

Figure 6 – High Fidelity prototype of Game Screen

4.5.3 Prototype Feedback

The high-fidelity prototype was used during the first round of user testing that was set up as a series of individual video calls held on the Microsoft Teams platform. A total of 5 participants took part at this stage and all were presented with the participant information sheet [See Appendix H] and informed consent form that they signed prior to the call.

During the initial stage, participants were asked to perform a series of actions on the prototype that they accessed in their browser via a provided link [See Appendix H]. Their actions were observed through screen sharing. This helped the researchers evaluate the layout and general ease of use and identify any bottlenecks. No major issues related to the layout were found, however some suggested the lifelines section seemed inconspicuous and the location of the timer was not intuitive, which was taken into consideration when building the actual application. The size of the lifelines section was increased, the buttons were highlighted in yellow, and the timer was moved up to the question section. One of the participants suggested introducing a clear division between the question/answers section on the left-hand side and the lifelines/scoreboard section on the right. This was achieved by adjusting the size ratio of the two sections and increasing the gap between them.

The second stage of each interview was answering open-ended questions to encourage comments regarding user behaviour, their primary goal and opinion [See Appendix H]. The question set was based on User Interview Example Questions from Yale University [33]. All participants had some previous experience with quiz applications, 'Kahoot' and 'QuizDuel' mostly, with the former being employed in various settings from academic use to pub quizzes and the latter serving primarily for entertainment purposes. The participants collectively agreed that web-based application would be their preferred choice as this is what they were already familiar with. There was no clear preference for mobile devices over laptops and desktop computers. The users praised the simplicity of the system, it's colour scheme, intuitive layout and limited options that did not confuse them.

4.5.4 Iteration

Iteration was a core focus of the design process [34]. The front-end team continued to improve the design of the product on a regular basis. This iteration of the designs was based on further research and user feedback. Iteration allowed the team to build core features quite early in the production process, while still allowing minor improvements to be added on an almost-daily basis.

5 Implementation and Testing

5.1 GitHub

GitHub [35] was chosen to be utilised throughout the project as the team felt that it was the best platform for managing and collaborating on a codebase. Additionally, most of the team already had experience with GitHub and were relatively proficient in its use. As the intention at the start of the project was to use all of the features of GitHub, such as GitHub Projects and Issues, but as time progressed, it became evident that issues were being resolved promptly and therefore were not necessarily required to be documented. However, one feature of GitHub which was utilised significantly was branches. As both the front end and back end teams had their own working branches which then branched off into individual features to merge back later. This feature was crucial in maintaining code stability and minimising potential risk.

5.2 Azure

Azure is a public cloud computing platform developed by Microsoft [36]. It is commonly used for building, testing, deploying, and managing applications. Additionally, the main advantages of Azure include its flexibility and scalability as a cloud service for users. Taking this into account and due to coding the API in C#, Azure was used to host the API and incorporate it into our workflow to ensure simple and quick deployment for front-end use.

5.3 Continuous Integration

Continuous Integration (CI) for this project would have been useful if it were set up, however the team had difficulties doing so. The team were planning to use Azure pipelines from Github marketplace for CI, however, when they started setting it up, they could not find premade templates for the frameworks being used. This then resulted in the team writing their own 'yml' file. No team member had experience doing this, so it proved rather difficult to set up. It was eventually decided by the team to not use continuous integration due to the fact that setting it up wasn't viable in the time constraints, and since this was a small project, CI is not something that would have impacted significantly. The team also initially planned to use Cypress framework for end-to-end testing. Overall, the decision to not spend more time trying to set up CI was correct for the team as there were only 3 weeks to produce the fully working quiz application. A key aspect was to ensure that a robust and refined product could be made.

It was decided by the team to make a multi-page application instead of a single page application. There were multiple reasons for this choice. Firstly, majority of the team were not comfortable with the idea of a single page application as few members had experience with this and were unsure as to how this would work. Since the team were confident on making multi page application, this option was selected. In addition, single page applications do not allow many features on one page, hence, multi-page application is better for loading times. With all the research and opinions from team members, the decision was made.

5.4 Implementation

5.4.1 API Implementation

One of the fundamental aspects of the application was to develop and integrate the API. The Back-end team initially intended to offer a CRUD interface [37]

between the web application and the database as this would allow simple and direct Create, Read, Update and Delete functionality for each of the tables. However, the first problem arose when the team tried to connect the API to the database in order for it to execute the required SQL statements. Since the ASP.NET entity model framework was considered to handle the SQL statements – using the controller Scaffolds to generate the important parts of the code that would perform JSON parsing between the web applications request and the API's models. This however, resulted in errors that the team were unable to solve, preventing the connection to the databases server. Due to the complexity of the issue and time constraints that the team faced, it was agreed that the back-end team would write the SQL manually instead. Constructing the SQL statements meant concatenating the model's properties (which parallel the fields in their respective table) with the rest of the SQL syntax. This led to some runtime errors however, due to errant apostrophes and overlooked misspellings, so in the future another method of constructing these strings would be preferable.

```
//setup connection and command
string connetionString = "Data Source=riddlers.database.windows.net;Initial Catalog=quizgame;User ID=team8;Password=b7zYDzh3);";
SqlConnection cnn = new SqlConnection(connetionString);
SqlCommand cmd = new SqlCommand("SELECT * FROM question;", cnn);

cnn.Open();
SqlDataReader data = cmd.ExecuteReader(); //execute command, returning data to a reader

if(data.HasRows == false) //check if any records were returned
{
    data.Close();
    cmd.Dispose();
    cnn.Close();
    return NotFound();
}
```

Figure 7 – Code for SQL Connection

When retrieving data from the database, the models had to be manually constructed from an SQL data reader. While there was some trouble with learning the correct way to use the reader - as certain methods needed calling to prep the data for reading and closing the data reader was required to perform any more operations with the established database connection – this was eventually overcome and a reliable way to implement the reader was understood.

5.4.2 Database Restructuring

At the beginning of the third sprint, the front-end team informed the back-end team that a new method of storing questions would be required. To fulfil this objective, the back-end team restructured the Lobby Table so that it no longer stored the questions. Instead, a Question Table was created, along with the necessary controller class. This utilised a compound key consisting of the lobbyId and questionIndex fields for unique identification and contained methods to get and delete all records belonging to a given lobby. This had to be done as the questions - while initially intended to be large strings that could exist in a single field - needed to be stored as objects themselves with a range of properties, which could not be stored in a single field.

```
// POST: api/Questions/getinfo
// READ
[HttpPost("getinfo")]
public async Task<ActionResult<Question>> GetQuestionSingle(Question q)
{
    //setup connection and command
    string connetionString = "Data Source=riddlers.database.windows.net;Initial Catalog=quizgame;User ID=team8;Password=b7zYDzh3);";
    SqlConnection cnn = new SqlConnection(connetionString);
    SqlCommand cmd = new SqlCommand("SELECT * FROM question WHERE questionIndex = " + q.questionIndex + " AND lobbyId = " + q.lobbyId + ";", cnn);
```

Figure 8 – Restructuring of the database

5.4.3 Lifelines

Lifelines were implemented into the final application in order to add to the players tactical resources. In the quiz, the player has two lifelines: 50/50 and 'Skip'. The 50/50 lifeline randomly removes two incorrect answers, thus leaving the player with a better chance of getting the right answer. The 'Skip' lifeline allows the player to skip their current question without points being deducted, whilst maintaining their current score streak.

5.4.3.1 Skip Lifeline

Implementing the 'Skip' lifeline [See Figure 9] was relatively straightforward and simple, yet a vital feature to the overall gameplay. This was implemented by calling the 'getNextQuestion()' function prematurely to send the player to the next question. The score and streak were not affected as this was dealt with in a separate function. At this stage, it was evident that cheating could be achieved if the relevant data was not updated correctly. Therefore, the Lifeline was disabled and the database was updated with this information to prevent the user from getting extra lifelines by simply refreshing the page.

```
/*
 *  Function to skip a question without losing points or their streak for skip lifeline
 */
skipLifeline() {
  this.lifelineSkip = false;
  document.getElementById("skip").disabled = true; //disable lifeline after 1 use
  this.updatePlayerTable();
  this.getNextQuestion();
},
```

Figure 9 – Implementation of Skip lifeline

5.4.3.2 50/50 Lifeline

The 50/50 lifeline [See Figure 10] was slightly more difficult to implement as it involved knowing what button the correct answer had been placed on to make sure this was not removed. As this information was never stored, the first thing that had to be checked was what button the answer was on and save this information within the function.

```
let answer;

if (document.getElementById("ansOne").innerHTML == this.allQuestions[this.currQuestion].correct_answer) {
  answer = 1;
}
else if (document.getElementById("ansTwo").innerHTML == this.allQuestions[this.currQuestion].correct_answer) {
  answer = 2;
}
else if (document.getElementById("ansThree").innerHTML == this.allQuestions[this.currQuestion].correct_answer) {
  answer = 3;
}
else if (document.getElementById("ansFour").innerHTML == this.allQuestions[this.currQuestion].correct_answer) {
  answer = 4;
}
```

Figure 10 – Implementation of 50/50 Lifeline

The team then used a random number generator [See Figure 11] to get two random button numbers to disable, checking that neither of these were the correct answer and also that the 2nd number was not a duplicate.

```
let random = this.getRandomNum(1,4);
while (random==answer) {
  random = this.getRandomNum(1,4);
}
let random2 = this.getRandomNum(1,4);
while (random2==answer || random2==random) {
  random2 = this.getRandomNum(1,4);
}
```

Figure 11 – Random number generation

It was then the case of disabling the random selected buttons and colouring them and the label background grey to provide a disabled visual cue. The 50/50 lifeline was also similarly set to disabled in the database with a visual cue for this action seen on the screen too.

5.4.4 API Testing

Throughout development of the API, the back-end team performed regular testing of new API functionality. Postman [38], an online service that performs https requests, was used for the bulk of testing. Due to the API being hosted on a cloud server, errors were more difficult to diagnose as exception messages were not openly accessible. To solve this, the back-end team frequently made use of response objects – such as 200 Ok responses – with custom messages attached, which included the exception type and message.

Breakpoints, another common tool for debugging, were also unavailable due to the code running remotely. This resulted in the values of relevant variables often being included in the responses as well.

5.4.5 Styling

Buefy [39] is a styling tool that creates lightweight UI components for Vue.js. This framework was chosen due to a member of the front-end team having had previous experience with it. Alternative options were researched but the team ultimately settled on Buefy due to it's compatibility with Vue and ease at which it could be learned. A downside of Buefy was that it is a relatively unknown software so support and troubleshooting resources were not readily available. This is something the team would take into consideration for future projects.

Buefy provides easy-to-understand documentation for all of their components as well as a number of code examples to take inspiration from. All the base components were easily customisable with CSS so they could be moulded to fit the requirements. For page formatting the team made use of the Buefy Grid and Tile systems. These systems provided a flexible approach to laying responsive components on the different pages of the application.

The original mock-up that was created in Adobe XD was kept in mind from the beginning of the front-end implementation. Components were designed based on the original responsive mock-up so the final product followed those plans very closely. The design was taken into consideration from the beginning to ensure the team did not need to shoehorn functionality into an appropriate layout later in the process.

Key functionality was often highlighted to the player through visual cues during the game. For example, disabling the lifeline buttons once they were used. The buttons would lose colour and turn translucent to indicate to the player these could no longer be used. Similarly, buttons were coloured green for correct answers and red for incorrect answers. These styling choices improved the usability of the game and ensured that the player knew what was happening.

Uniformity between pages was an important point for the team. Luckily, Vue.js has a feature where individual components can be created in component files and then imported into the page files. This helped to prevent code duplication and also discrepancies between pages.

5.4.6 Prevention of cheating

Once the game had basic functionality the front-end team started to look into any bugs or vulnerabilities that could be fixed. A major problem that was identified was that the player could spam the correct answer button in the 2 second delay it takes to change to the next question. This allowed them to accumulate a lot of points very quickly. To combat this issue, all buttons were temporarily disabled during the 2 second wait time to ensure that nothing could be tampered with. A function was created for this to reduce code duplication.

```
disableAnswerButtons() {
  document.getElementById("ansOneA").disabled = true;
  document.getElementById("ansTwoA").disabled = true;
  document.getElementById("ansThreeA").disabled = true;
  document.getElementById("ansFourA").disabled = true;
},
```

Figure 12 – Disabling the answer buttons

Another issue that was identified was that the player could gain extra lifelines by refreshing the game page mid-quiz. This clearly gave an unfair advantage, so lifeline fields were created in the player table of the database so this information could be stored and checked when a quiz was re-loaded.

A Timer was introduced to prevent the player from looking up the answers to their questions while also adding some pressure to make the quiz more exciting.

The front-end team printed a lot to the console during development for troubleshooting and testing. The team made sure to remove all these comments once finished to ensure the player could not see the correct answer by inspecting the page.

5.4.7 Game Music

Music was implemented into the website so that the user could be more engaged with the game. To achieve this, an upbeat and fun track was needed. The team came up with different ideas for music and eventually they agreed to use Clear Day by Benjamin Tissot. This music is royalty free which meant it could be used in the game for free without running into any issues. Bensound was credited in the about page of the website. MP3 format was chosen for the audio due to most browsers supporting it.

To implement audio in Nuxt, the team had to extend Webpack to load audio files. The audio files needed to be processed by file-loader. This was included in the default Webpack configuration, however, to set it up to handle audio files, some code was required to be added to the nuxt.config.js file [Figure 13]. Before this, the team tried to use Howler audio library, however this was difficult to set up and there was a lack of information online regarding how to configure this in Nuxt. Hence, the team abandoned this idea.

```
76      extend(config, ctx) {
77        config.module.rules.push({
78          test: /\.(ogg|mp3|wav|mpe?g)$/i,
79          loader: 'file-loader',
80          options: {
81            name: '[path][name].[ext]'
82          }
83        })
```

Figure 13 – Audio file handling

The member of the group that was implementing game music struggled with getting the audio to run throughout the game without restarting from the beginning on each page. The main problem was the team's approach of doing a multi-page application instead of single page application. This resulted in it being difficult to make work correctly, so extensive research had to be carried out. From the team's research, one solution that was discovered was to use cookies to save audio information. However, the issue with this solution was that using cookies would require the team to ask the user for consent. It was decided from the very beginning that cookies should not be used unless it is necessary. Another solution was to use iframes or popups for the audio. This too, however, wasn't a suitable solution as popups are considered bad for user experience and iframes bring security risks. Eventually, one team member came up with the idea of using audio in a layout file. This solved the issue the team were having and the music was able to run continuously from the homepage to the questions page. The only area where it restarted was the results page. However, this was deemed as not being a major issue as during majority of the game the music was able to play continuously. To control the music, a toggle button was required to be implemented. This was quite a simple task as this could be done in JavaScript and the trigger would be the user clicking the sound icon. In addition, the music auto plays when the page loads.

5.4.8 Game Sounds

Game sounds were implemented in this application as it informs the user if they have answered correctly or incorrectly while building a sense of interactivity within the website. The team decided to implement this in the game page instead of creating a separate component file for this since it would only be used in this page. The JavaScript code to control when this audio plays was added to the checkAnswer method. This meant that when the answer is checked, the correct audio would play depending on whether it is correct or incorrect. The team used royalty free sounds for this feature too.

5.4.9 Unit Testing

In this project, unit testing was used to reduce defects in newly developed features and reduce bugs when changing existing functionality [40]. It helped verify the accuracy of each unit by executing these small snippets of code in isolation to track down bugs [41]. It was acknowledged that development with unit testing would take longer however, the main aim of this project was to produce a high-quality product as a team in three weeks, so unit testing was required to ensure this. It contributed to producing a high-quality product by reducing the chances of having bugs. Also, it was a form of documentation because it expressed how the software is supposed to work. This was beneficial for supporting teamwork as it helped others in the team to understand how certain components were expected to function. The team decided that it was only necessary to test the component files as most of the research done by the team pointed out that testing every line of code is not necessary. Additionally, the short deadline of the project meant that testing every line of code was not a practical endeavour.

The team had to decide on what unit testing framework to use. The Nuxt framework came with options to add Jest [42] or Mocha [43]. Jest was chosen instead of Mocha because it was a fast testing library due to its parallel testing and the fact that it came with built in mocking and assertion abilities. Hence, the main reason for selecting Jest being that there is no extra configuration to do, which saved the development team a lot of time, and this time meant that other important aspects of the game were perfected. In addition, one of the team members had experience with Jest, so if another member of the team had difficulties, there would always be someone to consult. Since the team was tight on time and the project was not particularly large scale, Jest was found to be adequate as it worked out the box [44] and the team did not require the customisation that Mocha offered.

For each component there was a series of tests that executed parts of the code in isolation. For example, in the image below it can be seen that the game music component is extensively tested. Each method in the component file was tested as well as the component to see if it was a Vue instance. These test files for each component helped other developers in the team to understand how each part of the component works, especially with the descriptions at the start of each test.

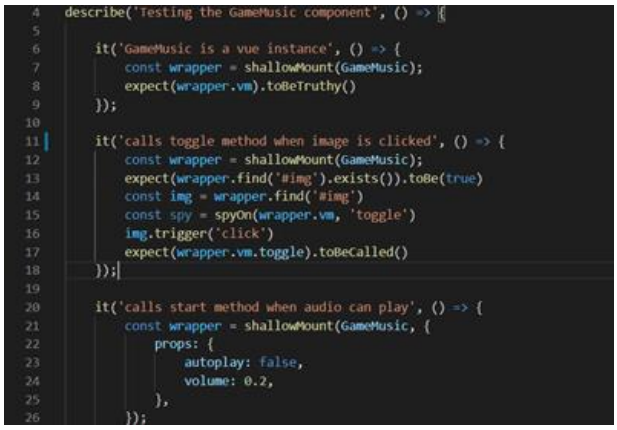

Figure 14 – Code snippet for Unit Testing

5.4.10 Commenting Code

Commenting code was extremely important in this project, especially since it was a large team. Comments helped improve teamwork because when reading someone else's code, it helps to understand how their code works or how they solved a problem. This also allowed the team to work more effectively due to less time spent trying to understand someone else's code. Furthermore, a benefit of using comments in the code is that experimentation can be done. Team members commented out working code so they were able to experiment to find an easier solution to a problem.

5.4.11 Code Review

Code reviewing is another important software engineering practice that was used for this project. It was done during every pull request, so if there was a problem the reviewer would add a comment on the merge request. Code reviews helped to fix a lot of problems in the code. Most of the code reviews in the team helped to improve readability of the code and reusability. Having a team member to review code means there is fresh set of eyes to pick out features missing and identify bugs. This improved the project quality and ensured that all the requirements are met.

5.5 User Manual

Although the finished product should be easy to navigate and understand, it was supplemented by the creation of a user manual [See Appendix G]. The team felt it was appropriate to create this in case any problems or misunderstandings arose as this would enable the users to easily refer to the user manual during their time using the website. Also, the manual includes instructions on how to play the game, as well as acting as a general guide for any users.

6 Evaluation

The team made considerations for user-centred design thought the project lifecycle. Weekly meetings with the client and two-stage user testing helped identify numerous issues that were then mitigated during development as indicated in the Prototype Feedback section. The aspects and features that could not be easily incorporated into the design were noted as future considerations.

6.1 Final Evaluation

The final evaluation comprised of two parts: usability testing and opinion gathering, in which 6 participants took part, including 2 Computing students as the team decided it would help evaluate technical aspects of the system thanks to the students' deeper understanding of design and development principles. This allowed for gathering specific input on robustness of the user interface as well as website responsiveness and overall quality of the application.

The participants were asked to take part in a video call held on the Microsoft Teams platform. Some of them had already taken part in the prototype evaluation stage of testing, but the ones that had not were presented with the participant information sheet [See Appendix H] and the informed consent form that they signed in advance of the call.

As the team were unable to deploy the application on a web server, the researcher ran the application locally on their machine. During the virtual meeting, participants were given control over the researchers' computer using MS Teams feature, in order to be able to freely interact with the system. The participants were presented with a set of tasks to do based on the user stories that guided the team during development [See Appendix H]. The researcher observed them using the system through screen sharing. None of the users had issues when navigating the application, although two mentioned that the start screen reminded them of a search engine and the long nickname input box with two buttons right below it resembled a search bar, however they found the placeholder text useful as it clearly indicated the real purpose of the box.

The second part of the call was conducted in the form of an interview in order to better understand how well the system supported the users' needs, what the participants liked and disliked about the current state of the application, and to welcome any general comments [See Appendix H]. As a whole, Quizz.io was found to make an enjoyable offering that meets the needs of the client and the end users alike, and that could serve a variety of purposes, with entertainment and expanding knowledge named as the leading ones. What the participants particularly enjoyed was the happy, upbeat music and

sound prompts helped fully immerse into the game. Diverse and original questions enhanced the overall experience, although the difficulty level seemed to be inconsistent across categories. Increasing the relative size of the lifelines box and incorporating bright yellow into the design was said to make the lifelines stand out and help assess how many were still available. One of the users liked the fact that they did not have to waste time reading instructions and figuring out how things worked, as creating and joining lobbies was intuitive and much clearer than in the prototype. Overall, the website proved to be responsive and to have a short loading time.

After the final presentation and demo, the client praised the game being asynchronous, referring to it as a "pretty unique" feature that distinguished Quizz.io from other solutions to their problem. The client liked the design, specifically the "brilliant colour scheme", as well as the music and the option to turn it on and off. Overall, the client was very satisfied with how the team managed to implement the key features from the brief.

6.2 Limitations

Finally, the team realised that even though it is recognized that best results come from testing up to 5 users, the findings also indicate that in order to discover all usability issues, one needs to test with at least 15 users [45]. At both evaluation stages, the number of participants were restricted due to time constraints and finite resources, hence the findings presented bear limitations with regard to diversity of experience, agility, age, background and general ability.

Accessibility testing was carried out throughout the project to comply with WCAG standards for colour-blindness and colour contrast ratios. Due to the nature of the agile process, some of these could not be implemented in the final product during these three sprints. Even though the product supports keyboard navigation and selection, certain keyboard functionalities like pressing the correct letter to select the answer, are missing from the final product.

7 Description of the final product

The final product, Quizz.io, is multiplayer cross-browser web-based quiz application for entertainment purposes. It is simple, light-hearted and moderately competitive thanks to its lobby-based structure. It has not been deployed yet and is run locally. Presented below [Figure 15] is the start screen of the application, with direct links to 'How to Play' and 'About Us' sections.

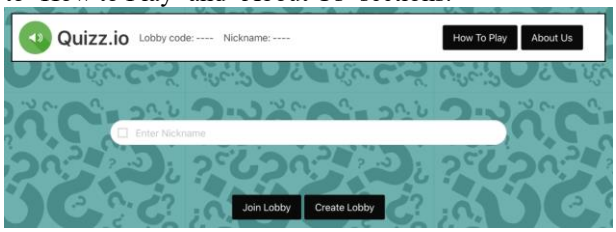

Figure 15 – Login Screen

From the homepage, the player enters the game by setting their nickname and providing a code of an existing lobby or by creating a new lobby. The game has replay ability allowing the player to re-join the same lobby with a different username to try and beat their high score. Upon joining the lobby, the player can see the current leaderboard and nicknames of other players in the lobby as pictured below [Figure 16].

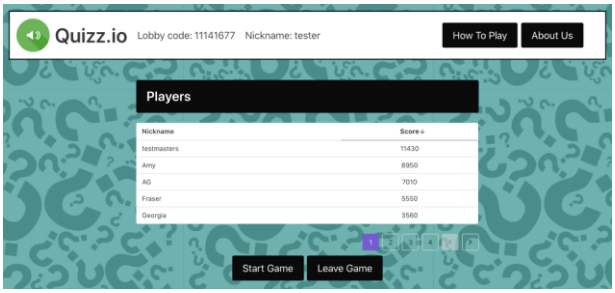

Figure 16 – Lobby Screen

Randomly selected categories for each round add variety to gameplay, with the last set of questions being general knowledge by default. The questions also vary by difficulty, with the game starting with random category easy questions, moving onto random category medium difficulty questions and finally hard general knowledge questions. Lifelines help maintain the positive experience, helping the players avoid losing points. An example question screen is presented below, with difficulty and category displayed above the question [Figure 17].

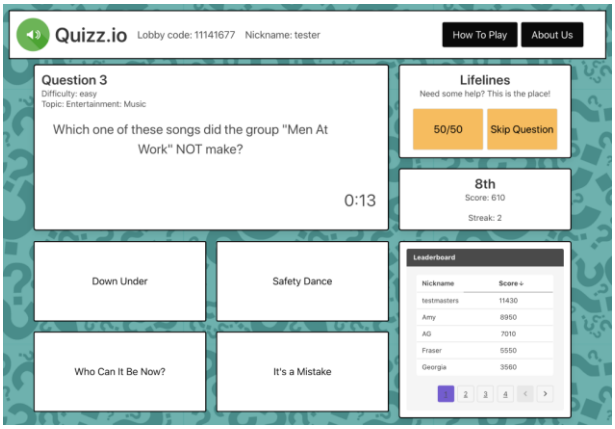

Figure 17 – Game Screen

A correct answer streak was also implemented. The more consecutive questions are answered correctly, the more points the player gets. The player can check their current score, their place and their answer streak at all times [See Figure 18]. The streak multiplier is reset after wrong answer selection.

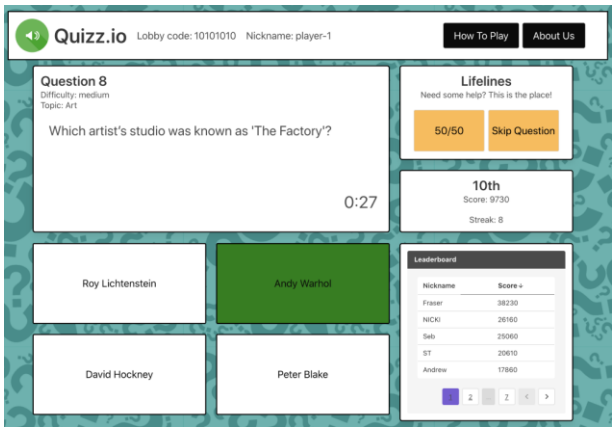

Figure 18 – Correct answer selection

One of the key features of the application is asynchronous play that allows flexibility, so that players can come in and out of their lobby and finish their session whenever they please. This also allows different players to play the game at their convenience rather than everyone playing at the same time. However, live leaderboards ensure users can compete in real time if they prefer to and get immediate feedback on everyone's performance [See Figure 19].

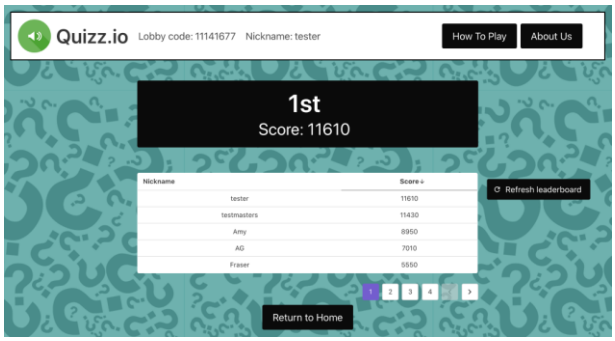

Figure 19 – Leaderboard screen

The colours, the sounds, the font, and the gameplay are all designed to be fun, entertaining, and calm. The questions are also meant to be simple yet challenging to a level where the player doesn't feel stressed or bored.

8 Summary and Conclusions

The goal of the project was to create a quiz application, this and more has been achieved by the team. By the end of the three-week period, the team created a ready-to-be-deployed application with some of the suggested optional features, i.e. multiplayer, lifelines and a points system.

During development the team faced a few issues, most of them appearing in week 2 with front-end development and the need to adapt to a new framework. The front-end team practiced using the framework on their own for them to have enough confidence to move on with the rest of the project. Once the team had gotten used to the new framework, development continued smoothly with some minor coding and design compatibility issues being addressed throughout the week by the API/backend team.

The team went over a structure change at the beginning of week 2, which was predicted and planned for during week 1, causing minor inconvenience as the project manager facilitated the restructuring during the most suitable time before development commenced. The need for two different team structures during the first and last two weeks was justified by the requirements and necessary in order to maximise team member efficiency.

The most important reason for successful project completion had been its organisation during week 1 and the preplanning done during it. It allowed the team to have a plan for every week and even allowed other team members to adjust to members away due to illness with relative ease. Morning scrums also ensured that all team members started working at the same time and that issues could be solved immediately after it. Introducing this routine helped the team keep organised throughout the week.

What the team were most proud of was their extraordinary communication and organisation as it allowed the members to branch out into fields they were unfamiliar with or not completely confident in and receive necessary support from the rest of the group in case of problems. Keeping an open mind and flexibility were key in adjusting to rapid design and team structure changes, and the proactive attitude persistent thought the project empowered the team, mitigating risks as they arose.

8.1 Future Considerations

Based on the designs and user feedback, the team have identified a number of considerations for future development.

The first and foremost is increasing accessibility. During the research the team carried out readability tests on colour contrast ratios and colour blindness filters. Both of those tests were passed by the prototype. However, the team could not implement some of the colour blindness features in the final product due to a lack of time. Based on WCAG standards, the application should be mobile responsive and keyboard support for answer selection should also be included.

As a result of testing, the team identified allowing players to select or create their own question sets as it was highly desired by the users, so was setting the lobby difficulty and choosing categories tailored to people's likes and interests. The users indicated that when playing alone, they would prefer to join random or public lobbies to fully enjoy the experience. Adding features for a more competitive game mode where the players are not able to see the correct answers or the leaderboard up until the end could also help expand the range of possible applications of the software.

Acknowledgments

The team would like to thank Dr. Brian Plüss and Hasith Nandadasa for their guidance and prompt feedback throughout the project.
The team would also like to thank all of the participants that took part in the user testing, as they provided vital feedback and constructive criticism which improved the final product.

References

[1] - Golby, J. (2020). Lockdown may be over, but you'll never take my Zoom quiz away. [online] the Guardian. Available at: https://www.theguardian.com/commentisfree/2020/jul/01/i-have-a-vested-interest-in-life-not-returning-to-normal-my-friday-night-zoom-quiz [Accessed 8 Oct. 2020].

[2] - Atlassian (2020). What is Agile? | Atlassian. [online] Atlassian. Available at: https://www.atlassian.com/agile [Accessed 8 Oct. 2020].

[3] - Bushey, R. (2013). How To Play The Fastest-Growing iPhone Game Of All Time - Business Insider. [online] Business Insider. Available at: https://www.businessinsider.com/quizup-2013-12?r=US&IR=T#this-is-how-the-questions-appear-on-your-phone-tap-once-to-answer-the-question-before-the-other-gamer-does-6 [Accessed 8 Oct. 2020].

[4] - Mitroff, S. (2014). QuizUp review: Social trivia that keeps you coming back. [online] CNET. Available at: https://www.cnet.com/reviews/quizup-android-review/2/#:~:text=QuizUp%20(Android%20%7C%20iOS)%20is,more%20points%20and%20level%20up. [Accessed 8 Oct. 2020].

[5] - Kahoot.com. 2020. [online] Available at: https://kahoot.com/ [Accessed 8 October 2020].

[6] - Interactive, C., 2020. Quizwitz. [online] QuizWitz.com. Available at: https://www.quizwitz.com/en/ [Accessed 8 October 2020].

[7] - Software Advisory Service (2020). Introduction to Kahoot: Learning by Gamification. [online] Softwareadvisoryservice.com. Available at: https://www.softwareadvisoryservice.com/en/blog/introduction-to-kahoot-learning-by-gamification/ [Accessed 9 Oct. 2020].

[8] - Gamespot Staff (2001). QOTW: What Makes an Online Game Successful? [online] GameSpot. Available at: https://www.gamespot.com/articles/qotw-what-makes-an-online-game-successful/1100-2774083/#:~:text=The%20element%20that%20makes%20online,or%20interact%20with%20other%20players. [Accessed 9 Oct. 2020].

[9] - Games. (2018). What Makes a Good Game? [online] Available at: https://serc.carleton.edu/introgeo/games/goodgame.html [Accessed 9 Oct. 2020].

[10] - Yan, J. and Randell, B. (2009). An Investigation of Cheating in Online Games. IEEE Security & Privacy Magazine, 7(3), pp.37–44.

[11] – Gilbert, N. (2020). How many active video gamers were there in 2019? There were 2.47 billion video game players worldwide in 2019. [online] Financesonline.com. Available at: https://financesonline.com/number-of-gamers-worldwide/ [Accessed 9 Oct. 2020].

[12] - GameRefinery. (2016). Know Your Game's Competitors and Target Audience. [online] Available at: https://www.gamerefinery.com/games-competitive-landscape-categorize/ [Accessed 9 Oct. 2020].

[13] - Juno Web Design. 2020. Good VS Bad Web Design: How To Do It Right | Juno. [online] Available at: https://www.junowebdesign.com/good-vs-bad-web-design/ [Accessed 9 October 2020].

[14] - Kensquiz.co.uk. (2010). Ken's Quiz Site. [online] Available at: https://www.kensquiz.co.uk/ [Accessed 9 Oct. 2020].

[15] - W3.org. 2020. Understanding Success Criterion 1.4.3: Contrast (Minimum). [online] Available at: https://www.w3.org/WAI/WCAG21/Understanding/contrast-minimum.html [Accessed 12 October 2020].

[16] - Domantas G (2018). What is HTML? The Basics of Hypertext Markup Language Explained. [online] Hostinger Tutorials. Available at: https://www.hostinger.co.uk/tutorials/what-is-html#So8230What-is-HTML [Accessed 12 Oct. 2020].

[17] - Lindsay Kolowich Cox (2018). Web Design 101: How HTML, CSS, and JavaScript Work. [online] Hubspot.com. Available at: https://blog.hubspot.com/marketing/web-design-html-css-javascript [Accessed 12 Oct. 2020].

[18] - melaloo. (2015). Intiwebsites for dummies: What are HTML, CSS, JavaScript and PHP? [online] Available at: https://melaloo.wordpress.com/2015/07/12/intiwebsites-for-dummies-what-are-html-css-javascript-and-php/ [Accessed 12 Oct. 2020].

[19] - Visual Studio. 2020. Visual Studio IDE, Code Editor, Azure Devops, & App Center - Visual Studio. [online] Available at: <https://visualstudio.microsoft.com/> [Accessed 12 October 2020].

[20] - Netbeans.org. 2020. Welcome To Netbeans. [online] Available at: <https://netbeans.org/> [Accessed 13 October 2020].

[21] - Vuejs.org. 2020. Vue.Js. [online] Available at: <https://vuejs.org/> [Accessed 14 October 2020].

[22] - Reactjs.org. 2020. React – A Javascript Library For Building User Interfaces. [online] Available at: <https://reactjs.org/> [Accessed 14 October 2020].

[23] - Angular.io. 2020. Angular. [online] Available at: <https://angular.io/> [Accessed 14 October 2020].

[24] - alexchopin (2020). Nuxt.js - The Intuitive Vue Framework. [online] NuxtJS. Available at: https://nuxtjs.org/ [Accessed 14 Oct. 2020].

[25] - Rigby, D.K., Sutherland, J. and Takeuchi, H. (2016). Embracing agile. Harvard business review, 94(5), pp.40-50. [online] Available at: https://hbr.org/2016/05/embracing-agile [Accessed 18 Oct. 2020]

[26] - Da Silva, T.S., Martin, A., Maurer, F. and Silveira, M. (2011). User-centered design and agile methods: a systematic review. In 2011 AGILE conference (pp. 77-86). IEEE. [online] Available at: https://www.researchgate.net/publication/224256376_User-Centered_Design_and_Agile_Methods_A_Systematic_Review [Accessed 18 Oct. 2020]

[27] - Slack (2020). Where work happens. [online] Slack. Available at: https://slack.com/intl/en-gb/ [Accessed 18 Oct. 2020].

[28] - Teamgantt.com. (2019). RACI Chart Definition, Template, & Example | TeamGantt. [online] Available at: https://www.teamgantt.com/blog/raci-chart-definition-tips-and-example [Accessed 7 Oct. 2020].

[29] - Usability.gov. (2020). User-Centered Design Basics | Usability.gov. [online] Available at: https://www.usability.gov/what-and-why/user-centered-design.html [Accessed 19 Oct. 2020].

[30] - Ganesh Krishnan R (2019). Using the power of familiarity in design - UX Collective. [online] Medium. Available at: https://uxdesign.cc/familiarity-in-design-70df1979f80 [Accessed 20 Oct. 2020].

[31] - Nielsen Norman Group. (2016). UX Prototypes: Low Fidelity vs. High Fidelity. [online] Available at: https://www.nngroup.com/articles/ux-prototype-hi-lo-fidelity/#:~:text=High%2Dfidelity%20interactivity%20frees%20the,be%20affected%20by%20human%20error. [Accessed 20 Oct. 2020].

[32] - Adobe.com. (2020). UI/UX design and collaboration tool | Adobe XD. [online] Available at: https://www.adobe.com/uk/products/xd.html [Accessed 22 Oct. 2020].

[33] - Yale University. 2020. User Interview Example Questions. [online] Available at: https://usability.yale.edu/understanding-your-user/user-interviews/user-interview-example-questions [Accessed 18 Oct. 2020]

[34] - Smartsheet. (2019). All about the Iterative Design Process | Smartsheet. [online] Available at: https://www.smartsheet.com/iterative-process-guide [Accessed 25 Oct. 2020].

[35] - GitHub. (2020). Build software better, together. [online] Available at: https://github.com/ [Accessed 25 Oct. 2020].

[36] - Microsoft.com. (2020). Cloud Computing Services | Microsoft Azure. [online] Available at: https://azure.microsoft.com/en-gb/ [Accessed 25 Oct. 2020].

[37] - Stackify. (2017). What are CRUD Operations? Examples, Tutorials & More. [online] Available at: https://stackify.com/what-are-crud-operations/ [Accessed 25 Oct. 2020].

[38] - Postman. (2020). Postman | The Collaboration Platform for API Development. [online] Available at: https://www.postman.com/ [Accessed 25 Oct. 2020].

[39] - Buefy. (2020). Buefy: lightweight UI components for Vue.js based on Bulma. [online] Available at: https://buefy.org/ [Accessed 25 Oct. 2020].

[40] - A Software Testing Company. (2019). Performance Lab. [online] Available at: https://performancelabus.com/unit-testing-importance/ [Accessed 25 Oct. 2020].

[41] - Ekaterina Novoseltseva (2019). 8 Benefits of Unit Testing. [online] dzone.com. Available at: https://dzone.com/articles/top-8-benefits-of-unit-testing#:~:text=Unit%20tests%20detect%20changes%20that,accuracy%20of%20the%20each%20unit [Accessed 25 Oct. 2020].

[42] - Jestjs.io. (2017). Jest · Delightful JavaScript Testing. [online] Available at: https://jestjs.io/ [Accessed 25 Oct. 2020].

[43] - Mochajs.org. (2020). Mocha - the fun, simple, flexible JavaScript test framework. [online] Available at: https://mochajs.org/ [Accessed 25 Oct. 2020].

[44] - Wheeler, K. (2018). Jest vs Mocha: Which Should You Choose? - Noteworthy - The Journal Blog. [online] Medium. Available at: https://blog.usejournal.com/jest-vs-mocha-whats-the-difference-235df75ffdf3 [Accessed 25 Oct. 2020].

[45] - Nielsen, J. (2000). Why you only need to test with 5 users. [online] Available at: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/ [Accessed 18 Oct. 2020]

Appendices

Please see the appendices directory provided within submission

Appendix A

See ‘A – Competitor Analysis’ directory

Appendix B

See ‘B – Product Backlog’ directory

Appendix C

See ‘C – User Stories’ directory

Appendix D

See ‘D – Sprint Backlogs & To-Do List’ directory

Appendix E

See ‘E – RACI Chart’ directory

Appendix F

See ‘F – Miro Board’ directory

Appendix G

See ‘G – User Manual’ directory

Appendix H

See ‘H - User Testing & Evaluation’ directory

Appendix I

See ‘I - Source Code’ directory

Appendix J

See ‘J - Team Meeting Minutes’ directory

Appendix K

See ‘K - Presentation’ directory