



**Module Code & Module Title**

**CS6PO5NT-Final Year Project**

**Assessment Weightage & Type**

**Final Year Project Final Report (40%)**

**Year and Semester**

**2023-24 Spring**

**Student Name: Aasmee Rai**

**London Met ID: 22072035**

**College ID: NP05CP4A220006**

**Assignment Submission Date: August 01, 2025**

**Internal Supervisor: Sonam Rai**

**External Supervisor: Utsav Dhungana**

**Title: Ingreedy (Smart Recipe App)**

*I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded*

# Abstract

The "Ingreedy" project proposes an innovative solution to the challenges of meal preparation and pantry management through the development of a Smart Recipe Finder application. Leveraging advanced technologies such as artificial intelligence (AI) for ingredient recognition and personalized recipe recommendations, the app addresses issues like food waste, inefficient grocery planning, and dietary accommodations. Key features include AI-powered ingredient recognition via OCR and barcode scanning, automated pantry management with expiry tracking and inventory alerts, and a personalized recipe generator that aligns with user preferences and available ingredients.

The app also fosters community engagement through a recipe-sharing platform and supports a diverse global audience with multilingual capabilities. Designed with user-friendliness, sustainability, and inclusivity in mind, "Ingreedy" enhances the cooking experience while promoting efficient ingredient usage. By incorporating a Scrum-based agile methodology, the project ensures iterative development and stakeholder feedback to achieve a functional and user-centric application. This project aims to redefine cooking and pantry management, making meal preparation an efficient, enjoyable, and resourceful process

Table of Contents

[Abstract i](#_Toc196685831)

[Chapter 1: Introduction 1](#_Toc196685832)

[1.1. Introduction to the topic 1](#_Toc196685833)

[1.2. Problem Statement 1](#_Toc196685834)

[1.2.1. Survey for the project 2](#_Toc196685835)

[1.3. Project as a Solution 2](#_Toc196685836)

[1.4. Aims 2](#_Toc196685837)

[1.5. Objectives 2](#_Toc196685838)

[Chapter 2: Background 3](#_Toc196685839)

[2.1. About client/end user 3](#_Toc196685840)

[2.2. Understanding the solution 3](#_Toc196685841)

[2.3. Review of similar projects 3](#_Toc196685842)

[2.4. Comparisons between similarity of the systems 4](#_Toc196685843)

[2.5. Review of technical aspects 4](#_Toc196685844)

[Chapter 3: Development 5](#_Toc196685845)

[3.1. Approach / Methodology Considerations 5](#_Toc196685846)

[3.2. Stages of chosen Approach / Methodology 5](#_Toc196685847)

[3.2.1. Work Breakdown Structure 5](#_Toc196685848)

[3.2.2. Gantt Chart 5](#_Toc196685849)

[3.3. Survey Results 5](#_Toc196685850)

[3.3.1. Pre-Survey Results 5](#_Toc196685851)

[3.3.2. Post-Survey Results 5](#_Toc196685852)

[3.4. Requirement Analysis 6](#_Toc196685853)

[3.4.1. SRS 6](#_Toc196685854)

[3.5. Designs 8](#_Toc196685855)

[3.5.1. Entity Relation Diagram 8](#_Toc196685856)

[3.5.2. Data Flow Diagram (DFD) 13](#_Toc196685857)

[3.5.3. Use Case Diagrams 14](#_Toc196685858)

[3.6. High-level Expanded Use Case Diagram 15](#_Toc196685859)

[3.6.1. Add Ingredients 15](#_Toc196685860)

[3.6.2. Search Recipe 17](#_Toc196685861)

[3.6.3. Make Shopping List 19](#_Toc196685862)

[3.6.4. Make Post 21](#_Toc196685863)

[3.7. Communication Diagram 23](#_Toc196685864)

[3.8. Class Diagram 24](#_Toc196685865)

[3.9. Activity Diagram 25](#_Toc196685866)

[3.10. Flowchart 26](#_Toc196685867)

[Chapter 4: Testing and Analysis 27](#_Toc196685868)

[4.1. Test Plan 27](#_Toc196685869)

[4.2. Unit Testing 27](#_Toc196685870)

[4.3. System Testing 27](#_Toc196685871)

[4.4. Critical Analysis 27](#_Toc196685872)

[Chapter 5: 28](#_Toc196685873)

[5.1. Conclusion 28](#_Toc196685874)

[5.2. Advantages 28](#_Toc196685875)

[5.3. Limitations 28](#_Toc196685876)

[Chapter 6: Future Work 29](#_Toc196685877)

[Chapter 7: References 30](#_Toc196685878)

[Chapter 8: Appendix 31](#_Toc196685879)

[8.1. Appendix A: Pre-Survey 31](#_Toc196685880)

[8.1.1. Pre-Survey Form 31](#_Toc196685881)

[8.1.2. Sample of Filled Pre-Survey Forms 31](#_Toc196685882)

[8.1.3. Pre-Survey Result 31](#_Toc196685883)

[8.2. Appendix B: Post-Survey 31](#_Toc196685884)

[8.2.1. Post-Survey Form 31](#_Toc196685885)

[8.2.2. Sample of Filled Post-Survey Forms 31](#_Toc196685886)

[8.2.3. Post-Survey Result 31](#_Toc196685887)

[8.3. Appendix C: Sample Codes 31](#_Toc196685888)

[8.3.1. Sample Code of The Ui 31](#_Toc196685889)

[8.3.2. Sample Code for The Automation Script 32](#_Toc196685890)

[8.4. Appendix D: Designs 32](#_Toc196685891)

[8.4.1. Gantt Chart 32](#_Toc196685892)

[8.4.2. Work Breakdown Structure 32](#_Toc196685893)

[8.4.3. Algorithms & Flowcharts 32](#_Toc196685894)

[8.4.4. 3d Modeling 32](#_Toc196685895)

[8.4.5. Hardware Architecture 32](#_Toc196685896)

[8.4.6. Data Flow Diagrams 32](#_Toc196685897)

[8.4.7. Use Case 32](#_Toc196685898)

[8.4.8. Wireframe 33](#_Toc196685899)

[8.5. Appendix E: Screenshots of The System 33](#_Toc196685900)

[8.6. Appendix F: User Feedback 33](#_Toc196685901)

[8.6.1. User Feedback Form 33](#_Toc196685902)

[8.6.2. Sample of Filled User Feedback Forms 33](#_Toc196685903)

[8.7. Appendix G: Future Work 33](#_Toc196685904)

[8.7.1. Readings for Future Work 33](#_Toc196685905)

Table of Figures

[Figure 1: Collaboration diagram of “Manage Pantry” 15](#_Toc196684896)

[Figure 2: Sequence diagram of “Manage Pantry” 15](#_Toc196684897)

[Figure 3: Collaboration diagram of “Search Recipe” 16](#_Toc196684898)

[Figure 4: Sequence diagram for “Search Recipe” 17](#_Toc196684899)

[Figure 5: Collaboration of “Make Shopping List” 18](#_Toc196684900)

[Figure 6: Sequence diagram for “Make Shopping List” 19](#_Toc196684901)

[Figure 7: Collaboration of “Make Post” 20](#_Toc196684902)

[Figure 8: Sequence diagram for “Make Post” 21](#_Toc196684903)

Table of Tables

[Table 1: Similarity comparison 2](#_Toc196560147)

Table of Abbreviations

|  |  |
| --- | --- |
| Abbreviations | Full Form |
| AI | Artificial Intelligence |
|  |  |

# Introduction

## Introduction to the topic

Ingreedy (Smart Recipe Finder) transforms meal preparation with innovative features like AI-powered ingredient recognition, personalized recipe suggestions, pantry tracking, and automatic shopping list generation. The app helps users make the most of their ingredients, explore recipes tailored to their dietary preferences, and plan meals efficiently.

Beyond its practical benefits, Ingreedy fosters creativity by encouraging users to try new recipes and reduce food waste. Its interactive community forum allows users to share tips, post recipes, and connect with like-minded cooking enthusiasts. Designed for convenience, sustainability, and inclusivity, Ingreedy ensures cooking is an enjoyable and resourceful experience for users of all skill levels.

## Problem Statement

In today’s fast-paced lifestyle, meal preparation often becomes a challenging task for individuals, families, and even professional cooks. A common problem is the inefficient use of available ingredients, leading to food waste and missed opportunities to create meals from what’s already in the pantry. (Porter & Reay, 2015) People often find it time-consuming and tedious to plan meals, especially when trying to accommodate dietary restrictions, nutritional goals, or personal preferences. (Huff, 2022)

Another significant issue is the lack of tools to manage pantry inventory effectively. Ingredients may expire or run out unnoticed, causing disruptions in meal planning and unnecessary trips to the grocery store. (THE OWL, 2024) The abundance of online recipes can overwhelm users, making it hard to find ones that suit their tastes or needs. Additionally, there's a growing demand for community and inspiration in cooking, highlighting a gap for an all-in-one kitchen management and recipe discovery tool.

### Survey for the project

A screenshot of a survey form

Description automatically generated

A screenshot of a survey

Description automatically generated

A screenshot of a survey

Description automatically generated

A screenshot of a survey

Description automatically generated

A screenshot of a survey

Description automatically generated

A screenshot of a computer

Description automatically generated

## Project as a Solution

The Ingreedy (Smart Recipe Finder) simplifies cooking and pantry management with a user-friendly platform powered by AI. By using photos or barcode scans, users can quickly digitize their pantry contents, enabling the app to suggest recipes based on the ingredients they already have. This feature helps reduce food waste, save time, and make meal preparation more efficient.

In addition to recipe suggestions, Ingreedy tracks ingredient quantities, monitors expiry dates, and sends timely alerts for low-stock or near-expiry items, helping users stay organized and plan meals effectively. The app also generates shopping lists for missing ingredients and allows users to customize recommendations based on dietary preferences or nutritional goals, ensuring a personalized experience.

Beyond practicality, Ingreedy fosters a sense of community through a recipe-sharing forum where users can exchange ideas, discover new techniques, and explore diverse cuisines. With multilingual support, the app is accessible to a global audience, transforming cooking into a creative, efficient, and enjoyable activity for users around the world.

## Aims

## Objectives

# Background

## About client/end user

The end users for the "Ingreedy" app are individuals or families looking to simplify their cooking and pantry management processes. This includes users with dietary restrictions, culinary enthusiasts, or people aiming to reduce food waste. The app is also designed for a global audience with multilingual support to ensure inclusivity.

## Understanding the solution

The app leverages AI-powered OCR for ingredient recognition, personalized recipe recommendations, and efficient pantry management. It integrates advanced technology to automate inventory tracking, provide dietary customization, and generate automated shopping lists. Additionally, the app fosters a community environment where users can share recipes and engage with others.

## Review of similar projects

Several projects and applications like Ingreedy have been developed in the past, each addressing various aspects of meal preparation and pantry management. Below is a concise review of some noteworthy examples.

1. **Yummly**

* The Yummly app is a smart cooking companion, offering personalized recipe recommendations tailored to your tastes, dietary preferences, and allergies. With features like ingredient-based searches to reduce food waste, quick recipe filters, a digital cookbook for saving favorites, and step-by-step guided cooking, Yummly makes mealtime effortless. Its smart shopping list and meal planning tools streamline your grocery trips and daily schedules, while Yummly Premium unlocks guided videos, nutritional insights, and advanced planning options. Whether you're exploring new cuisines or optimizing leftovers, Yummly simplifies cooking for every lifestyle. (Aptoide, 2025)

1. **Tasty**

* Tasty is a comprehensive cooking platform offering over 10,000 recipes, combining detailed instructions with video content and personalized recommendations. Its AI assistant, Botatouille, uses user preferences to suggest recipes, encouraging exploration of diverse culinary options. The platform integrates with Walmart, allowing users to shop for ingredients and arrange delivery or pickup directly through the app. Tasty also fosters a community-driven experience, enabling users to share recipes, gain inspiration, and exchange cooking knowledge. Additional features include custom recipe collections, an ingredient-based search function, adjustable serving sizes, and insights from creators and community feedback, making it a versatile tool for home cooks. (Sensor Tower, 2025)

1. **Cookpad**

* Cookpad is a global platform for recipes and meal planning, showcasing home-cook-created dishes with clear, step-by-step instructions. Users can search for recipes by ingredients to reduce food waste, organize them into custom collections, and plan weekly menus. The app also features a vibrant cooking community where users can share recipes, exchange tips, and post photos of their creations. Designed for all skill levels and dietary needs, Cookpad provides inspiration for everything from quick breakfasts to intricate dinners and international cuisine. (Aptoide, 2025)

## Comparisons between similarity of the systems

Table : Similarity comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Features | Yummly | Tasty | Cookpad | Ingreedy |
| AI-powered ingredient recognition | No | No | No | Yes |
| Pantry Management | No | No | No | Yes |
| Recipe generation based on pantry items | No | No | No | Yes |
| Profile management | Yes | Minimal | Yes | Yes |
| Community engagement | Yes | No | Yes | Yes |
| Automatic shopping list generation | Yes | No | No | Yes |

## Review of technical aspects

# Development

## Approach / Methodology Considerations

For the development of the Smart Recipe Finder App, several development methodologies were carefully evaluated to determine the most suitable approach for the project's needs and objectives. Each methodology presented unique strengths and weaknesses when applied to the dynamic and feature-rich nature of this project.

* **Waterfall**
* Waterfall provided a structured and linear framework, offering clear stages for planning, design, development, and testing. However, its rigid structure proved unsuitable for accommodating evolving requirements and iterative refinements, which are critical for a project of this scope and complexity.
* **Kanban**
* Kanban excelled in tracking workflow and ensuring visibility across tasks, promoting a smooth and efficient process. Despite its strengths in workflow management, it lacked time-boxed iterations, which are essential for managing the complex and interdependent features of the app, such as AI-based ingredient recognition and pantry management.
* **Lean**
* Lean emphasized minimizing waste and maximizing efficiency, aligning well with the goal of delivering value-driven outcomes. However, it fell short in supporting the frequent collaboration, adaptive planning, and iterative refinement necessary for such a dynamic and user-centric project.

By understanding the limitations and benefits of each approach, the team was able to make an informed decision to adopt a methodology that best aligns with the app's goals and ensures successful delivery.

## Stages of chosen Approach / Methodology

After evaluating the Waterfall, Kanban, and Lean methodologies, Scrum was chosen as the most suitable methodology for the development of the "Ingreedy" Smart Recipe Finder app. Scrum's emphasis on flexibility, collaboration, and iterative progress makes it ideal for handling the app's dynamic requirements and complex feature set. Below is a brief description of the key phases in Scrum:

* Product Backlog Creation
* The team identifies and prioritizes all the tasks and features required for the project. These are documented in the Product Backlog, serving as a dynamic to-do list.
* Sprint Planning
* Before each sprint (a time-boxed iteration), the team selects high-priority tasks from the Product Backlog and defines clear goals for that sprint.
* Sprint Execution
* During the sprint, the team works collaboratively to develop the selected features. Daily Scrum meetings are held to discuss progress, address obstacles, and ensure alignment.
* Sprint Review
* At the end of the sprint, the team demonstrates the completed work to stakeholders, gathers feedback, and identifies areas for improvement.
* Sprint Retrospective
* The team reflects on the sprint process to identify successes, challenges, and opportunities for improvement in future iterations.
* Increment Delivery
* Each sprint delivers a functional increment of the app, contributing to the project’s overall progress and ensuring continuous value delivery.

This iterative process ensures that the app evolves based on stakeholder feedback, allowing the team to adapt to changing requirements while maintaining a focus on delivering high-quality features.

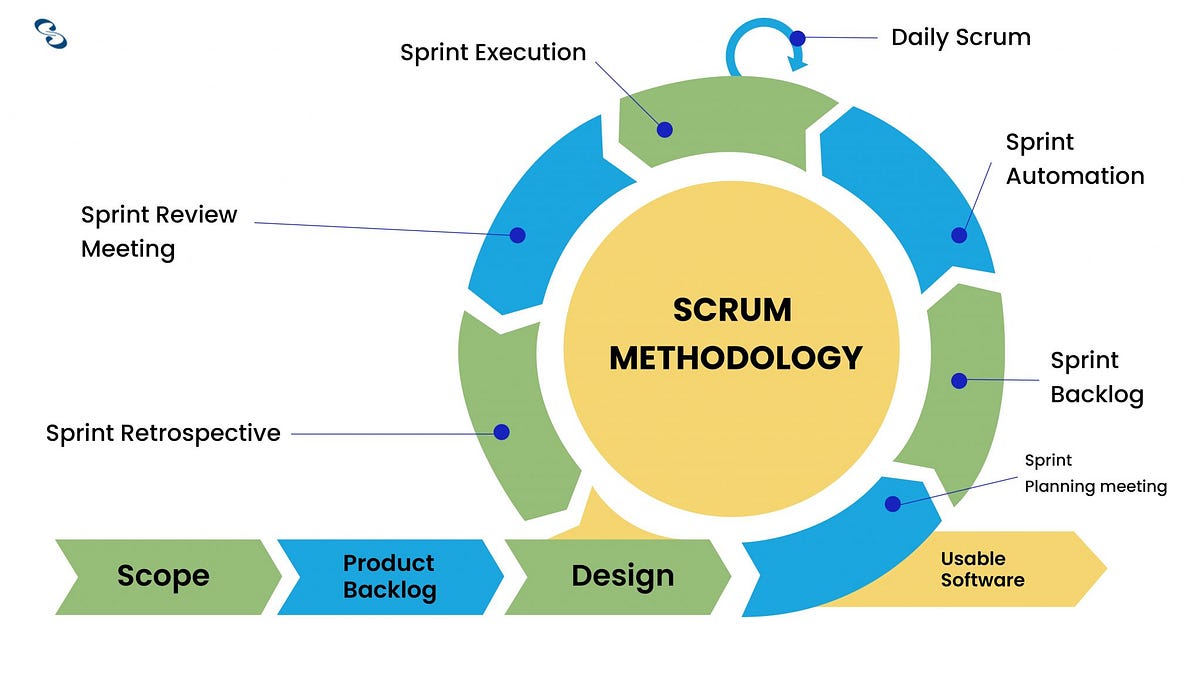


Figure : Scrum Methodology

### Work Breakdown Structure

### Gantt Chart

## Survey Results

### Pre-Survey Results

## Post-Survey Results

## Requirement Analysis

### SRS

**Functional requirements**

**Add Ingredients:**

Users can add ingredients manually or via an OCR scanner, and the system will store details such as expiry dates and stock levels. Notifications will be sent for items nearing expiration or running low.

**Search Recipe:**

Users can search for recipes based on available ingredients and apply filters like dietary preferences. The system will display a personalized list of recipe suggestions.

**Make Shopping List:**

Users can generate a shopping list by selecting ingredients that are low in stock or needed for recipes.

**Make Post:**

Users can create and publish posts in the community forum after content validation by the system.

**Non-functional requirements**

1. **Design and Implementation constraints**

The "Ingreedy" application is designed with scalability, security, and usability in mind. The system must handle a growing user base without compromising performance, ensuring smooth functionality even with large datasets of recipes and ingredients. Security protocols like data encryption safeguard user information and pantry details. Additionally, the technology stack integrates AI for OCR and barcode scanning, ensuring seamless ingredient recognition and inventory management.

1. **External interfaces required:**

**User interfaces**

* The app provides an intuitive, mobile-friendly interface that allows users to easily add ingredients, search recipes, manage their pantry, and engage with the community. The design prioritizes accessibility and ease of navigation to cater to users of all skill levels.

**Hardware interfaces**

* Integration with smartphone cameras enables users to scan barcodes and capture images of ingredients for OCR-based recognition. This feature ensures efficient digitization of pantry data, making the app accessible to a wide range of devices.

**Software interfaces**

* The application interacts with APIs for fetching recipe data, cloud storage for managing pantry information, and third-party libraries for OCR and barcode functionality. This modular approach ensures the app's robustness and maintainability.

**Communication interfaces**

* The app requires a stable internet connection to access cloud-based functionalities like inventory synchronization, personalized recipe suggestions, and community forum interactions. It supports real-time data transfer to keep user information up to date across devices.

## Designs

### Entity Relation Diagram

A diagram of a computer

AI-generated content may be incorrect.

**Entity and their atributes:**

**users**

* + UID (Primary Key)
  + username
  + email
  + password
  + profile\_pic
  + createdAt

**posts**

* + postID (Primary Key)
  + UID\* (Foreign Key)
  + caption
  + comments\_allowed
  + created\_at

**post\_images**

* + id (Primary Key)
  + post\_id\* (Foreign Key)
  + image\_url

**comments**

* + comID (Primary Key)
  + postID\* (Foreign Key)
  + UID\* (Foreign Key)
  + content
  + created\_at

**likes**

* + likeId (Primary Key)
  + UID\* (Foreign Key)
  + postID\* (Foreign Key)
  + created\_at

**bookmarks**

* + markID (Primary Key)
  + UID\* (Foreign Key)
  + postID\* (Foreign Key)
  + created\_at

**followers**

* id (Primary Key)
* follower\_user\_id
* followed\_user\_id
* createdAt

**pantry**

* itemID (Primary Key)
* UID\* (Foreign Key)
* itemName
* category
* quantity
* unit
* expiryDate
* createdAt
* updatedAt

**recipes**

* + recipeId (Primary Key)
  + recipeName
  + description
  + image\_url

DROP TABLE recipe\_ingredients;

**recipe\_ingredients**

* + ID (Primary Key)
  + ingredientName
  + recipeId\* (Foreign Key)
  + quantity
  + unit

**recipe\_steps**

* + stepId (Primary Key)
  + recipeId\* (Foreign Key)
  + stepNumber
  + stepDescription

**saved\_recipes**

* + id (Primary Key)
  + recipeId\* (Foreign Key)
  + UID\* (Foreign Key)
  + savedAt

**tags**

* + tagID (Primary Key)
  + name

**recipe\_tags**

* + id (Primary Key)
  + recipeID\* (Foreign Key)
  + tagID\* (Foreign Key)

**search\_history**

* + historyID (Primary Key)
  + UID\* (Foreign Key)
  + search\_query
  + applied\_tags
  + created\_at

**recipe\_views**

* + viewID (Primary Key)
  + UID INT (Foreign Key)
  + recipeID INT (Foreign Key)
  + viewed\_at

**Relationship:**

|  |  |  |
| --- | --- | --- |
| From Table | → To Table | Relationship Type |
| users | posts | 1-to-many |
| users | pantry | 1-to-many |
| users | saved\_recipes | 1-to-many |
| users | search\_history | 1-to-many |
| users | comments | 1-to-many |
| users | likes | 1-to-many |
| users | bookmarks | 1-to-many |
| users (self) | followers | many-to-many (self-ref) |
| posts | post\_images | 1-to-many |
| posts | likes, bookmarks | 1-to-many |
| posts | comments | 1-to-many |
| recipes | recipe\_ingredients | 1-to-many |
| recipes | recipe\_steps | 1-to-many |
| recipes | recipe\_tags | many-to-many |
| recipes | saved\_recipes | many-to-many |
| recipes | recipe\_views | many-to-many |
| tags | recipe\_tags | many-to-many |

**Entity-Relationship Breakdown**

**users (UID)**

Central entity — related to almost everything:

1 user → many posts, likes, comments, bookmarks, saved\_recipes, pantry items, search\_history, recipe\_views, and followers.

**Posts / Community**

1 user → many posts

1 post → many comments, likes, bookmarks, post\_images

**Relationships:**

posts.UID → users.UID

post\_images.post\_id → posts.postID

comments.postID → posts.postID, comments.UID → users.UID

likes.UID/postID, bookmarks.UID/postID → composite keys

**🛒 Pantry**

1 user → many pantry items

**Relationship:**

pantry.UID → users.UID

**🍲 Recipes**

Each recipe:

* + Has many ingredients (via recipe\_ingredients)
  + Has many steps (recipe\_steps)
  + Has many tags (recipe\_tags)
  + Can be saved by many users (saved\_recipes)
  + Can be viewed (recipe\_views)

**Relationships:**

recipe\_ingredients.recipeId → recipes.recipeId

recipe\_steps.recipeId → recipes.recipeId

recipe\_tags.recipeID → recipes.recipeId

saved\_recipes.recipeId → recipes.recipeId

recipe\_views.recipeId → recipes.recipeId

Optional:

* You can add recipes.createdBy → users.UID if users can submit recipes

**🏷️ Tags**

* Many-to-many between recipes and tags

**Relationships:**

recipe\_tags.recipeID → recipes.recipeId

recipe\_tags.tagID → tags.tagID

**❤️ Likes & Bookmarks**

* 1 user can like/bookmark many posts
* 1 post can be liked/bookmarked by many users

**Relationships:**

* likes.UID, bookmarks.UID → users.UID
* likes.postID, bookmarks.postID → posts.postID

**💬 Comments**

* Each comment is linked to a user and a post

**Relationships:**

* comments.UID → users.UID
* comments.postID → posts.postID

**📚 Search & View History**

* 1 user → many search history entries
* 1 user → many recipe views

**Relationships:**

* search\_history.UID → users.UID
* recipe\_views.UID → users.UID
* recipe\_views.recipeID → recipes.recipeId

**👥 Followers**

* Self-referencing table for users following users

**Relationship:**

* followers.follower\_user\_id and followers.followed\_user\_id → users.UID

**📐 Final Diagram Summary (Relationships)**

| **From Table** | **→ To Table** | **Relationship Type** |
| --- | --- | --- |
| users | posts | 1-to-many |
| users | pantry | 1-to-many |
| users | saved\_recipes | 1-to-many |
| users | search\_history | 1-to-many |
| users | comments | 1-to-many |
| users | likes | 1-to-many |
| users | bookmarks | 1-to-many |
| users (self) | followers | many-to-many (self-ref) |
| posts | post\_images | 1-to-many |
| posts | likes, bookmarks | 1-to-many |
| posts | comments | 1-to-many |
| recipes | recipe\_ingredients | 1-to-many |
| recipes | recipe\_steps | 1-to-many |
| recipes | recipe\_tags | many-to-many |
| recipes | saved\_recipes | many-to-many |
| recipes | recipe\_views | many-to-many |
| tags | recipe\_tags | many-to-many |

Would you like me to generate this as a **visual ERD** (PDF or image)? Or export a .dbml or .sql file for dbdiagram.io or MySQL Workbench import?

### Data Flow Diagram (DFD)

### Use Case Diagrams

## High-level Expanded Use Case Diagram

### Add Ingredients

**Use case:** Manage Pantry

**Actor:** User

**Description:** Users add ingredients to the system, which stores their details, including expiry dates. It actively monitors the ingredients and tracks their status. Notifications are sent for items nearing expiration.

**Typical Course of Actions:**

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. Users add ingredients, manually or via OCR scanner. | 2. System stores ingredient data in the database. |
|  | 3. System tracks expiry dates and stock levels of ingredients using stored data. |
|  | 4. System detects ingredients nearing expiry or running low. |
| 6. Users receive and review the notifications | 5. System sends notifications about expiry and low stock. |

**Domain Class:**

1. Ingredients
2. Pantry

**Collaboration diagram:**

A screenshot of a computer program

Description automatically generated

Figure : Collaboration diagram of “Manage Pantry”

**Sequence diagram:**

A diagram of a computer program

Description automatically generated with medium confidence

Figure : Sequence diagram of “Manage Pantry”

### Search Recipe

**Use case:** Search Recipe

**Actor:** User

**Description:** Users search for recipes, and the system generates a list of results. Users apply filters to narrow down the list. The system updates the list based on the selected filters.

**Typical Course of Actions:**

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. Users navigate to recipe page. | 2. System displays recipe list from the database. |
| 3. Users filter the recipe list based on their needs. | 4. System displays the filtered recipe list. |

**Domain Class:**

1. Recipe

**Collaboration diagram:**

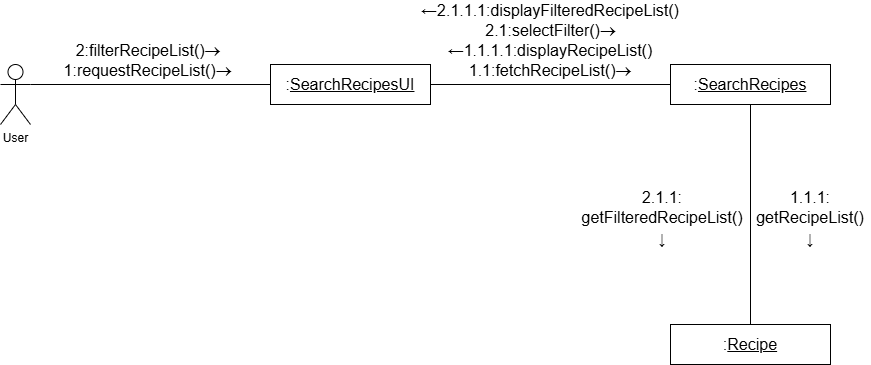


Figure : Collaboration diagram of “Search Recipe”

**Sequence diagram:**

A screenshot of a computer

Description automatically generated

Figure : Sequence diagram for “Search Recipe”

### Make Shopping List

**Use case:** Make Shopping List

**Actor:** User

**Description:** Users select ingredients that are low in stock or need to be added for a recipe or that was finished. The system provides an ingredient list for users to select from. The shopping list is then generated.

**Typical Course of Actions:**

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. Users navigate to shopping list page. | 2. System displays the user’s already existing list. |
| 2. Users add ingredients to the list. | 4. System add the newly selected ingredients to the list. |

**Domain Class:**

1. Ingredients
2. Shopping List

**Collaboration diagram:**

A screenshot of a computer

Description automatically generated

Figure : Collaboration of “Make Shopping List”

**Sequence diagram:**

A diagram of a diagram

Description automatically generated

Figure : Sequence diagram for “Make Shopping List”

### Make Post

**Use case:** Make Post

**Actor:** User

**Description:** Users input the content for a post, and the system validates it. Once validated, users submit the post. The system then publishes it.

**Typical Course of Actions:**

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. Users inputs the content of the post. | 2. System validates the content. |
| 3. Users submit the post. | 4. System creates new post and stores it. |
|  | 5. System confirms that the post was created. |

**Domain Class:**

1. Post

**Collaboration diagram:**

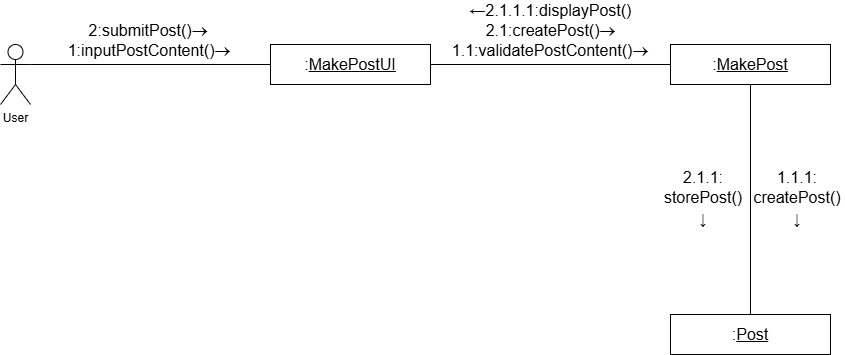


Figure : Collaboration of “Make Post”

**Sequence diagram:**

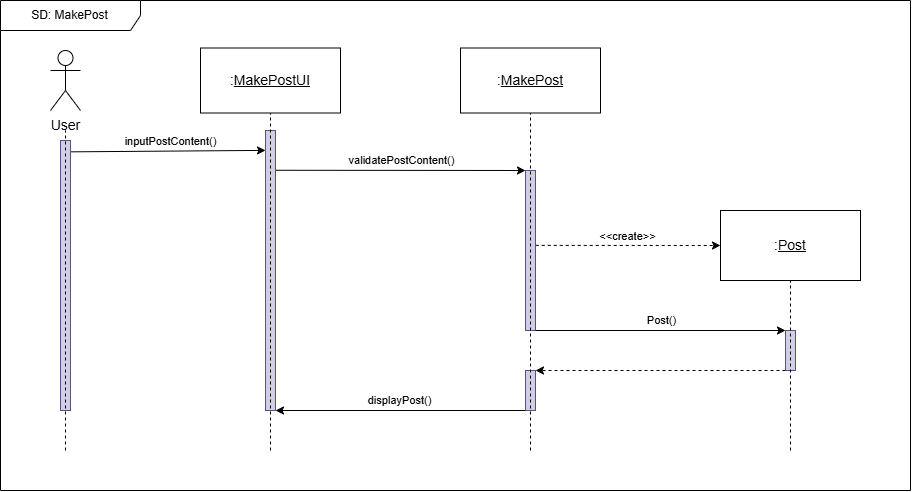


Figure : Sequence diagram for “Make Post”

## Communication Diagram

## Class Diagram

## Activity Diagram

## Flowchart

# Testing and Analysis

## Test Plan

## Unit Testing

## System Testing

## Critical Analysis

# 

## Conclusion

## Advantages

## Limitations

# Future Work

# References

1. Aptoide. (2025). *Cookpad recipes, homemade food -* . Retrieved January 06, 2025, from https://cookpad.en.aptoide.com/app
2. Aptoide. (2025). *Yummly Recipes & Cooking Tools*. Retrieved January 06, 2025, from https://yummly.en.aptoide.com/app
3. Huff, B. (2022). *'What shall we have for dinner?' Choice overload is a real problem, but these tips will make your life easier*. Retrieved December 1, 2023, from https://theconversation.com/what-shall-we-have-for-dinner-choice-overload-is-a-real-problem-but-these-tips-will-make-your-life-easier-193317
4. Porter, S. D., & Reay, D. S. (2015). *Addressing food supply chain and consumption inefficiencies: potential for climate change mitigation*. Retrieved December 1, 2023, from https://link.springer.com/article/10.1007/s10113-015-0783-4
5. Sensor Tower. (2025). *Tasty: Recipes, Cooking Videos - Overview*. Retrieved January 06, 2025, from https://app.sensortower.com/overview/1217456898?country=US
6. THE OWL. (2024). *What is Inventory Expiration and How Can Food Manufacturers Prevent It?* Retrieved December 1, 2024, from https://theowlsolutions.com/inventory-expiration-and-how-to-prevent-it/

# Appendix

## Appendix A: Pre-Survey

### Pre-Survey Form

### Sample of Filled Pre-Survey Forms

### Pre-Survey Result

## Appendix B: Post-Survey

### Post-Survey Form

### Sample of Filled Post-Survey Forms

### Post-Survey Result

## Appendix C: Sample Codes

### Sample Code of The Ui

#### Login Page

import 'package:flutter/material.dart';

import 'package:frontend/services/authServices.dart';

import 'package:frontend/widgets/button.dart';

import 'package:frontend/widgets/txtfield.dart';

class LoginPage extends StatefulWidget {

  @override

  \_LoginPageState createState() => \_LoginPageState();

}

class \_LoginPageState extends State<LoginPage> {

  final emailController = TextEditingController();

  final passwordController = TextEditingController();

  bool rememberMe = false;

  void login() async {

    final response = await AuthService.login(

      emailController.text,

      passwordController.text,

    );

    if (response['token'] != null) {

      Navigator.pushReplacementNamed(context, '/navbar');

    } else {

      ScaffoldMessenger.of(context).showSnackBar(

        SnackBar(content: Text(response['message'] ?? 'Login failed')),

      );

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Container(

        decoration: BoxDecoration(

          image: DecorationImage(

            image: AssetImage('images/login.jpeg'),

            fit: BoxFit.cover,

          ),

        ),

        child: Center(

          child: Padding(

            padding: const EdgeInsets.symmetric(horizontal: 15),

            child: SingleChildScrollView(

              child: Column(

                children: [

                  const Text(

                    "Login",

                    style: TextStyle(

                      color: Colors.white,

                      fontSize: 32,

                      fontWeight: FontWeight.bold,

                    ),

                  ),

                  const Text(

                    "Welcome back!",

                    style: TextStyle(color: Colors.white, fontSize: 16.5),

                  ),

                  SizedBox(height: 16),

                  Textfield(

                    controller: emailController,

                    hintText: "Email",

                    label: "Email",

                    labelColor: Colors.white,

                    keyboardType: TextInputType.emailAddress,

                  ),

                  const SizedBox(height: 9),

                  Textfield(

                    controller: passwordController,

                    hintText: "Password",

                    label: "Password",

                    labelColor: Colors.white,

                    isPassword: true,

                  ),

                  Row(

                    children: [

                      Checkbox(

                        value: rememberMe,

                        onChanged: (val) => setState(() => rememberMe = val!),

                      ),

                      Text(

                        "Remember me",

                        style: TextStyle(color: Colors.white),

                      ),

                      Spacer(),

                      TextButton(

                        onPressed: () {},

                        child: Text(

                          "Forgot password?",

                          style: TextStyle(color: Colors.white),

                        ),

                      ),

                    ],

                  ),

                  Button(

                    text: "Login",

                    color: Colors.white,

                    txtColor: Colors.black,

                    onPressed: login,

                  ),

                  Row(

                    mainAxisAlignment: MainAxisAlignment.center,

                    children: [

                      const Text(

                        "Don't have an account?",

                        style: TextStyle(color: Colors.white, fontSize: 14),

                      ),

                      TextButton(

                        onPressed:

                            () => Navigator.pushNamed(context, '/register'),

                        child: const Text(

                          "Register",

                          style: TextStyle(color: Colors.red, fontSize: 14),

                        ),

                      ),

                    ],

                  ),

                ],

              ),

            ),

          ),

        ),

      ),

    );

  }

}

#### Register Page

import 'package:flutter/material.dart';

import 'package:frontend/services/authServices.dart';

import 'package:frontend/widgets/button.dart';

import 'package:frontend/widgets/txtfield.dart';

class RegisterPage extends StatefulWidget {

  @override

  \_RegisterPageState createState() => \_RegisterPageState();

}

class \_RegisterPageState extends State<RegisterPage> {

  final nameController = TextEditingController();

  final emailController = TextEditingController();

  final passwordController = TextEditingController();

  final confirmPasswordController = TextEditingController();

  bool agreeTerms = false;

  void register() async {

    final response = await AuthService.register(

      nameController.text,

      emailController.text,

      passwordController.text,

      confirmPasswordController.text,

    );

    ScaffoldMessenger.of(context).showSnackBar(

      SnackBar(content: Text(response['message'] ?? 'Registration failed')),

    );

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Container(

        decoration: BoxDecoration(

          image: DecorationImage(

            image: AssetImage('images/login.jpeg'),

            fit: BoxFit.cover,

          ),

        ),

        child: Padding(

          padding: const EdgeInsets.symmetric(horizontal: 15),

          child: Center(

            child: SingleChildScrollView(

              child: Column(

                children: [

                  Text(

                    "Register",

                    style: TextStyle(

                      fontSize: 32,

                      fontWeight: FontWeight.bold,

                      color: Colors.white,

                    ),

                  ),

                  SizedBox(height: 10),

                  Textfield(

                    controller: nameController,

                    hintText: "Name",

                    label: "Name",

                    labelColor: Colors.white,

                  ),

                  const SizedBox(height: 9),

                  Textfield(

                    controller: emailController,

                    hintText: "Email",

                    label: "Email",

                    labelColor: Colors.white,

                    keyboardType: TextInputType.emailAddress,

                  ),

                  const SizedBox(height: 9),

                  Textfield(

                    controller: passwordController,

                    hintText: "Password",

                    label: "Password",

                    labelColor: Colors.white,

                    isPassword: true,

                  ),

                  const SizedBox(height: 9),

                  Textfield(

                    controller: confirmPasswordController,

                    hintText: "Confirm Password",

                    label: "Confirm Password",

                    labelColor: Colors.white,

                    isPassword: true,

                  ),

                  Row(

                    children: [

                      Checkbox(

                        value: agreeTerms,

                        onChanged: (val) => setState(() => agreeTerms = val!),

                      ),

                      Text(

                        "I agree to terms and conditions",

                        style: TextStyle(color: Colors.white),

                      ),

                    ],

                  ),

                  Button(

                    text: "Register",

                    color: Colors.white,

                    txtColor: Colors.black,

                    onPressed: register,

                  ),

                  Row(

                    mainAxisAlignment: MainAxisAlignment.center,

                    children: [

                      const Text(

                        "Already have an account?",

                        style: TextStyle(color: Colors.white, fontSize: 14),

                      ),

                      TextButton(

                        onPressed: () => Navigator.pushNamed(context, '/login'),

                        child: const Text(

                          "Log In",

                          style: TextStyle(color: Colors.red, fontSize: 14),

                        ),

                      ),

                    ],

                  ),

                ],

              ),

            ),

          ),

        ),

      ),

    );

  }

}

#### Home Page

#### Recipe Page

#### Post Creation Page

#### Pantry Page

#### Profile Page

### Sample Code for The Automation Script

## Appendix D: Designs

### Gantt Chart

### Work Breakdown Structure

### Algorithms & Flowcharts

### 3d Modeling

### Hardware Architecture

### Data Flow Diagrams

### Use Case

### Wireframe

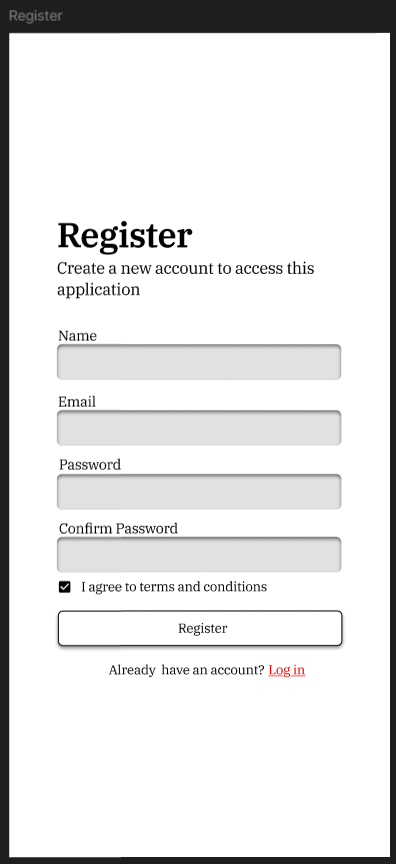


Figure : Register Page Wireframe

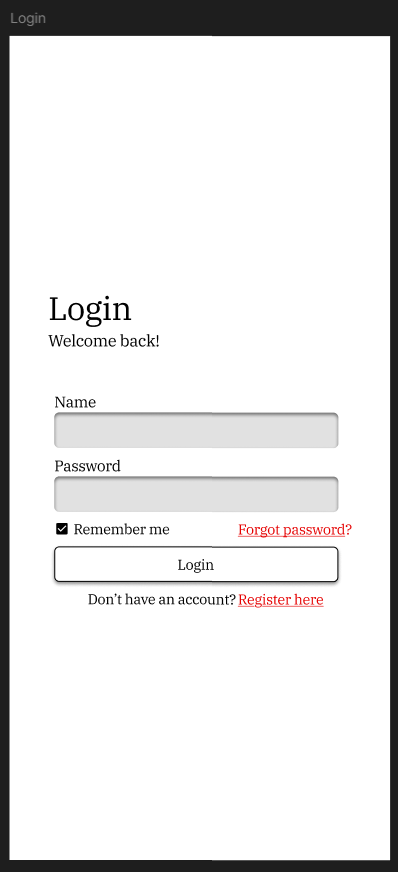


Figure : Login Page Wireframe

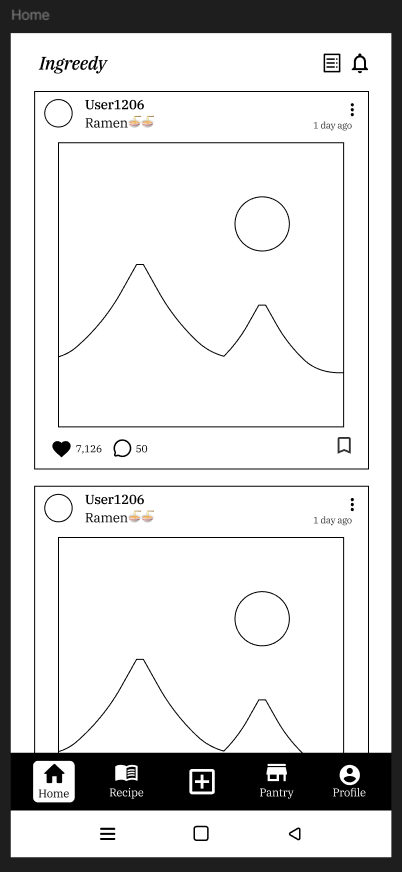


Figure : Home Page Wireframe

## Appendix E: Screenshots of The System

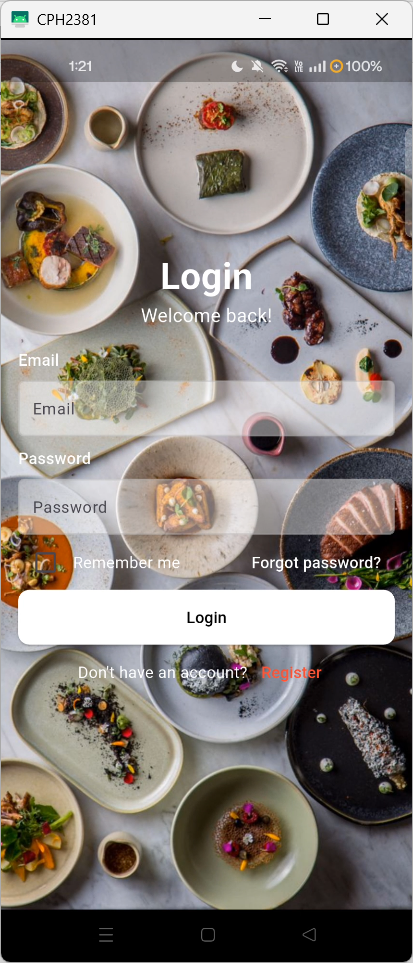


Figure : Login page

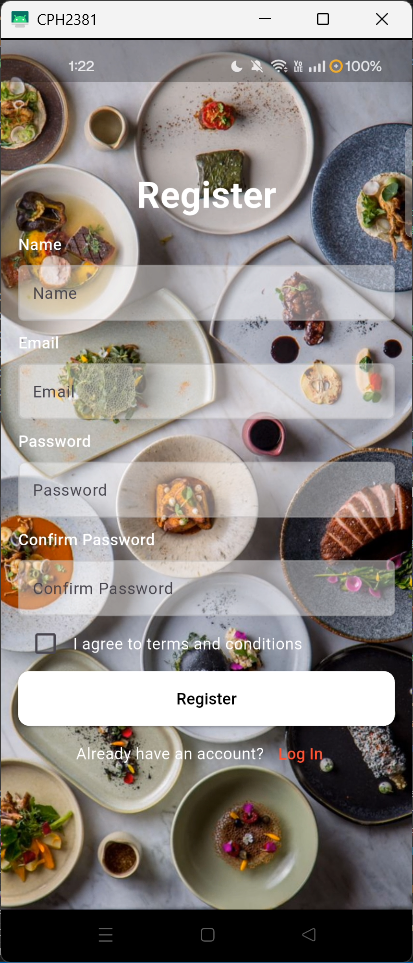


Figure : Register page

A screenshot of a social media post

AI-generated content may be incorrect.

Figure : Home Screen

## Appendix F: User Feedback

### User Feedback Form

### Sample of Filled User Feedback Forms

## Appendix G: Future Work

### Readings for Future Work