Bilkent University
Department of Computer Engineering

Senior Design Project

T2420

On the Shelf

Analysis and Requirement Report

Team Members

22003354, Gülbera Tekin, gulbera.tekin@ug.bilkent.edu.tr
22103295, Defne Gürbüz, defne.gurbuz@ug.bilkent.edu.tr
22101848, İrem Hafızoğlu, irem.hafizoglu@ug.bilkent.edu.tr
22003049, Emirhan Büyükkonuklu, e.buyukkonuklu@ug.bilkent.edu.tr
22103772, Ümmügülsüm Sümer, ummugulsum.sumer@ug.bilkent.edu.tr

Supervisor

İbrahim Körpeoğlu

Course Instructors

Atakan Erdem

Mert Bıçakçı

16.12.2024

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Contents

Analysis and Requirement Report

On the Shelf

1 Introduction

In the modern, fast-paced world, the effective management of household tasks, such as grocery shopping, has become increasingly essential. As a fundamental aspect of daily life, grocery shopping poses various challenges related to planning, organization, and time management. Both families and individuals frequently encounter difficulties in maintaining kitchen inventory, planning meals, and preparing precise shopping lists tailored to their specific needs.

Despite the advent of digital applications aimed at addressing similar challenges, there remains a lack of a comprehensive, user-friendly solution that integrates these features into a cohesive and seamless experience. To address this gap, this project introduces an innovative grocery management application designed to transform the way users monitor and manage their kitchen supplies. The application utilizes a combination of receipt scanning and manual input to track inventory effectively and notify users when supplies are running low. Furthermore, it provides recipe suggestions based on available ingredients, enabling users to maximize the utility of their kitchen resources while simultaneously reducing food waste.

Additionally, the application generates intelligent shopping lists, automating the process to save users time and enhance the efficiency of their grocery shopping. Beyond these functionalities, the application incorporates a collaborative element, allowing multiple users, such as family members, to contribute by adding notes or updating the shopping list. This feature promotes transparent and collective grocery planning within households.

In summary, this project seeks to address the need for efficient and personalized household management. The application aims to simplify grocery shopping, optimize resource utilization, and minimize food waste by leveraging advanced tools and introducing collaborative features. It aspires to provide users with a balanced, effective, and socially integrated solution for managing their kitchen and grocery needs.

1.1.1 Current System

The current state of grocery management solutions reflects a growing demand for digital tools to help individuals and households streamline their kitchen operations. While various applications have been developed to address specific aspects of grocery management, such as meal planning, recipe discovery, and inventory organization, many of these tools lack the comprehensiveness and integration required to holistically solve the challenges of food waste, efficient grocery planning, and household collaboration. This creates a significant opportunity for innovation in this space.

One prominent example is *Whisk*[1], an application focused on meal planning and recipe discovery. *Whisk* utilizes AI to create personalized recipes based on user preferences and available ingredients. It also allows users to share grocery lists, adding a collaborative element to its functionality. However, despite these features, *Whisk* does not prioritize kitchen inventory management, which is critical for tracking what users already have in their pantry. This omission makes it less effective for households aiming to reduce food waste. Moreover, the lack of receipt scanning functionality in *Whisk* means users must manually input their inventory data, a process that can be time-consuming and prone to error.

Similarly, *KitchenPal*[2] is another app that focuses on pantry management and meal planning. It allows users to organize their kitchen inventory, generate shopping lists, and explore recipes tailored to available ingredients. While *KitchenPal* performs well in these areas, it lacks a shared grocery list feature, which could be highly beneficial for households managing their shopping collaboratively. Additionally, like *Whisk*, *KitchenPal* does not support receipt scanning, requiring manual data entry for inventory updates. This lack of automation reduces its efficiency and overall user-friendliness.

Paprika[3], on the other hand, is primarily a recipe manager and meal planner. It While *Paprika* excels in organizing recipes, its pantry management features are basic and lack the depth needed for detailed tracking of quantities. Unlike *On the Shelf*, *Paprika* does not analyze a user's current inventory to suggest recipes, relying instead on users manually selecting recipes from their library. Furthermore, *Paprika* does not include receipt scanning enables users to save recipes from websites, create meal plans, and generate shopping lists. functionality or collaborative features for shared shopping lists, which limits its utility as a comprehensive grocery management solution.

What's Left[4] is an app that shares several features with *On the Shelf*, such as receipt scanning for automatic inventory updates, digital receipt storage, and account sharing for collaborative grocery management. While these features are practical for tracking grocery spending and inventory, *What's Left* lacks advanced personalization options like dietary preference management and allergen tracking. These omissions limit its ability to provide tailored meal suggestions for households with specific needs. *What's Left* also falls short in integrating sustainability-focused features, which are increasingly important to environmentally conscious users.

While these apps address certain aspects of grocery and kitchen management, none offer a unified platform that seamlessly integrates inventory tracking, automated updates, meal suggestions, and collaborative shopping list functionalities. Existing solutions either emphasize meal planning at the expense of inventory management or provide basic pantry organization tools without addressing the broader goals of efficiency and sustainability.

The *On the Shelf* app is designed to overcome these limitations by providing a comprehensive solution that combines the strengths of these applications while addressing their shortcomings. By integrating advanced features like receipt scanning for automated inventory updates, personalized meal suggestions based on dietary preferences, and shared shopping list functionality. Additionally, its emphasis on sustainability, through features designed to reduce food waste and optimize consumption habits, sets it apart from competitors. This positions *On the Shelf* as a groundbreaking tool that not only meets the needs of modern households but also promotes environmentally responsible practices.

2 Proposed System

2.1 Overview

On the Shelf is an extensive grocery management application designed to enable easier household inventory tracking, encourage consumption with reduced waste, and provide efficient shopping practices. Using smart data handling methods and easy-to-use interfaces, this system aims to solve everyday problems people face when managing their kitchen supplies and grocery shopping.

Our application offers functionalities such as inventory tracking, consumption prediction, and smart shopping list generation. The app also offers innovative recipe recommendations based on available inventory, ensuring that users can maximize the use of their household supplies while reducing food waste. Additionally, the system supports the creation of households, enabling collaborative grocery management among household members.

On the Shelf focuses on being easy to use, dependable, and fast while meeting standards for accessibility and security. We tried to ensure that the system remains robust and adaptable over time with non-functional requirements such as maintainability and efficiency. To meet different user needs, the system includes simple options like manually adding products as well as more advanced features like scanning receipts to add items automatically. This design ensures a smooth experience for users of all technical skill levels. Furthermore, the system is designed with scalability and adaptability in mind, addressing potential risks such as OCR inaccuracies and data privacy concerns.

To ensure smooth teamwork, we use tools like Jira for task management, WhatsApp and Zoom for communication, and GitHub for version control. These tools keep the team connected and provide real-time updates on progress. Regular team meetings help us review progress, tackle challenges, and stay aligned on priorities. These meetings encourage open communication and brainstorming. We follow an Agile development process, dividing the work into sprints. Each sprint focuses on specific features, followed by feedback and adjustments. This flexible approach helps us adapt to changes and improve the project over time. Conflicts are resolved through discussions within the team, seeking advice from our Course Instructor, and consulting our Innovation Expert to ensure decisions meet the project goals.

2.2 Functional Requirements

Begin with, the users shall use *On the Shelf* on their Android or iOS mobile devices.

2.2.1 Product Tracking

- The system shall allow users to add/edit/delete products from their inventory manually.
- The system shall allow users to scan or upload receipt images.
- The system shall use OCR to extract products from receipts.

- The system shall verify extracted data for accuracy and allow users to correct errors before adding to the inventory.
- The system shall filter the products in the inventory.

2.2.2 Consumption Prediction

- The system shall record product consumption history, such as quantity and frequency.
- The system shall predict when a product will likely run out based on consumption history.
- The system shall log and report waste due to expired or unused products.

2.2.3 Shopping List Creation

- The system shall automatically generate a shopping list for products predicted to run out soon.
- The system shall allow users to add, remove, or edit items in the shopping list.

2.2.4 Recipe Recommendations

- The system shall suggest recipes based on the products available in the user's inventory.
- The system shall allow users to input their allergies and diet preferences.
- The system shall recommend meals according to their allergies and diet preferences.

2.2.5 User Interface and Usability

- The system shall allow users to access features like inventory, shopping lists, and recipes.

2.2.6 Family Account

- The system shall share the same shopping list between family accounts.
- The system shall allow all family members to add products to the list.

2.2.7 Additional Features

- The system shall send notifications for low-stock alerts and suggested shopping list updates.
- The system shall synchronize data for the same user account across multiple devices.

2.3 Non-functional Requirements

2.3.1 Usability

- The system shall provide a clear and understandable interface suitable for all users.
- Features shall be intuitive, requiring no prior technical knowledge to use effectively.

- The system shall be optimized for mobile devices, ensuring seamless operation on both iOS and Android.

2.3.2 Reliability

- The system shall be available 99% of the time, minimizing the downtime.
- The system shall handle failures delicately, providing meaningful error messages and logging issues for debugging.
- The system shall ensure that no data is lost or corrupted during operations such as synchronization or updates.

2.3.3 Performance

- The system shall respond to simple user actions, such as adding products, generating shopping lists, etc., within 2 seconds under normal circumstances.
- The system shall display the user's inventory in under 3 seconds.

2.3.4 Maintainability

- The system shall be implemented using modular architecture to enable future updates and enhancements.
- All code shall be well-documented and clearly written to allow developers to understand and maintain the system efficiently.
- Regular updates shall be provided to address security vulnerabilities and improve functionality.

2.3.5 Accessibility

- The system shall be accessible for both iOS and Android users.
- The system shall consist of soft colors and stay away from bright notifications.
- The system shall recommend meals according to allergies and diet preferences.

2.3.6 Security

- The system shall store the sensitive information in an encrypted format.

2.3.7 Efficiency

- The system shall use device resources efficiently, consuming minimal battery, CPU, and memory on mobile devices.

2.4 Pseudo Requirements

- The app should be developed with Dart/Flutter.
- The app should use Firebase and Amazon Web Services for the database.
- The app should support the Turkish language.
- The app should be available on iOS and Android platforms.
- GitHub should be used for collaborative work and version control.
- The app should be free for all users. No subscription model will be used.

2.5 System Models

2.5.1 Scenarios

Adding Products via Receipt Scanning

- **Use-case Name**: Add Products via Receipt Scanning
- **Actor**: Family Member, System (OCR module, Python FuzzyWuzzy Library)
- **Entry Condition**: The fm has a shopping receipt and wants to update their inventory.
- **Exit Condition**: Kitchen-related products are successfully added to the fm's inventory, with accurate details.

Flow of Events:

1. The fm opens the app and selects the "Scan Receipt" option.
2. The fm takes a photo of the shopping receipt or uploads an image.
3. The system uses OCR to extract product details (e.g., product names and quantities) from the receipt.
4. The system cross-matches the extracted product names with a kitchen product dataset using the FuzzyWuzzy library to identify relevant kitchen-related items.
5. The system filters out non-kitchen products from the receipt.
6. The system displays the filtered, kitchen-related product data to the user for confirmation.
7. Fm reviews the data and makes corrections if necessary.
8. Fm confirms the details, and the system updates the inventory.

Alternative Flows:

- **A1**:
    - The OCR fails to extract data due to poor image quality.
    - The system prompts the user to retake the photo or upload a clearer image.
- **A2**:
    - The system fails to classify a kitchen product due to low matching confidence.
    - The system flags such items for user review, allowing manual classification.
- **A3**:
    - The receipt includes a product already in the inventory.
    - The system updates the quantity instead of adding a duplicate entry.
- **A4**:
    - A non-kitchen product has a high similarity score with a kitchen product in the dataset.
    - The system highlights the item for user review before confirming its addition to the inventory.

Generating Smart Shopping Lists

- **Use-case Name**: Generate Smart Shopping List
- **Actor**: Family Member, System
- **Entry Condition**: The fm's inventory contains items that are either low in stock or predicted to run out soon.
- **Exit Condition**: A dynamically updated shopping list is available for the user.

Flow of Events:

1. The system monitors the user's inventory for low-stock items or items with predicted depletion based on consumption history.
2. The system automatically adds these items to a "Suggested Shopping List."
3. The user opens the app and views the shopping list under the "Smart Shopping List" section.
4. The user reviews the list and:
    - Removes items they don't wish to purchase.
    - Adds additional items manually if needed.

5. The user confirms the shopping list.

Alternative Flows:

- **A1**:
    - The system incorrectly predicts the depletion of a product.
    - The user removes the product from the shopping list and flags it as incorrect.
- **A2**:
    - A family member adds an item via collaborative requests.
    - The system appends the request with the contributor's name to the shopping list.

Manually Adding Products to Inventory

- **Use-case Name**: Manually Add Products to Inventory
- **Actor**: Family Member, System
- **Entry Condition**: Fm has registered to the system and needs to set up their kitchen inventory by manually adding their existing products.
- **Exit Condition**: user's inventory contains all manually added kitchen products with accurate details.

Flow of Events:

1. After registration, the system prompts the user to set up their kitchen inventory.
2. User selects the "Add Product" option in the app.
3. The system displays input fields for the product name, quantity, category (optional), and expiration date (if applicable) additional fields may be required.
4. User enters the details for each product and confirms.
5. The system updates the inventory with the new product details.
6. User repeats steps 2–5 until all kitchen products are added.
7. The system displays a summary of the inventory to the user for review.

Alternative Flows:

- **A1**:
    - User attempts to add a product already in the inventory.

    - The system prompts the user to update the quantity instead of adding a duplicate entry.
- **A2**:
    - The user mistakenly enters incomplete or invalid details (e.g., non-numeric quantity).
    - The system highlights the errors and requests corrections.

Editing and Deleting Products from Inventory

- **Use-case Name**: Edit and Delete Products in Inventory
- **Actor**: Family Member, System
- **Entry Condition**: User has an existing inventory with products and wishes to update or remove items.
- **Exit Condition**: The inventory is updated with accurate product details, or unwanted products are removed.

Flow of Events for Editing:

1. The user navigates to the inventory section of the app.
2. The user selects a product they want to edit.
3. The system displays the product details, including name, quantity, expiration date, and category.
4. The user updates one or more fields and confirms the changes.
5. The system validates the new details and updates the inventory.

Flow of Events for Deleting:

1. The user navigates to the inventory section of the app.
2. The user selects a product they want to delete.
3. The system asks for confirmation to delete the product.
4. The user confirms, and the system removes the product from the inventory.

Alternative Flows:

- **A1**:
    - The user attempts to edit a product but provides invalid input (e.g., a negative quantity).

    - The system displays an error message and highlights the invalid fields for correction.
- **A2**:
    - The user edits a product to reduce the quantity to zero.
    - The system prompts the user to confirm if they want to delete the product instead of keeping it with zero quantity.

Predicting Product Depletion Based on Usage Patterns

- **Use-case Name**: Predict Product Depletion
- **Actor**: Family Member, System
- **Entry Condition**: The user's inventory contains products with recorded consumption history (e.g., quantity updates over time).
- **Exit Condition**: The system provides predictions for products with sufficient and consistent consumption data, while flagging or excluding others.

Flow of Events:

1. The user updates the inventory by marking quantities as used.
2. The system logs the update, including the date and quantity used for each product.
3. The system calculates the **average daily consumption rate** arithmetically.
4. Using the current stock and the calculated average consumption rate, the system estimates the **remaining days** for products with sufficient data.
5. The system flags products that lack sufficient or consistent consumption data, indicating "Prediction Not Available" for these items.
6. The system highlights products likely to run out soon (e.g., within 3 days) in the inventory.
7. Predictions are updated dynamically as the user logs new consumption data.

Alternative Flows:

- **A1**:
    - The system detects insufficient or irregular data for a product.
    - The system displays "Prediction Not Available" for that product, allowing the user to manually monitor or prioritize its stock to add to shopping list manually.
- **A2**:

    - The user's consumption pattern changes drastically (e.g., sudden increase or decrease).
    - The system recalculates predictions dynamically, prioritizing recent data for accuracy.
- **A3**:
    - A product with previously insufficient data gains enough usage history through recent updates.
    - The system begins providing predictions for that product.

Receiving Expiration Date Notifications

- **Use-case Name**: Expiration Date Notifications
- **Actor**: Family Member, System
- **Entry Condition**: The user has products in their inventory with possible assigned expiration dates.
- **Exit Condition**: The user is notified of products nearing expiration and can take appropriate action.

Flow of Events:

1. The system regularly checks the inventory for products with expiration dates approaching.
2. The system triggers a notification for each product nearing expiration. Notification content includes product name and expiration date.
3. The system updates the inventory status based on user actions.

Suggesting Recipes Based on Inventory

- **Use-case Name**: Suggest Recipes Based on Inventory
- **Actor**: Family Member, System
- **Entry Condition**: The user has an inventory of products, and the system utilizes a recipe recommendation model to generate suggestions.
- **Exit Condition**: The user views personalized recipe suggestions based on available ingredients in their inventory.

Flow of Events:

1. The user navigates to the "Recipes" section of the app.
2. The system retrieves the user's inventory and inputs the data into the recipe recommendation model.
3. The recommendation model processes the inventory data and generates recipe suggestions, prioritizing:
    - Recipes that use the maximum number of available ingredients.
    - Recipes that align with user preferences (e.g., allergies or dietary restrictions).
4. The system displays the recommended recipes with the following details:
    - Recipe name.
    - Required ingredients (distinguishing between available and missing ones).
    - Preparation steps.
5. The user selects a recipe to view detailed instructions or adds missing ingredients to their shopping list.

Alternative Flows:

- **A1**:
    - The system generates no complete recipe matches.
    - The model suggests partial matches with missing ingredients highlighted.
- **A2**:
    - The user specifies dietary preferences or allergies in their profile.
    - The system filters out recipes that conflict with these preferences.

Filtering Inventory for Specific Needs

- **Use-case Name**: Filter Inventory for Specific Needs
- **Actor**: Family Member, System
- **Entry Condition**: The user has products in their inventory and has specified allergies or dietary preferences in their profile.
- **Exit Condition**: The user views a filtered list of products in their inventory that match their specified needs.

Flow of Events:

1. The user accesses the inventory section of the app.
2. The system retrieves the user's specified allergies or dietary preferences (e.g., "no nuts" or "vegetarian").
3. The system filters the inventory to exclude products that conflict with the user's preferences or allergies.
4. The system displays a filtered list of products that align with the user's specified needs.

Resolving Incorrect Fuzzy Matching of Products

- **Use-case Name**: Resolve Incorrect Fuzzy Matching of Products
- **Actor**: User, System
- **Entry Condition**: The system uses fuzzy matching to identify kitchen-related products from receipt data, but some matches are incorrect or ambiguous.
- **Exit Condition**: The user reviews and corrects mismatched or unrecognized products, ensuring the inventory is updated accurately.

Flow of Events:

1. The user uploads a receipt for processing.
2. The system applies fuzzy matching to compare extracted product names with a predefined kitchen product dataset.
3. The system flags products with low confidence matches or multiple potential matches for user review.
4. The user reviews the flagged items and:
    - Confirms the correct product.
    - Manually specifies the product if no match is correct.
5. The system updates the inventory based on the user's corrections.

Alternative Flows:

- **A1**:
    - The system suggests multiple potential matches for a product.
    - The user selects the most appropriate match, and the system updates the inventory accordingly.

Household Creation and Joining

- **Use-case Name:** Create and Join Household
- **Actor:** Family Member
- **Entry Condition:** A family member has logged into the app and wants to create a household or join an existing one.
- **Exit Condition:** A new household is created with an initial setup, or the family member successfully joins an existing household.

Flow of Events for Household Creation:

1. The family member selects the "Create Household" option in the app.
2. The system prompts the user to enter a household name.
3. The system validates the household name (e.g., checks for duplicates within the user's account).
4. The family member optionally adds an initial inventory during setup:
    - The system displays input fields for product names, quantities, and expiration dates.
    - The family member enters the details and confirms.
5. The system generates a unique invitation code for inviting other family members.
6. The household is created, and the family member becomes a member of the household.

Alternative Flows for Household Creation:

- **A1**:
    - The household name is invalid or already exists.
    - The system displays an error message and prompts the user to enter a unique name.

Flow of Events for Joining a Household via Invitation Link:

1. The family member receives an invitation code shared by another member of the household.
2. The family member selects the "Join Household" option in the app.
3. The system validates the invitation code.
4. Upon successful validation, the family member joins the household.

2.5.2 Use-Case Model

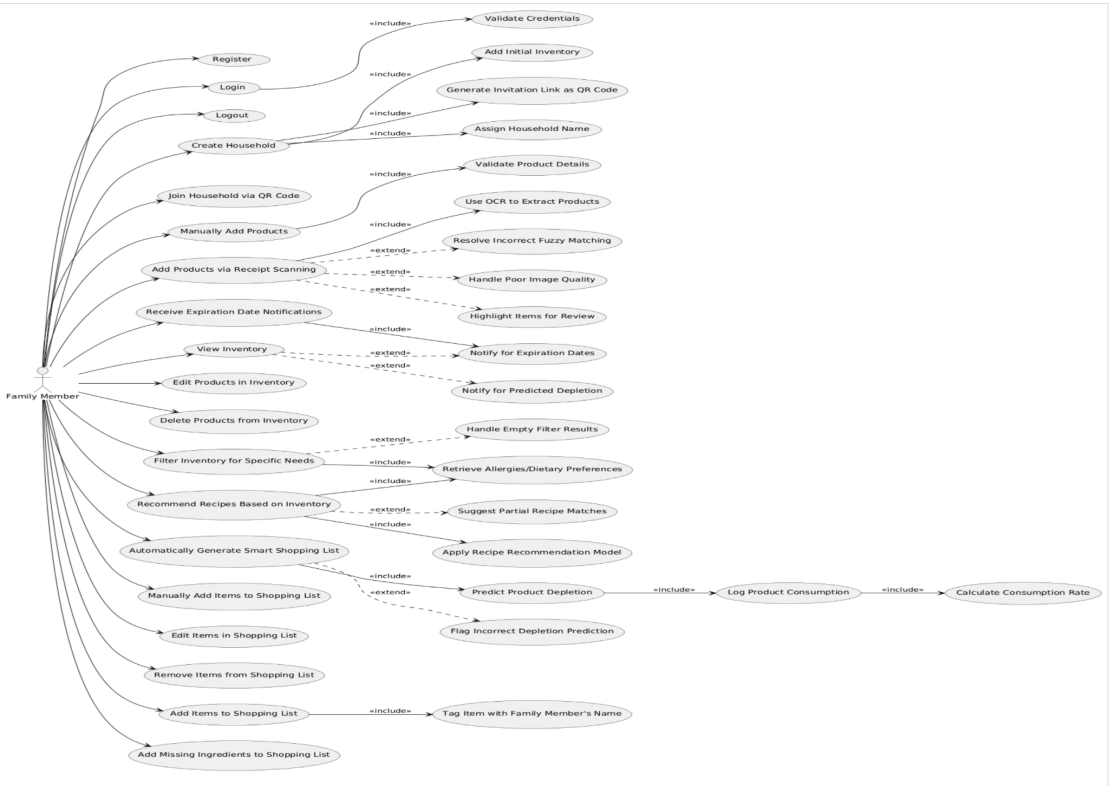

Fig.1 Use-Case Model

2.5.3 Object and Class Model

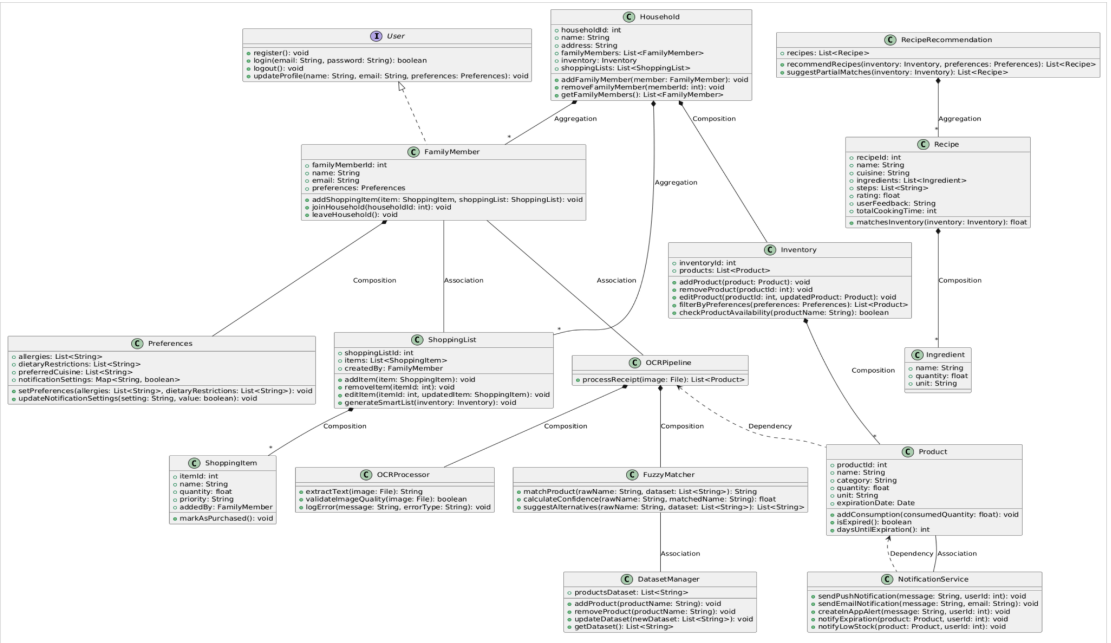

Fig.2 Object and Class Model

2.5.4 Dynamic Models

2.5.4.1 Activity Diagrams

2.5.4.1.1 Login and Register

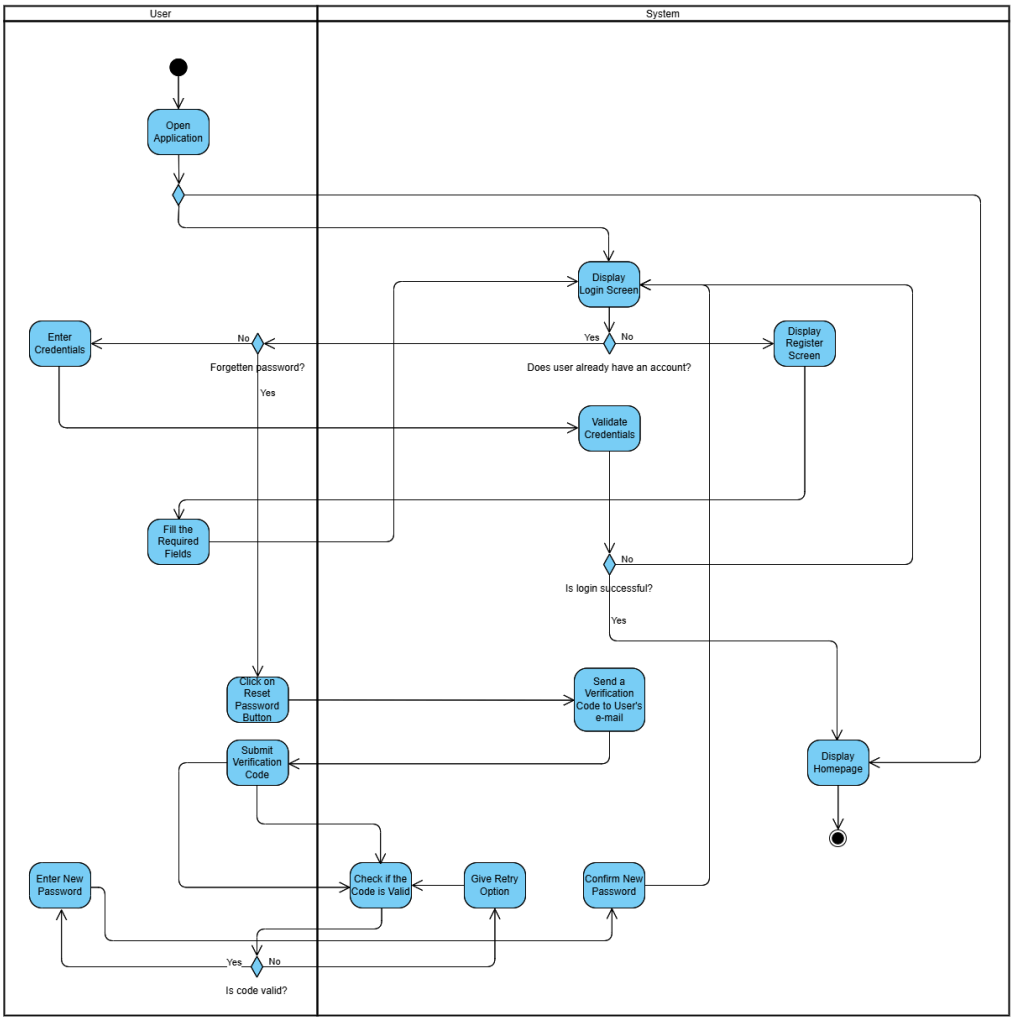

Fig.3 Activity Diagram of Login and Register

This diagram showcases the login and register activities. It consists of two main partitions, User and System. Password resetting via verification code option is also displayed. These activities end after the user is directed to the homepage.

2.5.4.1.2 Household Selection

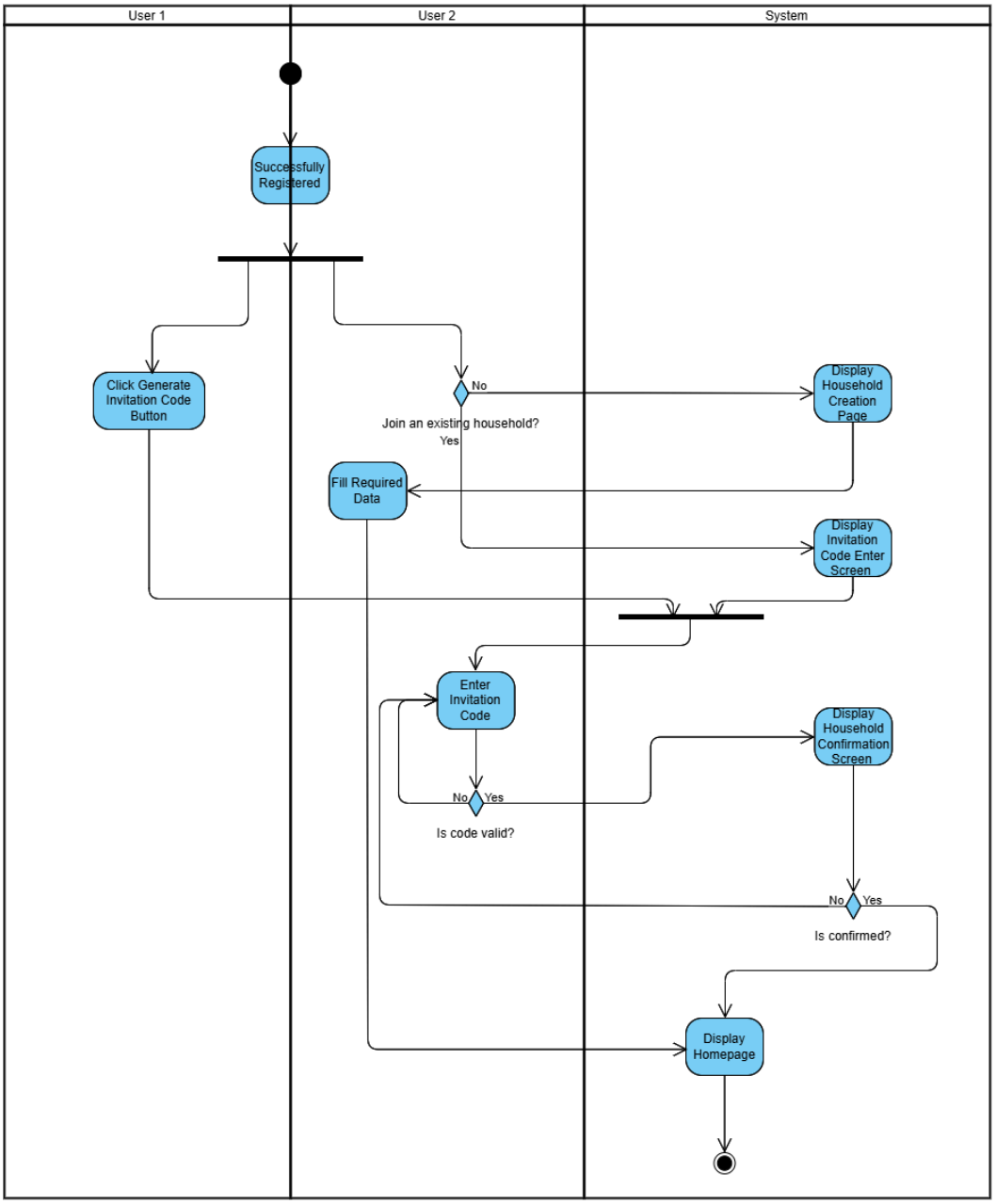

Fig.4 Activity Diagram of Household Selection

This diagram showcases household selection activity. The newly registered user (User 2) either creates a new household or joins an existing one. If the user selects to create a new household, s/he is directed to a household creation page. After the required fields are filled the process is completed. Alternatively, users can enter an invitation code generated by another member (User 1) of an existing household to join one. Both activities end after the user is directed to the homepage.

2.5.4.1.3 Registering Products

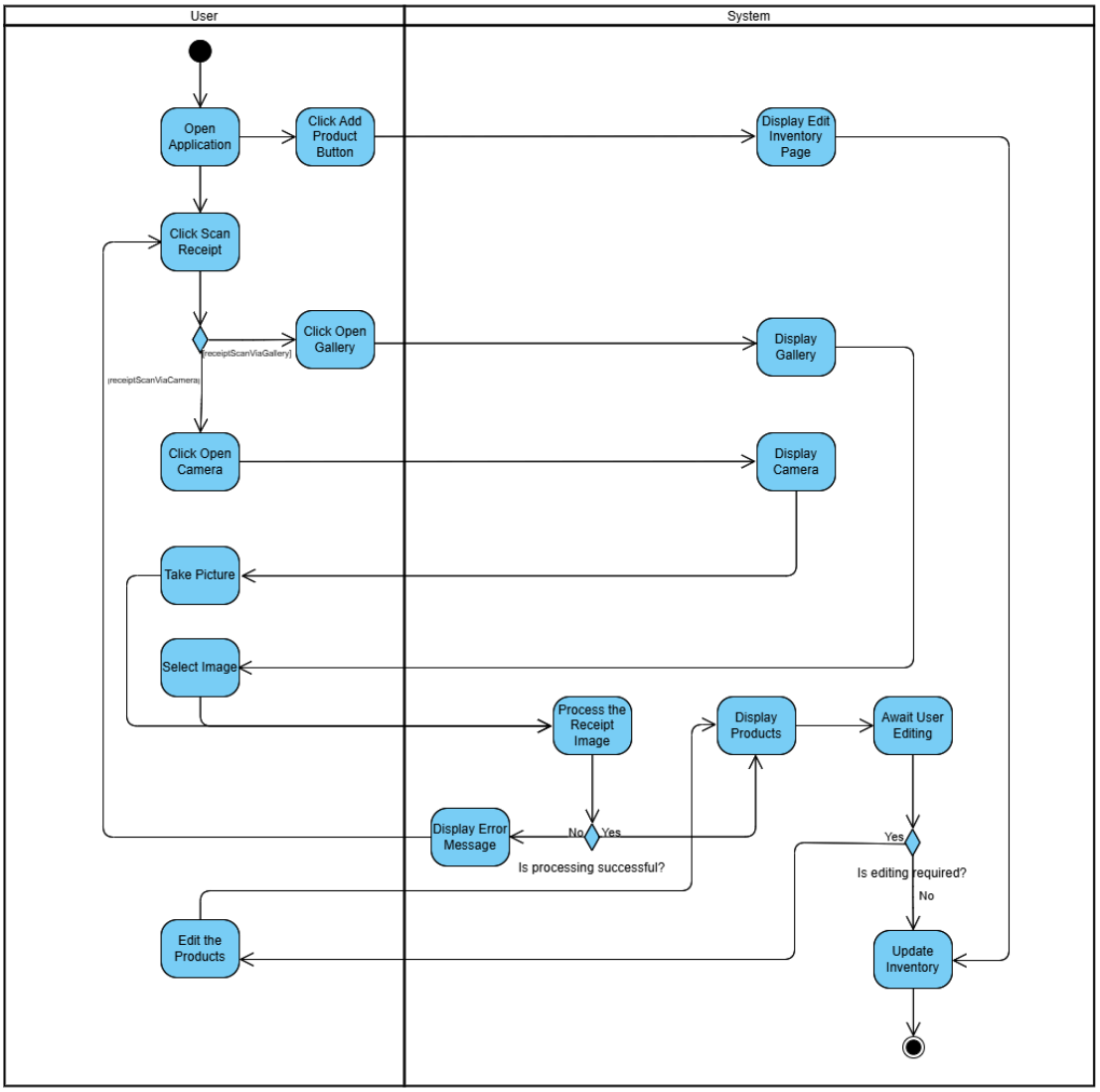

Fig.5 Activity Diagram of Registering Products

This diagram showcases product registration to the inventory. There are two main paths to be followed. First is scanning the shopping receipt. It can be done via selecting a photo from the gallery or taking a new photo. If processing of the receipt fails, the user is redirected to the page where the scan receipt button is present. Alternatively, users can select to register the products manually. After both pathways, the inventory is updated accordingly.

2.5.4.2 Sequence Diagrams

2.5.4.2.1 Recipe Recommendation

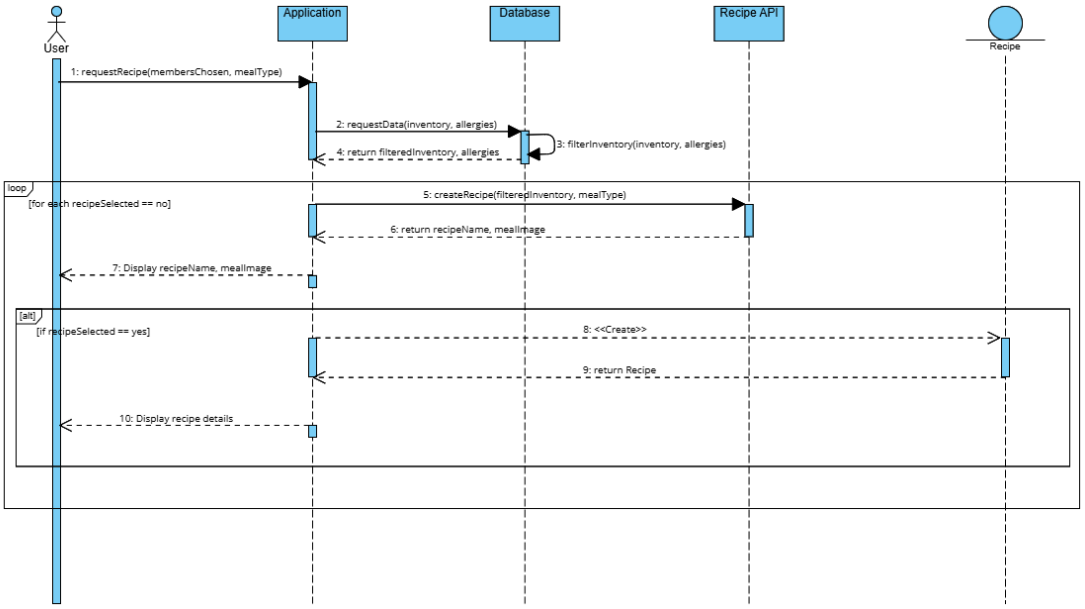

Fig.6 Sequence Diagram of Recipe Recommendation

This diagram showcases our recipe recommendation function. User selects the meal type and the user's that the meal will be prepared for and the application automatically takes their allergies into consideration by filtering the inventory accordingly. User is presented with the meal name and picture of the newly generated recipe only until s/he chooses one. Later, when recipe is selected, all details are shown.

2.5.4.2.2 Automatic Shopping List Update

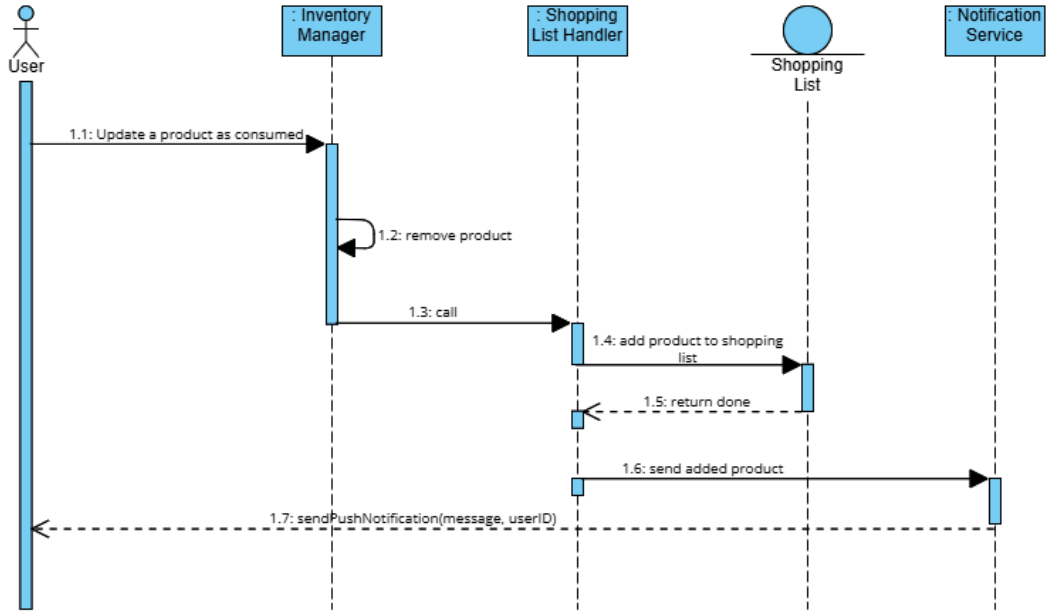

Fig.7 Sequence Diagram of Automatic Shopping List Update

This diagram showcases our automatic shopping list update function. When a product is consumed, it is removed from the inventory by the user. This deletion event triggers the shopping list handler and the product is added to the shopping list. And when the automatic update event is concluded, the product information is directed to notification service for sending related push notification to the user.

2.5.5 User Interface

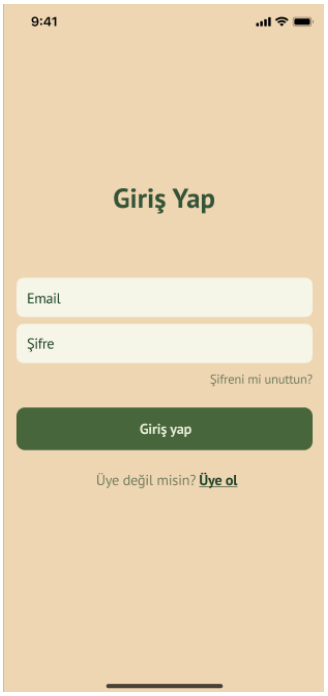

Fig.8 Login Page UI

Fig.9 Register Page UI

Once the user opens the app, the login screen will be shown. Users can choose to log in if they have an account with their credentials, or they can sign up with the button on the bottom. If they choose to sign up, the app will direct the user to the signup page.

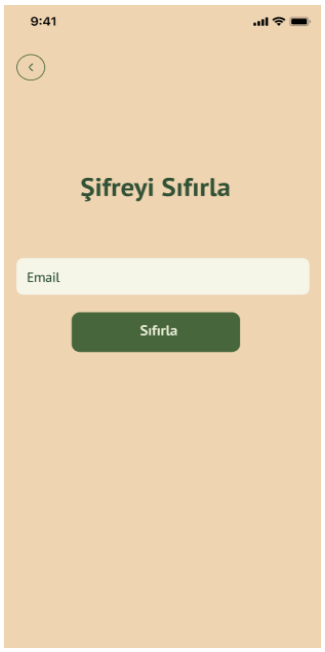

Fig.10 Reset Password UI

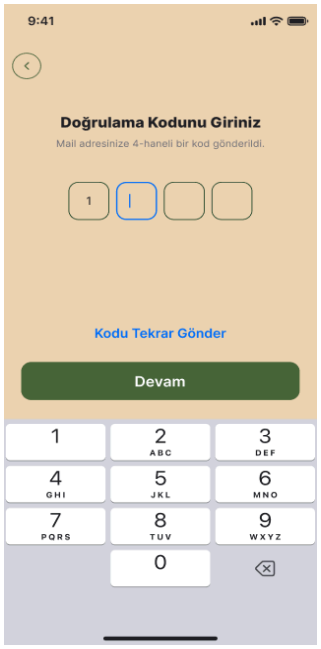

Fig.11 Verification Code UI

Fig.12 New Password UI

If the user chooses the “forget my password” option, it will lead the user to a page where they can reset their password. User should enter their mail addresses and confirm it with the code that is sent to their mail. Then, the user can enter the new password.

Fig.13 Household Account UI

Fig.14 Create a New Household UI

Once the user is signed up, they can either choose to create a new household, or they can join an existing household. If they choose to create a new household, the app expects the user to enter the new household’s name.

Fig.15 Invitation Code for Household UI

Fig.16 Join a Household UI

If the user chooses to join an existing household, the app expects the user to enter the household invitation code, which can be found in a household member profile. then a confirmation page comes up.

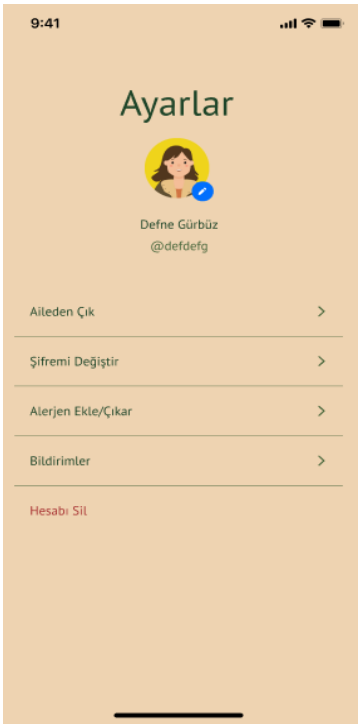

Fig.17 Settings Page UI

Users can use the settings page to exit a household, change the name, change the password, change the profile picture, edit the allergens, manage the notifications or delete the account.

Fig.18 Home Page UI

Fig.19 Receipt Scan UI

Fig.20 Inventory Categories UI

After the user logs in, they will be directed to the home page. On the home page, users can scan a receipt, ask to generate a recipe and reach their grocery list or inventory. When the user clicks on the “scan receipt” button, the camera opens. The user can either take a photo of the receipt or click on the button on the bottom left of the screen to choose a photo from the gallery. If they choose to review their inventory the categories will be shown.

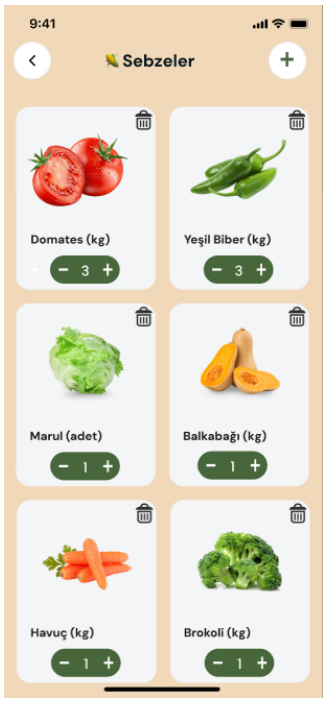

Fig.21 Inventory Page UI

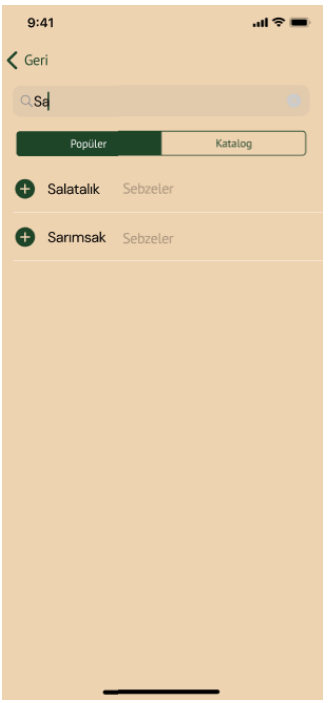

Fig.22 Search for Product UI

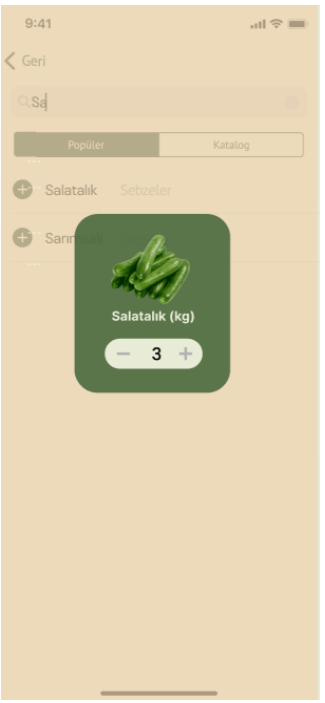

Fig.23 Add Product Manually UI

The user can review the products in their inventory under each category. The user can search for the products and edit the products manually if they like.

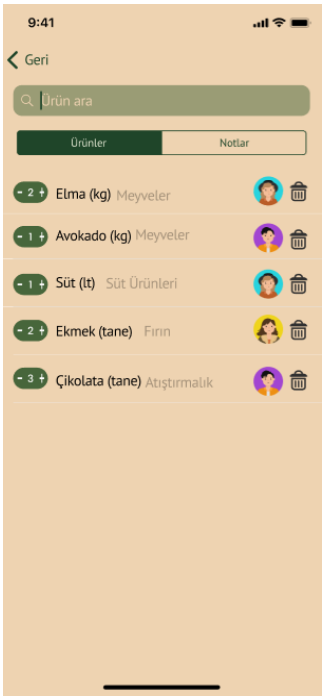

Fig.24 Shopping List UI

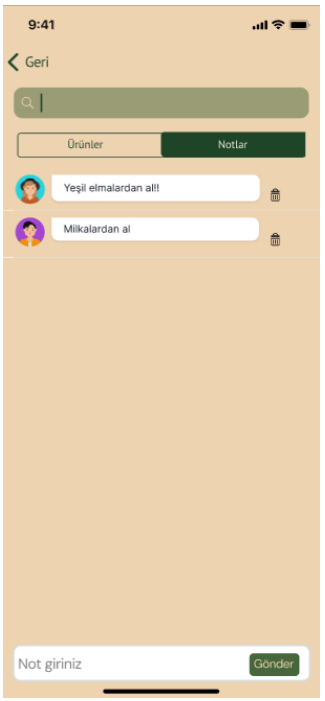

Fig.25 Shopping Notes UI

If the user clicks on the “view shopping list“ button, they get directed to a screen where they can view each product on their shopping list, which household member added it and they can also reach the notes family members can add.

Fig.26 Generate Recipe UI

Fig.27 Generated Recipe UI

Fig.28 Recipe of Meal UI

If the user clicks on the “generate recipe” button they are directed to a screen where they can choose the family members they will prepare the meal for and which type of meal they want. Once the recipe is generated based on the products in the inventory, the user can either choose the generated recipe or ask for a new recipe. When the recipe is chosen, they can view the recipe.

3 Other Analysis Elements

In this section, various factors affecting On the Shelf will be discussed.

3.1 Consideration of Various Factors in Engineering Design

3.1.1 Constraints

In addition to non-functional requirements, the following can be listed as constraints:

Aesthetics

- The user interface shall prioritize simplicity and visual appeal, ensuring a clutter-free and intuitive design.
- Colors and icons must adhere to universal design principles to make the app attractive and easily understandable.

Codes

- The app shall comply with coding standards for mobile application development for secure coding practices.

Constructability

- The app shall be deployable on both iOS and Android platforms using a single codebase, Flutter.

Ergonomics

- The app shall require minimal user input, utilizing automation features like OCR for receipt scanning to reduce manual effort.
- To prevent user's overhead, notifications must be simple.

Extensibility

- The architecture shall allow for easy addition of new features.

Functionality

- The app must deliver the features outlined in the functional requirements, such as inventory management, and recipe recommendations.

Interoperability

- The app shall support synchronization across multiple devices for the same user account.

Legal Considerations

- The app shall comply with data privacy laws like KVKK, ensuring user information is protected.

Marketability

- The app shall address the key pain points of users, such as reducing food waste and simplifying grocery management, to attract a broad audience.

Policy

- The app shall align with policies promoting sustainability and responsible consumption.

Regulations

- The app shall adhere to regional app store guidelines for deployment on Google Play Store and Apple App Store.

Schedule

- Development shall be completed within the defined timeline, with key milestones for testing, feedback, and final deployment.

Standards

- The app shall adhere to industry standards for mobile apps.

Sustainability

- The app shall promote sustainable practices by encouraging reduced food waste and better resource utilization.
- Energy-efficient design practices shall be followed to minimize resource usage during operation.

Minimum Hardware and Software Specifications

iOS

- Minimum OS Version: iOS 13 or later (to ensure compatibility with Core ML and Vision frameworks used for OCR).
- Hardware: Devices with at least an A11 Bionic chip (iPhone 8 or later) for efficient OCR processing.

Android

- Minimum OS Version: Android 8.0 or later.
- Hardware: Devices with at least 2GB of RAM.

Global Factors

- Cross-platform support: Making the app available on both Android and iOS widens its reach globally.

Cultural Factors

- Recipe Recommendations: The app shall consider cultural differences in dietary preferences and food habits, providing region-specific recipes.
- Collaborative Features: Sub-accounts can enhance collaboration in cultures where families share grocery responsibilities.

Social Factors

- Reducing Food Waste: The app addresses a significant social issue by minimizing food waste and promoting responsible consumption.
- Inclusivity: Features like accessibility options and dietary customization cater to diverse user groups.

Environmental Factors

- Sustainability: Encouraging users to reduce waste directly contributes to environmental well being.

- Energy Efficiency: Designing the app to be resource-light ensures lower energy consumption during operation.

Table 1: Factors that can affect analysis and design.

Constraint	Effect level (0 - Lowest / 10 - Highest)	Effect
Aesthetics	7	Enhances user satisfaction and retention.
Codes	8	Supports secure and stable app performance.
Constructability	7	Ensures feasibility for cross-platform development.
Ergonomics	8	Reduces user effort and improves experience.
Extensibility	10	Enables future feature additions.
Functionality	10	Provides essential features for grocery management.
Interoperability	10	Allows integration with multiple devices.
Legal Considerations	9	Ensures compliance with data privacy laws.
Marketability	9	Attracts users by addressing key needs.
Policy	7	Aligns with sustainability goals.
Regulations	8	Adheres to app store and regional standards.
Schedule	6	Ensures timely project delivery.
Standards	9	Aligns with accessibility and usability requirements.
Sustainability	9	Promotes eco-friendly practices.
Minimum Hardware and Software Specifications	7	Maximizes compatibility across devices.
Global factors	9	Ensures relevance in diverse regions.

Cultural factors	8	Adapts to different dietary and cultural needs.
Social factors	7	Promotes collaboration and addresses social issues.
Environmental factors	9	Contributes to reducing food waste.

3.1.2 Standards

- UML 2.5.1 for modeling
- IEEE 830 for requirements documentation
- IEEE for referencing style

3.2 Risks and Alternatives

3.2.1 OCR Inaccuracy

One of the primary risks is the potential inaccuracy of the OCR technology used to extract product information from receipts. Misread items or quantities could result in incorrect inventory records, leading to poor recommendations and shopping lists. This could frustrate users and lead to reduced trust in the app.

3.2.2 Data Privacy Concerns

Since the app will collect sensitive information about users' shopping habits, dietary preferences, and potentially personal details, there is a risk of data privacy violations. Users may be concerned about how their data is handled, stored, and shared, especially in light of increasingly stringent data protection laws.

3.2.3 High Development and Maintenance Costs

Developing and maintaining the app with features like OCR could incur significant costs. This might strain resources, especially during the initial development phase, and could impact the long-term financial viability of the project.

3.2.4 Limited Adoption and Market Competition

The grocery management and recipe recommendation app market is competitive, with many established players already providing similar functionalities. The app might struggle to differentiate itself sufficiently to attract a large user base, limiting its adoption and growth potential.

3.2.5 System Downtime and Bugs

The app may face periods of downtime or encounter bugs, especially as new features are added, or updates are made. These technical issues could disrupt the user experience, damage the app's reputation, and lead to user dissatisfaction if not addressed promptly.

3.2.6 Cultural Misalignment in Recipe Suggestions

Recipe recommendations may not suit the cultural or dietary preferences of all users, particularly in a global market. Users from different cultural backgrounds might find the app's suggestions irrelevant or unsuitable, leading to lower user engagement and dissatisfaction.

3.2.7 Dependency on Internet Connectivity

Since the app requires internet connectivity for certain features like syncing shopping lists, scanning receipts, and receiving recipe suggestions, users with limited or unstable internet access may face difficulties using the app. This could reduce the app's usability in areas with poor connectivity.

3.2.8 Regulatory and Legal Compliance

The app will need to comply with various regulations, including data protection laws, consumer protection laws, and potentially food safety regulations. Failing to comply with these legal requirements could result in fines, legal action, or forced shutdowns, affecting the app's credibility and long-term viability.

Table 2: Risks

Risk	Likelihood (0 - Lowest / 5- Highest)	Effect on the project	B Plan Summary
OCR Inaccuracy	3	High risk of incorrect inventory records, leading to poor recommendations and user dissatisfaction.	Implement manual data verification to ensure accuracy. Continuously improve the OCR algorithm.

Data Privacy Concerns	1	Users may be reluctant to use the app due to concerns over data handling, potentially leading to legal issues.	Adhere to data privacy regulations. Clearly communicate data policies and offer control.
High Costs	3	Could strain resources, delay development, or limit long-term scalability.	Prioritize core features first. Use open-source frameworks to reduce costs and adopt a phased development approach.
Limited Adoption and Market Competition	4	Difficulty in differentiating the app in a crowded market could limit user acquisition and retention.	Focus on unique features (e.g., waste tracking, collaborative management) to create a distinct value proposition.
System Downtime and Bugs	2	Technical issues could disrupt user experience, leading to negative reviews and decreased user retention.	Ensure robust testing and continuous maintenance. Establish a responsive support system for bug fixes.
Cultural Misalignments	2	Users may find recipe suggestions irrelevant or unsuitable, leading to reduced engagement.	Allow users to customize preferences and exclude certain recipes. Incorporate user feedback for improvement.
Dependency on Internet	4	Limited functionality in areas with poor connectivity could hinder user experience, affecting app adoption.	Enable offline functionality for essential features. Sync data when the user has internet access.
Regulatory and Legal Compliance	2	Non-compliance with regulations could result in legal penalties, data breaches, and loss of user trust.	Ensure full legal compliance from the start, with periodic audits to stay aligned with evolving laws.

3.3 Project Plan

Table 3: Project Plan

	Work Package Name	Leader	Contribution
WP1	Requirement Analysis	Gülbera Tekin	All members
WP2	Market and Competitor Research	Defne Gürbüz	All members
WP3	Funding and Budget Management	Ümmügülsüm Sümer	All members
WP4	CS491 Reports	İrem Hafızoğlu	All members
WP5	CS491 Demo	Ümmügülsüm Sümer	All members
WP6	Client Side Development	Defne Gürbüz	Irem Hafızoğlu Gülbera Tekin Ümmügülsüm Sümer
WP7	Server Side Development	Emirhan Büyükkonuklu	Emirhan Büyükkonuklu Defne Gürbüz
WP8	CS492 Reports	Emirhan Büyükkonuklu, Gülbera Tekin	All members
WP9	CS492 Demo/CS Fair	İrem Hafızoğlu	All members

Work Packages

WP 1: Requirement Analysis			
Start date: 16th of September **End date:** 30th of October			
Leader:	Gülbera Tekin	**Members involved:**	All members
Objectives: The objective of this work package is to gather and document the functional and non-functional requirements of the project. This includes defining user needs, identifying system constraints, and drafting use case scenarios. The analysis will serve as			

the foundation for the system's design and development phases by ensuring alignment with user expectations and project goals.

Tasks:

Task 1.1: Stakeholder Identification and Interviews: Conduct meetings and interviews with stakeholders to understand their needs and expectations from the system. Identify key features and prioritize them based on user feedback.

Task 1.2: Documentation of Functional Requirements: Write detailed functional requirements, specifying the expected system behavior and features. Include use cases, flow diagrams, and descriptions for all core functionalities.

Task 1.3: Documentation of Non-Functional Requirements: Identify system constraints, such as performance, scalability, and security requirements. Specify any operational constraints, such as platform compatibility.

Task 1.4: Use Case and Class Diagram Preparation: Develop use case diagrams to visualize system functionality and interactions.Create a class diagram to outline the structural design of the system.

Deliverables

D1.1: Functional and Non-Functional Requirements which are present in reports.

D1.2: Use Case Diagrams for core system functionalities.

D1.3: Class Diagram for system structure.

D1.4: Stakeholder Interview Report summarizing key insights.

WP 2: Market and Competitor Research			
Start date: 16th of September **End date:** 30th of October			
Leader:	Defne Gürbüz	**Members involved:**	All members

Objectives: The objective of this work package is to conduct an analysis of the market and existing competitor applications. This includes identifying trends, key features, pricing models, and gaps in the current solutions. The insights gathered will help in designing a system that meets user needs while offering competitive advantages.

Tasks:

Task 2.1: Market Trend Analysis: Research the current market for grocery management and recipe recommendation systems. Identify emerging trends, user expectations, and common pain points.

Task 2.2: Competitor Feature Analysis: Analyze competitor applications to identify their core features, strengths, and weaknesses. Focus on apps with similar functionalities, such as receipt scanning, inventory management, and collaborative shopping lists. **Task 2.3: Pricing and Monetization Study:** Examine competitor pricing models, such as subscription plans, freemium features, or one-time purchases. **Task 2.4: User Feedback Collection:** Gather user reviews and feedback from app stores or surveys to understand the strengths and limitations of existing solutions. Identify features that users find valuable or problematic.
Deliverables **D2.1**: Market Analysis insights. **D2.2**: Competitor Feature Comparison insights. **D2.3**: User Feedback Survey Summary.

WP 3: Funding and Budget Management			
Start date: 15th of October **End Date:** 15th of November			
Leader:	Ümmügülsüm Sümer	**Members involved:**	All members
Objectives: The objective of this work package is to plan and manage the project's financial resources effectively. This includes identifying funding sources, allocating budgets for necessary tools and services, and ensuring that expenses stay within the allocated budget.			
Tasks: **Task 3.1: Identify Funding Sources:** Explore potential funding opportunities, such as university grants, personal contributions, or sponsorships from industry.			
Deliverables no deliverables needed			
WP 4: CS 491 Reports			
Start date: 15th of September **End date:** 15th of December			
Leader:	İrem Hafızoğlu	**Members involved:**	All members
Objectives:			

The objective of this work package is to prepare and submit all required reports for the CS491 course, including the Project Information Form, Project Specification Document**,** and the Analysis and Requirements Report.
Tasks: **Task 4.1: Prepare Project Information Form:** Document the project name, high-level description, supervisor, innovation expert, and URL of the project's webpage **Task 4.2: Prepare Project Specification Document:** Write a brief description of the project, including requirements and constraints. Identify and document relevant engineering standards and ethical responsibilities. Include high-level system architecture and components of the proposed solution. **Task 4.3: Prepare Analysis and Requirements Report:** Conduct a detailed analysis of the system requirements. Develop system models, including use case diagrams, class diagrams, and user interface mockups. Address considerations for public health, safety, global, and cultural factors in engineering design. Plan for teamwork, risks, alternatives, and learning strategies.
Deliverables **D4.1**: Project Information Form. **D4.2**: Project Specification Document. **D4.3**: Analysis and Requirements Report.

WP 5: CS491 Demo			
Start date: 6th of December **End date:** 25th of December			
Leader:	Ümmügülsüm Sümer	**Members involved:**	All members
Objectives: The objective of this work package is to deliver a polished and professional demo and presentation of the project's progress at the end of the first semester. The team will present the project's background, objectives, architecture, current status, and achievements, followed by a demo of the work completed so far.			
Tasks: **Task 5.1: Preparation of Presentation Slides:** Create a concise introduction slide to explain the project's objectives and innovation. Prepare a project progress slide with a			

status dashboard to highlight completed and pending tasks. Include a graphical representation of the designed system architecture.

Task 5.2: Team Rehearsal for Presentation: Practice the presentation as a team to ensure smooth transitions between speakers. Ensure each team member presents their specific contributions to the project.

Task 5.3: Demo Implementation: Prepare the demo to showcase: User interface prototypes. Any integrated third-party tools (e.g., OCR, APIs, datasets). Discuss the challenges and successes encountered during the project so far.

Task 5.4: Question Handling and Risk Analysis: Anticipate potential questions from instructors and supervisors.

Deliverables

no deliverables needed

WP 6: Client Side Development			
Start date: 1st of December **End date:** 30th of April			
Leader:	Defne Gürbüz	**Members involved:**	Irem Hafızoğlu, Gülbera Tekin, Ümmügülsüm Sümer

Objectives: The objective of this work package is to develop the client-side functionality of the system, focusing on providing an intuitive and responsive user interface. This includes the implementation of features like household creation and management, inventory management, shopping lists, and recipe recommendations.

Tasks:

Task 6.1: Household Management Interface: Design and implement the UI for creating and managing households. Develop features for joining households.

Task 6.2: Inventory Management Interface: Build user interfaces for adding, editing, deleting, and viewing inventory items. Integrate forms for inputting product details like quantity, category, and expiration dates.

Task 6.3: Receipt Scanning Integration: Create the front-end interface for receipt scanning, allowing users to upload or capture images of receipts. Display extracted data from the OCR and allow users to review and confirm inventory updates.

Task 6.4: Shopping List Management: Develop UI components for creating, editing, and viewing shopping lists. Include features for both manually adding items and automatically generating lists based on inventory depletion.

Task 6.5: Recipe Recommendation Interface: Implement UI components to display recipe suggestions based on inventory. Highlight missing ingredients and provide options to add them to shopping lists.

Task 6.6: Additional Pages and UI Elements: Design and develop auxiliary pages not mentioned in previous tasks, such as: User Profile and Preferences page (e.g., managing allergies and dietary restrictions). Notification Settings page. Dashboard or Home page summarizing key user data.

Task 6.7: Responsive Design Implementation: Ensure the UI is responsive and compatible with both mobile and web platforms. Test on various devices and screen sizes for optimal usability.

Deliverables

D6.1: Household management UI.

D6.2: Inventory management UI (add, edit, delete, view items).

D6.3: Receipt scanning interface integrated with OCR outputs.

D6.4: Shopping list management UI (manual and automatic generation).

D6.5: Recipe recommendation interface with missing ingredient handling.

D6.6: Additional pages UI (User Profile, Notification Settings, Dashboard).

D6.7: Fully responsive design tested across mobile and web platforms.

WP 7: Server Side Development			
Start date: 1st of December **End date:** 30th of April			
Leader:	Emirhan Büyükkonuklu	**Members involved:**	Emirhan Büyükkonuklu, Defne Gürbüz

Objectives: The objective of this work package is to design and implement the server-side functionality of the system. This includes handling data persistence, integrating with third-party libraries and services, and ensuring seamless communication between the client-side and back-end. The server-side will also be responsible for user management, business logic, and system performance optimization.

Tasks:

Task 7.1: API Design and Development: Design RESTful APIs to support client-side functionalities.

Task 7.2: Database Design and Implementation: Design and implement a scalable and normalized database schema to store data for users, households, products, inventory, and shopping lists.

Task 7.3: OCR Integration: Integrate the OCR pipeline to process receipt images and extract product information. Use the Product dataset to validate and match extracted product data with FuzzyWuzzy library. Handle edge cases such as ambiguous matches or missing items in the dataset.

Task 7.4: Recommendation Model Integration: Connect the recipe recommendation model to the server to dynamically suggest recipes based on user inventory and preferences. Implement endpoints for fetching complete and partial matches.

Task 7.5: Notification System: Build a notification service to alert users about product expirations, low-stock items, and flagged items during receipt processing.

Task 7.6: Data Synchronization: Implement data synchronization across devices for multi-user households. Resolve conflicts arising from simultaneous updates.

Task 7.7: Error Handling and Logging: Establish error-handling mechanisms for all server-side operations.

Task 7.8: Integration of Product Dataset: Clean and preprocess the Product dataset to create a searchable and structured product database. Develop matching algorithms to integrate the dataset into the OCR pipeline for receipt scanning and inventory updates.

Deliverables

D7.1: RESTful APIs supporting all core functionalities.

D7.2: Fully functional and optimized database schema**.**

D7.3: OCR pipeline integrated with the product dataset.

D7.4: Recipe recommendation endpoints linked to the client-side.

D7.5: Notification service.

D7.6: Data synchronization mechanism across devices.

D7.7: Comprehensive error handling mechanisms.

D7.8: Integrated and dynamically updatable product dataset.

WP 8: CS492 Reports

Start date: 15th of January **End date:** 20th of April

Leader:	Emirhan Büyükkonuklu,Gülbera Tekin	**Members involved:**	All members

Objectives: The objective of this work package is to prepare and deliver all required reports for the CS492, including the Detailed Design Report and the Final Project Report. These documents will comprehensively cover the project's architecture, design, development, implementation, testing, and maintenance. They will also include reflections on teamwork, ethics, professional responsibilities, and the overall learning experience during the project.

Tasks:

Task 8.1: Prepare Detailed Design Report: Document the detailed system design, including the transition from the analysis model to the design model. Include subsystem decomposition, hardware/software mapping, persistent data management, and access control/security mechanisms. Write a section discussing public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors in the design

process. Provide detailed functional and non-functional test cases, their procedures, and expected results for manual testing.

Task 8.2: Prepare Final Project Report: Refine the requirements and detailed design sections based on the project's progress. Document the final architecture and design details, including development/implementation specifics. Include test case results and the maintenance plan for the system. Discuss teamwork details and ethical responsibilities observed during the project.

Task 8.3: Create User Manual: Develop a comprehensive user manual to guide end-users in using the system effectively.

Task 8.4: Project Repository and Website: Finalize and commit all source code and executable files to the project repository.

Deliverables

D8.1: Detailed Design Report.

D8.2: Final Project Report

D8.3: User Manual.

D8.4: Project repository and website with all finalized content.

WP 9:CS 492 Demo/CS Fair			
Start date: 1st of April **End date:** 8th of May			
Leader:	İrem Hafızoğlu	**Members involved:**	All members

Objectives: The objective of this work package is to deliver a professional and comprehensive demo for the CS492 presentation session and show the project at the CS Fair. The team will present the project's progress, final implementation, and key features.

Tasks:

T**ask 9.1: Prepare Presentation Slides for CS492 Demo:** Create an overview of the project, including its objectives, design, and status.

Task 9.2: Plan and Rehearse the CS492 Demo Flow: Plan a demo sequence starting with the user interface, followed by the core functionalities and concluding with the system's benefits. Rehearse as a team to ensure smooth transitions between presenters and efficient time management.

Task 9.3: CS Fair Booth Preparation: Prepare posters and video demos, to explain the project effectively. **Task 9.4: Technical Setup for Presentations:** Ensure all hardware and software dependencies are set up and functioning correctly for both the CS492 demo and the CS Fair. Prepare a backup system to handle technical issues during the demo. **Task 9.5: Handle Q&A Session:** Prepare for potential questions about the system's design, implementation, and challenges.. **Task 9.6: Documentation for CS Fair:** Create a summary of the project for display at the fair, including its features, innovations, and benefits.
Deliverables **D9.1**: Presentation slides for the CS492 demo. **D9.2**: Rehearsed demo flow, focusing on core functionalities and benefits. **D9.3**: Engaging booth setup for the CS Fair, with interactive demo stations and posters, flyers. **D9.4**: Fully prepared technical setup for the demo. **D9.5**: Comprehensive Q&A preparation among team members. **D9.6**: Project summary for the CS Fair.

Fig.29 Gantt Chart for Work Packages

3.4 Ensuring Proper Teamwork

A structured approach is employed to ensure efficient teamwork during the development of *On the Shelf*. This approach involves establishing clear communication channels, assigning roles and responsibilities, and fostering a collaborative environment where every team member contributes effectively. The following methods will be adopted:

3.4.1 Collaborative Tools

We use collaboration tools such as Jira for task management, WhatsApp and Zoom for communication, and GitHub for version control will be utilized to ensure smooth coordination among team members. These tools will enhance transparency and provide real-time updates on project progress.

3.4.2 Regular Team Meetings

Regular team meetings are scheduled to review progress, discuss challenges, and align on priorities. These meetings foster open communication and provide a platform for brainstorming and problem-solving.

3.4.3 Agile Development Methodology

While coding, the team will follow an Agile approach, breaking the project into manageable sprints. Each sprint will focus on delivering specific features, followed by evaluation and feedback. This iterative approach allows flexibility to adapt to changes and incorporate improvements.

3.4.4 Conflict Resolution Plan

Disagreements are resolved by exchanging ideas among group members, evaluating the situation with our Course Instructor, and getting the opinion of our Innovation Expert, ensuring decisions align with the project's objectives.

3.5 Ethics and Professional Responsibilities

Developing an app like *On the Shelf* comes with significant ethical and professional responsibilities to ensure that it is not only effective but also respects users, society, and the environment.

3.5.1 User Privacy and Data Security

Protecting user data is paramount. The app collects sensitive information such as shopping habits, dietary preferences, and product inventories, which must be handled responsibly. Adhering to data protection laws ensures transparency in how data is collected, stored, and used. Users should have full control over their data, including the ability to delete

their information at any time. Any breach of these responsibilities could harm users and erode trust in the app.

3.5.2 Inclusivity and Accessibility

The app must be designed to be accessible to users with diverse needs, including those with allergies. Features such as diet preferences and a user-friendly interface will ensure safe access. Ignoring accessibility could alienate potential users and reflect poorly on the project's ethical commitment to inclusivity.

3.5.3 Reducing Food Waste

A core mission of the app is to address food waste, which has both environmental and ethical implications. By helping users optimize their grocery purchases and make the most of their inventory, the app contributes to a more sustainable lifestyle. This aligns with the responsibility of reducing harm to the environment and promoting social good.

3.5.4 Transparency and Honesty

All features and limitations of the app should be clearly communicated to users. Misleading claims about functionality or data usage would violate ethical principles and professional standards. Transparency builds trust and reflects a commitment to honesty in all interactions with users.

3.5.5 Cultural Sensitivity

Recipe recommendations and dietary suggestions must respect cultural and religious preferences. Incorporating user-defined settings for dietary restrictions and cultural preferences ensures that the app serves a diverse audience without offending or alienating users.

3.5.6 Professional Accountability

As professionals, the development team must ensure the app is reliable, secure, and well-tested before release. Regular updates and maintenance are necessary to fix bugs, improve features, and adapt to changing user needs. Failing to uphold these standards could lead to user dissatisfaction and ethical breaches.

In summary, the development of *On the Shelf* requires a steadfast commitment to ethical practices and professional standards. By prioritizing user privacy, inclusivity,

environmental sustainability, and cultural sensitivity, the app can deliver meaningful value while upholding its responsibility to humanity.

3.6 Planning for New Knowledge and Learning Strategies

The development of *On the Shelf* involves tackling complex challenges, requiring a continuous commitment to acquiring new knowledge and adopting effective learning strategies. To ensure the project's success, the following approaches will be utilized:

3.6.1 Staying Updated with Emerging Technologies

As the app heavily relies on advanced technologies like OCR and AI-based recipe predictions, keeping up with the latest advancements is essential.

3.6.2 Collaborative Learning and Knowledge Sharing

The team will foster a culture of collaboration by organizing knowledge-sharing sessions and maintaining comprehensive documentation. This ensures that all members are aligned and can contribute effectively. Pair programming and code reviews will also encourage learning from peers and improve the overall quality of the project. In addition, consulting our Supervisor and Innovation Expert will guide us in areas where we get stuck.

3.6.3 User-Centered Design Practices

Understanding user needs and preferences is a key learning focus. This involves gathering user feedback through surveys. Insights gained from them will guide the iterative development of features and improve the app’s usability and functionality.

3.6.4 Learning from Competitors and Market Trends

Analyzing existing grocery management apps and understanding market trends will help identify gaps and opportunities. This involves studying competitor features and user reviews to refine the app’s unique value proposition.

3.6.5 Skill Development in Cross-Platform Mobile App Development

Building a robust and user-friendly mobile app requires proficiency in frameworks like Flutter we are planning to use. The team is allocating time to learn and practice to enhance their technical skills in these areas.

4 Glossary

- OCR: Optical Character Recognition
- GB: Gigabyte
- AI: Artificial Intelligence
- UML: Unified Modelling Language
- IEEE: Institute of Electrical and Electronics Engineers
- FM: Family Member
- UI: User Interface

5 References

[1] "Whisk" [Online]. Available: https://www.whiskapp.net/. [Accessed 15 December 2024].

[2] "KitchenPal" [Online]. Available: http://kitchenpalapp.com/en/. [Accessed 15 December 2024].

[3] "Paprika" [Online]. Available: https://www.paprikaapp.com/. [Accessed 15 December 2024].

[4] "What's Left" [Online]. Available: https://tellmewhatsleft.de/index_en.html. [Accessed 15 December 2024].