

**NELSON MANDELA UNIVERSITY**

**School of Information Technology**

**Module Name (Module Code)**

**Bachelor of Information Technology (BIT)**

**Date**

***Assessment Name***

**Compiled by**

|  |  |  |  |
| --- | --- | --- | --- |
| **Lecturer** | **Student Name** | **Student Number** | **Date Submitted** |
|  |  |  |  |

Table of Contents

[Table of Figures i](#_Toc181281208)

[1. Business Case 1](#_Toc181281209)

[1.1 Introduction 1](#_Toc181281210)

[1.1.1 Meal planning and cooking for busy people 2](#_Toc181281211)

[1.2 Project Objectives 3](#_Toc181281212)

[1.3 Problem background 4](#_Toc181281213)

[1.4 Related Systems Analysis 7](#_Toc181281214)

[1.5 Project Plan 10](#_Toc181281215)

[1.6 Risk Analysis 12](#_Toc181281216)

[System Requirements 15](#_Toc181281217)

[2. Introduction 15](#_Toc181281218)

[2.1 Plan to address objectives 16](#_Toc181281219)

[2.2 Project Scope 18](#_Toc181281220)

[2.1 Information Scope 18](#_Toc181281221)

[2.2 Functional Scope 20](#_Toc181281222)

[2.3 Communication Scope 21](#_Toc181281223)

[2.3 Business Requirements 22](#_Toc181281224)

[2.4 Hardware and Software Requirements 23](#_Toc181281225)

[2.4.1 Software Requirements 23](#_Toc181281226)

[2.4.2 Hardware Requirements 23](#_Toc181281227)

[2.5 Design Constraints 24](#_Toc181281228)

[2.5.1 Security Constraints 24](#_Toc181281229)

[2.5.2 Interface Constraints 24](#_Toc181281230)

[2.5.3 Performance Constraints 24](#_Toc181281231)

[2.5.4 Data Storage Constraints 25](#_Toc181281232)

[2.6 High-level use case diagram 26](#_Toc181281233)

[2.7 UML Class Diagram 27](#_Toc181281234)

[2.8 Relational database diagram 28](#_Toc181281235)

[2.9 User interface design 29](#_Toc181281236)

[3. Introduction 42](#_Toc181281237)

[3.1 Choice of Tools 43](#_Toc181281238)

[3.2 Extracts of complex code 45](#_Toc181281239)

[3.3 Source code references 51](#_Toc181281240)

[3.4 Problems encountered 53](#_Toc181281241)

# Table of Figures

**No table of figures entries found.**

# 

# Business Case

## 1.1 Introduction

This business case document aims to give a comprehensive overview of our proposed solution to the challenges of planning meals and cooking in today’s fast-paced world. We aim to identify the various challenges that people face when preparing meals to then showcase the benefits of developing a recipe app that addresses these issues.

This document details our understanding of the problem domain and the proposed solution. This document includes 6 key sections. The problem section which provides a detailed description of the real-world problems faced by people. The project objectives section which provides the specific goals our app aims to achieve by solving the problems specified before. The problem background section which will go into the background of existing literature related to the problem that ultimately supports our proposed solution. The related system analysis section where we compare existing systems that are like our proposed system which show the advantages and unique features of our system. The project plan section where we present the project plan for our proposed system and lastly the risk analysis section detailing the potential risks associated with our project and how they can be mitigated.

### 1.1.1 Meal planning and cooking for busy people

Cooking remains a beloved activity among many, with only 16% of South Africans stating that they do not enjoy cooking (Bashir, 2024). This indicates that most people do enjoy cooking and are most likely hindered by their busy daily routines and other obstacles. Despite this love for cooking, most people find meal planning and preparation to be a chore and a very overwhelming one at that. Consequently, people then resort to fast food and takeaway meals, which in the long run may greatly affect their health negatively and lead to additional spending.

The problem is most definitely not a lack of recipes, in fact, people have access to a variety of enormous recipe databases online. Despite the vast availability of online recipes, a survey found that a whopping 53% of people still have trouble figuring out what meal to eat (Botev, 2018). It seems finding a recipe that is relevant, beginner friendly with easy-to-follow instructions is often difficult. Recipes also tend to use advanced cooking terminology, assume a certain skill level and fail to be accommodating to various dietary preferences and measurement systems. The experience on these websites/apps really discourages beginners to even try cooking and really ends up perpetuating the notion that cooking is a chore.

This problem is most prevalent in modern South African households, particularly in urban areas where convenience is often prioritized over cooking a meal at home. This accounts for 60% of South Africa’s population, of which more than half relies on fast food products (Allied Market Research, 2019).

There is a wide demographic of people that encounter these problems, these include:

* students and young adults (entering or have entered the job market) who may be inexperienced when it comes to cooking and only shop for specific ingredients
* parents and families who need to prepare meals as quickly and efficiently as possible while juggling other responsibilities
* people with dietary restrictions who really struggle to find recipes that cater to their specific needs
* people who live by themselves and would prefer not to spend so much time cooking for themselves and need quick and easy to make recipes.

There is a significant need for a solution that can assist users in meal planning and cooking that is convenient, timesaving, and personalized to accommodate various users’ busy schedules, dietary restrictions and cooking abilities.

### 1.2 Project Objectives

Our project aims to address the common challenges faced by home cooks such as accommodating dietary needs, managing time effectively, dealing with a lack of ingredients, finding motivation and improving cooking skills. The software solution we propose will achieve the following objectives:

* Offer personalized meal plans which cater to various dietary preferences
* Provide a means to filter recipes and search for recipes based on available ingredients
* Offer step-by-step instructions with integrated timers for efficient cooking
* Enable offline saving of recipes and grocery lists.
* Allow users to adjust recipes (such as changing metric systems and serving sizes)
* Make planning meals easier by incorporate a meal planner
* Grocery lists generation based on selected recipe
* Allow users to add and share their own recipes
* Support multi-platform access so the app is accessible across various devices

### 1.3 Problem background

The covid-19 pandemic brought about massive changes in eating habits, specifically there has been a notable spike in the number of people eating at home. This is largely in part due to the rising food prices post-pandemic (Innova Market Insights, 2024). Even so, many still depend on fast food as the more convenient option as it is widely perceived as the faster and easier choice. This perception has been cultivated over decades leading many to believe that cooking requires a considerable amount of time and skill, deterring a lot of people from even trying (Rodale, 2017). Recipe apps have the potential to change this misconception by offering quick and simple recipes that make cooking accessible to everyone.

Finding recipes on the internet however can sometimes feel like finding a needle in a haystack. Despite the countless options available, users still find it rather difficult and time consuming to find recipes that suit their specific needs and preferences. Once a recipe is found, users often must wade through what seems like endless text just to get to the actual cooking instructions (Fance, 2023). Moreover, having to manage and keep track of these recipes often leads to further frustration.

Since the pandemic, the search for recipes has been steadily decreasing over the years as indicated in Figure 1.1 likely reflecting the daunting nature of finding suitable recipes online.

Figure 1. 1: Google Trend Graph for "Recipes" over a 5-year period (Google Trends, n.d.)

A significant issue is food waste. Research indicates that about two-thirds of food waste in most homes is due to food not being used before it goes bad (FoodPrint, 2024). This waste in food is often the result of poor meal planning and not being able to make use of available ingredients before they spoil. Recipe apps could help households make use of these foods before they spoil by helping users find recipes based on the ingredients they already have, reducing food waste.

Another common issue is indecisiveness about meal choices. A poll, of 2000 adults found that the typical person spends about 43 minutes each week, about 37 hours every year, trying to figure out what to eat, with dinner being the most difficult decision (Lumley, 2024). Reasons for this included a lack of inspiration and a difficulty finding the right recipes. According to the study, 21% of the respondents believed the reason for their indecisiveness to be their own lack of planning and 11% attributing it to inability to find appealing recipes. By having meal planning tools and specially curated recipe suggestions integrated, recipe apps/websites can then address such challenges and make meal planning more efficient and overall, an enjoyable experience.

Despite the potential benefits, many recipe apps and websites are plagued by issues that deter users (Davis, 2024). To ensure that our website/app doesn’t follow the same issues it is essential to investigate what is making recipe apps and websites so bothersome to use.

A few of these issues are mentioned below:

• Navigation can be difficult due to a lack of clearly labelled categories

• Recipes can often be misleading with vague instructions

• Free apps and websites will be plagued with intrusive/distracting ads

• Poor and outdated UI design choices (Davis, 2024)

• lack of dietary filters for people who are vegan or follow a keto diet etc.

|  |  |
| --- | --- |
| This screenshot is from the AllRecipes website. All rights reserved. (AllRecipes, 2022) A screenshot of a menu  Description automatically generated  Figure 1. 2: Lack of Dietary Options offered by AllRecipes | Figure 1. 3: Intrusive and distracting ads on the page  This screenshot is from the Food.com website. All rights reserved. (Food, 2002) |
| In Figure 1.1, it is evident that many recipe sites generally offer limited dietary options. This presents a challenge, as today, many people adhere to a variety of dietary preferences and restrictions.  Figure 1.2 illustrates the presence of intrusive ads on recipe sites, with Food.com serving as a prime example. A large banner ad, which is even larger than the website’s own banner, distracts the user and detracts from the overall browsing experience. | |

### 1.4 Related Systems Analysis

#### 1.4.1 Recipe app name: Samsung Food

* + Platform: Android, iOS and website
  + Description of system: Personalized food and recipe app that allows users to save recipes, receive AI smart cooking skills and helps users with weekly food and meal panning. Allows users to share recipes, shopping lists and meal planner with friends and family.
  + A screenshot of a phone

    Description automatically generatedA screenshot of a phone

    Description automatically generatedScreenshots:

Figure 1. 6: Samsung Food grocery list section

Figure 1. 4: Samsung Food meal planner

Figure 1. 5: Samsung Food saving of recipes

These screenshots were taken directly from Samsung Food website. Copyright 2024 SamsungFood (Samsung Food, 2024). All rights reserved

* + List of features to adapt: user friendly user interface, meal planner, shopping list generator from recipe, ability to save online recipes in app, ability to edit recipes based on servings and metric system, sharing recipes with others and a good login process (setting user preferences)
  + List of features to avoid: Not having a save offline option for meal planner, shopping list and recipes

#### 1.4.2 Recipe app name: SuperCook

* + Platform: Android, iOS and website
  + Description of system: SuperCook is a recipe search engine that allows users to find recipes based on the ingredients they have in their pantry, manage their shopping lists and save their favourite recipes.
  + A screenshot of a phone

    Description automatically generatedA screenshot of a phone

    Description automatically generatedA screenshot of a phone

    Description automatically generatedScreenshots:

Figure 1. 7: SuperCook's ingredient filter

Figure 1. 8: SuperCook's favourites page with missing ingredient

Figure 1. 9: SuperCook's shopping list manager

These screenshots of the SuperCook (version 2.0.25) app were taken directly from the app. Copyright 2024 SuperCook (SuperCook, 2024). All rights reserved

* + List of features to adapt: Ingredient filtering and dietary filtering, having a digital pantry as well as showing what missing ingredients is in the recipe
  + List of features to avoid having long page of ingredient options, the overwhelming use of ads and recipes are very long winded

#### 1.4.3 Recipe app name: Yummly

* + Platform: Android, iOS and website
  + Description of system: Is an app/website that helps users find recipes that are right for them using artificial intelligence to meet their dietary, allergy and unique taste preferences. It allows users to plan their meals, search recipes using ingredients they have a smart thermometer feature. Yummly makes use of big data which allows them to recommend the recipes based on dietary needs, taste preference and if they are allergic to any ingredients
  + A screenshot of a recipe

    Description automatically generatedA screenshot of a menu

    Description automatically generatedScreenshots:

Figure 1. 11: Yummly's shopping list manager

Figure 1. 12: Yummly's meal planner

Figure 1. 10: Yummly's nicely laid out Ingredients

These screenshots of the Yummly app (version 8.7) were taken directly from the app. Copyright 2024 Yummly (Yummly, 2024). All rights reserved.

* + List of features to adapt: Uniform user experience, very nice design, categorizing shopping lists (Figure 1.11) based on recipe or category, time a recipe takes, layout of screens and very comprehensive recipe details page. Integration with phone calendar (Figure 1.10) and their smart thermometer which provides guidance as one cooks.
  + List of features to avoid: Not having a save offline option for recipes and subscription plans

### 1.5 Project Plan

Project Name: Thyme To Cook (Recipe App)

Client Name: Big Appetite Solutions

Version: Version 1.0

Final Delivery Date: 4 - 8 November 2024



Figure 1. 13: The Roadmap for the project (NathanTh3Gr3at, 2024)

Figure 1.13 shows the current roadmap for the project. The major deliverables are laid out and along the way the minor deliverables are added. The minor deliverables relate to the delegation of the work to the different team members.

Each major deliverable requires that the minor deliverables be completed before it can be completed.

The work is split up evenly between the group members, the work done by a member is reviewed by the other members to ensure that the information is relevant and well structured.

Communication:

Discord – main line of communication

Outlook (email) – Backup line of communication if there is an issue with Discord.

Table 1. 1:The major deliverables of the project

|  |  |  |
| --- | --- | --- |
| Deliverable | Start date | End date |
| Project Proposal Submission | 5 August 2024 | 9 August 2024 |
| Business Case Submission | 26 August 2024 | 6 September 2024 |
| System Requirements, Specifications and Technical Design Submission | 23 September 2024 | 27 September 2024 |
| Implementation Documentation Submission | 14 October 2024 | 18 October 2024 |
| System Presentation | 4 November 2024 | 8 November 2024 |
| Final Documents and System Submission | 11 November 2024 | 15 November 2024 |

Table 1.1 displays the date range for when the major deliverables need to be delivered by.

### 1.6 Risk Analysis

1.6.1 Data Loss

* Type of risk: Avoidable Risk
* How to handle: Implement regular backups and store them in multiple locations. Make use of version control systems like Git.

1.6.2 Miscommunication about requirements

* Type of risk: Minimizable Risk
* How to handle: Regular communication and discussion about what the requirements are which can be done through weekly meetings. Ensure that every member of the reviews and agrees on the requirement changes being made

1.6.3 Power Outages

* Type of risk: Minimizable Risk
* How to handle: Make use of cloud-based tools and services that that automatically save and synchronize work online. Consider investing in a backup power source (solar or uninterruptible power supply) if possible, or work in places with reliable power sources.

1.6.4 Skills gap within the team

* Type of risk: Minimizable Risk
* How to handle: Identify areas where our team may be lacking skills and find online tutorials, ai and documentation. Allocation of tasks should be based on team members’ strengths.

1.6.5 Data Security Issues

* Type of risk: Avoidable Risk
* How to handle: Implement security best practices, such as encryption of user passwords and secure authentication methods.

1.6.6 Misalignment of project scope

* Type of risk: Minimizable Risk
* How to handle: Ensure constant communication with lecturer to ensure our current project aligns with the project scope and specification. If unsure, best to ask.

1.6.7 Regulatory Compliance Issue

* Type of risk: Avoidable Risk
* How to handle: Ensure compliance with data protection laws such as POPIA in South Africa and dietary information standards.

1.6.8 Team member illness/ Unforeseen Circumstances

* Type of risk: Minimizable Risk
* How to handle: Ensure all work related to the project is well document (comments in code, descriptions for what is being done different parts of the development process) so that someone else can pick up the work if needed.

1.6.9 Limited access to software (Trial software, require subscription)

* Type of risk: Acceptable Risk
* How to handle: Work around these limitations by finding freely available alternatives or if possible, establish a budget for this software.

1.6.10 Minor Bugs or Issues in the Code

* Type of risk: Acceptable Risk
* How to handle: Prioritize fixing critical issue/bugs first and allocate time for testing and debugging. Also once solved, document these issues for future reference.

1.6.11 Inconsistent project deliverable documents

* Type of risk: Avoidable Risk
* How to handle: Ensure documentation is updated during the development process and shared on GitHub

|  |
| --- |
| Figure 1. 14: Diagram summary on Risk Analysis  In Figure 1.14 the risk analysis has been summarized into a fishbone diagram. |

# System Requirements

2. Introduction

This document aims to outline the system requirements, specifications, and technical design for the Thyme to cook recipe app. This document will provide a comprehensive overview of the proposed software solution and explain how it will be developed to address the objectives outlined in the business case, including the features, functionality, and constraints of the system.

This document contains 9 key sections. The plan to address objectives section which explains how the system will address the real-world issues identified in the business case. The project scope section which provides a high-level overview of the system’s information, functional and communication requirements. The business requirements section which lays out the expectations and needs of the specific end-users that the system will provide. The hardware and software requirements section which covers the software and hardware tools that will be used during development of the app as well the system’s expected hardware and software requirements for end users. The design constraints section where limitations related to security, interface design, data storage and system performance are discussed. The high-level use case diagram section which provides a visual representation showing how different user types will interact with the system. The UML class and relational database section will visually detail the core classes and tables that are involved in the system’s architecture ultimately outlining how data is stored and managed in the application. The user interface design section presents possible designs of various key screens accompanied by a description of its functionality.

This document serves as blueprint to help plan and consider important aspects when developing the app to ensure the final product adheres to technical and design constraints.

2.1 Plan to address objectives

Plans to address the objectives stated in the business case document are discussed below:

* Offer personalized meal plan that cater to various dietary preferences

To cater to users with varying dietary requirements (paleo, vegan, etc.), the app will implement a comprehensive user profile system. During the registration process, users will be prompted to specify their dietary preferences and any restrictions they may have. Based on these selections, the app will provide recipe recommendations using a simple filtering algorithm. Initially, the recommendations will focus on matching users with recipes that fit their chosen preferences, without complex dynamic updates.

* Provide a means to filter and search for recipes based on available ingredients

Users often find it difficult to match their available ingredients with suitable recipes. The app will allow users to input their available ingredients via a simple input interface. The backend will utilize a search algorithm that compares the provided ingredients against the database of recipes, filtering results accordingly. This feature will ensure users can easily make use of what they already have, minimizing waste.

* Offer step-by-step instructions with integrated timers for efficient cooking

To help users better manage their time in the kitchen, each recipe will be broken down into clearly defined steps. The app will offer integrated timers for any step that requires precise timing, such as baking or simmering. The Flutter framework will be used to design this real-time feature, allowing the app to handle multiple timers concurrently, enhancing the user experience.

* Enable offline saving of recipes and grocery list

Many users may find themselves without internet access whether that’s while grocery shopping or cooking, the app will be designed to work offline. Recipes and grocery lists will be saved locally on the user’s device and will sync with the cloud whenever internet connectivity is restored. This ensures users can access important data in any situation while also keeping all devices up to date once they reconnect.

* Allow users to adjust recipes (such as changing metric systems and serving sizes)

To address the diverse cooking habits of users from different regions, the app will allow for real-time recipe adjustments. Users will be able to change serving sizes, and the ingredient quantities will update automatically to suit these changes. The app will also support conversions between metric and imperial units, giving users flexibility based on their preference or geographic location.

* Make planning meals easier by incorporating a meal planner

Meal planning can be a daunting and time-consuming task. To simplify this process, the app will include a meal planner that allows users to schedule their meals for the week. Users will then be able to select recipes for specific days of the week. This feature will also integrate nicely with the grocery list generation tool to ensure users have everything they need for the week’s meals.

* Generate grocery lists based on selected recipes

To eliminate the hassle of manually creating grocery lists, the app will automatically generate a list of ingredients based on the user's selected recipe. This list will be customizable, allowing users to remove ingredients they already have and add any additional items they may need. The app will also enable users to sync their grocery lists across devices for seamless access while shopping.

* Allow users to add and share their own recipes

Many home cooks have personal recipes that they may want to share or store digitally. The app will include a feature that allows users to share their own recipes with others. Additionally, users will be able to categorize their recipes with custom tags, making it easier for others to discover them through search filters.

* Support multi-platform access so the app is accessible across various devices

To provide a seamless experience across different devices, the app will be developed using Flutter, a cross-platform development tool. This ensures that the app will be accessible on Android, iOS, and the web, allowing users to switch between devices without losing progress or saved data. Cloud synchronization will ensure that user data (grocery lists, meal plans, etc.) is consistently updated across all platforms.

2.2 Project Scope

2.1 Information Scope

* User information

Basic profile information such as usernames, email addresses, users’ dietary preferences, any past meal plan data. Also, user-generated data such as recipes saved, grocery lists generated/made and personalized meal plans will also be stored. Grocery lists and meal plans will be stored locally on the user’s device as well as on the cloud so that data can be synced across all their devices. User login credentials will be encrypted both in transit (HTTPS) and at rest since they will be stored on the Firebase database. Personal information such as the user’s dietary preferences (allergies, diet) and email will also be encrypted to ensure data privacy and protection from threats.

* Recipe information

Recipe details such as the ingredients, step by step instructions, name of the recipe and the associated dietary tags will be stored. Since users will also be able to upload their own recipes. The names, photos and related information will also be stored for each recipe. This information will be stored in a cloud-based database Firestore, recipes that are accessed regularly will be cached locally on the user’s device so they can access them offline. Recipes that are added by the user will first go through moderation by moderators before they are uploaded to the cloud globally to all users. Recipes that are public to all users and do not contain sensitive information will not be encrypted.

* Grocery List and Meal Planner information

Grocery lists and meal plans created will be stored so users can reference them when needed. Information such as ingredients, quantities and recipes linked to the grocery list and plans will all be saved. This data will be stored locally on the users’ device as well as on the cloud. Allowing users to then access their grocery lists and meal plans from any device and view them offline. Basic encryption will be applied to the data syncs with the cloud since it is not highly sensitive, to ensure the integrity of the data.

* Images

Images linked to recipes; this includes user-uploaded photos will be stored. These images will mainly be stored on the cloud (Firestore). Recently viewed images will be cached locally so user experience is improved a bit.

* Offline Storage and Syncing

Saved recipes, grocery lists, and meal plans selected by the user will be stored locally. This information will be stored on the user’s device in a secure format. When an internet connection is established, the local data will be synced with the cloud data.

In summary the types of information that will be stored in the app will be user-data, recipe information, grocery list and meal planning information. These will be stored securely in the cloud with the use of Firestore. Recipes can be saved locally on the user’s device to allow accessibility offline as well. User credentials will be encrypted and stored securely on the Firebase database. Local data that has been saved offline will be synced with the cloud once the user has gone online. This allows users to edit their meal plans or grocery lists while offline, which can then be synced across different devices. This includes their selected preferences as well as account information to check if they have been registered.

User data to be stored:

* Usernames
* Email address
* Dietary preferences
* Favourites and saved recipes
* User generated content
* Meal plans
* Grocery list items

Recipe data to be stored:

* Titles
* Images
* Ingredients
* Instructions
* Nutrition information
* Tag data

2.2 Functional Scope

* Searching and filtering recipes

This feature will allow users to input their ingredients, diet, recipe name/or part of it and the system will return a list of recipes that match those parameters.

* Planning meals

This feature will allow users to create customized meal plans tailored to their dietary restrictions and preferences. The app will provide an interface where users can select recipes and assign them to specific days of the week.

* Customizing recipes to suit preferences

This feature will allow users to modify serving sizes, the app will automatically update the recipe by adjusting the ingredient quantities. The app will also have the option switch between metric and imperial measurement systems.

* Sharing of recipes

This feature will allow users to add and share their own recipes, meal plans and grocery lists.

* Generating grocery lists

This feature will allow users to automatically generate a grocery list containing all necessary ingredients based on a selected recipe. Users will also be able to customize this list by adding or removing items as they choose as well as adjusting the quantity of each ingredient in the list.

* Saving of recipes, grocery lists, and meal plans offline

This feature allows for recipes, grocery lists, and meal plans to be saved offline. All data will automatically sync with the cloud once a connection is established.

* View recipes with step-by-step instructions and integrated timers

Most recipes will include step by step instructions to guide users through the cooking process. This feature will offer timers for specific cooking stages, ensuring users can manage their cooking times effectively.

* Generating reports and recipes to moderate for administrators

This feature will generate a report regarding which recipe was most searched, tried, liked, etc. This feature will send admins recipes to approve.

2.3 Communication Scope

Communication within the app will occur through these channels:

The app will send push notifications to alert users to new recipes or reminders for their meal plans. The app will notify users when their data has been successfully synced after they are back online. The app will function over both mobile data and Wi-fi, allowing users to access features regardless of their internet connection type. The app will also notify admins of any recipes that need to be moderated.

2.3 Business Requirements

For the project we aim to build a recipe app which home-cooks or any beginner will be able to follow. The app will be cross-platform which allows users to share and sync their recipes and preferences whether they make use of a mobile device or access the website from a desktop. The three key user types for this system are registered users, system administrators and unregistered user/guests, each with their own specific needs and requirements.

**Registered users**

These are the primary users of the app; they will require a user-friendly and dependable interface for meal planning and cooking. Their main requirements are:

* Offline accessibility: Users will be able to access their saved recipes, meal plans and grocery lists offline.
* Personalized recipe suggestions: New recipes will be available to users based on their preferences or diet. These personalized suggestions will be synced across multiple devices for the user.
* Meal plans and grocery lists: Users can plan meals and generate a corresponding grocery list with the necessary ingredients so that they can prepare for future recipes that they may wish to cook.

**System administrators**

System admins are responsible for moderating and maintaining the app to ensure that it runs smoothly, and everyone adheres to community standards. Their requirements are:

* Monitoring and moderation: Admins will be able to moderate user submitted recipes, ensuring only recipes that meet the standards are posted. The most liked and disliked recipes will be logged for the system admin to view and post.
* Manage user account: Admins will be able to manage user accounts by adjusting access levels or deleting accounts.
* Monitoring performance: Admins will have access to tools that will allow them to monitor the analytics of the database.

**Unregistered users/guests**

Unregistered users or guests will have minimal amount of access to the app’s features. Their requirements are:

* Basic recipe access: They will still be able to view and browse through some recipes.
* No access to advanced features: They will not have access to features such as adding to a grocery list, meal planning or the user preference features.

2.4 Hardware and Software Requirements

This section covers the software and hardware tools that will be used during development of the app and the system’s expected hardware and software requirements for end users.

2.4.1 Software Requirements

* The application will be developed using Visual Studio Code as the IDE
* Flutter will be used for the development of the app within VS Code.
* Firebase will be used to handle the login and registration as well as the security covering them.
* Firebase will manage both online storage and offline caching.
* The mobile application requires Android 5 (API level 21) or higher.
* The web app will run on any browser but will be optimized for Microsoft Edge.
* The database will be hosted on the cloud using Firebase Cloud Firestore.

2.4.2 Hardware Requirements

* The app will require a phone or tablet with at least 2GB of RAM and running android 5.0 or higher. Adequate storage is also needed to handle downloaded recipes and cached data.
* The web app can be accessed from any device with a web browser; however, users will need a stable internet connection.
* The database will be hosted on Firebase’s cloud infrastructure.

2.5 Design Constraints

This section outlines the key constraints that may impact the app/website development and performance of the system, including security, interface, performance and data storage constraints.

2.5.1 Security Constraints

Firebase will manage user authentication which ensures all login and registration data are securely processed. It automatically handles password hashing and salting using bcrypt behind the scenes. Additional security measures, such as multi-factor authentication will also be implemented to enhance security.

When a user adds a recipe to the app or website all text must be verified as text and not code as this could create security vulnerabilities.

All sensitive data such as user credentials, preferences and recipe details must be encrypted both in transit and at rest. Firebase encrypts all data stored in Firestore at rest with AES-256 encryption and has built in security features during transmission between the client and server.

Having role-based access control will ensure different roles whether it be admin, registered users or guest users have varying levels of access to app features. This restricts users from having access to features they prohibited to access (based on their permissions) which safeguards both the system and user privacy.

2.5.2 Interface Constraints

The interface of the app needs to be easy to use and not feature many complex navigation options that cause confusion. The app must not be overly simplified to the point where features are sacrificed.

Users should be given the opportunity to be able select a diet preference and have a section that recommends recipes that conform to the preferences. Additionally, an area that allows users to explore new and trending recipes will enhance the user experience.

The interface should be designed to be responsive and adapt smoothly to various screen sizes such as those of tablets and smartphones. Ensuring consistency across devices is very important so users have a uniform experience and design even when switching platforms.

2.5.3 Performance Constraints

The performance of the app may differ across platforms. Since the app is cross-platform, it will be harder to maintain consistent performance across multiple devices. The app should be optimized to minimize CPU and memory usage so that it can perform smoothly even on low-end devices.

The responsiveness of the interface and UX design will need to be consistent so that the user’s do not have different experiences in terms of the performance of the system. Regardless of whether the app is accessed on an android or web browser should be irrelevant and the system should provide consistent load times, navigation speeds and responsiveness.

Considering mobile users will be able to access offline features such as saved recipes or meal plans, the app should be optimized to use minimal resources such as battery power and network data.

2.5.4 Data Storage Constraints

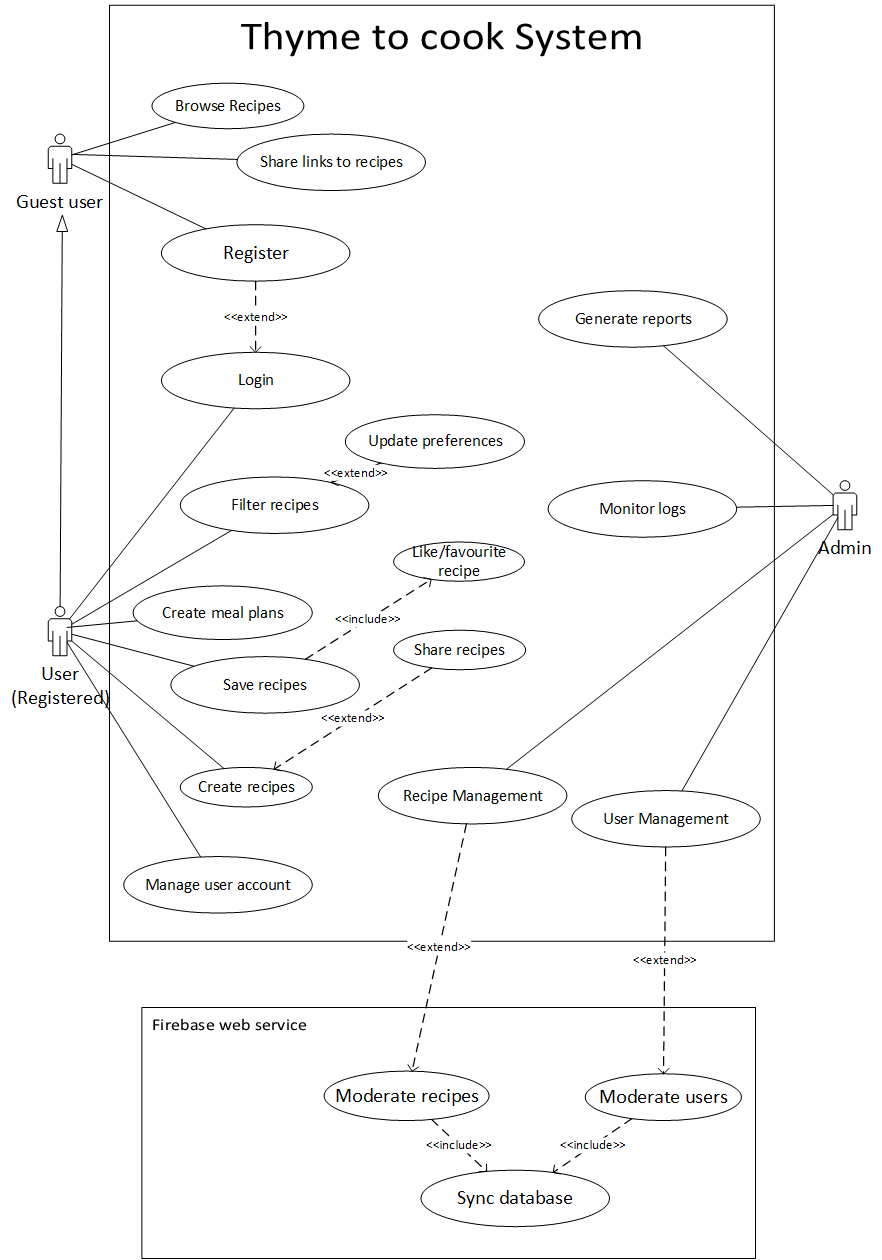
Firestore which will be used to store the recipe images on the cloud has a free-tier model which is limited and will be used for this app. The free tier has a limit of 1GB storage, as well as 1GB of outbound data per month. The database will continue to function normally, but additionally charges will be placed when exceeding this limit which could impact the system’s scalability and in the long term its sustainability.

Users that want to upload their own recipes with images will need to be constrained so that they do not take up the amount of cloud storage space. The maximum file sizes for the image that a user wants to upload should also be limited to ensure the system still operates efficiently within its data limits.

As the user saves the recipes too it should be considered if they have enough local storage space to save the recipe offline. The app will need to monitor available storage on the device so that when space limits are reached, the user is prevented from downloading the recipes. Users should also be informed of potential sync delays.

2.6 High-level use case diagram

Figure 2.1 represents the high-level use case diagram of the Thyme To Cook system.



*Figure 2. 1: High-Level Use Case Diagram*

2.7 UML Class Diagram

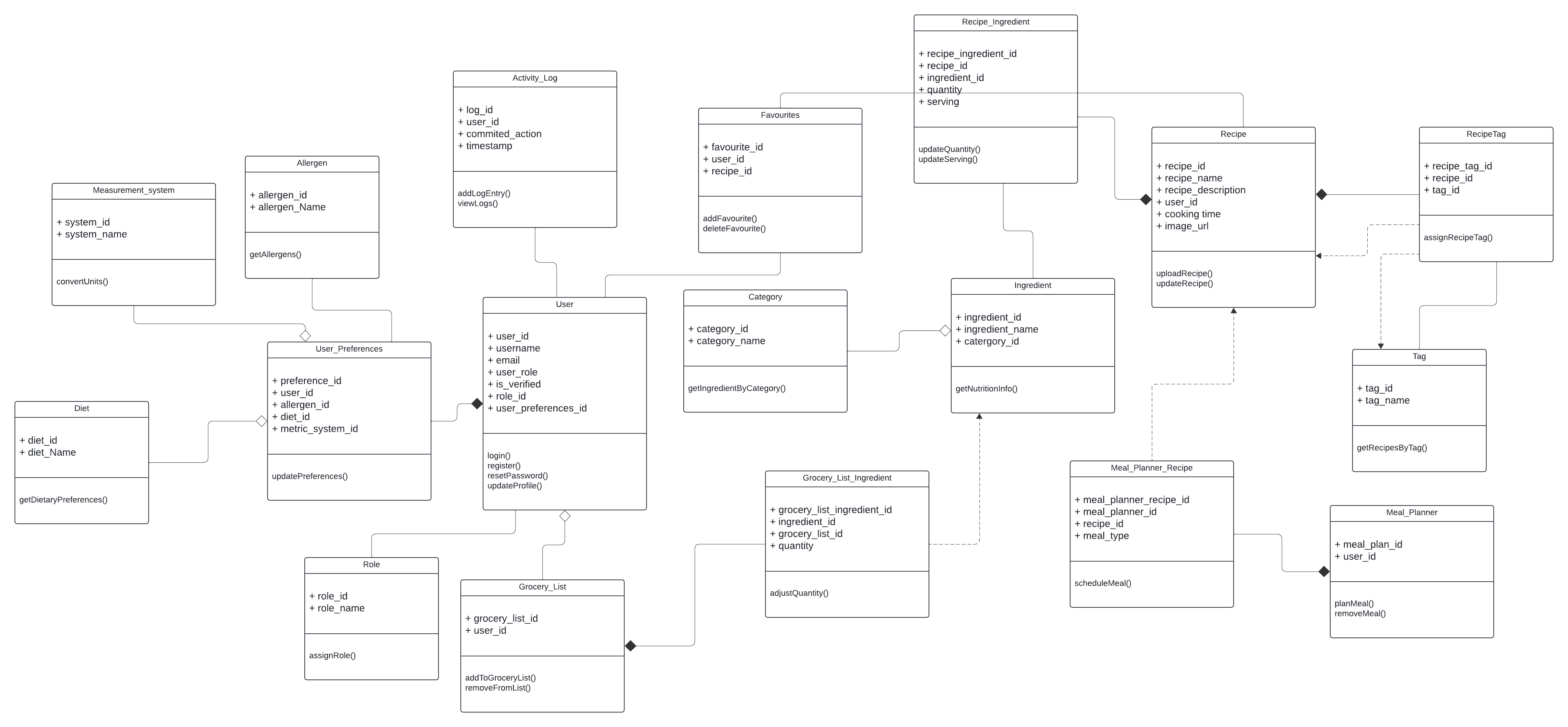
Figure 2.2 represents the UML class diagram of the Thyme To Cook system

Figure 2. 2: UML Class Diagram

2.8 Relational database diagram

Figure 2.3 represents the relational database diagrams of the Thyme To Cook system

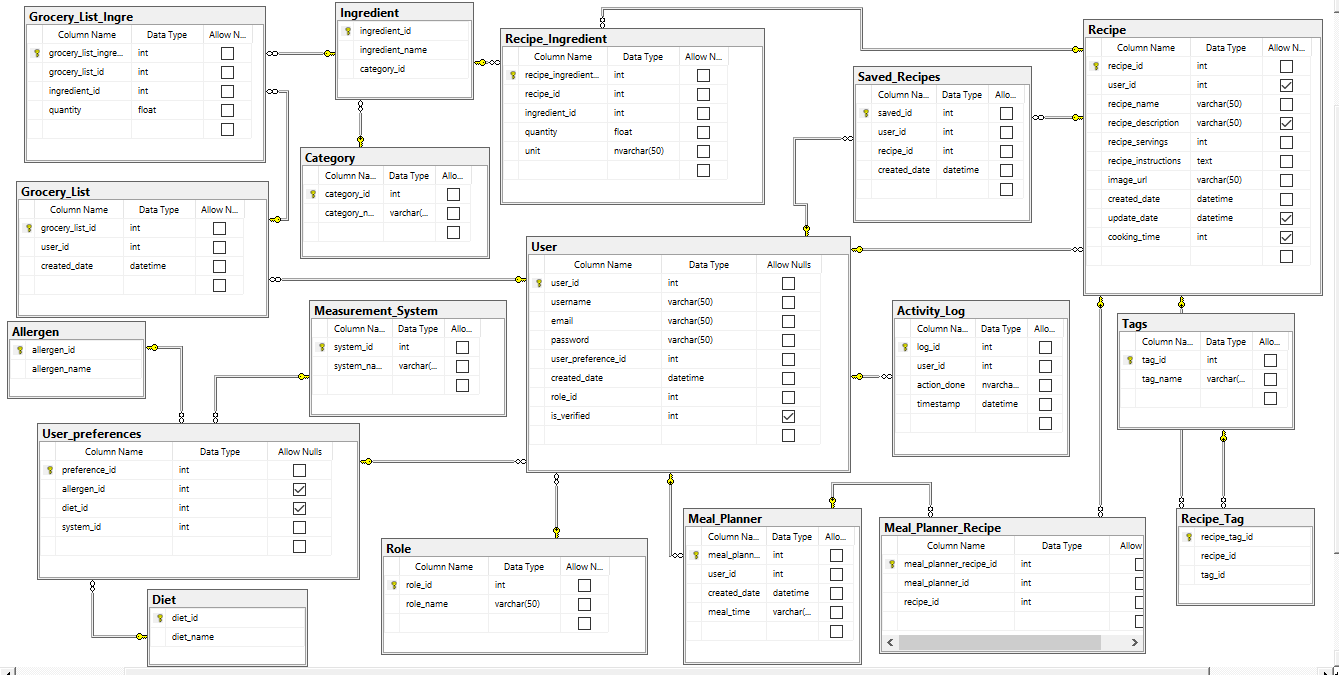


Figure 2. 3: Relational Database Diagram

2.9 User interface design

Figure 2.4 represents the login screen for the thyme to cook recipe app where users can enter their credentials to access the additional features of the app. Users can log in to the app using an email and password. A “Forgot Password?” option is also available for users who need to reset their password. Once users have successfully logged in, they will be navigated to their personalized home page.

A screenshot of a login form

Description automatically generated

Figure 2. 4: Login screen

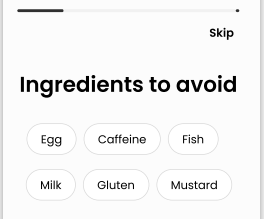
Figure 2.5 represents the sign-up screen for the Thyme to cook app where new users can create an account by providing a username, email address and password. This will also be the starting point to personalize their experience in the app.

A screenshot of a login form

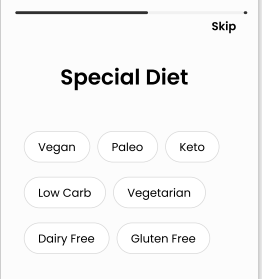
Description automatically generated

*Figure 2. 5: Signup screen*

Figure 2.6, Figure 2.7 and Figure 2.8 represent preference screens where users can choose their preferences and personalize their cooking experience. Figure 2.6 shows the ingredients to avoid screen where users can select ingredients they wish to avoid in recipes. Figure 2.7 shows the special diet selection screen where users can choose the specific diets they follow. Figure 2.8 shows the metric system preference screen where users can choose their preferred measurement system (metric or imperial).



*Figure 2. 6: Ingredients to ignore screen*



*Figure 2. 7: Special diet screen*

A screenshot of a measurement system

Description automatically generated

*Figure 2. 8: Measurement system preference screen*

Figure 2.9 represents the home screen which serves as the main dashboard where users can explore recipe recommendations based on their preferences, see recently viewed recipes and discover popular dishes. From here users can navigate to their profile as well.

A screenshot of a phone

Description automatically generated

*Figure 2. 9: Home screen*

Figure 2.10 represents the profile screen where users can view and edit their account details, see their kitchen activity (what they’ve been cooking), view created recipes and manage their settings.

A screenshot of a phone

Description automatically generated

*Figure 2. 10: Profile screen*

Figure 2.11 represents the search page where users can search for recipes by keyword or filter recipes based on ingredients, diets or nutrition. This screen allows users easy access to a wide range of recipes tailored to their specific needs.

A screenshot of a phone

Description automatically generated

*Figure 2. 11: Search screen*

Figure 2.12 represents the saved recipes page where a user’s recipe collections are kept. Users can then filter these recipes based on recently viewed or browse all saved recipes.

A screenshot of a phone

Description automatically generated

*Figure 2. 12: Saved recipes screen*

Figure 2.13 represents the meal planner screen where the user can plan meals for each day of the week, selecting recipes for breakfast, lunch, dinner and even snack time.

A screenshot of a phone

Description automatically generated

Figure 2. 13: Meal planner screen

Figure 2.14 represents the grocery list screen which helps users manage their grocery list based on the ingredients required for their selected recipes. Users can adjust ingredients and update their list as needed by adding or ticking off various ingredients. Figure 2.15 shows how users can then edit the quantity of the ingredients in their grocery list. These lists can then be shared with friends and family.

A screenshot of a cell phone

Description automatically generated

*Figure 2. 14: Grocery list screen*

A screenshot of a phone

Description automatically generated

*Figure 2. 15: Editing ingredient quantity*

Figure 2.16 represent the recipe detail screen which is broken up into two tabs, the ingredients tab and the instructions tab. The ingredients tab which shows the recipe ingredients with options to adjust serving sizes and convert between metric and imperial systems. The instructions tab which provides the recipe instructions and navigation to the cook with me feature shown in figure 2.17 which has integrated timers for easier cooking. Recipes can also be saved and added to their meal planner and grocery list.

A screenshot of a recipe

Description automatically generated

*Figure 2. 16:Recipe detail screen*

A screenshot of a phone

Description automatically generated

*Figure 2. 17: Cook with me feature*

Implementation documentation

3. Introduction

This document outlines the implementation of our Thyme To Cook recipe application and website, detailing all the tools and technologies that were used during development, complex code excerpts, and all the challenges encountered during development. The main purpose of this document is to explain the choices made during development, demonstrate the complexity of the code that drives the app’s unique features, and reflect on the technical problems faced and how they were resolved.

This document has 4 key sections. The Choice of tools section describes the specific tools, frameworks and APIs used in developing the app and why they were chosen. The extracts of code section displays code snippets of complex code that caused significant challenges during the development of the app and provides detailed explanations for each code snippet. The source code reference section provides references for any external code or resources we used or adapted for our project. The Problems encountered section provides a reflection on the technical difficulties we faced while developing the app, such as specific coding errors or technical limitations and how we eventually resolved them.

This document serves as the comprehensive overview of the technical implementation of our project.

3.1 Choice of Tools

**3.1.1 Visual Studio 2022**

We chose Visual Studio Code (VS code) 2022 as our primary Integrated Development Environment (IDE) because it is very lightweight, highly customizable and provides various useful extensions for Dart and Flutter. VS code integrates Git for version control which provides a useful way of working in a group. VS Code also allows for the projects to easily developed within isolated environments away from external packages on the system which made it the best choice for our app development.

**3.1.2 Firebase**

We used Firebase to store all recipe data, user details and images in real-time. Firebase was selected because of its built-in security features and smooth integration with flutter. Firebase handles authentication for user login and registration and ensures the data is handled securely. It also supports real-time updates through Firestore which allows for data synchronization when internet is available. Firebase has a console that has monitoring tools allowing us to track app usage and performance.

**3.1.3 Kaggle**

To build our initial recipe dataset, we used data from Kaggle, a platform with various datasets, we chose the “Food Ingredients and Recipes Dataset with Images” dataset (Sashi Goel, 2019) with license [CC BY-SA 3.0](https://creativecommons.org/licenses/by-sa/3.0/). This allowed us to quickly get a collection of recipes which were later enhanced using the Edamam API for nutritional information.

**3.1.4 Edamam API**

We used Edamam API to get accurate and detailed nutritional information for our recipe data which will be needed for fihltering and search functionality. The API provided various nutritional information such as diet labels, calories and macronutrients.

**3.1.5 Python and Java**

We used Java and Python to preprocess the recipe dataset before sending it to Firebase. Python was used to clean ad format the data into an appropriate format, so all our data entries were consistent. Java was used for automating the uploading process for images and recipes data to the Firebase and Firestore which saved a lot of time.

**3.1.6 Dart and Flutter**

We chose flutter and Dart to develop our application because of its ability to create native like applications for various platforms from a single codebase. Flutter also has a rich set of widgets and customizable components which provide a lot of flexibility when it comes to building visually appealing user interface.

**3.1.7 Figma**

Figma was used to create the interactive prototypes for the app. The designs were shared and used to guide our development.

**3.1.8 ChatGPT**

ChatGPT was used to create the logo, assist with general programming questions and colour palette queries.

3.2 Extracts of complex code

Figure 3.1 shows a loop that goes through all entries in the dataframe (csv file) and for every recipe ingredient list, the method “GetNutritionData” shown in Figure 3.2 is called which makes a request to the Edamam API and passes through the ingredient list (first converted to a literal string) and gets the nutritional information of the recipe. The API provides the calories, diet labels, health labels and total nutrients. To not hit the rate limit set by Edamam, the “GetNutritionData” method waits 10 seconds before retrying a request. For further protection the program also waits an additional second after every request and 1 minute after every 10 requests. The omitted if statements were each adding the specific nutritional information (calories, dietLabels, etc) to their respective column in the dataframe. Since there were cases where the program would fail halfway through, the dataframe gets saved to a separate csv file after every 100 requests to the API. The method ends off by doing a final save of all the recipe data to the recipes.csv file.



*Figure 3. 1: Code that saves relevant nutritional data to a csv file*



*Figure 3. 2: A method that requests nutritional information from Edamam API*

Figure 3.3 shows the formatIngredient method. The formatIngredient method is used to extract key details from an ingredient string from a recipe. This method finds the quantities, units, and the ingredient name from text strings that come in various formats, such as “1/2 cup sugar “or “200g flour”. This method makes use of two regex expressions (shown in Figure 3.4 and Figure 3.5) to match the ingredients to known quantity and unit patterns

The method starts by removing any extra inverted commas and spaces from the ingredient string. If the ingredient includes both a quantity and a unit (“2 cups” or “½ teaspoon”), this pattern is caught by the “quantity\_unit\_pattern” regular expression. Any fractions are then converted into proper numbers by the replaceFraction method for easier storage and later calculations.

If a unit is not found, the method searches for just a quantity using the “quantity\_pattern” which finds ingredient strings like “3 eggs”.

If there is no quantity and unit present in cases such as “pinch of salt”, the method checks if the first word in the string is a known unit such as “pinch” or “dash” and then extracts the ingredient name from the remaining words in the string disregarding the text before like “of”.

Finally, the method removes any extra information such as text in brackets like “¼ cup (60 g) mayonnaise” and replaces cases of “plus more” with empty space, leaving only important ingredient data which will be added to the database.

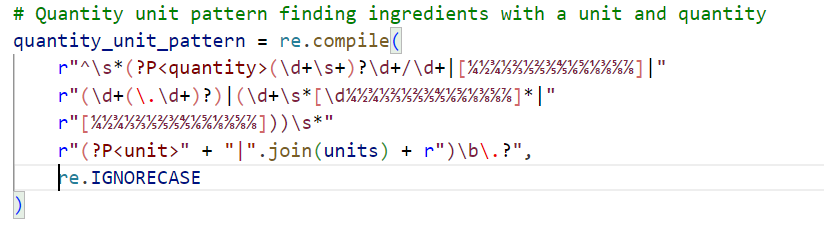
Figure 3.4 shows the “quantity\_unit\_pattern” which is responsible for identifying ingredient strings that contain both a quantity and a unit. It is flexible enough to match common formats like "1/2 cup," "100 g," or even fractions like "¾ teaspoon." The pattern captures number quantities and fractions and matches units from a predefined list ("cup," "teaspoon," "gram", etc.).

Figure 3.5 shows the “quantity\_pattern” which is used for cases where the ingredient has a quantity but no unit ("2 eggs"). This pattern finds fractions and number quantities, ensuring that even ingredients like "1/4" or "½" are correctly extracted.

A screenshot of a computer program

Description automatically generated

*Figure 3. 3: A method that formats ingredients from recipes*



*Figure 3. 4: Regex pattern that matches ingredients with quantities that are followed by units*

A screenshot of a computer program

Description automatically generated

*Figure 3. 5: Regex pattern that matches ingredients with just a quantity and no unit*

Figure 3.6 shows the ExtractTimeInfo method which is used to extract time-related data from the recipe instructions. This is done for the “Cook with me” feature of our app, which will help users follow recipes with built in timers.

The method uses two regex expressions which are also shown in Figure 3.7 and Figure 3.8 to identify both single times and time ranges ("bake for 20 minutes" or "simmer for 5 to 10 minutes").

First, the method looks for individual time mentions using the “time\_pattern”, which matches time units such as "minutes," "hours," or "seconds" along with their corresponding numeric values ("20 minutes" or "1 hour"). Each time match is converted into an integer and stored in a dictionary, which contains the time value and the unit, and this is added to the “time\_data” list.

Next, the method looks for ranges of time using the “time\_range\_pattern”, which captures expressions like "5 to 10 minutes" or "1 to 2 hours." Both the start and end times are extracted and converted into integers, with the unit identified. This data is then added to the time\_data list as well.

The method returns a list of dictionaries, where each dictionary contains either a single time or a time range with its corresponding unit, allowing the app to create timers based on the recipe’s instructions.

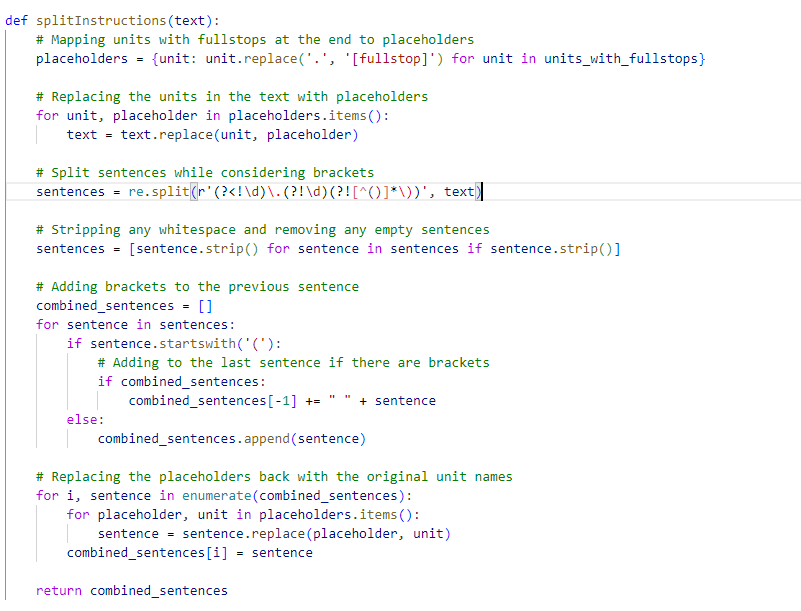
A screen shot of a computer program

Description automatically generated

*Figure 3. 6: A method that extracts the time from the instructions of a recipe*

Figure 3.7 shows the SplitInstructions method which is used to separate the recipe instructions into sentences while also ensuring that the units and brackets are handled correctly. The method starts by using a predefined list of unit (grams, litres,etc) called “units\_with\_fullstops” to create a placeholder for those type of units, this is done to prevent them from being seen as a sentence break.

The method then checks for any text that has such units with fullstops. The regex expression is then used to divide the text into sentences correctly. The method then handles any cases where the sentences start with bracket. This ensures instructions like “Mix ingredients (including the spices)” stay together. The method then replaces all the placeholders with the original unit names then returns the modified sentences as a list.



*Figure 3. 7: A method that splits instructions into list of strings*

3.3 Source code references

**3.3.1 Flutter and Firebase Tutorial**

**Description**

To learn more about the dart programming language, flutter and the fundamentals of firebase we followed a course created by Vandad Nahavandipoor.

**Usage**

* Flutter setup
* Login View
* Email Verification
* Firebase backend setup
* Auth service
* Migration to Firestore Service
* Bloc for routing and dialogs

**Reference**

freeCodeCamp.org (Director). (2022, February 24). *Flutter Course for Beginners – 37-hour Cross Platform App Development Tutorial* [Video recording]. <https://www.youtube.com/watch?v=VPvVD8t02U8>

**3.3.2 Implementing a grid view UI**

**Description**

A grid view was implemented to display and organize the UI elements efficiently. The GridView.builder was used to structure the layout dynamically.

**Usage**

The grid view is used to display the saved recipes on the Saved Recipes screen.

**Reference**

Mitch Koko (Director). (2022, November 4). *Donut App UI • Flutter Tutorial* [Video recording]. <https://www.youtube.com/watch?v=OmYL-VK75-o>

**3.3.3 Implementing Bottom Navigation Bar**

**Description**

To be able to navigate to different screens in the app, a navigation bar was needed that followed the Material Design 3 guidelines.

**Usage**

Used for the main navigation of the app

**Reference**

*BottomNavigationBar class—Material library—Dart API*. (n.d.). Retrieved 18 October 2024, from <https://api.flutter.dev/flutter/material/BottomNavigationBar-class.html>

**3.3.5 Uploading recipe data to Firebase Firestore**

To upload formatted recipe data from a CSV file to Firebase Firestore, the recipe ingredients, instructions, and nutritional information had to be extracted and formated to ensure that the data structure follows the Firebase's NoSQL schema. The code includes error handling to avoid data corruption during parsing, and it formats the data correctly before saving it to Firestore.

**Usage**

To upload recipe data (ingredients, instructions, calories etc) to the Firestore database.

**Reference**

ChatGPT (2024). Code assistance for formatting and uploading recipe data to Firebase Firestore. Retrieved October 17, 2024, from https://chatgpt.com.

3.4 Problems encountered

**3.4.1 Creating a Flutter App**

To create a cross-platform app we had to find a framework that would allow us to do so efficiently so we chose Flutter. We had never used Flutter and dart before, and the learning curve was quite steep

However, we found a 37-hour long video on YouTube that really helped us to gain insight on how to use Flutter and code an app using dart. The tutorial allowed us to learn and use that knowledge to start developing the app.

**3.4.2 Using Firebase**

Having only worked with relational databases before, it was quite an adjustment to work with Firebase and to understand how it is structured as well as the difference in formats. Watching some YouTube videos and playing around on Firebase helped us understand collections and documents better.

**3.4.3 Key Down Event error**

There was an issue with how keyboard events were being handled in the search field. This occurred because there was a conflict with the listener and input handling. Online forums suggested upgrading flutter since it is a common issue with how flutter handles physical keystrokes, but it seemed too risky. It was solved by having a check on the search controller text. Also, the error would loop because the text editing controller was being created every time the widget was built, so the search controller that was declared was called instead to fix it.

**3.4.4 Bottom Overflow Error**

Occasionally we encountered an error that would pop up whenever a text field was pressed and brought up the keyboard which was the bottom overflow by x pixels.

This error comes about when the screen is made to be static and then an element such as the keyboard will push the content below it off the screen. To fix this we wrapped the widget that is shown under a Single Child Scroll View which allows the screen to be scrollable.

**3.4.5 Right Overflow Error**

When designing the UI elements for the saved recipe screen there was a right overflow error as seen in Figure 3.8. This occurred when using the GridView.builder widget to allow two items per row. Causing an overflow due to the items overlapping.

To fix this error we needed to ensure the width of the grid items do not overflow by using a combination of crossAxisSpacing and padding. To avoid a bottom overflow error for the grid items we used mainAxisSpacing as well. The aspect ratio of the grid items was also changed so that all contents could be fitted inside.

A screenshot of a recipe

Description automatically generated

*Figure 3. 8: A screenshot that demonstrates the right overflow error*

**3.4.6 Rate limit and requests limit**

When working on getting nutritional information from the Edamam API, we encountered an issue which was causing the requests to fail. Because we had so many recipes in our original csv file, Edamam could not handle so many requests per second. To fix this we had to limit the number of requests to 10 then make the process sleep for 1 – 10 seconds. We encountered a similar problem when uploading the recipes data and images to Firebase Firestore which was fixed in a similar way. Edamam also had a limit on the number of requests we could make to the API per month, so we had to cut our original dataset of 13000 recipes to only 3000 recipes.

**3.4.7 Stack Overflow Error**

On the search screen where we were testing a placeholder model for displaying data, we got a stack overflow error shown in Figure 3.9. There were no errors highlighted in the code, but the debug console showed it was occurring because of the model class.

The problem occurred because it was calling the getRecipe() method, which caused an infinite loop of the data being displayed. To fix the issue the recipe list itself was returned.

A red square with a white border

Description automatically generated with medium confidence

*Figure 3. 9: A screenshot of the stack Overflow error on the search screen*

**3.4.8 Formatting Recipe ingredients and instructions**

The recipe ingredients and instructions were in format that was not very useful for our objectives such as the ingredients didn’t have the quantities, unit and the name of the ingredient laid out very nicely and the instructions did not have times specified which is needed for our “cook with me” feature. We sorted this out by implementing a formatting script that would go through all the recipes and extract such information before being put into the database.

**3.4.9 Regex Implementation**

Since we had never worked with regex before it was quite challenging to get a handle on how it works. After some digging, a regex cheat sheet really came in handy when it came to figuring out how to capture certain data (*Regex Cheat Sheet*, n.d.).