

CTEC3451_1920_520

P-Number: P1750791

Development Project

Abstract

This report addresses the procedures taken in order to make the project successful. It serves as an overview of the project and includes all the steps taken in order to ensure its success. Starting from the bottom up in a very clear and coherent manner with critical evaluation of the steps taken. A brief introduction of the project including the reasonings behind each phase including methodologies used, design patterns, programming concepts, database management, application analysis, test plans and critical analysis.

Table of Contents

Abstract.....	1
Section 1: Introduction.....	3
Background	3
Aims and Objectives.....	3
Section 2: Software Development Methodology & why is it important?.....	4
Methodology Chosen	4
Waterfall Model	4
Why the waterfall model?.....	6
Why the model suited this project.....	7
Section 3: Diagram Designs	8
Use Case Diagram:	8
What it is?	8
My project.....	8
UML Diagram:.....	10
What is it?	10
Section 4: Integrated Development Environment	11
Language	12
Swift	12
Organizing Code.....	12
Model	13

View	14
Controller	14
Section 5: Firebase.....	15
Benefits of Firebase	15
Firebase Authentication	15
Real-Time Database	17
Section 6: User Interface	18
Section 7: Test cases	23
Section 8: Critical Review.....	25
Software tools	25
Improvements	25
Issues along the way	26
Bibliography.....	27

Section 1: Introduction

Background

The Idea of the project initially was to create an educational game quiz-like application for smartphone/device users. I decided to create an application for mobile users simply because of the sheer volume of people who own a device in this day and age. A multiple-choice styled quiz game app with a touch of education stood out for me and after extensive research, I had proceeded to go ahead with this idea. There were many factors to take into account such as compatibility. Will the application work on android smartphones, IOS or both? I had developed the project strictly on IOS due to the fact that it was convenient for me having owned a MacBook and an iPhone, this was a good start for me into the world. During the researching phase of this project, the literature review documentation that was made highlighted studies showing 86% of people aged 12-18 regularly use a mobile device within the UK (Bogdanović et al., 2013). Typically, people who fit these age ranges undergo some form of education and developing a quiz-game application with elements of education is one of many things that compelled me into advancing with the idea. Another reason why I chose to develop a mobile phone application is to develop new skills in this field including database management (set-up, maintain, manage), designing UI, learning a new language. These are all skills that personally I had wanted to obtain, and this project focuses on these areas.

Aims and Objectives

This project aims to create an IOS mobile game that test user's knowledge on various topics in a form of multiple-choice questions. The objectives of the project comprise of reviewing and reporting on the literature surrounding IOS applications as a guide to designing and developing the quiz game. To design an interface following the IOS design principles using model view controller pattern. To develop the mobile application using software and swift language. To create a user-friendly interface which is easily navigable.

Section 2: Software Development Methodology & why is it important?

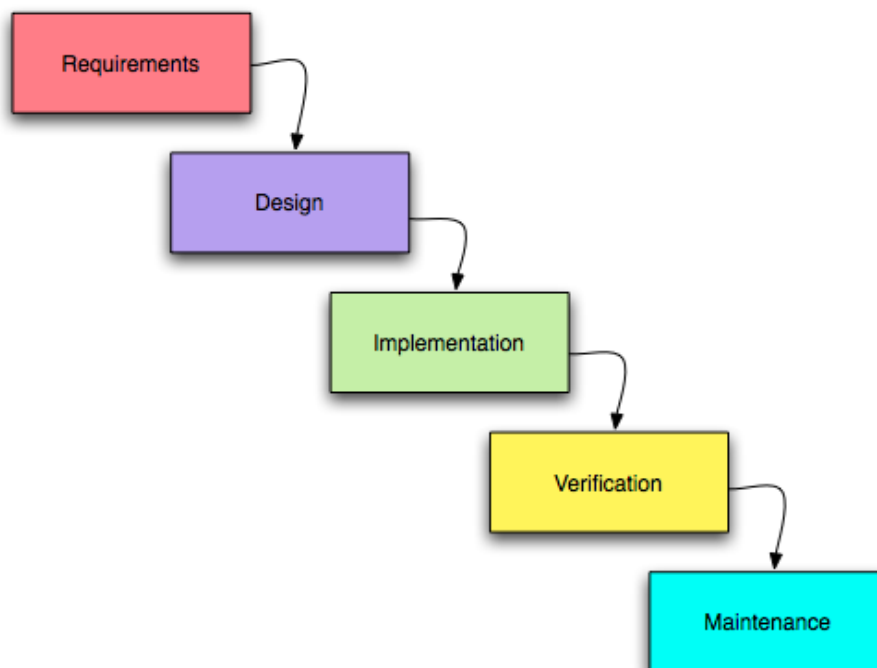
What is the importance/reason that a project must follow a certain software development methodology? It is important to first of all define what a methodology is. A methodology helps to keep a consistent framework through the life cycle of any small or large project which consequently raises the chance of success for that particular project. Studies show that projects that do not follow any type of methodology will have a larger failure rate. There are many different types of methodologies that cater to different projects, some of which can be detrimental to a project, so it is vital to pick the right one. (Charvat, 2003).

Methodology Chosen

The waterfall Methodology comprises of 5 different phases to be completed in a sequential order completing one stage before moving to the next in order to develop a software solution. This was the software methodology chosen for this particular project. The waterfall methodology has a very clear structure which is easy to understand and follow (Alshamrani and Bahattab, 2015).

Waterfall Model

As you can see from the picture below, the waterfall model contains 5 stages including requirements, design, implementation, verification and maintenance. The waterfall model is the oldest model to exist and is widely known/used. Below, there will be a brief description to each of the stages and how the processes work.



Requirements

In the requirements phase, the system behaviour, aims and objectives are gathered. Usually these descriptions are gathered from clients. A detailed requirement documentation will be written to make sure that everything is agreed upon and to ensure the strategy and approach. It establishes an agreement between both the client and the developers. This is extremely important because every other stage will be based on making sure that the requirements are met and fulfilled.

Design

The information used from the requirements phase is evaluated and a proper implementation is formulated. The design phase effectively uses the previous information by choosing appropriate algorithm designs, software designs, database design techniques such as ERD diagrams UML diagrams and use case diagrams. Furthermore, an appropriate programming language will be chosen in the design phase however the coding will only start in the next phase. The design phase basically is a process of planning and solving problems and finding solutions to those problems by structuring everything before implementation (Alshamrani and Bahattab, 2015).

Implementation

In this phase the requirements will be used alongside the design ideas and used to create a solution using the selected language and structure. The implementation is where the developers excel in, essentially creating the software and making sure to meet the requirements put out by the client (Alshamrani and Bahattab, 2015).

Verification (testing)

In the verification phase, the software solution is monitored and checked to see if there are any glitches, system failures, bugs. If there are any, then it is in this phase that they find and fix these issues. Furthermore, this phase deals with checking if the software solution has met the original requirements to its fullest. If issues arise with not meeting certain requirements these will be corrected in this stage before moving into the next (Alshamrani and Bahattab, 2015).

Maintenance

The waterfall models' final stage is known as the maintenance stage. This phase takes place after the software is released. The software may still need some sort of modification, corrections or refinements to it. These are addressed in the maintenance stage. The project may need to return back to the first stage of the model if serious issues arise (Alshamrani and Bahattab, 2015).

Why the waterfall model?

Waterfall is one of the most traditional and commonly used methodology in software development. There are many advantages to choosing this model some of which include:

1. Clear structure – Compared to other methodologies which are very complicated in the sense that their stages are many and can overlap, the waterfall model is very simple to understand and navigate.
2. Sequential – Each stage must be completed before moving onto the next. You do not have to go back to a stage complicating things.
3. Planning – All the planning and setting requirements is done during the requirements phase making it easy to understand.
4. Determines the end goal early - Committing to an end goal from the beginning of the phase allows users to have a clear understanding of what must be done in order to create the software
5. Information is passed on well – waterfall prioritizes accessibility towards information
6. Works well with small projects (lucidchart, 2017).

Despite the waterfall model being the most widely used, it does have some disadvantages to it because not every single project will work with it. These include:

1. Changes become difficult – Since waterfall is an incremental step by step approach. It leaves no room for unexpected changes to specific details of the project maybe certain ideas become outdated but since you cannot work backwards with this approach it becomes hard to do.
2. Risk – There can be a huge risk to this methodology for the exact reason.
3. This methodology does not suit certain projects that are very complex and require constant change.
4. Focuses little on the end user – If the requirements are completely met from the first stage, it doesn't matter if the user isn't completely satisfied with it.
5. Delays testing after completion – leaves little room to test which can be risky.
6. Documentations can be extremely long
7. Not suited to huge projects (lucidchart, 2017).

Why the model suited this project

The reason why the waterfall software development methodology was adopted into this project is because of the size of the project most importantly. Since this project is relatively small, the waterfall model is very effective. Furthermore, the quiz mobile application is a project which as a student needed to be completed and because we know that this type of project won't require significant updates/changes with requirements, this makes it suit the needs of an iterative process with the requirements listed at the start. Waterfall methodology caters to all of this which is why it was picked for this particular mobile application. In the beginning phases for this project (requirements and design), detailed description of functional requirements, system designs, stakeholders, use cases were produced. First, we described each of the requirements that needed to be met in order to make it a success. A system design documentation was also made which included initial designs of the UI and some diagrams including ERD, Use cases. This helped visualize what the application would look and act like and following these steps were extremely important to make the project a success. During the implementation stage, as the only developer in the team. The responsibility was solely on me. Which makes it less complicated having huge teams due to miscommunications that could happen. For these reasons the best methodology to ensure success for this project had to be the waterfall model.

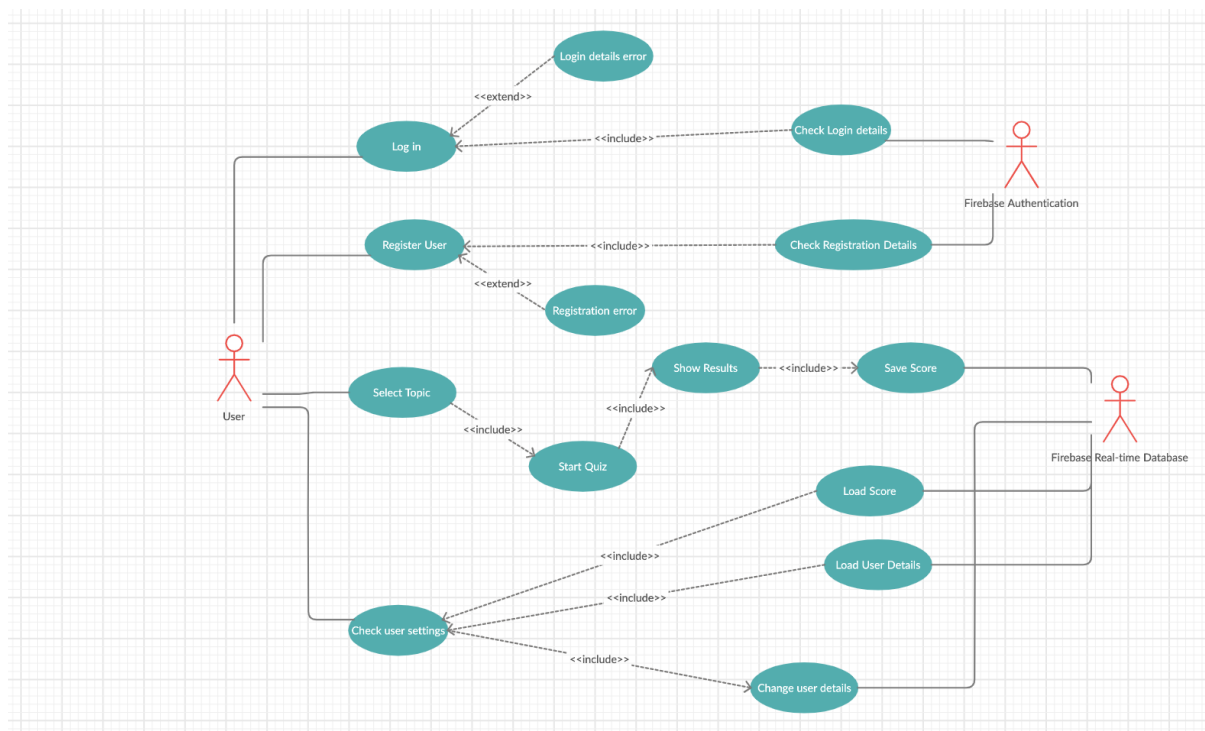
Section 3: Diagram Designs

Use Case Diagram:

What it is?

A use case diagram is an object-oriented modelling construct that is used to define the systems behaviour (Snegupta and Bhattacharya, 2016). Use case diagrams are important especially when it comes to visualizing how a certain system operates and works. Interactions between the User, system and database is described in a logical manner with connections so that it is very easy to read and understand. It acts like a prototype in that sense. The benefits associated with use case diagrams are:

- Helps visual functional requirements
- Easy to understand
- Can be used to estimate and schedule tasks (Diagrams?, 2020)



My project

As you can see from the image above, we have a user at the start. The user refers to anyone who is interested in the mobile quiz application. Every user experiences these different phases in the diagram. On the other side we have the Firebase Authentication which is responsible for authenticating users checking login details and registration details. The firebase real-time database is another part of firebase which is responsible for saving user

Yacquub Adan
P17150791

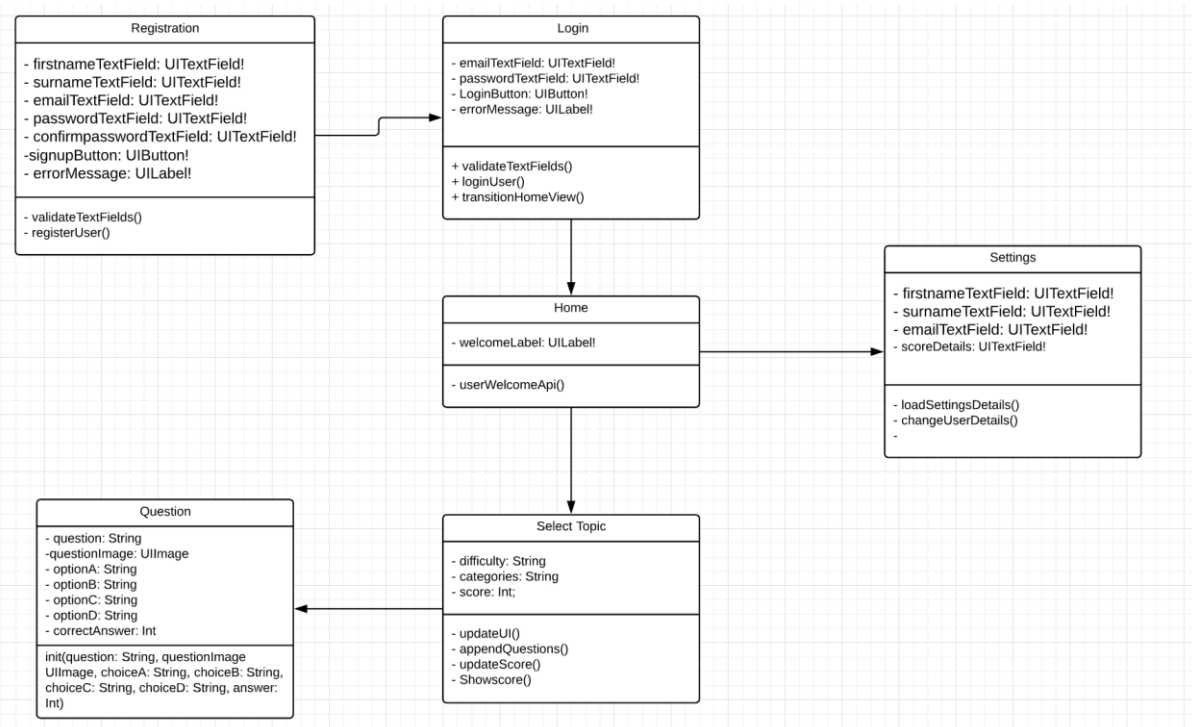
details loading scores, updating scores and loading user details. Each user can either register an account or login if they have an existing account already. If they type in the wrong details, then the firebase authentication server will detect it and an error will be sent back. This applies to logging users in. From the home screen the users can check their user settings details which will load and showcase data such as the score of the last quiz taken, the first name of the user, the last name of the user, and the email address. Users can also change their details which will be updated by the firebase real time database. When the user decides to start a quiz, after they have finished the quiz the results will be shown and stored in the real time database. Each time they finish another quiz these details will be updated in the settings page.

UML Diagram:

What is it?

Unified Modelling Language is a standard modelling language for object-oriented modelling and documentation (Spetian, et.al., 2017). It has similar purpose to use case diagram in the fact that its purpose is to visually represent a system along with more complications such as identifying the different classes, variables, actions, functions and events (Torre, et.al., 2018). Furthermore, identifying how each class relates to each other. The advantages of UML Diagrams are:

1. Visual Representation – like use case diagrams it can help visualize and speed up implementation if designing a proper UML diagram
- Reusability – can look back into this design even in the implementation stage (Friesen, 2018).



As you can see from the UML diagram above, the registration class has many text fields which represent what the user inputs when they would like to register. The functions are all listed below as you can see the registration class has 2 functions one that validates text fields making sure the user hasn't entered any details incorrectly and if they have it should correct them. The other function is register user function which basically registers the user to the database. The login class has similar text fields but fewer due to the fact that users don't need to re-enter specific details only the important ones. Again, there is the validation method that checks to see where users have correctly typed in their details without any errors. The login users function basically checks to see if the user has already an account registered and if so logs them on, if not there will be error messages preventing from going further. The home view has few variables a welcome label which welcomes the user via the function. As you can see every other class has different variables and functions associated with them and it helps visualize the entire application and how it should run/work.

Section 4: Integrated Development Environment

An IDE, or otherwise Integrated Development Environment is a software suite that enables developers/programmers to test and write up software. Most IDE's have many tools including test libraries, compilers, debuggers etc. Most IDE's are used because it detects syntactical errors by understanding the language making it easier for the user to find the errors in the code. Also, IDE's offer colour coordination which makes the code easier to read and decipher (AppleInsider, 2018). The picture below shows this.

```
// without syntax highlighting

public class NiceDay {
    public static void main(String[] args) {
        System.out.println("It's a nice day out!");
    }
}
```

```
// with syntax highlighting

public class NiceDay {
    public static void main(String[] args) {
        System.out.println("It's a nice day out!");
    }
}
```

For this particular project, the IDE chosen was XCode. XCode is an integrated development environment pulling in all essential tools to produce an application. XCode is apples official and free of charge IDE for iOS developers. It is primarily used for creating apps within its ecosystem (iOS, iPadOS, macOS, tvOS, and watchOS). XCode can be used for designing user interface, testing code, writing up code, compiling and building. The IDE is also used once completing an app to send to apple store market. This software was first called Project Builder and was introduced in 2003 and later renamed. XCode is a very powerful and complex program some of the features of it includes:

1. App Simulator – You can run your application with a simulator of an iPhone device on your MacBook.
2. User Interface Prototyping– Allows users to design user interface (storyboards) with many tools to do so.
3. Auto code completion – suggests users' advice with what they are looking to type speeding up the process
4. Coding assistance
5. Split window – look at multiple files simultaneously

Language

Xcode supports various different languages some of which include C, C++, Java, AppleScript, Python, Ruby, Rez and Swift. Swift and Objective C is the most popular and primarily used language in Xcode for developing IOS applications. XCode has multiple programming models such as Cocoa, Carbon and Java so that users have many options to choose from (Das, et al., 2016). Choosing the right programming language to use was difficult however the most popular language used is swift and objective c. The chosen language for this project was swift because it offers a host of features and due to the fact that the database chosen supported the language.

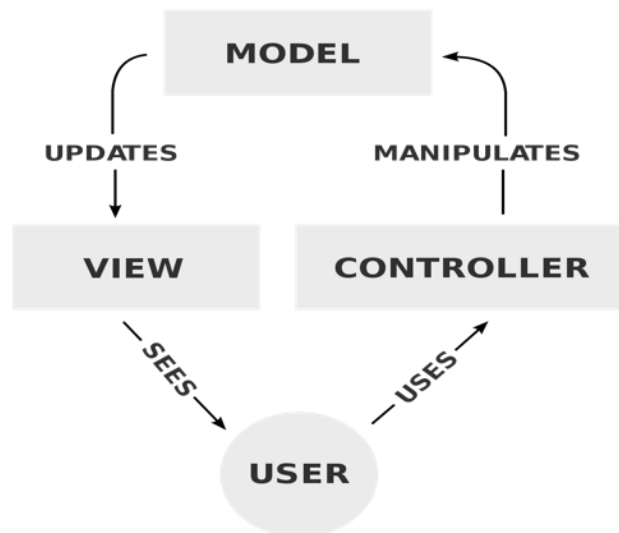
Swift

Apple introduced a new programming language called Swift on June 2014. Previously Objective C was Apples main programming language but due to it being an old language adopted in 1983, swift was introduced and designed to be an improved version of Objective C without any constraints and used for all the apple platforms. The swift language is known to be multi-paradigm. This means that it has features of an object-oriented programming language but can also be functional and imperative (Reboucas, et al., 2016). The benefits of Swift are that it was designed to work with Cocoa and Cocoa Touch which are frameworks of XCode, and they have recently announced SwiftUI which is dedicated to this specific language. Other benefits include the fact that it is an improved version of the old language objective C and has features such as optional making it far more concise. This compelled the project to be written in the swift language (Neuburg, 2016)

Organizing Code

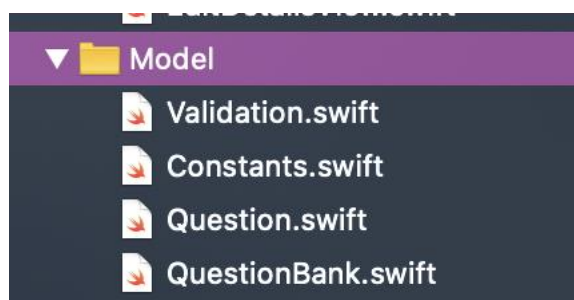
Organizing code is an extremely important practise that every developer should make in order to create an environment where the code is easy to navigate, read and test. There are several different software design patterns. A software design pattern consists of one or many software design elements such as interfaces, classes, objects, methods, functions and their relationships between each elements. Design patterns are reusable some of the famous ones include model view controller (mvc), Layered System(Tichy, 1997). Design patterns can speed up the implementation phase and improve the quality of the software as a whole. It allows developers to tackle specific issues with ease. Every design pattern has its own set of advantages and disadvantages which is why it is very important to choose the correct like choosing the correct methodology.

The Model View Controller architecture was software design pattern used in this project simply because it was the one that was familiar personally due to the modules that have taught this architecture during the second year of university. Due to the experience this pattern was chosen also because it fits completely with XCode and the swift language which is multi-paradigm making use of object-oriented programming. This design pattern also is an easy concept to understand and works especially well with IOS development and supported by frameworks such as UIKit. Model View Controller is separated into 3 different parts which will be showcased below.



Model

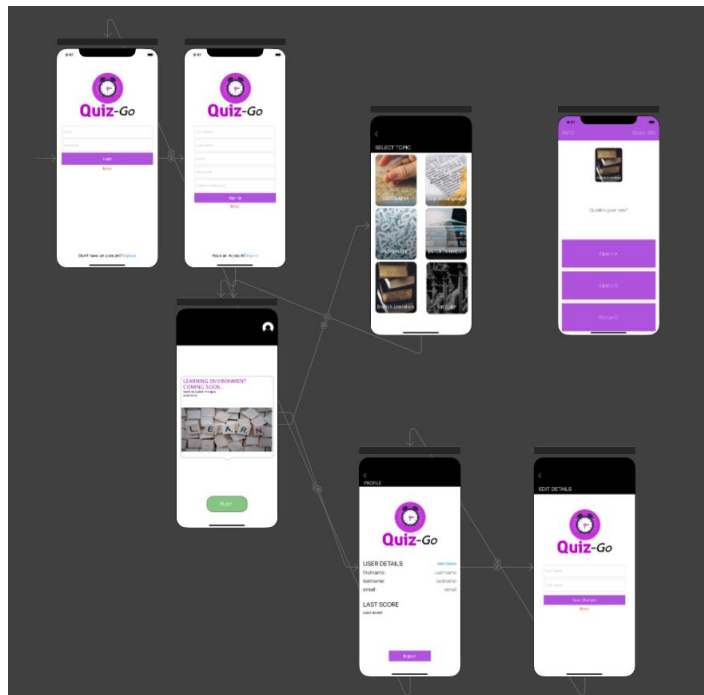
The model part of the architecture is the applications central structure. This manages the data and logic of the application as a whole it can be as simple as an integer (model of a counter) or string or it could be complex(Krasner and Pope, 1988). The quiz application also has some models shown below.



As you can see from the image above, the model of the quiz application has a question bank containing the question data and has another class called question which basically controls the logic (LearnAppMaking, 2020).

View

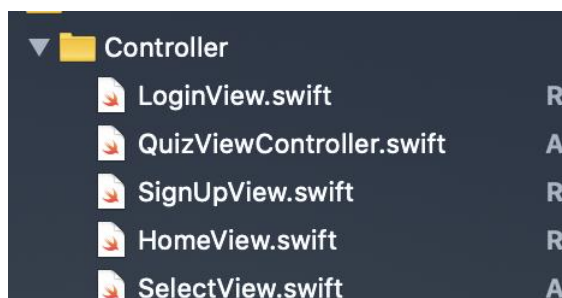
The view deals with all the graphics and user visuals, the view requests data from the model and this it is displayed as the view(Krasner and Pope, 1988). This is what the user is exposed to. The user interface which was designed in the project will be showcased to the user. An example of the view for this project is:



The storyboard is something specific to XCode which allows developers to design the user interface which is exposed to the viewer

Controller

The controller is responsible for scheduling interactions with input devices such as keyboard, pointing device, buttons, time etc. Controllers contain the interface between model and views(Krasner and Pope, 1988). In this project an example of the controller was:



Section 5: Firebase

Google released Firebase in the summer of 2016. Firebase is a google-owned toolkit and infrastructure that supports and helps developers build better applications. Firebase's goal is to provide all the necessary tools that you need to build great apps, grow a successful business and earn from your hard work (Moroney and Anglin, 2017). The quiz application made use of Firebase as its backend service due to its many benefits which will be listed below.

Benefits of Firebase

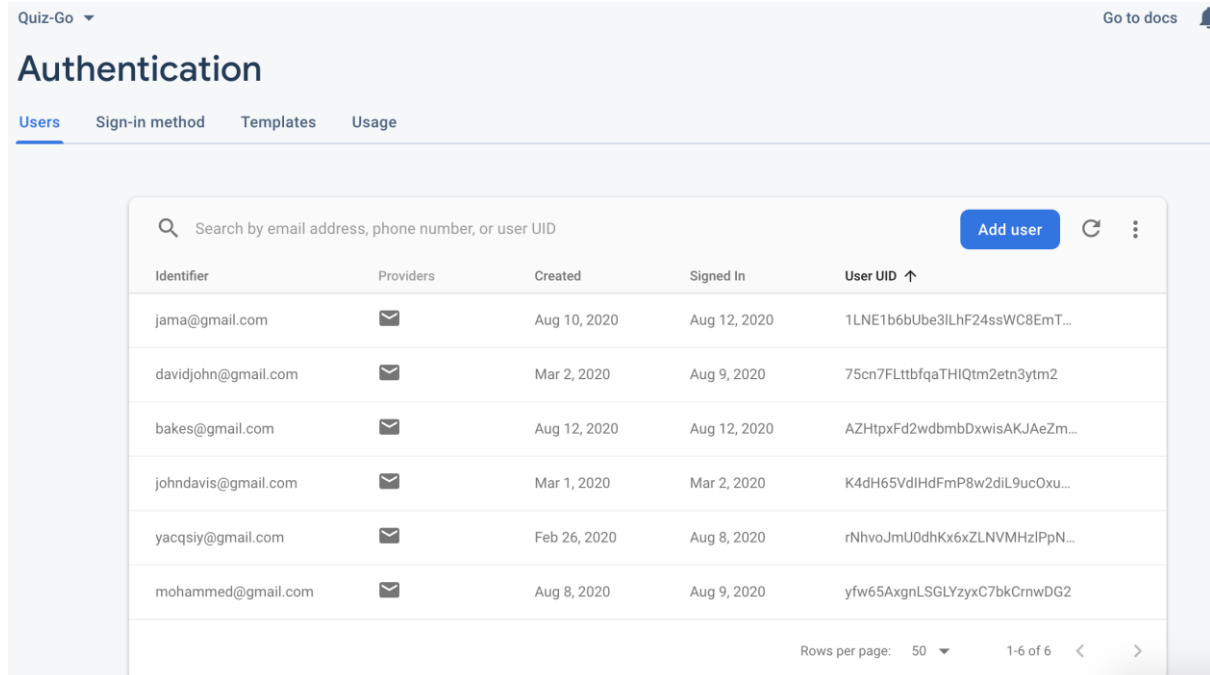
1. Firebase analytics – Provides insights, stats and figures into app usage. This is really useful for the developers to keep updated into customer retention and how well the app is doing overall
2. Firebase Authentication – supports login providers such as google, Facebook, GitHub etc.
3. Firebase Storage – easy and secure file transfer.
4. Cost effective – much of the services are completely free of charge.
5. No need for API – it's an enhancement to API's.

The mobile quiz application has used Firebase to store, authenticate users within various systems. The registration system stored the user's credentials into firebase. The login system checked the user's details to see if it matched what was stored and there were other uses of the firebase database such as updating, adding the score etc. The first firebase service which was used in the app is Firebase Authentication. We will be looking into exactly what this service is responsible for and where we used it in the app.

Firebase Authentication

Mostly every modern app need to somehow identify the user in order to provide unique data to that particular user. This is most commonly done by a sign in/register process. The firebase authentication makes this whole process much easier and simpler. Some users may find trouble in giving personal data to an application due to risk factors and ultimately not continue on with the app however applications that allow users to login via different methods such as Facebook, Google, Twitter tend to make life much easier and have risen in popularity (Khawas and Shah, 2018). For these reasons Firebase Authentications helps with this issue allowing users to log in with past credentials from other providers or an easy email password scheme. Firebase Authentication also can be integrated with other services like the real time database which we will explain shortly.

The quiz application project has incorporated this service by allowing users to register and login. The authentication process in the app that we have used is the email and password authentication which is the standard one.



Identifier	Providers	Created	Signed In	User UID ↑
jama@gmail.com	📧	Aug 10, 2020	Aug 12, 2020	1LNE1b6bUbe3ILhF24ssWC8EmT...
davidjohn@gmail.com	📧	Mar 2, 2020	Aug 9, 2020	75cn7FLttbfqaTHlQtm2etn3ytm2
bakes@gmail.com	📧	Aug 12, 2020	Aug 12, 2020	AZHtpxFd2wdbmbDxwisAKJAeZm...
johnDavis@gmail.com	📧	Mar 1, 2020	Mar 2, 2020	K4dH65VdIHdFmP8w2dIL9ucOxu...
yacqsiy@gmail.com	📧	Feb 26, 2020	Aug 8, 2020	rNhvoJmU0dhKx6xZLNVMHzlPpN...
mohammed@gmail.com	📧	Aug 8, 2020	Aug 9, 2020	yfw65AxgnLSGLYzyxC7bkCrmwDG2

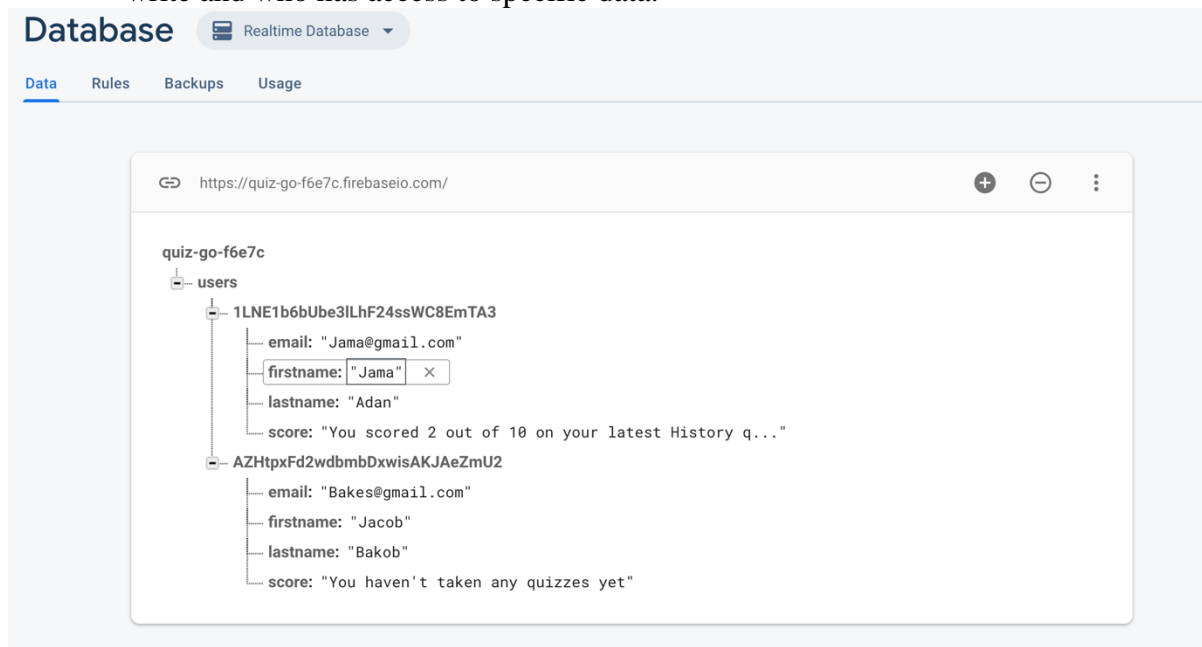
As you can see from the image above the sign in method that was chosen was the email/password method being the standard one used for most apps of the century. Each email shown below is from users who have already registered with the quiz-go project app. As you can see each record shows the identifier (email) followed by the provider and the date created. Each user has a unique user ID otherwise known as UID which can be seen as long combination of letters and numbers. By clicking on the Add user button you can create a new user through firebase console if you wish to test something out. Furthermore, the refresh icon button refreshes to see if there were any recent users who have just registered with each new registration comes a new identifier, date and user ID.

Real-Time Database

The real-time database is a cloud hosted NoSQL database that provides syncing across connected devices and is available even when there is no network connectivity through local cache. All the coding is done through the client. Data is stored as JSON and synchronized to every connected client in real time and automatically receive updates. Its design is to be responsive (Khawas and Shah, 2018). The real time database differs from relational databases due to its different functionalities and optimizations. Some of the capabilities of it include:

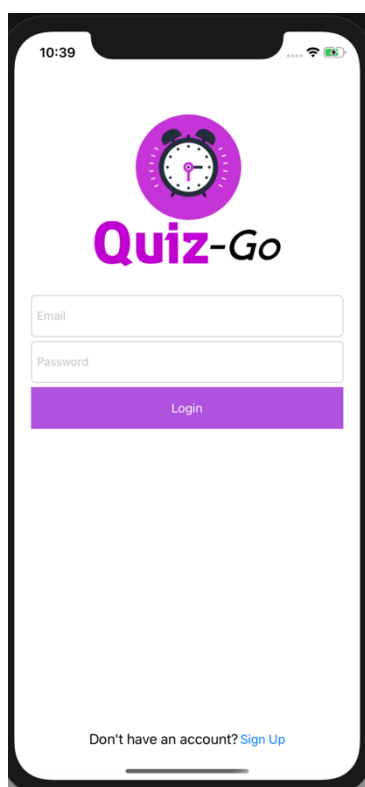
1. Realtime
2. Offline
3. Access from client devices

If this is integrated with firebase authentication, developers can define who can read and write and who has access to specific data.



As you can see from the picture above. The UID that was created from the authentication is also used in the real time database to distinguish users. Looking at a specific user detail we see that information such as their email, first name, last name and score is saved. In the mobile quiz application real time database has been used primarily to save, update and change user details and keep track of the latest scores. As you can see the user with the first name “Jama” has scored 2 out of 10 on his latest history quiz. The real time database is really easy to understand and read which is why it was used and useful for this particular project.

Section 6: User Interface



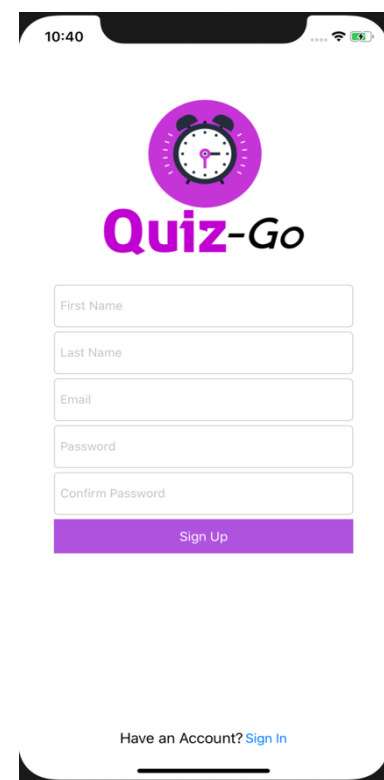
Appearance

The Login screen is the first screen the user will see after the loading screen. The login screen consists of the quiz logo, two text-fields, a login button, a bottom label, and a sign-up button.

The logo contains an image of a clock, a purple and black theme and the name of the app itself. The Importance of having a name in the logo of the app is so that users remember the name if they ever happen to come across it. The name of the app Quiz-Go was established because the whole idea behind the app is a quiz-like game with multiple choice questions. The name is unique since there isn't any other quiz game with that name which is why it ended up being used for the project. I decided to use this black and purple theme throughout the whole application because purple is a very eye-grabbing colour and black fits well with it.

Functionality

The two text fields in this page are email and passwords. It requires the user to input their existing email address and password and it assumes that the user has already registered an account. If the said user doesn't have an account, they won't be able to log in. The login button works in a way that checks to see if the user has written in both of the fields and if not, an error message will appear highlighting the user to fill in the fields. It spots syntactical errors of email and password fields too so if the user types in an email address that isn't formatted correctly an error message will appear highlighting to change it. This is the same with the password. If you type in an email and password that isn't stored in the firebase authentication database, then it will refuse to log in also. By clicking on the sign up button on the bottom of the screen you will be directed to the registration page.



Appearance

The logo is located in the same position as in the login page. There are 5 fields that all need to be filled in in order to register. The fields include first name, last name, email, password, confirm password. These are all standard information that most apps take from users in order to register them to the application. Below the text fields you can find the sign-up button. If you notice they all follow the same theme as the login page (majority purple some black). Below that there is a label similar to the login page and a sign in button in case users have registered and want to go back to the login page.

Functionality

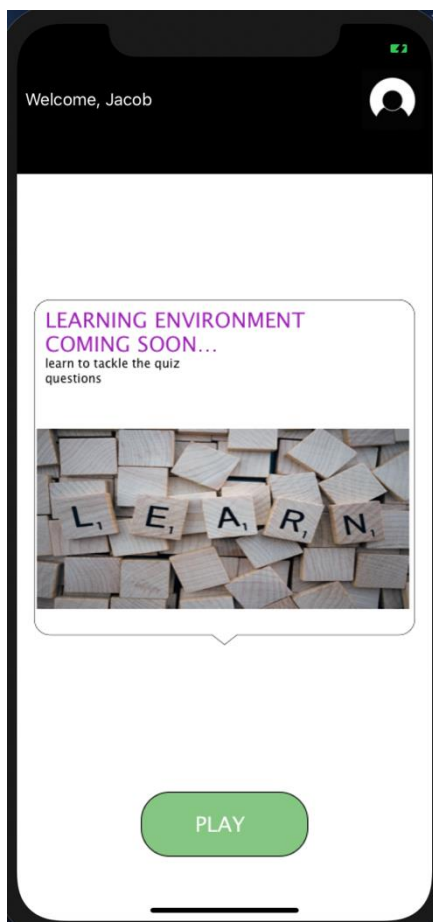
The text fields similar to the login page require users to fill them out correctly and in the right format. If done so correctly, the user will be able to register. Below the text fields the sign-up button is located in which users will be ready to register their accounts if the details are all correct. Once they user touches sign up, they will either encounter errors with validation or successfully register and their authentication details will pop up in the firebase authentication log (see firebase section for more details). If the user has successfully registered a message will show up bellow the sign-up button informing the user that he/she has been successful in signing in. The sign in button at the bottom of the page is almost identical to the login page however it directs the users back to the login screen.

Please make sure that the password is 8 characters long, includes a special character and a number

Please fill in all fields

Invalid Password

Here are some examples of the error messages that I have mentioned in the login and registration view. Not all of them are shown however the general idea is here.



Appearance

The home screen or view contains a label at the top left welcoming the user with their name. It also contains a profile icon which can be used to check out the settings of that particular user. The top part of the home screen has a section with a black background colour which once we explore this application further is common and becomes a pattern throughout the application. In the middle section of this page you find the update log which contains details of upcoming updates in this case a learning environment will be introduced where users can learn topics. Underneath this section there is a green play button that users can click on to play the game

Functionality

The welcome label at the top left of the home page is connected to the firebase real-time database and reads the users first name based on what they have inputted from the registration page. Once a user logs in their individual names will be on this label with a welcome text along with it. The profile icon leads to a separate page known as the settings view which users can go into to check/change their details. The play button directs the user to the select topic page which is typically pressed if the user would like to start choosing a topic and playing the quiz.



USER DETAILS

[Edit Details](#)

firstname: Jama
lastname: Adan
email: Jama@gmail.com

LAST SCORE

You scored 2 out of 10 on your latest History quiz!



Save Changes

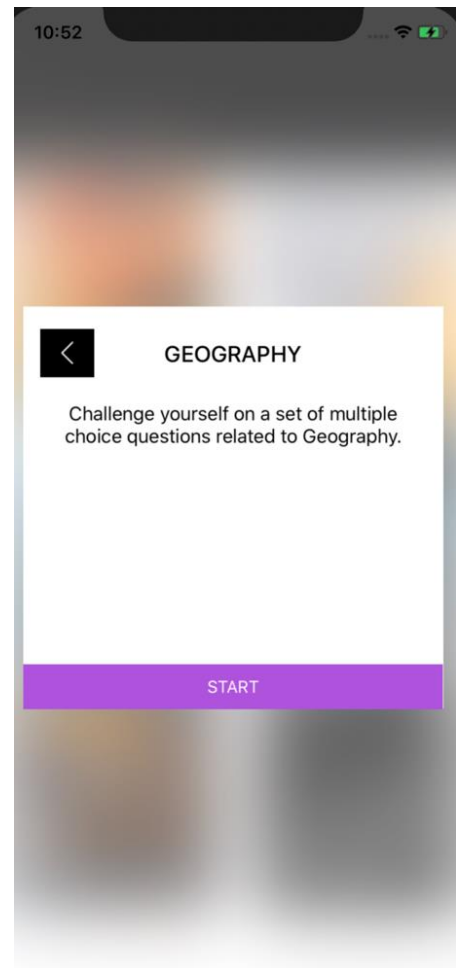
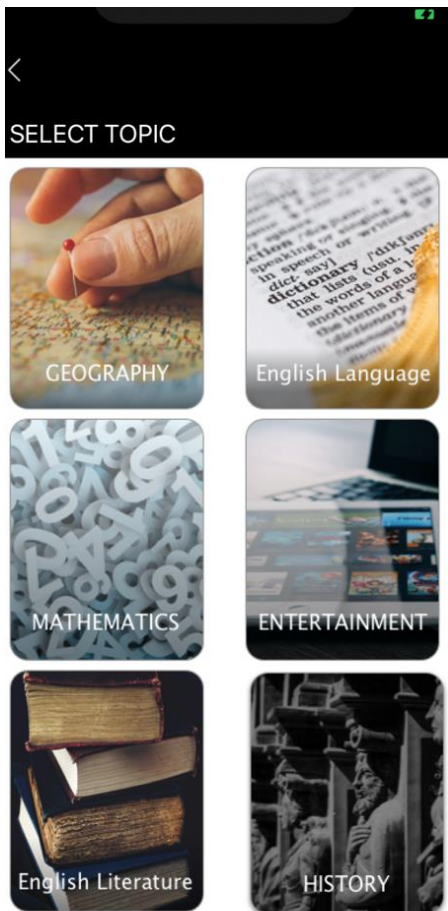
logout

Profile View.

The settings or profile page is where users go in order to check if their details are correct in case, they inputted the wrong names etc. If so, they can edit their details which leads to a separate page to do so. As you can see the theme stays true and consistent so far throughout the mobile application. There are several labels in the settings page. Under user details you can see that it shows the details of that user which is acquired from reading the contents of the real time database (see firebase section). Each user will have different details specific to theirs. Under the user details section contains information about the user score. Each user has different scores, and this is read from the firebase database. Below this is the logout button which securely and safely logs the user out. Upon clicking this button, the user is redirected to the login screen. At the top of the page the label profile makes sure that the user understands where they currently are in the app. The back button directs the users back to the home page.

Edit Details View.

The edit details view looks almost identical to the login page and the registration page. I wanted it to have the same layout staying true and making the app look consistent. Users are able to change their first and last name through firebase database. Users are required to fill in both fields before pressing the save changes button, if they fail to do so then similar errors will show such like the one in registration page. On top of the page there is a label called Edit details that show the user where they currently are at and a back button. The back button works in a way that makes it so that users are redirected to the settings page

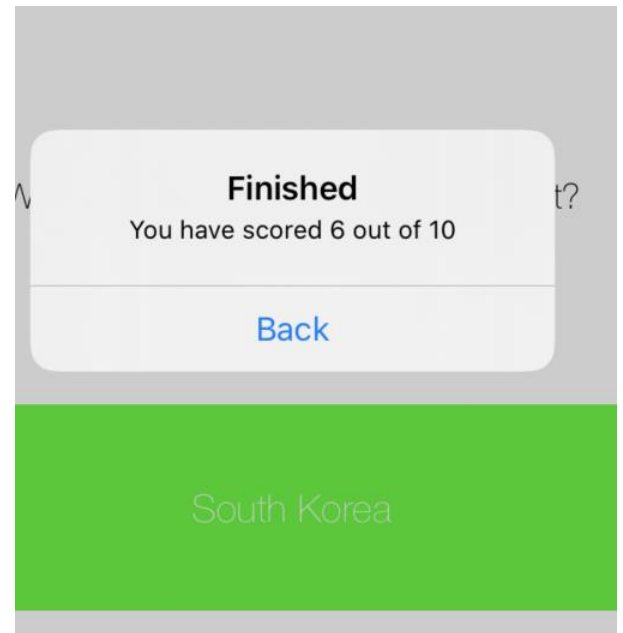
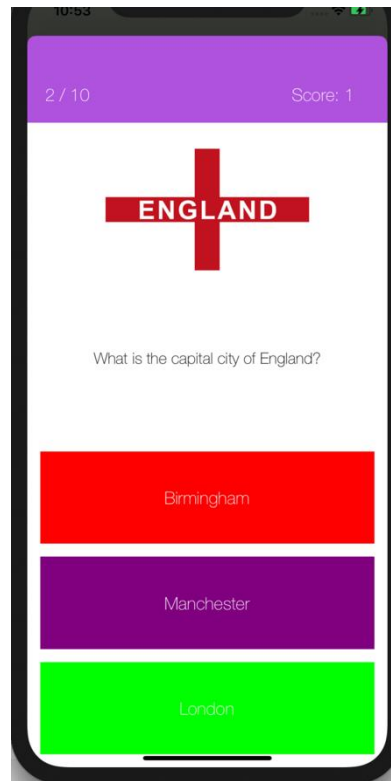


Select Topic View

The select topic page has the same exact upper part as the home, settings, edit detail pages where they have a black background section with a label informing the page title and a back button. Below this section is where you will find all the topics. The first topic is geography, as you can see the geography topic has a background image of the world map with the name of the topic too. The English Language topic also has the name on it with a zoomed in dictionary. The mathematics topics background image is a bunch of numbers crammed up near each other. The Entertainment topic has a picture of devices as its background. The English literature topic has an image of books piled up one another. The history topic has an ancient looking image as its background image. All of these images relate to the topic which is why it was chosen. Users can select any of these topics that they desire to test themselves in. The back button at the top of the page directs users back to the home screen with the play button. Each of the topics are buttons that can be pressed and interacted by the user. Upon pressing/clicking on the appropriate topic, a popup will appear.

Pop-up view

The popup view changes for each specific topic pressed. For instance, the image above shows what would appear if the user clicked on the geography topic. The popup window shares the same black and purple theme as highlighted in previous pages. On the top left of the popup screen you see a back button that leads back to the select topic page. The title for each topic changes and it's the label on the top centre that changes when users select different topics. The label at the centre below the title is a brief description of what type of questions users will get in this case multiple choice questions. The start button on the button is pressed when the user wants to enter the quiz and it directs them to the quiz view



Appearance

As you can see the quiz section follows the purple theme. On the top you can see that there's a question counter label and on the other side of it there's a score label. This section has a purple background colour. Below that there is an image view of the quiz questions. In the case of the geography quiz test this change with every question. Below is the actual question label where the user sees and reads out the question that must be answered. Below are the options that the user can press. There are 3 options that are all interactive buttons.

Functionality

The question counter changes upon clicking on an option. The question counter increments, and the score label increments based on whether the user has pressed on the correct option. The image view changes with each question. For example, in the geography quiz test above the image changed from a Spanish flag to an English flag. As you can see from the image above, If the user selects the wrong answer the background colour of that option button changes to red and the real answer is shown to the user as a green background colour after 0.5 seconds. If the user clicks on the wrong answer the score will not increment however if he does it will. Furthermore, the question itself changes upon clicking on an option. After clicking on an option of the users choosing the next question will load in 2 seconds after. At the end of the quiz a popup alert will appear with the title finished which will have the exact score and the total number of questions. Clicking the back button on the popup will redirect you back to the select topic screen and it will update your score details into the firebase real-time database and consequently showcase in the settings view.

Section 7: Test cases

Testing is important especially for software because it will point out the errors and defects during the development stage and will help minimize them. The test plan will outline how we plan to test the application throughout its development and showcase some test cases and their expected and actual result.

Test Case	Expected Result	Actual Result
Loading page	While the application is loading the loading, page appears with the logo in the middle and white background	Pass
Login	User types in his email and password and logs in successfully	Pass
“Don’t have an account? Sign up” button	When clicked it should lead to the registration view for users to sign in.	
Registration field validation	If users do not type in anything in a field, then the error label will instruct users to fill them in	Pass
Registration Email validation	If users type in the wrong email address combination e.g. with the wrong combination, then the error label will read “Invalid Email Address”	Pass
Registration Password Validation	If the user types a password shorter than 8 characters without any special characters or numbers then the error label will read “please make sure that password is 8 characters long, includes a special character and a number”	Pass
Registration Confirm Password Validation	If the confirm password text field doesn’t match the password text field, then the label will state “Make sure that passwords are the same”	Pass
Login button	Considering the user has registered it should lead to home page	Pass
Sign up button	Once the user has successfully typed in all the fields with the right information then the database adds that user.	Pass
Home page play button	When click it should lead to the select topic page	Pass

Profile icon button (home page)	When pressed it should lead to the profile/settings page	Pass
Profile back button	When pressed it should lead to home page screen	Pass
Edit details button (in profile page)	When pressed it should lead to the edit details page	Pass
Edit details back button	When pressed it should lead to the profile/settings page	Pass
Edit details save changes button	When pressed it should save new user details if they have typed in both first name and last name fields (save to firebase)	Pass
Edit details Validation	If users do not fill in both fields then they will be instructed to do so via error message	Pass
Logout button (settings page)	When pressed it should log out the user and direct them back to the login page	Pass
Select view back button	Once click it should lead to the previous page which is home page	Pass
Select View topic buttons	Once clicking a topic, it should present a popup that highlights the topic name and description with a back button and a start button	Pass
Select View topic pop-up back button	When pressed should lead back to select view	Pass
Select view topic play button	When pressed should enter the quiz view	Pass
Quiz view finished back button	When pressed should lead back into the select view and save the quiz score	Pass
Quiz option buttons	When pressed should lead to the next question	Pass
Quiz questions	Tests to see if the quiz questions are loaded, successfully incremented.	Pass
Updating score	Checks to see if the scores are updated after each quiz in the settings page	Pass
Save Details	Checks to see if detail changes are correctly saved in the real time database and in the settings page	Pass
Correct Answer	Checks to see if the user has pressed the correct answer that the background colour changes to green	Pass
Wrong Answer	Checks to see if the user has pressed the wrong button that	Pass

	the background colour changes to purple	
Button functionality	All the buttons must function as specified.	Pass
Keyboard	Users should be able to use their device keyboards to input text into the fields required	Pass
Censored text fields	Passwords and sensitive information should be censored when typed	Pass

Section 8: Critical Review

To conclude, A critical assessment had to be done in order to review the outcomes of the project. All of the objectives and requirements have been completely met for this project. The initial aim of the mobile quiz application was creating an IOS mobile game that tested user's knowledge on various topics in a form of multiple-choice questions. This aim has been met because the application contained 6 topics which were different in their own ways and users were able to select the topic of their choice and test themselves in the exact proposed format. So, with regards to this aim it was successful in fulfilling it. Some of the objectives met were designing the user interface which was met during the first deliverable and also researching on the literature. Furthermore, objectives included producing a report creating test plans making sure users are logged in securely, managing databases were all met making the project a success.

Software tools

The software and tools used were very suited to this particular project since the app was developed on XCode an IDE that caters to IOS devices which personally had owned making the process much easier to setup. Furthermore, key skills were learned in the process of making this application such as learning a new language called swift and experiencing the tools of a new IDE (XCode). Other skills include being more experienced with the software design pattern known as MVC.

Improvements

Some things that could've went differently and improved the app more would've been to include a picture saving system for each user so that they could set up their own profile pictures. Another possible improvement could've been to have used an API to get all the questions from an online question bank. Other improvements include making the login system have more options like using existing huge platforms like google, Facebook to log in. adding a timer to the questions would have also improved the application as a whole.

Issues along the way

There were many issues that had come along which were unexpected such as the global pandemic which had forced everyone to stay indoors resulting in the lack of certain key resources which made it extremely hard to stay on schedule. This created issues regarding deadlines for the project but because of the pandemic we were issued extra time in compensation to finish up what we have missed out on. Furthermore, the pandemic caused meetings to be done over the internet instead of in person which led to miscommunication issues due to the lack of resources. However I believe the skills that have been acquired are more beneficial than all the issues along the way.

Bibliography

Bassil, Y., 2012. A simulation model for the waterfall software development life cycle. *arXiv preprint arXiv:1205.6904*.

Alshamrani, A. and Bahattab, A., 2015. A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model. *International Journal of Computer Science Issues (IJCSI)*, 12(1), p.106.

Lucidchart.com. 2017. *The Pros And Cons Of Waterfall Methodology*. [online] Available at: <<https://www.lucidchart.com/blog/pros-and-cons-of-waterfall-methodology>> [Accessed 12 July 2020].

AppleInsider. 2018. *Xcode | Updates. Features, Languages*. [online] Available at: <<https://appleinsider.com/inside/xcode>> [Accessed 13 July 2020].

Neuburg, M., 2016. *IOS 10 Programming Fundamentals with Swift: Swift, Xcode, and Cocoa Basics*. "O'Reilly Media, Inc."

Das, S., Singh, G. and Kumar, B., 2016. Windows based graphical objective C IDE. *International Journal of Advancements in Technology*, 7(1), pp.1-5.

Rebouças, M., Pinto, G., Ebert, F., Torres, W., Serebrenik, A. and Castor, F., 2016, March. An empirical study on the usage of the swift programming language. In *2016 IEEE 23rd international conference on software analysis, evolution, and reengineering (SANER)* (Vol. 1, pp. 634-638). IEEE.

Tichy, W.F., 1997, August. A catalogue of general-purpose software design patterns. In *Proceedings of TOOLS USA 97. International Conference on Technology of Object Oriented Systems and Languages* (pp. 330-339). IEEE.

LearnAppMaking. 2020. *Model-View-Controller (MVC) On Ios – Learnappmaking*. [online] Available at: <<https://learnappmaking.com/model-view-controller-mvc-swift/#why-you-should-use-mvc>> [Accessed 13 July 2020].

Krasner, G.E. and Pope, S.T., 1988. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, 1(3), pp.26-49.

Moroney, L., Moroney and Anglin, 2017. *Definitive Guide to Firebase*. Apress.

Khawas, C. and Shah, P., 2018. Application of firebase in android app development-a study. *International Journal of Computer Applications*, 179(46), pp.49-53. (Khawas and Shah, 2018)

Sengupta, S. and Bhattacharya, S., 2006, June. Formalization of UML use case diagram-a Z notation based approach. In *2006 International Conference on Computing & Informatics* (pp. 1-6). IEEE.

Diagrams?, W., 2020. *What Are The Advantages Of Use Case Diagrams?*. [online] Uml-questions.blogspot.com. Available at: <<http://uml-questions.blogspot.com/p/what-are-advantages-of-use-case-diagrams.html>> [Accessed 13 July 2020].

Torre, D., Labiche, Y., Genero, M., Baldassarre, M.T. and Elaasar, M., 2018, May. UML diagram synthesis techniques: a systematic mapping study. In *Proceedings of the 10th International Workshop on Modelling in Software Engineering* (pp. 33-40).

Septian, I., Alianto, R.S. and Gaol, F.L., 2017. Automated test case generation from UML activity diagram and sequence diagram using depth first search algorithm. *Procedia computer science*, 116, pp.629-637.

Friesen, M., 2018. *List Of Advantages Of UML*. <https://www.techwalla.com/articles/list-of-advantages-of-uml>.

Bogdanović, Z., Barać, D., Jovanić, B., Popović, S. and Radenković, B. (2013). Evaluation of mobile assessment in a learning management system. *British Journal of Educational Technology*, 45(2), pp.231-244.