A mobile app to aid self-study in subjects covered by the Kent Test

Matthew Shore
UP879148

School of Computing
Final-year Project
PJE40

Abstract

The rise in smartphones and other portable devices has led to many young people using these devices for many aspects of their lives. Education is one such area that has been greatly enhanced by technology, from classroom learning equipment to mobile apps which can be used to aid learning. This project aims to create a mobile application which provides a set of learning tools that can be used for self-study for materials aimed around the Kent Test. The Kent Test is a series of exams that students take at the end of primary school that determine if the student is able to attend a selective secondary school. The motivation for this is because the client, who runs an education company, has identified a group of people who would be interested in such an app. Currently, there is no mobile app that exists for this syllabus that meets the needs of the client. This report covers the research, design, requirements elicitation, implementation and testing of the app and will come to a conclusion as to whether or not the initial goals of the project were met. Furthermore, the report suggests areas of future research or development that the app could take.

Table of Contents

1.0 Introduction

1.1 Project Context & Problem Statement

The Kent Test is a series of three exams that students can choose to take at the end of primary school education in Kent. These exams test the student's knowledge in Maths, Verbal Reasoning and Non-verbal Reasoning. The purpose of these exams is to assess suitability for selective secondary school places. Consequently, the exams are very important to the students and there is a great deal of effort and pressure to succeed in them.

The client, who runs an education company based in Kent, would like a mobile app to be made to cover the syllabus of the Kent Test. The main reason for this is because they have thousands of paper-based questions which they would eventually like to be accessible digitally. Furthermore, many of the client's customers have expressed interest in such an app as there seems to be a lack of an existing app that can perform all the functionality required by the client. This means that there is a potential gap in the market that the client would like to take advantage of.

1.2 Project Aims

The project aims are split into two sections. Firstly, there is the high-level project aim which is the general goal that the project is aiming to accomplish. Secondly, there are the milestones, which are more specific objectives that the project can be considered a success if they are met.

High-level project aim: To create an educational mobile app, to aid individual learning for the Kent Test.

Project milestones:

- The app should be able to run on Android and iOS devices.
- The app should have accessibility considerations.
- The user should be able to take an assessed test in a chosen subject.
- The user should be able to take a non-assessed quiz in a chosen subject.
- There should be a feature that allows users to view their progress.
- The app should be easy and intuitive to use and there should be functionality to teach users about how to use the app.
- The user should be fully informed of what data the app is using and should be able to erase any stored data at any time if they wish.

1.3 Project Assumptions and Constraints

This section details the technical/non-technical assumptions and constraints for the project. It provides brief definitions for each term so that proper context can be given. Additionally, the section provides information about how the constraints and assumptions were obtained.

1.3.1 Definitions

Technical assumptions: These are technical factors that are assumed to be true, for example, the type of operating system that the user will be using.

Non-technical assumptions: These are non-technical factors that are assumed to be true, for example, the availability of the client for feedback.

Technical constraints: These are technical design decisions that impose a limit or restriction upon the system. For example, operating system requirements or storage capacity limits.

Non-technical constraints: These are non-technical design decisions that impose a limit or restriction upon the system. For example, time and availability.

1.3.2 Elicitation

The technical assumptions and constraints were obtained through a process of brainstorming. During this process, technical factors such as what programming language the app will be programmed in and the devices that the app will be running on were considered.

The non-technical assumptions and constraints were obtained by a combination of brainstorming and discussions with the client. Non-technical assumptions and constraints are less specific than technical ones meaning they can be harder to obtain.

1.3.3 Results

Table 1.1 - Technical vs Non-Technical Assumptions

Technical Assumptions	Non-technical Assumptions
The user will be using a mobile device that runs either iOS or Android.	Covid-19 will not impact the progress of the project.
React Native Expo can be used to develop the app.	The client will be available for feedback on the project regularly (once per fortnight).
The user will have enough space to store the app on their phone.	The questions will be delivered by the client on time.
There will be no technical issues that could impact development, such as a computer malfunction.	Support from the university and my project supervisor for this project will be given.

Table 1.2 - Technical vs Non-Technical Constraints

Technical Constraints	Non-technical Constraints
The app will have to run offline meaning the method of data storage will have to be compatible with this.	Time is a constraint. There is a limited amount of time to complete this project and there is other, unrelated, work to do that will take up time.
The app needs to be accessible to people with visual impairments so the UI will be limited to allow for this.	Client and stakeholder availability could constrain the project.
No access to a system running macOS which makes developing for Apple devices harder.	

1.4 Report Structure

Table 1.3 - Report Structure.

Section	Description
1.0 - Introduction.	This section provides context and explanation about the project. It also details the high level and specific project aims and explains any constraints and assumptions that have been made.
2.0 - A literature review.	This section aims to assess existing literature on areas relevant to this project. It will also compare this project with similar applications currently being used to see if anything can be learnt from them.
3.0 - Methodology.	This section discusses the project management strategies for this project. This includes the software lifecycle methodology that will be used as well as other project management strategies.
4.0 - Requirements gathering.	This section discusses the requirement gathering techniques that will be used in this project and why requirements are needed. It will list both the functional and non-functional requirements.
5.0 - Design.	This section discusses both front-end and back-end designs for the project and explains the design justifications for each.
6.0 - Implementation.	This section discusses how each component of the application was developed. It explains the logic behind the implementation and discusses any pitfalls and how they were fixed and can be avoided.
7.0 - Testing.	This section discusses the testing for each functional and non-functional requirement of the application. It explains the testing strategy and whether or not each test was successful or not.
8.0 - Evaluation against requirements.	This section discusses whether or not the requirements of the project have been met or not. It evaluates both the high-level project aim and the milestones that were set out in section 1.0.
9.0 - Project conclusions.	This section discusses how well the author felt the project had gone. It also explains potential routes for future work on this project.

2.0 Literature Review

2.1 Introduction

In the past decade, there has been a significant increase in mobile phone usage. (Do, T., & Gatica-Perez, D. 2010) Today more than ever people are using mobile apps to do many different tasks. Education is no exception, with many apps being developed to aid learning. This literature review aims to review current literature and methods for creating a mobile app. As well as to analyze and evaluate products that are currently being used that are similar to this app. With the ultimate goal being to use the knowledge gained from this research to inform the design and implementation process for this project.

This literature review will contain sections based on the IMRaD approach. Following the introduction, there will be a section focused on gathering information and data on important topics of mobile app development as well as similar products. Finally, there will be a discussion and evaluation of the data gathered and how it applies to this project.

2.2 Mobile Operating Systems

2.2.1 Background

An operating system (OS) is a piece of software that manages all the hardware and software on a mobile device. (Fundamentals of operating systems, 2016) Consequently, apps that are built to run on a certain operating system may not be able to run on another operating system. It is relevant when designing a mobile app to determine what operating system to work on.

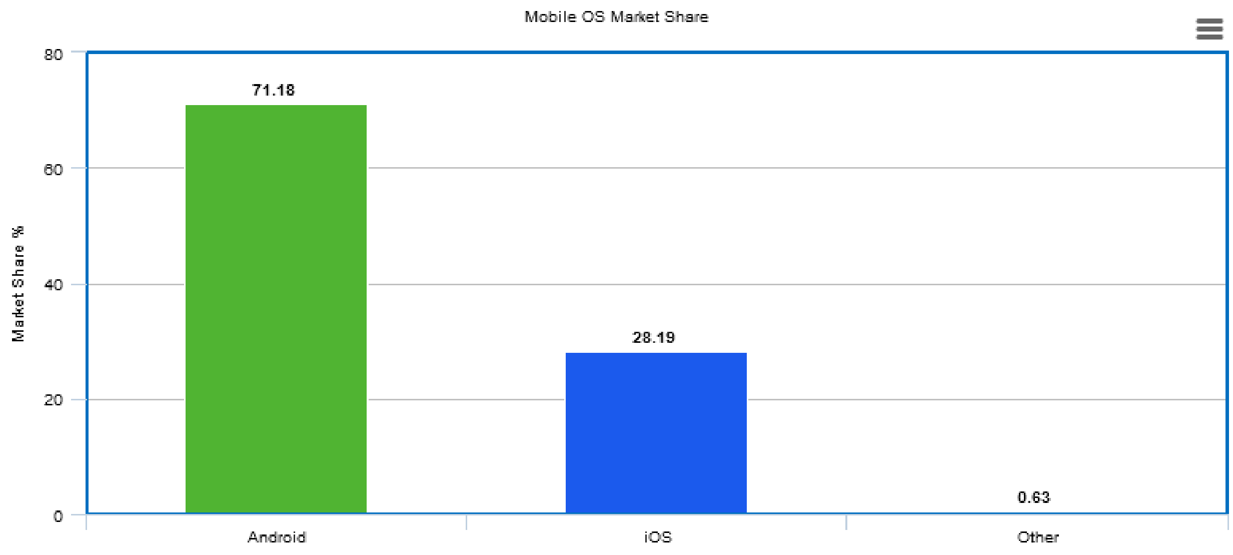


Figure 1 - Mobile OS market share (Mobile OS market share, 2019)

2.2.2 iOS

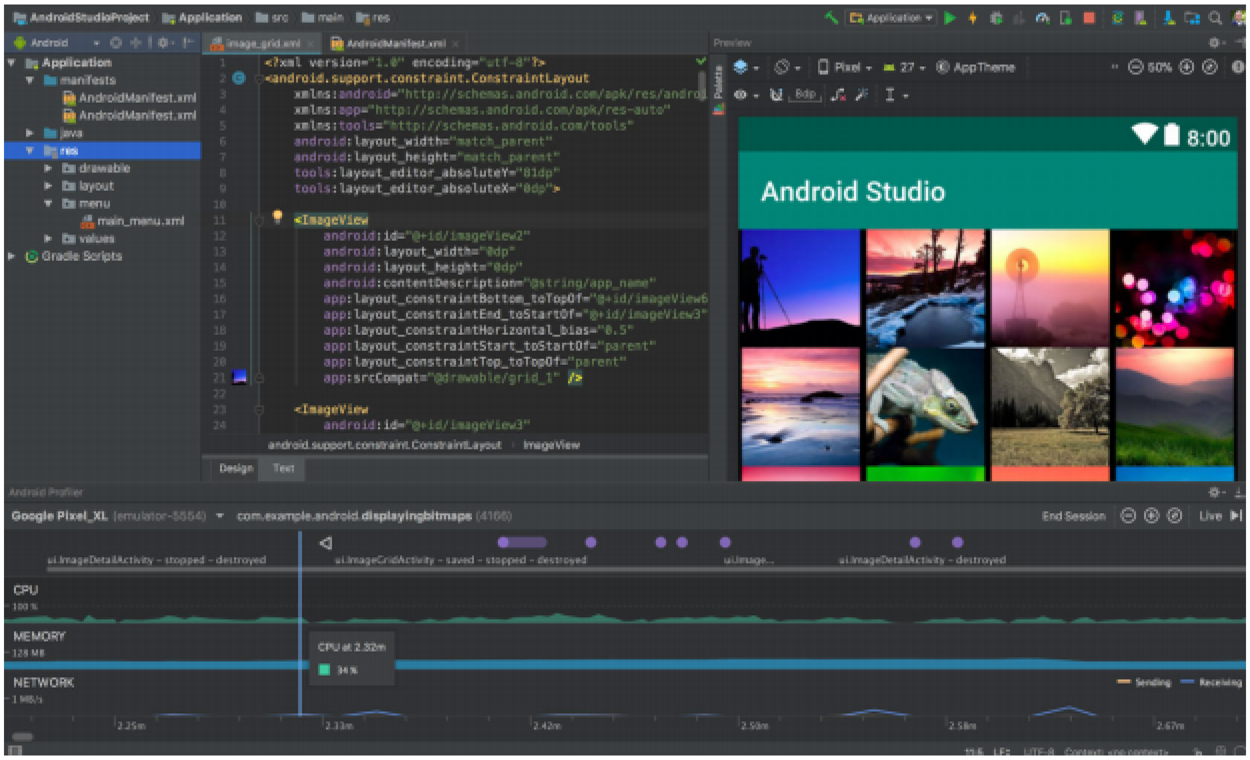
iOS is the operating system developed by Apple with the express purpose of being able to run on their hardware. (Sheikh, et al, 2013) This means that an app developed for iOS would only be able to run on Apple devices which is potentially limiting as only 28.19% of mobile users are running on iOS as shown in figure 1.

XCode is the tool created by Apple that facilitates the creation of apps for iOS and Mac devices. (Xcode Apple Developer, 2021) Another constraint of developing solely for iOS is that XCode requires an Apple operating system to run. Meaning you cannot develop for iOS on a Windows computer unless you are prepared to use emulation software. (Goadrich & Rogers, 2011)

A benefit of developing an app for iOS is that Apple's platform arguably has better security when compared to Android. Each app that is uploaded to the app store is tested for security violations and there is a robust system that prevents tampering with already uploaded apps. (Mohamed & Patel, 2015) The reason for iOS being more secure to develop on is because unlike Android it is not open source and thus it is easier for Apple to ensure greater security, albeit at the expense of being more restrictive to develop on.

2.2.3 Android

Android is the operating system developed by Google and is currently the most popular mobile operating system in the world with a market share of 71% (Mobile OS market share 2019, 2019). The main difference between Android and iOS is that Google has offered Android as an open-source operating system which has meant that significantly more companies and developers use it - hence why it has a larger user base than iOS. (Butler, 2011)



```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/andro
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:layout_editor_absoluteY="81dp"
    tools:layout_editor_absoluteX="0dp">

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:contentDescription="@string/app_name"
        app:layout_constraintBottom_toTopOf="@+id/imageView6
        app:layout_constraintEnd_toStartOf="@+id/imageView3"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/grid_1" />

    <ImageView
        android:id="@+id/imageView3"
```

Figure 2 - Android Studio. (Download Android studio and SDK tools, 2021)

Another benefit of Android is the Android SDK (software development kit), shown in figure 2, which offers good features to make developing an app for Android easy. Some of the features included are libraries, emulators, documentation, sample code, and other tools. (Gandhewar et al, 2010)

The main benefit of developing Android, however, is the market share that Android has currently. In 2020 it is estimated that Android has a market share of 71.18% of overall mobile devices currently in use (figure 1). This is very large and would allow an app developed for Android to reach a larger number of people than an app developed for iOS.

2.3 Hybrid OS Development

2.3.1 Background

Hybrid OS development means developing the app to run on both iOS and Android (and potentially other operating systems). The most obvious way of doing this is to create the app for both operating systems individually. This, however, is slower and less efficient than other methods as it potentially doubles the amount of work needed to create and maintain the applications. (Bosnic, Papp, & Novak, 2016)

2.3.2 Web Application

One way of implementing cross-platform development is to develop a mobile web application. The app can be created using HTML, CSS, and JavaScript and any mobile device with access to a web browser can access it. Because the application is running on the web and isn't installed on the device itself. This means that issues like the application taking up space and requiring updates are not an issue. (Hansson & Vidhall, 2016) The main drawback is that web applications require an active internet connection in order to be able to use. This means that this method is incompatible with the requirements of this project.

2.3.3 Native Scripting Languages

Another way of creating cross-platform applications is to use native scripting. Native scripting languages can use any language which can be interpreted by the device. For example, JavaScript. The native script interpreter executes code while the program is running to native APIs. This is good because it means you can share the same code across different platforms as well as use native components. Native scripting frameworks include React Native, Titanium, and NativeScript. (Hansson & Vidhall, 2016)

2.4 React Native

2.4.1 Background

Released in 2013 by Facebook, React is an open-source Javascript library designed for building user interfaces. The main use of React is for developing dynamic web-based applications that can update without having to reload the page. (Gackenheimer, 2015)

When a regular Javascript application interacts with the Document Object Model (DOM) on a site it is resource-intensive. React solves this solution by creating a "virtual DOM". The virtual DOM is a representation of the DOM but created using Javascript. This structure is shown in figure 3. (Novick, 2017)

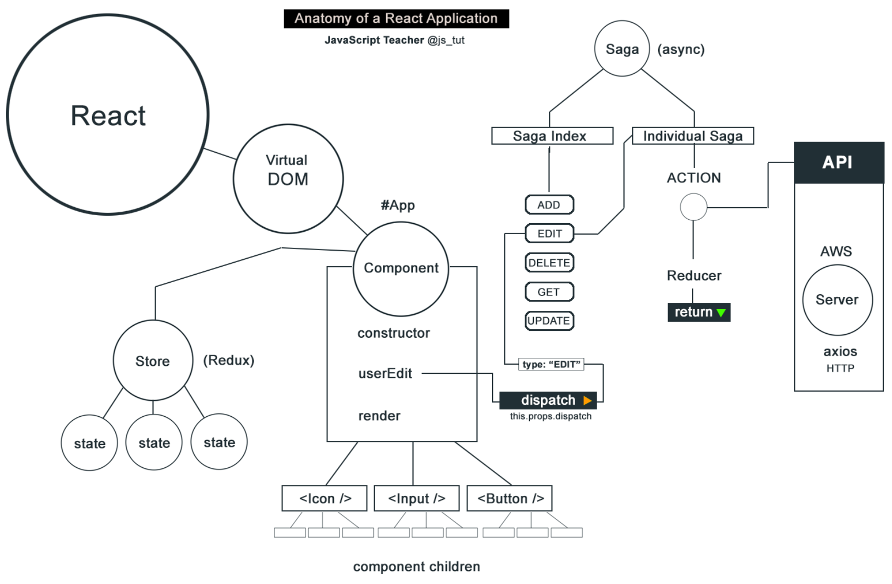


Figure 3 - React structure (Anatomy Of A React Application, 2020)

Consequently, React Native is a Javascript framework that builds and expands upon the React library and allows for the development of cross-platform mobile applications. React Native was developed to solve some common bottlenecks in mobile development. For example, allowing for the same APIs to be used on different platforms. (Novick, 2017) Initially, React Native only supported the creation of iOS applications. However, in September 2015 Android also became supported (Hansson & Vidhall, 2016) meaning it is possible to have a single React Native app for both iOS and Android.

2.4.2 Advantages and Disadvantages of React Native

Table 2.1 An assessment of React Native.

Advantages	Disadvantages
React Native applications can run on both iOS and Android devices.	There are potential bugs as React Native is still in development.
It is easy to create reusable UI components.	There are potential compatibility issues. Components will not be as smooth as with purpose-built OS development tools such as XCode.
React Native applications are fast.	File sizes for React Native applications can sometimes be large.
There is a strong community of support.	
React Native has lots of existing modules and APIs can be used.	

2.4.3 Developing with Expo

Expo is a framework for React Native which helps with developing a React Native application. Expo allows for quick deployment and testing - allowing for the testing of apps without having to set them up on a mobile device which is great for prototyping. (Novick, 2017)

Expo works by running the application on a local server. Any device with the Expo app can connect to this server, by link or QR code, and run the app. (Novick, 2017) This is great because it removes the need for emulation software and allows windows users to test their application on iOS (assuming they have an iPhone) which would not be possible otherwise.

Expo does, however, have limitations and constraints. Firstly, not all iOS and Android APIs are available on Expo currently - potentially reducing functionality for an app. Secondly, Expo can take up a fair amount of space on a device meaning if space is a concern expo might not be appropriate. Additionally, Expo only currently supports Android 5+ and iOS 10+ operating systems meaning any devices running older operating systems are not supported. (Expo Documentation, 2021)

Consequently, Expo is good to use if the project needs quick and easy testing, the project is reasonably large and if you don't own a device with MacOS as it will allow you to develop for iOS devices.

2.5 Flutter

2.5.1 Background

Created by Google, Flutter is an alternative to using React Native - also allowing for the creation of cross-platform apps for Android and iOS devices. (Napoli, 2020)

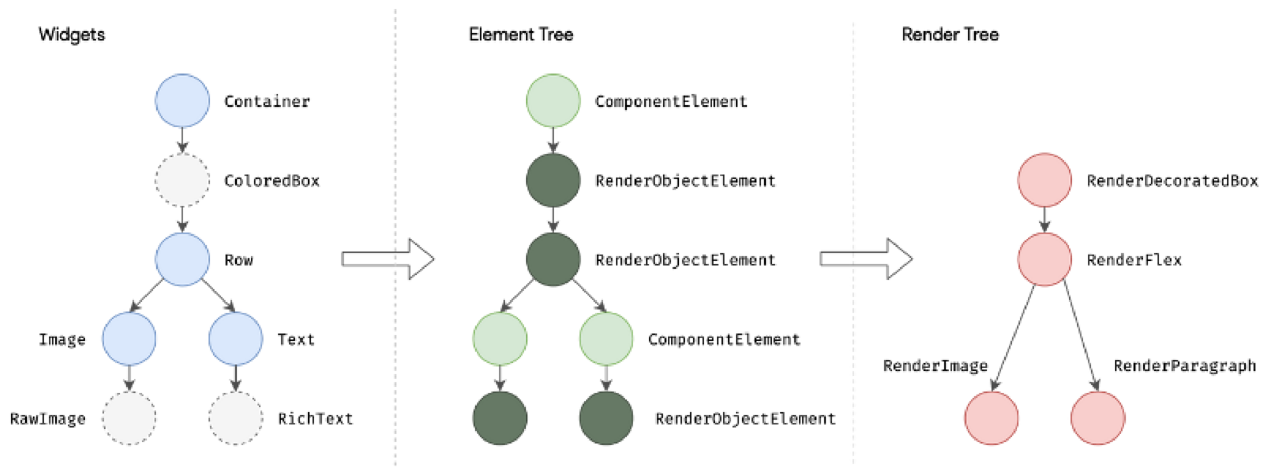


Figure 4 - Flutter structural diagram. (Flutter architectural overview, 2021)

Flutter is built using the Dart programming language which was also developed by Google. The advantage of this is to use Flutter you do not need to learn any other languages. Additionally, Dart is very similar to Javascript meaning people who know Javascript should find it relatively easy to pick up. The main problem with Dart is that it is a relatively new language, still being in development, meaning there is a limited amount of online resources and documentation for it. (Napoli, 2020)

Flutter builds its UI by using widgets, which are similar to components in React. These widgets can be combined to create more complex Widgets. The state of Widgets can also be updated and changed similar to React allowing for dynamic content. This structure is shown in figure 4. (Flutter Documentation, 2021)

2.5.2 Advantages and Disadvantages of Flutter

Table 2.2 - An assessment of Flutter.

Advantages	Disadvantages
Cross-platform app development.	New framework - not as stable as other frameworks.
Cross-platform UI logic.	Large app sizes.
Fast applications.	Less documentation than other frameworks.
Flutter has its custom rendering engine.	

2.6 Quizlet

2.6.1 Background

Released in 2007, Quizlet is an education tool that can either be used on desktop or mobile devices. Quizlet allows users to practice any previously created topic or create their own and practice. Additionally, Quizlet has a premium version called Quizlet Plus which uses machine learning to improve the user experience. (Quizlet, 2021)

2.6.2 Features and Design

Table 2.3 - An assessment of Quizlet's main features.

Feature	Discussion	Rating
Users can create flashcards to help them learn a topic. Users can also use sets of pre-existing flashcards.	The tool is really useful for learning content. It is easy to create and use. There is also lots of pre-made content available.	**9/10**
Users can practice spelling by listening to words spoken out.	The tool is useful for listening exercises but is somewhat limited in scope.	**7/10**
Users can practice writing by finding the right word given an existing sentence.	The tool is good. But the answers have to be very specific which can lead to answers which should be right being marked wrong.	**5/10**
Users can do quizzes which increase in difficulty. Based on the user's answers there are four categories of "new", "seen", "familiar" and "mastered".	The tool is useful for learning, the categorisation of the answers is good for focusing on the areas which are most difficult.	**8/10**
There is a gravity game mode in which users have to answer a question before it reaches a location. If they fail twice the game is over. It gets harder as the game progresses.	The game mode is simple and is good for testing knowledge as it is timed.	**6/10**
There is a matching tool in which users can match the word to the definition.	A basic tool that is useful to practice definitions.	**4/10**
Users can take a standard test on a chosen topic.	A basic test that asks questions and provides feedback at the end.	**5/10**

2.6.3 Discussion

Quizlet is a good application for learning topics. Features like the flashcards and matching could be an interesting place to potentially begin to expand the scope of this project. However, a large amount of the learning material on Quizlet is created by users meaning that the quality is sometimes inconsistent. This is because Quizlet is designed more as a general learning tool. Unlike this project which has the aim of building a purpose-built app for certain topics using pre-made questions.

2.7 Duolingo

2.7.1 Background

Released in 2011, Duolingo is an app that helps the user to learn languages. The app is similar to this app because it is a focused education app that also allows the users to take pre-existing quizzes, tests, and view statistics on their progress. (Duolingo, 2021)

2.7.2 Features and Design

Table 2.4 - An assessment of Duplingo's main features.

Feature	Discussion	Rating
Users can take practice tests on the chosen topic.	Clear menu for choosing which topics the user would like to select.	**8/10**
Users can take tests on the chosen topic.	Clear menu for choosing which topics the user would like to select. Annoying because it is limited by the heart system.	**5/10**
Users can select a tip button for each topic.	Gives clear tips for each topic - this feature is beneficial to the goals of the app and is easy to implement	**9/10**
Users can view statistics on their progress.	The feature is useful for viewing how much XP you've gained over the past few days. The free version does not go into much detail. The paid version shows additional information such as previous mistakes.	**5/10**
Users can follow other uses allowing them to see their progress and compare it against their own.	The feature is good to see how you're progressing compared to your friends. However, it is limited and doesn't go into much detail.	**6/10**
Users can take tests on their previous mistakes.	The idea is good however, it is not available in the free version.	**4/10**

2.7.3 Discussion

Duolingo is very similar to this app in both its style, aims, and design. However, there are important factors that impact the design of Duolingo which will not be present within this app.

Firstly, Duolingo is free to download. This means components of their app are made in a certain way to encourage users to purchase their paid version. This generally creates a worse user experience. In contrast, this app will be paid for directly on download. This means systems similar to the Duolingo hearts system will not be present and no content will be gated behind any additional paywalls after purchase. This is advantageous because it means the usability of this app is not compromised. The negative to this, however, is that the app will potentially reach a smaller audience than if it was free to download. This is because free apps are more widely downloaded than paid apps.

Secondly, Duolingo requires the user to be online if they are using the free version of the app. The questions are loaded from an online database meaning if you are not connected you cannot use the app. In contrast, this app will need to provide full functionality to the user both offline and online.

Finally, Duolingo is built for learning languages. This means that the design will be different because languages are learned in a different way to the topics focused on in this app. Duolingo's approach focuses much more on repetition which is needed to learn a language. In contrast, this project aims to focus more on learning how to solve the problems, as the focus is on problem-solving topics such as maths.

2.8 Discussion

The most important thing is that this app can run cross-platform. This means solely developing the app in XCode or Android Studio is not an option. Flutter and React Native both provide a good base for developing cross-platform apps, however, React Native will be more suitable. Firstly, the author has significantly more experience developing in React Native than Flutter so it will reduce the amount of time that has to be spent learning allowing for faster app development. Secondly, React Native is older than flutter meaning it has more community support in the form of documentation and libraries. This will be beneficial for some of the more complex parts of the project such as graphs and storage.

Additionally, having decided to develop in React Native, using Expo over vanilla React Native is the better option as it allows for easier testing of the app on iOS and Android, reducing development time. The disadvantages of Expo - app size, functionality limitations, and OS limitations are also not a major concern for this app as the file size and app complexity is going to be low compared to apps larger than apps developed in teams.

From researching similar apps, such as Duolingo and Quizlet, the conclusion is that Duolingo is more similar to the app being developed in this project. This is because it is a more focused education app than Quizlet. With purpose-built lessons and tests for specific topics. In contrast, Quizlet is more general - relying a lot on user-created content. However, Quizlet has features that could inspire potential expansion within this project. Some of the learning tools in Quizlet such as flashcards could be easily implemented which could improve the user experience.

2.9 PRISMA Chart

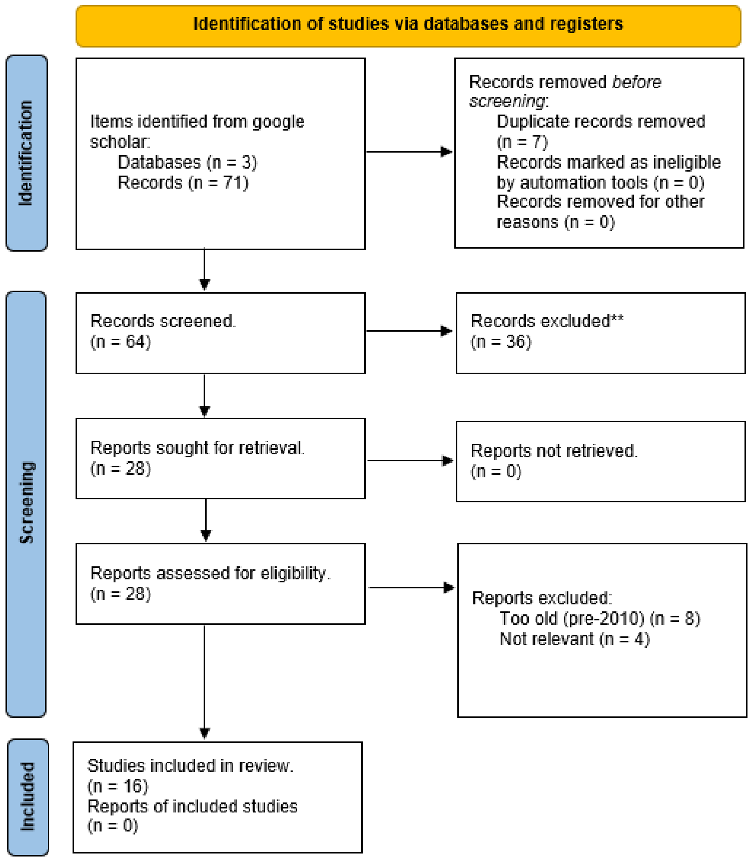

Identification of studies via databases and registers
Identification
Items identified from google scholar:
Databases (n = 3)
Records (n = 71)
Records removed before screening:
Duplicate records removed (n = 7)
Records marked as ineligible by automation tools (n = 0)
Records removed for other reasons (n = 0)
Screening
Records screened.
(n = 64)
Records excluded**
(n = 36)
Reports sought for retrieval.
(n = 28)
Reports not retrieved.
(n = 0)
Reports assessed for eligibility.
(n = 28)
Reports excluded:
Too old (pre-2010) (n = 8)
Not relevant (n = 4)
Included
Studies included in review.
(n = 16)
Reports of included studies
(n = 0)

Figure 5 - PRISMA chart

Figure 5 shows the PRISMA chart for this literature review. The purpose of a PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) chart is to depict the flow of information in a systematic review. The chart shows records that were identified as part of the review. It then shows how many records were included and excluded from the review and provides a reason for this. (Prisma. 2021)

Using google scholar as the main search engine, the main databases which the registers were identified from were: IEEE, Association for Computing Machinery and Springer Link. From here 71 potential papers were identified as being relevant. After further analysis, 28 reports were found to be suitable for this paper. Of which, 8 were excluded for being too old and 4 were excluded for not being as relevant as initially thought. This resulted in 16 papers which were included in this literature review.

3.0 Methodology

3.1 Introduction

This section will discuss what software methodology will be used for this project. The section outlines several methodologies which could be a fit for this project and then assess them against each other to determine which one is best. The methodology for this project needs to be flexible because the requirements are likely to change in small ways as the project progresses. Additionally, it needs to be relatively simple. There is no point in using a complex methodology because there is not a team of developers working on this project.

3.1.2 What is a software methodology?

Software development can be complicated. Software methodologies are used to break down the software development into smaller parts in order to make it easier to manage. Software methodologies generally have phases that describe how software development should take place. Standard phases that are common in many software methodologies include requirements gathering, design, implementation and testing. (Dora, S., & Dubey, P. 2013)

There are two main types of software methodology. The first is incremental software methodologies such as the waterfall methodology. Incremental methodologies go through the phases in a sequential order aiming to provide the client with a full product that they can use. In contrast, iterative software methodologies such as the spiral methodology allow for the client to provide feedback mid-development on unfinished work. This means that development can be changed mid-development based on the feedback provided which adds flexibility. (Alshamrani, A., & Bahattab, A. 2015)

3.2 Waterfall Model

3.2.1 Diagram of Waterfall Methodology

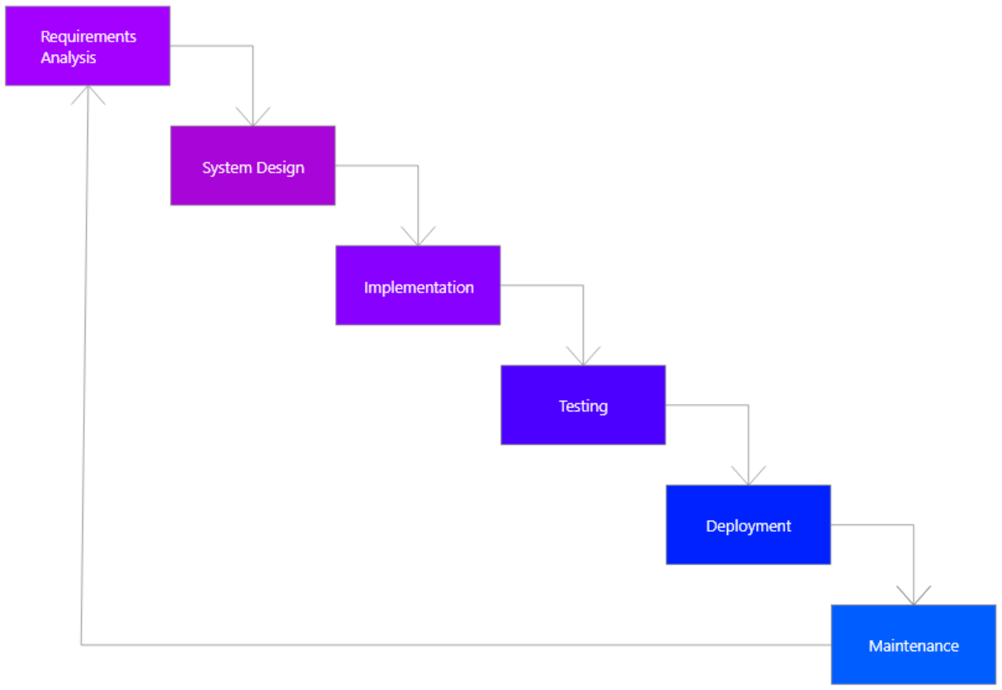


Figure 6 - Waterfall Model Diagram

3.2.2 What is the Waterfall Methodology?

Introduced in 1970, the Waterfall Model is a linear model for creating a piece of software. In the waterfall methodology, the development goes from one phase to another once the previous stage is complete as shown in figure 6. (Casteren, W, 2017)

Table 3.1 - Waterfall methodology assessment.

Advantages	Disadvantages
Very simple and easy to use.	Simplicity means it is poor for complex projects.
Works well for smaller projects with well-defined aims.	Rigidity means it is poor for projects where the requirements might change.
Easy to organise because the stages are well defined.	Poor for projects which need ongoing maintenance and support.

3.3 Spiral Model

3.3.1 Diagram of Spiral Methodology

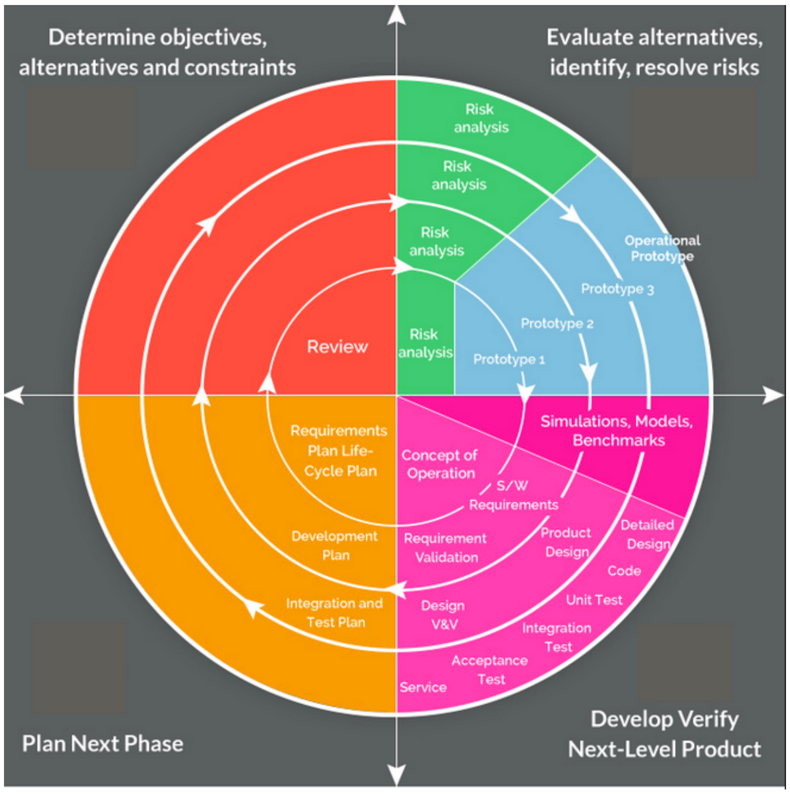


Figure 7 - Spiral Model. (Top 5 software development methodologies, 2020)

3.3.2 What is the Spiral Methodology?

The Spiral Model (figure 7) is a combination of the iterative model and the Waterfall Model. The Spiral Model is driven by risk. The development process is split up into phases (spirals) and each phase begins with a stated design objective. At the end of the phase, the client reviews the progress. This continues until the application being developed is completed. (Saeed et al, 2019)

Table 3.2 - Spiral methodology assessment.

Advantages	Disadvantages
Low risk because there is a high amount of risk analysis.	Risk analysis can be complicated.
Good for large projects.	Poor for smaller projects due to potential costs.
Strong documentation is produced.	Not suitable for low-risk projects.
The prototype is produced early and updated at regular phases during development.	May be hard to define milestones.
Good for gathering requirements as there is regular feedback with the client.	
Additional functionality can be added during a later phase meaning it is flexible.	

3.4 Agile Model

3.4.1 Diagram of Agile Methodology

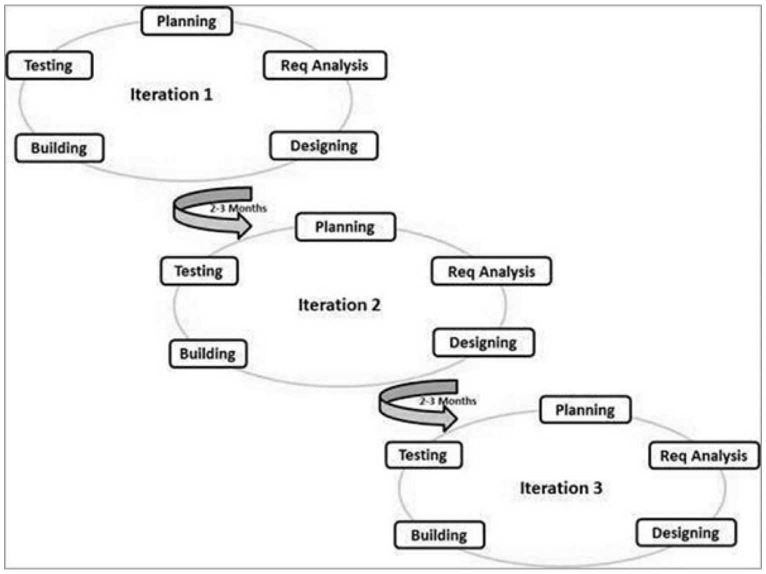


Figure 8 - Agile Model (Tutorials Points, 2021)

3.4.2 What is the Agile Methodology?

The Agile methodology (figure 8) was launched in 2001 when the Agile Manifesto was launched. The core values of the Agile methodology, as stated in the manifesto are:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

(Manifesto for Agile Software Development, 2001)

Consequently, the Agile methodology is a very flexible methodology due to its focus on communication with the client and the stakeholders and its focus on adapting and responding to change over following a set plan. Additionally, the methodology's focus on people encourages teamwork, self-organization, and accountability to allow for rapid software development. (Saeed et al, 2019)

Table 3.3 - Agile methodology assessment.

Advantages	Disadvantages
Continuous and rapid delivery of software.	Lack of focus on documentation.
People and interactions are a priority over processes and tools.	The project can lose focus from its original aim. Also known as scope creep.
Software is delivered quickly.	Can be complicated for an inexperienced developer.
Good communication with the client.	Harder to measure progress than some other methodologies.
Very flexible, can cope with a sudden or late change in requirements.	The product can lack cohesion if you are not careful.

3.5 Software Methodology Justification

Table 3.4 assesses each of the mentioned models in five important categories. Each model will be rated from one to ten on how well it fits the corresponding category with an overall score being given at the end. The ratings are just the authors assessment of each methodology and are subjective.

Table 3.4 - Methodology assessment.

	Waterfall	Spiral	Agile
Scalability	**3/10**	**8/10**	**8/10**
Flexibility	**2/10**	**7/10**	**9/10**
Delivery Speed	**3/10**	**8/10**	**9/10**
Testing	**6/10**	**6/10**	**5/10**
Difficulty	**9/10**	**4/10**	**2/10**
Overall Rating	**5/10**	**8/10**	**7/10**

Out of the models which have been discussed, the conclusion is that the Spiral Model is the most suitable for this project. The reason for this is that this project is highly client-driven. This means that the model used has to be flexible and also allow for regular communication with the client. The Spiral and Agile models allow for this, while the Waterfall model is too rigid for this project. However, Agile is not suitable for this project either because this is an individual project and the Agile Model is very team-oriented.

3.6 Evaluation of software methodology

As mentioned in section 3.0 the software development methodology that was chosen for this project was the spiral methodology. Overall, the methodology worked really well for this project, it was relatively simple to understand whilst providing the flexibility and client feedback that was so essential to creating a successful app. The evolutionary approach allowed for the implementation of all core functional requirements first and after feedback allowed for improvements to be made to them as well as the less important functional requirements in the subsequent spirals.

3.7 Project Management

3.7.1 Introduction

This section will discuss the management strategies that are being used in the project to maximise the chances of a successful outcome.

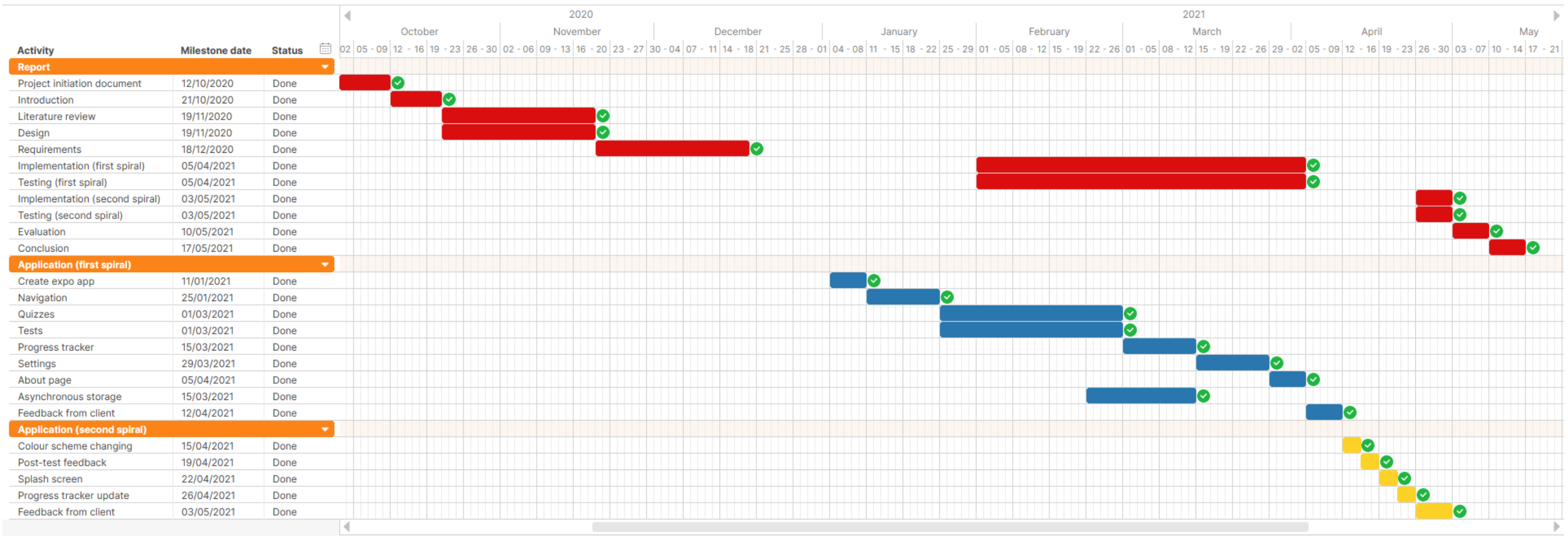
3.7.2 Project Initiation Document

The project initiation document (Appendix A) was created at the start of this project to begin the planning process for this project. The project initiation document covered topics such as the project aims and objectives, the risks, a preliminary time management plan and also an ethics assessment.

Overall, the project initiation document was really useful for starting this project. It provided a good way of summarizing the author's ideas for the project as well as providing a good early structure of what the project will look like.

Most of the contents of the project initiation document were stuck throughout this project. However, there were a few changes from the project initiation document later in the project. Firstly, the time management plan outlined in the project initiation document was changed to what is seen in section 3.7.3. The reason for this change was that it was found that the plan was too simple and lacked sufficient detail for this project. Secondly, the project initiation document stated that there would be no ethical considerations in this project. This has since been changed due to the inclusion of people in the requirements gathering and in the testing phases.

3.7.3 Gantt Chart



Activity	Milestone date	Status
Report		
Project initiation document	12/10/2020	Done
Introduction	21/10/2020	Done
Literature review	19/11/2020	Done
Design	19/11/2020	Done
Requirements	18/12/2020	Done
Implementation (first spiral)	05/04/2021	Done
Testing (first spiral)	05/04/2021	Done
Implementation (second spiral)	03/05/2021	Done
Testing (second spiral)	03/05/2021	Done
Evaluation	10/05/2021	Done
Conclusion	17/05/2021	Done
Application (first spiral)		
Create expo app	11/01/2021	Done
Navigation	25/01/2021	Done
Quizzes	01/03/2021	Done
Tests	01/03/2021	Done
Progress tracker	15/03/2021	Done
Settings	29/03/2021	Done
About page	05/04/2021	Done
Asynchronous storage	15/03/2021	Done
Feedback from client	12/04/2021	Done
Application (second spiral)		
Colour scheme changing	15/04/2021	Done
Post-test feedback	19/04/2021	Done
Splash screen	22/04/2021	Done
Progress tracker update	26/04/2021	Done
Feedback from client	03/05/2021	Done

Figure 9 - Project Gantt chart.

3.7.4 Time Management Discussion

Figure 9 shows the Gantt chart for this project. This chart illustrates when each section of the project will be completed, demonstrating how the time will be managed for the project. The information about the Gantt chart is shown at the left of figure 9. Each section contains a milestone and a status which denotes the time that particular section is expected to be completed and if it has been successful or not. Additionally, on the chart itself, there are green ticks that also show this.

The first section shows the time frame for the report. These sections are shown in red. The aim of the report implementation is to complete it somewhat sequentially. This is shown by how in most cases one section begins as another section ends. The reason for this approach is because in most cases each section feeds into the next. For example, it is easier to complete the literature review after the project initiation document and introduction as there is a greater understanding of the focus of the project. Furthermore, the sequential method is simple and easy to understand.

The second and third sections show the time frame for the first spiral and second spiral of the app respectively. These sections are shown in blue and yellow. While both sections are covering the same app implementation it is necessary to split them up because this project is following the spiral methodology. The first spiral is longer as it contains a much greater amount of implementation. In the end, there is a time slot for the client's feedback. Once their feedback is received the second spiral can begin.

The Gantt chart aims to provide an initial thought and structure on how the project will be managed. It will be adhered to as much as possible. However, unforeseen events such as technical failure or a sizable change in requirements might necessitate the adjustment of when sections of the project are completed.

3.7.5 Time Management Evaluation

The time management strategies implemented in this project were very successful. The Gantt chart shown in figure 9 shows that every milestone objective that was set was achieved within the required time frame (green ticks). Sticking to the pre-planned time frame allowed for every requirement that was required in the project to be completed. Furthermore, completing the project on time allowed for extra time to be spent resolving bugs at the end of the project development. This resulted in a more polished and smoother running app.

The process did have a few stumbling blocks however. For example, some sections of the report took longer to write than anticipated meaning that occasionally there would be some unintended overlap with other sections. However, this was not a major concern as these sections were finished shortly after.

3.8 Ethical Considerations

Throughout this project, there will need to be data which is collected from people such as in the requirements gathering or in the usability testing. It is very important that the data be collected and used ethically. There are a few ways that this project will ensure that data collection is done ethically:

- **Consent** - Data will only be collected from people who agree to take part through a consent form. If someone decides they no longer want to participate then their data will be removed from the project.
- **Transparency** - Participants will be told exactly why their data is needed, how it will be stored and how it will be used. Participants will receive a form that explains this to them.
- **Anonymity** - Participants will be kept fully anonymous at all times to protect their identity and so that no collected data can be traced back to them.
- **Relevance** - All of the data collected from participants will only be used for relevant areas. No unnecessary data will be collected.

These considerations are shown in Appendix C which is the consent form for the questionnaire. They are also shown in Appendix E which is the consent form for the usability testing.

4.0 Requirements

4.1 Introduction

This section will detail how the requirements for the project have been gathered from the relevant stakeholders. and the justifications for using the method. Additionally, the section contains the results of the requirements gathering and lists the functional and non-functional requirements for the project.

4.2 Stakeholders

A stakeholder is defined as "a person or company that is involved in a particular organization, project, system". (Simpson, J. A. (2008). *The Oxford English Dictionary: Vol. 1-)* So, anyone who has an interest in this project is a stakeholder. Therefore, it is essential to define the stakeholders of the project before any requirement gathering can take place otherwise some requirements may be omitted.

For this project, there are two types of stakeholder:

- A direct stakeholder is someone who has a direct role in the app.
- An indirect stakeholder is anyone else who may benefit or has an interest in the app.

(*Identification of Stakeholders*. 2008)

Stakeholders for this project have been identified using a stakeholder analysis process. The stakeholder analysis process aims to assess the system and how it can relate to interested parties. The process involved brainstorming potential parties who may have an interest in the project and assessing what type of relationship they will have with the app.

Table 4.1 shows the final results of the stakeholder analysis process. There are only two stakeholders as the scope of this project is relatively narrow. The client is a direct stakeholder as they have a direct role in the development and direction of the app. The client's customers are indirect stakeholders because they are going to be benefiting from the app; however, they do not have a direct role in the app.

Table 4.1 - Stakeholders.

Direct Stakeholders	Indirect Stakeholders
Client	Client's customers

4.3 Requirement Gathering Techniques

4.3.1 Introduction

Requirements gathering or elicitation is the process of establishing the system requirements for the end-users and for the stakeholders. The successful development of a system is dependent upon the requirements gathering being conducted properly. Missing requirements can lead to a system not meeting the needs of the users' or stakeholders. (Zachariah, B., & Nonyelum, O. 2020*)*

Requirement elicitation techniques are used to define the aims, scope and limitations of the system. Every elicitation technique has pros and cons which must be carefully considered before they are used in a project. Consequently, implementing the right requirement gathering techniques is essential to the success of a project. (Abbasi, M. A et al, 2015)

This section will detail the requirement elicitation techniques that will be used in this project. It will also assess their pros and cons and justify their use in this project.

4.3.2 Brainstorming

The initial way of gathering requirements involved drawing a mind map branching to the features that the client mentioned they would like to see in the app. Then from each feature, there are branches showing ideas for how that feature could be implemented and potentially expanded upon. The reason that a mindmap was chosen for the initial phase of requirements gathering is that it is a simple and easy way to visualize how the requirements for the system look and what work is required to create them.

4.3.3 Interviews

Interviews are the primary and most important method of requirements gathering for this project. The justification for using interviews is because the project is primarily driven by the client rather than other stakeholders such as users. An interview provides a great way to directly involve the client in the requirement elicitation process.

Due to Covid-19, it may be necessary to carry out interviews with the client online. The structure of the interview will be initially a discussion with the client about the app and how things are going with it. This followed by a series of structured questions about functionality and requirements. The results of these questions will be entered into a google form so they can be accessed at later dates.

4.2.4 Questionnaires

Questionnaires have also been carried out on the client's users to assess if there is anything in the app which could be changed, added, or removed. The questionnaire will be given to the client who will then distribute it to as many people as they are able. The client will then return the results for analysis.

The structure of the questionnaires will be a series of questions asking questions about functionality, usability and accessibility. The questionnaires will be carried out using Google forms so that they can be accessed at later dates.

The reason for choosing questionnaires as a method of requirements elicitation is that the method provides a way of cheaply and quickly getting results that have wide coverage.

Participants have been provided with a consent form and information sheet (Appendix C & D respectively) which ensures the questionnaires are in line with the ethical considerations outlined in section 3.8.

4.4 Requirement Gathering Results

4.4.1 Mind Map

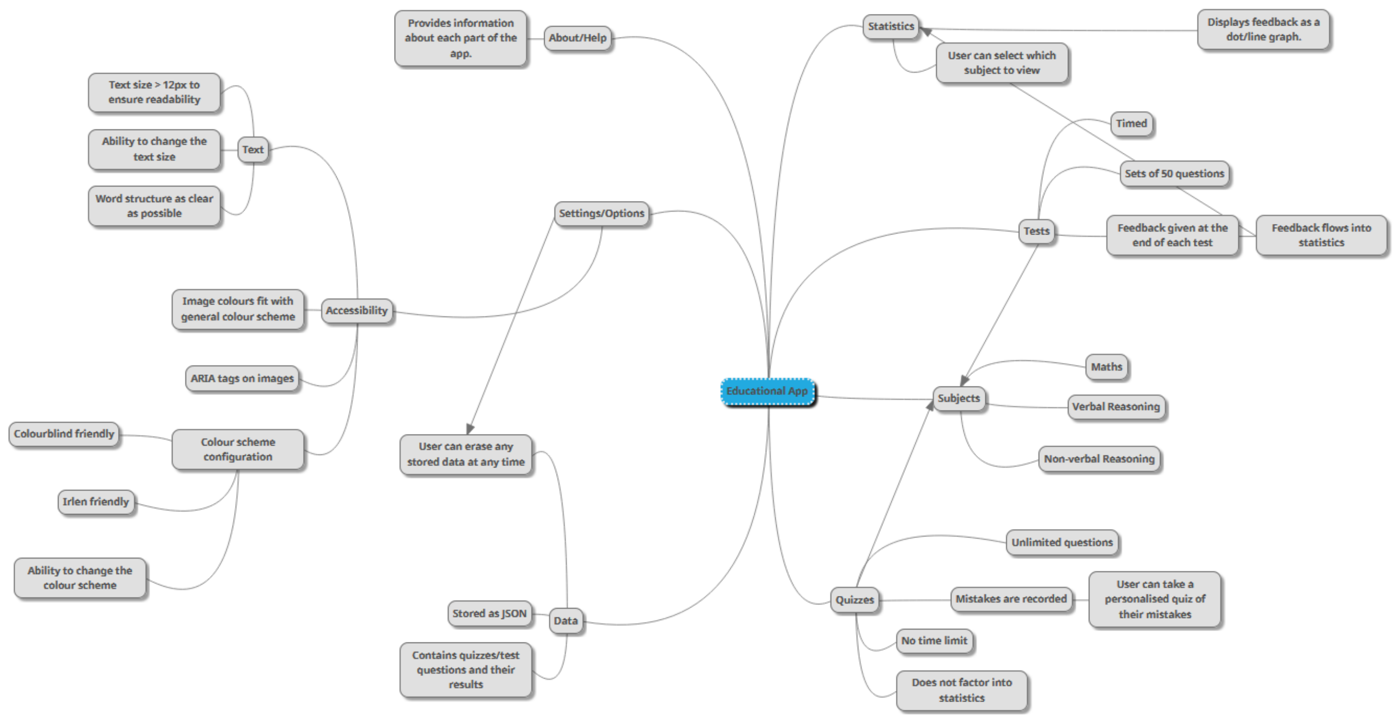

Educational App
About/Help
Provides information about each part of the app.
Settings/Options
Accessibility
Text
Text size > 12px to ensure readability
Ability to change the text size
Word structure as clear as possible
Image colours fit with general colour scheme
ARIA tags on images
Colour scheme configuration
Colourblind friendly
Irlen friendly
Ability to change the colour scheme
User can erase any stored data at any time
Data
Stored as JSON
Contains quizzes/test questions and their results
Statistics
Displays feedback as a dot/line graph.
User can select which subject to view
Tests
Timed
Sets of 50 questions
Feedback given at the end of each test
Feedback flows into statistics
Subjects
Maths
Verbal Reasoning
Non-verbal Reasoning
Quizzes
Unlimited questions
Mistakes are recorded
User can take a personalised quiz of their mistakes
No time limit
Does not factor into statistics

Figure 10 - Mind map. (Digital copy: https://bit.ly/2NoYGZW)

The results of the mind map (figure 10) show a preliminary brainstorming of requirements for the app after informal discussions with the client as well as my own ideas about what could be useful in the app. Starting from this as a base to inform the design of the interviews and questionnaire.

4.4.2 Interview Results

Question 1: What is the primary functionality that the app needs to have?

Answer: "The app should allow users to do questions based on the 11+ syllabus. The app should provide feedback when the user completes a question."

Evaluation: The result of this question shows that the quiz feature is the most important as it is the basic requested functionality. Everything else is technically just additional features. This means that the quiz requirement is the #1 priority for development.

Question 2: How important is it that the app should run while offline?

Answer: "It is very important that the app should work if there is no internet connection. Sometimes where I work there have been internet connection problems so it would be useful to be able to use the app regardless of connection level."

Evaluation: The result of this question shows the importance of using a method of data storage that is able to be accessed while offline. This means that online databases are not suitable for this project.

Question 3: How important is it that the user should be able to view the statistics?

Answer: "It is not the most important part of the app, the main requirement is the ability to just do questions. However, if a progress tracking feature could be implemented then it would be very nice."

Evaluation: The results of this question show that the test and statistics feature is desired but not required. Consequently, this feature is a medium priority.

Question 4: How important is accessibility and what are the most important accessibility considerations?

Answer: "I would say overall that making the app as accessible as possible is a priority once the main features of the app are done as many of the students that I work with have learning disabilities such as dyslexia."

Evaluation: The results of this question highlight the importance of accessibility considerations in the app as a medium priority requirement. The client specifically mentioned dyslexia which means that considerations for that are important.

Question 5: What operating system is desired?

Answer: “I think the best thing would be to have the app be able to work on Android and iOS because the people I work with use different devices.”

Evaluation: The results of this question show the importance of cross-platform support as a high priority requirement. This is already covered by the fact that the app is being developed on Expo which supports this.

Question 6: How important is UI scalability?

Answer: “It would be useful to have a scalable UI because sometimes we use iPads for learning so if the app could function there as well as on a phone that would be great.”

Evaluation: The results of this question show the need to have UI scaling as a high priority requirement. This can be achieved through reactive components.

4.4.3 Questionnaire Results

The questionnaire was sent to 67 people. From this we obtained results from 48 participants using google forms. This was more than expected and has been useful for formulating the requirements of the project. This section shows the data and draws conclusions for each question. The google form for the questionnaire is shown in Appendix D.

Question 1: Would a mobile app allowing you to take 11+ style questions be something that you are interested in?

Answer:

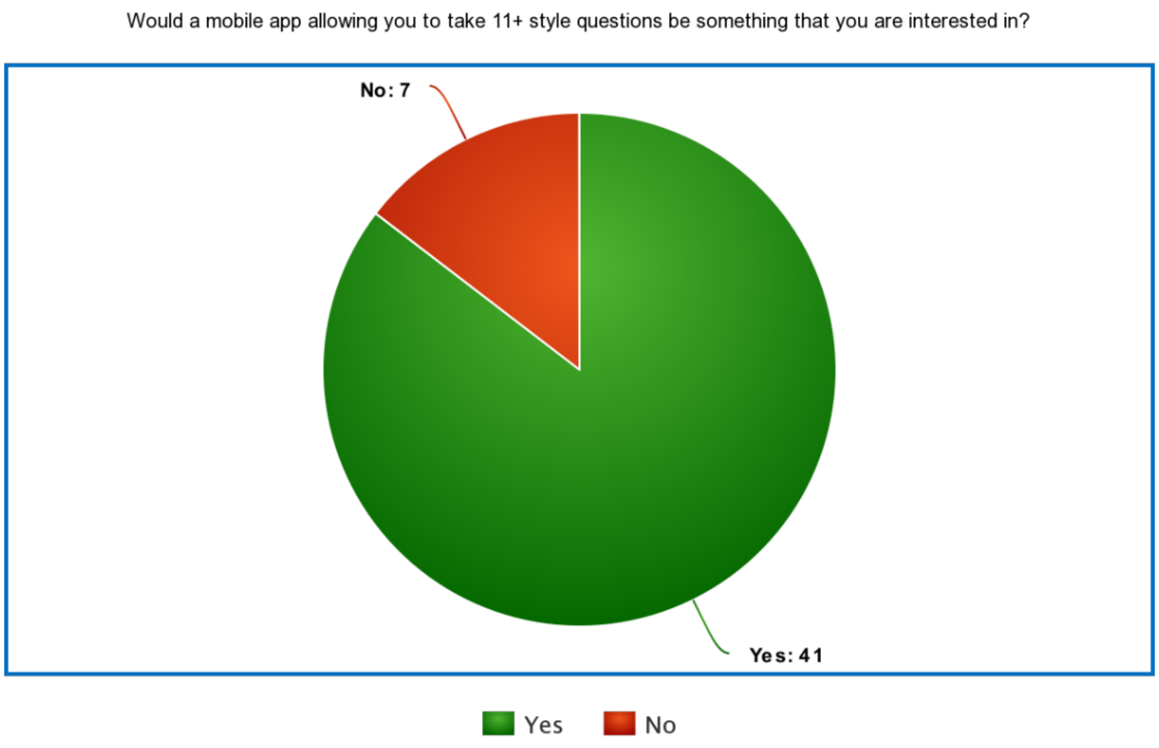


Figure 11 - Results of the first questionnaire question.

Evaluation:

- 41 respondents said that they would be interested in the app idea.
- Only 7 respondents said they would not be interested in the app idea.
- A positive result of 85%.

Conclusion: The results show interest in the app and show a solid indication that the app is useful and solves a real problem that people have. This emphasises the need for this project to take place.

Question 2: Do you have additional learning requirements such as requiring a specific colour scheme or larger text?

Answer:

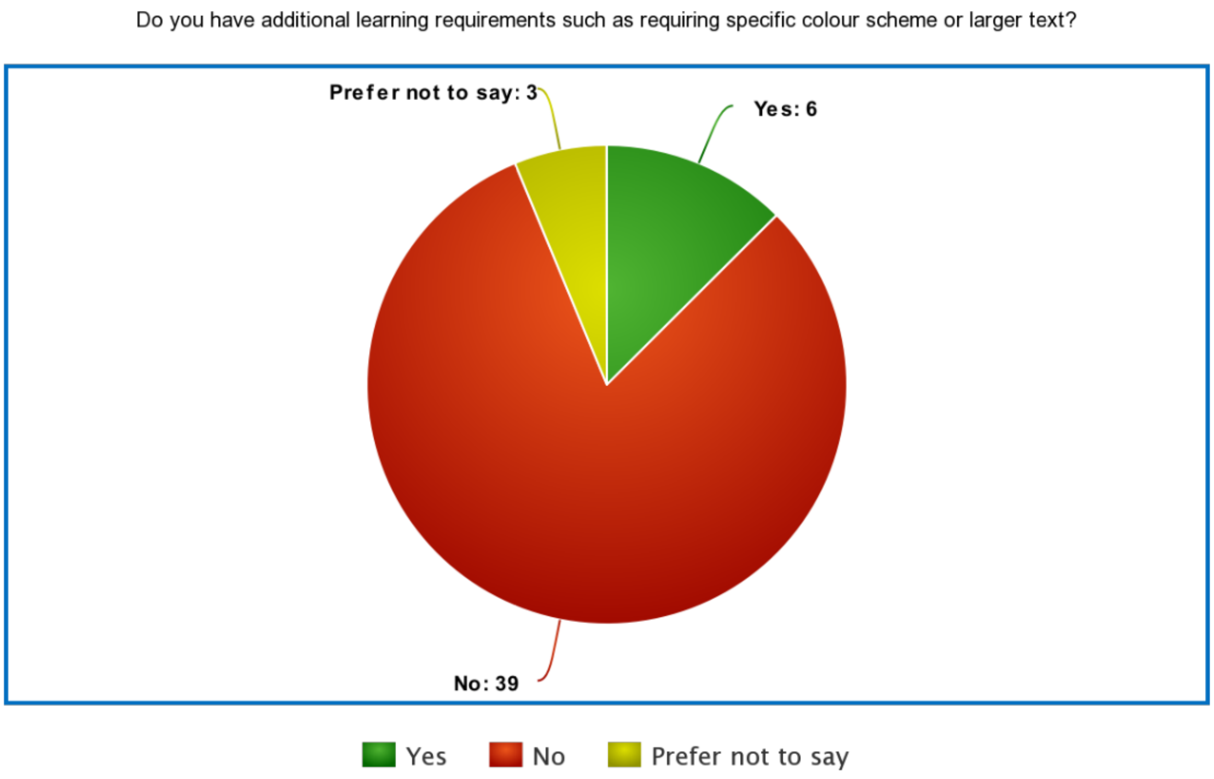


Figure 12 - Results of the second questionnaire question.

Evaluation:

- 39 respondents said they do not have any additional requirements.
- 6 respondents said they do have additional requirements.
- 3 respondents said they would prefer not to say.
- A positive result of 13%.

Conclusion: 6/48 people would like a degree of accessibility in the app. This is a reasonable proportion of the user base (higher than expected) and shows why it is important to include accessibility when designing the app. Consequently, adding accessibility will be listed as a high priority requirement.

Question 3: Would marked tests and a progress tracker in addition to basic questions be a feature you would be interested in?

Answer:

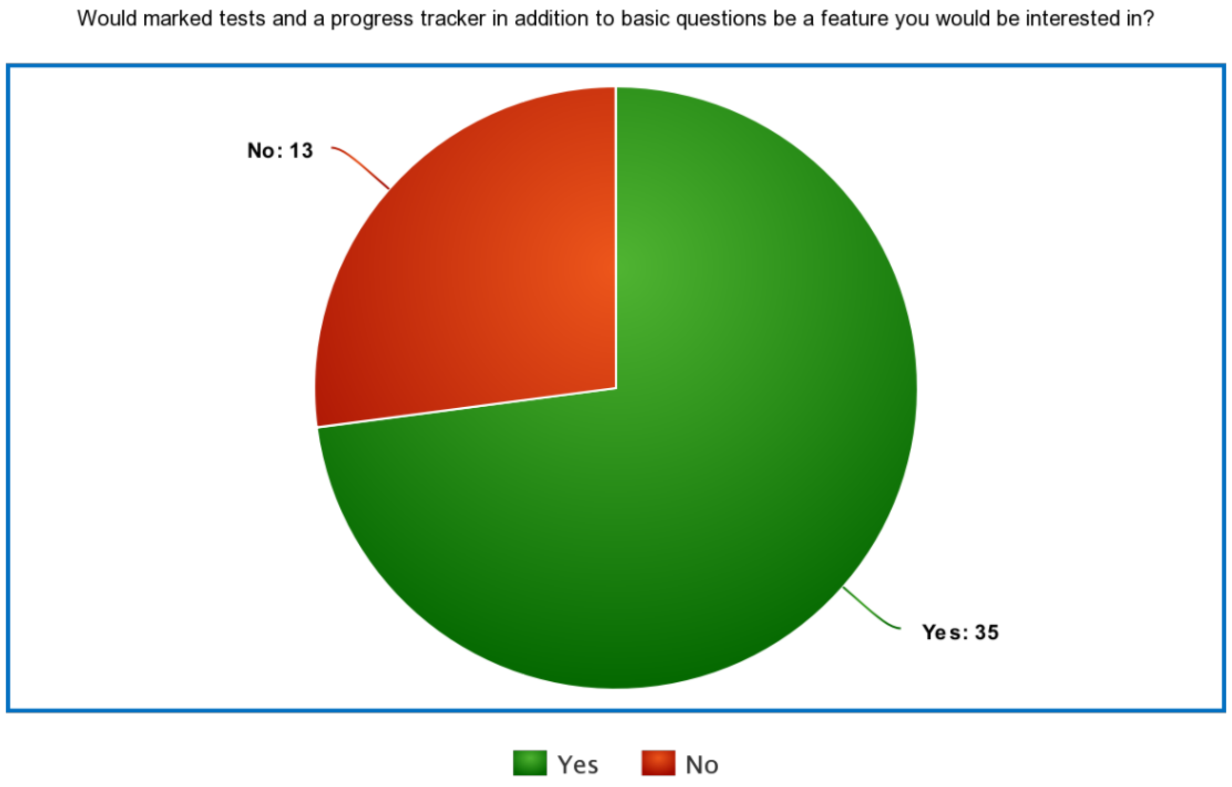


Figure 13 - Results of the third questionnaire question.

Evaluation:

- 35 of the respondents said they would be interested in a progress tracker.
- 13 of the respondents said they would not be interested in the progress tracker.
- A positive result of 73%.

Conclusion: There seems to be a high degree of interest in having tests and a progress tracker. However, the client has listed the progress tracker as a "would be nice" feature. This means that the test will be high priority and the progress tracker will be medium priority.

Question 4: Would you be interested in a personalised quiz that is made up of previous mistakes?

Answer:

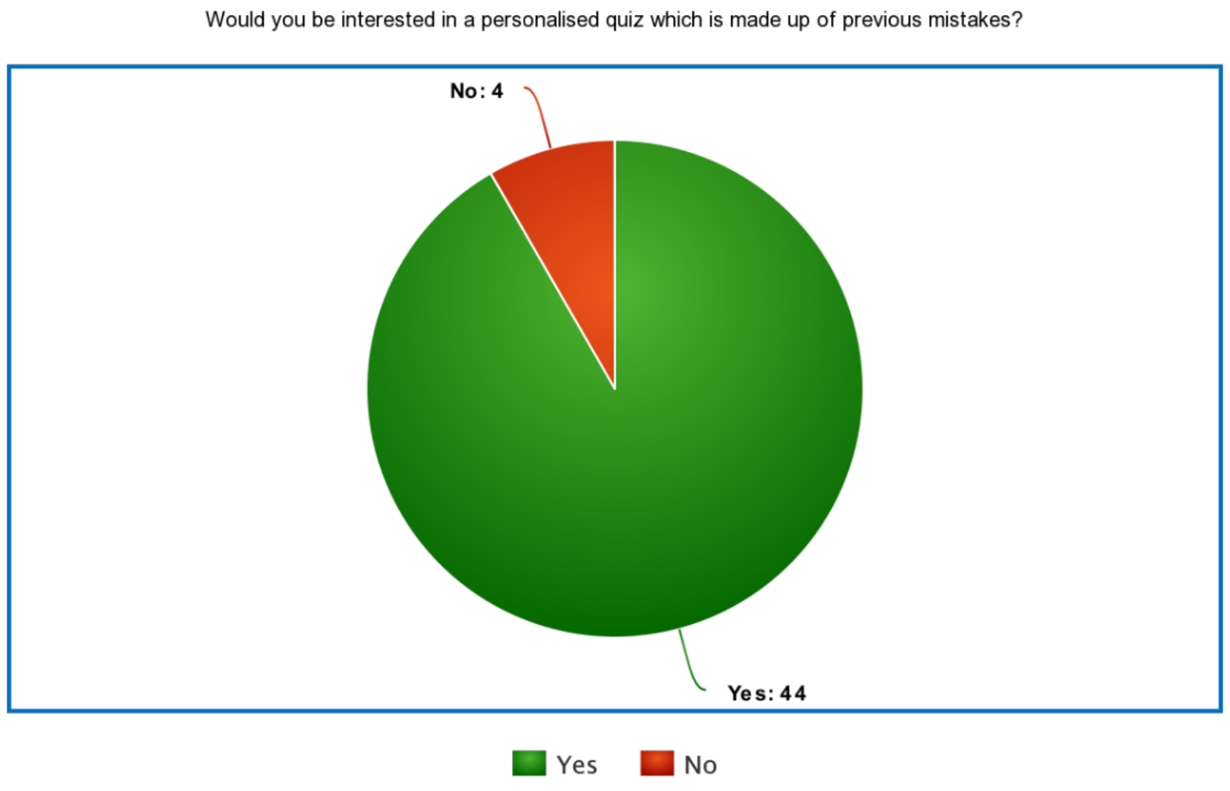


Figure 14 - Results of the fourth questionnaire question.

Evaluation:

- 44 of the respondents said they would be interested in a personalised quiz made up of previous mistakes.
- 4 of the respondents said they wouldn't be interested in a personalised quiz made up of previous mistakes.
- A positive result of 92%.

Conclusion: The result shows that there is a high level of interest in a personalised quiz created from previous mistakes. Additionally, the client said that this would be a good idea once core functionality is completed. Meaning, that the requirement is a medium priority.

Question 5: Would you primarily use the app on a phone or a tablet?

Answer:

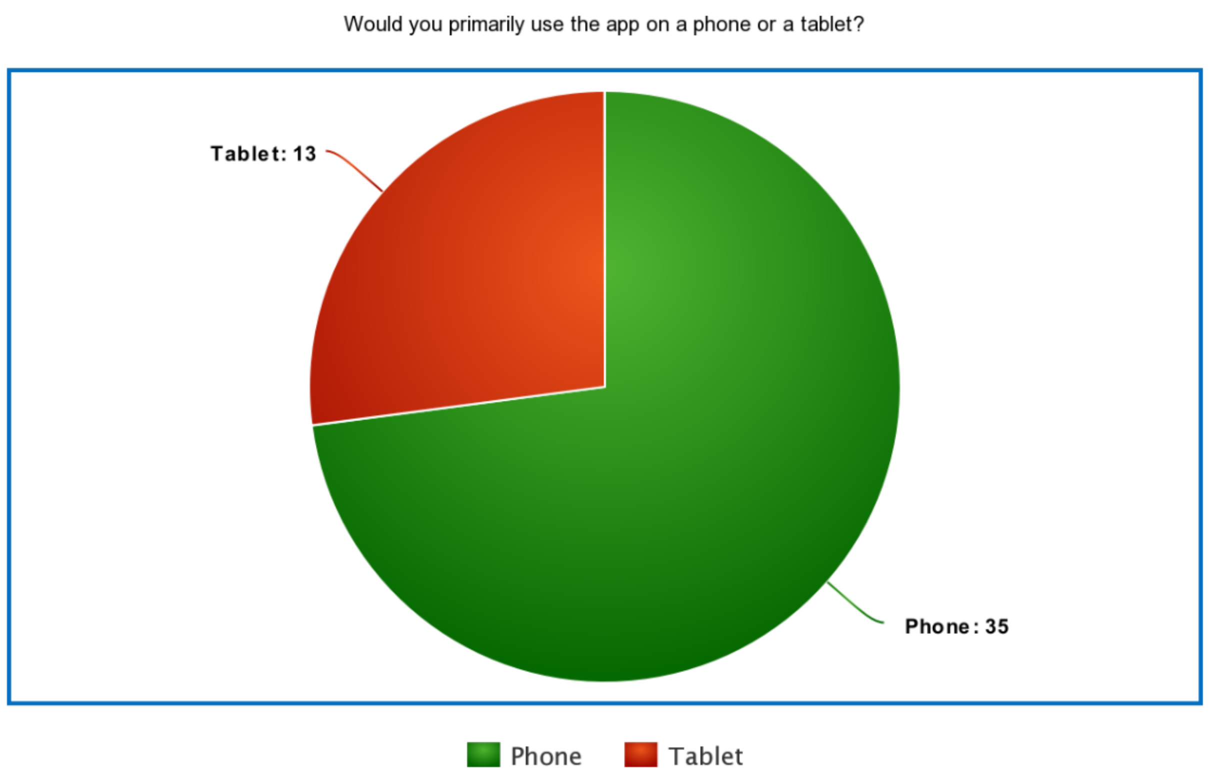


Figure 15 - Results of the fifth questionnaire question.

Evaluation:

- 35 of the respondents said they would primarily use a phone over a tablet when using the app.
- 13 of the respondents said they would primarily use a tablet over a phone when using the app.

Conclusion: While there is a skew in favour of phones, the results are much more mixed than expected. It would have expected that a larger majority of users would want to use the app on a mobile phone. What this shows is the importance of having a scalable UI that can work on different size screens.

Question 6: On a scale of 1 to 10, with 10 being the most important, how important would online features such as being able to compare your score with other users be to you?

Answer:

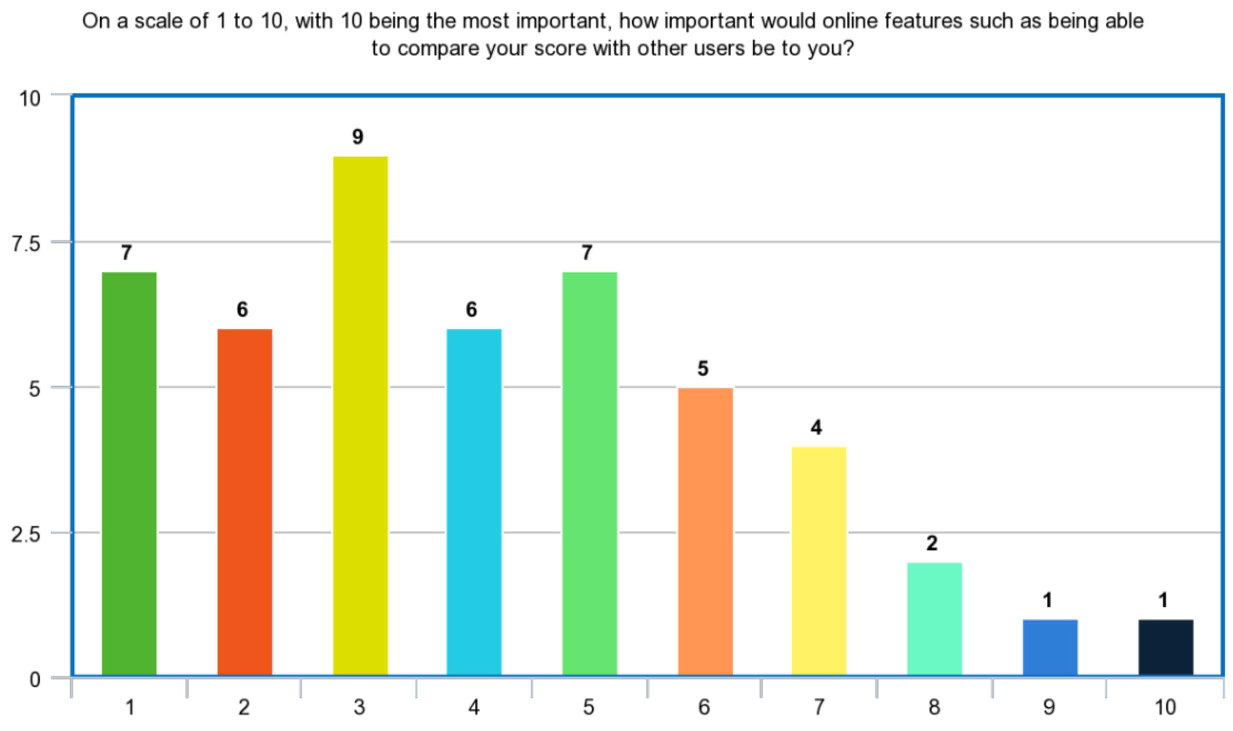


Figure 16 - Results of the sixth questionnaire question.

Evaluation:

- The majority of answers were on the lower end of the scale (less than 5).
- Not many people answered in the top 4 categories.

Conclusion: The results of the questionnaire show that online functionality is not really a priority for potential users. Very few answered above 5. This means that any online functionality in the app would come after the final product is developed and wouldn't be prioritised.

Question 7: On a scale of 1 to 10, with 10 being the most important, how important would additional learning features such as flashcards or minigames be to you?

Answer:

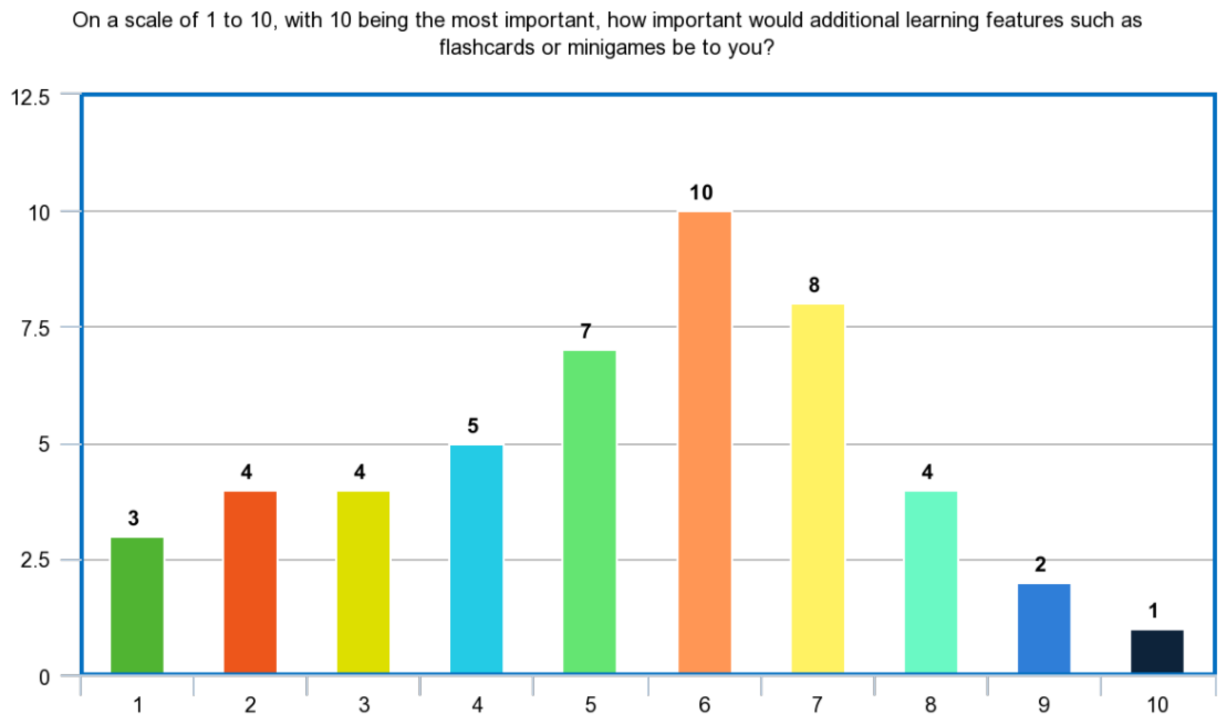


Figure 17 - Results of the seventh questionnaire question.

Evaluation:

- The main distribution is located around 6.
- More people are less interested than very interested.

Conclusion: The results suggest a reasonable amount of interest in other learning tools such as flashcards. However, the interest is not as strong as other questioned features and a fair amount of people also mentioned they would not consider such a feature very important. Consequently, the requirement will only be given low priority.

4.5 Requirements Specification

4.5.1 Introduction

For this project, there are two categories of requirements. Functional requirements, for this project, are defined as requirements that describe how the system must behave, what features and functions it should have. Non-functional requirements, for this project, are defined as being the general characteristics that the system should have. (Kurtanovic, Z., & Maalej, W. 2017)

After the requirements gathering methods are completed each requirement can then be given a unique identifier and categorized into functional and non-functional requirements. Additionally, each requirement has been given a priority relating to how important it is that requirement is implemented into the app. When the implementation phase is being carried out the requirements with the highest priority will be implemented first to minimize the risk that they are not included due to unforeseen issues or a lack of time.

4.5.2 Functional Requirements

Table 4.1 - Functional requirements.

Requirement ID	Requirement	Priority
F1	The application must allow users to take quizzes on chosen topics.	**High**
F2	The application must allow the users to take tests on chosen topics.	**High**
F3	The application must allow the user to change the colour scheme to make the application easier to use.	**High**
F4	The application must give the user the ability to clear the stored data.	**High**
F5	The application should contain an “about” section that can inform the user on how to use the application.	**High**
F6	Tests should provide instant	**High**

	feedback to the user when they are completed.	
F7	The application must allow users to view statistics about past performance.	**Medium**
F8	The user should be able to have more than one way of viewing statistics. For example, the default is a line graph. However, options for bar graphs or pie charts should be available.	**Medium**
F9	The application should contain an option that allows the user to select from at least 10 different colour schemes to increase accessibility.	**Medium**
F10	The app should allow the user to edit the size of the text to increase accessibility.	**Medium**
F11	Attempting to exit a test prematurely should trigger a popup alerting the user to prevent them from ruining their progress.	**Low**
F12	Test menus should provide the user with their best current score for each test.	**Low**
F13	The app should contain a splash screen displaying the client's company name.	**Low**

4.5.3 Non-functional Requirements

Table 4.2 - Non-functional requirements.

Requirement ID	Requirement	Priority
NF1	The app needs to be scalable. If new questions need to be added they should be able to cope.	**High**
NF2	The app needs to be completed by the 21st of May 2021.	**High**
NF3	The app needs to be able to run while offline.	**High**
NF4	The app needs to be able to run on iOS, Android and on an expo web test server for demonstration.	**High**
NF5	The app should have accessibility considerations for visual disabilities.	**High**
NF6	The app should be created in a way that maximises maintainability.	**Medium**
NF6	The app needs to be able to work effectively on larger devices such as tablets.	**Medium**

4.6 Requirements Gathering Evaluation

In the view of the author, the requirements gathering process was successful. Each elicitation method used proved greatly beneficial to the overall formation of the requirements. The decision to involve users through questionnaires was also great as it provided new insights from a wider group of people which greatly helped inform the process. In the author's view, the only way this process could have been improved is by including more people and asking a wider breadth of questions. However, this is not essential as the results obtained are good.

5.0 Design

5.1 Introduction

This section of the paper will talk about the design of the application. The section details how each component in the app will be designed and the reason behind it. Additionally, this section details how the app will be made accessible for as many users as possible.

5.2 Accessibility Design

5.2.1 Introduction

Accessibility is an important and often overlooked, part of the design process for any application. The reason for including accessibility in the design process is to ensure that the app is as inclusive as possible and that everyone can get the most out of the app regardless of personal difficulties. (Kavcic, A, 2005) There are several types of accessibility categories that will need to be considered:

1. **Visual issues:** Where the user has issues seeing particular content which impairs their ability to fully use the application. For example, colour blindness.
2. **Auditory issues:** When the user has difficulty hearing so they may have problems using parts of the application which use sound.
3. **Learning issues:** Such as dyslexia or Irlen Syndrome which can impair a users ability to use the application.
4. **Mobility issues:** Such as not being able to move their fingers to press on the screen as easily.
5. **Seizures**: Such as photosensitive epilepsy triggered by stimuli such as flashing patterns.

(Hill, H, 2013)

5.2.2 Visual Accessibility

The colour scheme is an important part of making the app accessible to people with visual disabilities. The colour scheme for the app will contain contrasting colours - these are colours that are opposites on a colour wheel (figure 18). The advantage of contrasting colours is that they stand out against each other making it much easier to view content. Furthermore, these colour combinations are considered "harmonious" because people tend to like these combinations, this would benefit app's usability. (Westland, S et al, 2007)

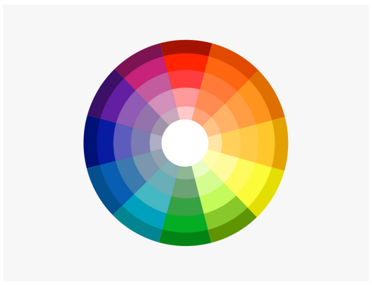


Figure 18 - Colour wheel (How to use a colour wheel, 2020)

Additionally, the colours of the colour scheme must be viewable for people with colour-blindness. So, colour combinations such as green/red, green/brown, blue/purple, green/blue, green/grey, blue/grey, etc.. (Robyn Collinge, 2017) cannot be used as a significant amount of people will have issues viewing content with these colour schemes.

This app, by default, will have a purple gradient colour scheme with white text. The white text stands out very well on a purple background and the colour combination is not listed as being a problem for any group of people to see. Additionally, the theme that it provides is also in line with the client's request that the default setting of the app should have a light and optimistic tone to it. The app will also include several alternative colour schemes. Allowing users to select their preference. This will aid usability as well as accessibility. There will be at least 10 different colour schemes to increase the chance of the user finding one which suits them. All of the colour schemes in the app will be in line with the previously mentioned requirements.

Image accessibility is also a factor that will have to be considered when designing the visual accessibility of the app. Aria stands for Accessible Rich Internet Applications and is a set of attributes that allow for content to be made more accessible. Some people have difficulty viewing images so all images in the app will contain aria tags**.** These tags describe the image to people if they require it and also if the image is not displayed on the screen for whatever reason. (MDN Web Docs, 2020)

5.2.3 Auditory Accessibility

Based on the current requirements there will not be any important sounds in the application so Auditory concerns are not a major issue.

The quizzes and test part of the app could have sound. If the user gets a question correct a “correct sound” could play. If a user gets an answer wrong then a “wrong sound” could play. However, this is not currently planned and if a user could not hear these sounds then it is not a major issue because there are visual changes to also represent this information. For example, if a user gets an answer correct then the button will go green, if they get an answer wrong then the button will go red.

5.2.4 Learning Accessibility

Learning disabilities can come in many different forms and many solutions to them can be found in other sections of this accessibility chapter. However, some of the more common ones will be expanded upon in this section. The reason for mentioning these specific learning conditions is that they are generally the most common ones that are likely to have the largest impact on users to use the app.

Dyslexia

Dyslexia is a learning disability that impacts a person's ability to read and spell. It is estimated that about 10% of the population globally suffer from Dyslexia to some degree (Dyslexia and Literacy International, 2014*),* therefore, considering the disability when designing this application is important. There are several design choices to make the app more accessible to someone who has Dyslexia.

The text should be of a font size of around 12-14 otherwise it can be harder to read. Additionally, the layout of the text should be even - left to right focused text keeps the left-hand side of the text consistent meaning the right is jagged. Additionally, whitespace in the text should be minimal to increase readability. (British Dyslexia Association, 2020). The app will allow the user to adjust the text size to their desired size to further accommodate this.

Irlen Syndrome

Irlen Syndrome is a condition in which the individual has difficulty processing visual information. Irlen Syndrome impacts a wide spectrum of activities, but, most relevant to this project is the ability to read written text. (Miyasaka, J. et al, 2019) The author has Irlen Syndrome. This means the author knows firsthand how helpful it is to have good accessibility tools for it. The best way to assist a user who suffers from Irlen Syndrome (or similar) is to have a wide spectrum of colour schemes for the app. Colour schemes make a big difference however, every person will have a different preference. For example, the colour scheme which works best for the author is yellow/black however, another person may find that something like purple/black works better. To accommodate for this the app will consist of at least 10 different Irlen appropriate colour schemes for the user to choose from.

The colour schemes on the official Irlen website (www.irlen.com) are: white/black, light blue/black, turquoise/black, light green/black, yellow/black, gold/black, purple/black, beige/black, dull pink/black and grey/black.

5.2.5 Mobility Accessibility

Some users may have disabilities that impair their movement. Consequently, it is important to design the user interface of the app to require as little physical movement as possible.

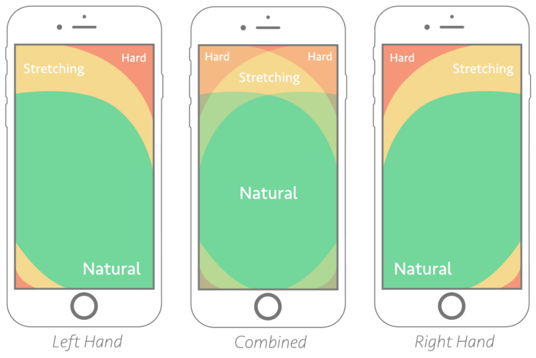


Figure 19 - Phone screen reachability (Samantha Ingram, 2016)

Figure 19 shows the easier locations on a standard phone screen for a user to interact with. Ideally, the vast majority of the interactivity will be in the green section. For this app, all functionality except the ‘back’ button and the ‘settings’ option are within this zone. This means the app should be as friendly as possible for people with mobility disabilities to use.

5.2.6 Seizures Accessibility

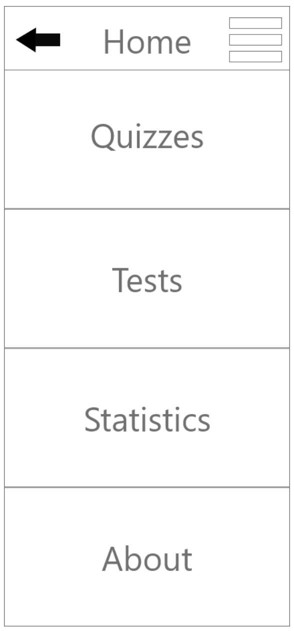
Photosensitive Epileptic seizures are triggered by flashing lights and high contrast colours and patterns such as bars or diamonds. It has been found that between 2-14% of the population has some form of Photosensitive response to these stimuli. (Poleon, S., & Szaflarski, J. P, 2017) Consequently, it is reasonable to assume that 2-14% of users will have these issues, emphasising its importance. In order to mitigate the risk to people with Photosensitive Epilepsy, there will be no flashing lights or sudden colour changes from bright to dark.

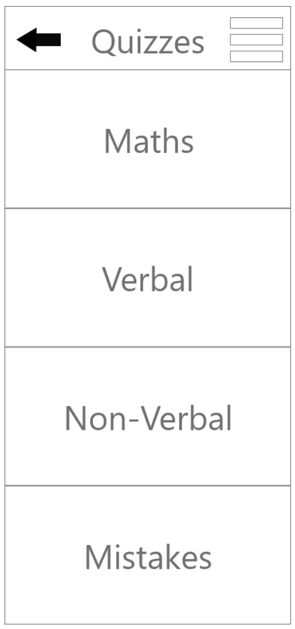
5.3 User Interface Mock-ups

This section will contain mock-up drawings of the user interface design and detail the choices behind each component. These mock-ups are not necessarily 100% representative of the final product, however, they will show the general design direction of the UI and the principles behind the design which will not change throughout the project.

Creating design mocks-ups is important and necessary to the design process because it allows for the easy visualisation of the app before the implementation process begins. This means that potential issues with the app can be more easily identified pre-implementation. Changes can then be made to fix these problems much more easily, as it is easier to change a mock-up than to change an already created app. Furthermore, it also means that it is possible to receive feedback on mock-up designs with stakeholders which can help to inform the design process.

5.3.1 Navigation Mock-ups





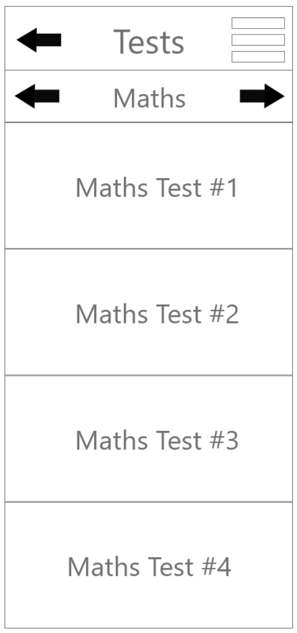


Figure 20 - Homepage navigation design.
Figure 21 - Quiz navigation design.
Figure 22 - Test navigation design.

Figures 20, 21 and 22 show UI designs for the different navigation screens that are in the app. All navigation screens share the same design characteristics. They use a full-screen navigation display, this is where all the navigation takes up the entire screen. The reason that this navigation design was chosen is that it provides a simple and clear display to the user whilst not wasting any screen space. Furthermore, the full-screen design fits well with the linear nature of navigation in the app.

Another design was considered but ultimately found to not be as suitable as a side navigation bar. This is where the navigation appears from the side of the app when prompted. The advantage of this is that it uses less screen space. However, it was determined that this approach would not work as well as the design works better in apps where you can have more flexibility in navigating. This app works best when you navigate through each level in a linear fashion.

Additionally, at the top of every page in the app, there is a header. This header has a title that is displayed to the user which page they are currently on, a button (left) that takes the user to the previous page they were on and a button (right) for displaying the settings. It was determined that using a header design would be the best way to provide this functionality and information to the user as it is clear, simple and fits with the existing navigation design.

5.3.2 Quizzes & Tests

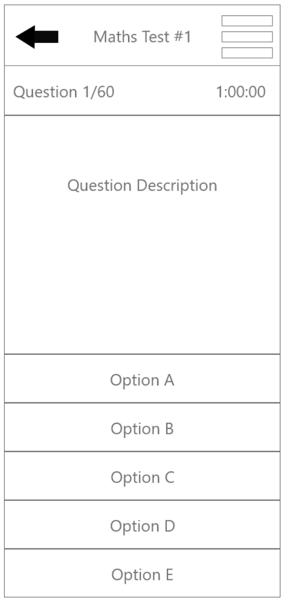


Figure 23 - Quiz question UI mock-up.
Figure 24 - Test question UI mock-up.

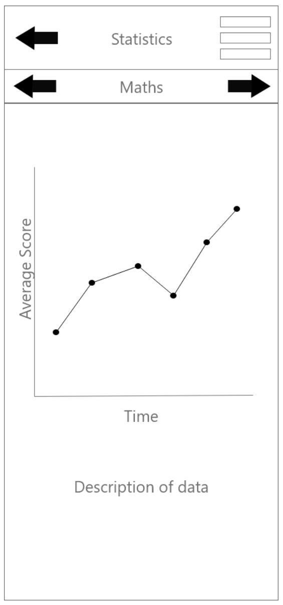
Figures 23 and 24 show the mock-up UI designs for the quiz and test section of the app respectively. The design decisions for both are similar. Firstly, the design of the potential questions answers buttons at the bottom follows a stacked design. This design is where each button is stacked on top of the other. The reason for choosing this design is because vertically, it does not take up much room allowing space for the question text. Whilst horizontally it provides plenty of room in case some answers are particularly long.

It was considered initially, to have the question-answer buttons be separated and placed in a square design - so two in the top row, two in the bottom row. However, this design idea was found to not be suitable because during discussions with the client it was requested that each question in the app should have five potential answers. This means that this design idea would not fit as it would only work with an even amount of questions. Furthermore, having gaps between the buttons is a waste of screen space which is not ideal.

Secondly, there is a large space where the textual question description will be placed. The design choice for having a large space comes from the fact that question descriptions can vary significantly in length meaning that enough space is needed to accommodate all questions. Additionally, providing buffer space between the question description and the other components should make the text easier to read and the UI less cluttered.

Finally, both figures 23 and 24 contain a subheader under the main header. The function in each is slightly different. In figure 23. In figure 24. However, the design goal is the same, to provide quiz or test information in a clear and concise way without obstructing functionality. The subheader is the only feasible method of doing this that fits with the project design. It works well due to its small vertical space but large horizontal space.

5.3.3 Statistics Page



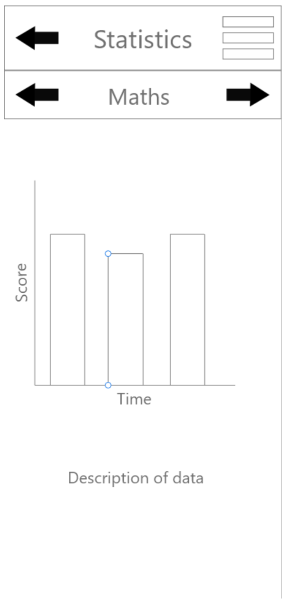


Figure 25 - Progress tracker (line chart) UI mock-up.
Figure 26 - Progress tracker (bar chart) UI mock-up.

Figures 25 and 26 show the mock-up UI design for the statistics page using the line graph format and the bar graph format respectively. This page needs to show the data for the user's past performance in tests. This means that the design needs to focus on clarity as the user needs to be able to understand what the information means. To achieve this design goal-line/bar charts were chosen as they are very common and thus users will likely have prior knowledge of how they work. Furthermore, they are good ways to display data and for displaying trends that fit into the design objective to show the user's improvement in tests. Secondly, a description under the chart will explain the meaning of the data to the user, further aiding clarity.

It was initially considered to simply have a table with the data into the user. While this would work it is not a very good way of visualising the data and it would take up more screen space. Therefore, it was deemed not suitable and the aforementioned design approach was chosen instead.

5.3.4 Settings Page

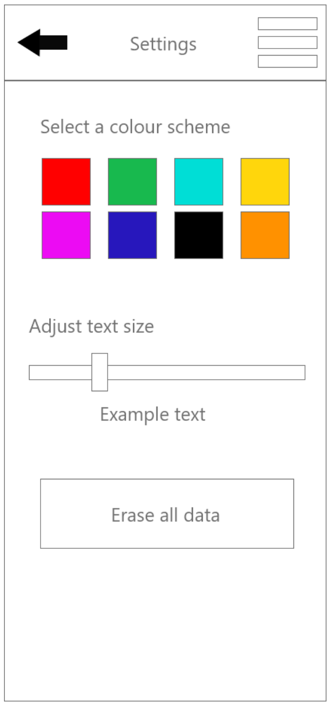


Figure 27 - Settings page UI mock-up.

Figure 27 shows the mock-up UI design for the settings page. This page is the page where the user can set their preferences and erase any stored data. The design makes the functionality of this page as easy to use as possible. Firstly, the buttons to change the colour scheme are all clearly visible and show the corresponding colour scheme colour. This approach is more visual than using text to explain, and thus should be easier to use and more accessible. This is a benefit as accessibility is the core reason for implementing this feature.

This method was also applied to the feature which allows the user to adjust text size. There is a slider that allows the user to adjust the text size with example text below. The slider was chosen because it is an easy and visual way of allowing the user to adjust a numeric value when compared to other methods such as a text box input. Furthermore, there is less room for error as the user cannot enter an invalid input with a slider bar.

Finally, the button to erase data has been designed so that it is large and easy to see. This is why the button takes up so much space. The reason for this is that it is important for it to be as clear and easy as possible for the user to see how they can manage their data. When pressed a pop-up will be displayed so that the user doesn't accidentally delete their data - providing safety. Additionally, the button will also be coloured in a distinctly visible colour such as red or pink to make it even more visible to the user.

5.3.5 About Page

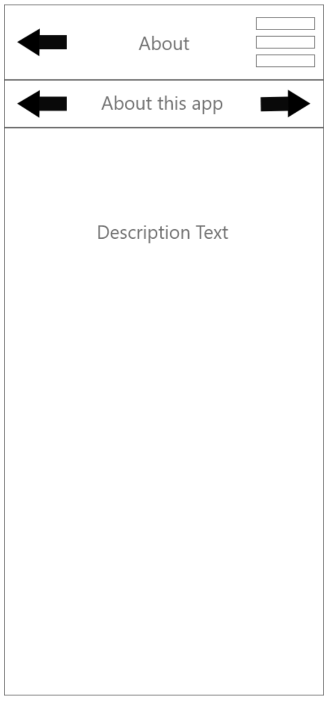


Figure 28 - About page UI mock-up.

Figure 28 shows the UI mockup for the about / help page. The design of this page is really simple. There is a subheader that allows the user to easily navigate between pages. Then there is a large space for text. The only design objective of this page is to display textual information to the user. So the design works well as the space for the text allows for flexibility as there is a reasonable amount of variation between different text lengths.

5.4 Back-end Design

This section will talk about the back-end design of the system. Including how the data will be stored and managed and how the backend of the system interacts with the frontend of the system.

For this app to function there will need to be some sort of data storage in order to be able to load up questions, settings and statistics. Research has been conducted into many different methods and systems of data storage for this project and this section will outline the advantages and disadvantages of the most promising ones before coming to a final conclusion on which method will be used in this project.

5.4.1 Proposed back-end approaches

MongoDB

A consideration was to use MongoDB for storing the data used in the system. MongoDB is a JSON based database that runs on the cloud and is a system that the author has a decent amount of experience using. The advantages of using MongoDB is that the free version provides more than enough space, performance and flexibility for this project. The fact that MongoDB used JSON format is also an advantage because this project is being built using Javascript.

However, the main problem with MongoDB is that the version that would be used requires an internet connection to access the server where the data is being stored. So, if a user is offline they will not be able to use the app if the data is stored on MongoDB. (MongoDB documentation, 2021*)*

MySQL Community Edition

Another consideration was to use MySQL Community Edition which is a free version of the open-source database software MySQL. The database software that uses SQL (Structured Query Language) and provides robust security and flexible scaling. Additionally, it is possible to run MySQL offline which is necessary for this project. However, it does require additional work to implement. (MySQL, 2021)

<u>JSON with Asynchronous Storage</u>

A method of implementing persistent data storage for this project which would be easier to implement than the other methods would be to use JSON files and Asynchronous Storage in combination. A simple JSON file can be used to store data that does not need to be changed, for example the questions for quizzes and tests.

For data that needs to be changed such as user preferences Asynchronous Storage can be used. Asynchronous Storage is a React Native library which allows us to have a persistent store of data which can be asynchronously retrieved and edited. This solution is optimal for the settings and custom mistakes quiz that will be in this project. Additionally, the library has a lot of support which makes learning how to use it easy.

While using this method wouldn't have the functionality and security that using proper database software would provide as neither use any sort of encryption. It would fully meet the needs of the projects and would not needlessly overcomplicate anything. Additionally, it would be lightweight and would allow for the data to be accessed while offline. (*Usage - Async Storage*)

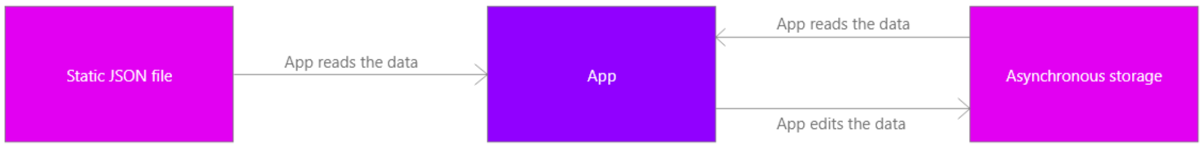


Figure 29 - Simple diagram showing the flow of data using Asynchronous storage and JSON.

5.4.2 Back-end Conclusion

This project will be implementing data storage using JSON files for unchanging data and asynchronous storage for changing data. The reason for this is, as previously mentioned, it enables all the functionality required without over complicating it. All this app needs from the data storage are the ability to store and retrieve sets of information and update user preferences. There is no need for anything more advanced such as selective querying. Additionally, none of the data is sensitive so security is not a major concern and many databases either require a constant internet connection or add a lot to the space the application takes up.

5.6 UML use case diagram

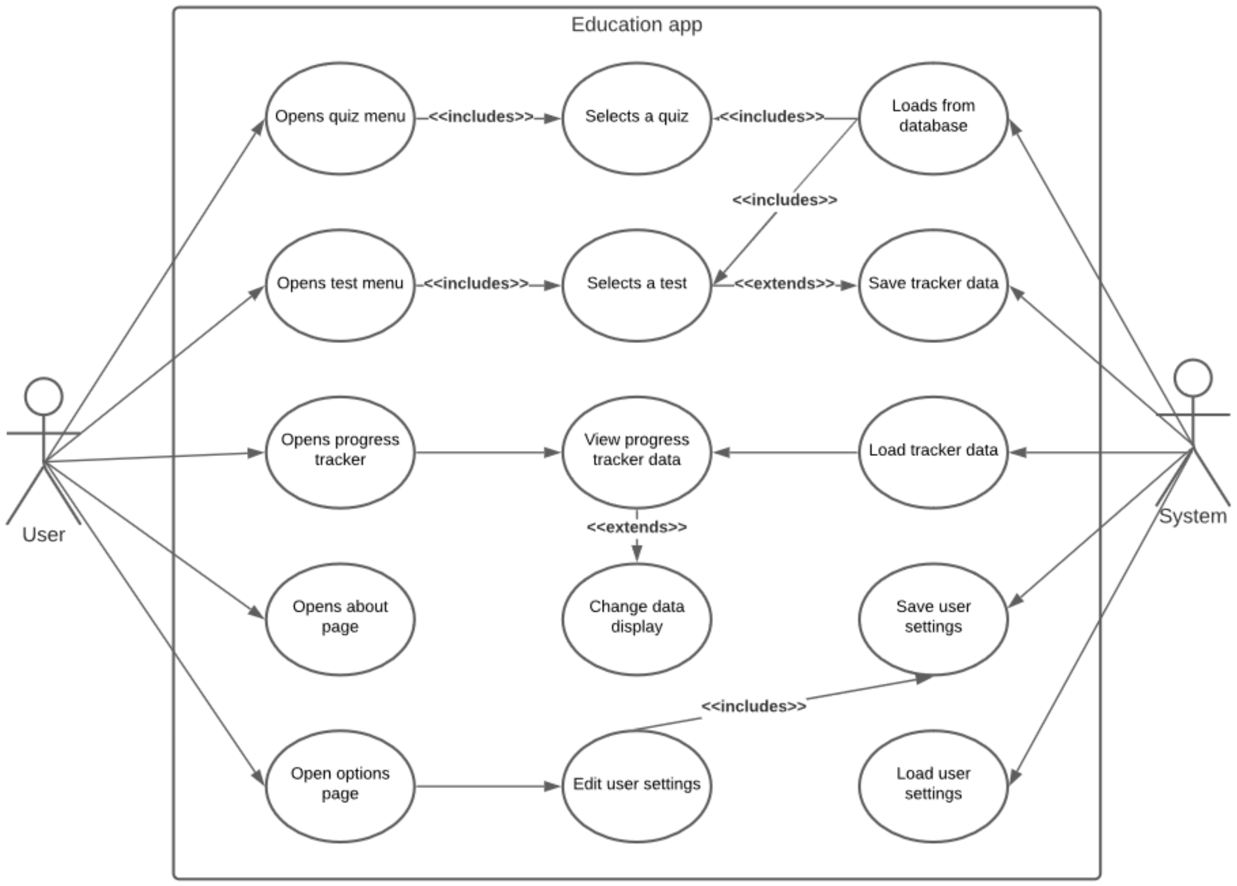


Figure 30 - UML use case diagram.

Figure 30 shows the UML (unified modelling language) use case diagram for this project. The purpose of a UML use case diagram is to create an abstract view of the system that highlights all of the potential ways that a user can interact with the system. (Ibrahim, N et al. 2011)

Components of a UML use case diagram:

- Actors - Denoted by stick figures. Represents an entity such as a user that interacts with the system.
- System - Actions and interactions between actors within the constraints of the project.
- Use cases - Denoted by an oval. Represents different uses in a system that an actor may interact with.
- Associations - Links between actors and use cases.
- System boundary - The box surrounding the diagram, all use cases in the box are considered within the scope of the system.

(*UML use case diagram tutorial*. Lucidchart)

Figure 30 contains two actors, the user and the system. The arrows from the actors to the use cases represent the associations between them. Associations that contain <<extends>> show that the second use case adds further functionality to the first one. Associations that contain <<includes>> show use cases that require the other use case in order to function properly.

5.7 Design Evaluation

In the view of the author, the design phase of the project was a success and was really important for the subsequent implementation phase. The UI mock-ups had their intended impact of making developing the UI in the app easier as the UI components did not need to be implemented from a clean slate. Upon reflection, however, it may have been better to involve the client more in the UI design process as some minor UI tweaks were requested later on during implementation. These change requests could potentially be avoided if the client was more involved earlier on.

6.0 Implementation

6.1 Introduction

This section of the report will demonstrate how the implementation phase of the project has been conducted. Each stage of development from setting up the development environment to the implementation of each requirement will be explained.

6.1.2 Development Environment

The first part of the implementation involved setting up the development environment. Firstly, the tools used require NPM (node packet manager) which was installed on the computer that the app was to be created on. Once NPM was downloaded the Expo package had to be downloaded. This was done by running the NPM command “npm install --global expo-cli”.

Once Expo is installed it is easy to start a project. The command “expo init final-year-project” was run which created a new directory with all necessary files for my project. To then run the command “npm run start” can be run from the project directory, which begins a local server that can be used to test the app.

A platform for the program was also needed. Visual Studio Code was chosen. VSCode is a free and open-source code editor developed by Microsoft and is currently the most popular free code editor. (*Documentation for visual studio code, 2016)* The reason VSCode was chosen over the other editors such as Atom is due to my previous experience using VSCode. Furthermore, the author considers the file navigation, plugins and the overall quality of VSCode to be the best compared to any free editor.

6.1.3 Version Control / Backup

This project will be using GitHub as a remote way of backing up my work to reduce the risk of work being lost in the event of a technical failure as outlined in my initial risk assessment. GitHub is a free version control and code hosting platform allowing me to directly push and commit my changes from my local pc to a remote repository. (*GitHub documentation*, 2021) The reason that Github will be used for this project is that the author has extensive prior experience using it and it has a good range of additional features such as identifying vulnerabilities in a project that will be helpful later.

6.1.4 Tutorials/Guides/Documentation Followed

The author has previous experience with Javascript and using React. However, the author did not have any experience of creating mobile apps with React Native so it was necessary to follow some tutorials to get the author started on development. Most of the learning material came from the official React Native documentation (https://reactnative.dev/) as well as the official Expo documentation (https://docs.expo.io/) as they contained the most up to date and complete information.

The author watched some youtube videos to help get started on creating basic components as watching someone explain and demonstrate something can sometimes be more helpful than reading documentation. Some of the videos used are:

- https://www.youtube.com/watch?v=FL9ITVE-MU0&ab_channel=ErHarinderSingh
- https://www.youtube.com/watch?v=NgDaPmxewcg&ab_channel=EstebanCodes
- https://www.youtube.com/watch?v=Hf4MJH0jDb4&ab_channel=TraversyMedia
- https://www.youtube.com/watch?v=0kL6nhutjQ8&ab_channel=MadeWithMatt

However, some of the information in the videos proved to be not fully up-to-date or could be improved so the author was able to take the knowledge from these videos and improve upon them.

6.2 First Spiral/Prototype

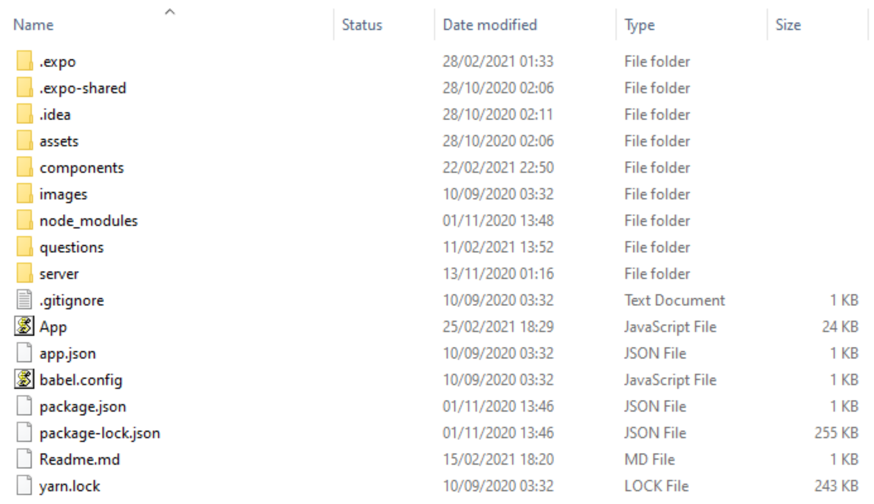
6.2.1 Introduction

The methodology section 3.0 determined that the implementation would be conducted using the Spiral methodology. This section details the results of the first spiral/prototype which were created during the implementation phase of the project. The aim of this spiral is to get all of the core/high priority functionality that were outlined in section 4.0 (requirements) completed. The order in which the item is mentioned pertains to the order in which it was completed. See table 6.1.

Table 6.1 - Order of completion for the project implementation.

Order of completion	Task
1	Setting up Expo.
2	Creating the necessary React components.
3	Creating the home page.
4	Creating the selector for quizzes.
5	Creating the selector for tests.
6	Creating the ability to do quiz questions.
7	Creating the ability to do test questions.
8	Creating the ability to view statistics.
9	Creating the ability to view an “about” page.
10	Creating a settings page.

6.2.2 Expo setup



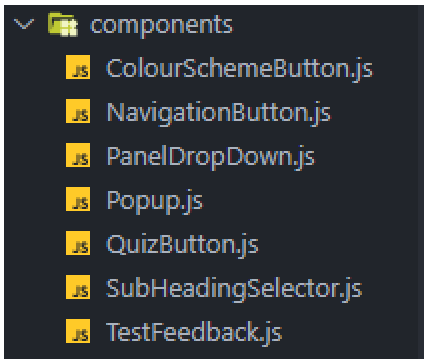
Name	Status	Date modified	Type	Size
.expo		28/02/2021 01:33	File folder	
.expo-shared		28/10/2020 02:06	File folder	
.idea		28/10/2020 02:11	File folder	
assets		28/10/2020 02:06	File folder	
components		22/02/2021 22:50	File folder	
images		10/09/2020 03:32	File folder	
node_modules		01/11/2020 13:48	File folder	
questions		11/02/2021 13:52	File folder	
server		13/11/2020 01:16	File folder	
.gitignore		10/09/2020 03:32	Text Document	1 KB
App		25/02/2021 18:29	JavaScript File	24 KB
app.json		10/09/2020 03:32	JSON File	1 KB
babel.config		10/09/2020 03:32	JavaScript File	1 KB
package.json		01/11/2020 13:46	JSON File	1 KB
package-lock.json		01/11/2020 13:46	JSON File	255 KB
Readme.md		15/02/2021 18:20	MD File	1 KB
yarn.lock		10/09/2020 03:32	LOCK File	243 KB

Figure 31 - The resulting expo project after initialization.

The first step for the implementation of this spiral was to install an expo and use it to initiate an expo project. Figure 31 shows the results of initiating the expo project, once this was done it was possible to run the project on a local expo server. This allowed for the testing of the app on an iPhone, Android emulator and web browser.

6.2.3 React Native Components

Before the implementation of the requirements could begin, it was necessary to create the React Native components. These are reusable UI components that are the building block of the app. As components are a core part of any React/React Native application this was very easy to implement and had a large amount of existing support. So choosing this method was really the obvious choice with no real alternatives.



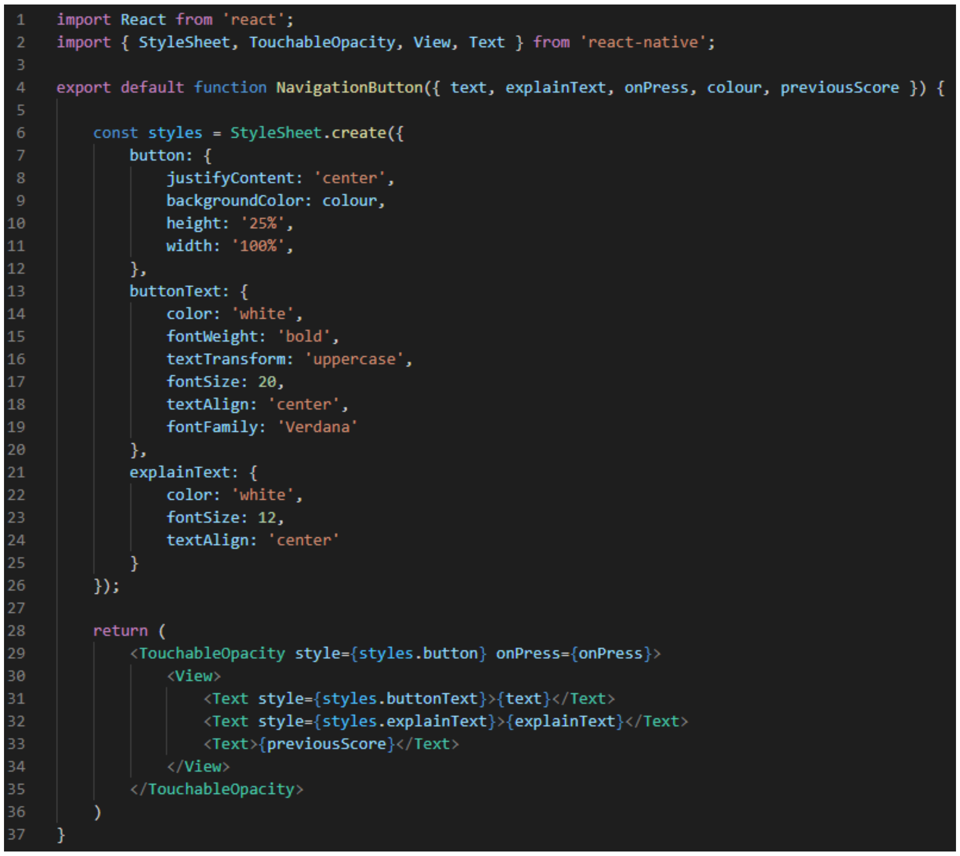
```
components
    ColourSchemeButton.js
    NavigationButton.js
    PanelDropDown.js
    Popup.js
    QuizButton.js
    SubHeadingSelector.js
    TestFeedback.js
```

Figure 32 - The components folder of the app.

The components are all Javascript files that are stored in their own folder (figure 32). This makes finding and maintaining the components easier. To use the components in the main app file it is possible to import them from the components folder. Then the JSX of the imported component is called with necessary parameters.

The subsequent subsections for this section of the report will detail how each component works and any pitfalls that were encountered. However, the justification for creating each component is the same. Each component is a reusable UI component that can simply be called with the right parameters, rather than having to repeat the JSX every time the component is needed. This increases significantly, the efficiency of the code and the maintainability of the project. For example, if a change is needed in the navigation buttons component then it is just a matter of updating the component file. Whereas if the component was not present then all 12 instances would have to be updated individually.

6.2.3.1 Navigation Button



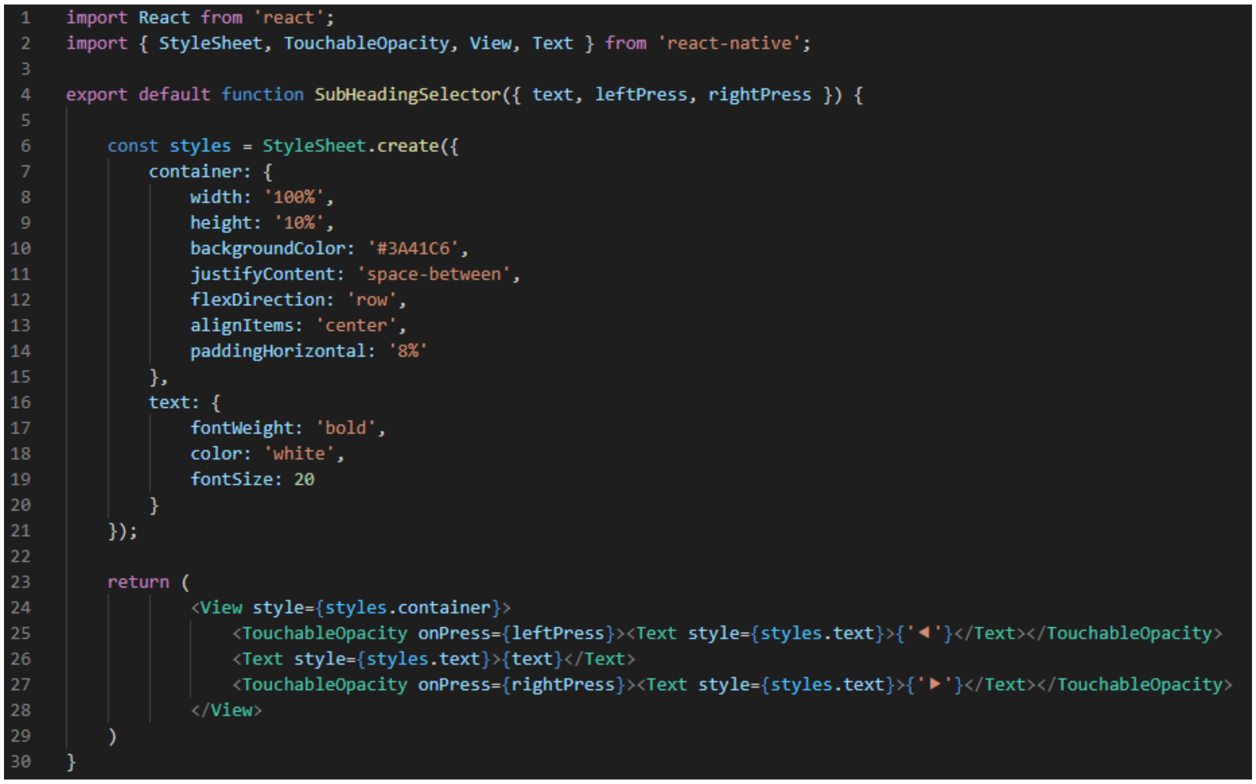
```
import React from 'react';
import { StyleSheet, TouchableOpacity, View, Text } from 'react-native';

export default function NavigationButton({ text, explainText, onPress, colour, previousScore }) {

    const styles = StyleSheet.create({
        button: {
            justifyContent: 'center',
            backgroundColor: colour,
            height: '25%',
            width: '100%',
        },
        buttonText: {
            color: 'white',
            fontWeight: 'bold',
            textTransform: 'uppercase',
            fontSize: 20,
            textAlign: 'center',
            fontFamily: 'Verdana'
        },
        explainText: {
            color: 'white',
            fontSize: 12,
            textAlign: 'center'
        }
    });

    return (
        <TouchableOpacity style={styles.button} onPress={onPress}>
            <View>
                <Text style={styles.buttonText}>{text}</Text>
                <Text style={styles.explainText}>{explainText}</Text>
                <Text>{previousScore}</Text>
            </View>
        </TouchableOpacity>
    )
}
```

Figure 33 - The code for the "NavigationButton" component.

Figure 33 shows the component "NavigationButton". The primary function is to facilitate navigation to other components of the app and is used on the home page, the quiz selector and the test selector.

The component takes five parameters. The first parameter is the main text which tells the user what the button does. The second parameter "explainText" is the small descriptor text below the main text to provide further information The third parameter is a function that tells the app what to do when the button is pressed. The fourth parameter determines the colour of the button. This is important to give the gradient colouring which is used in the app. The final parameter is used solely for tests and shows the user their previous score for a specific test.

6.2.3.2 Subheading Selector



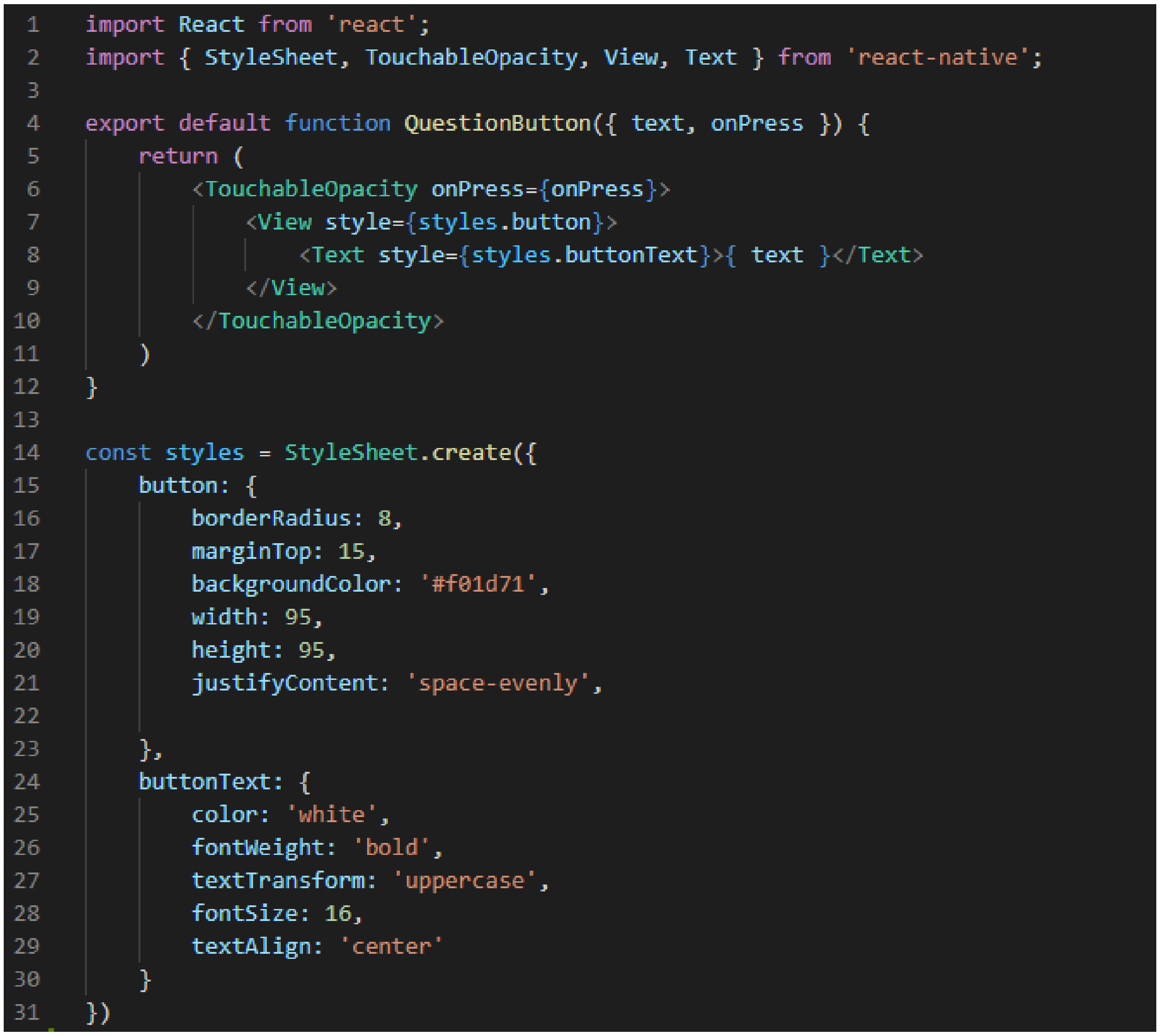
```
import React from 'react';
import { StyleSheet, TouchableOpacity, View, Text } from 'react-native';

export default function SubHeadingSelector({ text, leftPress, rightPress }) {

    const styles = StyleSheet.create({
        container: {
            width: '100%',
            height: '10%',
            backgroundColor: '#3A41C6',
            justifyContent: 'space-between',
            flexDirection: 'row',
            alignItems: 'center',
            paddingHorizontal: '8%'
        },
        text: {
            fontWeight: 'bold',
            color: 'white',
            fontSize: 20
        }
    });

    return (
            <View style={styles.container}>
                <TouchableOpacity onPress={leftPress}><Text style={styles.text}>{'◄'}</Text></TouchableOpacity>
                <Text style={styles.text}>{text}</Text>
                <TouchableOpacity onPress={rightPress}><Text style={styles.text}>{'►'}</Text></TouchableOpacity>
            </View>
    )
}
```

Figure 34 - The code for the "SubHeadingSelector" component.

Figure 34 shows the component "SubHeadingSelector". This is the component that is used on the about page, the test selector page and the progress tracker page. It allows for sub-navigation within a given feature of the app without having to use the main header. For example, the test selection page allows the user to select what subject they would like to do a test in without having to leave the test menu and reselect a subject.

The component takes 3 parameters, the first is the text that is to be displayed. For example, this could be the subject currently selected on the test menu. The second parameter is a function that tells the app what to do when the user clicks on the left arrow. The third parameter is a function that tells the app what to do when the user clicks on the right arrow.

6.2.3.3 Question Button

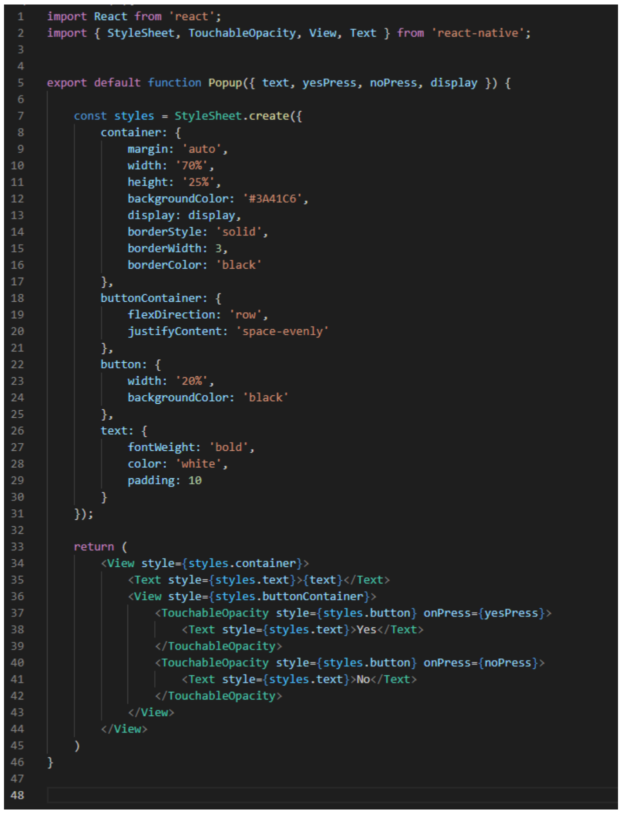


```
import React from 'react';
import { StyleSheet, TouchableOpacity, View, Text } from 'react-native';

export default function QuestionButton({ text, onPress }) {
    return (
        <TouchableOpacity onPress={onPress}>
            <View style={styles.button}>
                <Text style={styles.buttonText}>{ text }</Text>
            </View>
        </TouchableOpacity>
    )
}

const styles = StyleSheet.create({
    button: {
        borderRadius: 8,
        marginTop: 15,
        backgroundColor: '#f01d71',
        width: 95,
        height: 95,
        justifyContent: 'space-evenly',

    },
    buttonText: {
        color: 'white',
        fontWeight: 'bold',
        textTransform: 'uppercase',
        fontSize: 16,
        textAlign: 'center'
    }
})
```

Figure 35 - The code for the "QuestionButton" component.

Figure 35 shows the component "QuestionButton". This component is used on the quiz and test pages as the buttons that the user can press to select which answer they would like to choose for a given question. The component is simple and only takes two parameters. The first parameter is simply the text that is in the button. For example, a question-answer. The second parameter is just a function that tells the app what to do when the button is pressed.

6.2.3.4 Popup



```
import React from 'react';
import { StyleSheet, TouchableOpacity, View, Text } from 'react-native';


export default function Popup({ text, yesPress, noPress, display }) {

    const styles = StyleSheet.create({
        container: {
            margin: 'auto',
            width: '70%',
            height: '25%',
            backgroundColor: '#3A41C6',
            display: display,
            borderStyle: 'solid',
            borderWidth: 3,
            borderColor: 'black'
        },
        buttonContainer: {
            flexDirection: 'row',
            justifyContent: 'space-evenly'
        },
        button: {
            width: '20%',
            backgroundColor: 'black'
        },
        text: {
            fontWeight: 'bold',
            color: 'white',
            padding: 10
        }
    });

    return (
        <View style={styles.container}>
            <Text style={styles.text}>{text}</Text>
            <View style={styles.buttonContainer}>
                <TouchableOpacity style={styles.button} onPress={yesPress}>
                    <Text style={styles.text}>Yes</Text>
                </TouchableOpacity>
                <TouchableOpacity style={styles.button} onPress={noPress}>
                    <Text style={styles.text}>No</Text>
                </TouchableOpacity>
            </View>
        </View>
    )
}

```

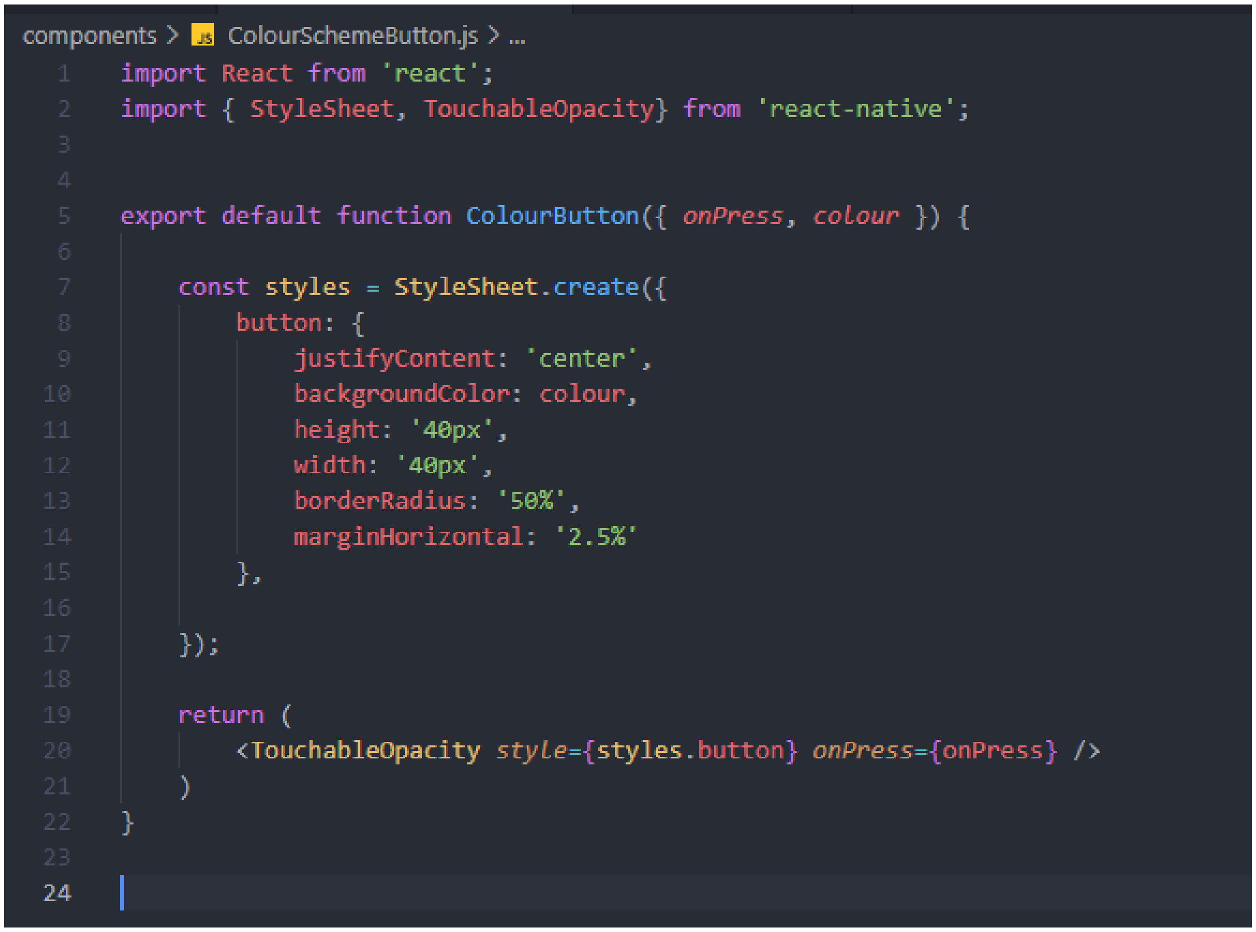
Figure 36 - The code for the "Popup" component.

Figure 36 shows the component "Popup". This component is used frequently throughout the app to display popup messages to the user. For example, on the test questions page, the user can exit out of a test at any time. If the user attempts to do this a popup is displayed warning them that their progress will not be saved.

The component takes four parameters, the first parameter is the text that will describe the purpose of the popup to the user. The second parameter tells the app what to do if the user presses the "yes" button on the popup. The third popup tells the app what to do if the user presses the "no" button. The final parameter tells the app when to display the popup.

One problem that was encountered while creating this component was the need to make the popup appear at the front of the screen. The best solution to do this is to add the CSS line "zIndex: 999" which ensures it will be displayed in front of all other elements. Prior to this solution, there were some UI issues that were caused by overlapping elements.

6.2.3.5 Colour Scheme Button

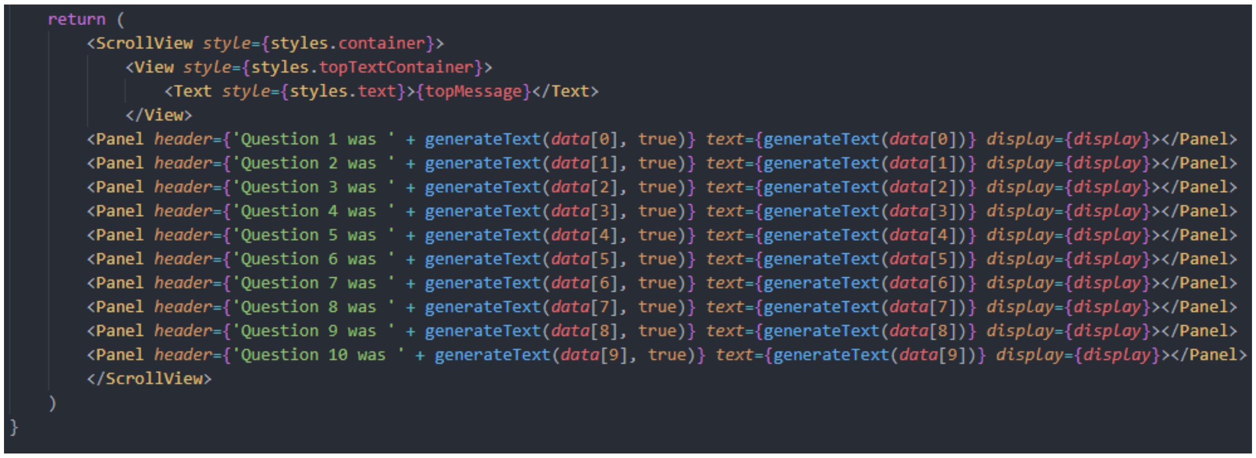

components > ColourSchemeButton.js > ...

```
import React from 'react';
import { StyleSheet, TouchableOpacity} from 'react-native';


export default function ColourButton({ onPress, colour }) {

    const styles = StyleSheet.create({
        button: {
            justifyContent: 'center',
            backgroundColor: colour,
            height: '40px',
            width: '40px',
            borderRadius: '50%',
            marginHorizontal: '2.5%'
        },

    });

    return (
        <TouchableOpacity style={styles.button} onPress={onPress} />
    )
}

```

Figure 37 - The code for the "Popup" component.

Figure 37 shows the code for the ColourButton component. This is the component which is used when changing the colour scheme of the app. There is not much to say here as the component is simple. It is a button which can be provided a colour and a function to execute when pressed. The component takes two parameters, firstly the function to be executed when it is pressed and secondly, the colour it should be. This allows it to be updated when the user changes the colour scheme.

6.2.3.6 Test Feedback & Panel Drop Down



```
    return (
        <ScrollView style={styles.container}>
            <View style={styles.topTextContainer}>
                <Text style={styles.text}>{topMessage}</Text>
            </View>
        <Panel header={'Question 1 was ' + generateText(data[0], true)} text={generateText(data[0])} display={display}></Panel>
        <Panel header={'Question 2 was ' + generateText(data[1], true)} text={generateText(data[1])} display={display}></Panel>
        <Panel header={'Question 3 was ' + generateText(data[2], true)} text={generateText(data[2])} display={display}></Panel>
        <Panel header={'Question 4 was ' + generateText(data[3], true)} text={generateText(data[3])} display={display}></Panel>
        <Panel header={'Question 5 was ' + generateText(data[4], true)} text={generateText(data[4])} display={display}></Panel>
        <Panel header={'Question 6 was ' + generateText(data[5], true)} text={generateText(data[5])} display={display}></Panel>
        <Panel header={'Question 7 was ' + generateText(data[6], true)} text={generateText(data[6])} display={display}></Panel>
        <Panel header={'Question 8 was ' + generateText(data[7], true)} text={generateText(data[7])} display={display}></Panel>
        <Panel header={'Question 9 was ' + generateText(data[8], true)} text={generateText(data[8])} display={display}></Panel>
        <Panel header={'Question 10 was ' + generateText(data[9], true)} text={generateText(data[9])} display={display}></Panel>
        </ScrollView>
    )
}
```

Figure 38 - The code for the "Popup" component.

Figure 38 shows the code for the test feedback component, used for displaying the feedback to the user at the end of their test. The component is essentially a lot of panel components stacked ontop of eachother. It was considered to have the components dynamically generated; however, every test in the app is the same length, so it is actually more efficient to use this method. The data which populates the component is generated dynamically however based on the user's test score.

6.2.4 Header & Home Page

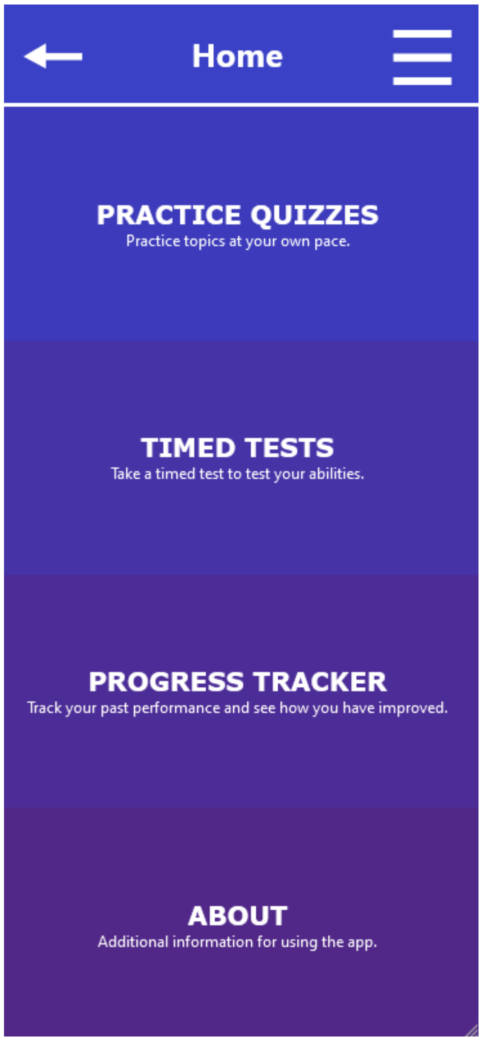

Home
PRACTICE QUIZZES
Practice topics at your own pace.
TIMED TESTS
Take a timed test to test your abilities.
PROGRESS TRACKER
Track your past performance and see how you have improved.
ABOUT
Additional information for using the app.

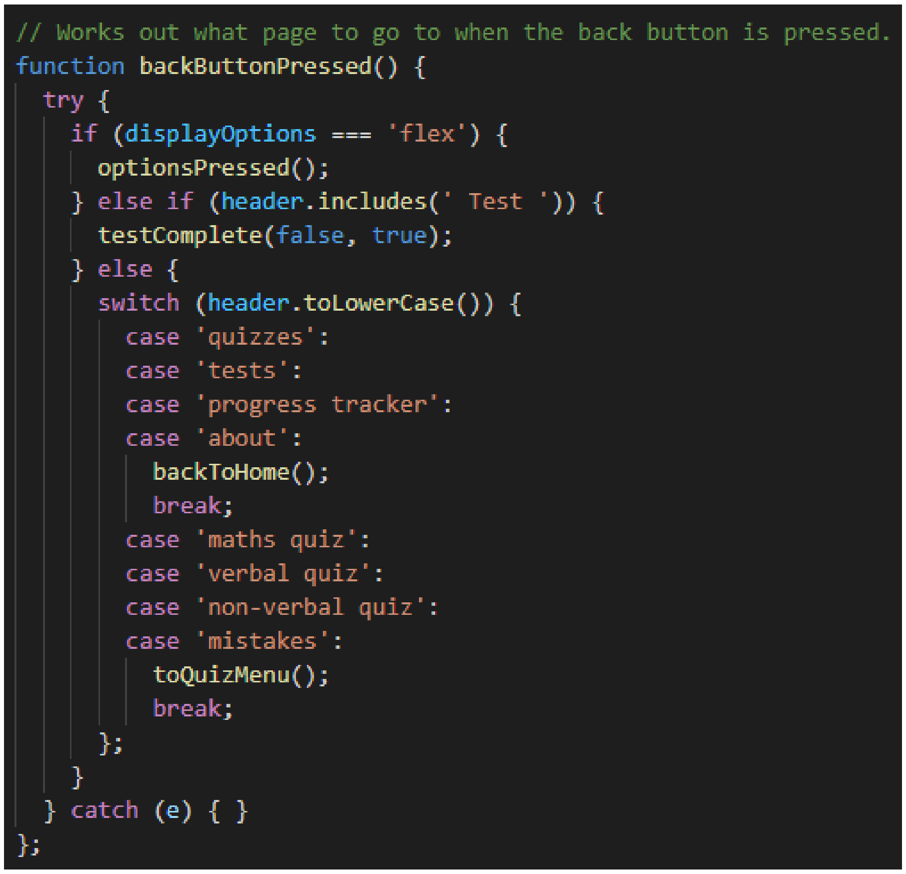
Figure 39 shows the implementation of the Home screen. The home screen was the first core component of the app to be implemented because it is the page from which all other features are derived from.

The header was the first part to be implemented. The reason for implementing the header first is that it allows for easier UI size and scaling throughout the app. By default, the header component has the background colour #3A41C6 to ensure it fits with the gradient colour scheme of the app. Additionally, the header dimensions are designed to be scalable, the height of the header is 10% of the screen height so it will always be a reasonable size regardless of the size of the screen.

The second part of the home screen that was implemented was the navigation buttons below the header. They use the “NavigationButton” component mentioned in 6.2.3.1. The navigation is fully scalable, so it will work on a wide array of devices regardless of screen size. Its height is always 90% of the total screen space. Additionally, the buttons are styles using the colours #3D3BBB, #4634A7, #4C2C96, #512888 so that they fit with the gradient colour scheme of the application.

Figure 39 - The implementation of the home page.

6.2.4.1 Back Button Functionality



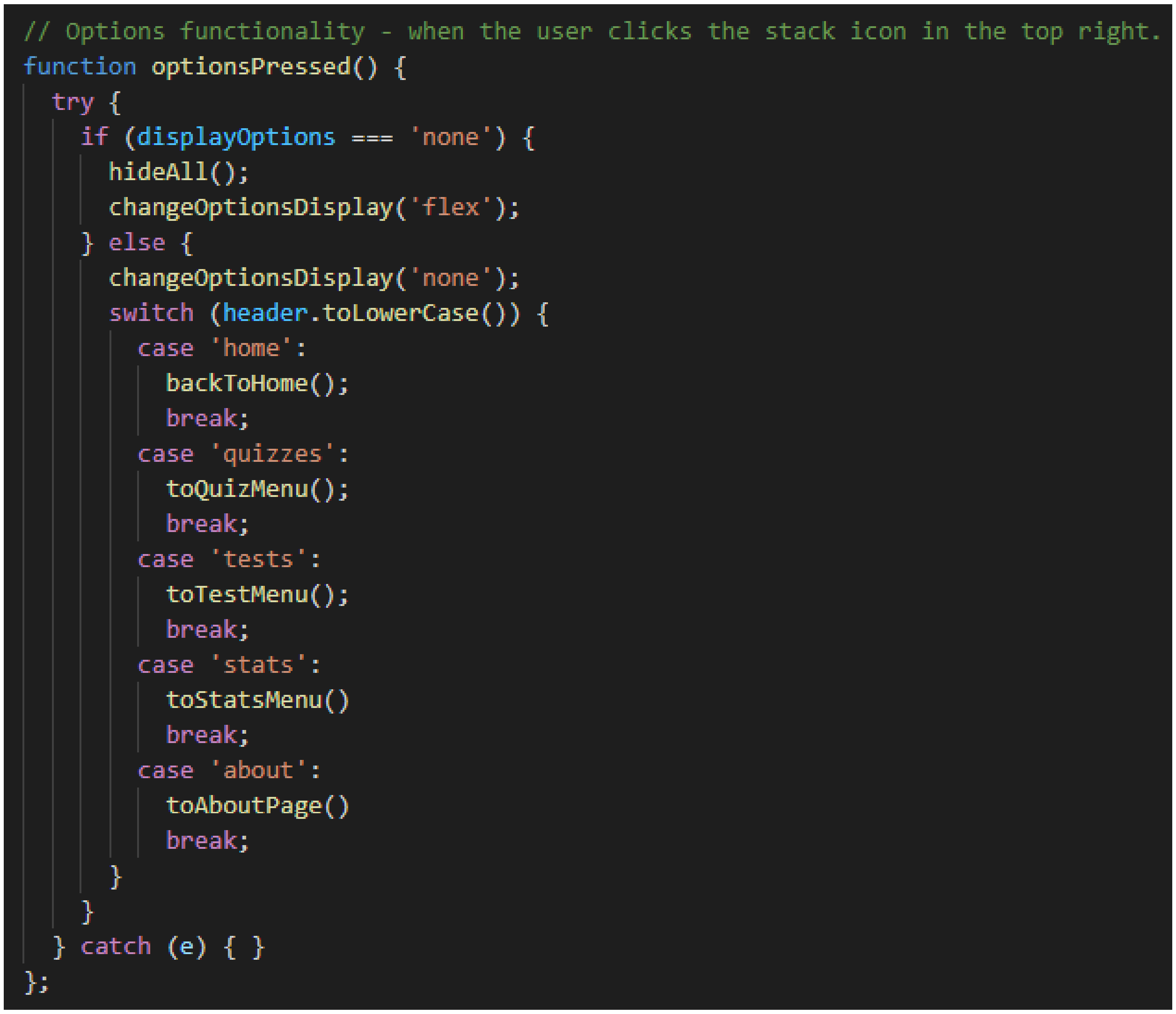
```
// Works out what page to go to when the back button is pressed.
function backButtonPressed() {
  try {
    if (displayOptions === 'flex') {
      optionsPressed();
    } else if (header.includes(' Test ')) {
      testComplete(false, true);
    } else {
      switch (header.toLowerCase()) {
        case 'quizzes':
        case 'tests':
        case 'progress tracker':
        case 'about':
          backToHome();
          break;
        case 'maths quiz':
        case 'verbal quiz':
        case 'non-verbal quiz':
        case 'mistakes':
          toQuizMenu();
          break;
      };
    }
  } catch (e) { }
};
```

Figure 40 - The handler for pressing the back button.

Figure 40 shows the function that provides the functionality to the back arrow located within the header. The function looks at what the current value of the header title is and then makes a decision about which page to load up based on it's value. For example, if the user was on the "Tests" page, then the function would take them back to the home page.

The reason for implementing the back button functionality using conditional statements that check the header is that it is the most simple way to implement this type of feature. Other methods were researched such as implementing a navigation stack. However, due to the linear nature of the application's navigation, it would have been unnecessary while needlessly over complicating it.

6.2.4.2 Options Button Functionality



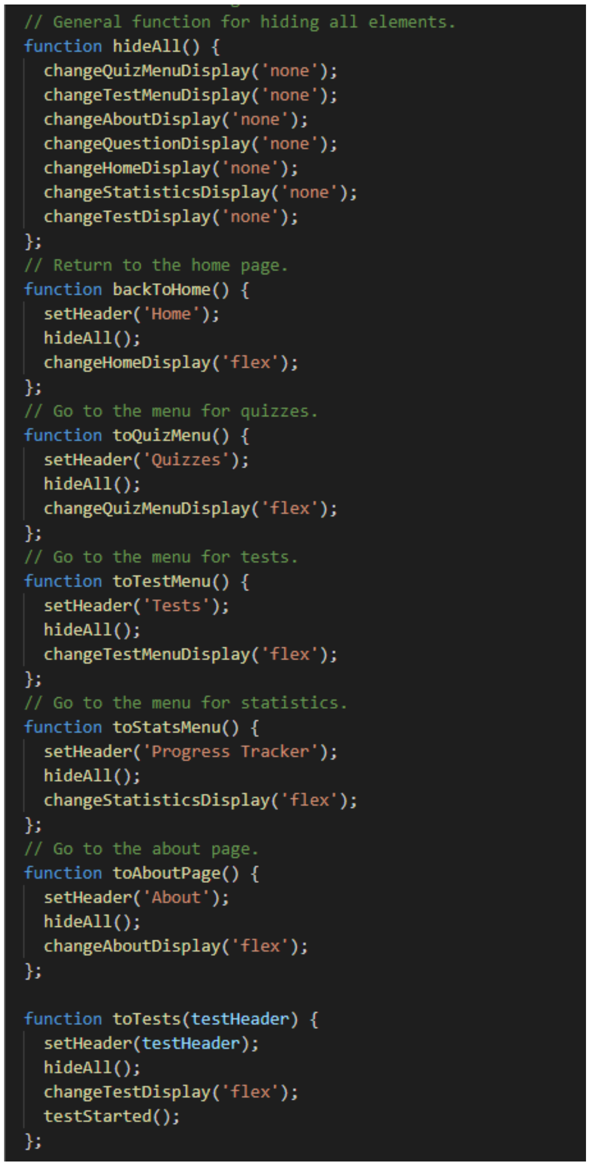
```
// Options functionality - when the user clicks the stack icon in the top right.
function optionsPressed() {
  try {
    if (displayOptions === 'none') {
      hideAll();
      changeOptionsDisplay('flex');
    } else {
      changeOptionsDisplay('none');
      switch (header.toLowerCase()) {
        case 'home':
          backToHome();
          break;
        case 'quizzes':
          toQuizMenu();
          break;
        case 'tests':
          toTestMenu();
          break;
        case 'stats':
          toStatsMenu()
          break;
        case 'about':
          toAboutPage()
          break;
      }
    }
  } catch (e) { }
};
```

Figure 41 - The handler for pressing the options button.

Figure 41 shows the function for displaying the options menu by pressing the button on the right of the header. The function checks the state of the variable “displayOptions”. If the variable is set to “none” then it means the options are hidden so the app will display them. If the variable is set to “flex” it means the options are shown meaning the app will hide them. The function uses the header to determine what page to send the user back to once the options page is hidden.

The rationale behind the design of this function is similar to the back button. Using conditional statements is a simple and effective way of getting the desired functionality without over complicating it.

6.2.4.3 Navigation Functionality



```
// General function for hiding all elements.
function hideAll() {
  changeQuizMenuDisplay('none');
  changeTestMenuDisplay('none');
  changeAboutDisplay('none');
  changeQuestionDisplay('none');
  changeHomeDisplay('none');
  changeStatisticsDisplay('none');
  changeTestDisplay('none');
};
// Return to the home page.
function backToHome() {
  setHeader('Home');
  hideAll();
  changeHomeDisplay('flex');
};
// Go to the menu for quizzes.
function toQuizMenu() {
  setHeader('Quizzes');
  hideAll();
  changeQuizMenuDisplay('flex');
};
// Go to the menu for tests.
function toTestMenu() {
  setHeader('Tests');
  hideAll();
  changeTestMenuDisplay('flex');
};
// Go to the menu for statistics.
function toStatsMenu() {
  setHeader('Progress Tracker');
  hideAll();
  changeStatisticsDisplay('flex');
};
// Go to the about page.
function toAboutPage() {
  setHeader('About');
  hideAll();
  changeAboutDisplay('flex');
};

function toTests(testHeader) {
  setHeader(testHeader);
  hideAll();
  changeTestDisplay('flex');
  testStarted();
};
```

Figure 42 - The handlers for the app navigation.

Figure 42 shows how the navigation is implemented. For each page in the app, there is a function which, when called, will take you to that page. The first thing each function does is change the header. Then the “hideAll” function is run which is a catch-all function that hides every page in the app. The reason for implementing this function is safety, if a page is accidentally marked as visible when it shouldn’t be and we tried to transition to another page it would break the functionality of parts of the app. So, hiding all pages is there to prevent this. Finally, the function sets the display of the page we want to go to visible.

The rationale behind having it done as functions is because it avoids repeated code. There are lots of places in the code where these functions are called so it is easier to have them in functions that can be called. Additionally, it helps with maintainability as it means if a change is needed only one part of the code needs to be updated.

Before the implementation of the “hideAll” function, there were issues with the navigation not working properly and the code was far less readable. This solution has solved this problem.

6.2.5 Quiz Selector

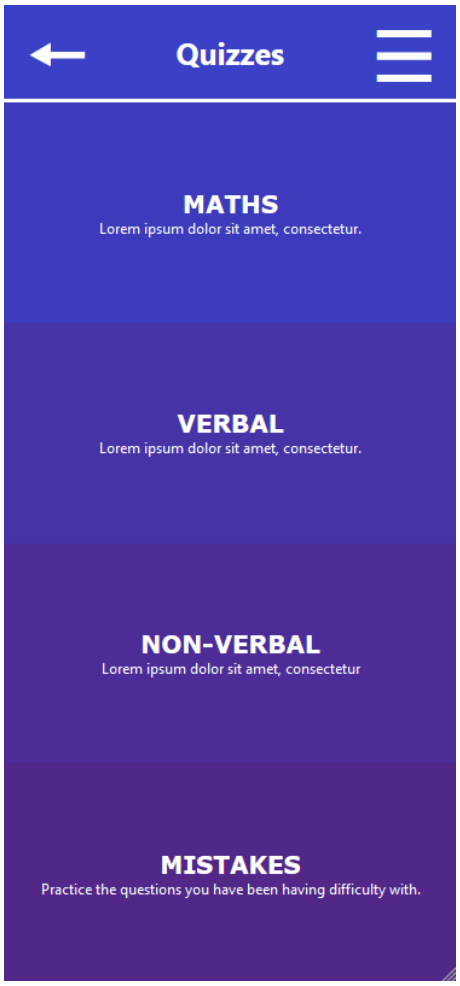

Quizzes
MATHS
Lorem ipsum dolor sit amet, consectetur.
VERBAL
Lorem ipsum dolor sit amet, consectetur.
NON-VERBAL
Lorem ipsum dolor sit amet, consectetur
MISTAKES
Practice the questions you have been having difficulty with.

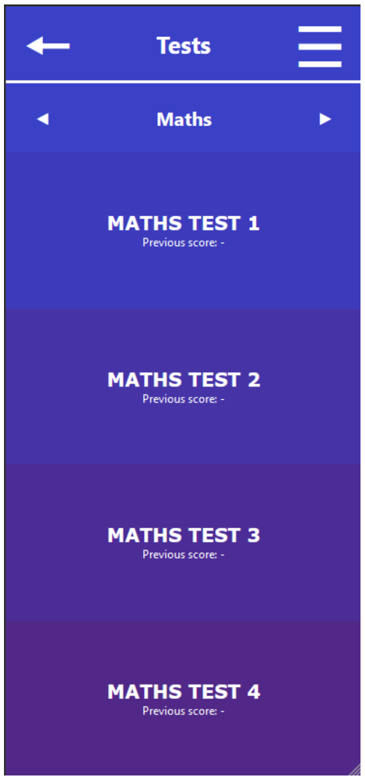
Figure 43 - The implementation of the quiz menu page.

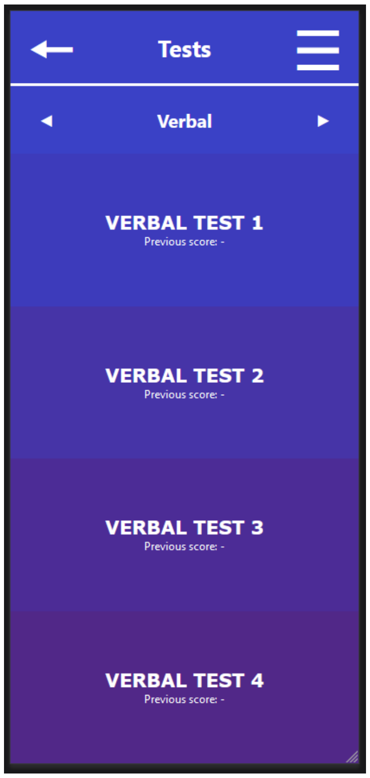
Figure 43 shows the UI implementation of the quiz selection page. The implementation of this page was done using four of the navigation button components mentioned in section 6.2.3.1. This means that the entire UI for this page only takes up has been implemented in only four lines of JSX.

Each component takes the same function when pressed, with the only difference being a parameter that defines what questions are to be loaded. The reason this is possible is that each button essentially takes you to the same page - being the quiz questions page. The only difference is the questions that need to be loaded.

Initially, the approach was to use separate functions for each navigation component, however, the current implementation that uses a single function is more efficient, clean and maintainable.

6.2.6 Test Selector





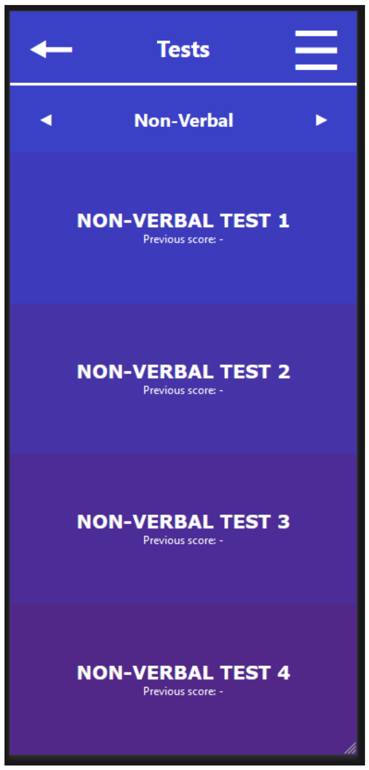


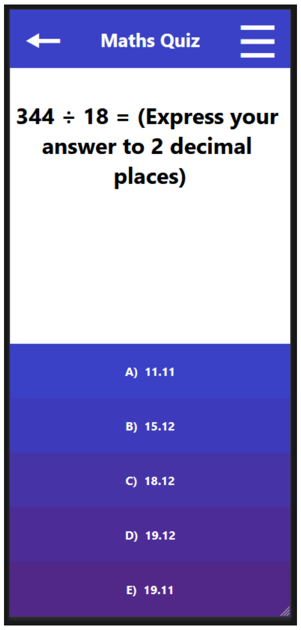
Figure 44 - Test selection implementation (maths).
Figure 45 - Test selection implementation (verbal).
Figure 46 - Test selection implementation (non-verbal).

Figures 44, 45 and 46 show the UI implementation of the test selection page. Functionally speaking, this page is similar to the quiz navigation shown in section 6.2.6. The efficient implementation using components and a singular function is also present here.

However, there are a few important differences between the implementation of this page and the implementation detailed in section 6.2.6. Firstly, to enable the user to distinguish between the subjects there is a subheader selector. This works by updating the state of all the navigation buttons to whatever is in the header. Then when a navigation button is pressed, the title is passed to the function to enable the app to determine what questions need to be loaded up. This method of implementation is beneficial because it means that additional components are not required when shifting between subjects.

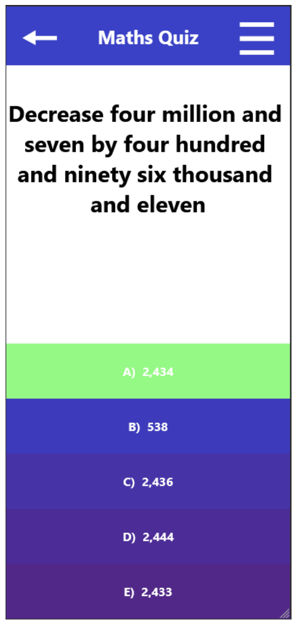
Furthermore, the test selection page also needs to display the users best previous score in each test. This is shown below the main navigation text. The implementation of this is planned in the second spiral with only the UI being completed currently as this is not an essential feature.

6.2.7 Question Page Implementation


Maths Quiz

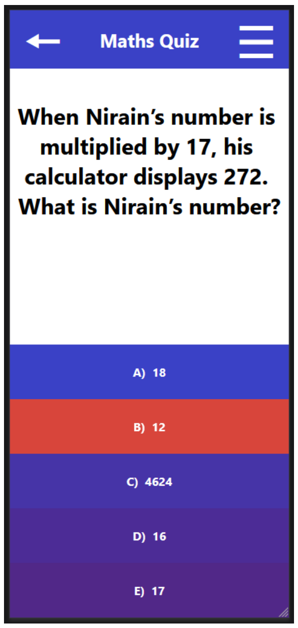
344 ÷ 18 = (Express your answer to 2 decimal places)

A) 11.11
B) 15.12
C) 18.12
D) 19.12
E) 19.11


Maths Quiz

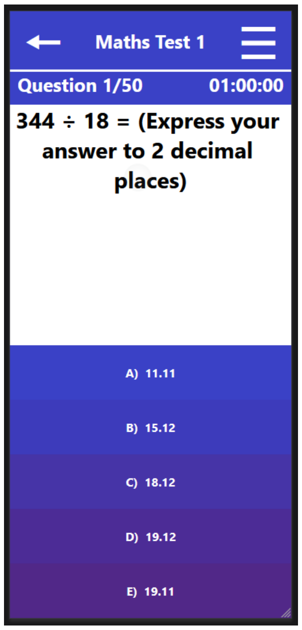
Decrease four million and seven by four hundred and ninety six thousand and eleven

A) 2,434
B) 538
C) 2,436
D) 2,444
E) 2,433


Maths Quiz

When Nirain's number is multiplied by 17, his calculator displays 272. What is Nirain's number?

A) 18
B) 12
C) 4624
D) 16
E) 17


Maths Test 1

Question 1/50 01:00:00

344 ÷ 18 = (Express your answer to 2 decimal places)

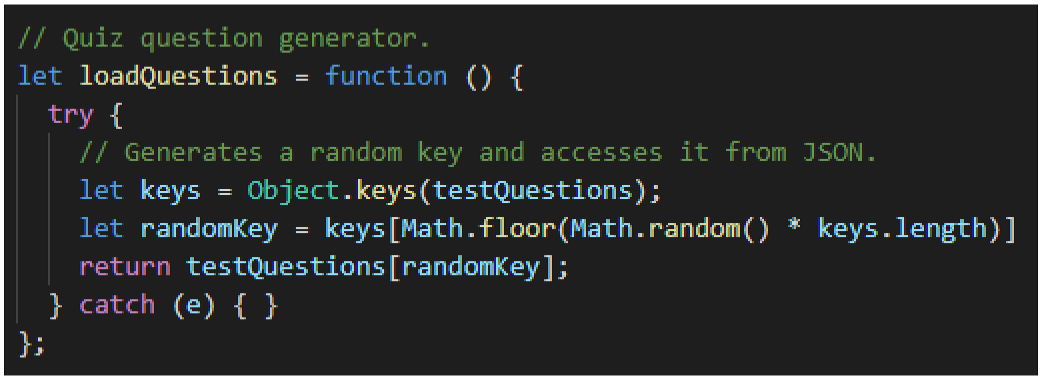
A) 11.11
B) 15.12
C) 18.12
D) 19.12
E) 19.11

Figure 47 - Quiz question implementation.
Figure 48 - Quiz question implementation (correct answer).
Figure 49 - Quiz question implementation (wrong answer).
Figure 50 - Test question implementation.

Figures 47, 48, 49 and 50 show the question pages of the app. Figure 47, 48 and 49 showing the implementation for the quiz questions and figure 50 shows the implementation for the tests.

Both quiz and test question pages have the same question layout. There are five options that the user can select for each question located in the bottom half of the screen. These answer buttons are made up of the question button component mentioned in section 6.2.3.3. When an answer button is pressed, a function is called. This function takes the answer that the user gave and compares it with the actual answer. A true or false value is then returned. From here the background of the question button pressed can be updated to either green (correct) or red (wrong). The reason for doing this is to provide instant visual feedback to the user on whether or not they have got their question correct.
Tests are timed which means that it is necessary to include a subheader that tells the user how much time they have left and how many questions are left (figure 50). This section simply uses the subheader component mentioned in section 6.2.3.2. The timer works by having a variable set to 3600 (one hour in seconds) and every second one is subtracted from the variable. The new

number is then translated to the hh:mm:ss format. One big issue that was experienced with the implementation was resetting the timer once a test was over. The problem was the fact that the timer wouldn't reset properly meaning it would get completely messed up when a new test was started. Fixing this required the changing of the scope of the timer declaration to be global. This allowed it to be better accessed by the timer reset functions.

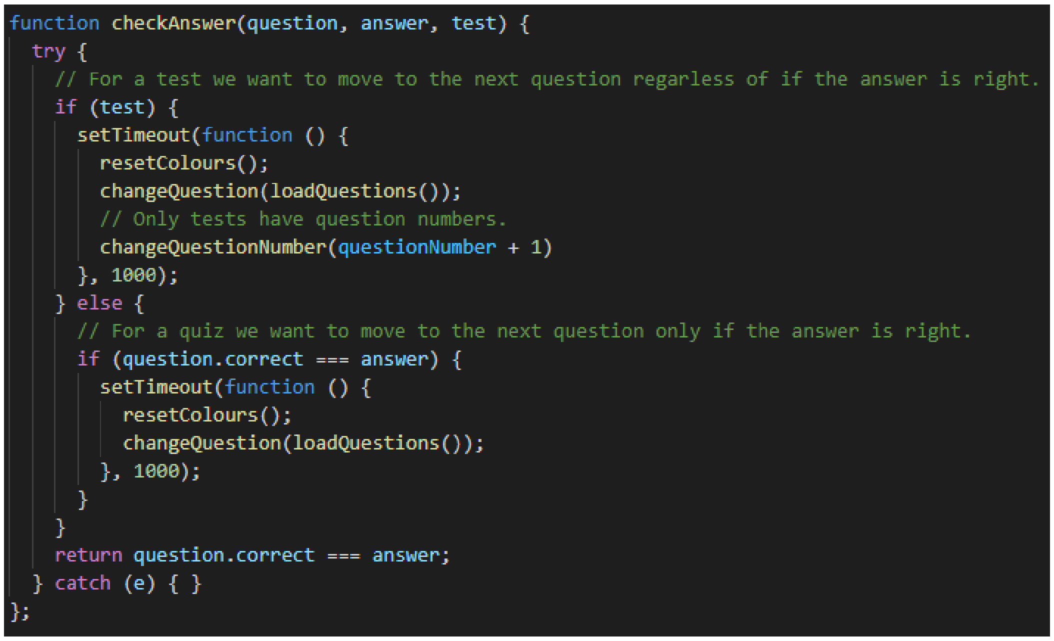


```
// Quiz question generator.
let loadQuestions = function () {
  try {
    // Generates a random key and accesses it from JSON.
    let keys = Object.keys(testQuestions);
    let randomKey = keys[Math.floor(Math.random() * keys.length)]
    return testQuestions[randomKey];
  } catch (e) { }
};
```

Figure 51 - Function to generate quiz questions from a JSON object.

One key difference between the quiz and test implementation is how questions are loaded. For tests the implementation is relatively simple, there is a JSON object that contains all the questions needed and the app simply goes through them sequentially in a linear order. However, for quizzes, this method is not ideal. This is because quizzes contain hundreds of questions for the user to practice. So, if quizzes using the same loading method as tests the user would likely not see most of the questions available and it would get very repetitive. The app, therefore, implements a random question loading system for quizzes (figure 51). The way this system works is it takes all the potential questions and then generates a random key. The question with the corresponding key is then selected. This method is ideal because it solves the quiz loading issue previously mentioned in a very short amount of code.

One potential downside to the random question generation system is that the user may end up seeing the same questions multiple times. Two solutions were examined to address this problem. Firstly, a skip question button allows the user to skip a question. Secondly, logging keys that had already been seen and preventing them from coming up. However, it was ultimately decided that this was unnecessary, and a potential waste of development time, as the client plans to put hundreds and potentially thousands of questions on the app. This means the chance of repetition is very small. Furthermore, it is not a major issue as the user can just select the answer and move on.



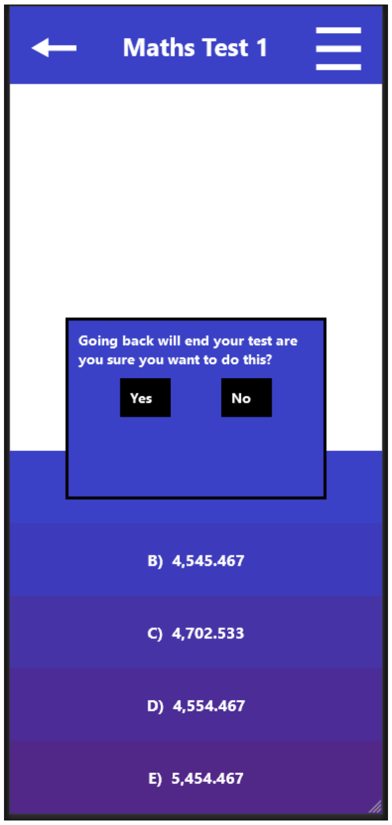
```javascript
function checkAnswer(question, answer, test) {
  try {
    // For a test we want to move to the next question regarless of if the answer is right.
    if (test) {
      setTimeout(function () {
        resetColours();
        changeQuestion(loadQuestions());
        // Only tests have question numbers.
        changeQuestionNumber(questionNumber + 1)
      }, 1000);
    } else {
      // For a quiz we want to move to the next question only if the answer is right.
      if (question.correct === answer) {
        setTimeout(function () {
          resetColours();
          changeQuestion(loadQuestions());
        }, 1000);
      }
    }
    return question.correct === answer;
  } catch (e) { }
};
```

Figure 52 - Function that checks if the user's answer was correct.

Another difference between how quizzes and tests have been implemented is in what happens when the user gets an answer wrong. When the user gets an answer correct both simply move on to the next question. However, for quizzes, the question does not change if the user gets an answer wrong. For tests, the question changes regardless of the user's answer. The reason for this difference is because the aim of a quiz is to practice so if the user gets an answer wrong they need chances to get it right. Whereas a test is to assess knowledge so the user will only receive feedback at the end.

Figure 52 shows the code which provides this functionality to both quizzes and tests. This was achieved by simply having a parameter that identifies if the user is on a test or a quiz. Then a conditional statement sets the functionality accordingly. There is an additional condition that only moves onto the next question for quizzes if the answer is correct. Originally, this problem was solved by using two functions. However, it was changed to this single function method which is superior because having all the functionality in one function makes it more maintainable and efficient.

A decision that was made here was to artificially add a one-second delay in the transition of changing questions. This was done to give the user a small amount of time to acknowledge their result and to help improve pacing between questions. When the transition was instant it felt too quick.

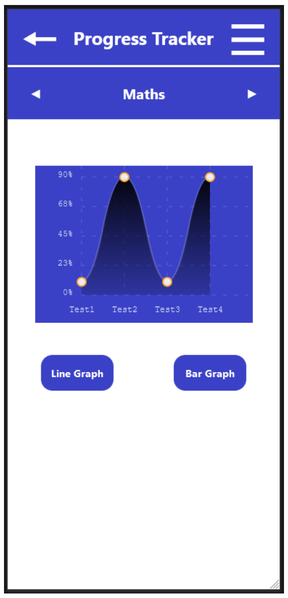

Maths Test 1

Going back will end your test are you sure you want to do this?

Figure 53 - Test exit popup.

Finally, for tests, it is important for users not to prematurely exit unless they really want to as it could cause them to lose progress. To address this a pop-up was implemented (figure 53) if the user tried to exit a test. This uses the popup component mentioned in section 6.2.3.4. This method is ideal because it uses an already existing component and thus is easy to implement and maintain.

6.2.8 Progress Tracker Page



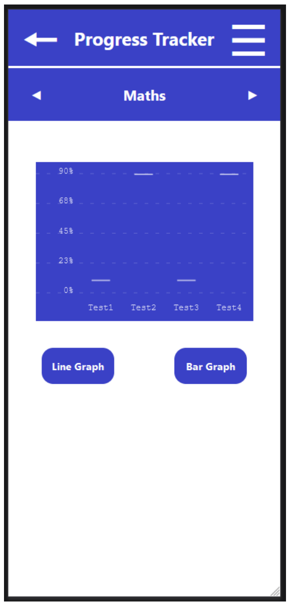


Figure 54 - Progress tracker: line graph.
Figure 55 - Progress tracker: bar graph.

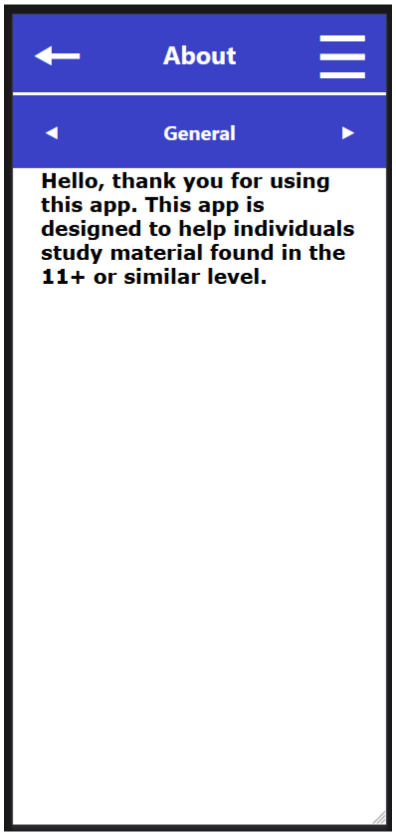
Figures 54 and 55 show the implementation of the progress tracker for some dummy maths data. The progress tracker displays the results of the user's test scores in a graph so that the user can see their performance over time. There are two buttons that allow the user to select the type of data representation they want from either a line graph or a bar chart.

The graphs were implemented using the "React Native Chart Kit" package. Which provides many options for representing data using scalable vector graphics. The package was installed using the NPM command: "npm i react-native-chart-kit". The reason this library was used is that it provides all the functionality needed in an easy to use pre-build package. It was initially considered to build the components manually however, this would take a lot of time, and the end result would likely not be as polished as this method.

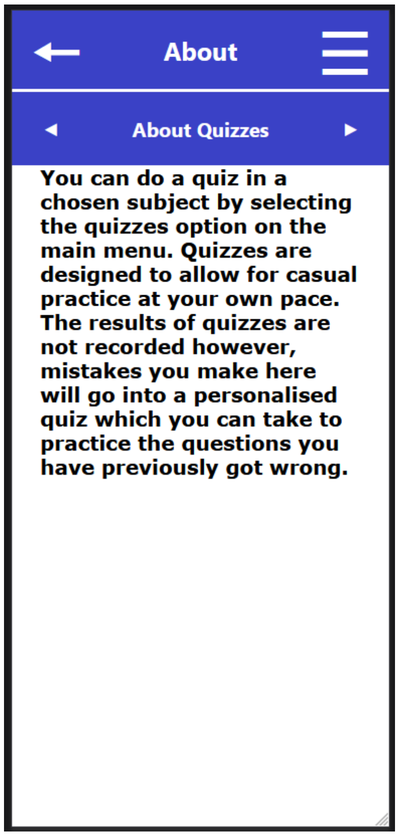
There were a couple of issues encountered during the implementation. Firstly, shown in figure 55, there is a graphical issue with the bar chart display. It was identified that this was caused when the component's colour state was updated when transitioning to the bar chart. To fix this it was necessary to ensure that the colour was updated alongside the state change.

Additionally, another limitation that was identified was that due to limited screen space only a certain amount of results can be shown on the screen at any given time. Otherwise, the graph becomes unreadable. To solve this, there is currently a cap of six items in the graph at any one time. When a new piece of data is added the oldest is deleted. However, further discussion with the client will take place after this spiral to determine potential solutions for the second spiral.

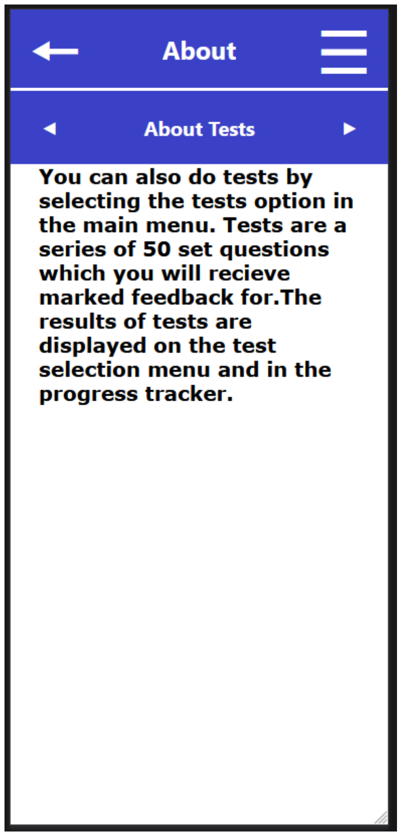
6.2.9 About Page



Hello, thank you for using this app. This app is designed to help individuals study material found in the 11+ or similar level.



You can do a quiz in a chosen subject by selecting the quizzes option on the main menu. Quizzes are designed to allow for casual practice at your own pace. The results of quizzes are not recorded however, mistakes you make here will go into a personalised quiz which you can take to practice the questions you have previously got wrong.



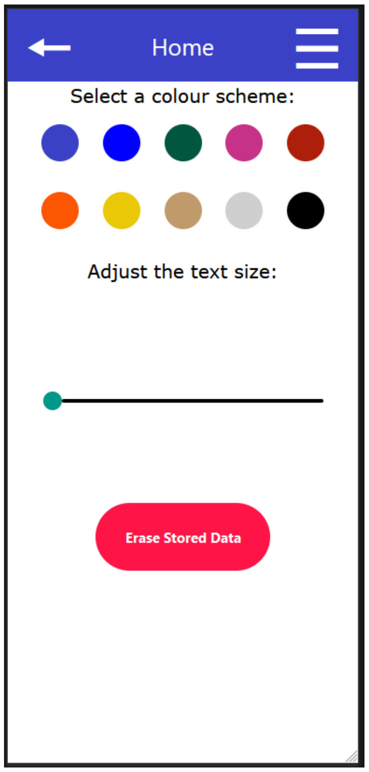
You can also do tests by selecting the tests option in the main menu. Tests are a series of 50 set questions which you will recieve marked feedback for.The results of tests are displayed on the test selection menu and in the progress tracker.

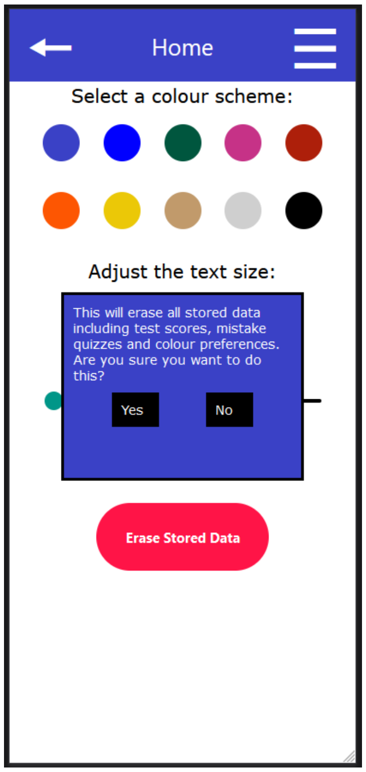
Figure 56 (left) - About page, on the general help screen.
Figure 57 (centre) - About page, on the about quizzes help screen.
Figure 58 (right) - About page, on the about tests help screen.

Figures 56, 57, and 58 show some of the implementation of the about (tutorial) page in some of its potential states. The implementation is simple, there is a text component whose state is updated every time the user selects a new topic from the subheading selector. This method is efficient as it eliminates the need to have multiple components defined for each section which is a method which was initially explored.

An issue that was encountered was that text was not displaying properly when the user initially loaded up this page. The reason for this was because the initial declaration of the text variable was located after the initial state assignment, meaning that it was null. Resolving this was simply a matter of moving the variable declaration above the state assignment so it had a value.

6.2.10 Settings Page





This will erase all stored data including test scores, mistake quizzes and colour preferences. Are you sure you want to do this?

Figure 59 - Implementation of the settings page
Figure 60 - Settings page popup

Figure 59 shows the implementation of the settings page. The first part of the settings page is a list of buttons, created using the "colourButton" component detailed in 6.2.3.5, which allow the user to change the colour scheme of the application. The section is clearly labelled by a title. Clicking on one of the coloured buttons will change the colour scheme to the colour shown on the button.

The second part of the settings page is a slider that allows the user to adjust the text size of all text throughout the app. The way it works is that the slider acts as a percentage multiplier from -30% to +30% which can then be multiplied onto the existing text sizes. The text size is then stored in asynchronous storage so that the user's preferences are remembered. This method is ideal because it preserves the relative scale of the different texts in comparison to each other. The slider component uses the community slider package (https://github.com/callstack/react-native-slider). Initially, the slider was done using the slider component however, this is deprecated so the community package is more suitable.

Finally, there is a button which when pressed will reset all of the data on the app to the default state. There is a popup (figure 60) that will warn the user so that they do not accidentally delete their data.

6.3 Second Spiral/Prototype

6.3.1 Introduction

This section will discuss the features that were implemented in the second spiral of the implementation. These features developed here are lower priority or were built upon existing functionality based on the client's feedback. The second spiral is shorter than the first spiral as there were fewer features that required developing and the client was generally happy with the features in the first spiral so there was a limited amount of existing functionality that needed reworking.

Table 6.2 - Order of implementation for the second spiral

Order of completion	**Task**
1	Colour scheme changing.
2	Post-test feedback.
3	Mistakes Quiz
4	Splash screen.
5	Progress tracker enhancements
6	General UI improvements and bug fixing

6.3.2 Colour scheme changing

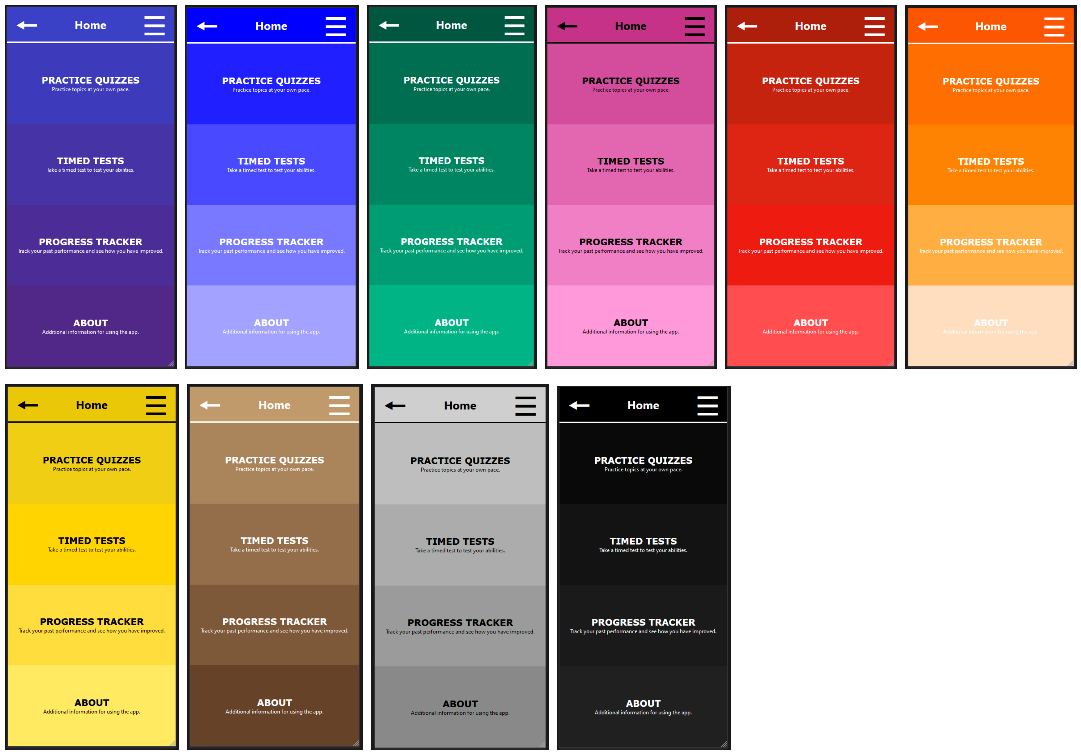
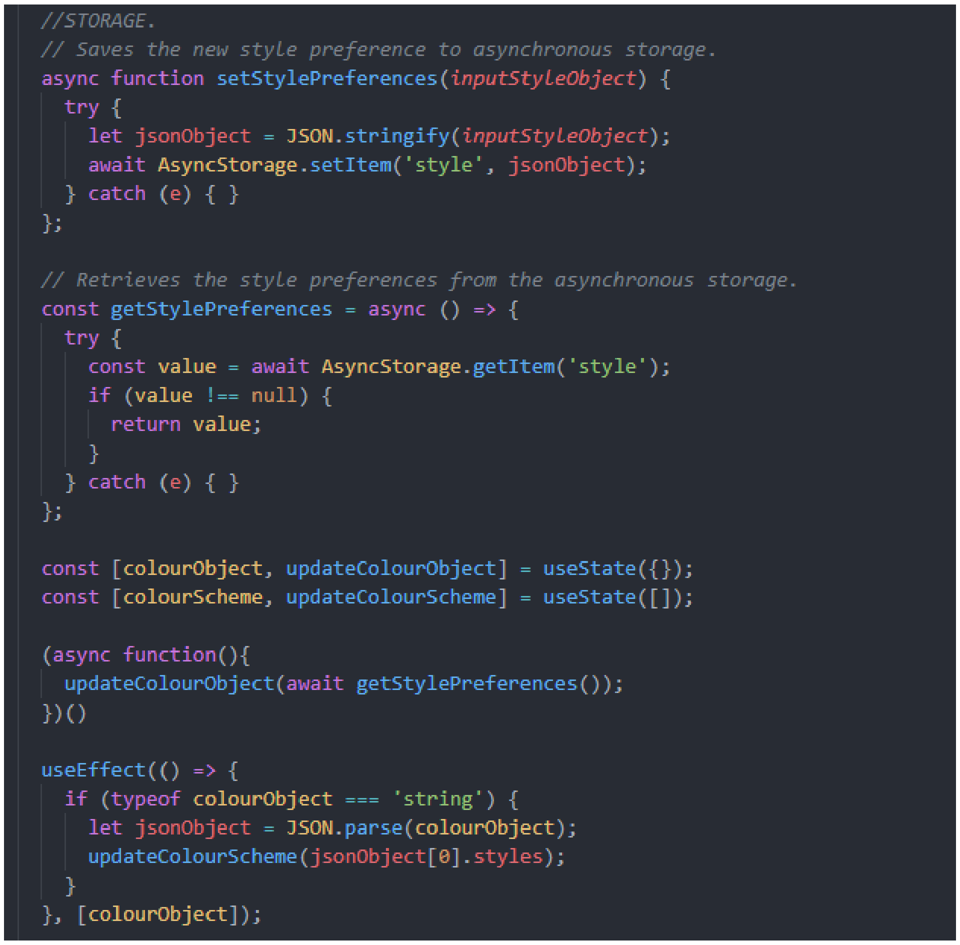

Home
PRACTICE QUIZZES
Practice topics at your own pace.
TIMED TESTS
Take a timed test to test your abilities.
PROGRESS TRACKER
Track your past performance and see how you have improved.
ABOUT
Additional information for using the app.

Figure 61 - App colour schemes.

Figure 61 shows the homepage in the different colour schemes that the app can be set to. There are ten possible colour schemes as mentioned in the design section. The justification for choosing these colour schemes is that each colour scheme contains high contrast gradient colouring to make it as visible as possible to the user.

The reason that this feature was implemented in the second spiral rather than the first is that it makes sense to implement colour switching after all of the core components of the app are created, as they will all need to have their colours switched. Additionally, the feature was not listed as a high requirement in section 4.0 and is not crucial to the running of the app.



```
//STORAGE.
// Saves the new style preference to asynchronous storage.
async function setStylePreferences(inputStyleObject) {
  try {
    let jsonObject = JSON.stringify(inputStyleObject);
    await AsyncStorage.setItem('style', jsonObject);
  } catch (e) { }
};

// Retrieves the style preferences from the asynchronous storage.
const getStylePreferences = async () => {
  try {
    const value = await AsyncStorage.getItem('style');
    if (value !== null) {
      return value;
    }
  } catch (e) { }
};

const [colourObject, updateColourObject] = useState({});
const [colourScheme, updateColourScheme] = useState([]);

(async function(){
  updateColourObject(await getStylePreferences());
})()

useEffect(() => {
  if (typeof colourObject === 'string') {
    let jsonObject = JSON.parse(colourObject);
    updateColourScheme(jsonObject[0].styles);
  }
}, [colourObject]);
```

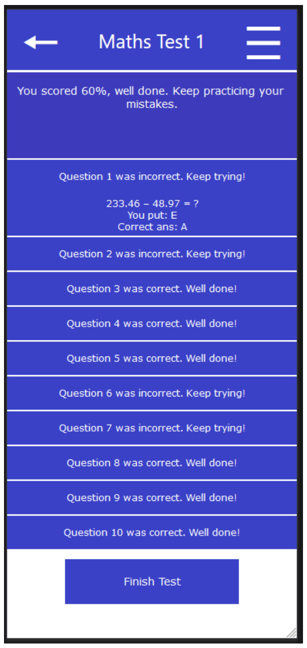
Figure 62 - Saving/Loading of colour schemes into storage.

The colour scheme uses Asynchronous Storage to store the data. There is a function "setStylePreferences" which when called takes a JSON object and saves it into the style key. This is how the app saves the user's choices. There is a function called "getStylePreferences" which retrieves the existing data. This is called when the app initially starts to determine what colour scheme needs to be set.

Additionally, the "useEffect" hook is also used to trigger an update to the colour scheme whenever it detects the colour scheme storage object has been changed.

There was a problem with implementation with asynchronous storage. The styles data object was being transferred as an array within a JSON object. However, using it as an array caused an infinite loop to occur as the useEffect function would be called each time, causing a re-render and causing it to be called again. To fix this, a stop condition was added to the useEffect function and the array was converted into a string.

6.3.3 Test results


Maths Test 1

You scored 60%, well done. Keep practicing your mistakes.

Question 1 was incorrect. Keep trying!

233.46 – 48.97 = ?
You put: E
Correct ans: A

Question 2 was incorrect. Keep trying!

Question 3 was correct. Well done!

Question 4 was correct. Well done!

Question 5 was correct. Well done!

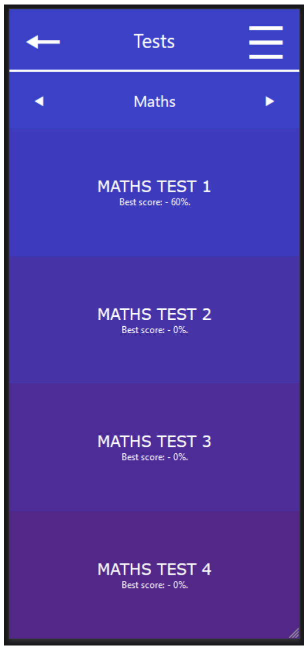
Question 6 was incorrect. Keep trying!

Question 7 was incorrect. Keep trying!

Question 8 was correct. Well done!

Question 9 was correct. Well done!

Question 10 was correct. Well done!



MATHS TEST 1
Best score: - 60%.

MATHS TEST 2
Best score: - 0%.

MATHS TEST 3
Best score: - 0%.

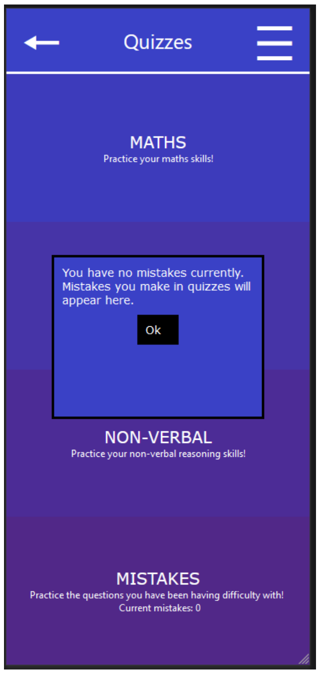
MATHS TEST 4
Best score: - 0%.

Figure 63 - Test results screen.
Figure 64 - Updated test menu.

Figure 63 shows the screen after the user has completed a test. The screen shows the percentage of correct answers the user gave during the test as well as feedback for each question. This was achieved by using the drop-down component from the react-native-drop-down-item library. Each question has a drop-down which displays whether or not the user got that question right. If the user wants further details all they have to do is click on the question and it will expand showing further information.

Additionally, figure 64 shows the updated test menu. In the previous spiral the test menu displayed the user's previous score. However, after consulting with the client this was changed to the user's best score as it was deemed this would be a more useful statistic. The text in each component has been updated accordingly. So in the case of figure 64, we can see that the user's best score in test #1 is 60%. While the UI implementation was done in the previous spiral, the functional implementation was straightforward, there is simply an object stored in Asynchronous Storage which contains the best score for each test. Every time the user completes a test their score is checked with the score in the storage. If they score better it is updated.

6.3.4 Mistakes Quiz


Quizzes
MATHS
Practice your maths skills!
NON-VERBAL
Practice your non-verbal reasoning skills!
MISTAKES
Practice the questions you have been having difficulty with!
Current mistakes: 0
Ok

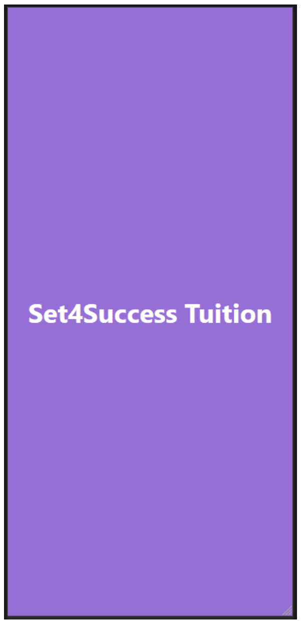
You have no mistakes currently. Mistakes you make in quizzes will appear here.

Figure 65 - Mistakes quiz popup.

The implementation of the mistakes quiz feature was achieved using Asynchronous Storage. Whenever the user takes a quiz, if they get a question wrong then the details of that question will be appended to a storage object. When the user then decides to take a quiz on their previous mistakes the app can then load the questions from this storage object. Additionally, to improve the usability of this feature, if the user tries to load the mistakes quiz and there are no questions available a popup will tell them of this - shown in figure 65.

While the implementation was relatively simple there was an issue that was encountered. If the user got the same question wrong multiple times it would be added to the mistakes object irrespective of if the question already was in that object resulting in duplicate questions being asked. To solve this, a check was added that loops through each question in the object and compares the name to the name of the object being added. If the names are the same then the duplicate is not added.

6.3.5 Splash Screen



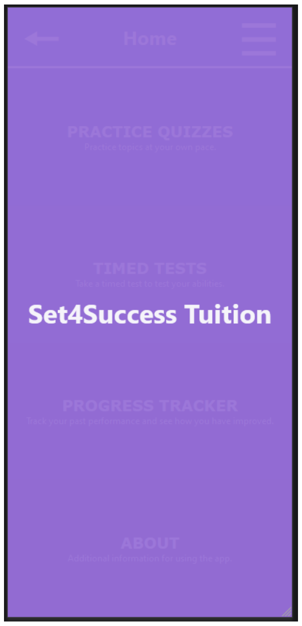
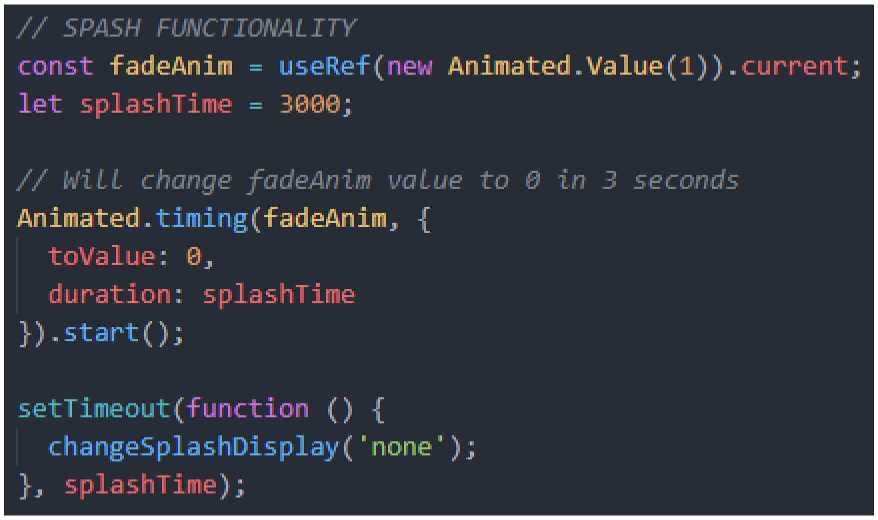


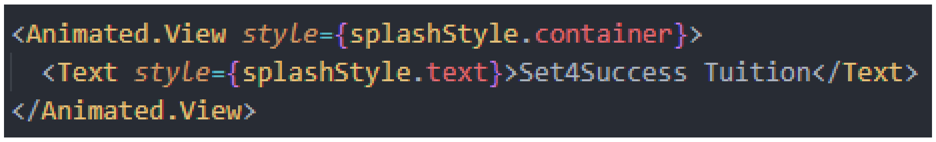
Figure 66 - Splash screen (full opacity).
Figure 67 - Splash screen (reduce opacity).

Figures 66 and 67 show the splash screen component in the app. The splash screen is text on a background that transitions from 100% opacity to 0% opacity over a 3-second time frame. The text displays the name of the company.



```
// SPASH FUNCTIONALITY
const fadeAnim = useRef(new Animated.Value(1)).current;
let splashTime = 3000;

// Will change fadeAnim value to 0 in 3 seconds
Animated.timing(fadeAnim, {
  toValue: 0,
  duration: splashTime
}).start();

setTimeout(function () {
  changeSplashDisplay('none');
}, splashTime);
```

Figure 68 - Splash screen code.



```
<Animated.View style={splashStyle.container}>
  <Text style={splashStyle.text}>Set4Success Tuition</Text>
</Animated.View>
```

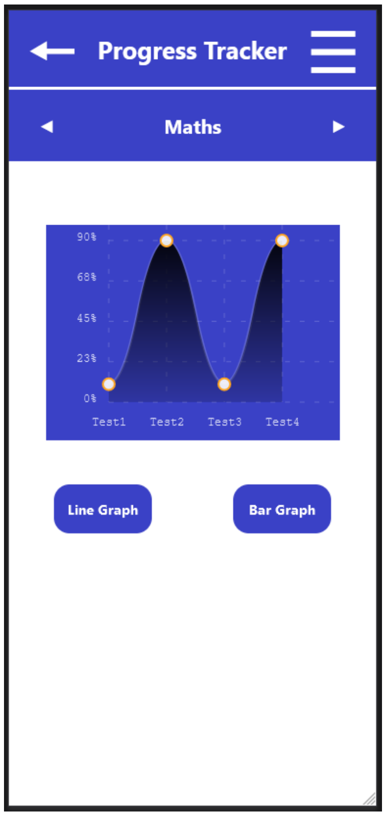
Figure 69 - Splash screen JSX.

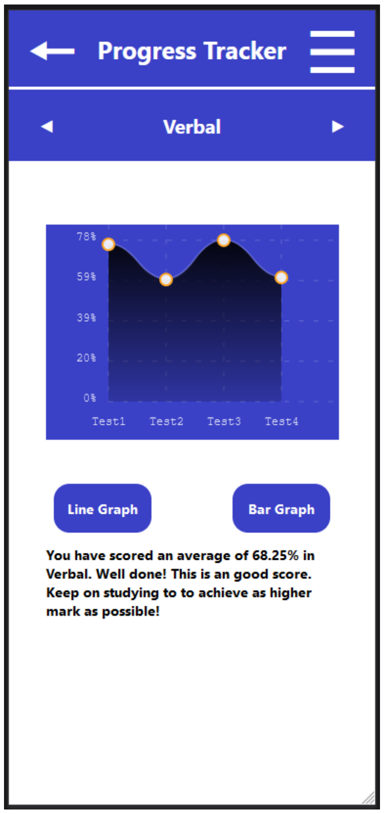
Figure 68 shows the code for the splash screen. The implementation was done using the React Native Animated library. This library provides easy animation functionality. The library provides special JSX that is shown in figure 69 and a special timer function shown in figure 68. (*Animated · React native*, https://reactnative.dev/docs/animated)

The implementation (figure 68) uses the variable "fadeAnim" to be the opacity value, initially set to 1 (100%). Then there is an animation timer that ticks for 3 seconds and reduces the opacity to 0 over this time. Finally, there is a timeout that hides the splash screen once it has disappeared. This is necessary to enable clicking functionality on other components.

Initially, the splash screen was attempted to be implemented using an interval updating an opacity state variable every 300 milliseconds. However, this solution was not ideal because it resulted in duplicate intervals every time the app needed to re-render to access asynchronous storage. This likely could have been resolved but it was easier to use the dedicated Animated library and also saved a lot of time.

6.3.6 Progress tracker enhancements



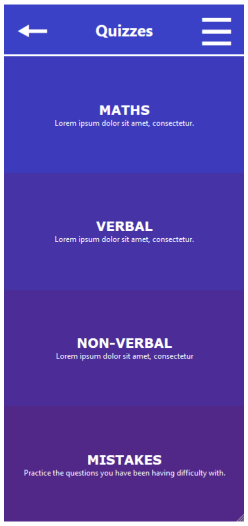


You have scored an average of 68.25% in Verbal. Well done! This is an good score. Keep on studying to to achieve as higher mark as possible!

Figure 70 - Previous progress tracker implementation.
Figure 71 - Progress tracker implementation after client feedback in the second spiral.

Figure 70 shows the progress tracker implemented in the first spiral. However, the client provided some feedback afterwards saying that it would be good if there could be some additional information describing the data in the chart. Consequently, figure 71 shows the implementation of the progress tracker in the second spiral after the client's feedback. Figure 71 shows that there is the added text below the graph selection buttons telling the user their average percentage over time and provides a custom feedback message based on their performance. This is a good addition because it enhances the information being given to the user and makes use of a previously wasted piece of screen space.

6.3.7 General UI improvements and bug fixes


Quizzes
MATHS
Lorem ipsum dolor sit amet, consectetur.
VERBAL
Lorem ipsum dolor sit amet, consectetur.
NON-VERBAL
Lorem ipsum dolor sit amet, consectetur
MISTAKES
Practice the questions you have been having difficulty with.

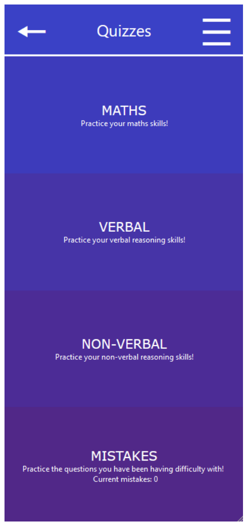

Quizzes
MATHS
Practice your maths skills!
VERBAL
Practice your verbal reasoning skills!
NON-VERBAL
Practice your non-verbal reasoning skills!
MISTAKES
Practice the questions you have been having difficulty with!
Current mistakes: 0

Figure 72 - Old UI example.
Figure 73 - Updated UI example

At the end of the second spiral, there was a decent amount of time left over. This time was spent making minor tweaks to the UI based on feedback from and discussions with the client. Time was also spent fixing bugs that were missed during the previously mentioned sections. For instance, figure 72 shows the old UI whereas figure 73 shows the new UI. Some of the UI differences include making the text weight reduced from being bold and improving explanation tips.

Additionally, some bugs that were fixed include:

- Sometimes questions would return undefined causing a crash - fixed by adding a fallback empty object so it will never be undefined.
- Some components could not update when the colour scheme was changed. This was due to a typo that got missed in the state update. This was easily fixed once identified.
- On certain pages the back button was not working. This is because they were not being checked in the back button function. This has been fixed.

6.4 Evaluation of Implementation

In the opinion of the author, the implementation was conducted successfully. The end application was produced to a high standard with all found bugs being eliminated. Furthermore, all functional requirements were met (see section 7.0) and the client was happy with the app. If the author could change anything about the implementation it would have been to learn React Native before beginning this project. This would have sped up development as the author had no prior experience so some time had to be spent learning it and refactoring old code which needed improving.

7.0 Testing

7.1 Introduction

This section of the report will contain the testing for the first full artefact after the second spiral is completed. The testing section will consist of two parts, the first part contains the functionality testing and the second part contains the usability testing.

7.2 Functional Requirements Testing

7.2.1 Introduction

This section contains the functionality testing. It will test each functional requirement outlined in section 4.0, that has been implemented, and provide details about how the results of the test compared with the expected results.

7.2.2 Functional Testing Approach

The functional requirements testing will be conducted using a unit testing approach. Unit testing is where each individual component of the app is tested. This allows for validation that each component is working as expected. (Patterson, A., Kölling, M., & Rosenberg, J, 2003) The reason for choosing this approach to functional requirements testing is because it fits well with the project. This is because the project has clearly defined and distinct components that can be tested and assessed separately. Furthermore, the functional requirements testing will be conducted manually. Using automated testing was considered however, it was determined that as the project requirements are relatively focused it would use up more development time and could over-complicate when compared to manual testing while not providing significant benefits.

7.2.3 Test Plan

Table 7.1 - Key for functional testing

Key	Definition
Requirement ID	The unique identifier of the requirement. Outlined in section 4.0.
Test ID	A unique identifier to allow for identification and reference of the test.
Test Description	A short description to explain the requirement that is being tested.
Action	A short description explaining the actions required to perform the test.
Expected Result	A short description to explain the outcome of the test we are hoping to see.
Actual Result	A short description to explain the outcome of the test that we did actually see.
Pass/Fail	A pass or fail tag to identify whether the test was successful or not.

7.2.4 Functional Test Results

Table 7.2 - Functional requirements testing.

Requirement ID	Test ID	Test Description	Action	Expected Result	Actual Result	Pass/Fail
F1	T1	The application must allow users to take quizzes on chosen topics.	Press“Quizzes”. Press “Maths Quiz”. Press on all answer buttons A to E to ensure correct functionality.	The user should be able to take a quiz and see instant feedback for their answers.	As expected.	Pass.
F2	T2	The application must allow the users to take tests on chosen topics.	Press “Tests”. Press “Maths Test #1”. Answer the questions.	The feedback should all be given at once at the end of the test.	As expected.	Pass.
F3	T4	The application must allow the user to change the colour scheme to make the application easier to use.	Press the options button in the header. Press on all the colour scheme buttons.	All the colour scheme buttons should update the colour scheme of the app to the corresponding colour.	As expected.	Pass.
F4	T5	The application must give the user the ability to clear the stored data.	Press the options button in the header. Press on the erase data button. Press confirm on the popup.	Popup should appear. Once confirmed the data should be reset to the default settings.	As expected.	Pass.
F5	T6	The application should contain an “about” section that can inform the user on how to use the application.	Press on the about button. Press the arrows on the subheader to view each page.	Information for each page should be shown. The information should also be accurate.	As expected.	Pass.
F6	T6	Tests should provide instant feedback to the user when they are completed.	Complete any test.	A feedback screen should display information about the test.	As expected.	Pass
F7	T7	The application must allow users to view statistics about past performance.	Complete T2 to get dummy data. Press “Statistics”. Press arrows to view each data representation.	Accurate data should be presented. Pressing the arrows should change the data represented.	As expected.	Pass.
F8	T8	The user should be able to have more than one way of	Navigate to the progress tracker	The method of display for the data	As expected.	Pass.

		viewing statistics. For example, the default is a line graph. However, options for bar graphs or pie charts should be available.	and select the bar graph option by pressing on the button.	should change to a bar graph.		
F9	T9	The application should contain an option that allows the user to select from at least 10 different colour schemes to increase accessibility.	Navigate to the settings page by pressing the button in the top right. Click on the colour scheme buttons.	The colour scheme should change corresponding tothe colour of the button that was pressed.	As expected.	Pass.
F10	T10	The app should allow the user to edit the size of the text to increase accessibility.	Navigate to the settings by clicking the button in the top right. Adjust the text scaling slider bar.	The text throughout the app should change in size as the slider is adjusted.	As expected.	Pass.
F11	T11	Attempting to exit a test prematurely should trigger a popup alerting the user to prevent them from ruining their progress.	Start a test, then try and exit the test before the test is complete.	A warning popup should be displayed.	As expected.	Pass.
F12	T12	Test menus should provide the user with their best current score for each test.	Navigate to the test selection menu.	Each test should display a score under the test title on the navigation button.	As expected.	Pass.
F13	T13	The app should contain a splash screen displaying the client's company name.	Load up the app.	The splash screen should appear and transition for 3 seconds.	As expected.	Pass.

7.2.5 Evaluation Of Functional Testing

The testing was very successful as every test passed as expected. This means that no action is required to fix any tests that failed. This is not unexpected because the time management plan and software methodology that was used meant that the features were able to be completed with plenty of time remaining. Of course, there were issues during development that were mentioned in section 6.0. However, at the end of the testing period, all of the outstanding issues had been resolved.

7.3 Usability Testing for Non-functional Requirements

7.3.1 Introduction

Usability testing was performed after the functional requirements testing was completed. The aim of the usability testing was to determine if the non-functional requirements were met. While the functionality testing allowed us to evaluate if the application met all of the functional requirements outlined by the client, the usability testing will allow us to determine whether or not the app meets non-functional requirements and the wider objectives of the projects.

The author has conducted usability testing on the app during development and has made several tweaks to the app that the author thought would improve usability, mentioned in the design section. However, because the author is also the developer of the app there may be a bias that the author has which may prevent the author from getting a fully clear assessment of the usability. Consequently, it is necessary to have the client and people who have not had any significant impact on the project perform some usability testing.

Ten people were able to take part in the usability testing. This was out of a total of 25 people who were invited to take part. While more people taking part would have been better. Due to Covid-19 restrictions, it was harder to find people who were able to take part and who had an interest in the app. However, ten people should be enough to get a good picture of how usable the app is. The people taking part are a mix of people that the author knows would be interested in the app and some that the client was able to find who expressed an interest.

Users were provided with a consent form (Appendix E) which details the conditions and ethics by which the study is being conducted. Users were also provided with an information sheet (Appendix F) which details how the testing will take place. The testing was conducted by giving the user a mobile device that had the app loaded or by allowing them to run the app in their web browser in mobile mode. The user was then asked to write down what they thought was good about the app and also any improvements they thought they could be made based on the guidelines provided in the information sheet. This allowed analysis of any common trends that appear amongst all the participants.

7.3.2 Usability Testing Results

Table 7.3 shows the results of the usability testing. The results are not the exact answers that the individual gave. They are instead a summary of the key points that they mentioned. This should make the results more concise and easy to formulate conclusions from.

Table 7.3 - Usability testing results.

User	Test results
Person 1	● They thought that the app was easy to navigate. ● They liked the ability to change colour schemes. ● Would like expanded functionality on the progress tracker such as being able to see more data at once.
Person 2	● They liked the user interface. ● Felt that the questions covered a wide range of topics on the syllabus. ● Would like to see expanded learning options.
Person 3	● They thought that the app flowed well, it was easy to use navigation. ● They liked the ability to do practice quizzes to prepare for tests. ● Would like to see more variety in test options.
Person 4	● Found the selections of questions to be good. ● They thought that the UI was intuitive to use.
Person 5	● They liked the progress tracker. ● They tried the app on an iPad device, while it was relatively easy to use as the app was scalable. The UI could be better on larger devices.
Person 6	● They liked the range of subjects in the app. ● Found the ability to see past performance to be useful. However, they'd like more options for the progress tracker.
Person 7	● They found the app easy to use and navigate. ● They liked the ability to change colour schemes. ● They would like to see more tools for learning.
Person 8	● They liked the ability to do a quiz on the mistakes they had made. ● Would like to see further test functionality.
Person 9	● They liked the number of questions in the app. ● They liked the user interface. ● Would like to see better support for larger devices.
Person 10	● They liked the ability to adjust the text size. ● They liked the ability to change colour schemes. ● Would like to have more test options.

7.4 Testing Evaluation

The results of both the functionality and usability testing were generally good. Overall, there were no issues with the functional testing. Every functional requirement that was tested was successfully implemented.

The usability testing also provided good results. Generally, most people who took part believed that the app flowed well, had a good array of core features and had a good UI for mobile devices. However, the results showed that there were a few common suggested areas of improvement. These include additional functionality for learning, and expanded progress tracker and better UI support on larger devices such as a tablet. Further information on how the app can be expanded upon in the future based on the results of the usability testing is detailed in section 9.3.

8.0 Evaluation

8.1 Evaluation against requirements

This section will evaluate the functional and non-functional requirements based on whether each one was met or not. This will be achieved by systematically going through each requirement as assessing whether or not it has been met. Each requirement will be given a status of either, fully completed, partially completed or incomplete. After all, requirements have been assessed there can be a wider evaluation of how successful the implementation of the requirements was.

The author believes that this method of assessing the requirements is suitable because each requirement has a relatively small scope and also because each requirement is separate from the other. This makes a manual and linear assessment suitable and effective.

Evaluating requirements is important because it is the primary way that the success or failure of the project can be demonstrated. It is an effective tool for identifying the parts of the project which went well and the parts of the project which require improvements.

8.1.1 Functional Requirements

Table 8.1 - Functional requirements.

Requirement ID	Requirement	Status	Comments
F1	The application must allow users to take quizzes on chosen topics.	Fully completed.	N/A
F2	The application must allow the users to take tests on chosen topics.	Fully completed.	N/A
F3	The application must allow users to view statistics about past performance.	Fully completed.	N/A
F4	The application must allow the user to change the colour scheme to make the application easier to use.	Fully completed.	N/A
F5	The application must give the user the ability to clear the stored data.	Fully completed.	N/A
F6	The application should contain an “about” section that can inform the user on how to use the application.	Fully completed.	N/A

F7	Tests should provide feedback to the user when they are completed.	Fully completed.	N/A
F8	The user should be able to have more than one way of viewing statistics. For example, the default is a line graph. However, options for bar graphs or pie charts should be available.	Fully completed.	N/A
F9	The application should contain an option that allows the user to select from at least 10 different colour schemes to increase accessibility.	Fully completed.	N/A
F10	The app should allow the user to edit the size of the text to increase accessibility.	Fully completed.	N/A
F11	Attempting to exit a test prematurely should trigger a popup alerting the user to prevent them from ruining their progress.	Fully completed.	N/A
F12	Test menus should provide the user with their best current score for each test.	Fully completed.	N/A
F13	The app should contain a splash screen displaying the client's company name.	Fully completed.	N/A

8.1.2 Non-functional Requirements

Table 8.2 - Non-functional requirements.

Requirement ID	Requirement	Status	Comments
NF1	The app needs to be scalable. If new questions need to be added they should be able to cope.	Fully completed.	JSON question files can support as many questions as needed.
NF2	The app needs to be completed by the 21st of May 2021.	Fully completed.	N/A
NF3	The app needs to be able to run while offline.	Fully completed.	App tested without internet connection.
NF4	The app needs to be able to run on iOS, Android and on an expo web test server for demonstration.	Fully completed.	App tested on all three platforms.
NF5	The app should have accessibility considerations for visual disabilities.	Fully completed.	Usability testing showed that people found the colour scheme and text settings to be useful.
NF6	The app should be created in a way that maximises maintainability.	Fully completed.	Division of the code into components increases maintainability.
NF6	The app needs to be able to work effectively on larger devices such as tablets.	Partially completed.	While the app works on larger devices, usability testing showed that further UI work for larger devices is needed.

8.1.3 Conclusion

The functional requirements were assessed in table 8.1. The table shows that each functional requirement for this project was completed successfully with no omissions or partially completed functional requirements. This means that the implementation of the functional requirements for this project can be regarded as a full success.

The non-functional requirements were assessed in table 8.2. The table shows that almost all of the non-functional requirements were met. The exception is NF6 which was only able to be partially completed. This means that overall, the implementation of the non-functional requirements can be considered successful.

The reason that NF6 is partially completed is due to limitations during development. The need for improvements to the UI on larger devices was only identified during the usability testing. This was relatively late in development meaning that it was not feasible to fully redesign the UI of the app for larger devices in a reasonable time frame. However, the app was built from the ground up to have a scalable UI so while there is not a bespoke UI for larger devices the existing UI is functional and works reasonably well. This is why NF6 is considered partially complete rather than uncompleted. Further work will be needed outside of the scope of this project to ensure NF6 is completed.

8.2 Evaluation against project aims

This section will evaluate the project aims to determine if they have been achieved and if so to what extent.

“High-level project aim: To create an educational mobile app, to aid individual learning for the Kent Test.”

The author believes that the high-level project aim outlined in section 1.2 (shown above) has been met. In the view of the author, the final app clearly demonstrates functionality and usefulness in aiding individual learning for the Kent Test. The difficulty is not so much in assessing whether or not the high-level project aim has been met but by what degree it has been met. The author believes that the high-level project aim has been met to a high standard as the usability testing demonstrated people found the app useful. Furthermore, the app is essentially free of defects - further emphasising the high level of quality attained. However, at the present time, the author believes there are limitations on the overall usefulness of this assessment. This is due to the subjective nature of assessing the degree to which the high-level project aim has been met. To address this there needs to be a larger user study to fully ascertain how useful users find the app at addressing the core aim of aiding individual learning for the Kent Test.

Table 8.3 - Project milestone assessment

Project milestone	Status
The app should be able to run on Android and iOS devices.	Achieved.
The app should have accessibility considerations.	Achieved.
The user should be able to take an assessed test in a chosen subject.	Achieved.
The user should be able to take a non-assessed quiz in a chosen subject.	Achieved.
There should be a feature that allows users to view their progress.	Achieved.
The app should be easy and intuitive to use and there should be functionality to teach users about how to use the app.	Achieved.
The app should be easy and intuitive to use and there should be functionality to teach users about how to use the app.	Achieved.
The user should be fully informed of what data the app is using and should be able to erase any stored data at any time if they wish.	Achieved.

Table 8.3 shows the initial project milestones which were outlined in section 1.2 and state whether or not each one has been achieved or not. It was stated that these milestones had to be met if the project was to be considered a success. Each milestone was successfully achieved through both functional and usability testing, meaning that, in the view of the author, the project has been successful.

The author believes that the project milestones success is easier to ascertain than the high-level project aim due to it being a specific set of objectives rather than a high-level statement. This means that the success criteria is less subjective.

9.0 Conclusion

9.1 Project Reflection

The author believes that all the aims and objectives for this project have been completed to the standard that was requested by the client. There was no feature that was initially requested by the client which was not able to be implemented.

Non-technical constraints which were initially outlined in section 1.0 (table 1.2) such as time constraints ended up not being a problem during this project. For example, the author was, mostly, able to stick to the time management plan outlined in the Gantt chart (section 3.7.3 figure 9). This was in part due to no unforeseen events happening during the project such as technical failure and also due to the usefulness of the planning that was done at the start of the project. Allowing the author to effectively manage their time.

Other sections of the project also went well. The requirements gathering was effective at ascertaining the functional and non-functional requirements for the project. However, the questionnaire did take longer than expected to return enough results to be useful, but it did not significantly impact project progress. The meetings that the author had with their project supervisor also greatly helped the progress of this project, the feedback provided on the author's progress was constructive and insightful.

If the project was to be done again, the author believes that a few changes could be made to further the success of the project. Firstly, a greater focus on project planning during the PID phase would have been beneficial. While there was basic time planning, it was later revised to a more complete version. Having this complete plan from the beginning would have enabled better time management during the early phase of the project.

9.2 Knowledge Gained

The completion of this project has allowed the author to develop their knowledge and skills in all areas which were required by this project. Most significantly, the author's understanding of programming and React/React Native has been significantly boosted as prior to this project the author had never created a React Native application.

Additional areas that the author gained knowledge in:

- **Mobile app development** - This project was the first time the author had created a mobile app. The author learned a great deal about the challenges and limitations of developing a mobile app. For example, prior to this project, the author was not aware of how important it was to get the right method of data storage for a mobile app.
- **UI design** - Prior to this project the author had knowledge of UI design principles as the author had done a lot of web design work. However, working on a mobile app offers a different challenge due to the limited screen space. The author's knowledge of creating a UI that is intuitive and presents as much information to the user as possible in as little screen space as possible has been increased.
- **Software development and research elicitation** - This was the first project that the author had done which involved sticking to a software methodology and actually having to gather requirements from a client. It was a good experience and made me much more aware of how important choosing the right methodology and requirements elicitation methods is to the success of a project.
- **Time and project management** - In past projects, the author would usually just go ahead and start programming without much consideration to time and project management. This project, however, was the first time that the author employed proper management techniques such as a Gantt chart. It made the project much smoother and allowed the author to complete the implementation with lots of time to spare.

9.3 Project Future

9.3.1 Summary

While the project aims were completed successfully, there are areas of the project which could be improved and expanded upon in the future. These areas were identified in the usability testing (7.3.2) and also with conversations with the client. Some areas of expansion were also covered in the research section of the paper (2.0), this section will provide further detail on these areas.

9.3.2 Online Capabilities

Adding online capabilities such as accounts, friends, group content for classrooms would be a good way of expanding the app in the future. While the initial scope of the app was for a self-study tool, the client has expressed interest in creating the aforementioned online features as it would bring the app to a wider group of people.

9.3.3 Additional Learning Tools

Additional learning tools such as flashcards were initially identified as a potential route for expansion in 2.0. However, the usability testing showed that users would like to see features such as this implemented. It would be a good future route for the app as it would increase the effectiveness of the app for self-learning which was the original goal for this project.

9.3.4 Additional Questions

Expanding the number of questions available to the user would be a good way to expand the existing functionality of the app. This would require the client to send more questions however which they currently do not have.

9.3.5 Additional Topics

Currently, there are only three topics in the app. These are Maths, Verbal and Non-Verbal Reasoning. A potential route for expansion could be to increase the number of topics. This would not be hard to do as the existing functionality could just be used for the new topics. This would take time however because currently, the client has not got questions for additional topics so they would have to be written first.

9.3.6 Custom Content

Allowing the user to create their own content is another potential route of expansion for the app. This could take the form of allowing users to create their own tests and quizzes using their own or preexisting questions. This could add a lot to the effectiveness of the app as a learning tool.

References

(2014). Dyslexia and Literacy International | Better training, better teaching. https://www.dyslexia-and-literacy.international/wp-content/uploads/2016/04/DI-Duke-Report-final-4-29-14.pdf

Abbasi, M. A., Jabeen, J., Hafeez, Y., Dur-e-Beenish Batool, & Fareen, N. (2015). Assessment of Requirement Elicitation Tools and Techniques by Various Parameters. https://doi.org/10.11648/j.se.20150302.11

Agile. (2001). Manifesto for Agile Software Development. https://agilemanifesto.org/

Alshamrani, A., & Bahattab, A. (2015). A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model. https://sijal-technology.com/images/company/86b122d4358357d834a87ce618a55de0.pdf

Animated · React native. (n.d.). React Native · Learn once, write anywhere. https://reactnative.dev/docs/animated

Aria. (2020, 2). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA

Bosnic, S., Papp, I., & Novak, S. (2016). The development of hybrid mobile applications with Apache Cordova. 2016 24th Telecommunications Forum (TELFOR). https://doi.org/10.1109/telfor.2016.7818919

British Dyslexia Association. (n.d.). Dyslexia friendly style guide. https://www.bdadyslexia.org.uk/advice/employers/creating-a-dyslexia-friendly-workplace/dyslexia-friendly-style-guide

Butler, M. (2011). Android: Changing the mobile landscape. IEEE Pervasive Computing, 10(1), 4-7. https://doi.org/10.1109/mprv.2011.1

Casteren, W. (2017) The Waterfall Model and the Agile Methodologies: A comparison by project characteristics. https://bit.ly/3flE9Bq

Do, T., & Gatica-Perez, D. (2010). By their apps you shall understand them. Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia - MUM '10. https://doi.org/10.1145/1899475.1899502

Documentation for visual studio code. (2016, April 14). Visual Studio Code - Code Editing. Redefined. https://code.visualstudio.com/docs

Dora, S., & Dubey, P. (2013). SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)ANALYTICAL COMPARISON AND SURVEY ON TRADITIONAL AND AGILE METHODOLOGY, 2(8). https://bit.ly/3fdE0OQ

Download Android Studio and SDK tools | Android developers. (2021). Android Developers. https://developer.android.com/studio

Duolingo, (2021). https://www.duolingo.com/

Flutter architectural overview. (2021). Flutter - Beautiful native apps in record time. https://flutter.dev/docs/resources/architectural-overview

Fundamentals of operating systems. (2016). Macmillan International Higher Education.

Gackenheimer, C. (2015). Introduction to react. Apress.

Gandhewar et al, (2010), Google Android: An Emerging Software Platform for Mobile Devices, International Journal on Computer Science and Engineering

GitHub documentation, 2021, https://docs.github.com/en

Gronli, T., Hansen, J., Ghinea, G., & Younas, M. (2014). Mobile application platform heterogeneity: Android vs Windows Phone vs Ios vs Firefox OS. 2014 IEEE 28th International Conference on Advanced Information Networking and Applications. https://doi.org/10.1109/aina.2014.78

Hansson & Vidhall, (2016), Effects on performance and usability for cross-platform application development using React Native, Linköping University, http://liu.diva-portal.org/smash/get/diva2:946127/FULLTEXT01.pdf

Hill, H. (2013). Disability and accessibility in the library and information science literature: A content analysis. Library & Information Science Research, 35(2), 137-142. https://doi.org/10.1016/j.lisr.2012.11.002

Ibrahim, N., Ibrahim, R., Saringat, M. Z., Mansor, D., & Herawan, T. (2011). Definition of consistency rules between UML use case and activity diagram. Communications in Computer and Information Science, 498-508. https://doi.org/10.1007/978-3-642-20998-7_58

Identification of Stakeholders. (2008). Guidelines and Toolkits for Urban Transport Development in Medium Sized Cities in India. https://sti-india-uttoolkit.adb.org/mod3/se2/003_2.html

Kavcic, A. (2005). Software accessibility: Recommendations and guidelines. EUROCON 2005 - The International Conference on "Computer as a Tool". https://doi.org/10.1109/eurcon.2005.1630123

Kurtanovic, Z., & Maalej, W. (2017). Automatically classifying functional and non-functional requirements using supervised machine learning. 2017 IEEE 25th International Requirements Engineering Conference (RE). https://doi.org/10.1109/re.2017.82

Limitations. (2021). Expo Documentation. https://docs.expo.io/introduction/why-not-expo/

Miyasaka, J. D. Vieira, R. V. Novalo-Goto, E. S., Montagna, E., & Wajnsztejn, R. (2019). Irlen syndrome: Systematic review and level of evidence analysis. Arquivos de Neuro-Psiquiatria, 77(3), 194-207. https://doi.org/10.1590/0004-282x20190014

Mobile OS market share 2019. (2021). Statista. https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/

Mohamed, I., & Patel, D. (2015). Android vs Ios security: A comparative study. 2015 12th International Conference on Information Technology - New Generations. https://doi.org/10.1109/itng.2015.123

MySQL :: MySQL community edition. (n.d.). MySQL. https://www.mysql.com/products/community/

Napoli, M. L. (2019). Beginning flutter: A hands-on guide to app development. John Wiley & Sons.

Novick, V. (2017). React native - Building mobile apps with JavaScript. Packt Publishing.

Patterson, A., Kölling, M., & Rosenberg, J. (2003). Introducing unit testing with BlueJ. ACM SIGCSE Bulletin, 35(3), 11-15. https://doi.org/10.1145/961290.961518

Poleon, S., & Szaflarski, J. P. (2017). Photosensitivity in generalized epilepsies. Epilepsy & Behavior, 68, 225-233. https://doi.org/10.1016/j.yebeh.2016.10.040

Proceedings of the 43rd ACM technical symposium on Computer Science Education - SIGCSE '12. https://doi.org/10.1145/2157136.2157330

Quizlet. (2021). Quizlet. https://quizlet.com/en-gb

Rogers, M., & Goadrich, M. (2012). A hands-on comparison of Ios vs. Android (abstract only).

Saeed, S., Jhanjhi, N. Z., Naqvi, M., & Humayun, M. (2019). Analysis of Software Development Methodologies. http://dx.doi.org/10.12785/ijcds/080502

Sheikh, A. A., Ganai, P. T., Malik, N. A., & Dar, K. A. (2013). Smartphone: Android vs IOS. The SIJ Transactions on Computer Science Engineering & its Applications (CSEA), 01(04), 31-38. https://doi.org/10.9756/sijcsea/v1i4/0104600401

Simpson, J. A. (2008). The Oxford English Dictionary: Vol. 1-.

Teacher, J. (2020, October 24). Anatomy of a react application — Architect with sagas. Medium. https://jstutorial.medium.com/anatomy-of-a-react-application-d465739ec6be

The thumb zone: Designing for mobile users — Smashing magazine. (2016, September 19). Smashing Magazine. https://www.smashingmagazine.com/2016/09/the-thumb-zone-designing-for-mobile-users/

Top 5 software development methodologies. (2020, November 20). DreamSoft4U Pvt. Ltd. https://dreamsoft4u.com/blog/top-5-software-development-methodologies

Agile methodology tutorial. (n.d.). Tutorialspoint https://www.tutorialspoint.com/agile/index.htm

UML use case diagram tutorial. (n.d.). Lucidchart.
https://www.lucidchart.com/pages/uml-use-case-diagram

Usage. (2021). Site not found · GitHub Pages.
https://react-native-async-storage.github.io/async-storage/docs/usage/

Westland, S., Laycock, K., Cheung, V., Henry, P., & Mahyar, F. (2007). Colour Harmony.
http://www.colour-journal.org/2007/1/1/

Xcode. (2021). Apple Developer. https://developer.apple.com/xcode/

Zachariah, B., & Nonyelum, O. (2020). A Comparative Analysis of Requirement Gathering Techniques.
https://search.proquest.com/openview/9a021d6caa5c653bfcb5f034c2657d49/1?pq-origsite=gscholar&cbl=2029993

Appendix

Appendix A - Project Initiation Document

UNIVERSITY OF PORTSMOUTH

School of Computing
Project Initiation Document

Matthew Shore

Educational App

PJE40

Basic details:

Student name:	Matthew Shore
Draft project title:	Educational Quiz Application
Course:	Computer Science
Project supervisor:	Haythem Nakkas
Client organisation:	Set4SuccessTuition
Client contact name:	Claire Shore

Degree Suitability:

The project involved programming a mobile app which can be run on both android and iOS. To do this I will be using React Native. This is a JavaScript framework which has been developed by Facebook. The project will build upon the knowledge I gained during my industrial placement year which was considered suitable for the degree.

Outline of the project environment and problem to be solved:

Who is the client?	Claire Shore (mother) is the client.
What do they do?	Runs an education/tuition company.
What is their problem?	They have thousands of questions they would like to put into an app so that their customers can do the questions on mobile.
Why does it need to be solved?	To make access to the questions much easier for the customers.

Project aims and objectives:

The aim of the project is to create an application which provides the following functionality to the user:

- Provide the ability for the user to do quizzes on Maths, Non-verbal and Verbal.
- Provide the ability for the user to do tests to test their knowledge.
- Provide statistics for the user to see how well they have done.

I will create the application components in the following order:

1. Create the navigation/menu options.
2. Create the ability to load questions.
3. Create the functionality for the quiz part of the app.
4. Building upon this I can then create the functionality for tests.
5. Once I have the test functionality, I can create statistics for the user's previously done tests.

Project deliverables

- Project report.
- Feedback from the client and potential users covering what they would like to see from the application. Additionally, feedback on different stages of the application's development.
- App will be delivered at stages of development for feedback.

Project constraints

I cannot think of any major technical constraints. The app will have to be able to run offline so that could potentially limit the database functionality (it would have to work offline).

Time is a non-technical constraint on the scope of the project. I must fit the project around my other university modules and personal life. However, I believe that I have designed the scope of the project to ensure that this should not be a major issue.

Additionally, the client is a potential constraint as if they are busy or do not respond in a reasonable time for whatever reason then it could slow development of the app. However, I think the risk of this is small.

Project approach

Background research: The background research will be done by investigating other apps currently available to see if there is anything I am missing or could improve on.

Requirements: To gather the requirements for the project I will be talking to the client and discussing it with them. Additionally, I might also talk to potential users about what they would like to see from the app at some point.

Development: Regular meetings with client to discuss if the app development is meeting their requirements. Testing after each component of the app is developed.

Literature review plan

The starting point for my research will be the iOS and Android app stores. I will be searching here for apps which perform a similar role. I am unlikely to come across any app which is going to be used in the same context as mine – due to the niche series of Exams I am making my app for.

The main components off these similar apps in the market that I will be critically reviewing is firstly, the user interface. The user interface is something that is hard to get right and reviewing how similar apps have designed theirs could highlight certain common design principles which could aid my own UI design.

Secondly, I will be reviewing their functionality. It will be important to review functionality to see what functionality other similar apps are using. This could potentially show me some gaps in my own functionality or give me some ideas to improve my existing functionality.

Other points of research for my literature review will be papers on UI and App design, it will be useful to critically assess these papers and perhaps gain a better understanding of it myself.

Facilities and resources

The only two resources I need is a PC and an iPhone. I require the pc to program and test the application and the iPhone to test the application on iOS. I have emulators on the PC which allow me to test the application on Android. There is not really a constraint to my access to this as I have a PC at my house, if it breaks, I can go to a back-up PC.

For software I am using Visual Studio Code and Expo both of which are free and easily accessible. I may have to use a database later but that has not been fully determined yet. Finally, for version control I am using Git.

Log of Risks

Description	Impact	Likelihood	Mitigation	First Indicator
PC failure	Losing work or not be able to continue for some time.	High impact, low chance.	Store all files on GitHub. Maintain pc and have a backup computer ready.	Potential computer crashes or hardware issues.
Running out of time	Not having done the report or full app functionality.	High impact, low chance	I am making good progress and through good time management I think I am on track.	Missing a deadline or having to rush to get a requirement done.
Increase in requirements	Client increases or adds a new requirement	Low impact, low chance	Unlikely as I have already spoken to the client about this.	They will tell me.
Covid-19	Having to isolate away from pc or being in hospital preventing me doing any work.	High impact, low chance.	It is unlikely as I am socially distancing and not interacting with too many people.	Symptoms of Covid-19

Project Plan

01/11/20 -> 15/11/20 (preparation)

- I am talking to the client about requirements and am researching other similar applications that exist.
- Creating the navigation UI is the first functional step. I have made some mock designs before implementing it into the application.
- During this part of the project I will start my report and write up the introduction I also will create a literature review to go in my report based on the research that I have done.

15/11/20 -> 01/01/21 (functionality + testing)

- Once the navigation is complete, I can work through the app feature by feature. In the order quizzes -> tests -> statistics.
- When each component of the application is completed, I will show it to the client in order to receive feedback from them.
- During this part of the project I will write up my methodology and describe the design process thus far in the report. I will show how each component of the app is working and my rational behind them.

01/01/21 -> 07/01/21 (review first iteration)

- Review of the app with the client to ensure it meets the project requirements.
- Potential discussions with potential end users to receive feedback.
- During this part of the project I will add the client feedback to my report and will update my methodology and design accordingly.

07/01/21 -> 07/02/21 (functionality + testing)

- Based on feedback will develop further functionality and changes. I will update the report to reflect this.

07/02/21 -> 14/02/21 (review second iteration)

- Second review of app with client and potentially with users.
- In the report I will start to write up how closely my app meets the requirements as I will have a clear picture of this now.

07/02/21 -> 12/05/21 (Functionality + Evaluation)

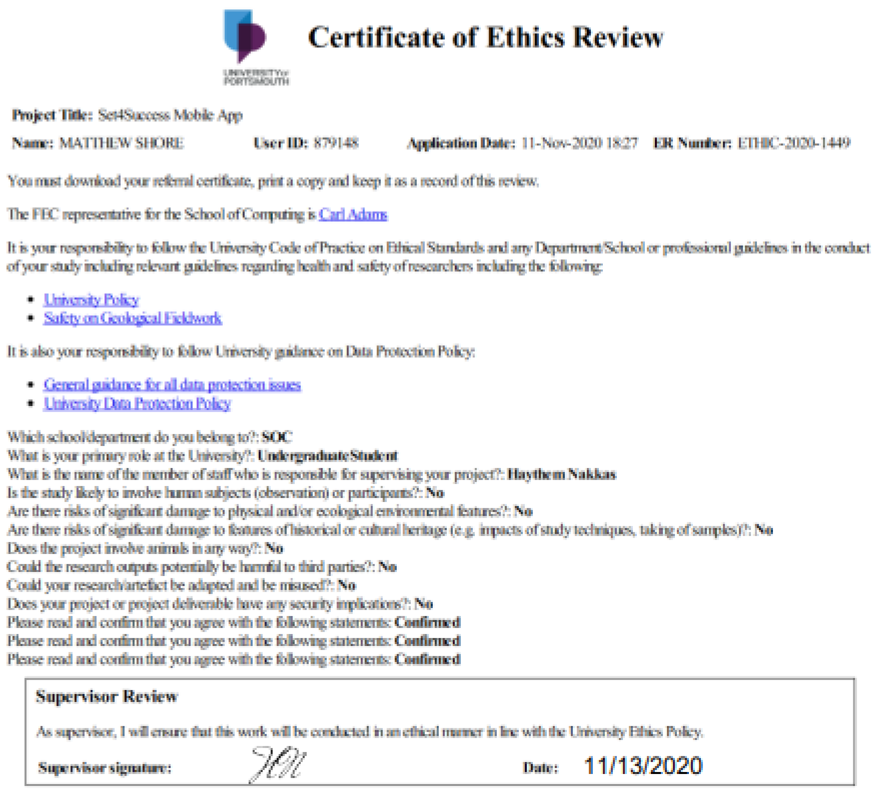
- Final iteration of functionality based on the second review from the client.
- Evaluation of project and report to be completed alongside the different stages. I will discuss if there was anything I could have done differently or better or included.
- Conclusion in the report as to whether the app met the original design specifications.

- Additionally, I will be updating/checking all the report to ensure that it meets the project requirements. I will be critically reviewing the planning and management that has gone on throughout the project.

Legal, ethical, professional, social issues

I do not think these apply as the application is a quiz-based application. We will not be storing any information about the user and the questions and statistics do not contain anything which would present a legal, ethical, professional or social issue. The user will have the option to disable these options or clear their history whenever possible.

Appendix B - Ethics form



Certificate of Ethics Review

Project Title: Set4Success Mobile App

Name: MATTHEW SHORE **User ID:** 879148 **Application Date:** 11-Nov-2020 18:27 **ER Number:** ETHIC-2020-1449

You must download your referral certificate, print a copy and keep it as a record of this review.

The FEC representative for the School of Computing is Carl Adams

It is your responsibility to follow the University Code of Practice on Ethical Standards and any Department/School or professional guidelines in the conduct of your study including relevant guidelines regarding health and safety of researchers including the following:

- University Policy
- Safety on Geological Fieldwork

It is also your responsibility to follow University guidance on Data Protection Policy:

- General guidance for all data protection issues
- University Data Protection Policy

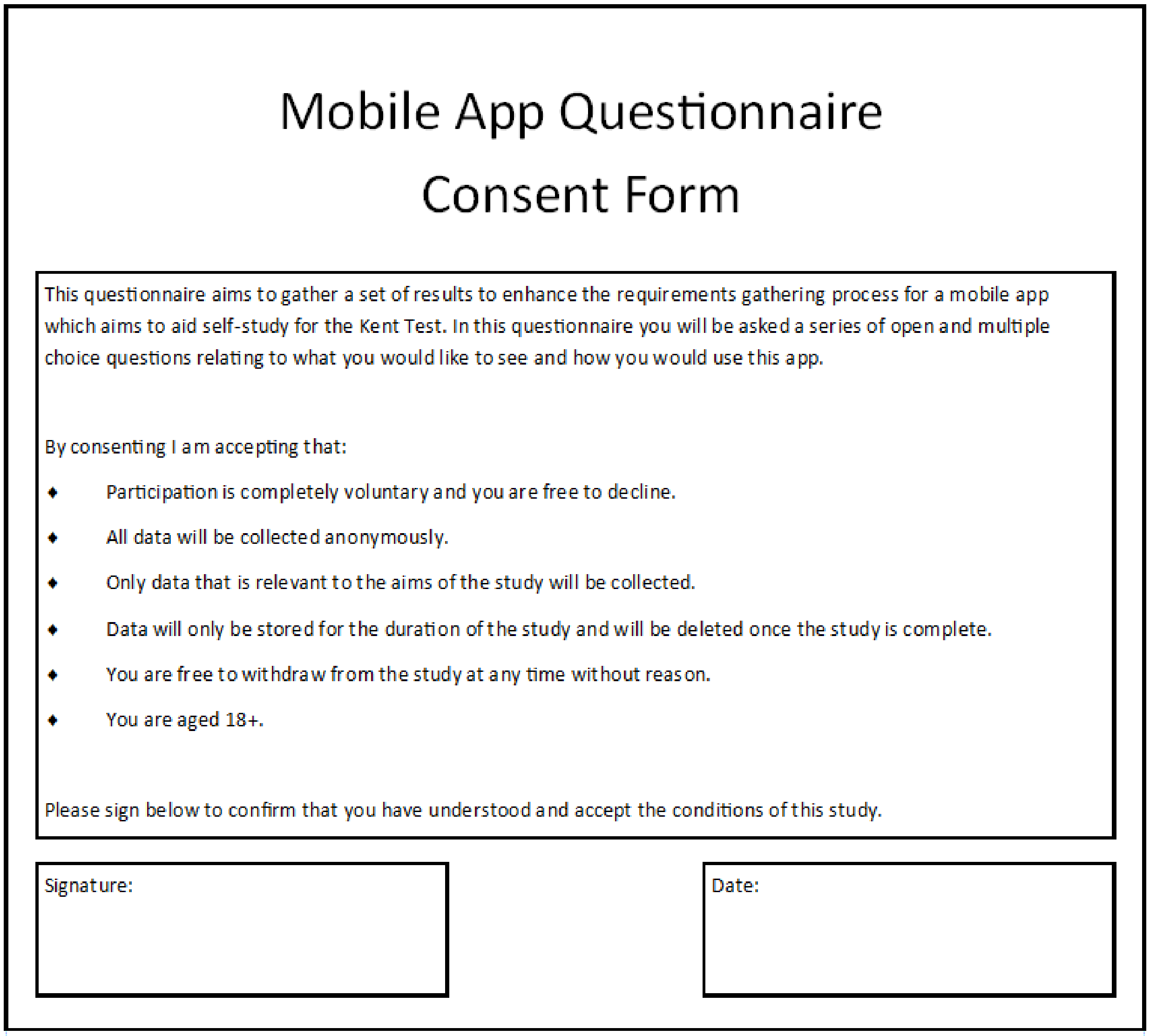
Which school/department do you belong to?: **SOC**
What is your primary role at the University?: **Undergraduate Student**
What is the name of the member of staff who is responsible for supervising your project?: **Haythem Nakkas**
Is the study likely to involve human subjects (observation) or participants?: **No**
Are there risks of significant damage to physical and/or ecological environmental features?: **No**
Are there risks of significant damage to features of historical or cultural heritage (e.g. impacts of study techniques, taking of samples)?: **No**
Does the project involve animals in any way?: **No**
Could the research outputs potentially be harmful to third parties?: **No**
Could your research/artefact be adapted and be misused?: **No**
Does your project or project deliverable have any security implications?: **No**
Please read and confirm that you agree with the following statements: **Confirmed**
Please read and confirm that you agree with the following statements: **Confirmed**
Please read and confirm that you agree with the following statements: **Confirmed**

Supervisor Review

As supervisor, I will ensure that this work will be conducted in an ethical manner in line with the University Ethics Policy.

Supervisor signature: [illegible] **Date:** 11/13/2020

Appendix C - Questionnaire consent form



Mobile App Questionnaire

Consent Form

This questionnaire aims to gather a set of results to enhance the requirements gathering process for a mobile app which aims to aid self-study for the Kent Test. In this questionnaire you will be asked a series of open and multiple choice questions relating to what you would like to see and how you would use this app.

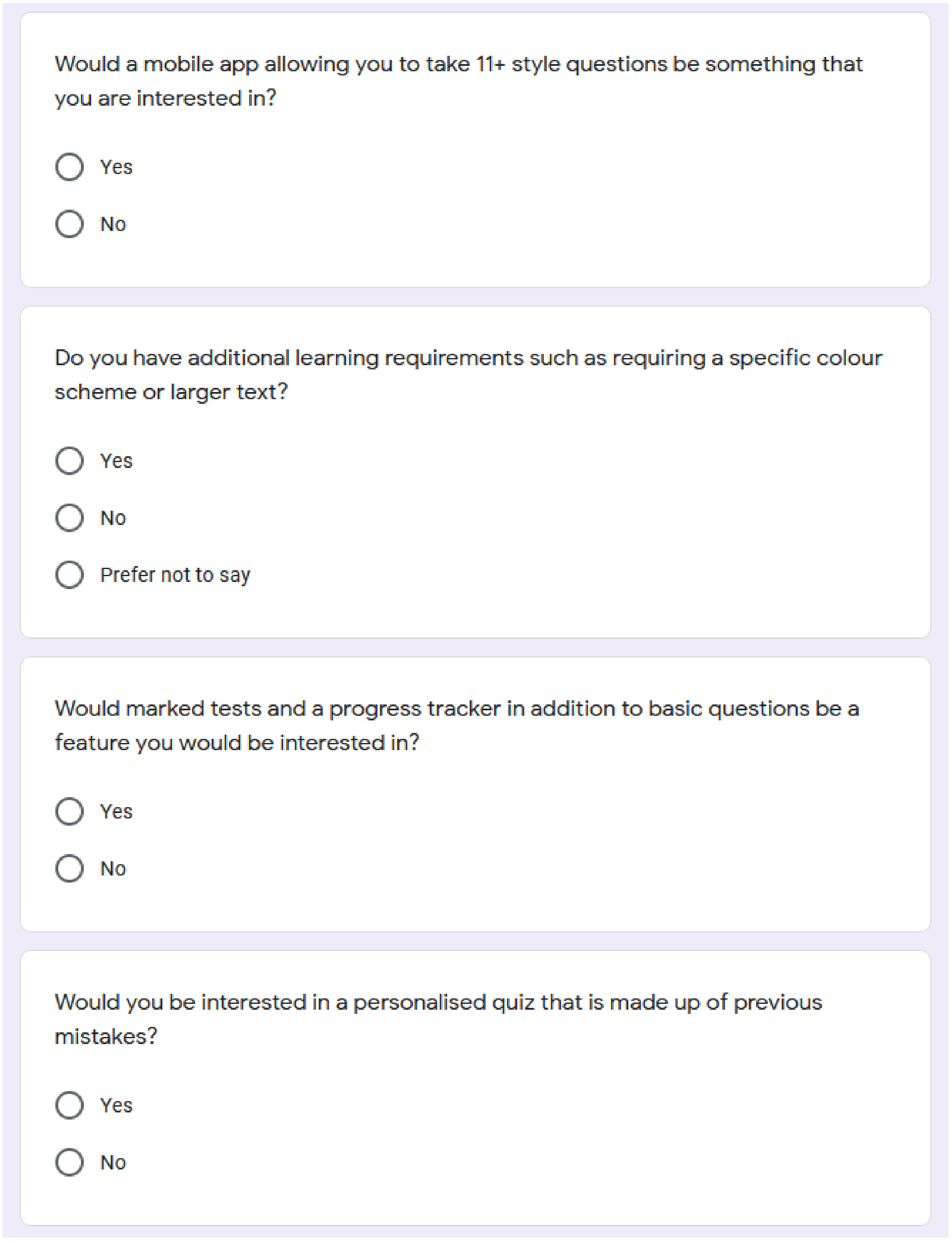
By consenting I am accepting that:

- Participation is completely voluntary and you are free to decline.
- All data will be collected anonymously.
- Only data that is relevant to the aims of the study will be collected.
- Data will only be stored for the duration of the study and will be deleted once the study is complete.
- You are free to withdraw from the study at any time without reason.
- You are aged 18+.

Please sign below to confirm that you have understood and accept the conditions of this study.

Signature:	Date:

Appendix D - Questionnaire questions



Would a mobile app allowing you to take 11+ style questions be something that you are interested in?

○ Yes

○ No

Do you have additional learning requirements such as requiring a specific colour scheme or larger text?

○ Yes

○ No

○ Prefer not to say

Would marked tests and a progress tracker in addition to basic questions be a feature you would be interested in?

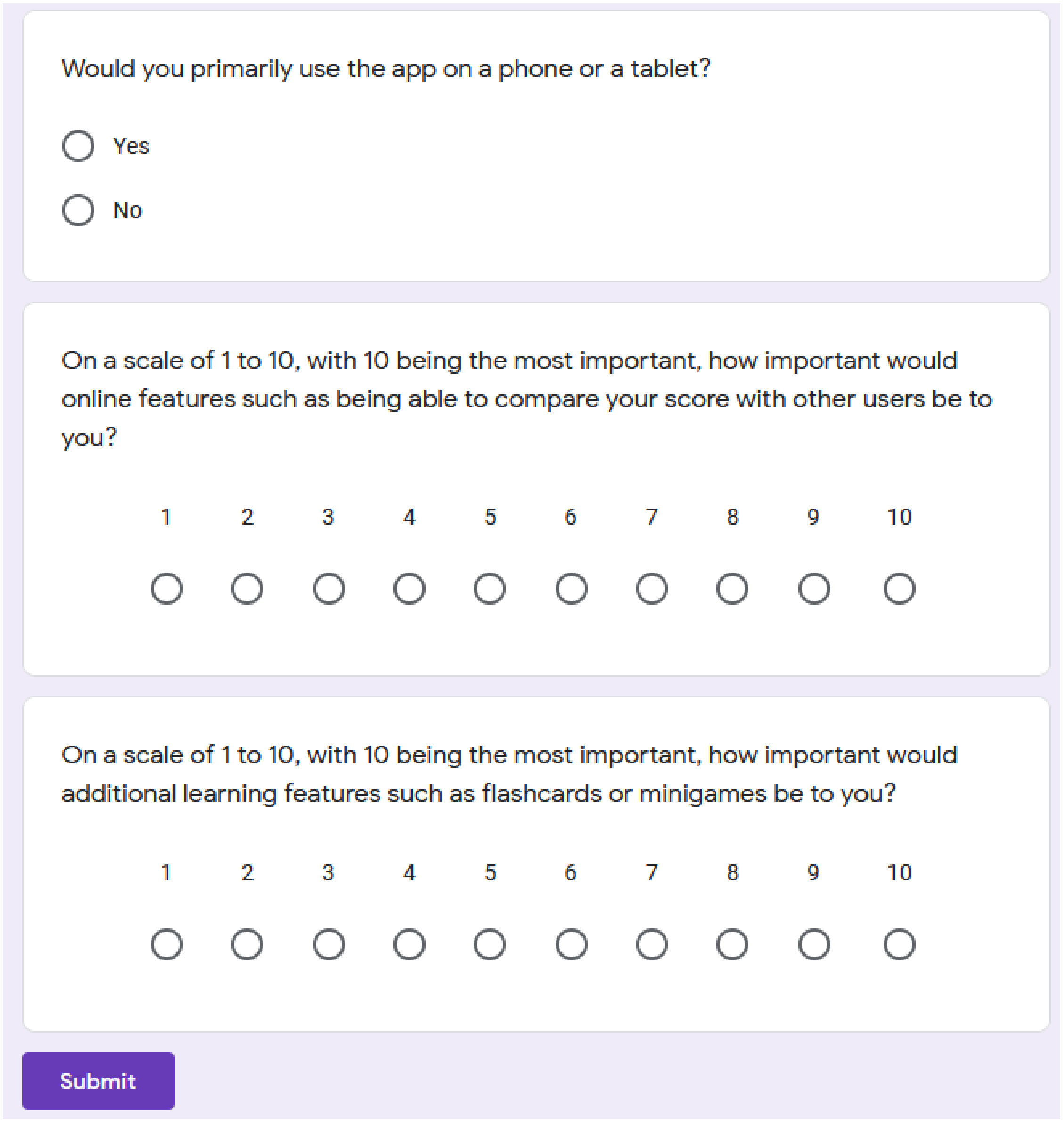
○ Yes

○ No

Would you be interested in a personalised quiz that is made up of previous mistakes?

○ Yes

○ No



Would you primarily use the app on a phone or a tablet?

○ Yes

○ No

On a scale of 1 to 10, with 10 being the most important, how important would online features such as being able to compare your score with other users be to you?

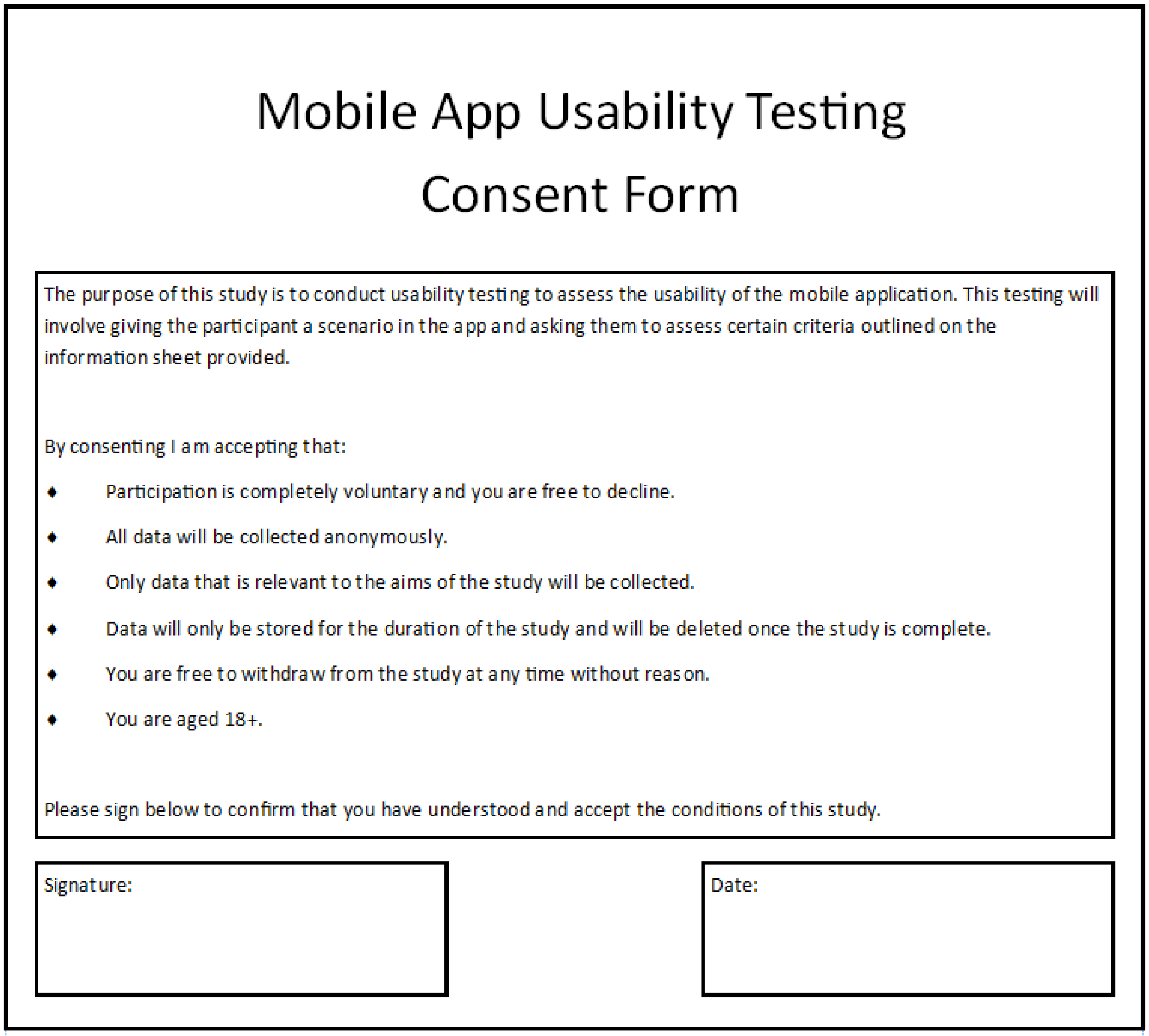
1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

On a scale of 1 to 10, with 10 being the most important, how important would additional learning features such as flashcards or minigames be to you?

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

Submit

Appendix E - Usability testing consent form



Mobile App Usability Testing

Consent Form

The purpose of this study is to conduct usability testing to assess the usability of the mobile application. This testing will involve giving the participant a scenario in the app and asking them to assess certain criteria outlined on the information sheet provided.

By consenting I am accepting that:

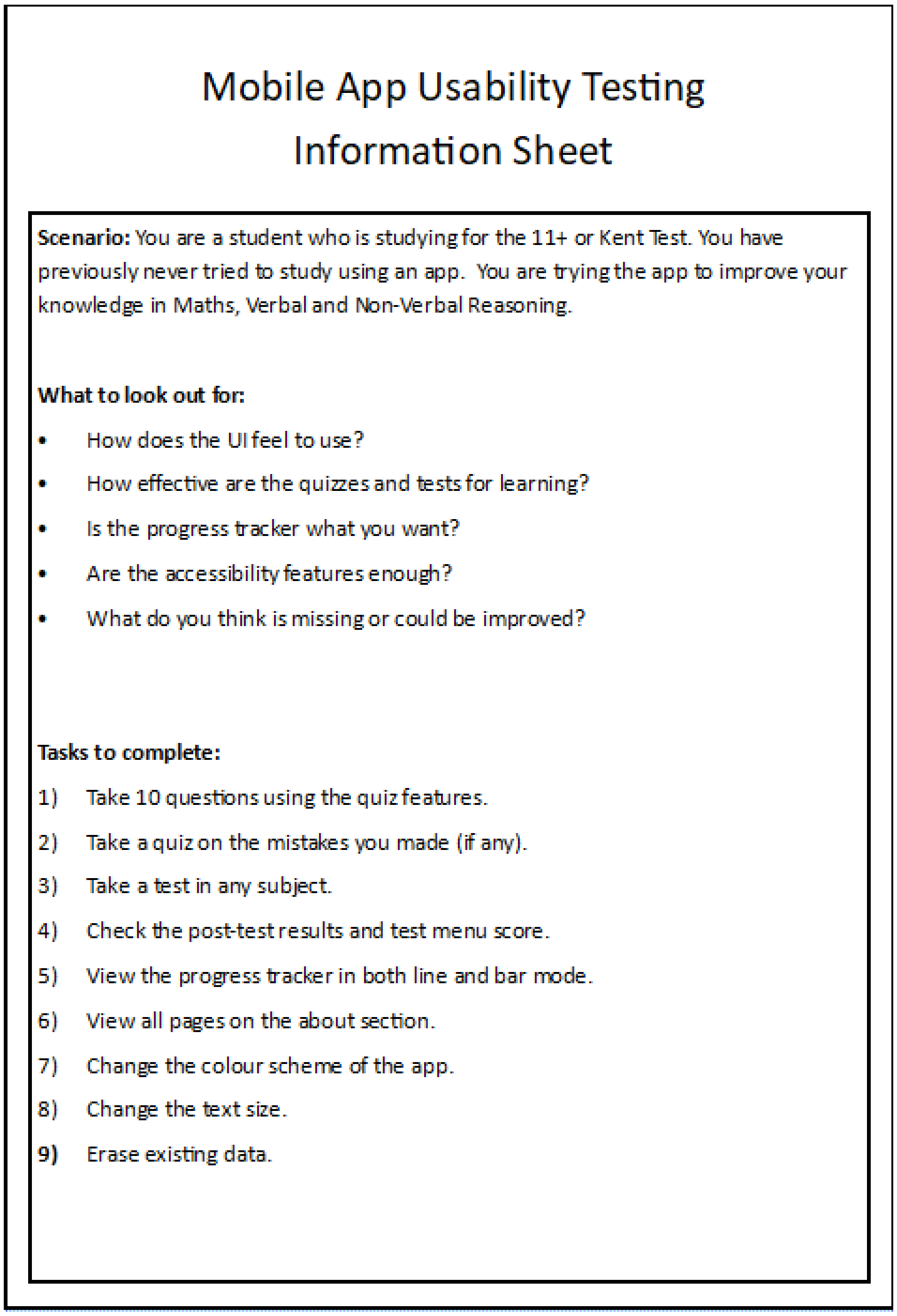
- Participation is completely voluntary and you are free to decline.
- All data will be collected anonymously.
- Only data that is relevant to the aims of the study will be collected.
- Data will only be stored for the duration of the study and will be deleted once the study is complete.
- You are free to withdraw from the study at any time without reason.
- You are aged 18+.

Please sign below to confirm that you have understood and accept the conditions of this study.

Signature:

Date:

Appendix F - Usability testing information sheet



Mobile App Usability Testing Information Sheet

Scenario: You are a student who is studying for the 11+ or Kent Test. You have previously never tried to study using an app. You are trying the app to improve your knowledge in Maths, Verbal and Non-Verbal Reasoning.

What to look out for:

- How does the UI feel to use?
- How effective are the quizzes and tests for learning?
- Is the progress tracker what you want?
- Are the accessibility features enough?
- What do you think is missing or could be improved?

Tasks to complete:

1) Take 10 questions using the quiz features.
2) Take a quiz on the mistakes you made (if any).
3) Take a test in any subject.
4) Check the post-test results and test menu score.
5) View the progress tracker in both line and bar mode.
6) View all pages on the about section.
7) Change the colour scheme of the app.
8) Change the text size.
9) Erase existing data.

Appendix G - Source Code

All the source code and relevant material is located in this public github repository:
https://github.com/MatthewShoreWeb/FinalYearProject