

Mixipe – Finding new cooking recipes

Final Project Report

TU857

BSc in Computer Science (Infrastructure)

**Sasha Kuechenmeister**

**C18404082**

**Anh Doung Trinh**

School of Computer Science

Technological University Dublin

**30/12/2021**

Abstract

Mixipe is a cooking recipe mobile application for users who want to find new recipes in a much easier, enjoyable, and addicting way through the use of a swiping-mechanism that is used in dating apps such as Tinder.

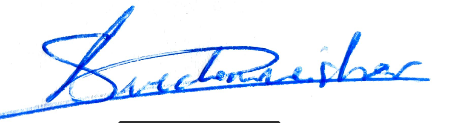
Mixipe is useful for when users want to cook something new but cannot be bothered to google aimlessly for new recipes, rather users can just open Mixipe and start swiping away. Pictures and names of recipes will be shown to users, which they can then like or dislike. All the liked recipes are then saved to their liked recipes, which they can then look at a later time. Users can also see recipes in more detail by simply tapping on one, and see all the necessary information (e.g. instructions, ingredients, cooking time, serving size, etc.)

Mixipe is being developed as a solution to Millennials least favourite process of cooking – finding recipes. A study suggests that roughly a third of all Millennials that cook, say that looking for what to cook is their least liked process, as they find it very boring and tedious. Therefore, the objective of this project is to make the process of looking for new recipes something to look forward to and enjoy.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:



Sasha Kuechenmeister

30/12/2021

Acknowledgements

I would like to acknowledge and thank my supervisor, for their support and guidance throughout the past few months. Every meeting we had really helped me to stay on track of everything and hit my personally assigned deadlines. My supervisor also gave me some helpful tips, which I really appreciated.

I would also like to thank my friends for giving me ideas to incorporate or change, and thank my family for helping with the testing of my firebase login prototype.

Table of Contents

[Table of Figures 7](#_Toc97028628)

[Table of Tables 9](#_Toc97028629)

[1. Introduction 10](#_Toc97028630)

[1.1. Project Background 10](#_Toc97028631)

[1.2. Project Description 11](#_Toc97028632)

[1.3. Project Aims and Objectives 13](#_Toc97028633)

[1.4. Project Scope 14](#_Toc97028634)

[1.5. Thesis Roadmap 14](#_Toc97028635)

[*1.5.1 Research* 14](#_Toc97028636)

[*1.5.2 Design* 14](#_Toc97028637)

[*1.5.3 Development* 14](#_Toc97028638)

[*1.5.4 Testing and Evaluation* 15](#_Toc97028639)

[*1.5.5 Redevelopment* 15](#_Toc97028640)

[*1.5.6 Conclusions and Future Work* 15](#_Toc97028641)

[2. Literature Review 16](#_Toc97028642)

[2.1. Introduction 16](#_Toc97028643)

[2.2. Alternative Existing Solutions 16](#_Toc97028644)

[*2.2.1 Yummly* 16](#_Toc97028645)

[*2.2.1 SuperCook* 17](#_Toc97028646)

[*2.2.1 ShuffleChef* 19](#_Toc97028647)

[2.3. Technologies Researched 20](#_Toc97028648)

[*2.3.1 Platform Selection* 20](#_Toc97028649)

[*2.3.3 Databases* 22](#_Toc97028650)

[*2.3.4 Application Programming Interfaces* 23](#_Toc97028651)

[2.4. Other Research 24](#_Toc97028652)

[*2.4.1 Usability and Accessibility* 24](#_Toc97028653)

[*2.4.2 Programming Languages* 25](#_Toc97028654)

[2.5. Existing Final Year Projects 26](#_Toc97028655)

[*2.5.1 Boppable, Emmet Doyle* 26](#_Toc97028656)

[*2.5.2 FoodSwipe, Eoin Lynch* 27](#_Toc97028657)

[*2.5.3 Similarities to Mixipe* 27](#_Toc97028658)

[2.6. Conclusions 28](#_Toc97028659)

[3. Prototype Design 29](#_Toc97028660)

[3.1 Introduction 29](#_Toc97028661)

[3.2. Software Methodology 29](#_Toc97028662)

[*3.2.1 Waterfall Methodology* 29](#_Toc97028663)

[*3.2.2 Agile Methodology* 30](#_Toc97028664)

[*3.2.3 Spiral Methodology* 30](#_Toc97028665)

[*3.2.3 Conclusions* 31](#_Toc97028666)

[3.3. Overview of System 32](#_Toc97028667)

[3.4. Requirements Table 33](#_Toc97028668)

[3.5. Front End 34](#_Toc97028669)

[*3.5.1 Introduction* 34](#_Toc97028670)

[*3.5.2 Low Fidelity Prototype* 34](#_Toc97028671)

[*3.5.3 Medium Fidelity Prototype* 35](#_Toc97028672)

[*3.5.4 Use-case Diagrams* 36](#_Toc97028673)

[*3.5.5 User Scenarios* 39](#_Toc97028674)

[3.6. Middle Tier 40](#_Toc97028675)

[3.7. Back End 40](#_Toc97028676)

[3.8. Conclusions 41](#_Toc97028677)

[4. Prototype Development 42](#_Toc97028678)

[4.1. Introduction 42](#_Toc97028679)

[4.2. Prototype Development 42](#_Toc97028680)

[*4.2.1 Firebase* 42](#_Toc97028681)

[*4.2.2 Spoonacular API* 43](#_Toc97028682)

[4.3. Front End 44](#_Toc97028683)

[*4.3.1 Registration page* 44](#_Toc97028684)

[*4.3.2 Login Page* 46](#_Toc97028685)

[*4.3.3 Bottom Navigation Bar* 48](#_Toc97028686)

[*4.3.4 Swipe Page* 50](#_Toc97028687)

[*4.3.5 Search Page* 52](#_Toc97028688)

[*4.3.6 Liked Page* 55](#_Toc97028689)

[4.4. Middle Tier 56](#_Toc97028690)

[*4.4.1 Registration Class* 56](#_Toc97028691)

[*4.4.1 Login Class* 58](#_Toc97028692)

[*4.4.1 Bottom Navigation* 60](#_Toc97028693)

[*4.4.1 Swipe Class* 61](#_Toc97028694)

[*4.4.1 Search Class* 62](#_Toc97028695)

[*4.4.1 Liked Class* 63](#_Toc97028696)

[4.5. Back-End 64](#_Toc97028697)

[*4.5.1 Spoonacular API* 64](#_Toc97028698)

[*4.5.2 Firebase* 65](#_Toc97028699)

[5. Testing and Evaluation 66](#_Toc97028700)

[5.1. Introduction 66](#_Toc97028701)

[5.2. Plan for Testing 66](#_Toc97028702)

[*5.2.1 Test Plan Table* 67](#_Toc97028703)

[5.3. Plan for Evaluation 68](#_Toc97028704)

[5.4. Conclusions 68](#_Toc97028705)

[6. Issues and Future Work 69](#_Toc97028706)

[6.1. Introduction 69](#_Toc97028707)

[6.2. Issues and Risks 69](#_Toc97028708)

[6.3. Plans and Future Work 70](#_Toc97028709)

[6.3.1. GANTT Chart 70](#_Toc97028710)

[Bibliography 71](#_Toc97028711)

# Table of Figures

[Figure 1 – projected 1 billion additional smartphone users within next 5 years 9](#_Toc97028443)

[Figure 2 – Swipe Prototype 11](#_Toc97028444)

[Figure 3 – Search Prototype 11](#_Toc97028445)

[Figure 4 – Yummly mobile app screenshots 16](#_Toc97028446)

[Figure 5 – SuperCook website screenshot 17](#_Toc97028447)

[Figure 6 – SuperCook mobile app screenshots 17](#_Toc97028448)

[Figure 7 – ShuffleChef mobile app screenshots 18](#_Toc97028449)

[Figure 8 – Mobile (green), Desktop (blue) & Tablet (purple) market share over the past year 19](#_Toc97028450)

[Figure 9 – Mobile operating system market share 20](#_Toc97028451)

[Figure 10 – How DVM differentiates from JVM 24](#_Toc97028452)

[Figure 11 – XML vs HTML code snippet 25](#_Toc97028453)

[Figure 12 – Waterfall Methodology 28](#_Toc97028454)

[Figure 13 – Agile Methodology 29](#_Toc97028455)

[Figure 14 – Spiral Methodology 30](#_Toc97028456)

[Figure 15 – Technical Architecture 31](#_Toc97028457)

[Figure 16 – Low Fidelity Prototypes 33](#_Toc97028458)

[Figure 17 – Low Fidelity Prototypes 34](#_Toc97028459)

[Figure 18 – Medium Fidelity Prototype 34](#_Toc97028460)

[Figure 19 – Use Case 1st Iteration 35](#_Toc97028461)

[Figure 20 – Use Case 2nd Iteration 36](#_Toc97028462)

[Figure 21 – Use Case 3rd Iteration 37](#_Toc97028463)

[Figure 22 – ERD Diagram 39](#_Toc97028464)

[Figure 23 – Firebase Authentication 42](#_Toc97028465)

[Figure 24 – Registration Page 43](#_Toc97028466)

[Figure 25 44](#_Toc97028467)

[Figure 26 – Login Page 45](#_Toc97028468)

[Figure 27 46](#_Toc97028469)

[Figure 28 47](#_Toc97028470)

[Figure 29 48](#_Toc97028471)

[Figure 30 48](#_Toc97028472)

[Figure 31 48](#_Toc97028473)

[Figure 32 49](#_Toc97028474)

[Figure 33 50](#_Toc97028475)

[Figure 34 51](#_Toc97028476)

[Figure 35 52](#_Toc97028477)

[Figure 36 53](#_Toc97028478)

[Figure 37 53](#_Toc97028479)

[Figure 38 – Registration Page Error Checking 55](#_Toc97028480)

[Figure 39 – Registers user in Firebase 55](#_Toc97028481)

[Figure 40 – Firebase Console | Authentication 56](#_Toc97028482)

[Figure 41 – Switch to Login Page 56](#_Toc97028483)

[Figure 42 – Login Page Error Checking 57](#_Toc97028484)

[Figure 43 – Login authentication 57](#_Toc97028485)

[Figure 44 – Switches to Registration Page 58](#_Toc97028486)

[Figure 45 – Forgot Password function 58](#_Toc97028487)

[Figure 46 59](#_Toc97028488)

[Figure 47 60](#_Toc97028489)

[Figure 48 61](#_Toc97028490)

[Figure 49 62](#_Toc97028491)

[Figure 50 - Gantt Chart 67](#_Toc97028492)

# Table of Tables

[Table 1 – Requirements 32](#_Toc92107442)

[Table 2 – Test Plan 50](#_Toc92107443)

# 1. Introduction

## Project Background

Taking advantage of technology on a day-to-day basis has become so common that the majority of humanity now relies on it. Roughly 6.4 billion people have access to a smartphone, which equates to about 80% of the world population (1). And this is projected to grow more and more each year.

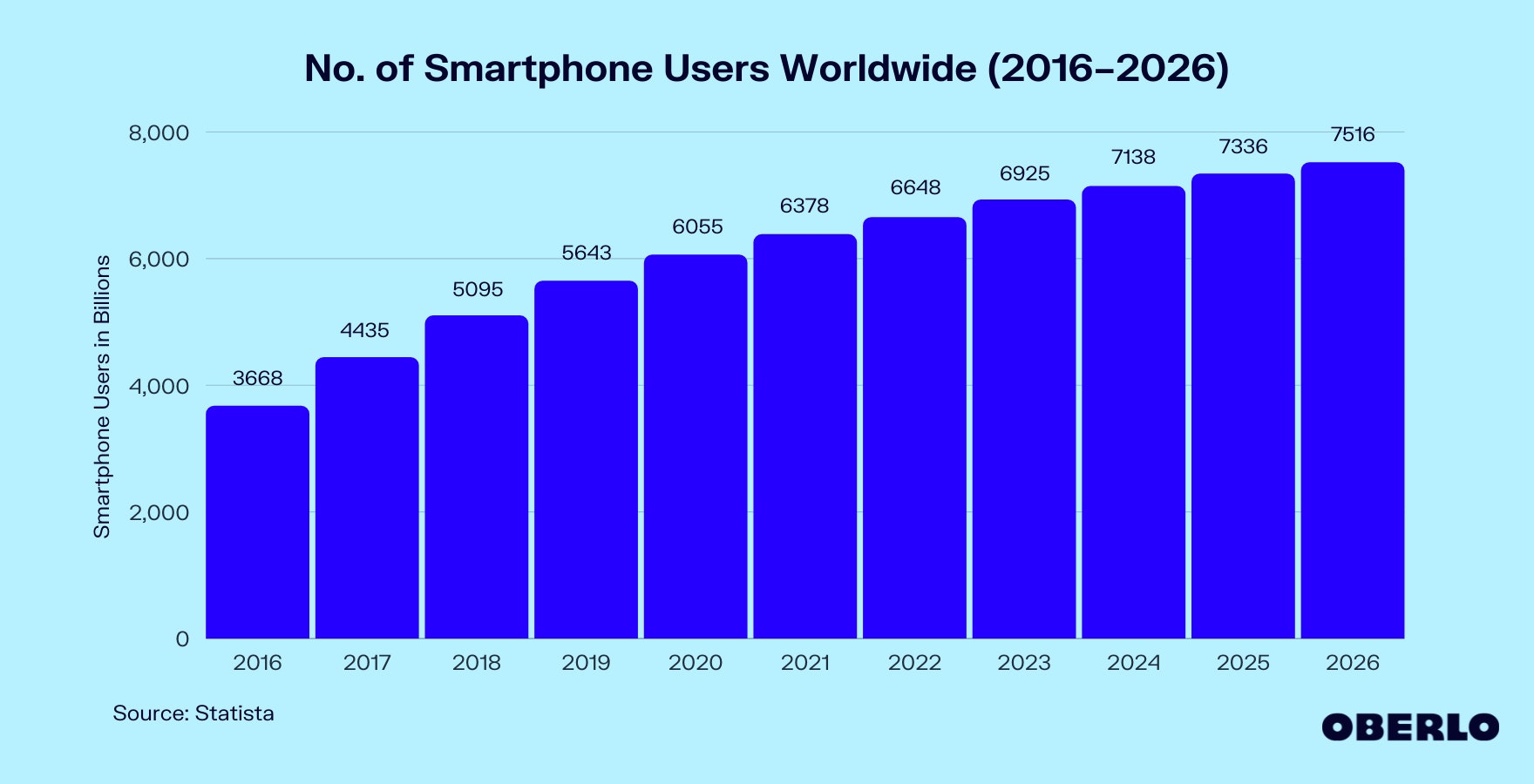


Figure 1 – projected 1 billion additional smartphone users within next 5 years

Each of these people uses their smartphone for everyday activities, ranging from online shopping to checking the weather. Technology has progressed so much that the average user spends nearly 3 hours on their smartphone every day (2). One example for which people use their smartphones is for cooking.

It is said that more than 70% of adults have “gone digital” when it comes to cooking (3). More and more people are turning to online recipes and recipe apps rather than traditional cookbooks.

While people over the age of 35 tend to print out a recipe they find online, ca. 59% of 25- to 34-year-olds use either their smartphone or tablet when cooking. Thanks to technology, younger generations are starting to learn how to cook online. However, one thing that younger generations find tedious is the search for what they want to cook. A study suggests that about a third of Millennials say that finding what to cook is their least enjoyable part of the cooking process (4).

There are many apps that try to solve this issue through recommending recipes based on a number of factors, however, there are near to no mobile applications that truly make it easier and a lot more fun. For that main reason, *Mixipe* is being developed – to make finding recipes less tedious.

## Project Description

*Mixipe* is a cooking recipe mobile application for users who want to find new recipes in a much easier, enjoyable, and fun way through the use of a swiping-mechanism that is used in dating apps such as Tinder.

*Mixipe* is launched by tapping the app icon on the user’s home screen or app drawer. The first time when a user opens up the app after installation, they must set up/create an account with a valid email address and password. If a user is returning to the app, they will be asked to enter their login details. After the user is logged in, they will be presented with the layout of the *Mixipe* app. At the bottom of the app there will be a toolbar which users will use to navigate to their liked recipes, the swiping feature, or the search page. The page which the user is on will be denoted on the taskbar by the colour surrounding the selected page.

The swiping feature is the default page which the user will see after logging in. Recipes will be viewed on cards which will contain an image of the recipe and its name. Users can swipe left or press the dislike button below the card if they do not like the recipe. If the user likes the recipe and wishes to save it, they can swipe right or press the like button below the recipe card. Recipes that the user liked will be added to their saved list where they can view it at a later time. If a user wants to know more information about the recipe shown, they can click onto the card which will show them additional data such as the ingredients used, instructions, cooking time and the number of servings it is for.

The saved list page is accessible via the toolbar at the bottom of the app. This will show all the user’s previously liked/saved recipes. Again, if the user wants to see more information about the recipe, they can just click on it. In case a user decides they no longer like the recipe, they have the option to delete it.

If a user doesn’t want to use the random generate swiping feature, they can navigate to the search page. Here, the user has the option to scroll through or for search recipes save any that they like.



Figure 2 – Swipe Prototype

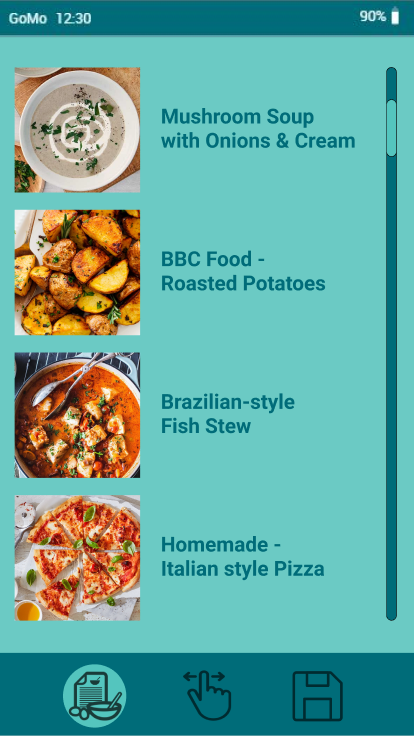


Figure 3 – Search Prototype

The *Mixipe* mobile application will employ the use of a Recipe API, Spoonacular API. This API will retrieve the recipe data (e.g., title, image, ingredients, cooking times, etc.). This is a lot more ideal than storing all of this data in the backend, which would be more costly and time-consuming.

## Project Aims and Objectives

As explained above, the overall aim of this project is quite simple – to design, document and develop a mobile application for discovering new recipes. The general aim is to make searching for recipes a lot more enjoyable. As I mentioned earlier, the younger generation finds that searching for a recipe is the only thing that they do not enjoy in the cooking process, as they find it often to be a long and boring process. This is why Mixipe is being developed, to make finding recipes quick, easy, and fun. For this to become a reality some aims, and objectives have been set below.

1. *Mixipe* is required to be a simple and intuitive design. Users of the mobile application should be able to use it naturally without needing to ask someone for help. The software must be fast and responsive so that the swiping feature flows nicely, so that users can browse through multiple recipes in a short time.
2. *Mixipe* must allow the user to create an account using a valid email address and secure password. Afterwards, users must be able to log in to the mobile app in order to save their liked recipes.
3. *Mixipe* requires being able to select a recipe at random and present them to the user on cards which they can interact with via the swipe page. It must have the functionality that users can like or dislike the shown recipe either by swiping left/right or pressing the like/dislike buttons.
4. *Mixipe* users must be able to view their liked recipes on the liked page so that they can use the recipe at a later time. Users need to have the option to delete any of their liked recipes if they no longer want them.
5. *Mixipe* must allow users to click on a shown recipe card in order for them to get more detailed information about the recipe. The user must be shown the instructions on how to cook their selected recipe, the amount of people it can serve, and what ingredients are needed.
6. If the user no longer wants to use the swiping feature, *Mixipe* is required to have an option where users can view all the recipes in a list view. Users should be able to scroll through or search this list and click on any recipe if they want more detailed information. If a user likes one of the recipes, they need to be able to add it to their liked recipes.

## Project Scope

The scope of this project is mainly focused on providing users with a mobile app that can make searching for recipes fun. Recipes will be shown to *Mixipe* users via cards, which users can swipe on to like or dislike them. Users liked recipes will be added to their liked list where they can view them at a later time if they have created an account. Disliked recipes will not be shown to them for a set period of time. Users can see more detailed information (instructions, ingredients, etc.) about each recipe by simply clicking on the shown card. Users of *Mixipe* will also be able to view all the recipes in a list view, which they can scroll through and select recipes they wish to save or just look at in more detail.

## Thesis Roadmap

This section will provide a summary of each of the chapters covered in this report.

### 1.5.1 Research

The literature review chapter explores into the background research related to alternative ways of recommending recipes to people and making the overall search for a new recipe a lot more enjoyable. It will also look into some similar systems and technologies used by others, other research, and past final year projects

### 1.5.2 Design

The prototype design chapter discusses the methods adopted for this project, as well as how those decisions came to be. Following that, the intended systems use-cases will be presented. Finally, the intended technical architecture will be discussed.

### 1.5.3 Development

The development chapter breaks down the entire system development process in terms of the technological architecture outlined in this chapter. Further, some difficulties experienced during the development process will also be discussed.

### 1.5.4 Testing and Evaluation

The testing and evaluation chapter outlines how the system's testing and evaluation were conducted. Each stage of testing will be detailed, followed by a detailed summary of all user comments obtained during user evaluation trials. Finally, the system will be examined to see if it followed Nielsen's User Interface Design Heuristics.

### 1.5.5 Redevelopment

The redevelopment chapter presents some of the development actions that were made as a result of the given user feedback. The adjustments that have been made, as well as the significance of these changes, will also be examined.

### 1.5.6 Conclusions and Future Work

The conclusions and future work chapter reflects on the entire project and will focus on the conclusions drawn, thoughts on what worked well and what did not, as well as what work is planned for this project in the future.

# 2. Literature Review

## 2.1. Introduction

This chapter examines the existing and possible technologies for use in the development of *Mixipe*. This is crucial to determine what capabilities are currently accessible in the market and how a product like Mixipe may differentiate itself from the competition.

The first section of the literature review looks at other existing applications that compete with *Mixipe* in the same market. It also examines how similar programs differ from *Mixipe* and what functionality they lack.

The next section looks into the research conducted into the technologies that are available. This research is important to help determine which technology should be implemented into *Mixipe*.

The final section of the research is used to discuss any other similar studies that have been done on this topic, including other final year projects.

## 2.2. Alternative Existing Solutions

Many of the apps that were examined were very useful for gathering requirements and deciding on how to approach this project. It provided valuable insight into how *Mixipe’s* mobile application could be designed. Each of the examined systems perform their role well and provide a sufficient number of features for users. However, surplus of features could also be a disadvantage, as it diverts attention away from the core element of making finding a recipe an enjoyable process of cooking.

### 2.2.1 Yummly

Yummly is a mobile app and website that provides users with recipe recommendations personalized to their individual taste, semantic recipe search, a digital recipe box, shopping list and one-hour grocery delivery (5). Yummly is available on iOS, Android, and web browsers and has 23 million users combined.

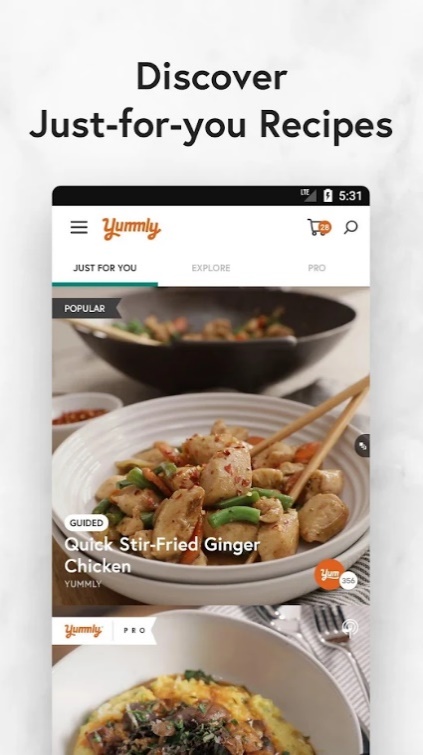
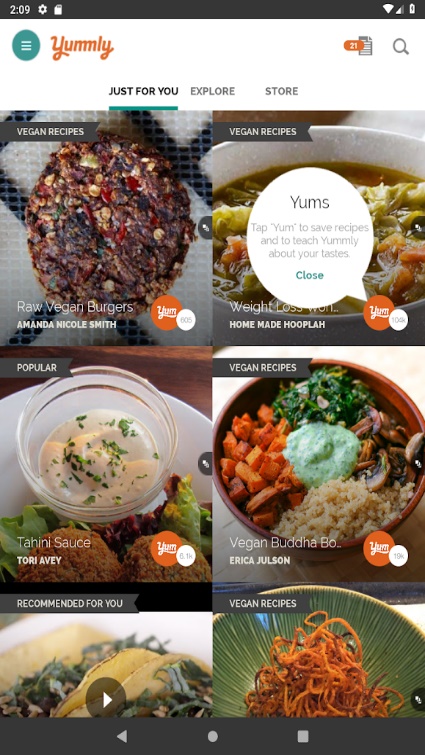
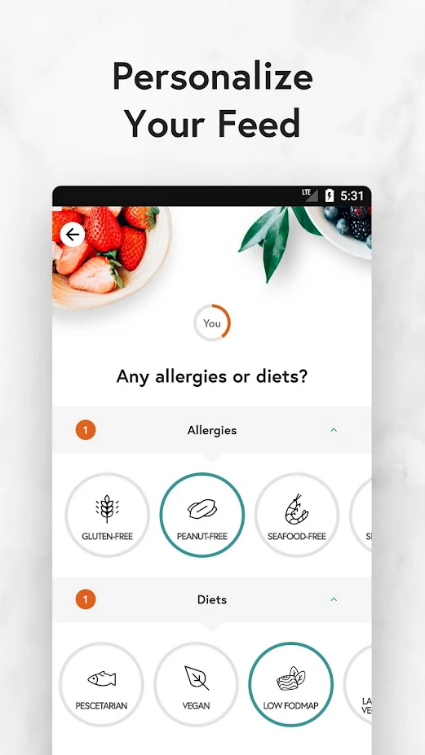
  

Figure 4 – Yummly mobile app screenshots

Advantages:

Yummly is a free to use application which features a clean and simple interface which is easy to understand. Users can also easily switch between imperial and metric measures.

Disadvantages:

The main limitation of Yummly is that it still lacks the “fun” aspect of looking for new recipes. If a new user joins Yummly and their first 5 likes are all Italian dishes, they will only be recommended other Italian recipes. This makes the app quite useless if a user is looking to find and explore new, different cuisines.

### 2.2.1 SuperCook

SuperCook is a mobile app and website search engine that lets users find recipes based on what ingredients they have at home. Users can find thousands of recipes that they can make straight away without having to go shopping. It is available on both iOS and Android, as well as via web browser.

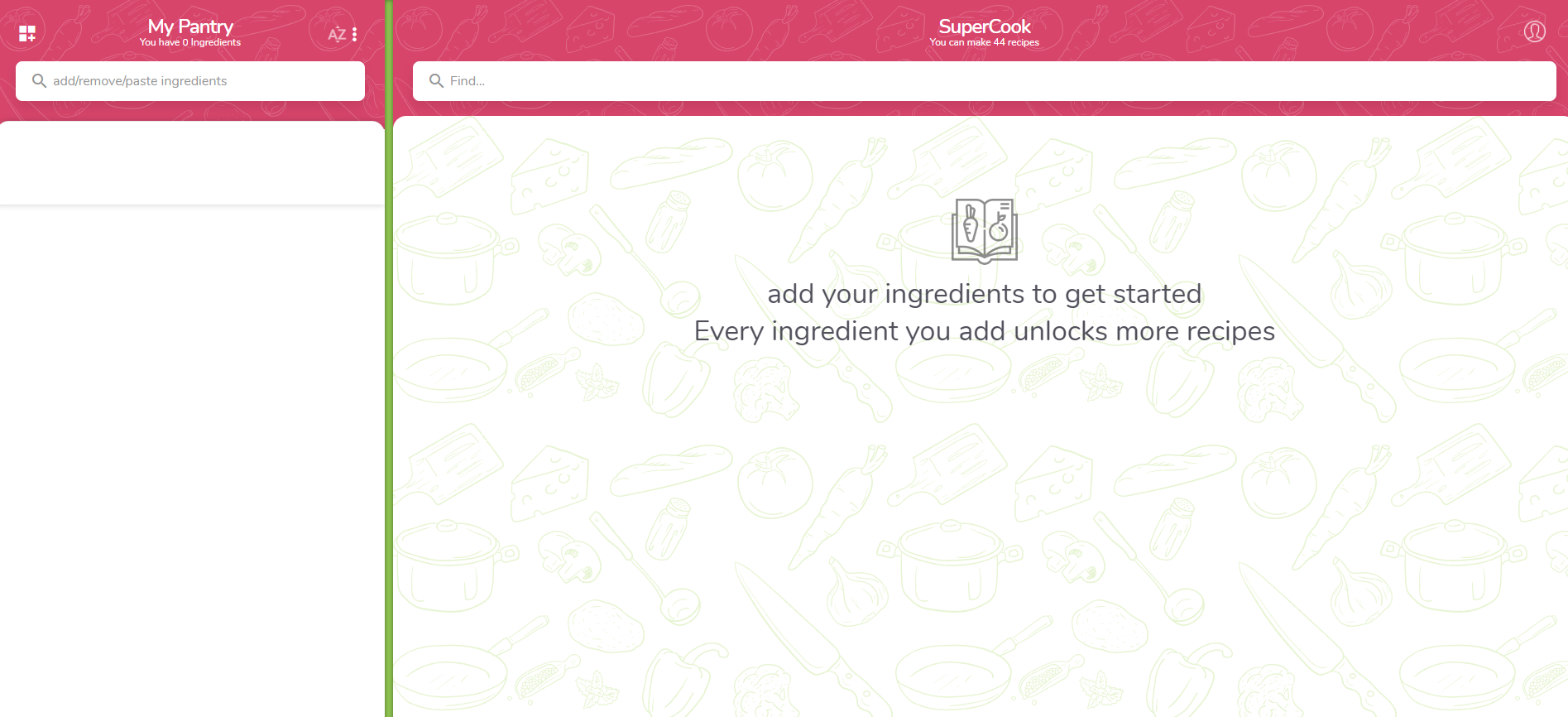


Figure 5 – SuperCook website screenshot

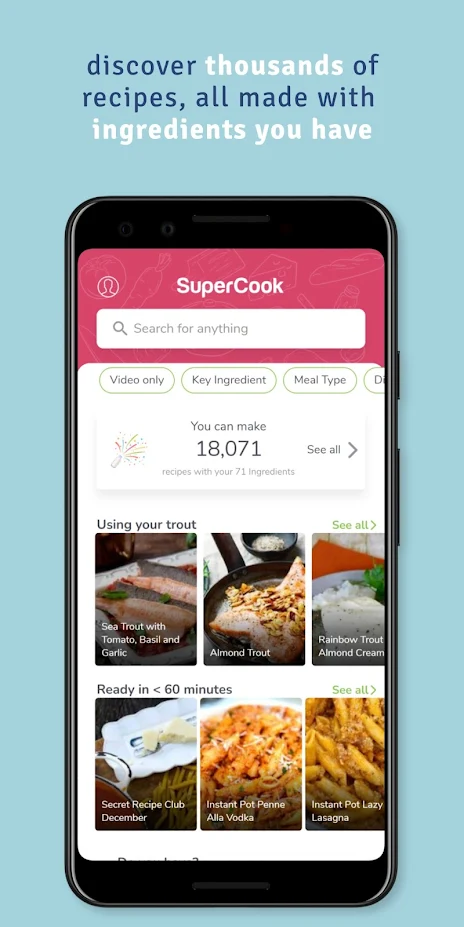
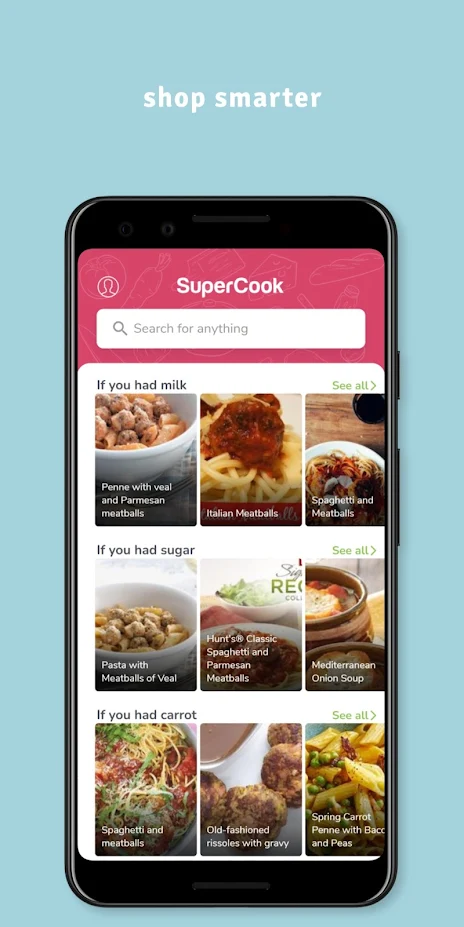
  

Figure 6 – SuperCook mobile app screenshots

Advantages:

SuperCook is great for user who want to find new recipes based on the ingredients that they have at home. It is a nice way to find recipes that you never thought to search for. Users can also exclude certain ingredient categories, which can be very beneficial to users with allergies.

Disadvantages:

Its main feature is also one of its biggest drawbacks, as it is limited by only showing recipes based on ingredients the user has. For example, if a user goes on holiday, they will not have any food at “home” and therefore this app becomes completely useless. Furthermore, SuperCook’s graphical user interface is cluttered. Another disadvantage of this application is that there is a lot of repetitive information, a lot of the time the same recipes are recommended just with different instructions, this can make it hard for a user to decide which recipe to choose.

### 2.2.1 ShuffleChef

ShuffleChef is a breakfast, lunch and dinner meal planning mobile application. It uses a swiping feature to discover new recipes. Users can also find recipes by searching for a single ingredient. ShuffleChef is only available on Android.

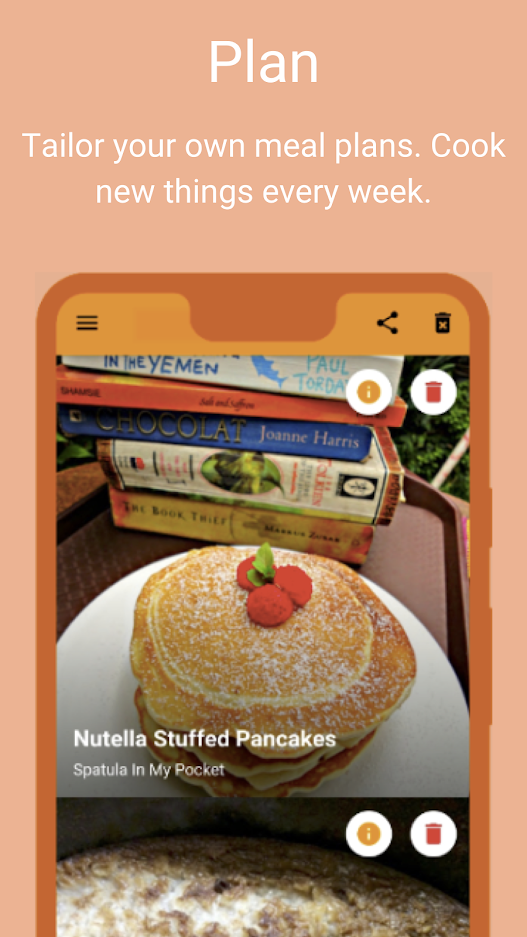
*  *

Figure 7 – ShuffleChef mobile app screenshots

Advantages:

ShuffleChef adds the “fun” element into searching for new recipes through their swiping feature. The graphical user interface is clean and simple.

Disadvantages:

ShuffleChef users cannot view any detailed information (ingredients, instructions, etc.) in the mobile app, which can be frustrating for allergists as they cannot quickly find out what ingredients are used in the shown recipe. If users want more information on a recipe, they first need to like it, then the recipe gets added to the “shuffle tab” where users can search liked recipes by a single ingredient, and then you can finally view it in your “meal plan” tab where you will be redirected to a web browser of the recipe. Therefore, the app feels quite counterintuitive.

## 2.3. Technologies Researched

This section will explain the various kinds of technologies that have been researched for the *Mixipe* System. This includes looking at platform selection, mobile technologies, databases, and API’s.

### 2.3.1 Platform Selection

Smartphones have the largest market share of 55.35% compared to desktops 44.65% and Tablets 2.48% share in the market (6). As smartphones get more advanced each year, the dynamic of internet use is continuingly shifting more from PCs to mobile devices, as seen in figure 9. People also tend to have easier access to a mobile device than a PC. All of these factors were considered when choosing the platform that could attract the largest number of users to *Mixipe*.

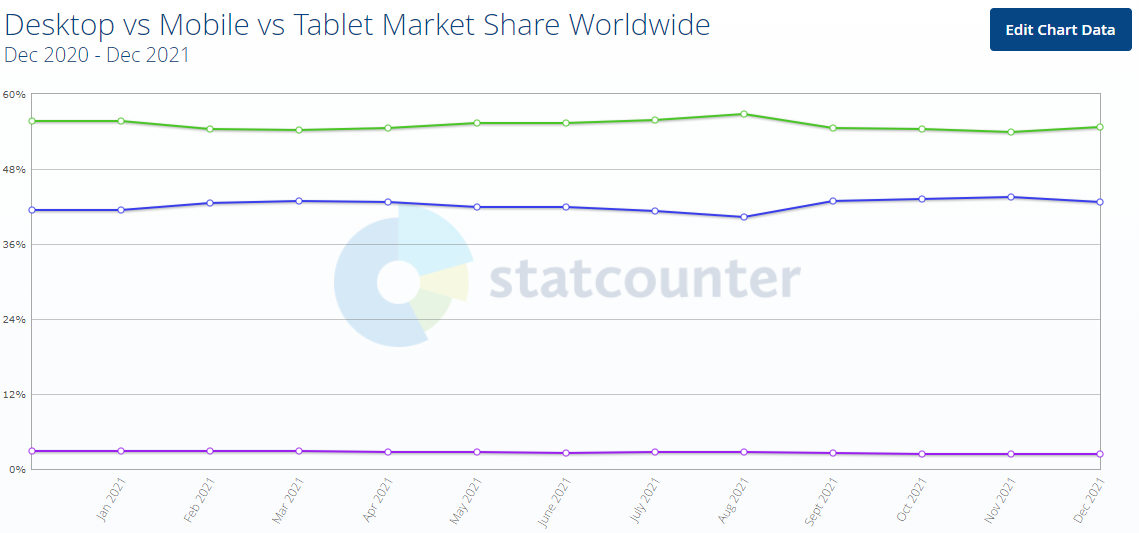


Figure 8 – Mobile (green), Desktop (blue) & Tablet (purple) market share over the past year

2.3.2 Mobile Technologies

When it came to making the decision to choose which mobile platform to develop on, there were two main choices. One choice was to develop *Mixipe* on iOS, which is Apple’s mobile operating system. iOS’s devices make up roughly 28% of the market share (7) which is a considerably large number given the amount of smartphone users. Applications on iOS are developed using XCode through the Swift, platform and developers must pay a fee of $99 annually (8).

The second option was to develop on Android OS, developed by Google. This operating system is based on a modified version of Linux and other open-source software. As seen in figure 10, Android makes up for over 71% of the mobile operating system market share. Android applications are developed using either Java or Kotlin. For developers to create Android applications and publish it to its official marketplace “Google Play Store”, they are required to only pay a one-time fee of $25 (9).

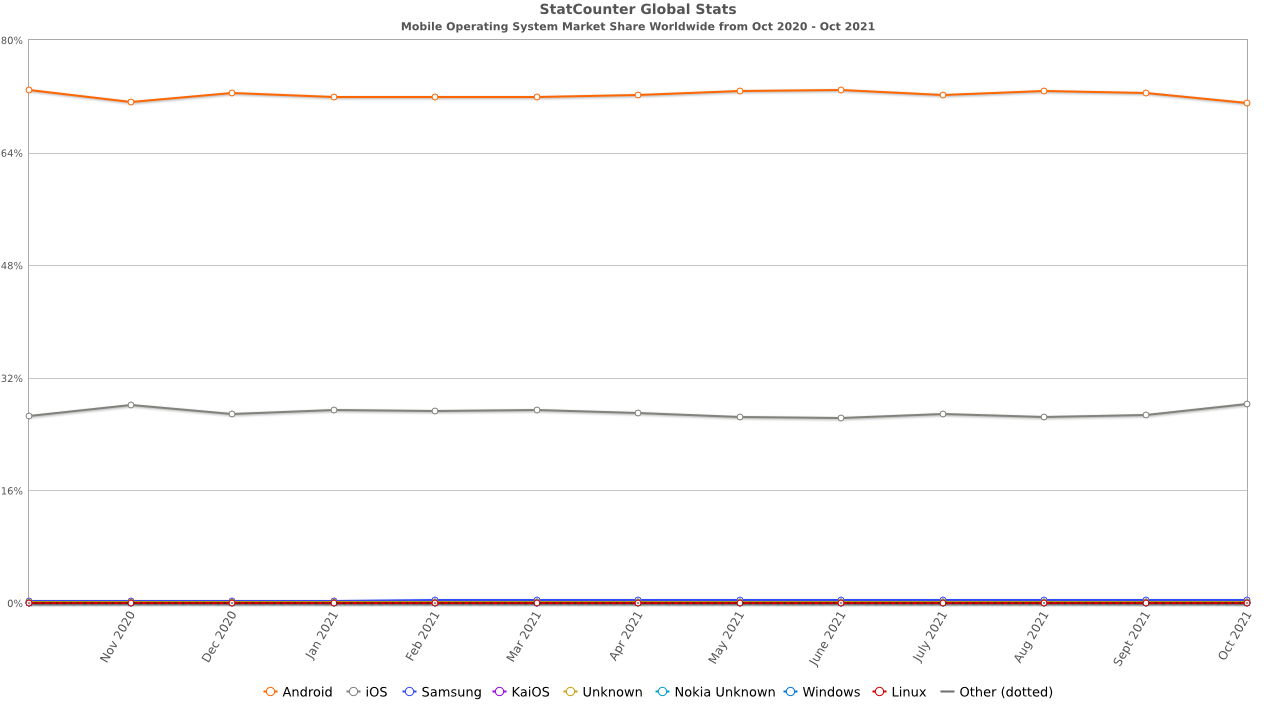


Figure 9 – Mobile operating system market share

From researching both of these options, it was clear that Android OS is the best to develop *Mixipe* on. Android Studio will be used for the development as it is the official integrated development environment for Android OS (10). The language used for the development will be Java.

### 2.3.3 Databases

While developing *Mixipe*, there were three main database solutions that were considered to be implemented. The three databases were Firebase, Realm and SQLite. Detailed below is which database was chosen and why.

1. **Realm** is an open-source database management system. It is available for Swift, C, Java, Kotlin, C# and JavaScript and can be implemented for mobile operating systems (iOS/Android) as well as Windows. Realm offers cross-platform integration, meaning users can use a single database to build all their apps for any of the named platforms. It also makes it simple to keep data in sync across all users, devices, and backend in real-time. Realm is a simple, lightweight database that uses memory, disk space and battery life efficiently (11).
2. **SQLite** is an open-source, C library which implements a serverless, self-contained, transactional SQL database engine. Unlike most other database management systems, SQLite is an embedded database management system and therefore has no separate server process (12). This makes it fast and easy to access important information. SQLite is a simple and lightweight database.
3. **Firebase** is a real-time database management system which is backed by Google (13). Firebase aims to solve three main issues for developers: build apps fast, release and monitor an app with confidence, and engage users. Being cloud-hosted, means that developers can easily perform scaling without any problems (14). This also means that Firebase does not require an internet connection when users enter data, instead Firebase saves the data and uploads in once users regain access to an internet connection.

After doing some thoughtful research into the three database systems, the chosen database for *Mixipe* was Firebase. There were a few reasons for this choice. Firstly, Firebase is easy to implement, and it is also available for platforms such as Android, iOS and Web browsers which would make future expansion a lot simpler (14). Another reason is that Firebases does not need a stable internet connection when users enter data, allowing *Mixipe* users to still use the application even when their internet connection is unstable. Firebase also has an automatic backup system, protecting any possible data loss on the platform (15).

### 2.3.4 Application Programming Interfaces

When developing *Mixipe*, it was necessary to have an API for importing all the Recipes to the app. There are many APIs available to choose from such as “Webknox Recipe”, “Yummly”, “Spoonacular”, and more. Below I will examine each of them in more detail and which API I chose and why.

1. **Webknox Recipe** is an application programming interface that allows users to use simple, plain language to search for recipes, such as “lactose-free pancakes”. This API also offers a lot of information on nutrients, pricing, tips, and more, as well as many widgets to show ingredients lists and price breakdowns. Webknox Recipe has access to over 330,000 recipes (16).
2. **Yummly** as mentioned earlier is an application that gives users recipe recommendations. It also provides an API which offers the ability to incorporate recipes and faceted recipe search into websites or mobile apps. Yummly provides their recipe recommendation, which allows users to search for recipes via their food preferences (17).
3. **Spoonacular** similarly to Webknox Recipe allows users to use natural language to search for a recipe, such as “low-fat pancakes”. This API also automatically analyses recipes to check for ingredients that may contain common allergens, such as wheat, dairy, soy, etc. It also allows users to see if a recipe is vegan, vegetarian, paleo friendly, and more (18). Spoonacular has access to over 360,000 recipes and includes an open-source recipe database, meaning anyone can add their recipes to the list. Other features that this API provides are the ability to recommend recipes, find random recipes, extract ingredients from the recipe instruction steps, and visualize recipe nutritional data (19).

After looking into the different APIs and determining the best option for *Mixipe*, it was decided to use the Spoonacular API. Firstly, Spoonacular was chosen as it offers the largest recipe database out of all the researched APIs. Another reason this choice was made is that it offers users to search for recipes in a natural language, making it a lot more user-friendly. Spoonacular’s large list of features as mentioned above is another reason why this API sets itself apart from the others. And finally, its automatic recipe analysis to check for common allergens and determine if a recipe is vegan, vegetarian, etc was deemed very important when deciding what API to use.

## 2.4. Other Research

### 2.4.1 Usability and Accessibility

**Nielsen’s Heuristics:** A set of ten principles for designing user interfaces. These concepts can be used to identify usability difficulties during the design phase. When evaluating these heuristics, a group of evaluators examines the interface and assesses how well it adheres to the ten criteria points. These heuristics are critical for a user-centric app since they can enhance the overall user experience.

The ten principles are (20):

1. Visibility of System Status
2. Match between System and Real World
3. User Controls and Freedom
4. Consistency and Standards
5. Error Prevention
6. Recognition rather than Recall
7. Flexibility and Efficiency of use
8. Aesthetic and Minimalist Design
9. Help Users Recognize, Diagnose, and Recover from Errors
10. Help and Documentation

**Accessibility:** When creating an app, accessibility is one of the most crucial factors to consider. It is critical to ensure that everybody who wants to use the app can, regardless of their disability. Roughly 2.2 billion people suffer from an impairment, therefore, it is critical to use the right design philosophy to ensure accessibility to all (21).

Some of the disabilities that need to be considered in the development are people with impaired vision, motor difficulties, cognitive or learning difficulties, as well as deafness or impaired hearing (22).

It is critical to create *Mixipe* to be accessible to persons with disabilities as its very likely that someone with a disability would use the app to discover new recipes.

### 2.4.2 Programming Languages

**Java:** Developers can choose to code in either Java or Kotlin when working in Android Studio. Java was chosen for this project as the developer had more experience with it than Kotlin. It is an object-oriented programming language and computing platform created by Sun Microsystems in 1995 (23). There are over 13 billion devices that run Java and more than 10 million java developers in the world (24).

Java code usually runs on the JVM (Java Virtual Machine), but for Android devices it runs on the DVM (Dalvik Virtual Machine), a continued process virtual machine optimized for Android mobile devices. It optimizes the VM for memory, battery, life, and performance. The way everything works is, first the .java source code is compiled into .class files, then all the class files are generated into a single .dex (Dalvik Executable) file, and finally packaged as an .apk (Android Package) file (25).

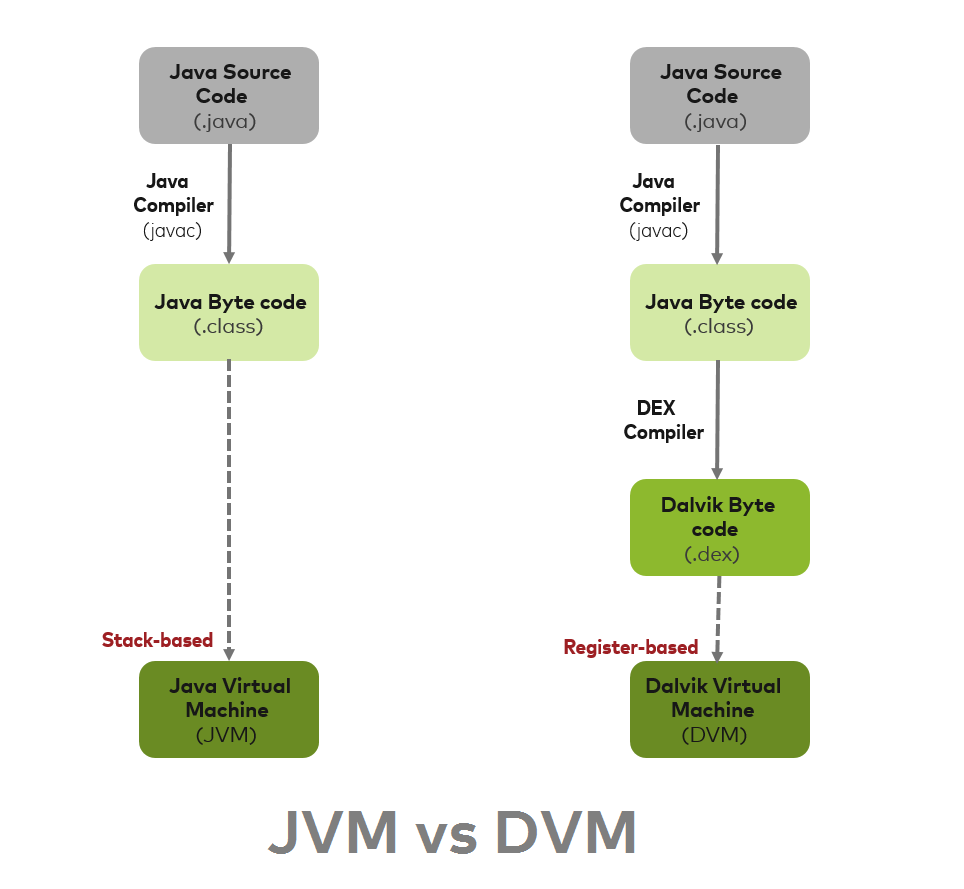


Figure 10 – How DVM differentiates from JVM

**XML:** XMLstands for Extensible Markup Language. It is a text-based markup language, which is a language that describes text in a digital document using a set of tags or code. Another text-based markup language is HTML (HyperText Markup Language) which is the most used markup language for web pages. While HTML was designed to display data and focus on how the data looks, XML was designed to carry data and focus on what data is. XML simplifies things, such as data sharing, data transport, platform changes and data availability (26). In Android, XML is used to implement UI-related data and because it’s a lightweight markup language, keeps file sizes small (27).

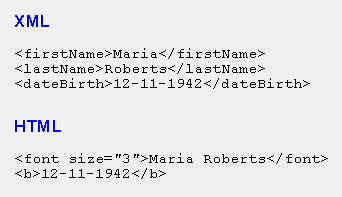


Figure 11 – XML vs HTML code snippet

## 2.5. Existing Final Year Projects

Two Final Year Projects were examined that were related to this project. Boppable by Emmet Doyle, and FoodSwipe by Eoin Lynch.

### 2.5.1 Boppable, Emmet Doyle

Boppable is a mobile application that was developed to provide attendants of parties or social events a means of having a say in what music gets played. Boppable is designed to have a host of a party or event set up a room that guests can join on their own separate device. Guests can then request songs to be played via Spotify. They can also upvote or downvote other requested songs, the more upvotes, the earlier the song gets played. Users can upvote / downvote through a swiping mechanism.

Technologies:

* Spotify Web API
* 3-tier architecture: Mobile App, Django middle tier, PostgreSQL database

Strengths of this project were having very few issues upon release and having a simple, intuitive User Interface.

One weakness of this project was that the host had the exact same control as guests, which in the testing phase led to some issues.

### 2.5.2 FoodSwipe, Eoin Lynch

FoodSwipe is a mobile application that was developed to tap into the food app market and offer users more features and functionality. FoodSwipe is designed to simplify the process of finding somewhere to eat when a user is undecided where to go. It allows users to swipe different restaurants to like or dislike them. Users can find the restaurants through an integrated map in the app. Furthermore, users have the option to favourite restaurants, as well as leave reviews.

Technologies:

* GeoFire library, Maps API, Places API
* 3-tier architecture: Android app, Web server, Firebase database

The strength of this project was that the main feature was achieved - displaying nearby restaurants to users.

A weakness of this project was that the app did not function as intended, such as, having a separate page for navigation rather than a small toolbar at the top or bottom of the page to make switching between features easier and smoother.

### 2.5.3 Similarities to Mixipe

Both of these projects contained some relevant aspects to the *Mixipe* project. Boppable and FoodSwipe both implement a swiping mechanism to like or dislike items shown to the user. They are both developed for mobile devices, FoodSwipe for Android, and Boppable for both Android and iOS. Both made use of Java and XML, with FoodSwipe also using the Firebase. These are also used for the development of *Mixipe*.

## 2.6. Conclusions

This chapter examined the project’s background research.

First, the various existing application were reviewed that were similar to *Mixipe*.

Then, research into the available technologies was carried out. This involved examining what platforms, various mobile technologies, and database options were available for this project.

Afterwards, other research such as Nielsen’s Heuristics and accessibility was looked into to ensure that *Mixipe* can be used by anyone regardless of any disabilities. The programming languages, Java and XML, that were used in this project were also outlined.

Finally, a look at past written final year projects was conducted. Looking at both projects “Boppable” and “FoodSwipe”, a lot of similarities were outlined. The research done for these two projects allowed for great insight into how *Mixipe* could be developed further.

# 3. Prototype Design

## 3.1 Introduction

In this chapter, the system design of *Mixipe* will be presented. The previous research from the previous chapter helped a lot in the design of this project. The first section will examine the software methodologies used and why they were chosen. Following that, various example use-cases will be presented. Finally, the system’s technological architecture will be outlined, which will go over in detail how the system design is intended to work. This will apply to both the front end and the back end of the system.

## 3.2. Software Methodology

### 3.2.1 Waterfall Methodology

The waterfall methodology, also known as the waterfall model, is a sequential development process that flows like a waterfall through all phases of the project (analysis, design, development, and testing), with each phase completely ending before moving on to the next (28).

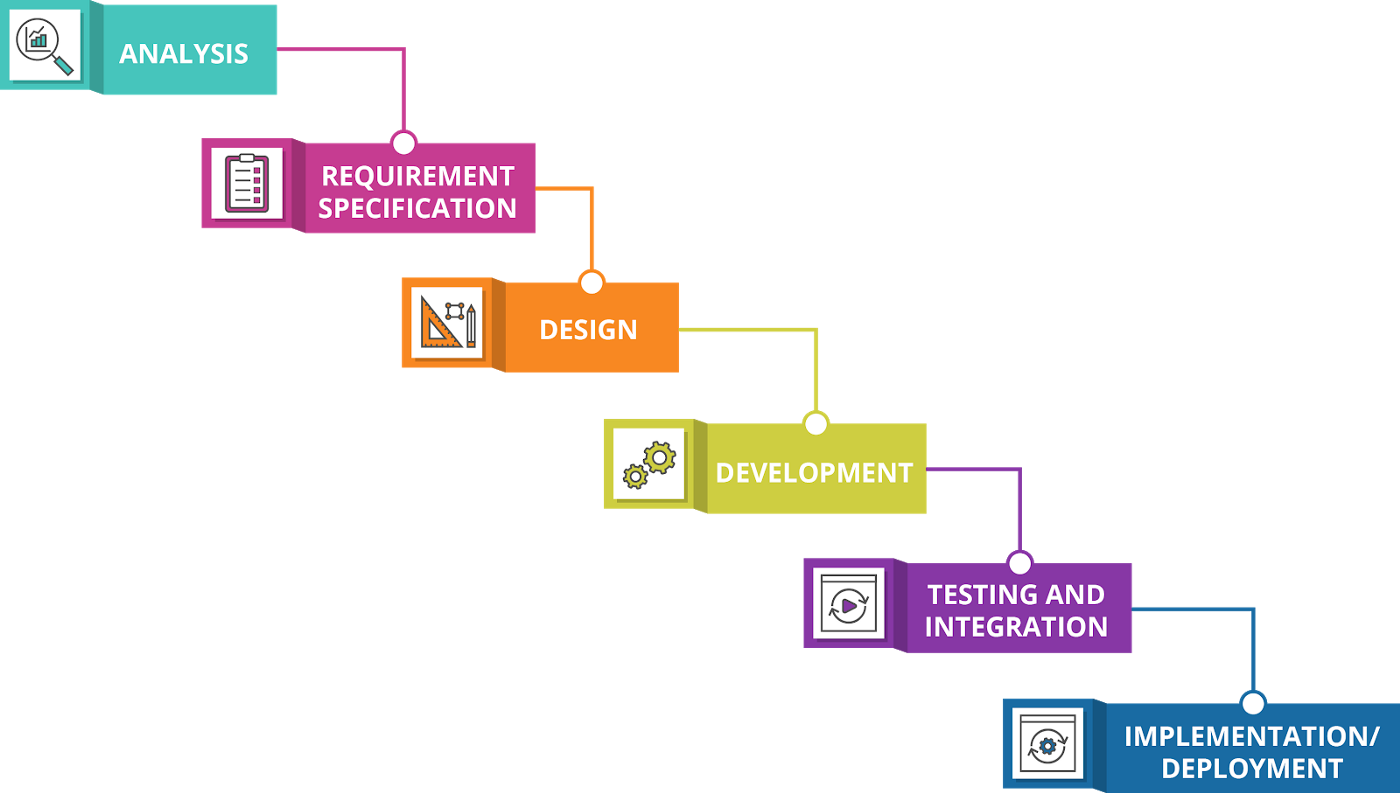


Figure 12 – Waterfall Methodology

The main benefit in using the waterfall methodology is its simplicity. It is also very easy to follow, allowing developers to easily get a grasp of their project’s scope. The main drawback of this methodology is its emphasis on upfront project and planning and commitment to predetermined progress. As a result, it is less adaptable later on. Any changes made later in the development process can be time-consuming, more difficult, and quite costly.

### 3.2.2 Agile Methodology

The Agile methodology is a way to manage a project by dividing it up into smaller phases or “sprints” (29). At the end of each sprint, the project’s priorities are assessed, and any adjustments that are judged essential are implemented.

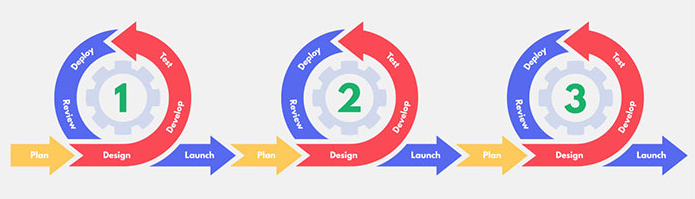


Figure 13 – Agile Methodology

Agile-based projects are best suited to applications that change a lot throughout the implementation. The main benefit in using this methodology is that because testing takes place at the end of each sprint, it makes it easier to detect any flaws that arise during development and reduces the amount of time spent debugging at the end of production. The main drawback of the Agile methodology is that it’s simple to go off track and start working on something completely different that’s not part of the project’s scope. As a result, it’s critical to keep track of progress and how it relates to the overall project.

### 3.2.3 Spiral Methodology

The Spiral Methodology, also known as spiral model, places its emphasis on risk analysis. The spiral model has four phases: Planning, Design, Construct, and Evaluation. A project repeatedly passes though these phases in iterations called spirals (30). The number of iterations varies for every project.

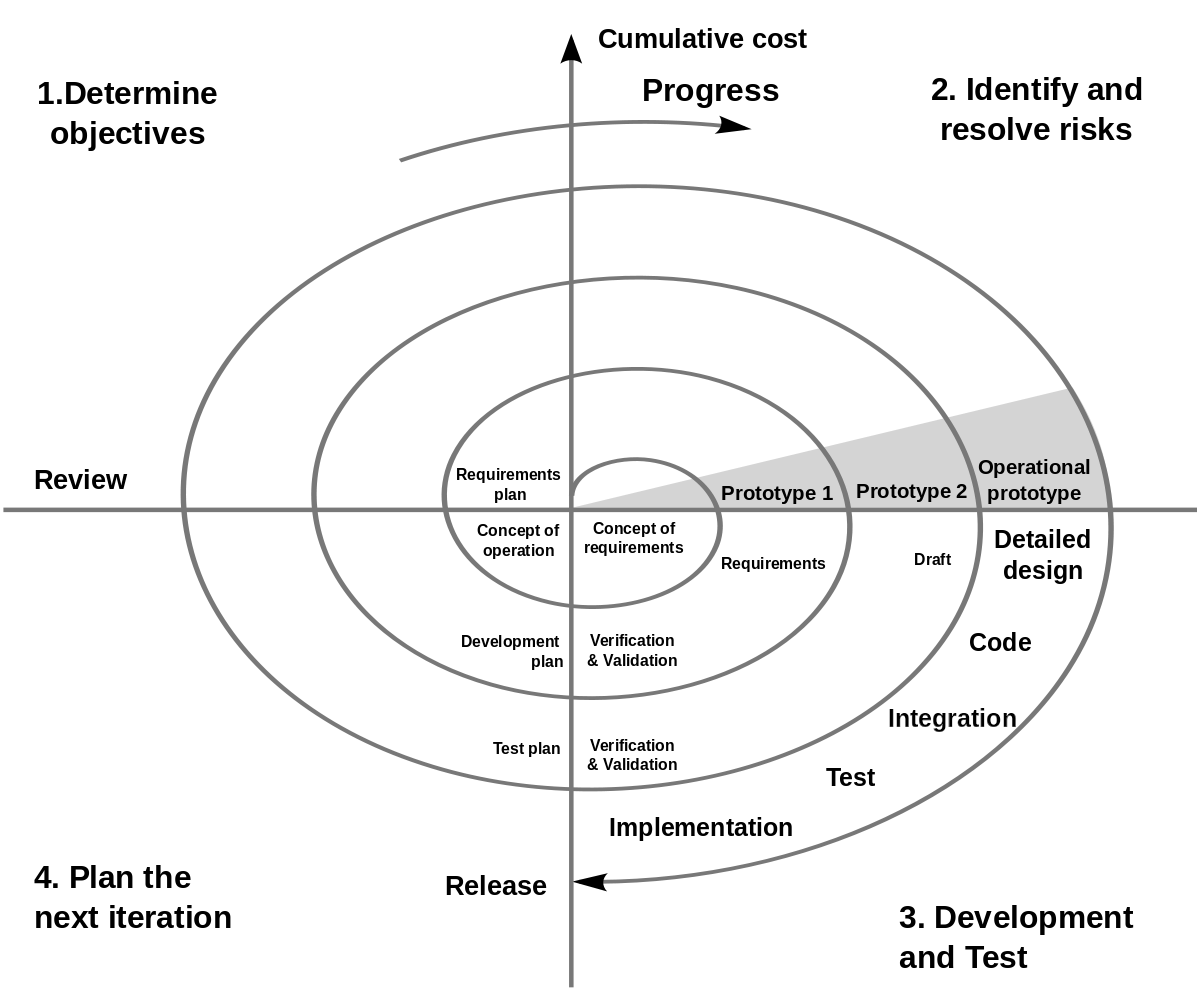


Figure 14 – Spiral Methodology

The main benefits of using the spiral is its risk handling, flexibility in requirements, and it is good for very large and complex projects. The main drawbacks of this methodology is that it is expensive, doesn’t work for smaller projects, and its risk analysis requires highly skilled experts.

### 3.2.3 Conclusions

The methodology chosen for this project is the Agile Methodology, specifically the Scrum Framework. Scrum emphasises the usage of sprints in feature development, which is critical in the context of a Feature Driven Development (FDD) strategy. This project will be developed with a focus on documentation and short development cycles.

## 3.3. Overview of System

For the development of *Mixipe*, a feature-driven development (FDD) strategy will be employed, in which a sprint will plan, implement, and test a feature. Following the implementation of that item, work on the following feature will commence.

The design and code for the project will be supplied in stages. Before implementation, a feature will be thoroughly designed and researched.

The general approach for *Mixipe*’s development will be as follows:

1. Design and implement a basic android application with login functionality. Thus, allowing demonstration of communication between layers.
2. Design a feature (such as the swipe-able recipe cards).
3. Implement the feature.
4. Test the feature.
5. Repeat steps 2-4, implementing all the desired features (by priority).

The technical architecture depicts the number of layers in the application and how they communicate with one another. A three-tire model will be employed for this project as shown below. Changes to one layer of the model should not affect the other layers, hence why this model was chosen.

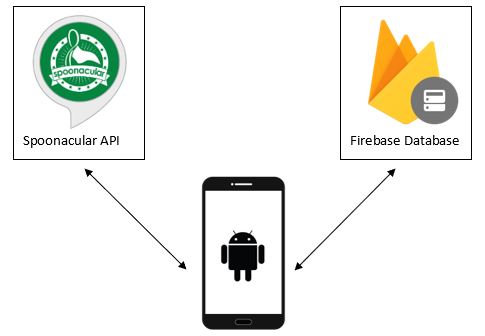


Figure 15 – Technical Architecture

## 3.4. Requirements Table

Below is a table which outlines the requirements for *Mixipe* and the level of importance each has to the system. High priority means that without one of these, the system would not be able to work as intended. Medium priority means that they are quite important to include in the app but are not the backbone of the project, and low priority means that they would make the app more functional and helpful but are not one hundred percent necessary.

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Priority** |
| User Login | Enables user to login on mobile device | HIGH |
| User Logout | Enables user to log out of the system | HIGH |
| User Create Account | Enables user to create an account | HIGH |
| Cards Swiper | Provides a swiping system for users to like / dislike recipes shown on cards | HIGH |
| Liked List | Enables user to view their liked recipes | HIGH |
| Search | Enables user to search for recipes | HIGH |
| View List | Enables user to view all recipes by scrolling through a list | HIGH |
| Ingredients | Enables user to view ingredients of a recipe | MEDIUM |
| Instructions | Enables user to view instructions of a recipe | MEDIUM |
| Servings | Enables user to view how many servings a recipe is for | LOW |
| Cooking Time | Enables user to view time it takes to prepare and cook a recipe | LOW |

Table 1 – Requirements

## 3.5. Front End

### 3.5.1 Introduction

The presentation tier is the front-end layer of the system. It is the part of the system with which the user will interact with *Mixipe* through their smartphone, and display content to them. Here the user will user their device to swipe on recipe cards, search for recipes and view their liked list.

In this section paper prototypes, medium-fidelity prototypes and use-case diagrams will be shown and examined.

### 3.5.2 Low Fidelity Prototype

To get a better understanfing of what kind of design might work for *Mixipe*, a paper prototype was designed, as shown in Figure 16 and Figure 17. These mock-ups are useful for the design stage as they may be given to potential customers to give them a feel for what the end product might look like. They were also useful as the customers could give feedback on whether they would add or change any of the design.

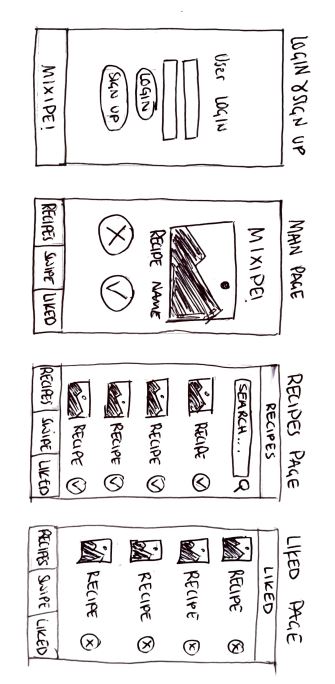


Figure 16 – Low Fidelity Prototypes

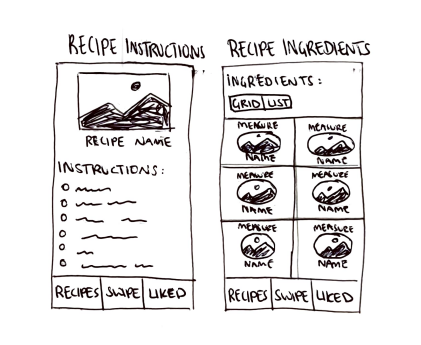




Figure 17 – Low Fidelity Prototypes

### 3.5.3 Medium Fidelity Prototype

After the paper prototypes, the next stage of the screen layout design phase began. To achieve this, a program called FluidUI was used. High-level designs for the screen layouts and interactions between each of the screens were designed using this program.

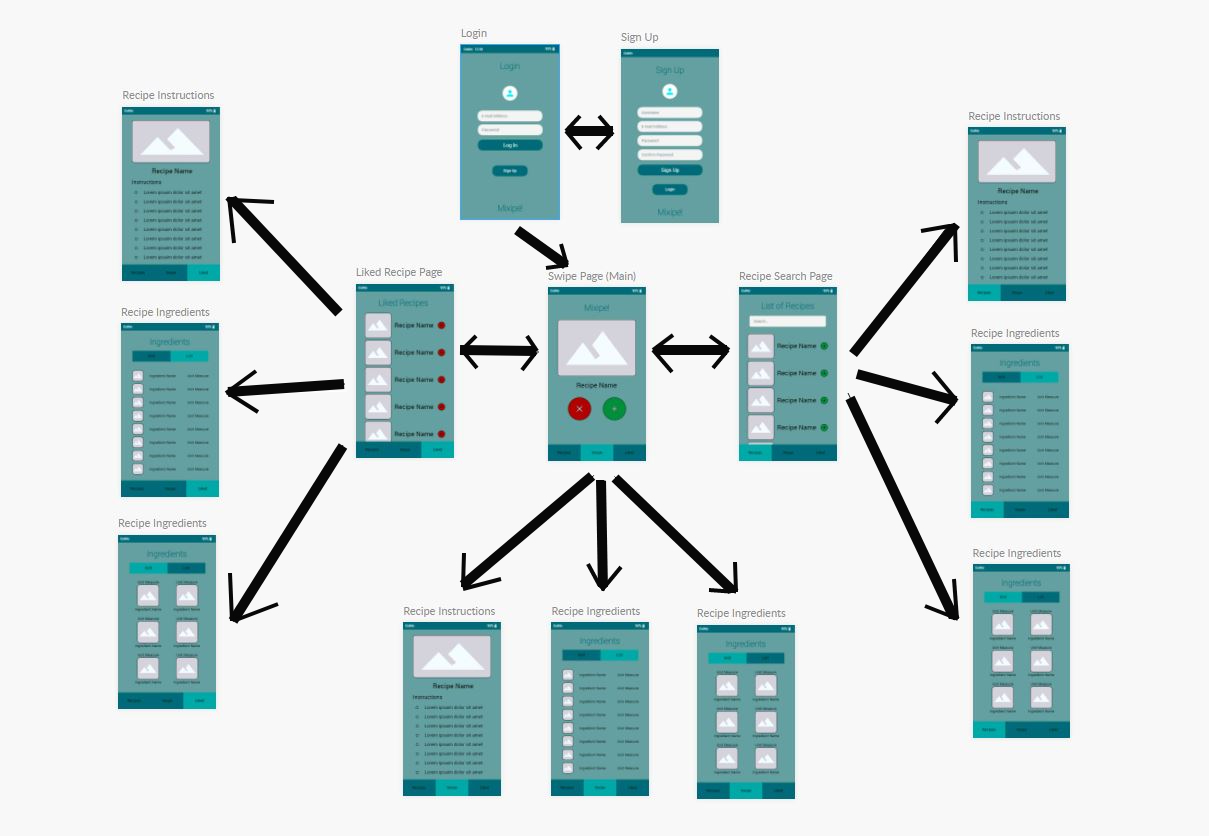


Figure 18 – Medium Fidelity Prototype

### 3.5.4 Use-case Diagrams

Use case diagrams are used to convey system behaviour and identify system functionality. The use case diagrams below show the progression of *Mixipe*’s system functionality.

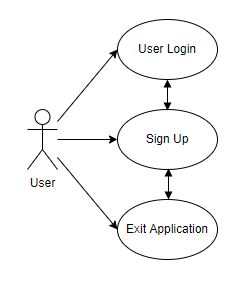


Figure 19 – Use Case 1st Iteration

In this use-case the user opens up Mixipe, click on sign up, then logs in to verify that their account was created and exits the application for later use.

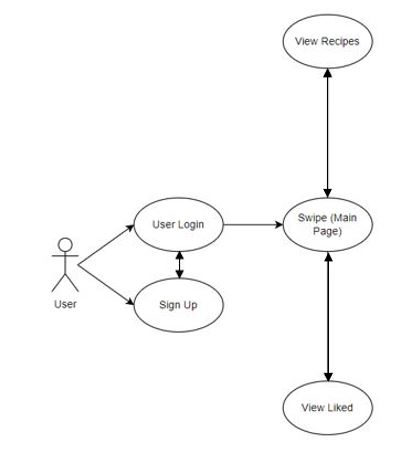


Figure 20 – Use Case 2nd Iteration

In this use-case the user opens up the application to look at Mixipe’s main features. They login, look at the swipe, view recipes and view liked pages to get a better understanding of the app.

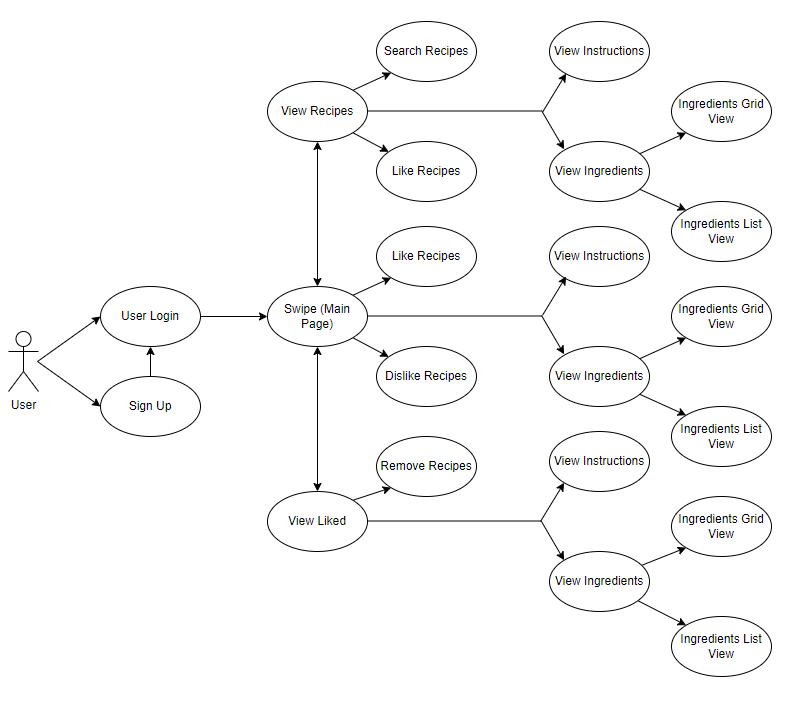


Figure 21 – Use Case 3rd Iteration

The above use-case demonstrates how a new Mixipe user might run through the application. First, they create an account and login. Then they are presented with the swipe page where they begin liking and disliking recipes. They click onto one of the recipes to get more detail (e.g. view the instructions or use the ingredients view as their shopping list). Alternatively, the user goes to the view recipes page and searches for a specific recipe using the search engine and likes a few recipes they find there. Later the user may view their list of liked recipes in the view liked page where they also have the option to remove previously liked recipes.

### 3.5.5 User Scenarios

Since there are many ways a user might use *Mixipe*, the following user stories were designed to help and explain the paths that a user might take.

**User Story 1:** Sarah opens the *Mixipe* app. She is presented with the login screen, however, Sarah does not have an account yet, so she clicks the signup button. She enters all her necessary details such as email address and password. Once signed up, she inputs her login details and successfully logs in and is presented with the swiping feature where she is shown a random recipe. She doesn’t like the recipe, so she dislikes it, and continues until she finds a recipe she likes. She adds the recipe to her liked list so she can use it to cook dinner later. She closes *Mixipe*.

**User Story 2:** Jack is walking around the supermarket deciding on what to buy for lunch. Jack opens *Mixipe* and starts swiping on recipe cards until he finds a recipe he likes. He goes to the ingredients tab to see what he needs to buy. He walks around the supermarket with *Mixipe* as his shopping list. Once Jack has found all the ingredients he needs, he closes *Mixipe*.

**User Story 3:** Lara is using the Mixipe swiping feature, but realises she wants to cook spaghetti Bolognese tonight rather than find something new. She clicks on the Recipes page and uses the search engine to search for “Spaghetti Bolognese”. Lara is shown a list of different Bolognese styles and sees one she likes. She adds it to her liked recipes page where she can view it later when she is cooking. Lara closes *Mixipe.*

**User Story 4:** Lara is ready to cook her dinner, so she opens *Mixipe* and goes to her liked recipes page. She sees her spaghetti Bolognese recipe saved and opens it. She views the ingredients and starts preparing them. Once she has done all her prep, she opens the recipe instructions and starts following them. Once she has finished the instructions and her spaghetti Bolognese is cooked, she closes the app.

## 3.6. Middle Tier

The middle tier, also known as the application layer, contains the functional login that will drive *Mixipe’s* core capabilities. The middle tier is made up of Java objects and classes that connect the frontend and backend. Things controlled by the application layer are examined in this section.

UI Navigation

The application layer allows the user to navigate across *Mixipe’s* UI by utilizing buttons and other elements that are visible to the user.

Data Transfer

The middle tier also assists with data transfer requests such as register and login. It accomplishes this by submitting users’ credentials to the database for verification.

APIs

The Spoonacular API works in the application layer to send receive data from the backend of the application.

## 3.7. Back End

It is important to create an ERD diagram before starting to build a database. ERDs are helpful for visualizing how the running database will look. They are data-structure diagrams that show entities and their attributes, as well was the relationships that exist between them.

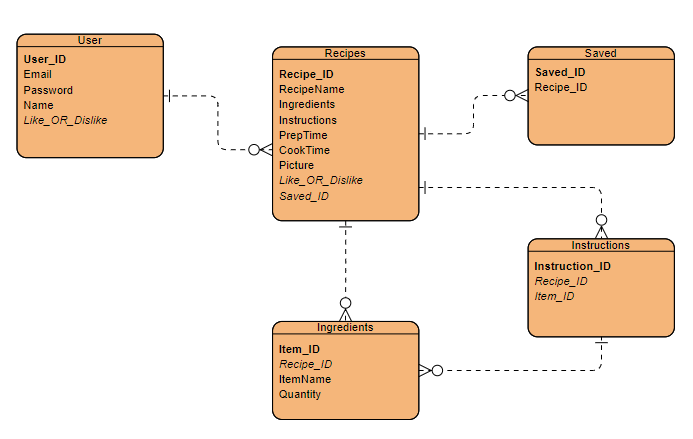


Figure 22 – ERD Diagram

*Mixipe’s* initial ERD draft is shown in the diagram above. The Recipes table is connected to four different tables; User, Saved, Ingredients, and Instructions. Lots of recipes can be saved (liked), recipes have many ingredients, there are many instructions per recipe and users can Like\_OR\_Dislike multiple recipes. The Like\_OR\_Dislike variable indicates whether a person liked or disliked a recipe.

## 3.8. Conclusions

The Prototype Design chapter utilized the material from the Literature Review section and defined the design plans that are required for *Mixipe’s* development. This was accomplished by first looking at the methodology that will be utilized in the development process, then a broad overview of the technological architecture, and finally the front-end, middle tier and back-end design in detail.

# 4. Prototype Development

## 4.1. Introduction

This chapter is intended to carry on from the previous design chapter. The design concepts described are expanded upon, as well as methods through which they were executed. The majority of this chapter focuses on the technical design employed in the development of *Mixipe*.

## 4.2. Prototype Development

A user login page and registration page were developed using the Android Studio IDE and Firebase. The Android Studio IDE makes it easy to create and edit new projects as it comes with support for Java, Kotlin, and XML. It also allows users who do not have access to an Android phone to use an emulator to test their applications functionality as they go on. In the case of Mixipe, the prototype testing was carried out on an Android phone called the OnePlus Nord. Android Studio also supports Firebase integration, guiding the user through the process of integrating it into the app.

### 4.2.1 Firebase

Firebase is planned to be the foundation for a lot of Mixipe’s functionality, therefore it had to be implemented first. Setting up Firebase to function with an Android application was quite simple thanks to a lot of documentation. The built-in Firebase tool in Android Studio was very helpful for developing the firebase authentication.

The figure below (Figure 23) shows how Android Studio assists developers during the development process. Its integrated Firebase wizard contains tutorials to help implement a variety of services such as the authentication system. Other firebase services can also be accessed through similar tutorial windows. These windows will automatically make changes to the applications gradle files to add necessary libraries and dependencies to the project. For example, Firebase Authentication was added to the application gradle. This allowed for a custom login and registration systems to be used. The registration system uses the users name, email, and password to create and account, and the login system uses the user’s email as a username and their password to log in to the application. Both of these systems were simple to set up and there were no complications with them thanks to the built in Firebase wizard and other tutorials.

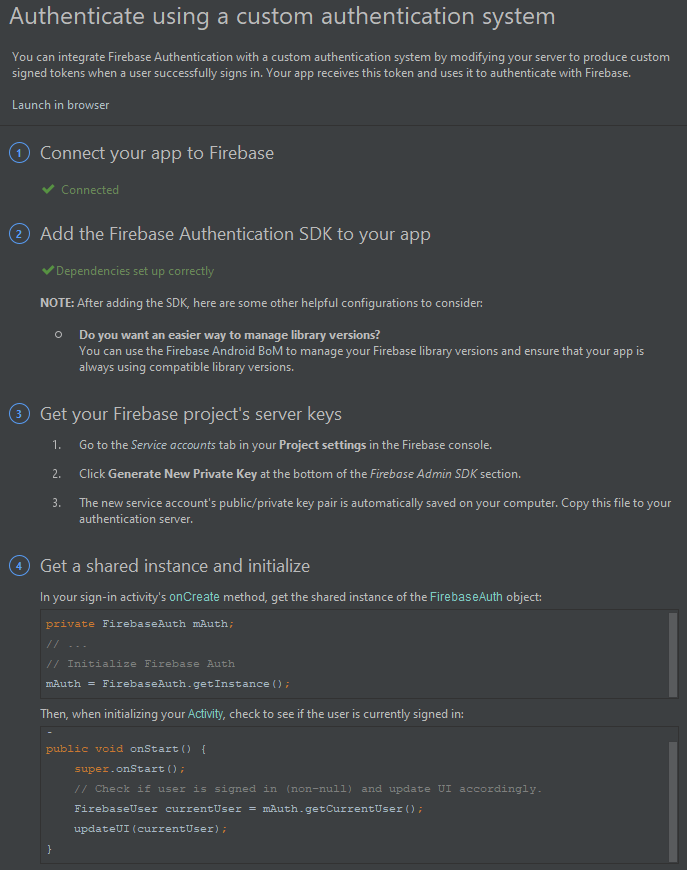


Figure 23 – Firebase Authentication

### 4.2.2 Spoonacular API

The Spoonacular API was key to the functioning of Mixipe.

## 4.3. Front End

### 4.3.1 Registration page

The registration page was the first section of the UI to be designed and developed. Users can use this page to create a Mixipe account or log in to an existing one. The page was designed in Android Studio using XML. Users are asked to enter their Name, email address, create a password and confirm that their password has been entered correctly. If the user already has an account, they can simply press the login button which will bring them to the login page. Some error checking has also been added, such as, ensuring that the user has given a valid email address, that they didn’t mistype their password by making users confirm their password, and passwords must be greater than or equal to 6 characters. The prototype can be seen in the below figure.

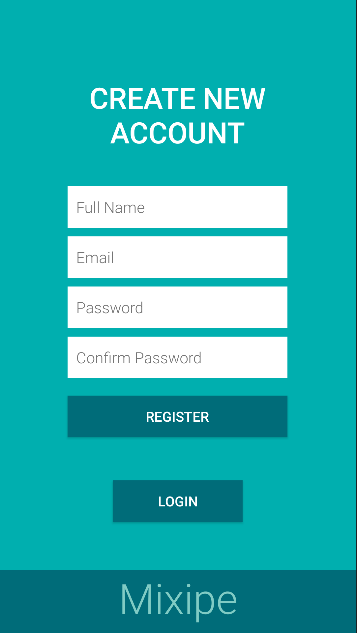


Figure 24 – Registration Page



Figure 25

The above figure is a brief snippet of the XML code used to create the register page. This XML code is where many of the registration page’s buttons, text, input fields, and overall layout is designed.

The first field from the above code snippet is the code for the password field. This is a text box where the user can type their desired password. Attributes such as background colour, text colour, box size, font style, constraints, and more are also initialized here. This field is linked to the Register class using the “@id/Password” id. This makes it possible for the value specified in this field to be passed to the class.This is the same regarding all the properties defined in this XML layout file, they each have an id that links them to the Register class to enable functionality.

### 4.3.2 Login Page

The login page is the first thing any Mixipe user sees when they open the app. If a user has not yet created an account, they can do so by clicking the Register button at the bottom of the screen which will bring them to the Registration Page. Users simply enter their email and password and press Login. Users email address and password are then checked against the Firebase authentication database and if they match, the user is logged in to Mixipe. If a user does not enter a valid email or password, an error will pop up. Likewise if they enter an email which has not been registered or if they mistype their password, an error will be shown to them. Another option on the login page is the “Forgot Password?”. This allows users who have registered an account but forgot their password, to request a password reset.

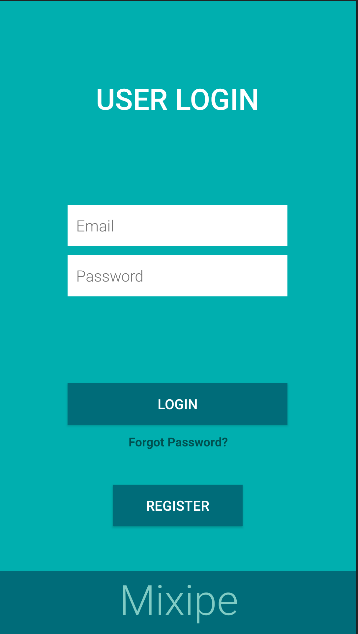


Figure 26 – Login Page



Figure 27

The figure above is a small section of the code found in the XML file used for the login page. The first field seen in the above code is a text field that asks the user to enter their email address. This is followed by the text field that asks the user to enter their password. Each of these (email and password) are then checked against the Firebase Authentication database through the Login class.

This file contains a number of attributes that are used in the login’s page functionality including, text fields, text views, buttons, and more. All of these attributes in unison create the overall design of the login page.

### 4.3.3 Bottom Navigation Bar



Figure 28

The bottom navigation view bar is used to allow the user to switch between the different pages. If the user clicks on the “Search” button, they will be brought to the search page where they can scroll through a long list of recipes or simply search for them. The “Swipe” button brings them to the swipeable recipe cards where they can like or dislike recipes. The “Liked” button opens the liked page where the user can view or remove any of their previously liked recipes.

This navigation bar will be visibile at all times to allow the user to transition between each of the pages seamlessly.

The three code snippets below show the XML code used to create the bottom navigation view bar. The first snippet assigns various properties to the navigation bar, including its size, colour, and constraints. It also links the text colour and icon tint to the navigation selector which can be seen in the second code snippet.

This second XML code snippet is used to indicate to the user which page they are currently on. If they click on the “Search” button for example, the button will change its colour from black to white indicating to the user that they are currently on the search page.

The third code snippet is used to create each of the three buttons as icons with titles.

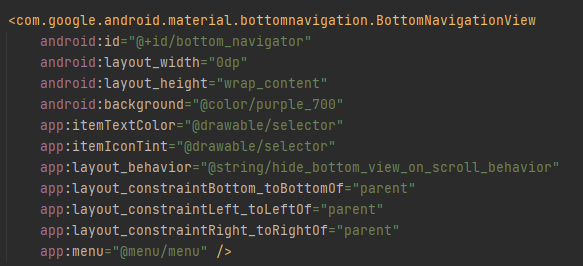


Figure 29

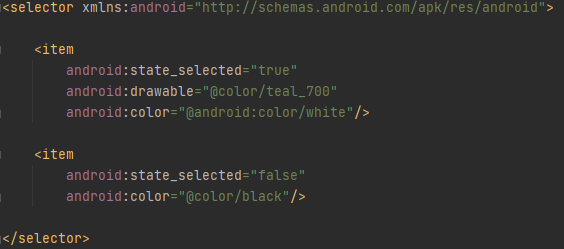


Figure 30

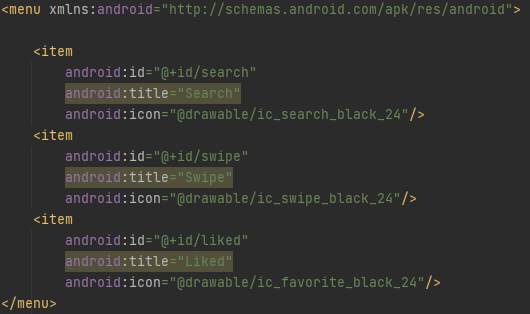


Figure 31

### 4.3.4 Swipe Page

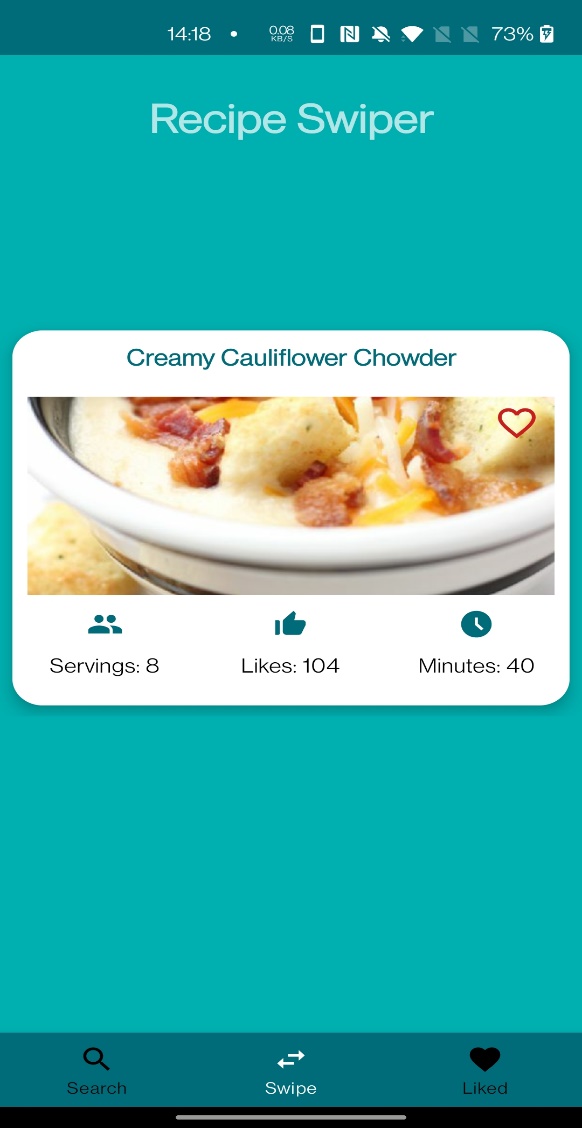


Figure 32

The swipe page is the main feature of Mixipe. Here the user is presented with a random recipe card containing the recipes title, image, servings, likes and cooking time. Using this information the user can choose to either like or dislike the recipe by swiping right or left. If the user wants to see more information, they can simply click the recipe card which will bring them to a page containing the additional information such as ingredients, method, etc. A small text notification appears at the bottom whenever the user swipes right or left that informs them which action they just made.

### 4.3.5 Search Page

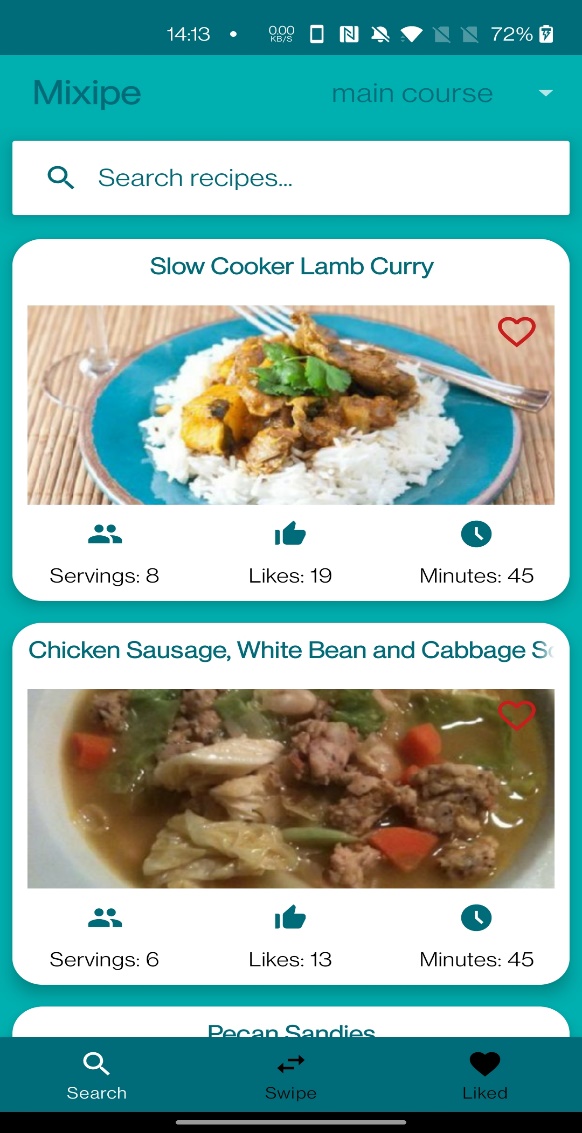


Figure 34

The above figure shows the search page which is used to allow the user to scroll through recipes based on a category (i.e. main course, dessert, drinks, etc.) by simply clicking on the drop-down menu in the top right corner or allow them to use the search bar to search for recipes using natural language. Recipe cards are shown to the user in a scrollable list. The recipe cards include important information such as the title, an image of the recipe, the number of servings, how many other users have liked it, and the cooking time. The user has the option to click on any of the recipe cards to view additional information including the source, ingredients, method, and similar recipes.



Figure 35

The above code snippet is a section of the XML file used to create the overall layout of the search page. The first field is used to create the search bar at the top of the page which allows users to enter a search query. It also assigns its properties including its size, colour, elevation, margin, and constraints. The second field is responsible for displaying various recipes cards in a vertical alignment so that the user can scroll through them.

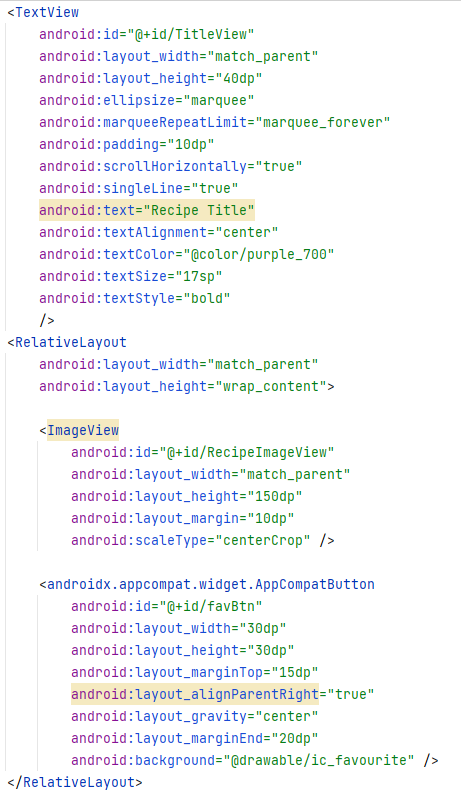


Figure 36

The XML code snippet above is used to create the recipe cards for the search and swipe page. These cards contain the recipes title, image, number of servings, likes, and cooking time. The first field of the XML code is used to assign the recipe title on the card and assign its properties. The second section in the above snippet assigns the recipe image along with its size, scale type and margin properties. The AppCompatButton is a button which the user can click to add the recipe to their liked list.

### 4.3.6 Recipe Details Page

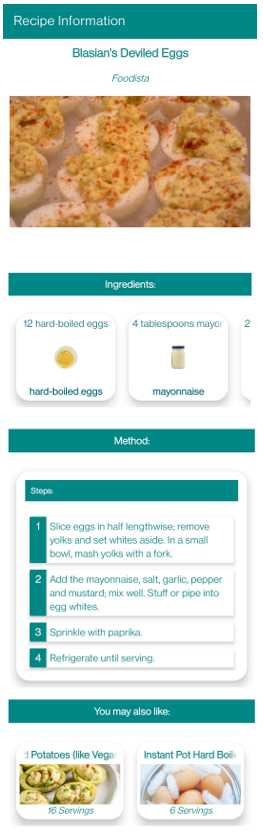
 

Figure - Recipe Details Page

The recipe details page is accessed when the user clicks on a recipe to see more information. By scrolling through this page the user can see additional information such as the recipe source, the ingredients used, the method, and some similar recipes are recommended to the user.

The user can scroll through the ingredients horizontally from left to right. Each ingredient contains its quantity, image, and title. For the method, the user is shown each step in a separate text box to make it visually clearer. At the bottom of the page, users are recommended up to 5 similar recipes. Similarly to the ingredients, the user scrolls horizontally to view them. The user can also click on any of the recommended recipes to see additional information.

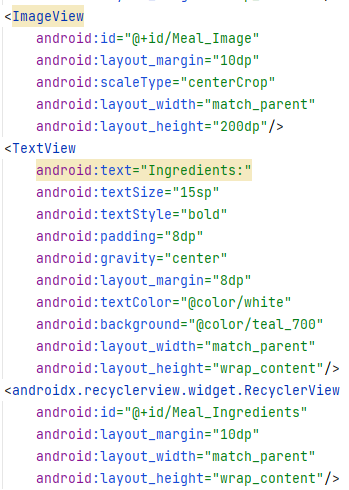


Figure - Recipe Details XML

The code above is a snippet of the XML code for the recipe details page. In this snippet the ImageView is used to assign the image properties such as scale type, size, margin as well as an id to links it to the Recipe Details class to enable functionality. The TextView creates a header box above the listed ingredients and the RecyclerView then assigns each ingredient card to its position.

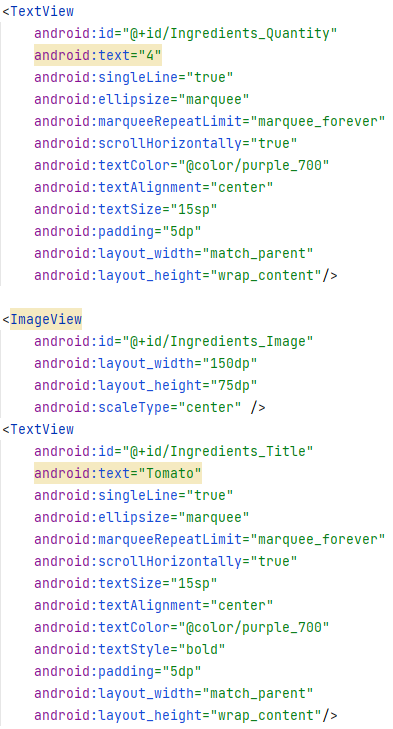


Figure - list ingredients

The XML code snippet above is used to create the ingredient cards for the recipe details page. It creates small cards that contain the ingredient quantity, image, and title. The first TextView assigns necessary properties to the ingredient quantity such as colour, size, alignment, paddings, etc. The text is displayed as a horizontally scrolling text from left to right using marquee in order to allow longer words and text to be shown.

The ImageView assigns the ingredients image properties and the other TextView assigns the ingredients title properties which are similar to the quantity properties.

### 4.3.7 Liked Page



## 4.4. Middle Tier

### 4.4.1 Registration Class

This first section of the middle tier that will be outlined is the registration activity. Code used for this class will be outlined and explained below.

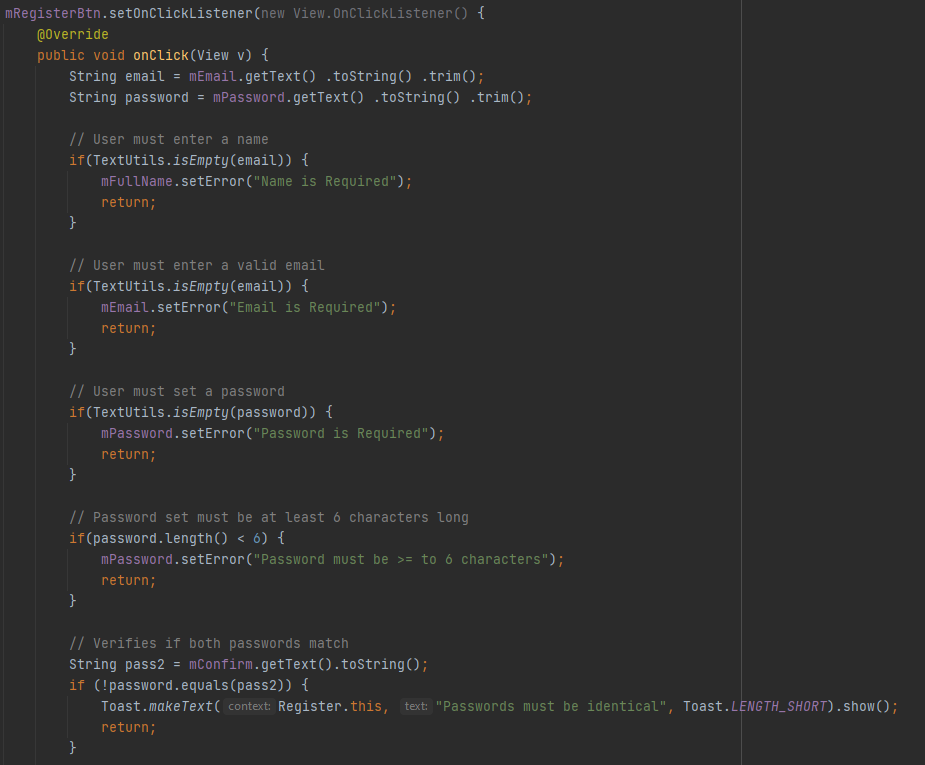


Figure 41 – Registration Page Error Checking

The above code is used for all the error checking on the registration page. Including making sure users enter a valid email, password of at least 6 characters and password verification.

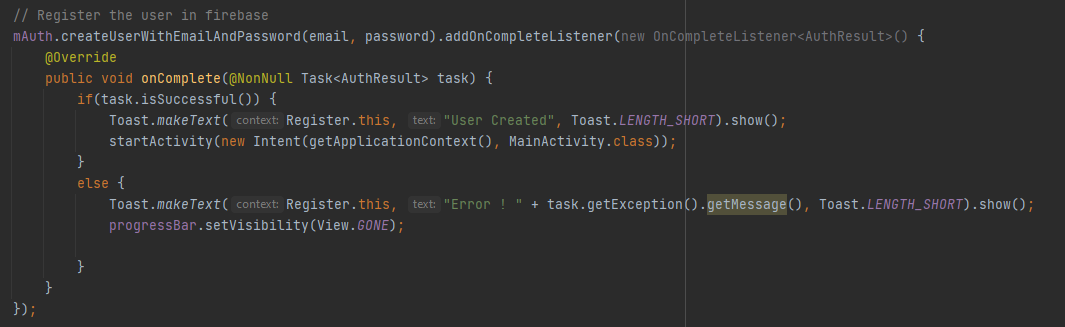


Figure 42 – Registers user in Firebase

The above code is used to register the user in Firebase. If a user already has an existing account with the entered email address, they will be thrown an error. Firebase handles all of this as it stores the user’s information in the Firebase Authentication section where only the users email address and uniquely generated User ID as seen below.

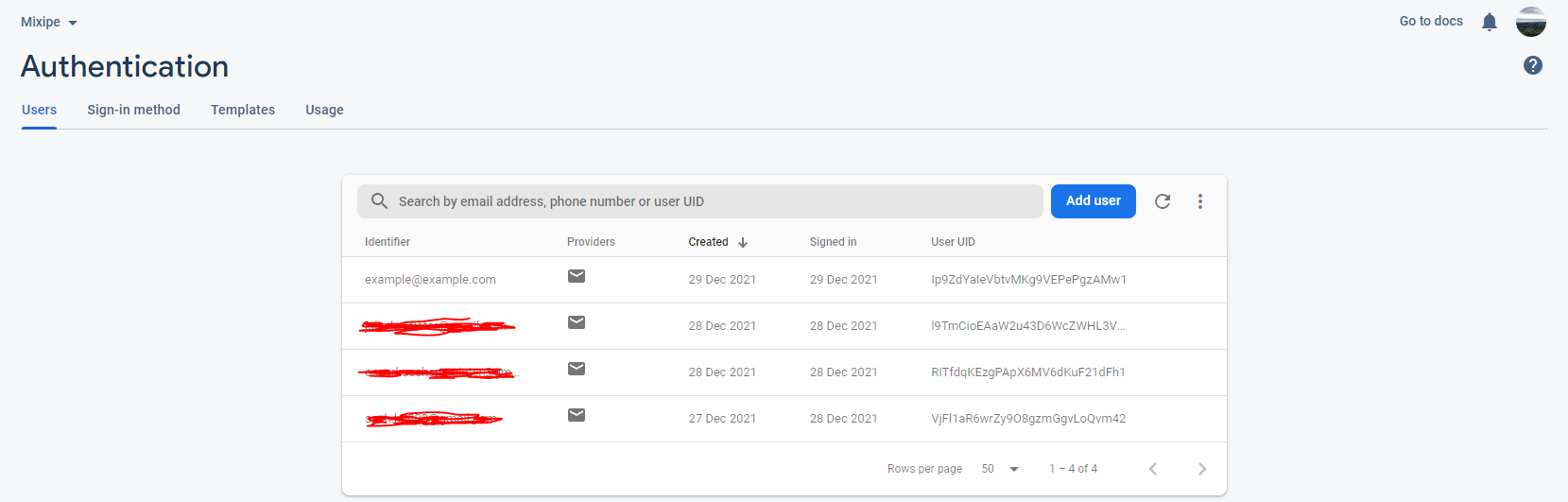


Figure 43 – Firebase Console | Authentication

The code shown below is used to switch from the registration page to login page when the user clicks the “Login” button

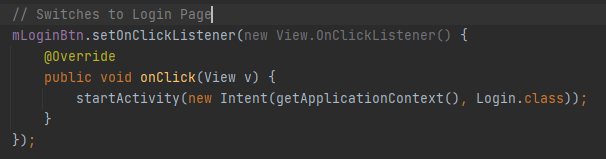


Figure 44 – Switch to Login Page

### 4.4.2 Login Class

This section is used to explain how the Mixipe login activity process works. Each section of code will be detailed below.

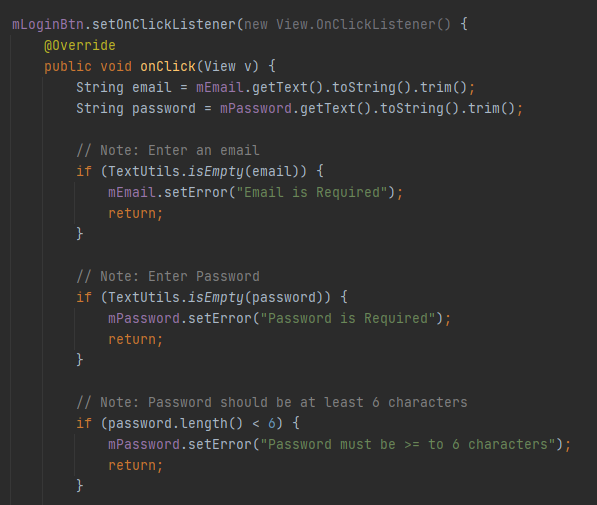


Figure 45 – Login Page Error Checking

The above code is used for simple error checking, such as, if a valid email address entered, password is entered, and if the password is at least 6 characters long.

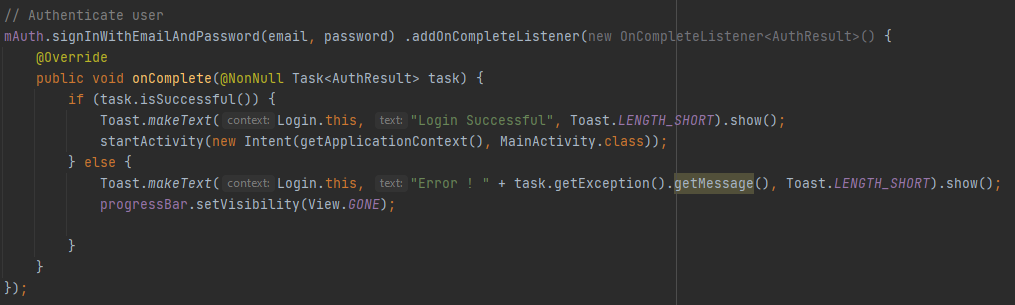


Figure 46 – Login authentication

The code above authenticates the user when they try to log in using their email and password. Using Firebase authentication the email and password are checked and verified, if they are correct, the user will be logged in and brought to the main page. If an email is entered that has not been registered or the password is incorrect an error will pop up.

The code shown below is used to switch from the login page to the registration page when the user clicks the “Register” button.

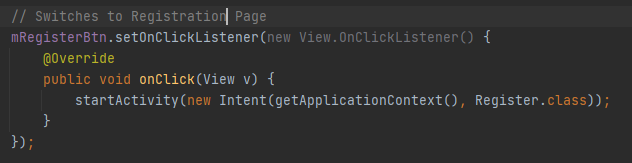


Figure 47 – Switches to Registration Page

The function below is used if the user forgets their password. They have the option to click on the “Forgot Password?” below the Login button which will then open a window in the middle of the screen and ask the user if they wish to reset their password. The user must enter their email address, and if the email address has been previously registered, the user will receive an email with a password reset link which they can use to set a new password. If a user who has not yet registered their email tries to reset their password, an error will be returned to the user.

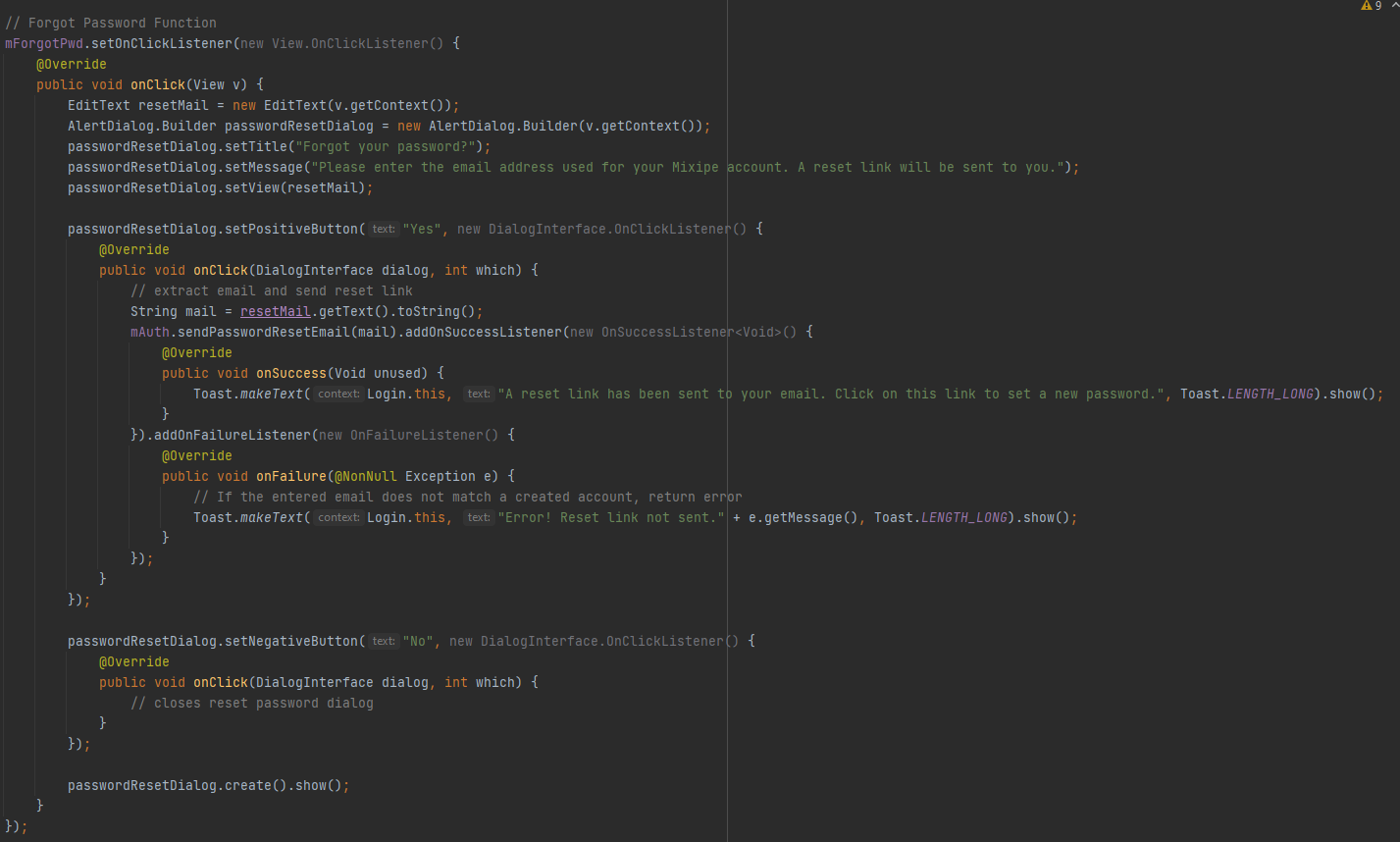


Figure 48 – Forgot Password function

### 4.4.3 Bottom Navigation

This part is used to explain how the bottom navigation bar works. The bottom navigation bar has three options that the user can select (search, swipe, liked). The figure below details the code used for this section.

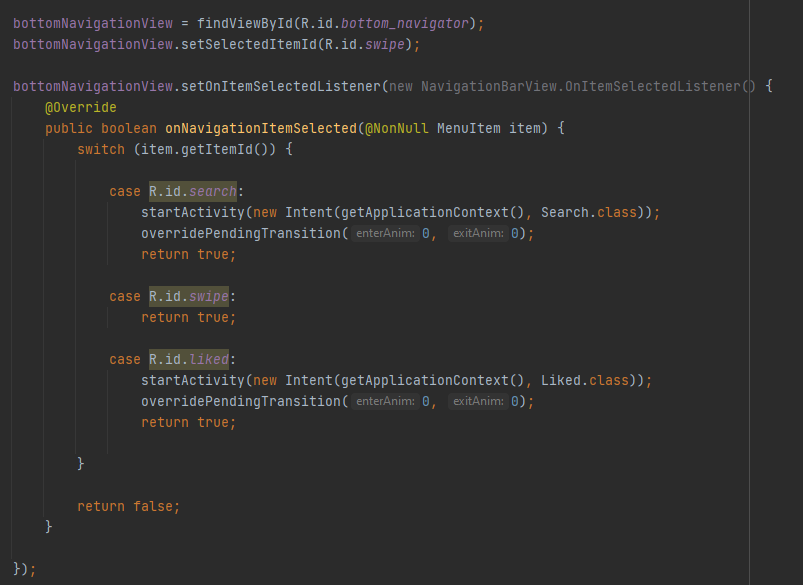


Figure 49

The above code shows how the bottom navigation view works. The above example is taken from the swipe class. The “bottomNavigationView.setSelectedItemId(R.id.swipe)” sets the current page to the swipe page. A switch statement is used to allow the user to change between the three activities. In the case that the user clicks on the search button, it will start the search activity and transition the user to the search page. The same applies when the user clicks on the login button; the liked activity is started, and the user is transitioned to the liked page.

The code for the bottom navigation view is used on each of the three main classes (search, swipe, liked) to ensure proper functionality across each of the three pages.

### 4.4.4 Swipe Class



Figure 50



Figure

### 4.4.5 Search Class

### 4.4.6 Recipe Details Class

### 4.4.7 Request Manager Class

### 4.4.8 Liked Class

## 4.5. Back-End

### 4.5.1 Spoonacular API

An example response from an API request to the Spoonacular API can be seen in the figure below.



Figure 52

The figure above is a code snippet returned by the Spoonacular API query. In the above example a GET request for a pasta recipe was passed to the API. The query then returns JSON file, that contains key details about the recipe. These include whether a recipe is vegetarian, vegan, etc., servings, cooking & preparation time, and more. GET requests like the example above are used to display recipes and their key information to the user in a simple and readable way.

### 4.5.2 Firebase

Authentication with Firebase and Realtime data storage was very important in the development of Mixipe’s backend.

The following figure shows the Firebase Authentication page from the Firebase console.

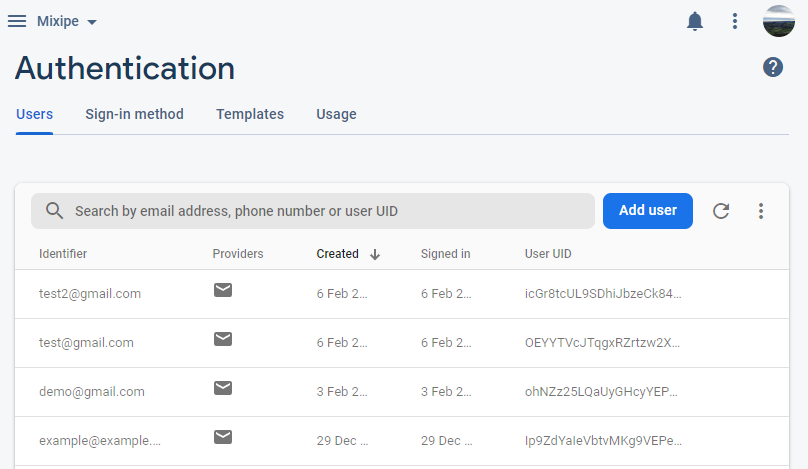


Figure 53

The Firebase authentication system makes it easy for developers to create users on their database. It creates a unique user ID for each user and encrypts their passwords (31).

# 5. Testing and Evaluation

## 5.1. Introduction

Testing and Evaluation are essential to guarantee that a project satisfies the requirements that have been set for it in order to ensure that the system is functioning properly. Testing and evaluation can be viewed as a filter that ensure that a product which is due to be released is actually ready for the market.

## 5.2. Plan for Testing

The project will be tested continuously during its entire life cycle. Throughout the development process, the app will be tested on both an emulator and real Android phones to give hands-on user trials and to confirm that it works as planned.

By setting up a GitHub repository and routinely backing up and committing changes, any modifications done to the project can be rolled back if there are any sever issues.

Another plan for assessing the project is to have test groups utilize the app. These groups will include a wide range of protentional end users.

Because there will be so many different elements to the project that make up the entire app, unit testing will be used. Unit testing is the process of creating tests that interact with the application directly. These tests will guarantee that every component of the software is working properly. The proposed agile strategy requires unit testing because all components of the program produced will be tested after each sprint.

Blackbox testing will also be utilized to test the concept from a different perspective. The tester will have no knowledge of the items internal structure, design, or implementation. The black box testing approach will look for flaws in areas like interface, performance, incorrect functions, initialization, and more (32).

Combining the use of unit testing and Blackbox testing was decided to be an adequate way of testing the project. Manual testing will also take place, in which the developer will assume the role of the end user and evaluate the app’s feature to guarantee its behaviour.

### 5.2.1 Test Plan Table

Outlined in the table below is the general plan that will be followed while testing the system*.* Ideally all tests will be passed before Mixipe is fully released.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No.** | **Test Description** | **Expected Outcome** | **Pass?** |
| 1 | Does the phone app load when the icon is selected? | The app will load and bring the user to the login screen. |  |
| 2 | Does the app close when the exit button is pressed? | The app will shut down correctly when exit is pressed. |  |
| 3 | Can a User successfully create an account? | User will be able to create and account which they can use to login to the application. |  |
| 4 | Can a User login with their registered details? | A registered user who enters details will be able to login and get to the main screen. |  |
| 5 | User login with invalid details? | Error is displayed to screen. |  |
| 6 | Can a user successfully log out? | The user can logout and return to the login screen from the main menu |  |
| 7 | Can the user delete all information stored about them? | When a “clear all” button is used it will remove all details stored about the user. |  |
| 8 | Can a user like a recipe? | Recipe will be added to “Liked Recipes” |  |
| 9 | Can a user dislike a recipe? | Recipe will not be shown to user for a certain period of time |  |
| 10 | Can a user unlike a recipe? | Recipe will be removed from “Liked Recipes” |  |
| 11 | Can a user search for a recipe? | User can use the search engine to search for a recipe using plain natural language |  |
| 12 | Can a user view recipe instructions? | User can click on a recipe to see its instructions |  |
| 13 | Can a user view recipe ingredients? | User can click on a recipe to see its ingredients |  |

Table 2 – Test Plan

## 5.3. Plan for Evaluation

The process of evaluating a systems is just as significant as the process of testing it. This is because one of the most difficult problems to address when designing a project is determining the ideal user experience. This can be difficult for any developer as it is intended that the program will be as user-friendly as possible while simultaneously containing very complex code. As previously stated, the systems intuitive and instinctive usability was a top emphasis throughout the development process. Having potential users evaluate the system would ideally ensure that its usability is of high quality, meets industry standards and ensures that all development is feature drive development.

## 5.4. Conclusions

This chapter looked at the systems testing and evaluation. Unit testing, Blackbox testing and Manual testing will be extremely helpful for the development phase of this project. The evaluation from a variety of potential users will also be of great help throughout the developing process.

# 6. Issues and Future Work

## 6.1. Introduction

This chapter will look into any issues and risks that could arise during the development of the project as well as what the plan for future work will be. There are many issues and risks involved in the development of an Android app which will be outlined in this chapter. Any future work planned for this project will be outlined and illustrated using a GANTT chart.

## 6.2. Issues and Risks

The following are issues that have yet to be tackled in this project:

* Lack of knowledge with Firebase, particularly when it comes to integrating it with App development
* Lack of understanding how to use APIs and other Android development features.
* Finding a balance between keeping important and sensitive user information safe and making the app as accessible as possible

The following is how the author intends to handle these distinct challenges:

* Complete an online Firebase course and watch videos about how to integrate it into app development.
* Try out different APIs to check which ones are most compatible with the project
* Continue to try to keep the best possible security/accessibility balance, since this will be a continuous issue.

The following are risks that the author may encounter while working on the project:

* Poor internet connection may prevent the author from demoing or using some of the app’s functions
* Not completing the proposed work in a timely manner.

The following is how the author intends to handle these various risks:

* Ensure internet connection is strong and stable by testing it before the demo and release
* Create and stick to a schedule for the project. This will ensure that the project’s proposed elements are broken down into manageable sprints and that they are finished within the proposed time frame.

## 6.3. Plans and Future Work

The project plan is depicted in the GANTT chart below. As the project progresses, another GANTT chart will be created, and the two charts will be compared in the conclusions to see the differences between the planned and actual approaches. The project will continue to be completed with discussions with test users to ensure that the final user experience meets the highest possible standards.

### 6.3.1. GANTT Chart

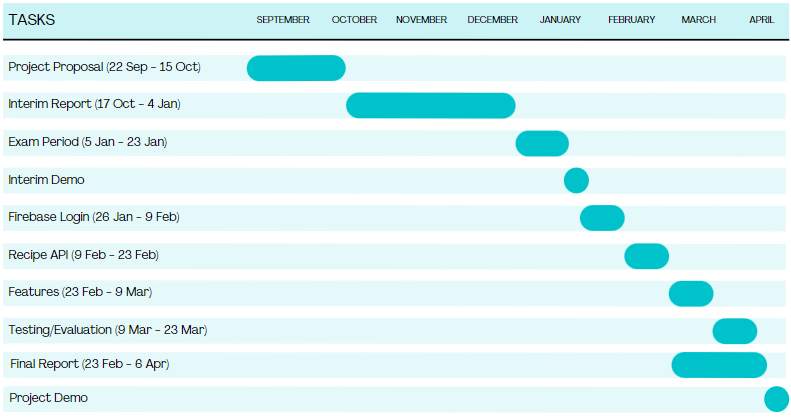


Figure 54 - Gantt Chart

# Bibliography

|  |  |
| --- | --- |
| 1. | Turner A. (2021) How many Smartphones are in the world?, <https://www.bankmycell.com/blog/how-many-phones-are-in-the-world?nowprocket=1>, Date Accessed: October 2021. |
| 2. | Lin Y. (2021) 10 Mobile Usage Statistics, <https://www.oberlo.com/blog/mobile-usage-statistics>, Date Accessed: October 2021. |
| 3. | Brimble L. (2020) More than 70% of adults use social media for recipes instead of cookbooks, <https://www.independent.co.uk/news/long_reads/science-and-technology/recipes-online-cookbooks-food-inspiration-social-media-facebook-instagram-b1397624.html>, Date Accessed: October 2021. |
| 4. | Cooper J. (2015) Cooking trends among millennials, <https://www.thinkwithgoogle.com/intl/en-gb/future-of-marketing/digital-transformation/cooking-trends-among-millennials/>, Date Accessed: October 2021. |
| 5. | Yummly. (-) About Yummly, <https://www.yummly.com/about>, Date Accessed: October 2021. |
| 6. | statcounter. (2021) Desktop vs Mobile vs Tablet Market Share Worldwide, <https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-202012-202112>, Date Accessed: October 2021. |
| 7. | statcounter. (2021) Mobile Operating System Market Share Worldwide, <https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-202010-202110>, Date Accessed: October 2021. |
| 8. | Apple. (-) Enrollment, <https://developer.apple.com/support/enrollment/#:~:text=The%20Apple%20Developer%20Program%20annual,currency%20during%20the%20enrollment%20process>, Date Accessed: November 2021. |
| 9. | Sharma A. (2021) How to Upload an App to Google Play Store?, <https://appinventiv.com/blog/how-to-submit-app-to-google-play-store/#:~:text=There%20is%20a%20one%2Dtime,Store%20Play%20apps%20for%20free.&text=Once%20you%20submit%20your%20account,48%20hours%20to%20get%20approved>, Date Accessed: November 2021. |
| 10. | Android Studio. (-) Android for Developers, <https://developer.android.com/studio>, Date Accessed: November 2021. |
| 11. | MongoDB. (-) Realm, <https://realm.io/>, Date Accessed: November 2021. |
| 12. | SQLite. (-) About SQLite, <https://www.sqlite.org/about.html>, Date Accessed: November 2021. |
| 13. | Firebase. (-) Firebase.google.com, <https://firebase.google.com/>, Date Accessed: November 2021. |
| 14. | Batschinski G. (-) What is Firebase? All secrets unlocked, <https://blog.back4app.com/firebase/#What_is_Firebase_and_how_it_works>, Date Accessed: November 2021. |
| 15. | Firebase. (-) Firebase Documentation, <https://firebase.google.com/docs/database/backups>, Date Accessed: November 2021. |
| 16. | Rapid API. (-) Rapid Api Documentation, <https://rapidapi.com/webknox/api/recipe>, Date Accessed: November 2021. |
| 17. | ProgrammableWeb. (-) Yummly REST API, <https://www.programmableweb.com/api/yummly-rest-api>, Date Accessed: November 2021. |
| 18. | Spoonacular. (-) Spoonacular API, <https://spoonacular.com/food-api>, Date Accessed: November 2021. |
| 19. | Spoonacular. (-) Spoonacular Documentation, <https://spoonacular.com/api/docs/recipes-api>, Date Accessed: November 2021. |
| 20. | Kaplan K. (2021) 10 Usability Heuristics Applied to Complex Applications, <https://www.nngroup.com/articles/usability-heuristics-complex-applications/>, Date Accessed: November 2021. |
| 21. | World Health Organization. (2021) Blindness and vision impairment, <https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment>, Date Accessed: November 2021. |
| 22. | Pilton R. (2020) How to make your app accessible, <https://www.creativebloq.com/advice/app-accessibility>, Date Accessed: November 2021. |
| 23. | Oracle. (-) What is Java technology and why do I need it?, <https://www.java.com/en/download/help/whatis_java.html#:~:text=Java%20is%20a%20programming%20language,services%20and%20applications%20are%20built>, Date Accessed: November 2021. |
| 24. | Oracle. (-) Timeline of key Java milestones, <https://www.oracle.com/java/moved-by-java/timeline/>, Date Accessed: November 2021. |
| 25. | TutorialsPoint. (-) XML Overview, <https://www.tutorialspoint.com/xml/xml_overview.htm>, Date Accessed: November 2021. |
| 26. | W3Schools. (-) Introduction to XML, <https://www.w3schools.com/xml/xml_whatis.asp>, Date Accessed: November 2021. |
| 27. | GeeksforGeeks. (2021) A Complete Guide to Learn XML For Android App Development, <https://www.geeksforgeeks.org/a-complete-guide-to-learn-xml-for-android-app-development/>, Date Accessed: November 2021. |
| 28. | WorkFront. (-) Waterfall Methodology, <https://www.workfront.com/project-management/methodologies/waterfall>, Date Accessed: December 2021. |
| 29. | Wrike. (-) What is Agile Methodology, <https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/>, Date Accessed: December 2021. |
| 30. | The Economic Times. (-) Definition of Spiral Model, <https://economictimes.indiatimes.com/definition/spiral-model>, Date Accessed: December 2021. |
| 31. | Firebase. (2022) Privacy and Security in Firebase, <https://firebase.google.com/support/privacy>. |
| 32. | Imperva. (-) Black Box Testing, <https://www.imperva.com/learn/application-security/black-box-testing/>, Date Accessed: December 2021. |