# Meal Bay

CS39440 Major Project Report

Author: Weronika Milkowska (wem3@aber.ac.uk)

Supervisor: Chris Loftus (cwl@aber.ac.uk)

18th April 2023

Version 2.0 (Release)

This report is submitted as partial fulfilment of a BSc degree in
Computer Science and Artificial Intelligence (with integrated year in industry) (GG47)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

## Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work, I understand and agree to abide by the University's regulations governing these issues.

Name: Weronika Milkowska

Date: 18th April 2023

## Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name: Weronika Milkowska

Date: 18th April 2023

## Acknowledgements

I'm thankful to Chris Loftus, my supervisor, for his great help and guidance with my major project. It wouldn't have been as successful without his involvement, advice, and insights.

I want to thank Neil Taylor for his valuable advice regarding the project. Throughout the second semester, I am grateful for the guidance he offered.

I want to thank my family for supporting me. They encouraged me and helped me stay focused on achieving my project's goal.

I also want to thank all the testers of my mobile application for providing valuable feedback that helped me improve the project. Their different perspectives were beneficial in improving the app.

I appreciate everyone's support and contribution to the successful completion of this project. Thank you all so much.

## Abstract

The topic investigated in this work is the development of a mobile application called "Meal Bay" that provides users with a comprehensive platform for managing their recipes, improving their cooking skills, and expanding their culinary knowledge. The application caters to both beginners and advanced chefs, intending to help users plan their diets or experiment with new meals.

The author developed the Meal Bay app with a range of features, including the ability to add and store customised recipes, access preloaded recipes in the database, and create custom collections to save and organise recipes. The app also includes a shopping list feature to help users keep track of the ingredients they need for their chosen recipes, and users can create accounts to store their recipes securely and access them from any device with the app installed.

The project utilised various technologies to ensure efficient and effective data management and user experience. The app's development method was creating a mobile application using Android Studio IDE, Kotlin programming language, Jetpack Compose and Firebase Cloud database. The project was carried out using Scrum with the implementation of a Kanban board to manage the development process.

The key findings and outputs of the work include a functional and user-friendly mobile application that provides users with a range of features for managing their recipes and cooking skills. The app's ability to store personal recipes and create custom collections sets it apart from other food-related apps. The preloaded recipe database offers diverse options for users to broaden their culinary horizons.

Overall, the Meal Bay app provides a comprehensive solution for users to manage their recipes and cooking skills while also expanding their culinary experience, with the added convenience of being accessible on any device with an internet connection.

# Contents

## List of figures:

## List of tables:



# 1. Background And Objectives

## 1.1. Background preparation

Cooking is an important part of our daily lives but is frequently neglected due to the fast-paced lifestyle of many individuals. The author conducted background research on potential

reasons why cooking is often disregarded in daily life to understand the issue better. It was discovered that elements such as lack of time, motivation, and culinary knowledge were significant barriers for people to cook regularly. Moreover, the overwhelming amount of online information, including recipes, cooking techniques, and ingredients, can make it challenging for individuals to plan and cook meals efficiently.

The author identified the need for a comprehensive mobile application that can help users organise their recipes, enhance their cooking abilities, and develop a deeper appreciation for cuisine. This will address the issues related to these concerns. The author's passion for cooking also led the author to initiate the app's development. She focused on creating a platform that is simple to use, aims to reduce decision time, and, most importantly, makes the cooking experience enjoyable.

Initially, the author explored various options, including a web-based platform, to develop the project. However, after careful consideration, they ultimately created an Android mobile app. The reason was that mobile apps are more widely available to users, with most of the population owning a smartphone, and provide a more convenient and comfortable platform for users to access the app regularly and personalise it [1].

## 1.2. Relevant literature

The author of this work conducted a study of relevant literature to gain a deeper understanding of the issue of neglecting cooking by individuals and to inform the development of objectives aimed at resolving such issues. The literature reviewed enclosed sources, including academic journals, blogs, and reports. Overall, the insights gained from this literature review were necessary for shaping the objectives presented in this work.

## 1.3. Analysis

### 1.3.1. Security

The security risk for this project is considered low. There are still potential dangers associated with having an Android mobile app, and an example of such risk for users is the chance of unauthorised access to their devices or data through malware or phishing attacks. It is necessary to recognise security measures for all projects, even those considered low-risk, for both user and app safety. When designing the Meal Bay app, several options were considered for managing the app's data, including using a custom database or a third-party service. Ultimately, the author adopted the Firebase database due to its security features and simple setup process. Firebase offers real-time synchronisation, which is another advantage over the custom database.

The Firebase database used in the Meal Bay app has security rules set to ensure that users cannot see or alter any other user's data. This is an important measure to prevent unauthorised access and ensure data privacy. While there were other possibilities, such as using a custom database, the decision to use Firebase was based on its security, synchronisation time and ease of use.

### 1.3.2. Cloud

For the Meal Bay mobile application, the decision was made to use Firebase Cloud Firestore as the database solution. Firestore is a great choice for this app due to its many advantages.

One example is its ability to deal with large amounts of data. This includes recipes that were already installed and ones that users added themselves. Another benefit is that it makes the app's performance more efficient by using less memory on the user's device. Firestore comes with integrated security features that ensure the privacy and security of user data. The users can log in and out while their data is always stored securely in the Firebase Cloud.

Using Firebase Firestore in the Meal Bay mobile application provides an efficient and secure way to store and handle large amounts of recipe data for users.

## 1.4. Project objectives

To analyse the problem of people neglecting cooking on a daily basis, the author first considered the ideal target audience and their needs. After researching, she discovered that adults today spend less than an hour preparing meals, compared to at least two hours in 1965 [2]. Based on this information, the author identified busy adults as the ideal target audience for the app.

As part of the analysis process for the problem of people neglecting cooking, the author considered several alternative approaches. For example, creating a recipe website. While this approach would have provided people with recipes and cooking inspiration, it would not have provided the convenience and organisation features that an app can offer. The author determined that an app would be the best approach for addressing the problem, as it could provide a convenient and easy-to-use platform for searching and organising recipes, as well as suggesting meals and creating shopping lists. The other approach that was considered has been to focus on creating a meal delivery service app. Offering ready healthy meal options could have solved people's lack of time to prepare their own meals. However, after careful consideration, it became clear that this approach was not necessarily addressing the root issue - individuals are reluctant to invest time and energy in preparing their own meals.

Once it was decided to develop a mobile application, the author did research on available recipe apps and noticed that many of them offered some meal options but could not add personalised recipes, like, for example, the Cookidoo app [3]. Additionally, most recipe apps demand a membership plan, which requires the users to pay to use the app. The author was inspired to create a free recipe app and assessed the key features and functionality that the app should include to meet the needs of the target audience. These features included the ability to search through already installed recipes, filter recipes by categories, suggest a meal for the day, allow users to add their own recipe ideas, create collections to manage and store favourite recipes, and create a shopping list to store ingredients and have it saved for later.

It was ensured that the development process was aligned with the needs of the target audience by analysing the problem and identifying the key features and functionality required for the app, as one of the purposes of this application is to help busy adults easily find and prepare healthy and delicious meals, even with a busy schedule.

### 1.4.1. Use Case Diagram

Since the author decided to follow the Scrum framework to develop the project, she created a use case diagram as a tool to visualise the interactions between the users and the software system. The use case diagram helped to define the functional requirements of the application, ensuring that the user's needs were well understood. By identifying the

essential use cases, the author was able to create a prioritised product backlog for the project during initial planning, which included a list of user stories which were used to guide the development work during each sprint.

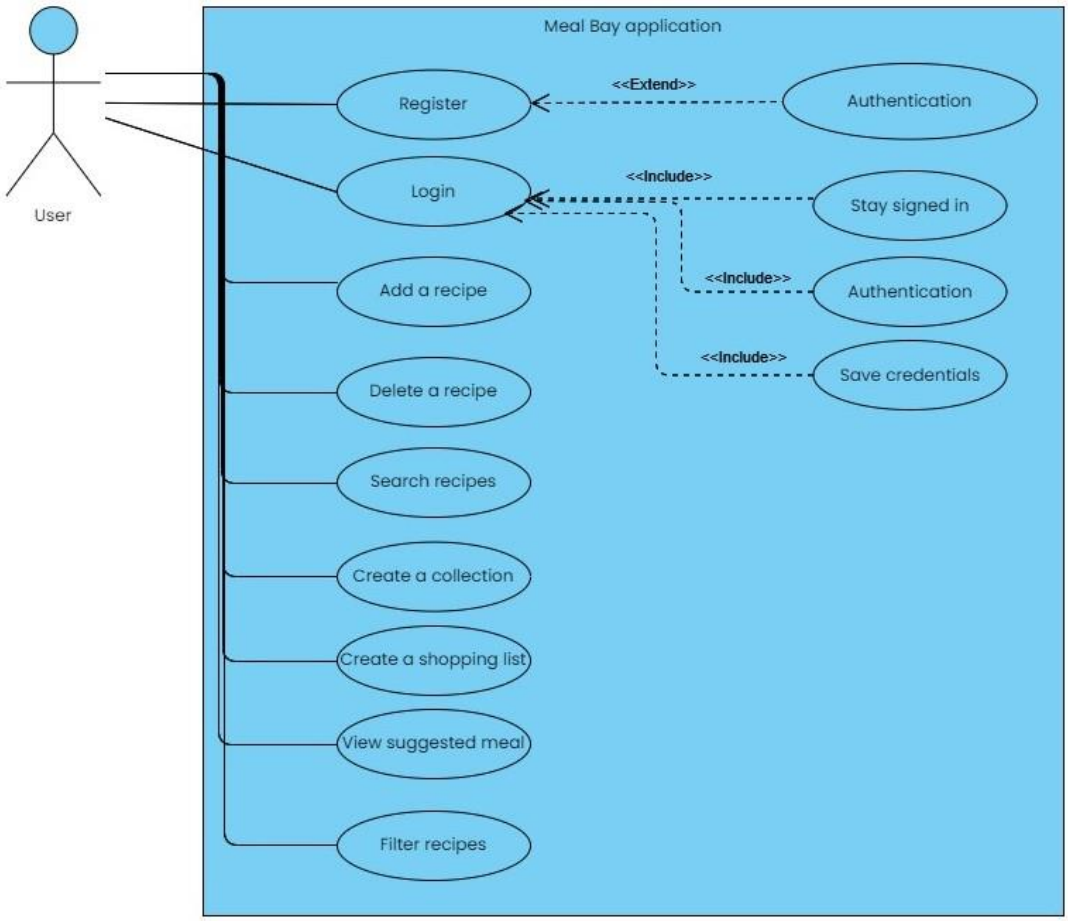



*Figure 1 - Use Case Diagram*

### 1.4.2. Functional Objectives

The end user is expected to experience these functional requirements in the app:

FR1. Search through already installed recipes to find specific meals.

FR2. Filter the recipes database by category buttons.

FR3. Create personal recipes private to the user.

FR4. Create a private collection of recipes.

FR5. Add any recipe to the selected collection.

FR6. Delete created recipes or collections.

FR7. Display the meal for the day that is different from any other day based on a local date.

FR8. Create a private shopping list and add items to it.

FR9. Clear the shopping list.

### 1.4.3. Technical Objectives

These are the technical requirements expected to be covered in the app:

TR1. Create a user and save the user in the database.

TR2. Authenticate the user given their email and password.

TR3. Retrieve and save recipes, collections, and shopping lists in the database.

TR4. Create concise documentation of the code.

TR5. Design a practical User Interface to maximise User Experience.

### 1.4.4. Requirements as User Stories

The author has chosen to use Agile Scrum methodology for this project and created user stories which are detailed and available in Appendix A.

## 1.5. Process

As the only person in the team, the author found it important to employ the Scrum [4] framework to complete complex tasks in an Agile [5] manner to deliver functional software in manageable increments and continuously enhance the quality of her work. The work was broken down into short iterations called sprints, which lasted one week for this project. Before each sprint, the detailed product backlog was defined, and tasks or user stories were prioritised based on their value.

Daily stand-up meetings were held to track progress, study any issues, and a digital Kanban board was used to manage and visualise the project workflow. Tasks that were committed to each sprint were worked on throughout the sprint.

At the end of each sprint, the author conducted a sprint review in the form of a progress report document [6] to showcase the completed work and gather feedback while reviewing it during a weekly meeting with supervisor Chris Loftus. After each weekly meeting, the author performed a sprint retrospective session to reflect on the sprint and identify areas for improvement.

There were several other software development methodologies considered, including Extreme Programming (XP) and plan-driven methodologies like Waterfall. XP emphasises rapid feedback, incremental changes, and continuous delivery of quality work [7], while plan-driven methodologies prioritise planning, documentation, and sequential execution [8].

Compared to Scrum, XP may be more suitable for small teams that need to respond rapidly to changes in requirements and produce working software quickly. Nevertheless, the author decided to use Scrum for several reasons. First, the project involved complex tasks that required close collaboration and communication with the supervisor. Second, the author was the only person in the team, and Scrum provided a structured framework [9] (see Figure 2) for managing the project and breaking down tasks into manageable increments. Third, the author wanted to continuously enhance the quality of her work, and Scrum provided a framework for continuous improvement through sprint retrospectives and feedback from the supervisor.

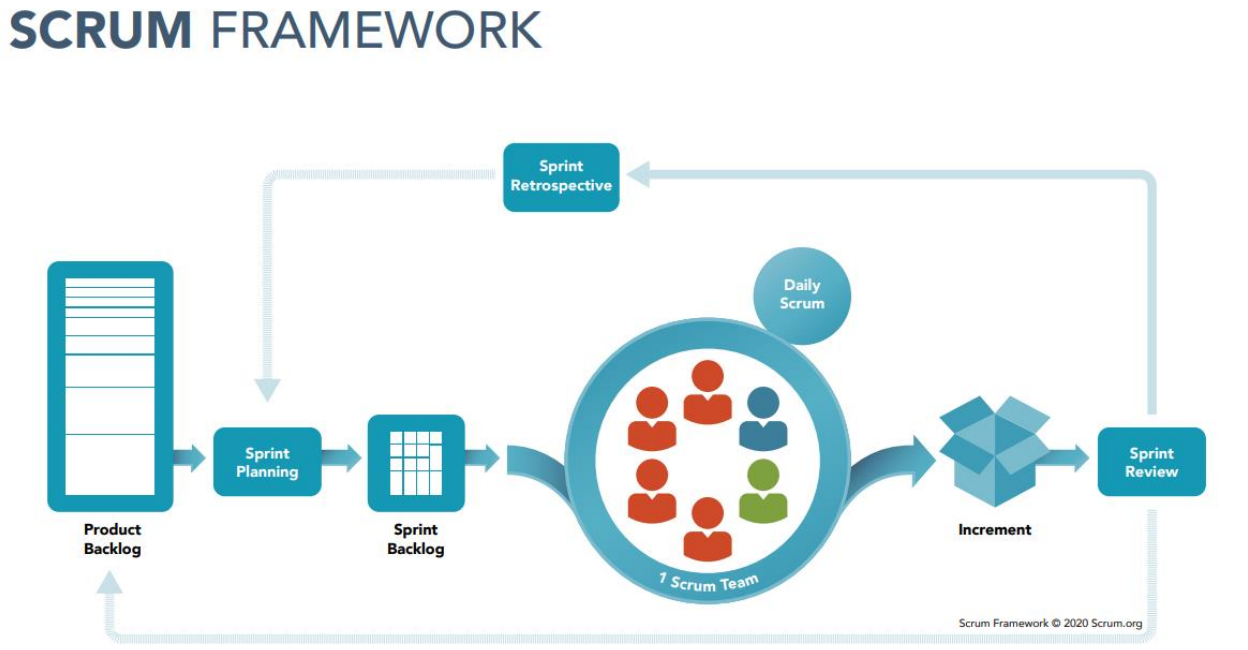



*Figure 2 - The Scrum Framework [9].*

### 1.5.1. Upfront design and iterations planning

The author of the mobile app has adopted the Scrum framework and has carefully followed the planning and iteration process to ensure the successful completion of the project. She began by conducting an initial backlog, during which she defined the key goals that needed to be achieved and the timeline for completing each goal. She then broke down the functional requirements into smaller stories, each of which could be completed in a single sprint.

To further streamline the process, the author divided each sprint into one week, starting on Thursdays, and created "issues" on Kanban Board to visually track the progress of each task. These were being worked on throughout the duration of one sprint, and the author was updating the board daily while reviewing the progress and potentially identifying new issues. Scrum focuses on delivering high-quality software while incorporating a continuous development practice, and it was carefully followed during the sprints.

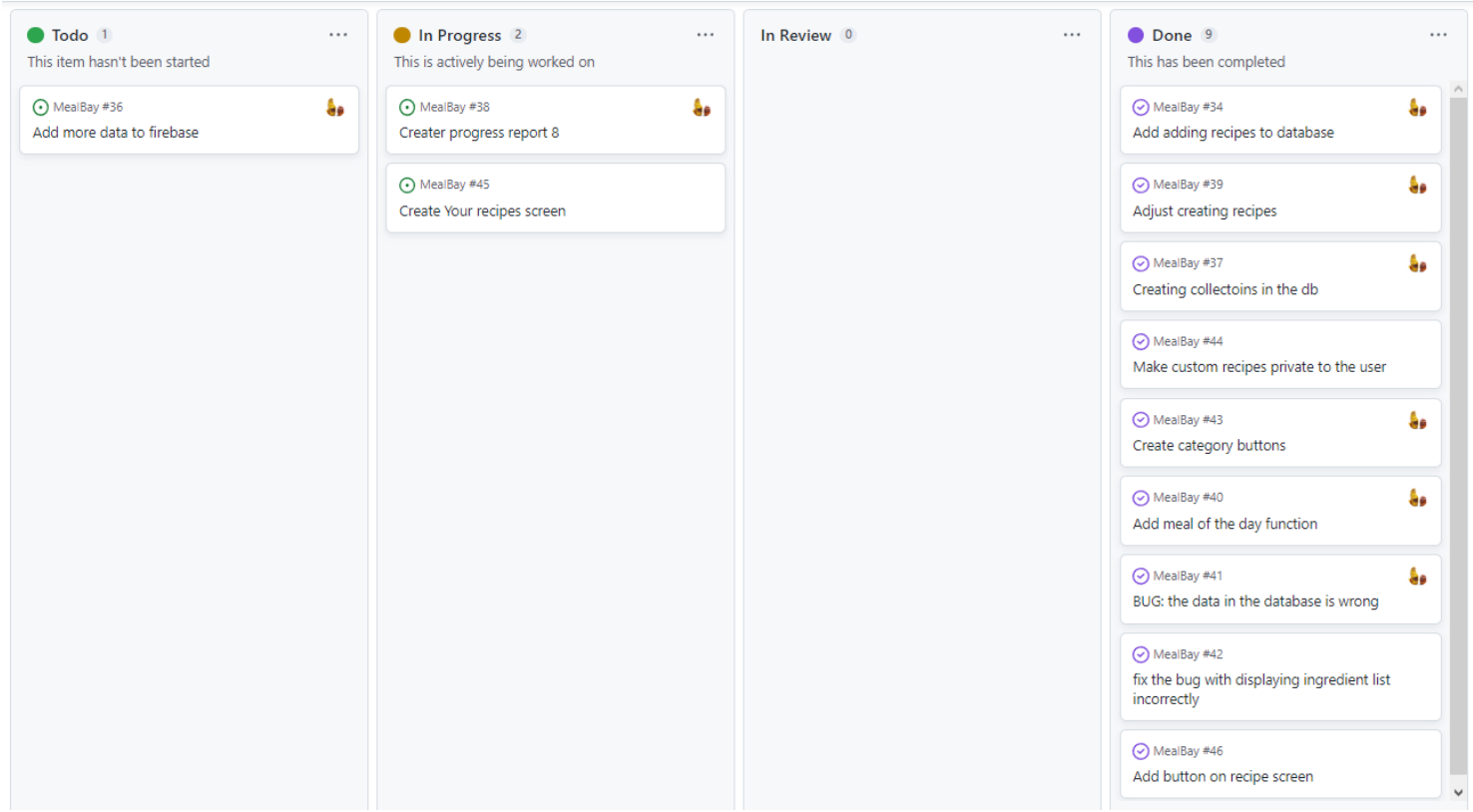
The author has been meeting with a supervisor on a weekly basis, where the progress report document was produced and updated every week. The meeting was a form of a sprint

review and helped the author to gather feedback, which was then taken into consideration during the sprint retrospective.

### 1.5.2. Kanban Board

In developing the Meal Bay mobile app, the author implemented Kanban using a digital board accessible on GitHub. The board consisted of columns labelled "To Do," "In Progress," "In Review," "Done," and "Blocked." Using a Kanban board, she increased her productivity by decreasing the time spent in progress. This is because the Kanban framework emphasises limiting work in progress and focusing on completing tasks before starting new ones. Additionally, the Kanban board allowed her to easily track the progress of each task and identify any issues that needed to be addressed.

Using a Kanban board with the Agile Scrum methodology was a good decision for this app development project, as it helped the author stay organised and focused on completing tasks conveniently and efficiently. Creating the board also gave the supervisor a visual representation of the workflow. Here is an example of the Kanban Board representation during one of the sprints:




### 1.5.3. Retrospectives

To make the most of retrospectives, the author has been meeting her supervisor at the end of each sprint. During these meetings, they reflected on the work, identified areas for improvement, and explored opportunities to simplify processes. After each weekly meeting with the supervisor, a retrospective was organised using Scrum with a Kanban board. This approach allowed the author to tailor any work processes to the feedback received during the meeting with the supervisor while also using the benefits of Scrum and Kanban to

manage workload and track the progress during the sprint. She was able to deliver high-quality work and continuously improve the processes with each sprint by combining these approaches.

#### 1.5.4. Adjustments and alterations implemented

As a solo developer, it was difficult to use the Scrum since it's meant for teams. However, adjustments were made to adapt the framework for one person. This included having daily Scrum meetings alone, sprints lasting one week, and fulfilling all Scrum roles independently. To help with the retrospectives, the author arranged to meet with a supervisor at the end of each sprint, in addition to conducting their own retrospective. The supervisor was regularly providing feedback on the work completed, pointed out potential areas for improvement, and provided an additional perspective on the author's work. Altering the Scrum framework in this way allowed the author to benefit from the structure of Scrum and continuously improve the project, even when working alone.

### 1.6. Tool selection

#### 1.6.1. Operating System

The author considered developing for both Android and iOS but ultimately decided to use Android as the operating system platform for Meal Bay mobile app development. A major advantage of iOS is its reputation for stability and security, which is attractive to many users. However, Android operates on an open-source ecosystem, making Android developers easily build an Android app [10]. Android offered more freedom and customisation options [11], which was important when building an app with specific features and functionalities the author desired to develop. The author also had prior experience creating Android App as part of a CS316 module at Aberystwyth University, making her more familiar with its development environment. It was decided that Android was the best choice for app development needs, and it has proven to be a reliable operating system [12] for building high-quality mobile apps.

#### 1.6.2. Programming language

The author has analysed the advantages and disadvantages of the two most widely used programming languages for Android app development: Java and Kotlin. After careful consideration, the author aimed to achieve optimal results in the app development process and decided to use Kotlin programming language due to its null safety and built-in coroutines support.

The app uses the 1.8 Kotlin version with several additional libraries and plugins, including androidx, Firebase, Jetpack Compose, Hilt, Coroutines and Coil for image loading.

#### 1.6.3. UI toolkit

The use of the Jetpack Compose UI tool was selected to develop the Meal Bay app along with following Material Design 3 guidelines [13]. Other options were considered while choosing the above such as Android XML Layouts or RecyclerView UI tools. However, having in mind the objectives of the Meal Bay project, the screen design that does not contain large data sets in a scrolling list format, Compose, was selected instead. Jetpack Compose provides a more straightforward and intuitive way to create User Interfaces, including lists, with less code and greater flexibility. The tool offers improved support for modern app

architecture patterns, simplifying the development and maintenance of apps as they become more complex over time.

The author used Google's design language, Material Design 3, to create visually appealing and user-friendly UIs. It was selected because it offers pre-built UI components and colour schemes that add depth and dimensionality to the UI, leading to a more enjoyable user experience.

#### 1.6.4. Development Environment

The author created the app using the Android Studio IDE [14]. This widely-used development environment offers a range of tools for Android app development and access to the Android SDK, which contains various APIs and libraries for building Android apps. With these features and more, the author decided that this IDE is the best choice for developing Meal Bay app to meet desired technical requirements.

#### 1.6.5. Version Control System

Version control is a useful tool for software development, whether you work alone or as part of a team. In this project, the author chose to use a VCS because it allows for storing each version of the codebase in a repository and tracking changes made to the codebase over time, among other benefits.

The author considered several Git servers, including Bitbucket, GitLab, and GitHub, with prior experience using each. After careful evaluation, the author eventually decided to use GitHub [15] for its simplicity and ease of use during the project's development.

#### 1.6.6. Diagram tool

All diagrams presented in this document have been created using a free online tool called Drawio [16].

# 2. Design

## 2.1. Overall Architecture

At the system-level architecture, the user interacts with the app by logging in and using the app to read, save or edit data. The app uses Firebase as its backend and has a database to store and retrieve data. Firebase Authentication ensures secure access to the app's features and functionality. Once a user successfully logs in, the app communicates with the Firebase database to retrieve their saved data, such as recipes, shopping lists, and private collections. The app's architecture guarantees secure storage of user data, which can only be accessed by the user who created it. The overall system's architecture prioritises secure user authentication, user experience optimisation, and, most importantly, ease of use for the app.

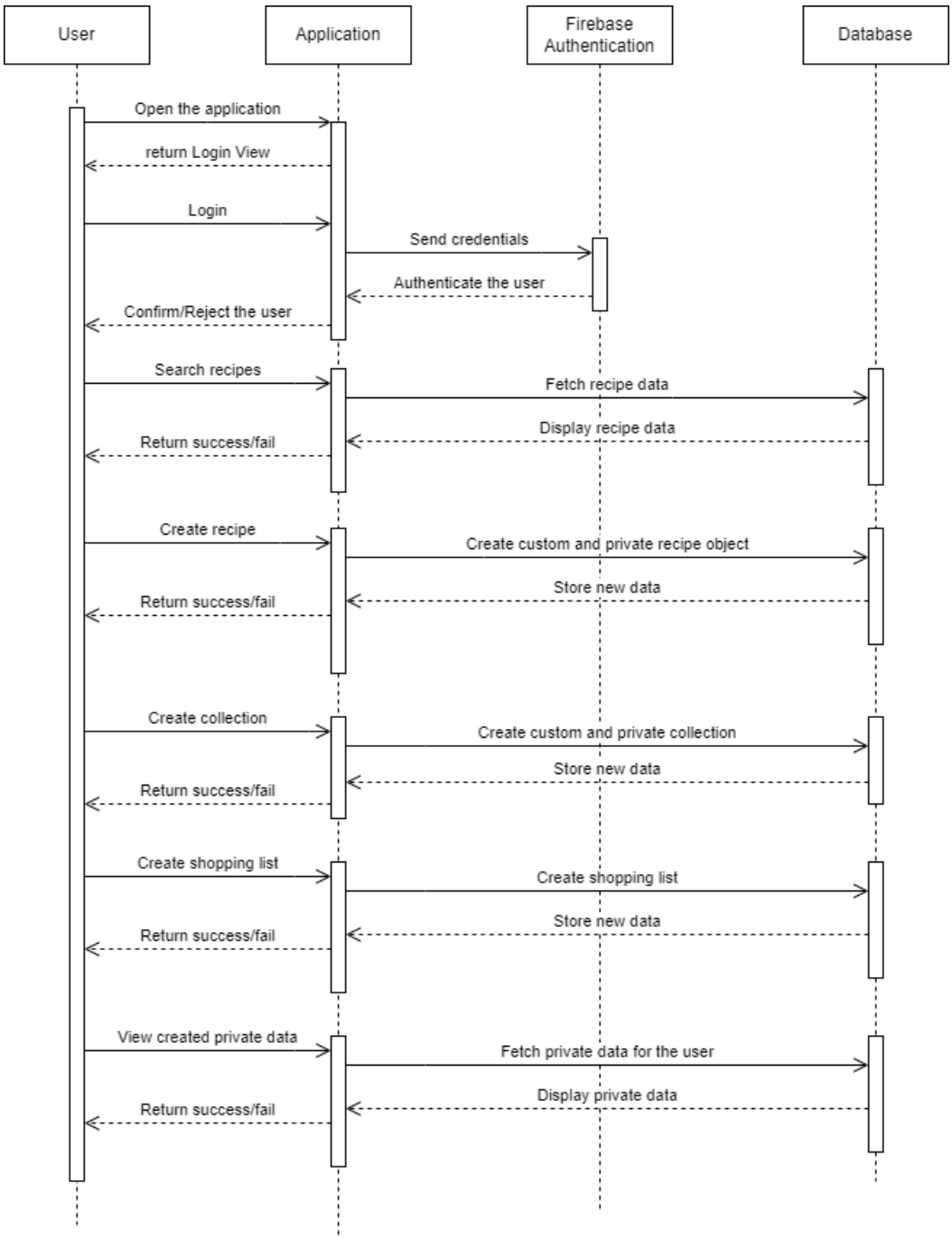



*Figure 3 - System-level sequence diagram*

## 2.2. High-level design

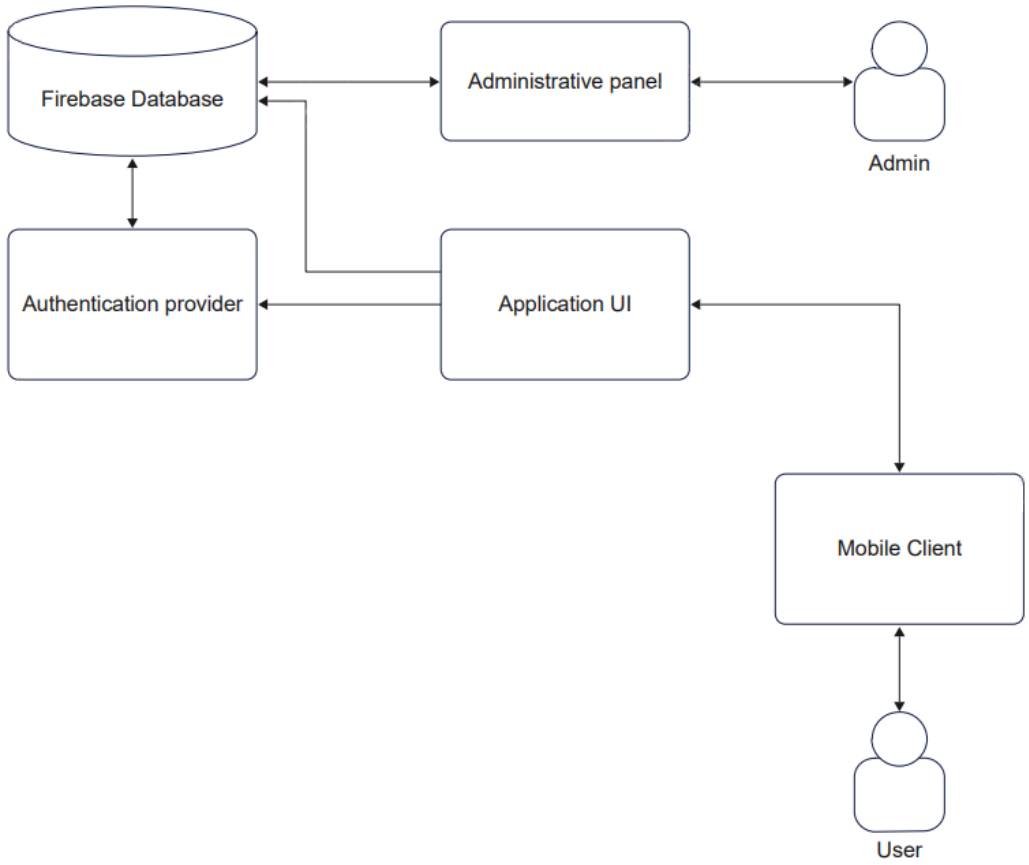



*Figure 4 - High-level design*

The app has a client-server architecture, with the mobile phone as the client and the Firebase database as the server. The database stores all user-related data, including recipes, shopping lists, and private collections. The developer can access the Firebase Console to manage the database. From there, they can view, edit, or delete user data. Firebase handles the authentication process to ensure users can securely log in to the app. To create an account, users must provide an email address and a password that meets specific requirements. The password must have at least six characters, and the email address should have the "@" symbol and a "." before the domain.

Users interact with the app's features and functions through the mobile client. The interface is simple to navigate, managing local data and handling interactions with backend services such as authentication and database. The app uses Firebase Cloud Messaging (FCM) and a connection server to communicate with the Firebase database. This forwards messages to the mobile device running the app with the help of APIs provided by Firebase. The app is also designed with the Model-View-ViewModel (MVVM) pattern, which separates the data (Model), user interface (View), and connection between the two (ViewModel) [19].

### 2.2.1. Other considered design

The author explored an alternative, offline-first approach [20] before fully implementing the design discussed in section 2.2. This design enables the application to function even when the user is not connected to the internet, such as when the user has an unstable network connection. This method is used for developing apps with offline capabilities. This approach

stores data locally on the user's device using a local database or storage system. When the device is online, the app periodically synchronises this data with the server to ensure the local copy is current.

If the author opted for this design, the users could browse the recipes and access their saved recipes and shopping lists with limited data access. The author was worried about storage limitations that could prevent users from using the app on their mobile devices, leading to a negative user experience. In addition, users most likely will use the app at home, where the internet connection is typically reliable, so the author realised that an offline-first design would not be very beneficial and therefore decided against it.

## 2.3. Database Design

Careful consideration of the application's data needs and the relationships between different data objects were required to design a good database. In particular, the decision to use Firebase, a NoSQL database, was made because of its powerful and flexible cloud-based database service for mobile applications. Hence, it was used as the main database in the app. Unlike traditional relational databases, NoSQL databases like Firebase do not use a fixed schema, meaning that data can be stored more flexibly [21]. This makes adjusting the data structure and expanding the database easier as the application grows with NoSQL databases. When dealing with unstructured or semi-structured data within the application, NoSQL databases are an excellent option due to their ability to handle such data efficiently. However, to prevent duplication or inconsistencies, a well-planned data model is necessary when using NoSQL databases.
The Meal Bay app found Firebase's NoSQL database an excellent choice, even when faced with challenges. The database proved to be highly reliable.

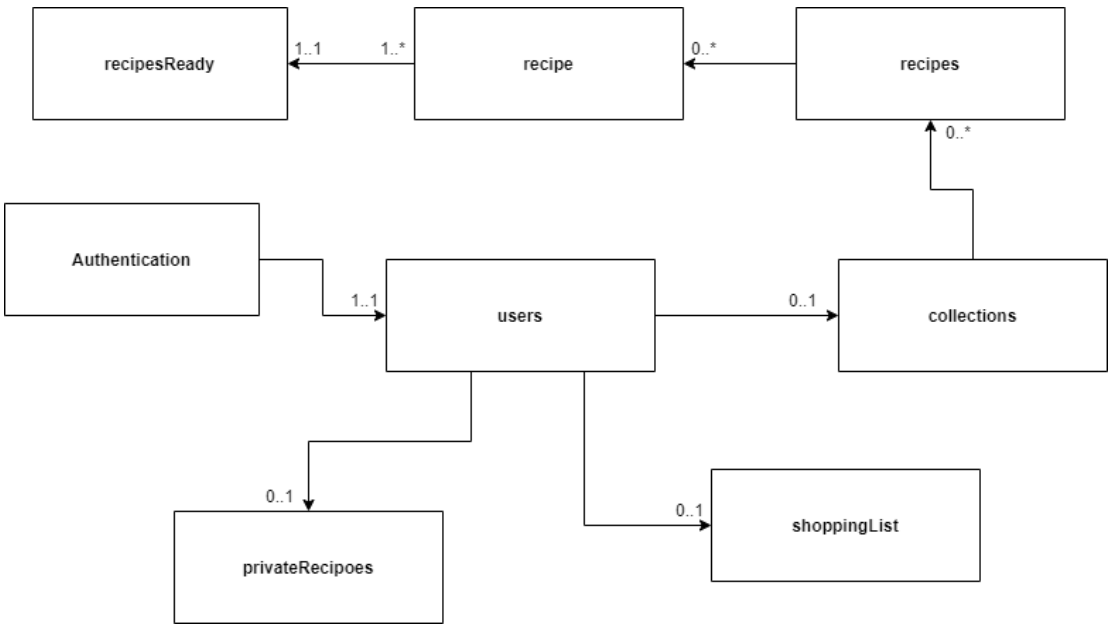



*Figure 5 - Database design.*

The author has decided to create a public "recipesReady" collection that contains all the publicly available recipes. This collection contains recipe documents, each including the following fields: title, total_time, ingredients, preparation, category, difficulty, rating, and photo (URL). The database has a separate collection called "users", which includes userIDs where each userID matches only one user.

When the user document is created, it has no private collection by default. These can be created depending on the user's behaviour in the app. The database will automatically create a suitable collection if the user creates any collection, shopping list or private recipe within the app. This approach promotes a lazy loading practice which is the process where "an entity or collection of entities is automatically loaded from the database the first time that a property referring to the entity/entities is accessed" [22]. The author decided to follow lazy loading early since the first step of the database design because it will be beneficial when dealing with large data. This practice prevents slowing down the app or fetching documents from the database when it grows (for instance, with new user accounts). This approach also helps conserve resources by not creating unnecessary data, reducing the initial loading time and the amount of memory the application uses.

When creating the database, it was important to consider how the collections were related. For instance, every item in a person's "shopping list" is connected solely to that user. This could be utilized to create personalized shopping lists that align with the user's preferences. While personalized content was not a functional requirement for this project, the author followed proper procedures in case there was a need to expand and improve on the project later on.

As a good database practice, the author normalised the data by breaking it into smaller, more manageable pieces. This process helped to reduce data redundancy [23], making the database more efficient. Additionally, the author avoided storing large binary data such as images in the database directly, instead opting to store only the URL to the image. This approach helped reduce the database's overall size and improve performance. To ensure the security of the data, the author set appropriate access controls and security rules. Such rules help to restrict unauthorised access and prevent malicious attacks on the database. The Meal Bay's database uses Firebase authentication to guarantee that only authorised users can access and modify their data. This added security feature helps prevent any unauthorised access.

A good database design is believed to be essential for an application's optimal performance. The database serves as the backbone of this application, and neglecting its design could lead to performance issues or security vulnerabilities. Therefore, the author prioritised the database design and followed best practices.

## 2.4. Detailed Design

### Model-View-ViewModel pattern

While creating this mobile application, the author decided to use the Model-View-ViewModel (MVVM) pattern as the architectural pattern. The author believes that MVVM is the most suitable option for this application because it is a better fit for the data binding capabilities of Jetpack Compose, which was used for building dynamic and responsive UIs.

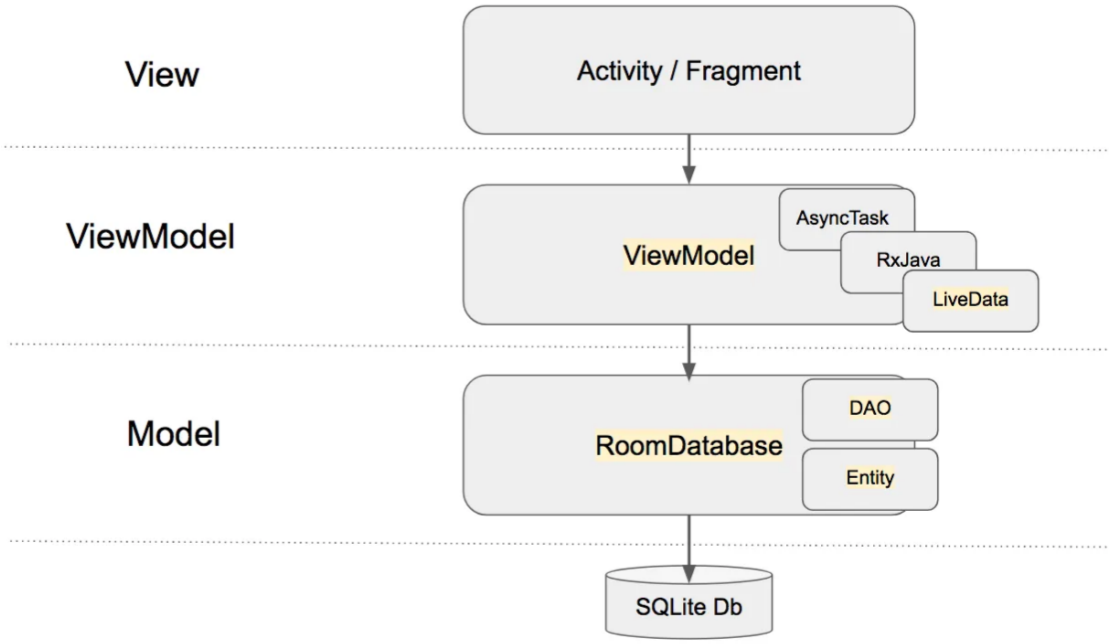



*Figure 6 - Model-View-View-Model diagram [24]*

In the MVVM pattern, the Firebase Cloud Database serves as the Model layer, responsible for data storage and retrieval needs, while the Jetpack Compose UI toolkit represents the View layer. The Firebase Cloud Database was chosen as the backend for the application's data storage and retrieval needs and Firestore Authentication to authenticate and manage user sessions. The view layer, instead, is responsible for presenting the app's user interface (UI) to the user. The View layer (Jetpack Compose) is a declarative UI toolkit that uses composable functions describing the UI elements and their layout. Composable functions are simple and intuitive and allow us to build and customise UIs. Jetpack Compose is also compatible with various Android libraries, for instance, Material Design and ViewModel, which are widely used in the project. The ViewModel layer, which serves as a Controller, handles the communication between the View and Model layers, managing business logic and data retrieval. The hilt Dependency Injection framework is used to instantiate and inject dependencies into components like ViewModels and Activities. One of the ViewModels used in the app is MealViewModel which handles the retrieval of recipe documents and manipulation of data from the Firebase Cloud Database.

LoginScreenViewModel is another class that extends ViewModel in the app and handles signing in and creating a new user by communicating with the model layer through the Firebase API to perform authentication and database operations.

The author considered other potential patterns, such as Model-View-Presenter (MVP) pattern. The MVP pattern is similar pattern to MVVM. The main difference is that MVP only specifies how to structure the View instead of defining the structure of the whole system [25]. I have rejected this pattern as a potential software pattern for the Meal Bay app because Model-View-Presenter is not well-suited for data binding [26], an essential feature of Jetpack Compose.

I believe that using a software architectural pattern, such as MVVM, is necessary while creating software because it provides a framework for organising and managing the

complexity of the software. This pattern helped me understand the best development practice, as it separates the core concepts and functions from the implementation details. Finally, a pattern promotes maintainability, performance, modularity, and encapsulation. Therefore, I decided to use the MVVM pattern, which is crucial for building effective and maintainable software over time.

## 2.5. User Interface Design

### 2.5.1. User Interface prototype

At the beginning of the development timeline, the author created a UI design prototype using Figma [27]. This allowed the author to sketch a visual representation of the app's interface, including its layout, colours, and navigation.

Designing a UI prototype in the early stages of development has several benefits. Firstly, it offers a user's perspective, enabling the author to identify potential problems early on. Secondly, it is a helpful reference point during development, saving time and effort. Moreover, based on this UI prototype, the author created user stories undertaken each week while following the Scrum development process.

The author has since used this UI prototype to build the actual UI, which as a final product, looks very close to the prototype with only a few changes made. It's worth noting that the design in the UI prototype was carefully following the Material Design 3 requirements. Building a UI prototype ensured consistency in the app's design while making it easier to implement.

### 2.5.2. User Interface enhancing tools

To boost the user experience, the author conducted research on available libraries enhancing UI. After considering various options, it was decided to use the Material Theme Library [28], given its benefits in improving the app's design. The Material Theme Library offers various customizable themes and styles that follow the Material Design system. This promotes a visually pleasing interface. Moreover, it includes pre-built components such as buttons and text fields that follow the Material Design guidelines. This makes it convenient to implement complex UI elements. Additionally, using the Material Theme Library helped to keep the app's design up to date with the best practices in UI design, which may contribute to a more enjoyable user experience. The following figures represent the colour palette generated for the Meal Bay app.

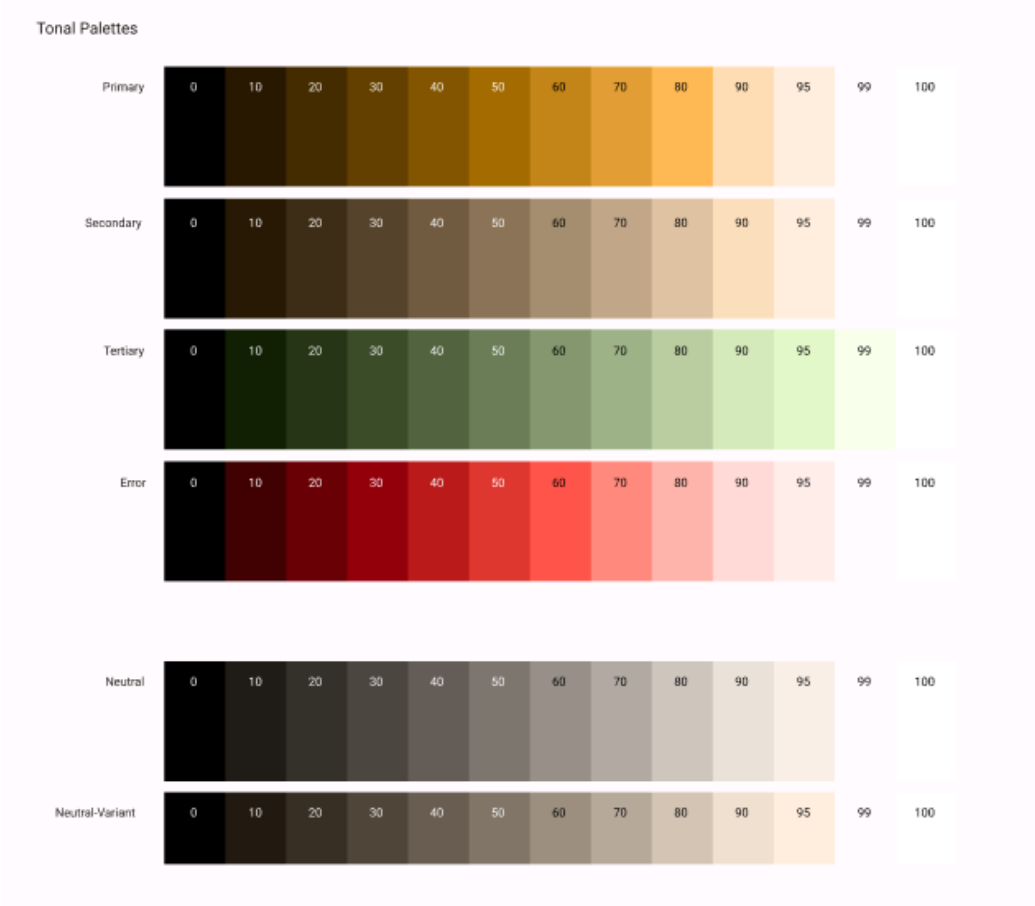



*Figure 7 - Tonal Palettes in Material Theme.*

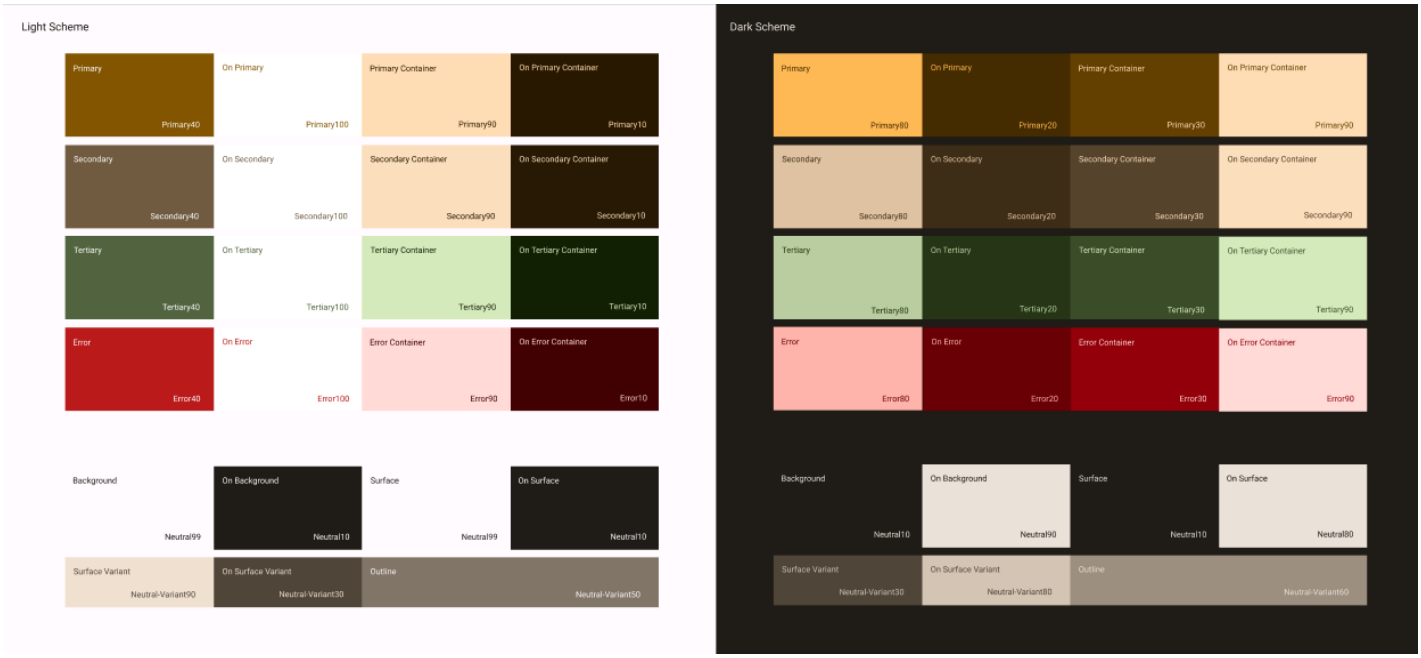



*Figure 8 - Light and Dark Scheme in Material Theme.*

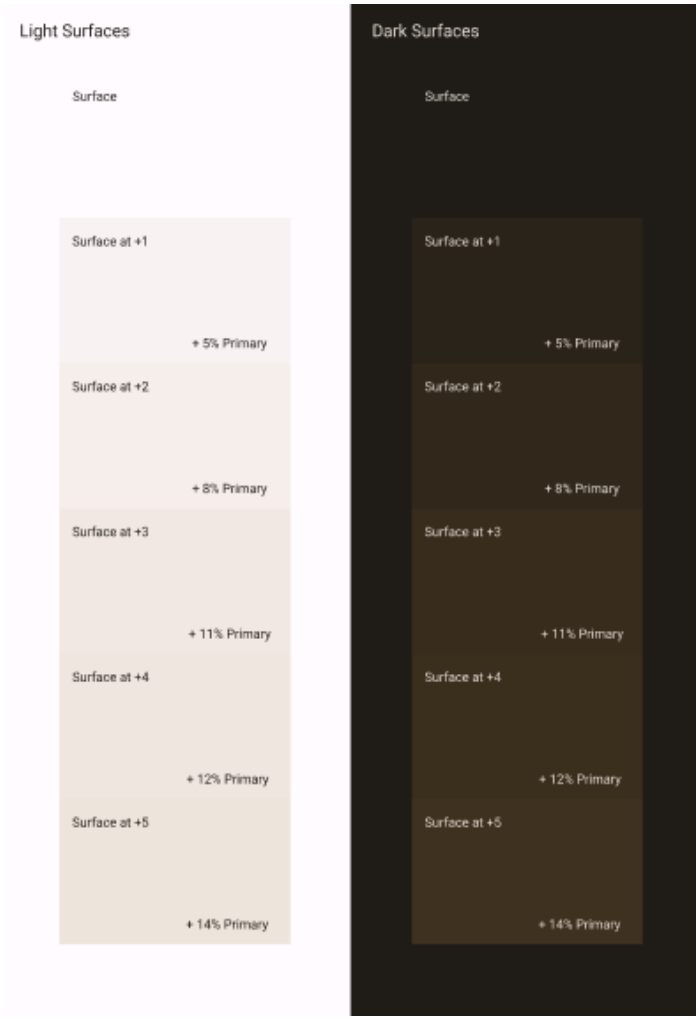



*Figure 9 - Light and Dark Surface of Material Theme.*

In addition to the Material Theme Library, the decision was made to use Google's Material Icons library [29]. This library offers a vast collection of high-quality, scalable vector icons that can be used throughout the app's interface. The use of recognizable icons in, for instance, the navigation or top bar can help enhance the clarity of the app, contributing to a more intuitive user experience. With this careful consideration and implementation of these UI-enhancing libraries, the app meets the expectations of today's modern mobile users, including the target audience.

## 2.6. Final product User Interface

The final product, User Interface, was based on the UI prototype, which was lightly refined through continuous iterations to ensure that it met the functional requirements. The UI was developed in stories following Scrum iteratively and incrementally. The UI was created using Jetpack Compose components, a modern toolkit for building native Android UIs. The use of Jetpack Compose simplified UI development and increased UI consistency. The end result is documented in screenshots of a working app below, while each figure is referenced to provide more information about the content of the screenshot.

## Login Screen

The login screen UI is the first interaction point for app users. Users must enter both their email address and password in the designated text fields on the screen to sign into their account. The text fields are user-friendly and have clear labels to ensure accurate information input. Users that do not have an account can easily create one by selecting the "Sign up" option at the bottom of the page.

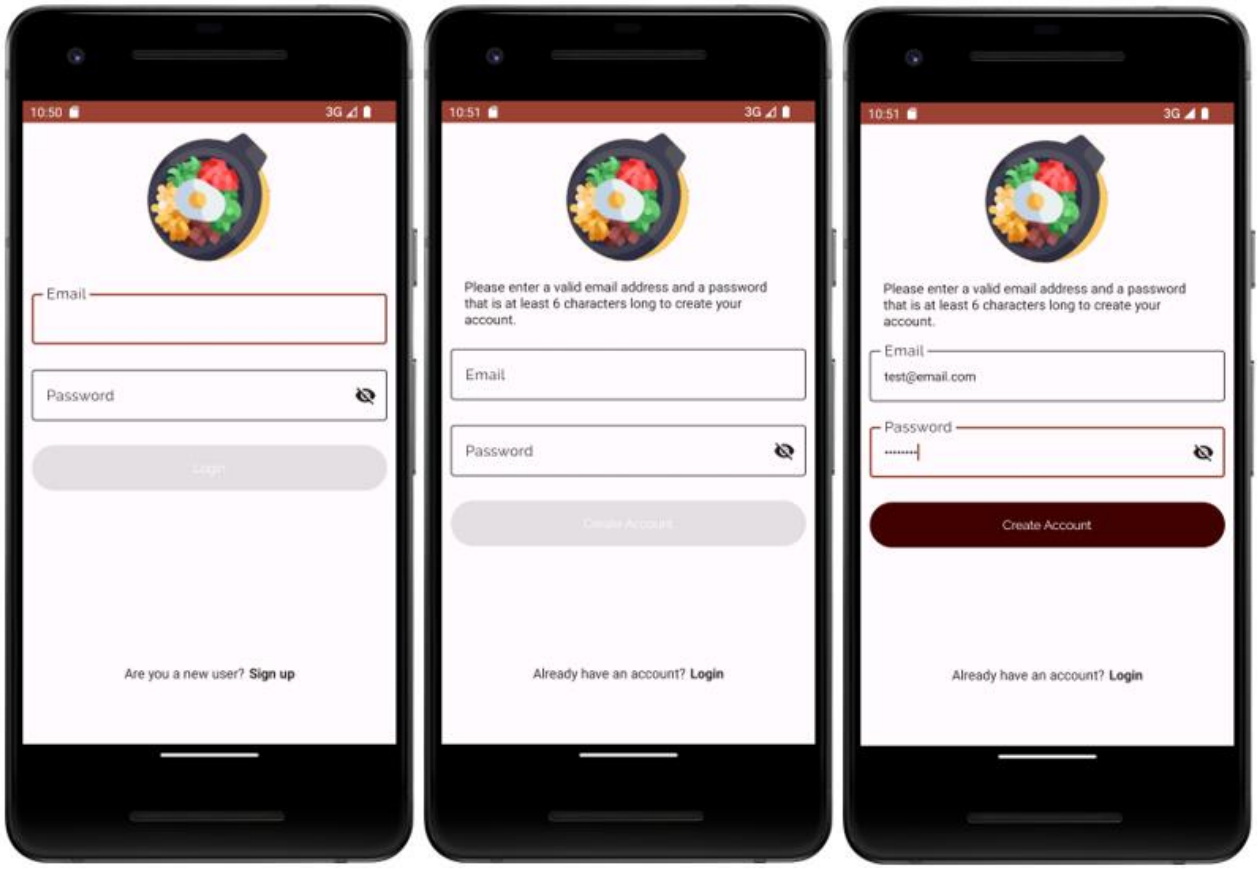



*Figure 10 - The Login Screen of the final product app.*

## Home Screen

The user can find all the vital information they need for easy navigation on the app's home screen. The home screen is scrollable and contains a drawer sheet that allows users to log out, but it is hidden to prevent unnecessary logouts. The "Meal for the Day" feature on the home screen generates a meal recommendation for the day [Figure 10]. The following home page content provides users with information on how to use the app's features. The home screen also includes category buttons that filter recipes by various categories. Additionally, users can use a navigation bar to navigate to other areas of the app, such as a shopping list or recipe list [Figure 11].

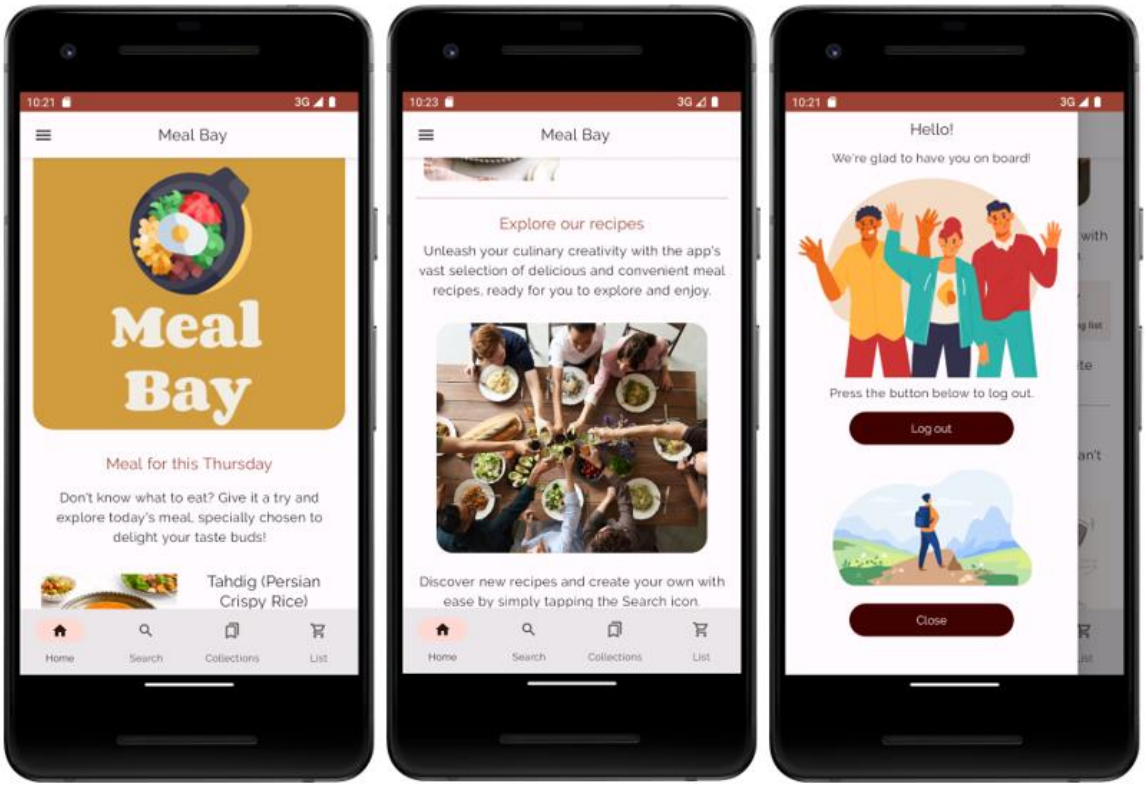



*Figure 11 - The Home Screen of the final product app.*

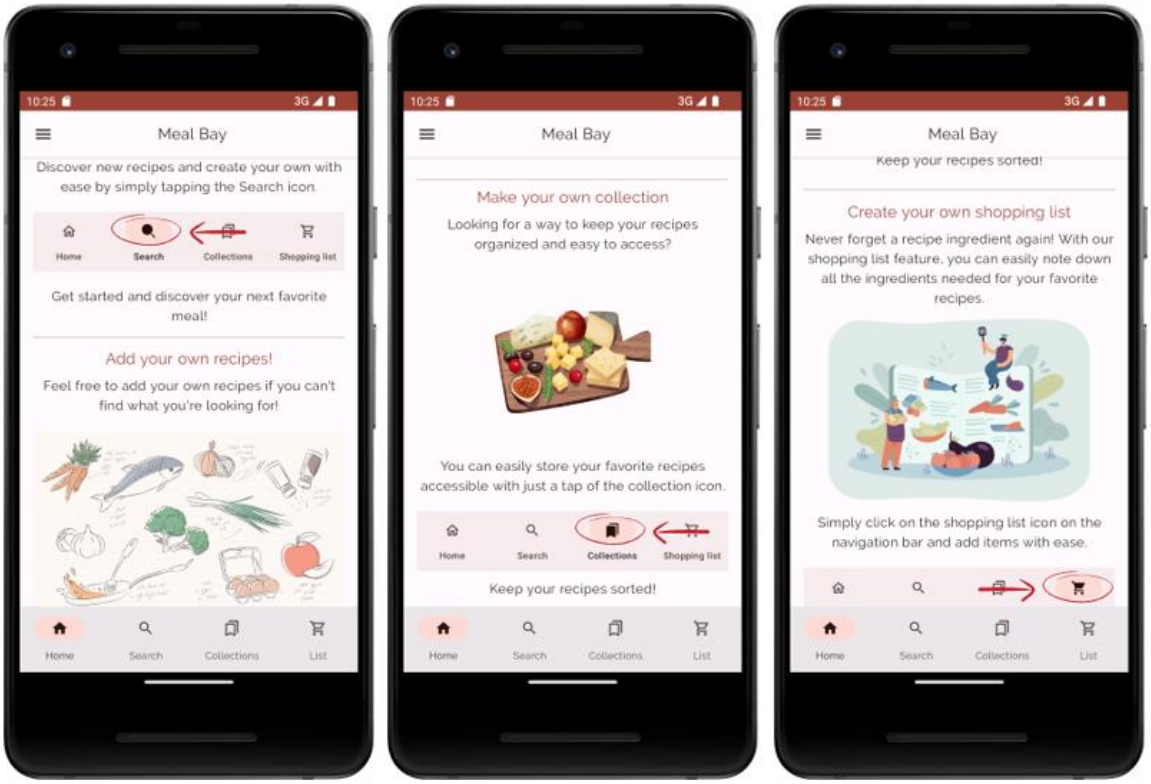



*Figure 12 - The Home Screen content continued.*

### Collection Screen

The UI of the Collection screen enables users to create and manage their recipe collections easily. Upon first visiting the screen, users will see an illustration and message informing them that they have not yet created a collection. The Floating Action Button [30] needs to be clicked to create a collection. Clicking FAB triggers a modal bottom sheet to open; it contains a text field, prompting the user to enter a name for the collection. When the user saves this collection, it will be displayed on the screen as a list. Any collection content can be seen by clicking on a desired collection. When the collection is newly created, it is empty by default, and a relevant information message appears to notify the user that this collection does not contain any recipe. However, if the collection has existing recipes; the content section will include their titles, categories, and images.

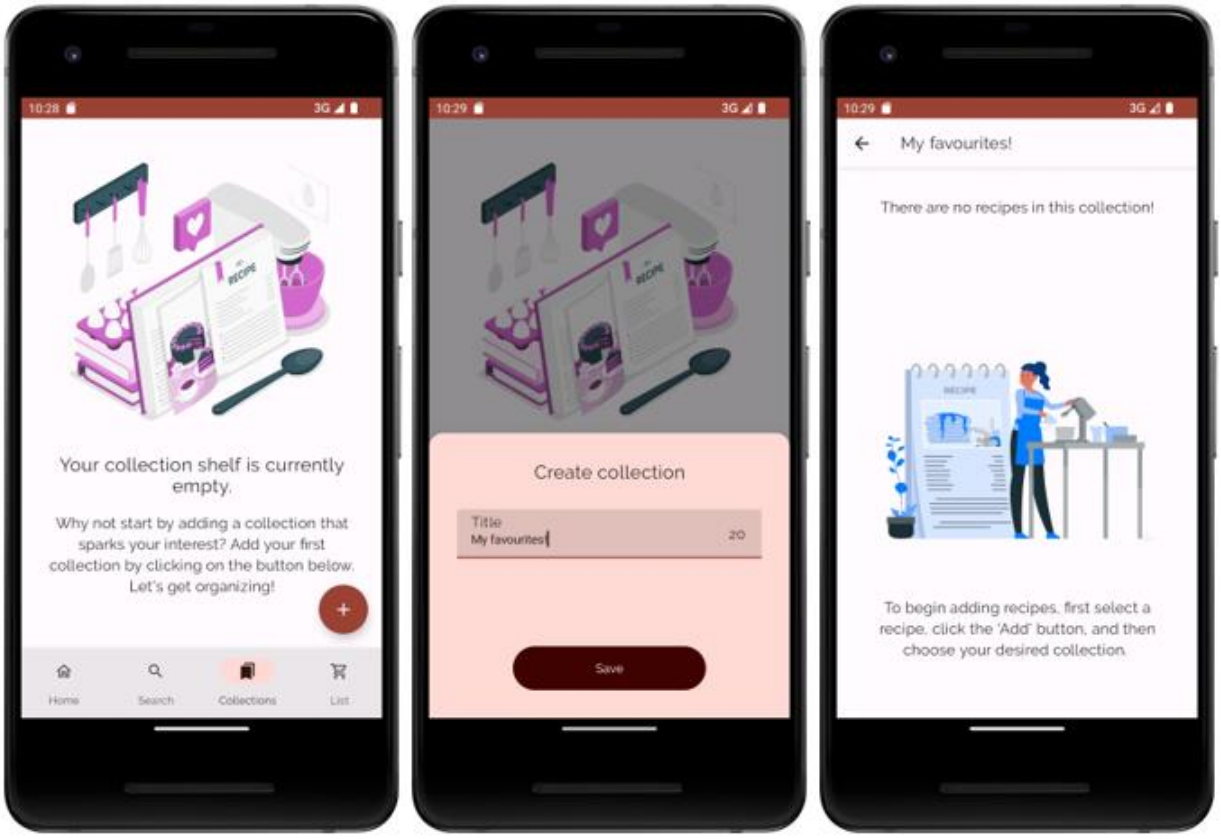



*Figure 13 - The Collection Screen of the final product app.*

### *Creating custom recipes*

The user can create unique recipes by accessing the search screen when using the app. To do that, the user must click the "Create New" button, which will navigate to a screen where the user can enter a title and the total preparation time of the new recipe. Both a difficulty level and rating for the recipe can be selected using a custom five-star system.

The user will be directed to a screen where they can input ingredients for their recipe when the "Next" button is clicked. This is done using a floating action button, which opens a modal bottom sheet where the user can add ingredient names and quantities. The data is refreshed on each item saved and displayed on the screen. The user can modify the ingredients or return to the previous screen [Figure 13].

Before proceeding to the next screen, the user is alerted the ingredients section will not be allowed to be modified after they save their progress. The next screen prompts the user to add preparation details for the recipe, such as cooking instructions and serving suggestions. Finally, the user is taken to a category screen that includes nine categories with checkboxes next to each. Only one category can be selected [Figure 14].

Once the user clicks the "Save" button, they are taken to a screen that confirms their recipe has been successfully created and saved. To view their recipe, the user can navigate to the "Your Recipe" screen of the app [Figure 15].

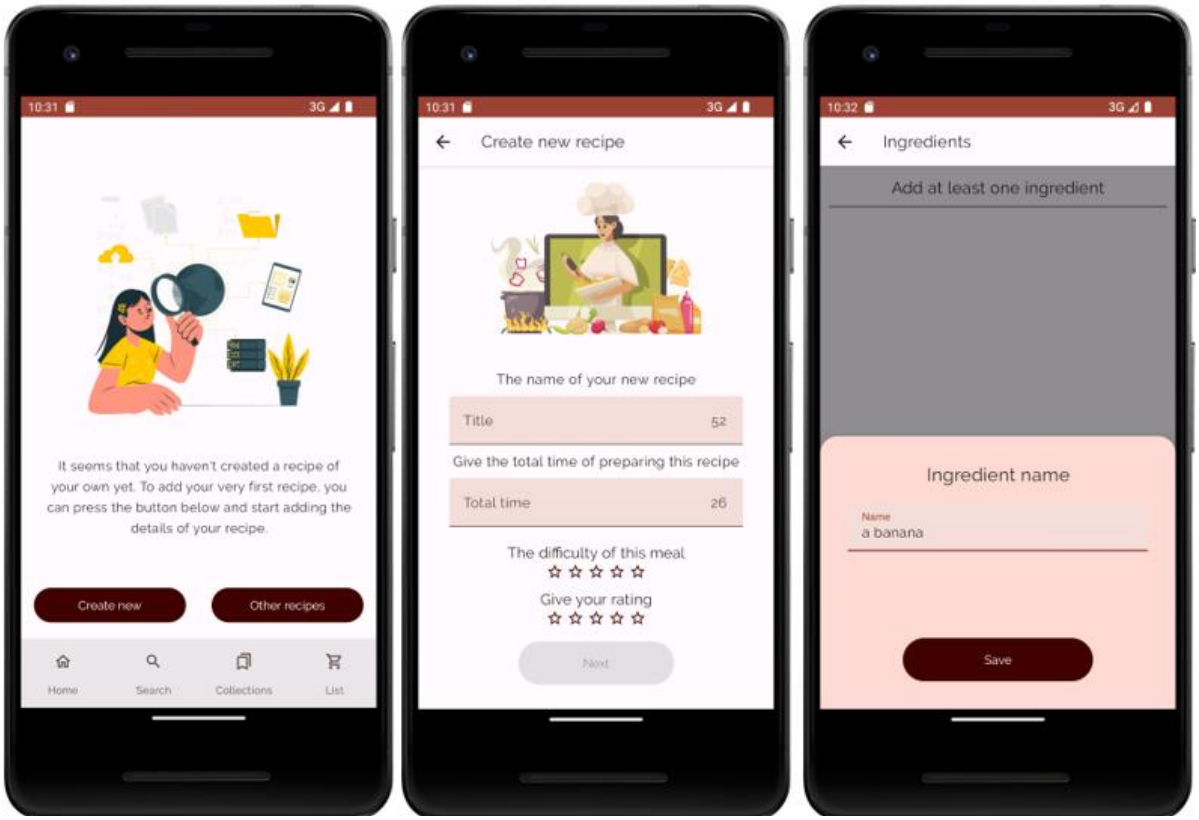



*Figure 14 - Creating new recipe section in the final product app.*

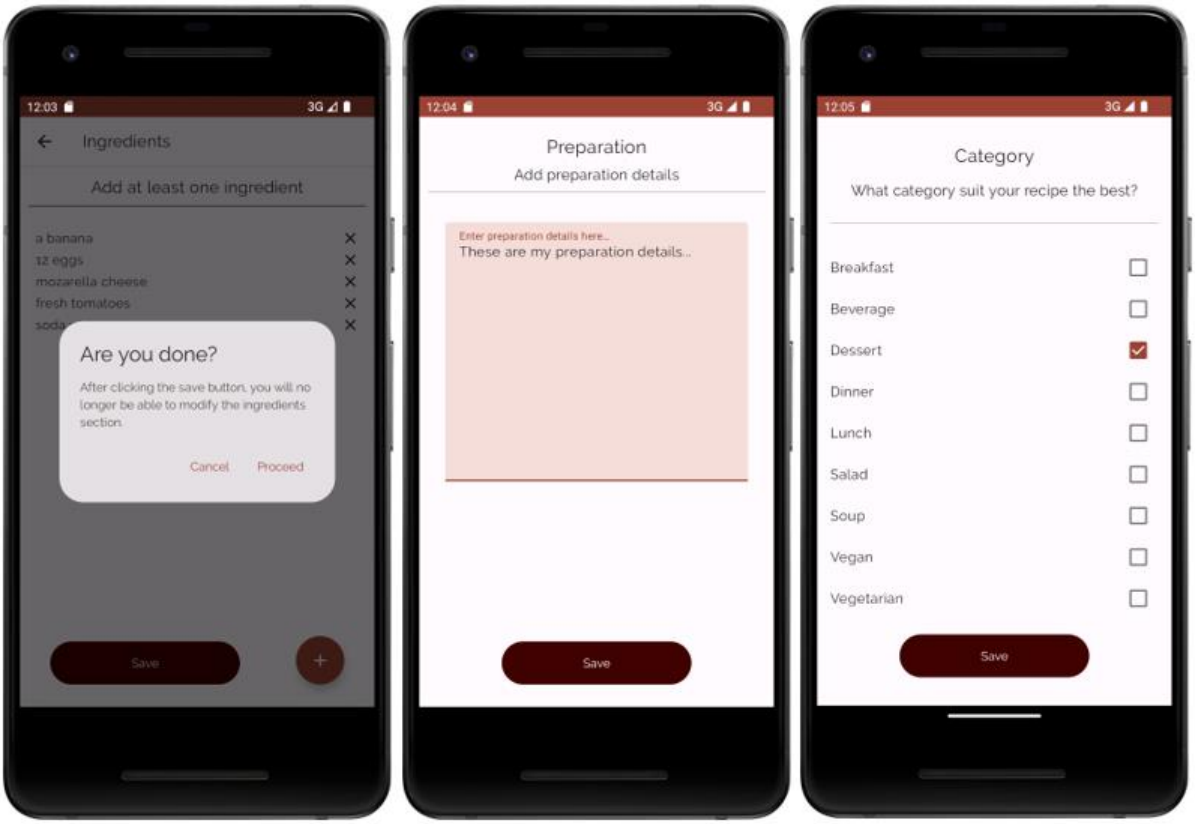



*Figure 15 - Adding ingredients, preparation and choosing category while creating a new recipe in the final product app.*

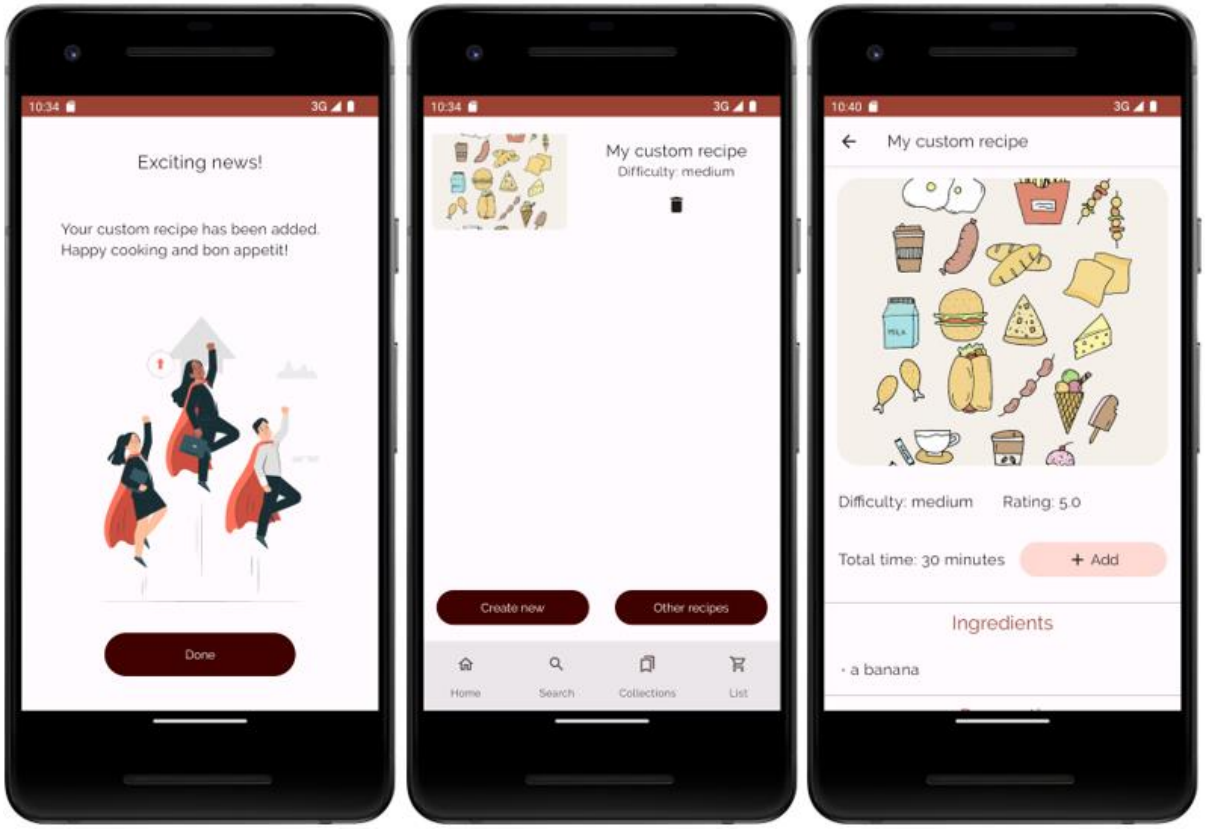



*Figure 16 - The successful recipe creation information and new recipe content of the final product app.*

### Adding recipes to collection

The adding recipes feature [Figure 16] in the app allows the user to add a recipe to a collection. To add a recipe, the user needs to find it and click the "Add" button on its page. However, the user should have created at least one collection beforehand. After clicking "Add," a box will appear on the screen, displaying the available collections. Users must select the desired collection to which they want to add the recipe. If successful, it cannot be added twice unless previously deleted. The newly added recipe will be displayed correctly in the collection it was initially saved into and will be available for the user to view at any time.

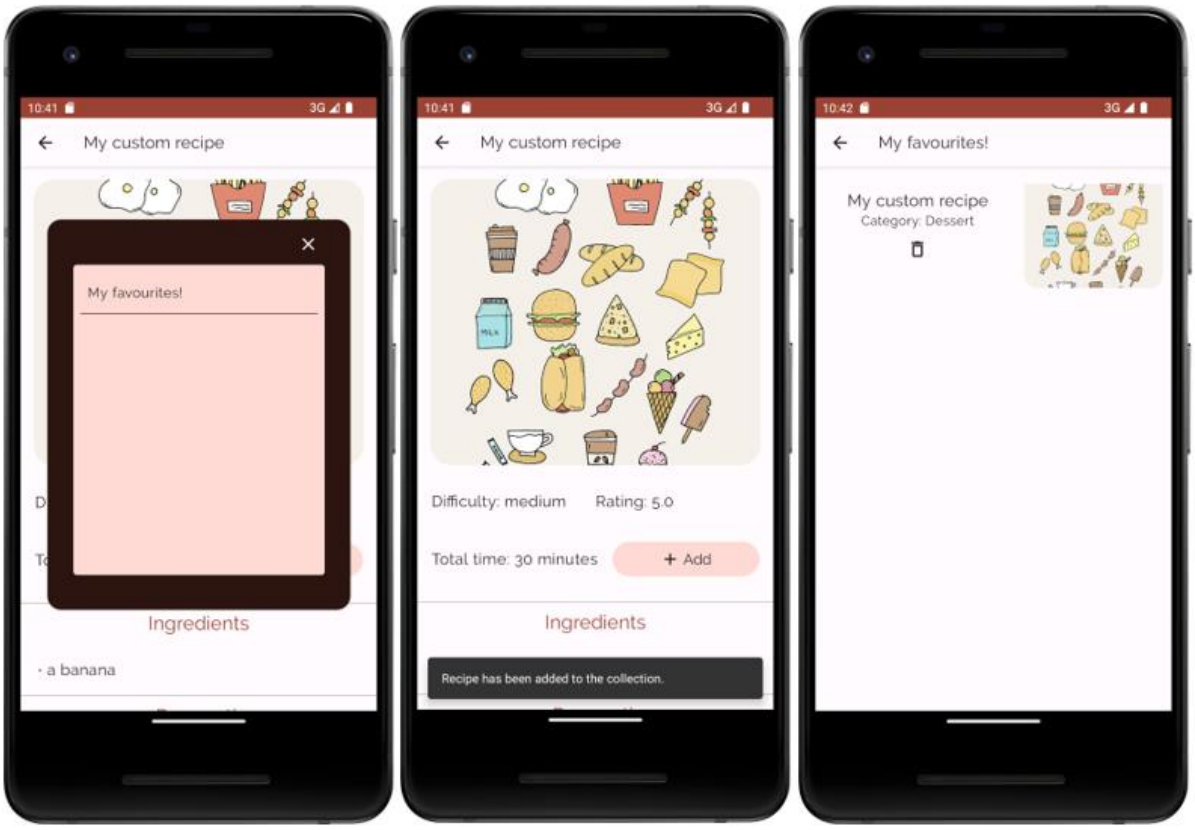



*Figure 17 - Adding recipes to the collection in the final product app.*

### Shopping list

The purpose of the Shopping List screen [Figure 17] is to help users easily manage their grocery needs. When first accessing the screen, an empty state with an illustration indicating an empty list is displayed. The user can click the floating action button to add an item and enter the name. The freshly added item will then appear on the screen. Users can click the "Clear" button if they wish to delete the list. However, an alert will pop up before the list is deleted to confirm the action.

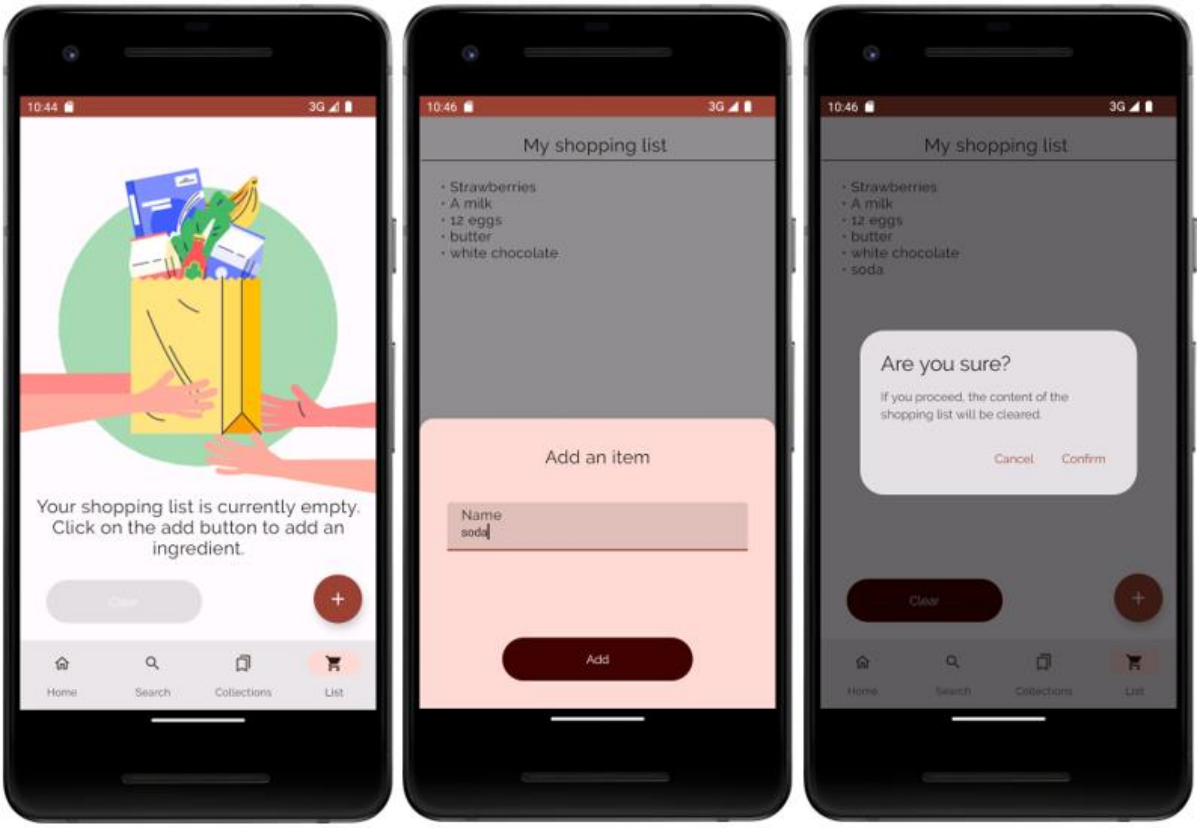



*Figure 18 - A shopping list feature in the final product app.*

### Recipe list

The recipe list can be found on the Search screen. It contains a text field where the user can enter a query based on which the list will be filtered and refreshed on the screen. By clicking on any recipe, the user can easily access its content. The user can go to the previous screen by clicking a back arrow on the top bar. The user must click the “Your recipes” button to change available recipes to custom recipes.

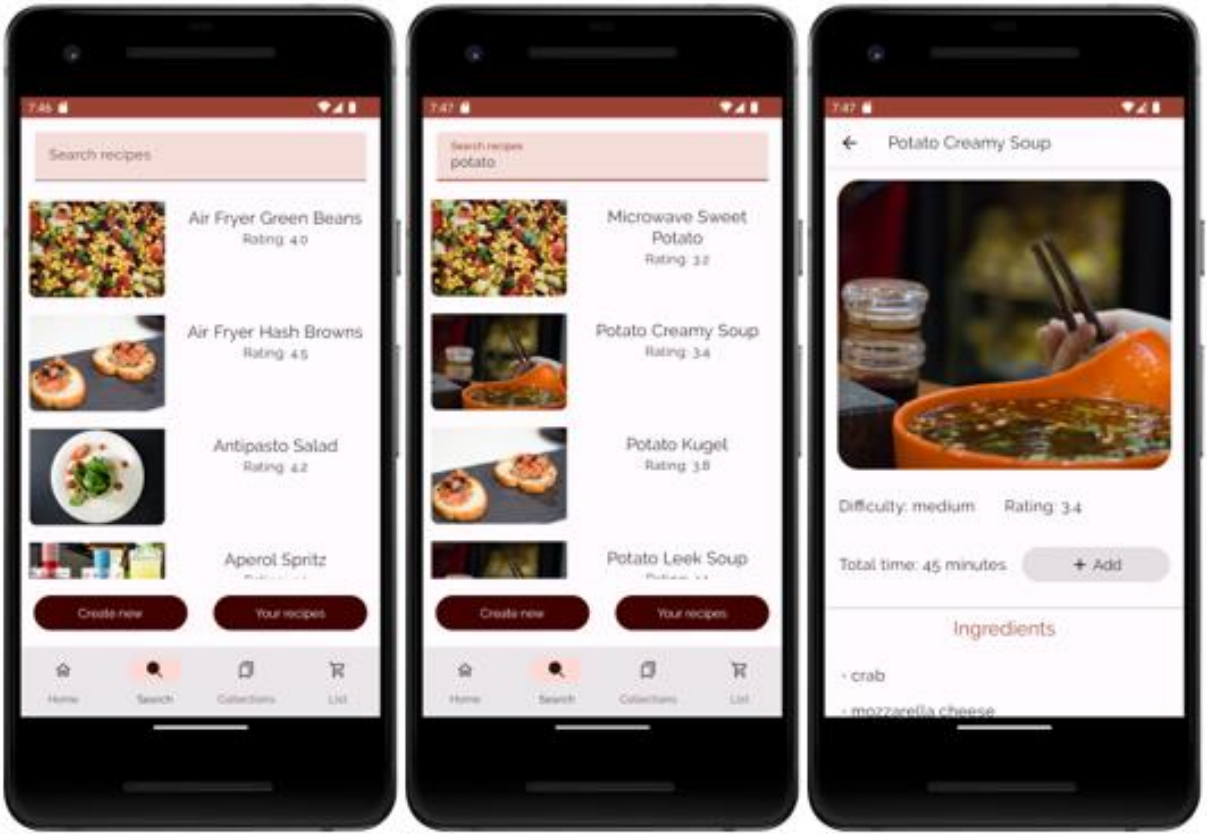



*Figure 19 - A recipe list feature in the final product app.*

### Dark Theme

The app is designed to function well in light and dark modes, ensuring users have a comfortable experience regardless of their visual preferences. In both modes, the design elements have been optimized to ensure that all features and functionalities are easily accessible and visible. The dark mode provides a modern look that makes the content stand out. (An example screen in a dark theme can be viewed in [Figure 18]).

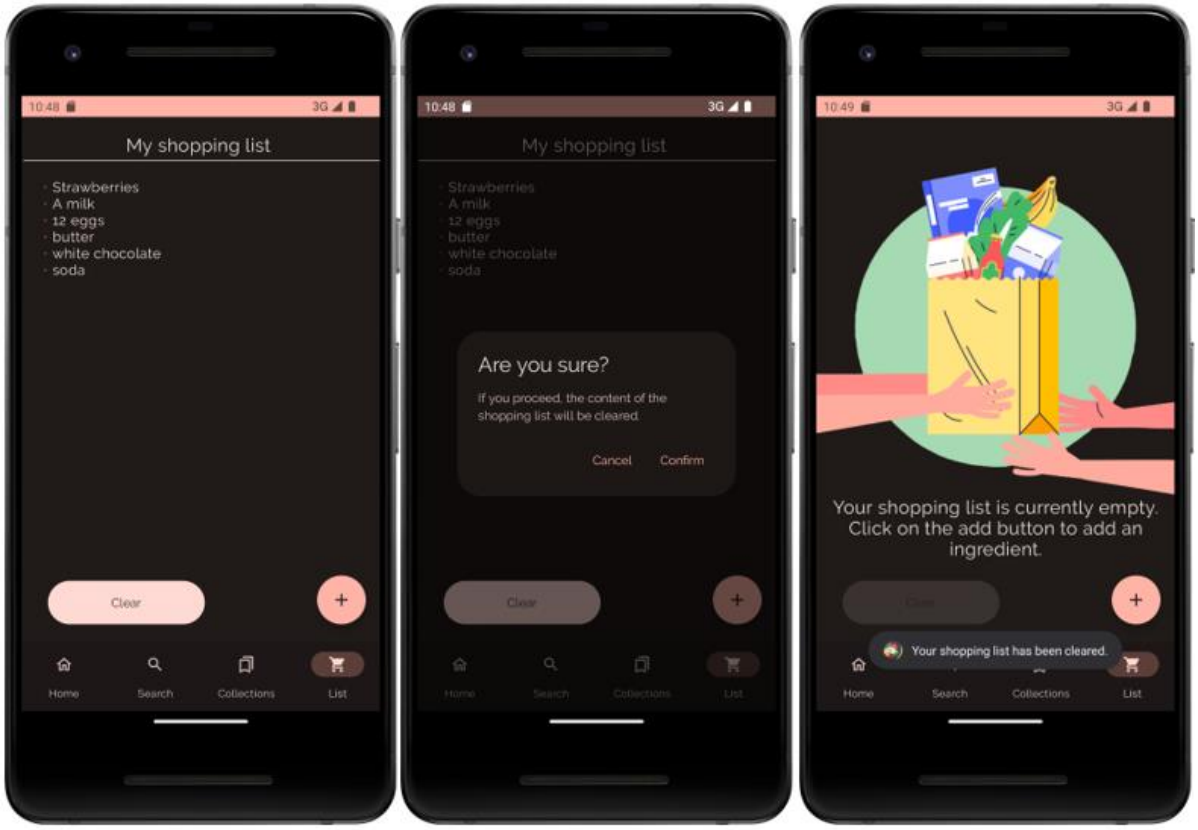



*Figure 20 - Shopping list feature shown in a dark theme in the final product app.*

### 2.6.1. Illustrations

The app's illustrations/drawings are license-free and can be used for personal or commercial projects. You can access all of them through this link http://www.freepik.com, except for the photo on the home screen depicting people eating around a table. To view this photo, click on the following link: https://www.pexels.com/photo/group-of-people-making-toast-3184183/. The photo credit goes to fauxels.

Each recipe category has its unique photo. For instance, all breakfast recipes share the same photo, but it differs from the photo used for any other recipe category. The photos used in our recipes are license-free and can be found at the following links:

*Breakfast*: https://www.freeimages.com/photo/pancakes-and-coffee-1641697
*Lunch*: https://www.freeimages.com/photo/bowl-of-spiced-potato-salad-1641827
*Vegan*: https://www.freeimages.com/photo/corn-and-black-bean-salad-1641679
*Vegetarian*: https://www.freeimages.com/photo/bruschetta-1641656
*Soup*: https://www.freeimages.com/photo/noodle-soupe-chinese-cuisine-chopstick-1634543
*Beverages*: https://www.freeimages.com/photo/party-drinks-1641647
*Dessert*: https://www.freeimages.com/photo/third-birthday-cake-1641667
*Salad*: https://www.freeimages.com/photo/plated-salad-1641688
*Dinner*: https://www.freeimages.com/photo/surf-n-turf-meal-1641663

# 3. Implementation

The Agile Scrum methodology was used to develop the project, which involved breaking down the project into smaller, manageable chunks. Initially, stories were defined to shape the sprint plan (all user stories can be viewed in Appendix A in detail). The author worked on the user stories during the sprints, and progress was reviewed at the end of the week. The implementation plan section is divided into sprints to cover each sprint's content.

## 3.1. UI prototype

At the beginning of the development process, a UI prototype template was created using Figma [27]. This prototype template played a critical role in shaping the way the implementation process was divided and in guiding the planning process. The Figma prototype served as a visual guide that helped the author understand how the app should look and function. The implementation process was made more efficient by having a prototype. It clarified what needed to be built and how it should be designed. The prototype also helped identify potential issues with the app's design and functionality early in development, saving time and resources.

## 3.2. Implementation plan

**Sprint 0:**
This sprint was not necessarily the usual sprint because the work done that week was only a set-up of the project work. A GitHub repository and a project outline document was created.

**Sprint 1:**

Worked on a UI prototype design using Figma to design the app and help shape the user stories. The prototype design was reviewed and refined to ensure it meets the user's expectations.
Based on this design, user stories were created, which helped to prioritise the work that needed to be done in the upcoming sprints.

**Sprint 2:**
All user stories (see Appendix A) were shaped and created to help to prioritise the work that needed to be done in the upcoming sprints. Sprint planning and review were held to discuss the progress with the supervisor and adjust any user stories.

**Sprint 3:**
Some research was conducted on how to implement certain features. For example, the author considered different options for storing the shopping list feature, including using a local SQL database. However, after some research, it was decided to store the shopping list in the Firebase cloud along with the rest of the data to make using the database consistently. Initial work was started to create a base for the app, including the navigation bar, creating screens, and adding Material Theme content to the app.

**Sprint 4:**
Worked on adding the database feature by connecting Firebase to the app and adding authentication (TR2). During this sprint, the author has researched how to implement the database and how to design it to follow the best practice. More on design can be seen in the Design section of this document. Necessary dependencies were added to the project, and the author made sure that the database was correctly connected to the project workspace.

**Sprint 5:**
During this sprint, the author implemented login and authentication in the app along with screens relevant to handle login. A LoginViewModel class was created to handle Firebase Authentication and Firestore operations for the login screen of the app. LoginScreen file was developed to display a form for users to enter their email and password to sign in or create a new account. At this point, the user's email and password were authenticated and stored in Firebase.

**Sprint 6:**
The focus of this sprint was Functional Requirement 1: Search through already installed recipes to find specific meals. The initial work of populating the database was done as well as a Recipe object containing eight fields in total was created. A design of a Search screen was first implemented to display the data in the app, and then the author worked on fetching the data from the Firebase database.
This function was used to fetch any data (although the body of this function was changed to match another collection in the database when needed).

```
@Composable
fun FirebaseFetcher(
    onSuccess: (QuerySnapshot) -> Unit,
    onFailure: () -> Unit
) {
    val db: FirebaseFirestore = FirebaseFirestore.getInstance()

    DisposableEffect(Unit) {  this: DisposableEffectScope
        val listenerRegistration =
            db.collection( collectionPath: "recipesready").orderBy( field: "title").addSnapshotListener { value, error ->
                if (error != null) {
                    onFailure()
                } else if (value != null && !value.isEmpty) {
                    onSuccess(value)
                }
            }

        onDispose {
            listenerRegistration.remove()
        }  ^DisposableEffect
    }
}
```

*Figure 21 - The FirebaseFetcher function used to fetch the data from the database.*

Two parameters are needed to use this function: onSuccess and onFailure. These functions will be called when data is successfully retrieved or in the event of errors. To be able to use the Firebase database, a default instance of the Firebase Firestore database needs to be created, then DisposableEffect(Unit) { ... } [31] is used to perform side effects, in this example, fetching data from an external source. The listenerRegistration value creates a listener for changes in the recipesready collection (containing all public recipes) in the database. The listener is added using the addSnapshotListener() method, which takes a callback function that is called when there are changes in the collection. The orderBy() method is used to sort the documents in the collection by their title field in ascending order. The listenerRegistration variable is assigned the listener registration object that is returned by the addSnapshotListener() method. If there is an error, the onFailure() callback is called. When there are no mistakes, and the collection has data, the onSuccess() callback gets triggered. It will provide the value parameter, a QuerySnapshot object showing the updated collection. OnDispose { listenerRegistration.remove() } is a Jetpack Compose cleanup effect which removes the listener to prevent memory leaks. [31]

**Sprint 7:**
The database was populated with new recipe data for testing. This sprint included adding a recipe display screen by fetching the content. This was carefully designed to match the UI prototype to make sure it looked appealing to the user.

**Sprint 8:**
The focus of this sprint was functional requirements 3 and 7. Adding custom recipes (FR3) and displaying the meal for the day (FR7) were developed. Adding recipes included a few screens to collect the user's entered data, which handles errors if needed.

```
// Updates the current hour, minute, and day of the week values every minute.
LaunchedEffect(Unit) {  this: CoroutineScope
    val updateJob = launch {  this: CoroutineScope
        while (true) {
            delay( timeMillis: 60_000L)
            val now = LocalTime.now()
            currentHour.value = now.hour
            currentMinute.value = now.minute
            currentDayOfWeek.value = LocalDateTime.now().dayOfWeek
        }
    }

    // If the current time is exactly midnight (00:00) a new meal is fetched from the server
    if (currentHour.value == 0 && currentMinute.value == 0) {
        mealViewModel.fetchMealOfTheDay()
    }
}
```

*Figure 22 - The logic of updating the meal for the day on every midnight.*

The coroutine LaunchedEffect is used, which creates a job that runs continuously in the background. The while (true) loop ensures that the coroutine runs indefinitely.
The coroutine inside the loop waits for 60 seconds, then updates the currentHour, currentMinute, and currentDayOfWeek values based on the current local time obtained using LocalTime.now() and LocalDateTime.now().dayOfWeek. After the loop, the function checks if the current time is exactly midnight (00:00) and if so, it calls the fetchMealOfTheDay() function on the mealViewModel object to fetch a new meal. This ensures that a new meal is fetched once per day at midnight.

**Sprint 9:**
In this sprint, the author worked on implementing category buttons (FR2) to filter the recipe database by category. This included fetching the recipe list of recipes that belong to the specified category. The list of filtered recipes was displayed by using a public RecipeList function that contained a constrained layout of fetched data. Because this function contains a lot of layout details, the author decided to make it a utility function to promote reusability and avoid duplication.

**Sprint 10:**
During this sprint, the author worked on the shopping list feature by creating a private shopping list and adding items to it (FR8) and clearing the shopping list (FR9).

**Sprint 11:**
Worked on creating a private collection of recipes, adding recipes to them (FR4), and deleting created recipes or collections (FR6). This was carefully implemented to make sure that the data was deleted not only within the app but in the database as well. This implies that when the screen is recomposed, it will indicate that the data is no longer available since it was typically fetched with each recomposition [32] to create a more real-time experience. Sprint planning and review meeting was held as usual.

**Sprint 12:**
This sprint focused on adding or improving existing documentation of the code (TR4) and testing the app manually to ensure everything was working as intended. A detailed test table was produced, and minor improvements were made based on the supervisor's feedback regarding the final product app.

**Sprint 13:**
In the final sprint, the author focused on bug fixing and code cleanup, ensuring everything was ready for submission.

## 3.3. Challenges

Several challenges were addressed during the implementation to ensure a satisfactory user experience, and one of them was performance. The user can get easily frustrated if an application takes a long time to load or responds slowly, which harms their overall experience. Coroutines [33] were used to solve the problem by handling asynchronous operations and preventing any performance issues. With the use of coroutines, the app manages resources efficiently, allowing concurrency without the overhead associated with threads [34]. Using coroutines, the app results in a more responsive user experience which makes coroutines an ideal choice where performance is an important factor.

There was another issue to worry about, such as achieving compatibility with older Android versions. The app was created using API level 33, which offers improved support for new features and functionality. However, this resulted in some features not functioning correctly on older versions of Android. For example, the advanced animation feature was not supported on older OS versions, so the author abandoned it to ensure the app ran smoothly on all devices.

During the implementation, the potential security issues that could arise were considered. The app could be vulnerable to various security threats, including data breaches, malware, and unauthorised access, and to address these concerns, security measures such as setting strict permission rules on Firebase were implemented. Login and password requirements were also added to be allowed to use the app. These actions ensure that only authorised users can access their data, reducing the risk of security breaches.

Finally, the author also considered usability issues. The plan was to ensure that users could easily navigate and find what they were looking for. Material Design 3 guidelines were strictly followed to address this, and information on where to find certain features was added on the home screen. This helped make the app more user-friendly and ensured that users could easily access the needed features.

## 3.4. Requirements covered

The app now meets all of its requirements. This includes both the core function and any extra features. Furthermore, the author has thoroughly manually tested the app. The details of the implemented features can be found in this document, in section 5.1.

## 3.5. Database data

In this mobile app, a database is used to store recipe data. To populate the database with recipe content, the author decided to use dummy data instead of real recipes. The content for ingredients and preparation steps was auto-generated using a Python script containing a list of random steps and ingredients, which updated the fields randomly. Although the recipe content might not make sense, it was decided that the real recipe content was not

necessary for the project and did not reflect any technical skill. This approach was decided to avoid copyright issues or using unauthorized data.

# 4. Testing

## 4.1. Overall Approach to Testing

In the author's approach to testing, it was decided to focus on manual testing. The author firmly believes that by doing an extensive and detailed manual test table, the software can be adequately experimented with and help to identify potential issues. Manual testing has helped to carefully examine each aspect of the app and identify the level of user experience and overall quality of the software.

To ensure that a product meets the highest standards, testing is divided into two tables, one of which is very detailed and can be viewed in "**Manual_Test_Table**" document, which is a separate document included in technical work folder for this project. Another one focuses on the functional and technical requirements of the app noted in Section 1.4.

Both test tables include references to figures, which are screenshots of the app running using an emulator in Android Studio.

## 4.2. Test Plan

The test table below provides a detailed record of the testing performed on the product. Each row corresponds to a specific functional requirement number, which can be viewed in Section 1.4 of this document. The table also includes a figure reference, which is available in Appendix C, and provides a visual representation of the feature or functionality being tested in the form of a screenshot. The test table includes a description of the test that was performed, data input as well as the expected and actual outcomes of the test. The last column indicates the test result, either pass or fail.

## 4.3. User Manual Testing

Requirements are listed in section 1.4 of this document. Figure references can be seen in Appendix C. A detailed test table can be viewed in "**Manual_Test_Table**" document, which is a separate document included in technical work folder for this project.

| FR number | Figure reference | Description | Expected Outcome | Data Input | Actual Outcome | Pass/ Fail |
|---|---|---|---|---|---|---|
| **FR1** | Figure 21 | Users should be able to search through the available recipes to find specific meals. | After entering a search query, the user should be shown a list of recipes that match the search criteria. | Entering a query "carrot" as a test. | The system successfully retrieved and displayed a list of recipes that matched the search criteria. | Pass |
| **FR2** | Figure 22 | Users should be able to filter the recipes by | After clicking on a category button, the user should be shown | Clicking "Breakfast" and "Salad" buttons. | The system successfully retrieved and displayed a list of | Pass |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | category buttons. | a list of recipes that match the selected category. | | recipes that matched the selected category. | |
| **FR3** | Figure 20 | Users should be able to create their own personal recipes that are private to their account. | After creating a new personal recipe, the recipe should be saved to the user's account and be accessible only to that user on the "Your recipes" screen. | Clicking “Create new” button and entering any relevant test data. | The system successfully created a new personal recipe and saved it to the user's account, making it accessible only to that user and viewable when clicking on the "Your recipes" screen. | Pass |
| **FR4** | Figure 23 | Users should be able to create their own private collections of recipes. | After creating a new collection, the collection should be saved to the user's account and accessible only to that user viewable on the Collections screen. | Entering a name “new collection” as a title for collection. | The system successfully created a new collection and saved it to the user's account, making it accessible only to that user viewable on the Collections screen. | Pass |
| **FR5** | Figure 24 | Users should be able to add any recipe to a selected collection. | After selecting a recipe and clicking on the add button, the recipe should be added to the selected collection. | Pressing “add” button on recipe content screen and choosing an available collection. | The system successfully added the selected recipe to the selected collection. | Pass |
| **FR6** | Figure 25 | Users should be able to delete their created recipes or collections. | After selecting a recipe or collection to delete, the recipe or collection should be removed from the user's account. | Clicking on delete icon next to “my custom recipe” recipe and “new collection” collection. | The system successfully deleted the selected recipe or collection from the user's account. | Pass |
| **FR7** | Figure 26 | The system should be able to display a meal for the day that is different from any other day. | The user should be shown a generated meal for that day that is different from any other day. | None. | The system successfully retrieved and displayed a meal for the current day that is different from any other day. | Pass |
| **FR8** | Figure 27 | Users should be able to create a private shopping list and add items to it. | After creating a new shopping list and adding items to it, the shopping list should be saved to the user's account and accessible only to that user. | Clicking FAB and entering “an item” value in the text field, pressing “Add” button. | The system successfully created a new shopping list and saved it to the user's account, making it accessible only to that user. | Pass |
| **FR9** | Figure 28 | Users should be able to clear their shopping lists. | After clicking on the clear button or clear icon next to an individual item, the | Pressing “Clear” button, and then Confirm. | The system successfully emptied the user's shopping list/ an individual | Pass |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | shopping list or individual item should be emptied. | | item in the list. | |
| **TR1** | Figure 29 | The system should allow for the creation of a new user account and store the user's information in the database. | After entering the user's information and submitting the form, the user should be saved in the database and redirected to the home screen. | Opening “Sign up” page. Entering email: “test@email.com” and password: “Testpassword123”. Clicking Create Account button to confirm. | The system successfully created a new user account and stored the user's information in the database. The user is redirected to the home screen. | Pass |
| **TR2** | Figure 30 | The system should allow users to log in to their accounts using their email and password. | After entering their email and password and clicking the "Login" button, the user should be authenticated and redirected to the home screen. | Opening “Login” page. Entering email: “test@email.com” and password: “Testpassword123”. Clicking “Login” button to confirm. | The system successfully authenticated the user and redirected them to the home screen. | Pass |
| **TR3** | Figure 31 | The system should allow users to save recipes, collections, and shopping lists and retrieve them from the database. | After adding a recipe, collection, or shopping list to their account, the user should be able to view and edit it at any time. | Logging into an existing account and checking if the previously saved test data is fetched correctly. | The system successfully saved and retrieved the recipes, collections, and shopping lists from the database and allowed the user to view and edit them at any time. | Pass |
| **TR4** | - | The code should be thoroughly documented with clear and concise comments and descriptions. | The code should be well-documented and easy to understand and modify. | None | The code is well-documented with clear and concise comments and descriptions, making it easy to understand and modify. | Pass |
| **TR5** | - | The system should have a user interface designed to be practical and intuitive, focusing on maximizing the user's experience. | The user interface should be straightforward to use and navigate. | None | The system has a practical and intuitive user interface. | Pass |

*Table 1 - Functional and Technical Requirements Test table.*

# 5. Critical Evaluation

## 5.1. Requirements and design decisions

All of the core functional requirements I noted at the start of the project I have developed as planned. There is some room for improvement in the current features and more ways to improve the app, but these could be considered if more time was allocated to the project or future development was undertaken. The following table shows the initial requirements and a corresponding column indicating whether each requirement was implemented or not.

| Requirement number | Description | Is implemented? |
|---|---|---|
| **FR1** | Search through already installed recipes to find specific meals. | Yes |
| **FR2** | Filter the recipes database by category buttons. | Yes |
| **FR3** | Create personal recipes private to the user. | Yes |
| **FR4** | Create a private collection of recipes. | Yes |
| **FR5** | Add any recipe to the selected collection. | Yes |
| **FR6** | Delete created recipes or collections. | Yes |
| **FR7** | Display the meal for the day that is different from any other day based on a local date. | Yes |
| **FR8** | Create a private shopping list and add items to it. | Yes |
| **FR9** | Clear the shopping list. | Yes |
| **TR1** | Create a user and save the user in the database. | Yes |
| **TR2** | Authenticate the user given their email and password. | Yes |
| **TR3** | Retrieve and save recipes, collections, and shopping lists in the database. | Yes |
| **TR4** | Create concise documentation of the code. | Yes |
| **TR5** | Design a practical User Interface to maximise User Experience. | Yes |

*Table 2 - Functional and Technical requirements for software.*

Having carefully studied the needs of the target audience which I had identified, I designed the product to meet those requirements, which is a notable strength. Testing these features has also proved they are working as expected, ensuring that the product will meet the needs of its users and perform the tasks initially designed.

In addition, my design decisions have resulted in a product that is both easy to use and appealing, as I focused on maximising the user's experience. The system-level and database designs are well-structured and adhere to best practices.

On the other hand, I have also noticed area for improvement in my conducted work; for instance, I mainly focused on meeting functional requirements, which may result in a less

adaptable product to changing user needs. For example, I developed a "meal for the day" feature as a recommended meal but did not offer personalisation options.

Another potential weakness is the UI design, where in some places, the aesthetics were prioritised over functionality, which could lead to issues when displayed on low-resolution mobile devices.

## 5.2. Technology and tools used

In developing the mobile app, mindful consideration was given to the choice of technology and tools to ensure that they were appropriate for the project's requirements. The decision to use Firebase Database and Firebase Authentication for the app's backend was smart because they provide a secure and reliable solution. Similarly, using Android Studio and the Kotlin language for development was wise as this combination is widely used and supported for developing Android applications.

Additionally, the use of Jetpack Compose and the integration of Material Design 3 guidelines are other strong choices for creating a dynamic UI experience.

However, I should have considered incorporating other forms of testing, such as automated testing or exploratory testing, to complement the manual testing done. Although manual testing is a valid approach, there are benefits of incorporating other forms of testing that manual tests do not necessarily have.

## 5.3. User satisfaction and needs

I strongly believe that the final product brings a lot of value to any person from the targeted audience. I have confirmed that the software satisfies many of the users' needs thanks to manual testing. Users can add personalised recipes with the app, giving them better control over their meal planning. Additionally, the incorporation of the Firebase database and authentication has enabled secure data storage and access for users, which will contribute to their satisfaction with the app. Finally, the app's interface, style and font are visually appealing, making the UI clear and clean.

However, I think there could be a greater emphasis put on personalisation options, such as the ability to get a personalised diet. There could be more work done to add notifications of already existing features, for example, a notification of a full shopping list, so the user knows they need to do the shopping.

## 5.4. Project goals and final product

The project was successful in achieving its goals, which included developing a mobile app that is easy to use, suggests meals, and allows customization of recipes. During testing, no bugs were found, which is also a positive result.

However, it believes that this results in the manual testing approach being chosen as the only testing approach. It may not have identified potential issues or bugs, and additional testing methods, such as automated testing, could be explored in future development. Despite this, I feel the final product has successfully met its intended purpose and goals.

## 5.5. Future work and improvements

The application could be further developed, and more features could be added to make it even more satisfactory and appealing. Currently, each user has private data which can not be viewed by any other, so the app could incorporate a social aspect where users can share their meal recipes with each other.

Regarding other features, a notification system could be developed to encourage the user to browse the app more often.

Additionally, the app could promote healthy habits by, for instance, recommending healthy collections.

An automated testing implementation is also worth considering ensuring that any future changes to the app will not introduce any new issues.

Regarding Scrum, one potential area for improvement is to establish a more comprehensive product backlog that includes all the necessary features and requirements, as I experienced designing the backlog not precisely enough.

## 5.6. Methodology

I applied Scrum with a Kanban board as a methodology in the development of this app, and it has proven to be practical in my project. One of the strengths of this approach is that it promotes a continuous improvement mindset. By using stories, the work was broken down into smaller chunks, and focus was put on delivering value in each sprint.

In addition to using Scrum with a Kanban board and having stories, I also conducted regular reviews and retrospectives and created a progress report to write down project progress on a weekly basis. These practices helped me keep the project on track and supported continuous progress. Writing progress reports as a review helped to keep my supervisor, Chris, informed on my project's development. To sum up, these practices helped ensure the project remained on schedule, and my adherence to Scrum rules played a significant role in the successful completion of the project.

I really enjoyed following this agile methodology because it allowed me to maintain a productive flow throughout the entire project timeline.

I noticed one potential area for improvement, which is ensuring that the stories are well-defined and provide in-depth detail to guide the development process. This would balance the work split over the weeks, as, for example, I experienced one story being at least twice as time-consuming as the others, and if the story were more detailed, it could be split even further to prevent such an issue.

## 5.7. Summary

As section 5.1-5.6 describes, the project has been successfully developed, with some areas for improvement identified. My work is very organised, primarily thanks to the methodology I chose.

I am very satisfied with my work, and I feel I have learned a lot, mainly since I have used some technology I have never used before, e.g. Firebase. I appreciate the opportunity to explore and experiment, and I am proud of my successful attempts.

If I could start again, I would consider writing down the objectives in a way more detail. I would try to gather user feedback based on my UI prototype and requirements written down at the start of the project instead of the finishing stages. This would help me understand the issue better, identify what is missing, and in general, I would see the app from others' perspectives. This might have impacted the stories, as I would focus more on the user's personalisation etc.

In general, I am proud of my work, but I recognise what could be done to make the app ideal. If it was further developed, this would be a game-changer app for some individuals. I am pleased with how I utilised the Agile Scrum methodology. My previous industrial placement experience and familiarity with Scrum allowed me to work effectively with a team. However, the downside was that I was the sole member of the team, which required me to fulfil the roles of all Scrum members, including the Scrum Master, Product Owner, and Developer, simultaneously. Working alone as a developer meant that I missed some opportunities that working in a team would offer, for instance, noticing issues during retrospectives that I could not see but someone else would.

# 6. References

[1] Summerfield, J. (2019). *Mobile Website vs. Mobile App (Application) – Which is Best for Your Organization?* [online] Hswsolutions.com. Available at: https://www.hswsolutions.com/services/mobile-web-development/mobile-website-vs-apps/. Accessed January 2023.

[2] Brenton, J., Bowen, S., June 20, S.E. and 2019 (2019). *Time to cook is a luxury many families just don't have*. [online] Today's Parent. Available at: https://www.todaysparent.com/family/family-health/time-to-cook-is-a-luxury-many-families-dont-have/. Accessed January 2023.

[3] cookidoo.co.uk. (n.d.). *Cookidoo® – the official Thermomix® recipe platform*. [online] Available at: https://cookidoo.co.uk/foundation/en-GB Accessed February 2023.

[4] www.nimblework.com. (2022). *What Is Scrum Methodology? & Scrum Project Management*. [online] Available at: https://www.nimblework.com/agile/scrum-methodology/#:~:text=Scrum%20is%20an%20agile%20development. Accessed February 2023

[5] Atlassian (2019). *Agile best practices and tutorials | Atlassian*. [online] Atlassian. Available at: https://www.atlassian.com/agile. Accessed February 2023

[6] Milkowska, W. ed., (2023). *Meal Bay Progress Report*. [online] Available at: https://drive.google.com/file/d/1oR7vu_zcuvQbU9WlVBbEZi5uw_5H8Yi9/view?usp=sharing [Accessed 23 Apr. 2023].

[7] www.nimblework.com. (2022). *What Is Extreme Programming (XP)? - Values, Principles, And Practices*. [online] Available at: https://www.nimblework.com/agile/extreme-programming-xp/#:~:text=Extreme%20programming%20is%20a%20software. Accessed April 2023

[8] en.wikiversity.org. (n.d.). *Plan-driven software development - Wikiversity*. [online] Available at: https://en.wikiversity.org/wiki/Plan-driven_software_development#:~:text=Plan-driven%20methodologies%20all%20incorporate. Accessed April 2023.

[9] Scrum.org. (2019). *The Scrum Framework Poster*. [online] Available at: https://www.scrum.org/resources/scrum-framework-poster. Accessed April 2023.

[10] BuildFire. (2019). *IOS vs Android: Which Should You Build Your Mobile App on First - BuildFire*. [online] Available at: https://buildfire.com/ios-android-which-to-develop-on-first/. Accessed January 2023.

[11] Loic, L. (2022). *6 Reasons Why You Should Pick Android Over iPhone*. [online] MUO. Available at: https://www.makeuseof.com/why-you-should-pick-android-over-iphone/. Accessed January 2023.

[12] admin (2019). *Is Android the Best Operating System?* [online] Magneto IT Solutions. Available at: https://magnetoitsolutions.com/blog/is-android-the-best-operating-system. Accessed February 2023.

[13] Material Design. (n.d.). *Material Design*. [online] Available at: https://m3.material.io. Accessed February 2023

[14] Android Studio (the official Integrated Development Environment (IDE) for Android app development) Android Studio Chipmunk 2021.2.1 Patch 2. Available at: https://developer.android.com/studio/.

[15] GitHub (2013). *Build software better, together*. [online] GitHub. Available at: https://github.com Accessed April 2023.

[16] Firebase. (2019). *Firebase Cloud Messaging | Firebase*. [online] Available at: https://firebase.google.com/docs/cloud-messaging. Accessed April 2023.

[17] app.diagrams.net. (n.d.). *Flowchart Maker & Online Diagram Software*. [online] Available at: https://app.diagrams.net.

[18] Firebase. (2019). *Firebase API Reference | Firebase*. [online] Available at: https://firebase.google.com/docs/reference.

[19] S, V. (n.d.). *MVVM In Xamarin.Forms Application For Android And UWP*. [online] www.c-sharpcorner.com. Available at: https://www.c-sharpcorner.com/article/mvvm-in-xamarin-forms-application-for-android-and-uwp/ Accessed May 2023.

[20] Android Developers. (n.d.). *Build an offline-first app*. [online] Available at: https://developer.android.com/topic/architecture/data-layer/offline-first Accessed 24 April 2023.

[21] MongoDB (2019). *NoSQL Databases Explained*. [online] MongoDB. Available at: https://www.mongodb.com/nosql-explained. Accessed April 2023.

[22] www.tutorialspoint.com. (n.d.). *Entity Framework - Lazy Loading*. [online] Available at: https://www.tutorialspoint.com/entity_framework/entity_framework_lazy_loading.htm#:~:text=Lazy%20loading%20is%20the%20process. Accessed 27 April 2023.

[23] www.conceptatech.com. (n.d.). *Best Practices: How To Design a DatabaseSoftware Engineering*. [online] Available at: https://www.conceptatech.com/blog/best-practices-how-to-design-a-database. Accessed 25 April 2023.

[24] Maynard-Reid, M. (2017). *MVVM on Android with the Architecture Components*. [online] Medium. Available at: https://margaretmz.medium.com/exploring-the-android-architecture-components-117515acfa8 Accessed April 2023.

[25] GeeksforGeeks (2018). *MVC Design Pattern - GeeksforGeeks*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/mvc-design-pattern/. Accessed January 2023.

[26] baeldung (2021). *Difference Between MVC and MVP Patterns | Baeldung*. [online] www.baeldung.com. Available at: https://www.baeldung.com/mvc-vs-mvp-pattern. Accessed February 2023.

[27] Figma (2019). *Figma: the collaborative interface design tool.* [online] Figma. Available at: https://www.figma.com. Accessed February 2023.

[28] Figma. (2023). *Material Theme Builder | Figma Community*. [online] Available at: https://www.figma.com/community/plugin/1034969338659738588/Material-Theme-Builder Accessed February 2023.

[29] Google (n.d.). *Google Fonts*. [online] Google Fonts. Available at: https://fonts.google.com/icons. Accessed February 2023.

[30] Android Developers. (n.d.). *Material Components and layouts | Compose*. [online] Available at: https://developer.android.com/jetpack/compose/layouts/material Accessed 27 April 2023

[31] Android Developers. (n.d.). *Side-effects in Compose*. [online] Available at: https://developer.android.com/jetpack/compose/side-effects. Accessed May 2023.

[32] Android Developers. (n.d.). *Thinking in Compose*. [online] Available at: https://developer.android.com/jetpack/compose/mental-model. Accessed May 2023.

[33] Android Developers. (n.d.). *Improve app performance with Kotlin coroutines*. [online] Available at: https://developer.android.com/kotlin/coroutines/coroutines-adv Accessed May 2023.

[34] Kotlin Help. (n.d.). *Concurrency and coroutines | Kotlin*. [online] Available at: https://kotlinlang.org/docs/multiplatform-mobile-concurrency-and-coroutines.html. Accessed May 2023.

# 7. Appendices

## A. User Stories

As Scrum was followed during the project's development, the work was based on the user stories created to assess the app's needs. The following user stories are related to the top-level requirements of creating a free recipe app for the development process:

1. *As a user, I want to filter recipes by title so that I can find recipes that meet my specific needs. (Priority: High)*

This user story focuses on the user's want to search for recipes by title. To help users find the recipe they need, they can filter recipes by their titles. This feature requires a search text field where a query can be entered. A search function that filters the recipe list needs to be added to find recipes. The search results should be presented in the form of a list.

2. *As a user, I want to search for recipes by category so that I can discover new and exciting dishes to cook. (Priority: High)*

This user story focuses on the user's desire to explore new recipes. Users can search for recipes based on specific categories such as diet type (e.g. vegan) or meal type (e.g. dinner). This allows them to discover new and exciting dishes they may have never considered or may not have encountered online. Implementing this feature requires accessing the database given a category the user will choose. To avoid confusion, the category was separated from the others. By doing so, users will have access to a wider range of recipes and be able to broaden their culinary skills.

3. *As a user, I want to add and save my personalised recipes to access them anytime. (Priority: High)*

This user story focuses on the user's goal to create and store personal recipes within the app. Users can easily keep track of their cooking creations by adding and saving their unique recipes. This feature allows users to organise their recipes based on their preferences, such as ingredients, preparation time, or rating, and access them whenever needed. Users have the option to delete their saved recipes at any time. They need to remove them from their

saved list to do this. The users must be able to input all necessary recipe information, including ingredients, preparation instructions, title, total preparation time, rating, difficulty level, and category. The app should allow users to search for their recipes easily, which should be stored separately from the public recipes. The main aim of this feature is to provide users with a personalised cooking experience and enhance their ability to cook and enjoy their recipes.

4. *As a user, I want to create and save my own shopping lists so that I can easily shop for ingredients needed for the recipes I want to cook. (Priority: High)*

This story is about users who want to create and save their shopping lists in the app. This feature helps the user keep track of ingredients needed for their recipes and ensure they have everything they need. The users should be able to create and delete their shopping lists. This feature aims to simplify the shopping process for the user.

5. *As a user, I want to review and add recipes to a collection so that I can easily find and remember the recipes I like or enjoy cooking. (Priority: Medium)*

This user story focuses on letting the user add recipes to a personal collection within the app. By enabling users to save recipes to a personal collection, they can easily find and remember the recipes they enjoy. This feature requires a button that allows users to save the current recipe to a personal collection and display the available collections. The app should allow the user to view these collections and their content. This feature aims to personalise the user's experience.

6. *As a user, I want to see recommended meals so that I can discover new dishes to cook and have ideas for meals when I'm not sure what to cook. (Priority: Low)*

This user story focuses on the user's wish to see recommended meals within the app. By enabling users to see recommended meals, they can discover new and exciting dishes to cook when they are not sure what they would like to cook. The implementation of this feature includes suggesting a meal for the day and displaying the recommended meals in an organised manner. This feature makes it easier for the user to discover recipes they may have missed.

## B. Detailed Test Table

The detailed manual test table was extracted and can be seen in the technical folder at this path -> wem3_TechnicalWork_2023/Manual_Test_Table.pdf.

## C. Figure References

These are the figure references showing the app running.

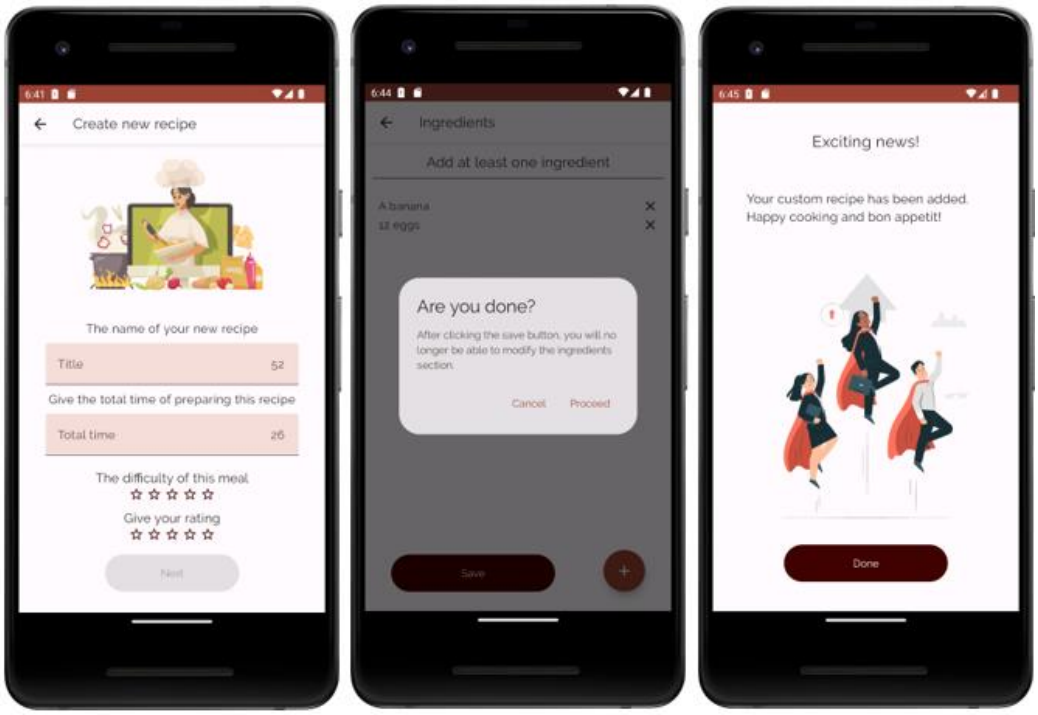



*Figure 23 - The app's implementation of Functional Requirement 3: The user can create personal recipes private to the user.*

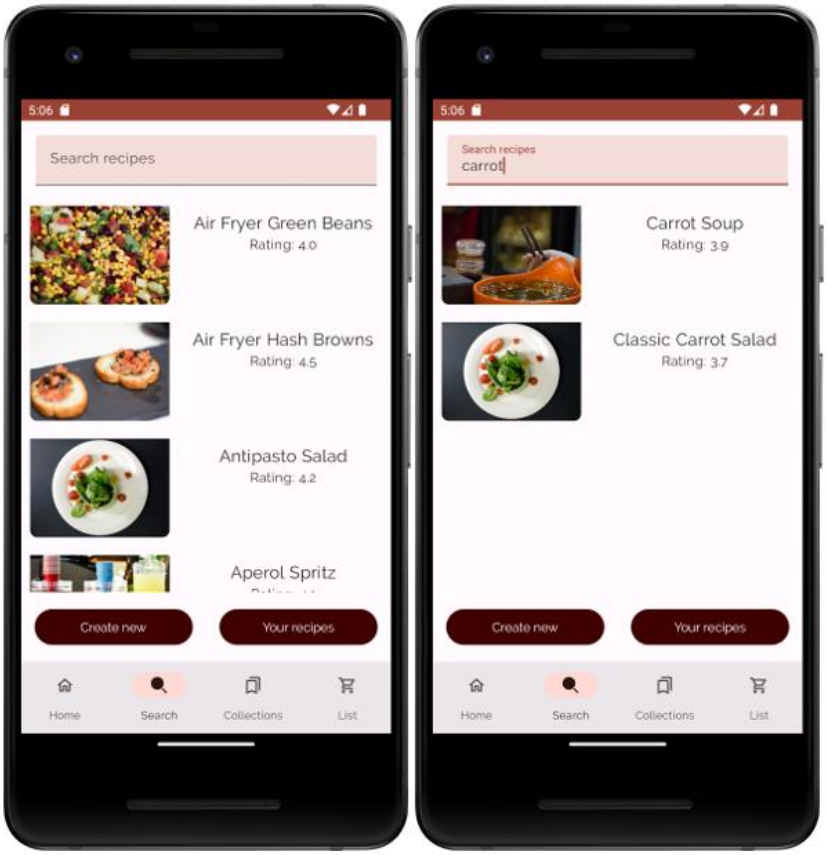



*Figure 24 - The app's implementation of FR1: Search through already installed recipes to find specific meals.*

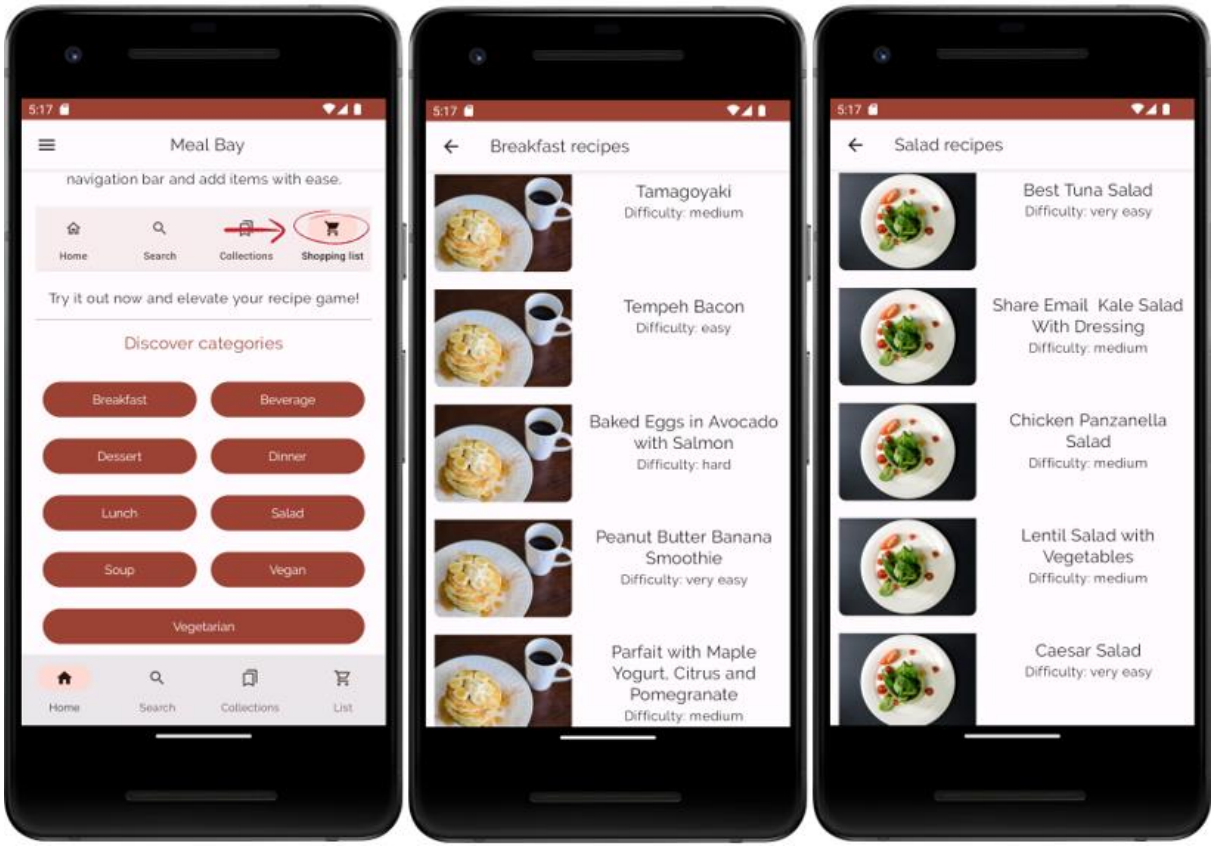



*Figure 25- The app's implementation of FR2: Filter the recipes database by category buttons.*

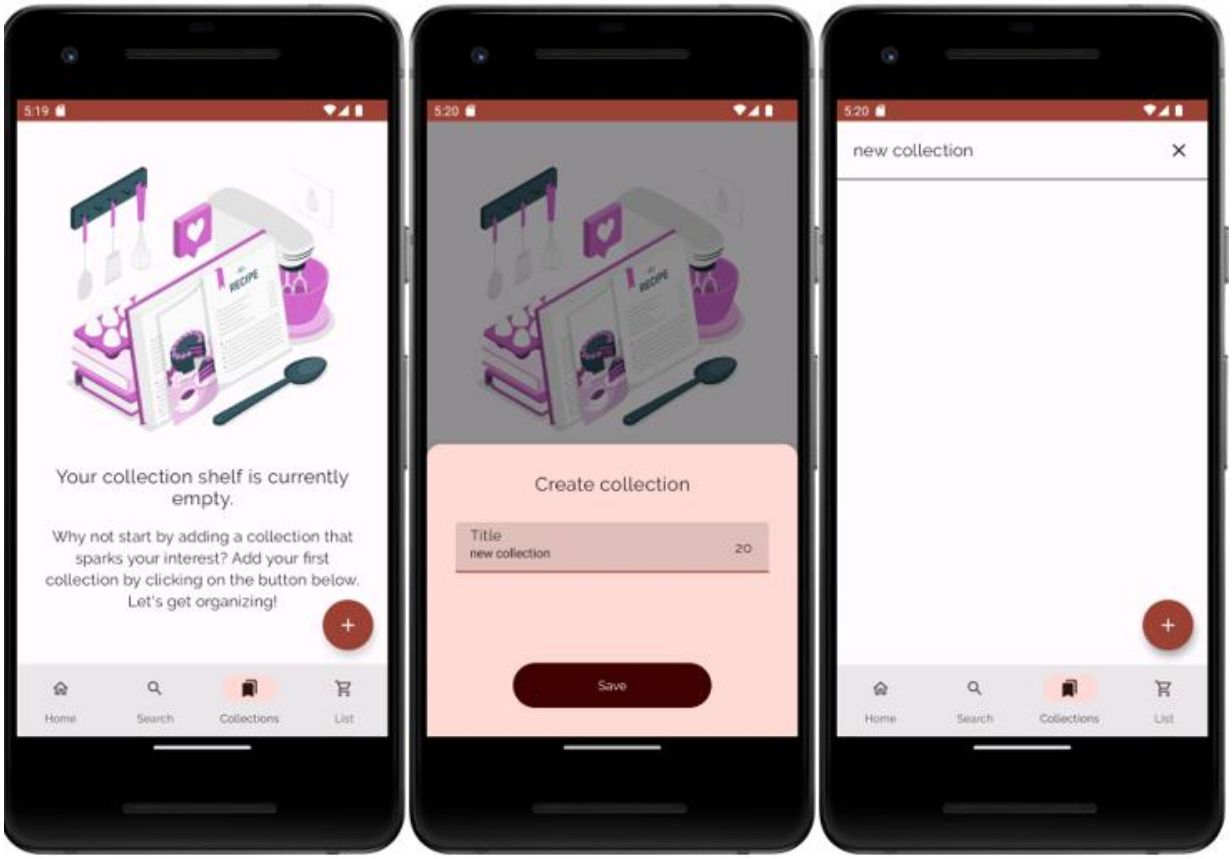



*Figure 26 - The app's implementation of FR4: Create a private collection of recipes.*

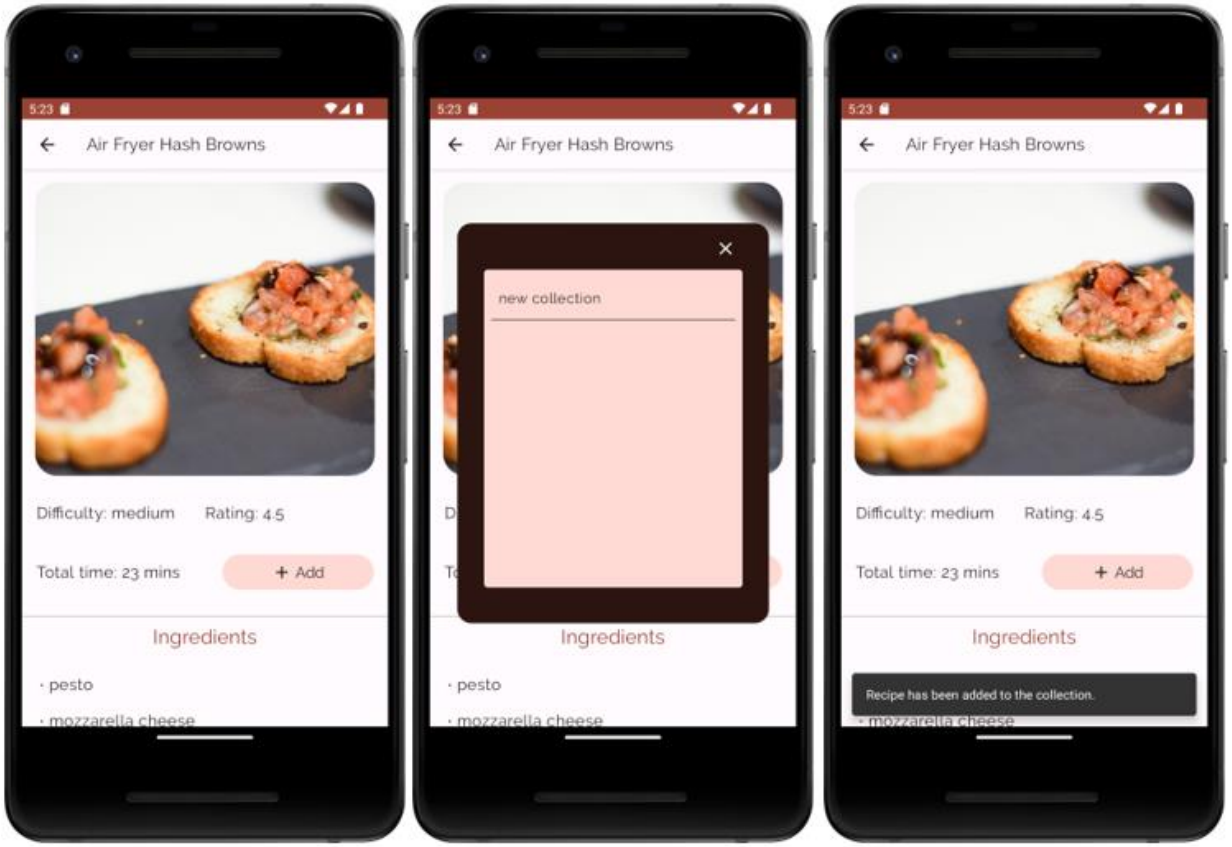



*Figure 27 - The app's implementation of FR5: Add any recipe to the selected collection.*

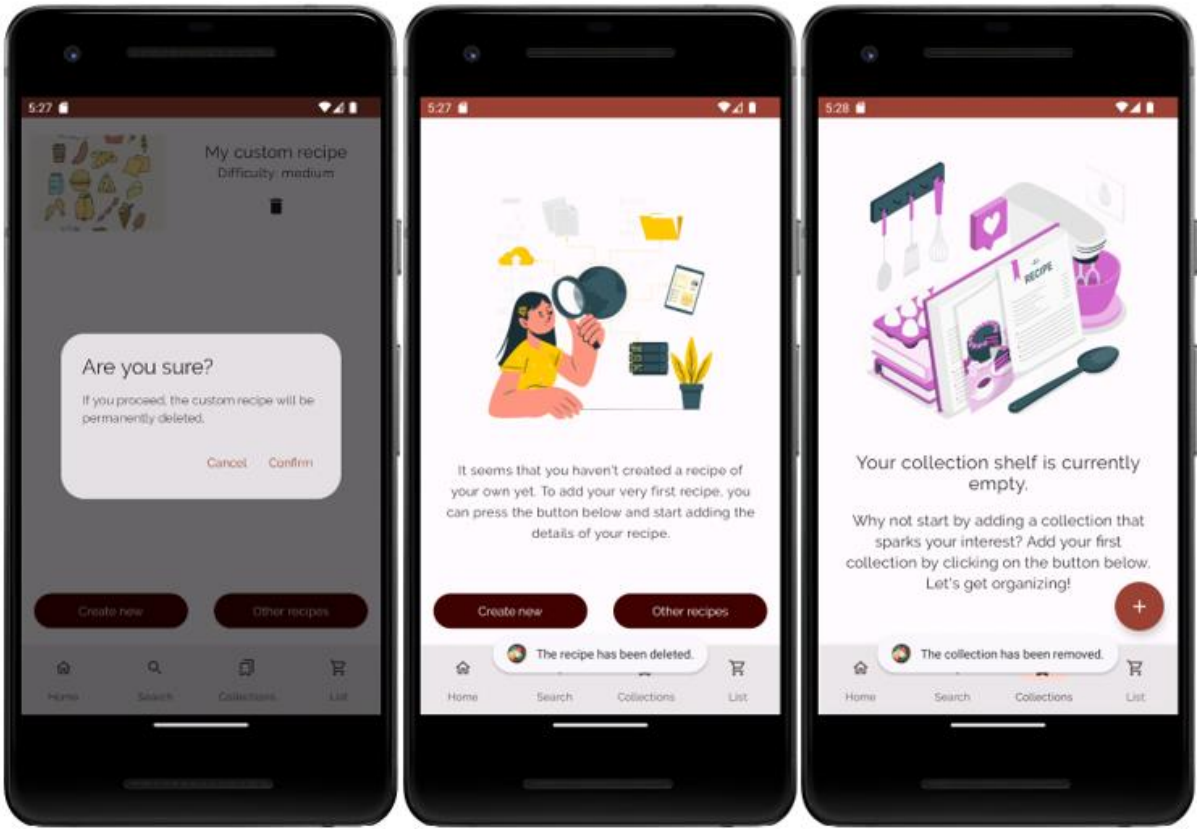



*Figure 28 - The app's implementation of FR6: Delete created recipes or collections.*

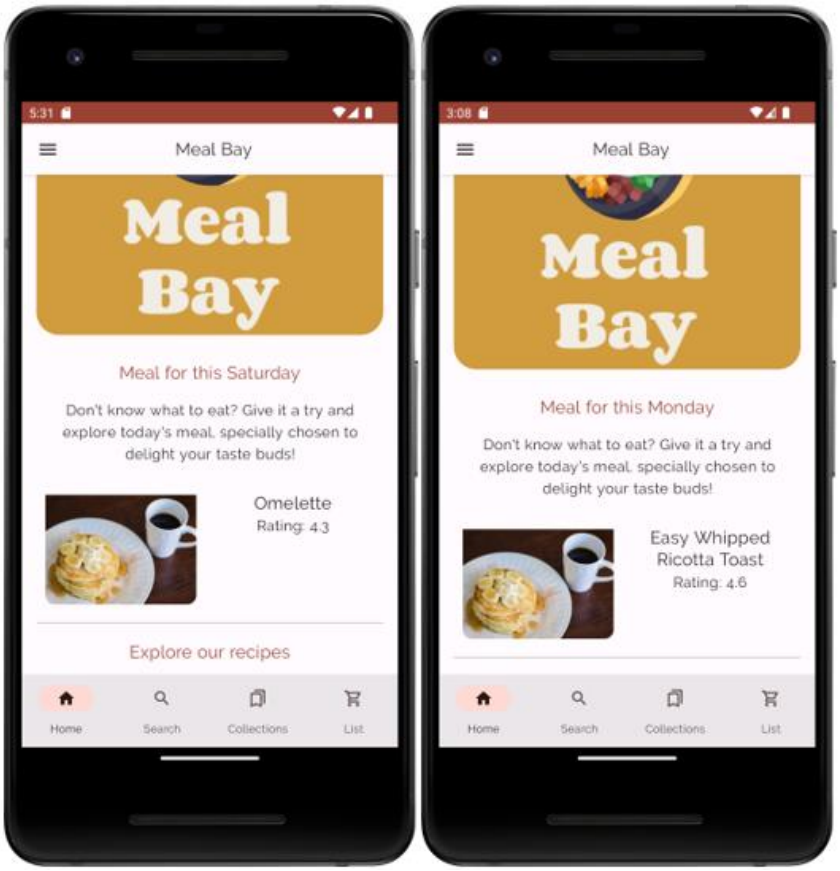



*Figure 29 - The app's implementation of FR7: Display the meal for the day that is different from any other day based on a local date.*

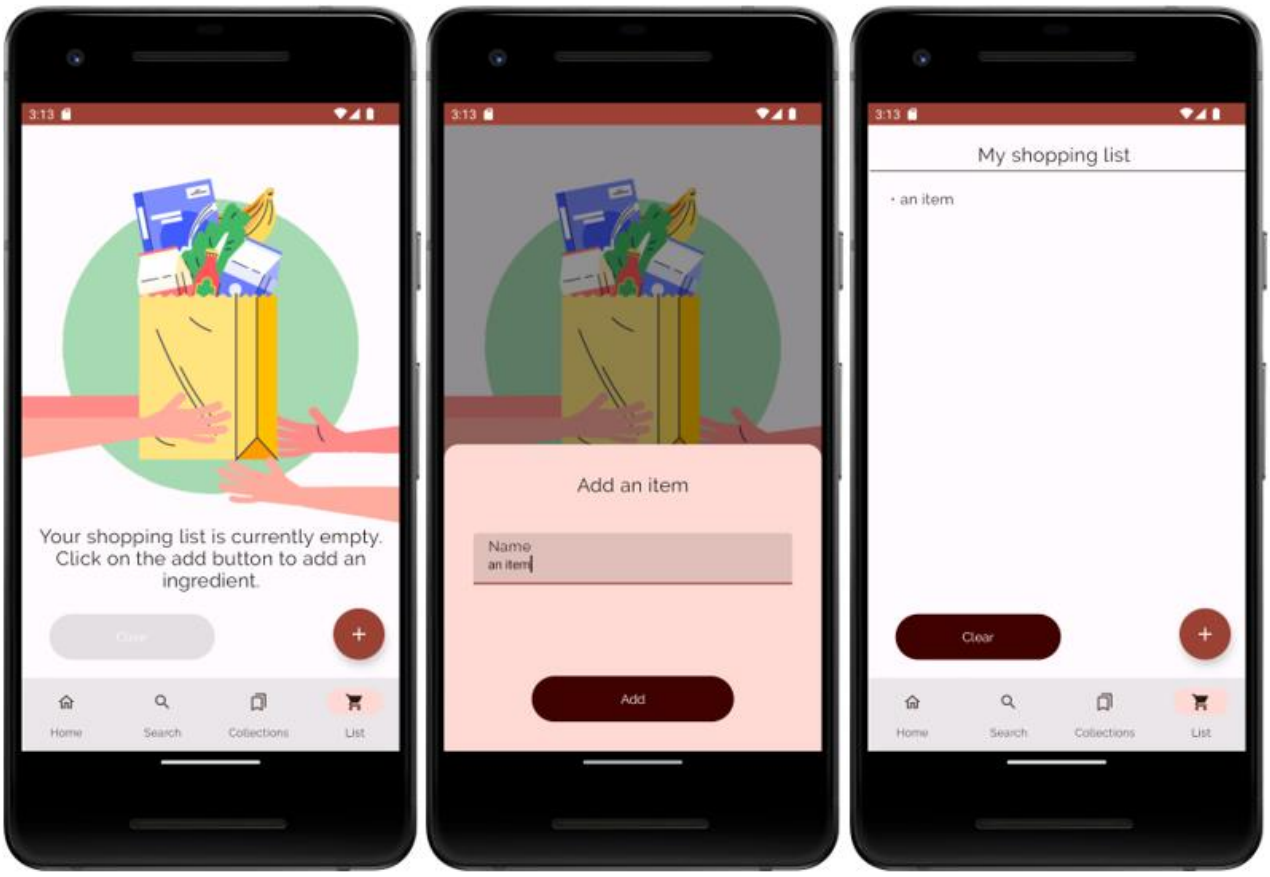



*Figure 30 - The app's implementation of FR8: Create a private shopping list and add items to it.*

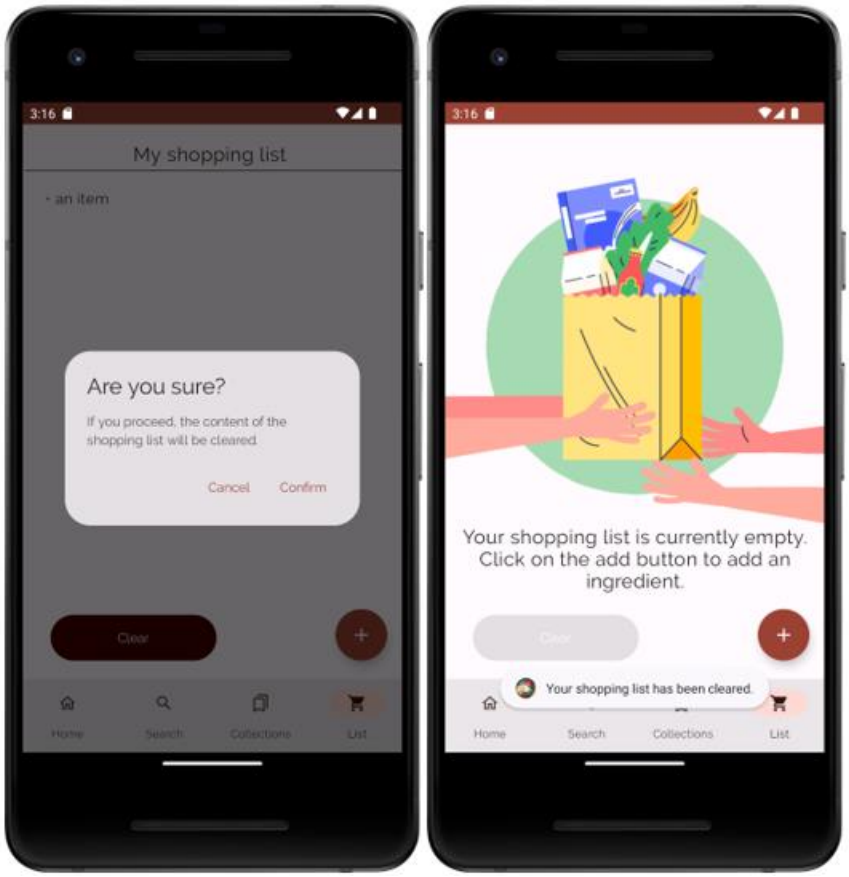



*Figure 31 - The app's implementation of FR9: Clear the shopping list.*

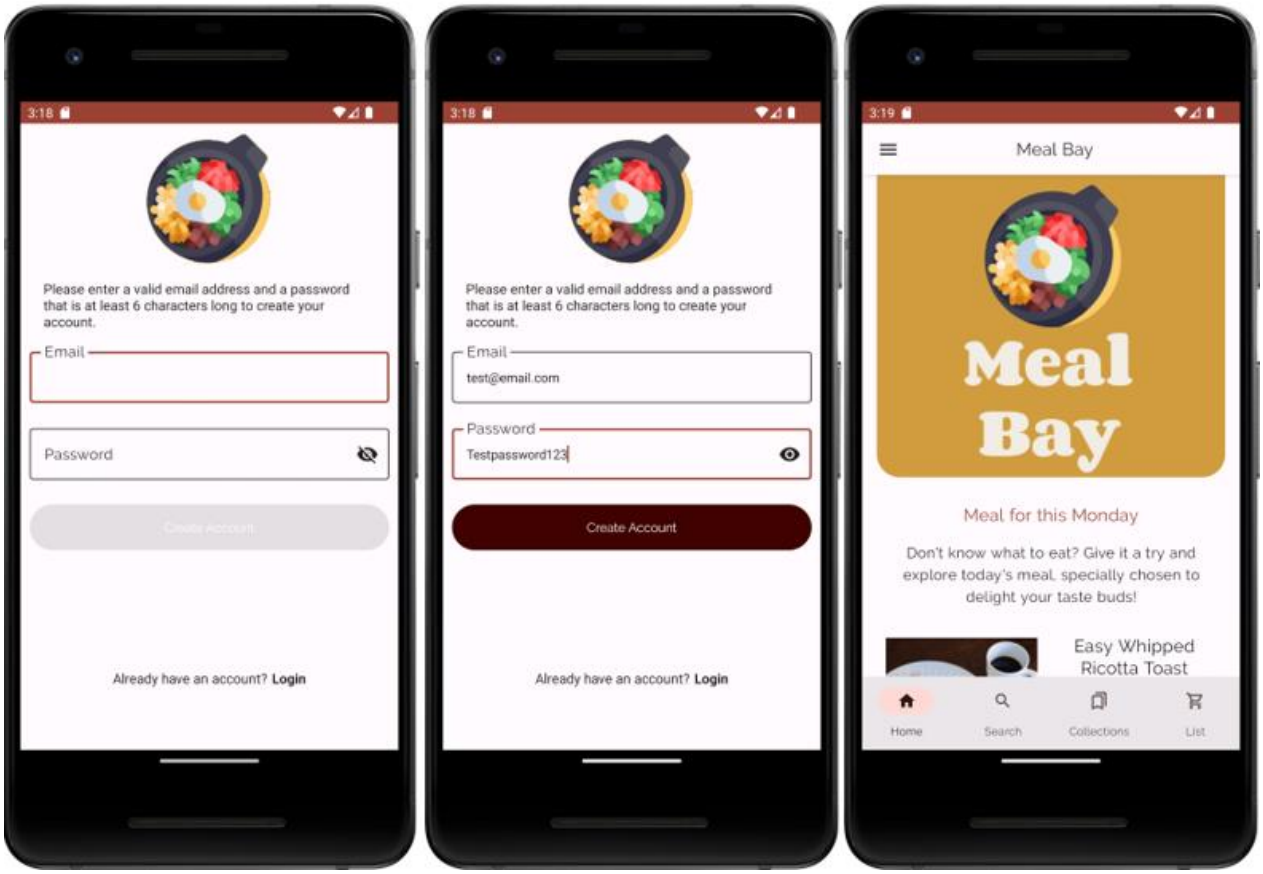



*Figure 32 - The app's implementation of TR1: Create a user and save the user in the database.*

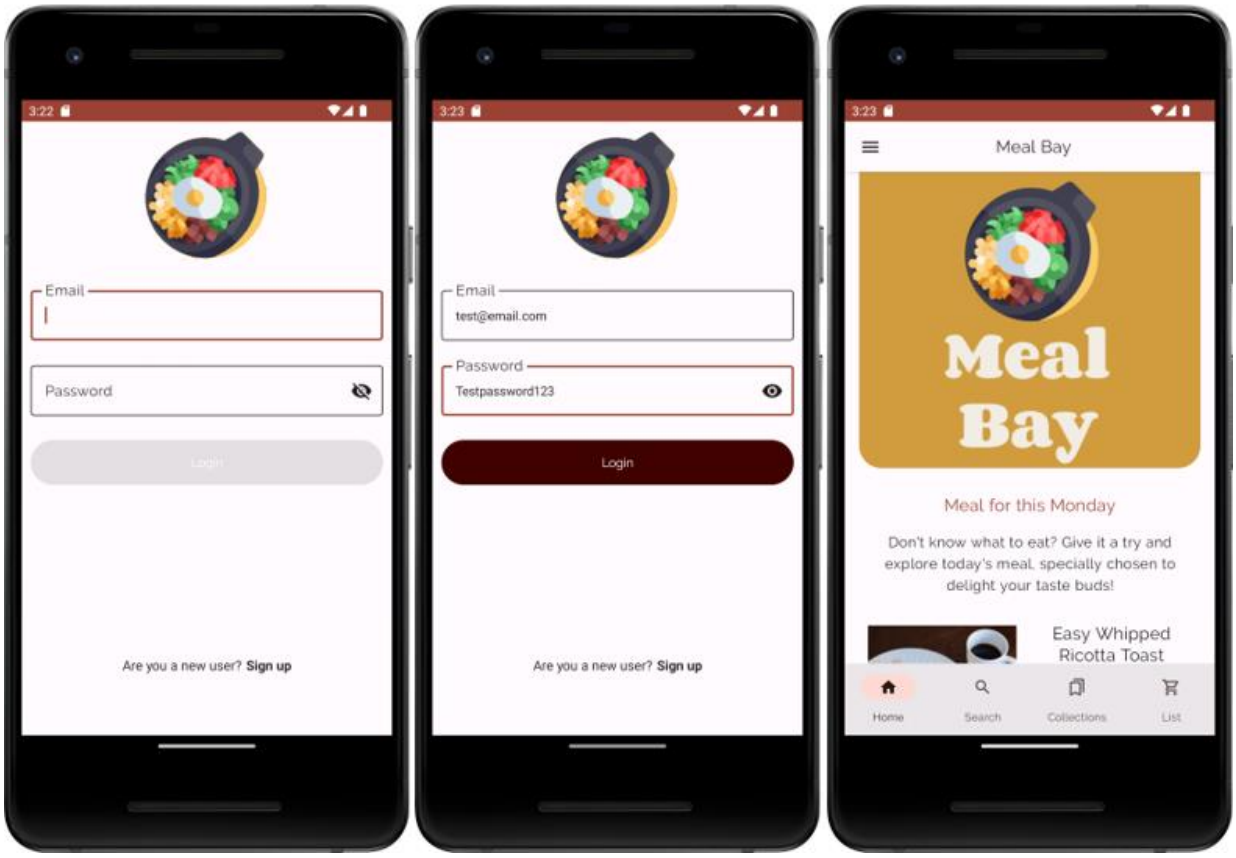



*Figure 33 - The app's implementation of TR2: Authenticate the user given their email and password.*

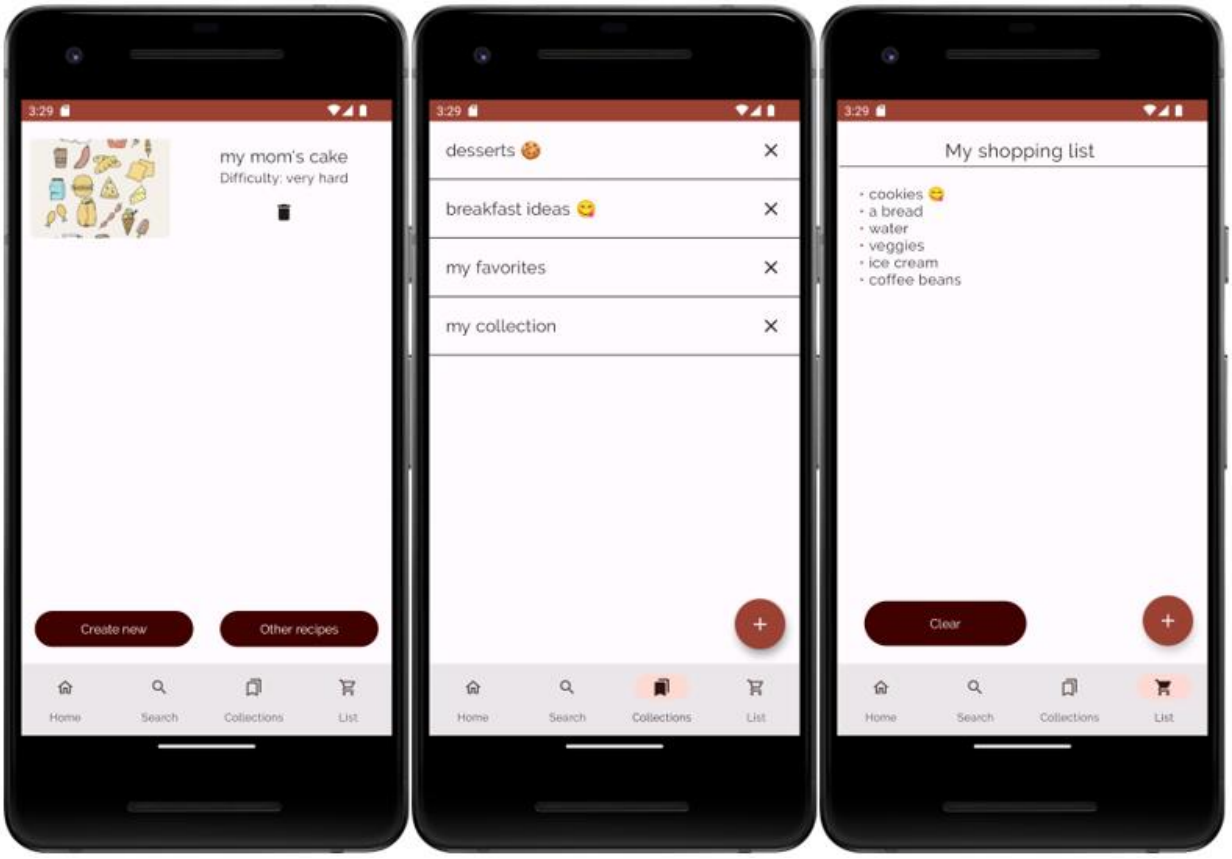



*Figure 34 - The app's implementation of TR3: Retrieve and save recipes, collections, and shopping lists in the database.*

## D. Ethics Form

**Full name**:
Weronika Milkowska

**Supervisor's name**:
Chris Loftus

**Title:**
Android App of your choice (A food book/recipe/meal ideas app)

**Using personal data:**
No

**Working with people:**
No

**Working with Animals:**
No

**Using equipment:**
No

**Project outline:**
My project idea is an application to be used on a mobile device for anyone that would be interested in managing their recipes, improving their cooking skills, and broadening their culinary knowledge by recreating meals from the application. The application will be useful for both beginners and more advanced people in cooking and aims to help them plan their diet or experiment with trying new meals. The user will be able to add their own recipes, where ingredients are added, and a preparation description and difficulty or total time are determined. The 'Meal Bay mobile application will contain pre-installed recipes to choose from. The user will be allowed to create a collection where any meal recipe can be added. An example collection created by the user could be a 'vegetarian' collection of meals that would contain no meat or a substitute. To handle both pre-installed and custom recipes, I will use a database to store and reuse such data. I am planning to make my application for personal use only, such that all the custom data the user will create is personal and private. The application will have an option of sorting meals given a criterion, for instance, 'desserts' or 'vegan' food, and these will be filtered and shown on the screen. I am planning to add a recommendation of suggested meal ideas for a day. This might be chosen randomly.

**Created at:**
23/02/2023 13:52

**Reviewed at:**
24/02/2023 08:23