BILKENT UNIVERSITY

ON THE SHELF

DETAILED DESIGN REPORT

CS491 SENIOR DESIGN PROJECT

Group T2420

Gülbera Tekin - 22003354

Defne Gürbüz - 22103295

Ümmügülsüm Sümer - 22103772

İrem Hafızoğlu - 22101848

Emirhan Büyükkonuklu - 22003049

Supervisor: İbrahim Körpeoğlu

Innovation Expert: İlyas Alper Karatepe

1. Introduction

1.1 Purpose of the system

In modern households, grocery management presents a persistent challenge, often leading to inefficiencies in inventory tracking, meal planning, and shopping organization. While various digital applications attempt to address these issues, existing solutions lack a seamless, integrated approach. The primary purpose of the On the Shelf application is to revolutionize grocery management by providing users with a comprehensive tool that automates inventory tracking through receipt scanning, suggests recipes based on available ingredients, and facilitates collaborative shopping lists for households. By integrating these features into a single platform, On the Shelf aims to minimize food waste, optimize grocery planning, and enhance the overall user experience.

1.2 Design goals

The On the Shelf application is designed to meet many key objectives to ensure optimal usability and performance:

- **Usability**: The interface will be intuitive, enabling users of all technical backgrounds to navigate and utilize features effortlessly.
- **Performance**: The system will operate efficiently, ensuring fast response times for scanning receipts, updating inventory, and generating shopping lists.
- **Reliability**: The application will ensure consistent performance, minimal downtime, accurate inventory tracking, and shopping recommendations.
- **Marketability**: On the Shelf is positioned as an innovative and unique grocery management tool, differentiating itself from competitors through superior integration and automation.
- **Extendibility**: The system will be designed to allow future expansions, such as integration with smart home devices and external grocery delivery services.
- **Security**: User data, including shopping habits and inventory records, will be securely stored and protected against unauthorized access.

- **Scalability**: The architecture will support a growing user base without performance degradation.
- **Maintainability**: The codebase will follow modular development principles to ensure easy updates and bug fixes.
- **Flexibility**: The application will accommodate user preferences, including dietary restrictions and household-specific shopping habits.
- **Modularity**: The system will comprise independent components for inventory tracking, meal planning, and collaborative shopping lists, ensuring efficient troubleshooting and enhancements.
- **Aesthetics**: The UI design will prioritize modern, visually appealing layouts that enhance the user experience while maintaining simplicity and functionality.

1.3 Definitions, acronyms, and abbreviations

- **OCR (Optical Character Recognition)**: A technology that extracts text from scanned receipts for inventory updates.
- **AI (Artificial Intelligence)**: Used in the application to suggest recipes and optimize grocery planning.
- **UI (User Interface)**: The application's graphical layout and interactive elements.
- **API (Application Programming Interface)**: Enables integration with external services such as online grocery stores.
- **Pantry Management**: The process of tracking food inventory within a household.
- **Collaborative Shopping List**: A shared list that multiple users can update in real-time.
- **Cloud Storage**: A remote database where user inventory and shopping lists are securely stored.

2. Current software architecture

2.1 Competitors and Alternative Solutions

The grocery management application market includes several competitors, each offering partial solutions but failing to integrate essential features comprehensively.

- **Whisk [1]**: Primarily a meal planning and recipe discovery app, lacking inventory tracking and receipt scanning capabilities.
- **KitchenPal [2]**: Focuses on pantry management but lacks collaborative shopping lists and automated inventory updates.
- **Paprika [3]**: A recipe manager with basic inventory tracking, but it does not leverage current pantry data for meal suggestions.
- **What's Left [4]**: Supports receipt scanning and inventory updates but lacks personalized dietary preference management and sustainability features.

While these applications address certain aspects of grocery management, they do not provide a unified platform that seamlessly integrates inventory tracking, automated updates, meal suggestions, and collaborative shopping list functionalities.

2.2 What On The Shelf Offers?

On the Shelf overcomes the limitations of existing solutions by offering:

- **Automated Inventory Tracking**: Using OCR to automatically scan receipts and update pantry contents.
- **Personalized Meal Suggestions**: AI-powered recipe recommendations based on available ingredients and dietary preferences.
- **Collaborative Shopping Lists**: Shared lists that multiple household members can update in real time.
- **Sustainability Features**: Tools to track expiration dates and reduce food waste through optimized consumption.

Combining these features, On the Shelf establishes itself as a comprehensive, user-centric grocery management solution that enhances efficiency, reduces waste, and simplifies household grocery planning.

3. Proposed Software Architecture

3.1. Overview

This section provides a summary of our system's structure. It outlines how the Flutter mobile app interacts with cloud services like Firebase for authentication and data storage, Cloud Functions and Cloud Run for serverless API and OCR processing, and external services like the Spoonacular API for recipe recommendations. It provides diagrams and highlights the key components and their interactions.

3.2. High-Level Architecture Diagram

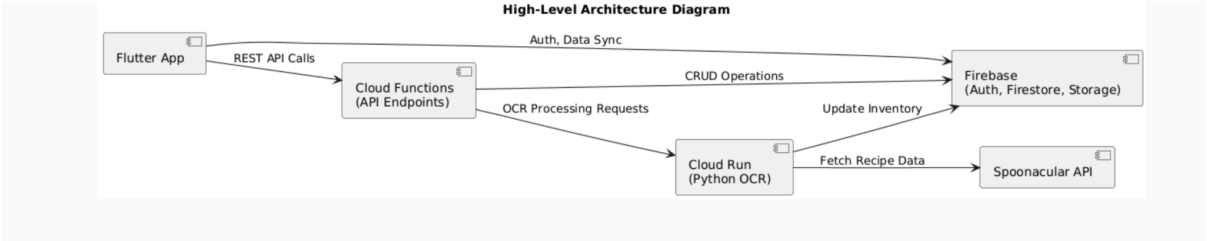


Figure 1: High Level Architecture Diagram

This diagram shows the main software components and their interactions at a high level.

Flutter Mobile App serves as the client interface.

Firebase manages core backend functionalities like user authentication, data storage, and notifications.

Cloud Functions exposes RESTful APIs for business logic.

Cloud Run hosts Python-based OCR and processing service.

Spoonacular API provides external recipe recommendations.

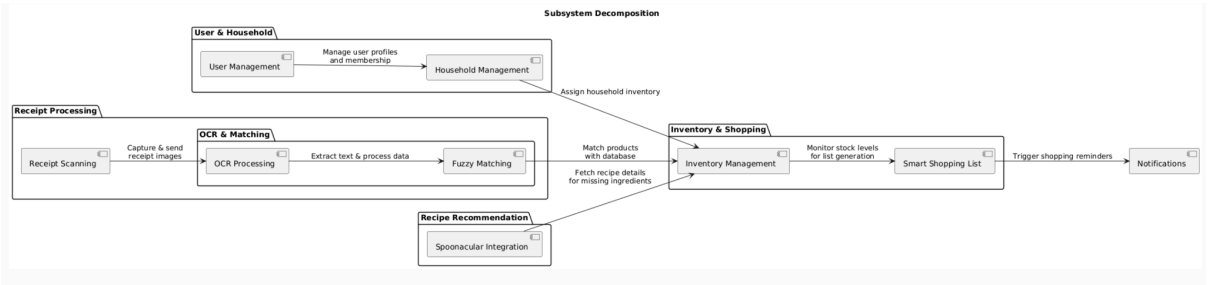
The Flutter app communicates with Firebase for authentication and data; API requests are routed through Cloud Functions to trigger OCR processing in Cloud Run, which fetches recipe data from Spoonacular and updates Firebase.

3.3. Subsystem Decomposition Diagram

Figure 2: Subsystem Decomposition Diagram

This diagram breaks the system into functional subsystems

User and Household: Contain



s User Management, Household Management.

Inventory and Shopping: Encompasses Inventory Management and Smart Shopping List generation.

Receipt Processing: Includes Receipt Scanning as well as the OCR & Fuzzy Matching components.

Recipe Recommendation: Focused on integrating with the Spoonacular API.

Notifications: A separate component dedicated to sending alerts.

3.4. Deployment Diagram

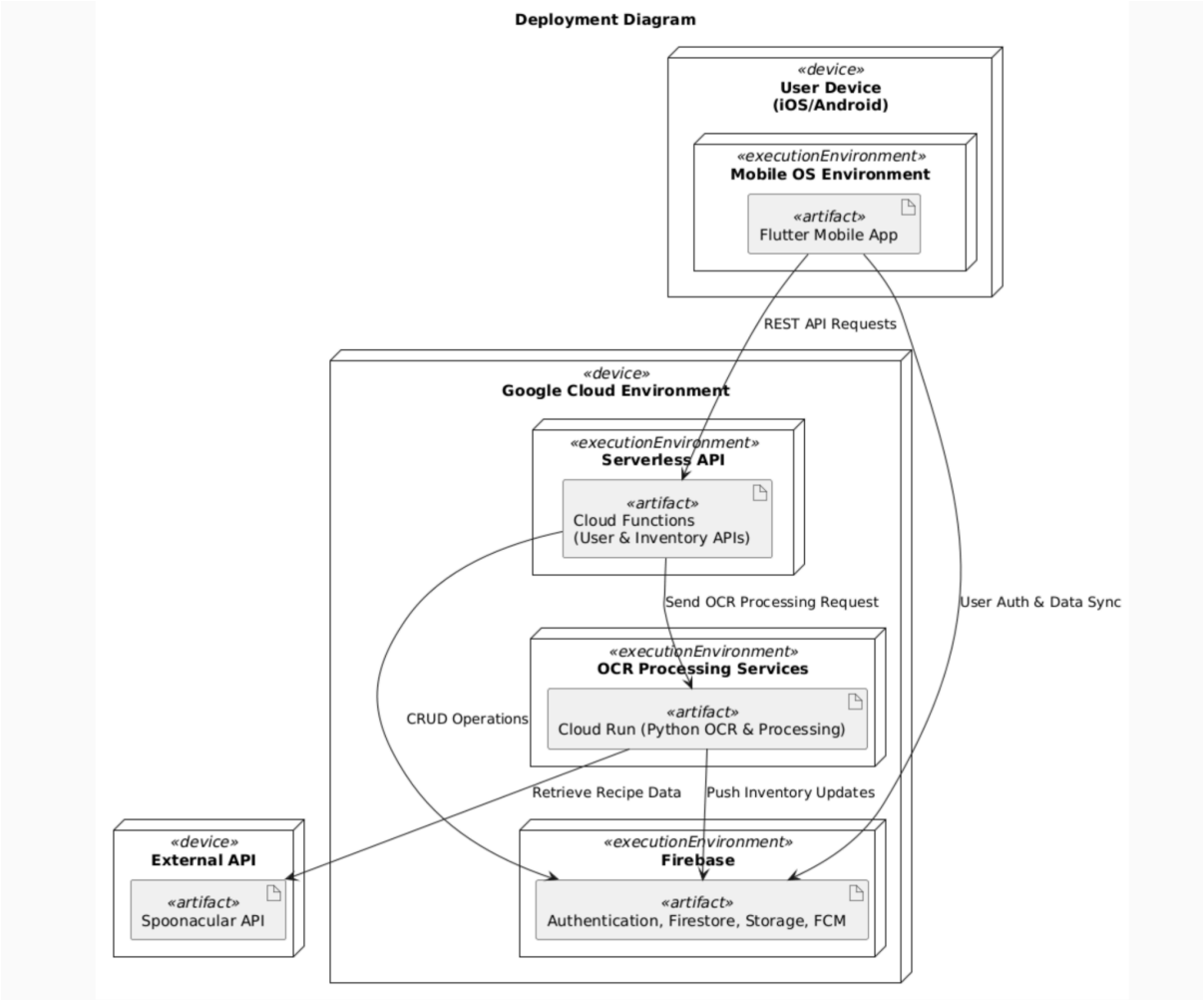


Figure 3: Deployment Diagram

This diagram illustrates where our software components will be deployed and how they interact:

User Device: The Flutter Mobile App runs on iOS/Android devices within the Mobile OS Environment.

Google Cloud Environment:

Firebase: Hosting essential backend services like Authentication, Firestore, Storage, and Firebase Cloud Messaging for notifications.

Serverless API: Cloud Functions expose API endpoints for user and inventory operations.

OCR Processing Services: Cloud Run hosts our Python-based OCR and processing services that extract text from receipts.

External API: The Spoonacular API is accessed for fetching recipe data.

The app communicates with Firebase for user authentication and data synchronization, makes REST API calls to Cloud Functions, which in turn trigger OCR processing on Cloud Run. Cloud Run updates inventory data and retrieves recipe data from Spoonacular.

3.5 Hardware/Software Mapping Diagram

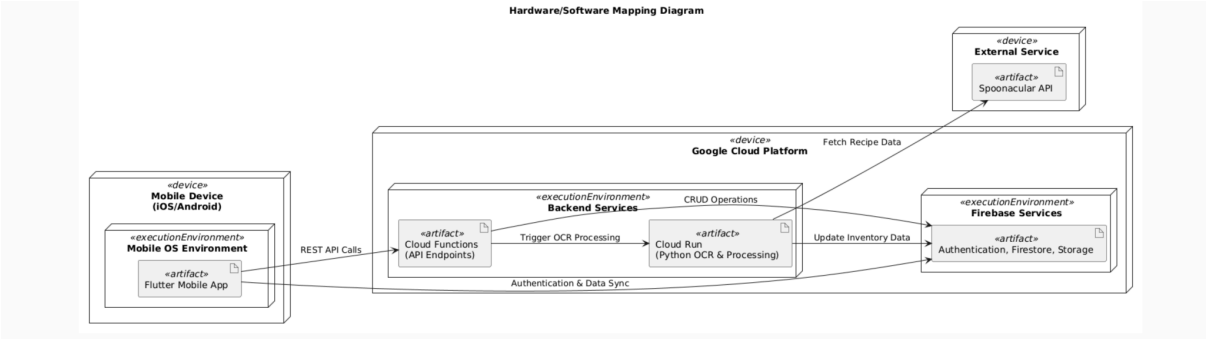


Figure 4: Hardware/Software Mapping Diagram

This diagram maps out which hardware and execution environments host our software artifacts:

Mobile Device (iOS/Android): The Flutter Mobile App runs on the Mobile OS Environment.

Google Cloud Platform is subdivided into:

Firebase Services: Runs authentication, real-time data storage (Firestore), and file storage.

Backend Services: Consisting of Cloud Functions (handling API endpoints) and Cloud Run (running the Python OCR & Processing service).

External Service: Spoonacular API is the external provider of recipe recommendations.

This diagram is correlated with the Deployment Diagram, and there is a high resemblance between the two. But they highlight different information.

3.6. Data Flow Diagram

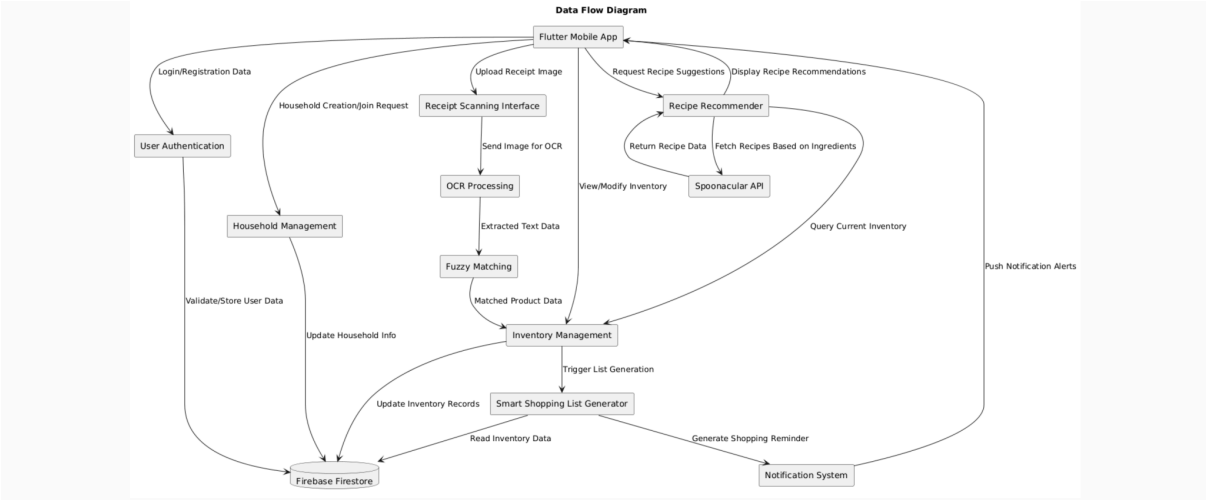


Figure 5: Data Flow Diagram

This diagram provides a detailed view of how data moves through the system.

User Authentication and Household Management: Data from the Flutter app is sent for login/registration; user data is validated and stored in Firestore.

Receipt Scanning and Inventory Update: When a receipt is uploaded, the image is sent to the receipt scanning, and then processed by OCR, which goes through fuzzy matching. Matched product data is used to update the inventory.

Smart Shopping List and Notifications: Inventory changes trigger the smart shopping list generator, which reads inventory data and prompts notifications via the notification system.

Recipe Recommendation: The app requests recipe suggestions by querying current inventory; the recipe recommender calls the Spoonacular API and displays the returned recipes.

3.7. Class Diagram

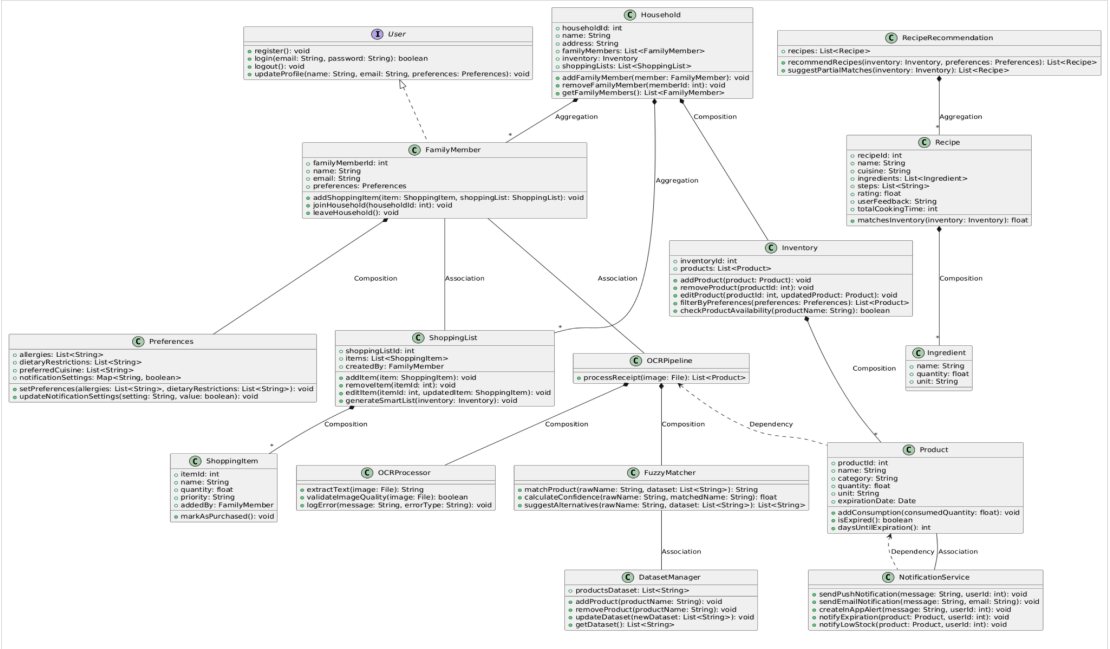


Figure 6: Class Diagram

4. Subsystem Services

The system is modularized into several subsystems, each providing services that together fulfill the overall application functionality. Below is an overview of each subsystem along with the services they provide.

1) **User Management Service**
    a) User Registration and Login: Handles the creation and authentication of user accounts using Firebase Authentication.
    b) Profile Management: Allows users to update personal information and preferences.
2) **Household Management Service**
    a) Household Creation and Joining: Enables users to create households and join them via invitation.

b) Membership Management: Manages the list of household members and their roles.

3) **Inventory Management Service**
    a) Product CRUD Operations: Allows adding, updating, and removing products from the inventory.
    b) Stock Monitoring: Checks inventory levels.
4) **Receipt Processing Service**
    a) Receipt Scanning: Captures and uploads receipt images from the mobile app.
    b) OCR Processing: Uses Tesseract via Python to extract text from receipt images.
5) **Fuzzy Matching Service**
    a) Text Matching: Processes OCR output to match product names against a standardized product dataset.
6) **Smart Shopping List Service**
    a) List Generation: Automatically generates a shopping list based on inventory.
    b) Manual Editing: Allows users to modify the generated list.
7) **Recipe Recommendation Service**
    a) Recipe Fetching: Uses the Spoonacular API to fetch recipe suggestions based on available inventory and user dietary preferences.
    b) Missing Ingredient Identification: Highlights ingredients that are missing from the inventory.
8) **Notification Service**
    a) Alert Generation: Sends push notifications for events such as expiring products, low inventory, and shopping list updates.
    b) Scheduling: Can schedule reminders based on user-defined criteria or automated triggers.

5. Test Cases

Test ID	TC01	**Category**	Functional	**Severity**	High
Objective	Verify that users can add an item to the pantry				
Steps	1. Open the app 2. Navigate to the “Mutfağımdakiler” 3. Select a category 4. Click on the add button on the top right of the page 5. Enter item details and save				
Expected	The item is successfully added to the pantry.				

Test ID	TC02	**Category**	Functional	**Severity**	High
Objective	Verify that users can delete an item from the pantry				
Steps	1. Open the app 2. Navigate to the “Mutfağımdakiler” 3. Select a category 4. Select an item 5. Click "Delete"				
Expected	The item is successfully removed from the pantry.				

Test ID	TC03	**Category**	Functional	**Severity**	Medium
Objective	Verify that users can edit an item in the pantry				
Steps	1. Open the app 2. Navigate to the “Mutfağımdakiler” 3. Select a category 4. Select an item 5. Edit details and save				
Expected	The item details are updated successfully .				

Test ID	TC04	**Category**	Functional	**Severity**	High
Objective	Verify that users can filter items by category				
Steps	1. Open the app 2. Navigate to the “Mutfağımdakiler” 3. Select a category				
Expected	Only items matching the selected category are displayed.				

Test ID	TC05	**Category**	Functional	**Severity**	Medium
Objective	Verify that users can search for items in the pantry				
Steps	1. Open the app 2. Navigate to the “Mutfağımdakiler” 3. Select a category 4. Enter an item name in the search bar 5. View results				
Expected	Only relevant items appear in search results.				

Test ID	TC06	**Category**	Functional	**Severity**	High
Objective	Verify that users can input their allergies				
Steps	1. Open the app 2. Navigate to the settings 3. Click manage allergens 4. Add allergies 5. Save				
Expected	Allergies are successfully saved.				

Test ID	TC07	**Category**	Functional	**Severity**	High
Objective	Verify that generated recipes exclude allergic ingredients				
Steps	1. Set allergies 2. Navigate to “Tarifler” 3. Generate a new recipe				
Expected	Generated recipes do not include allergens.				

Test ID	TC08	**Category**	Functional	**Severity**	High
Objective	Verify that users can save recipes				
Steps	1. Open recipes section 2. Click "Tarifi Kaydet" 3. Enter details and save				
Expected	The recipe is saved successfully.				

Test ID	TC09	**Category**	Functional	**Severity**	High
Objective	Verify that the app syncs data correctly				
Steps	1. Add an item to the pantry 2. Restart the app				
Expected	The item remains in the pantry after restart.				

Test ID	TC10	**Category**	Non-Functional	**Severity**	High
Objective	Verify app loading time is under 3 seconds				
Steps	1. Open the app				
Expected	The app loads within 3 seconds.				

Test ID	TC11	**Category**	Non-Functional	**Severity**	Medium
Objective	Verify app response time for adding an item				
Steps	1. Add an item to the pantry 2. Measure response time				
Expected	The item is added within acceptable response time.				

Test ID	TC12	**Category**	Functional	**Severity**	High
Objective	Verify that users can log in using Firebase authentication				
Steps	1. Open the app 2. Enter valid credentials 3. Click "Giriş Yap"				
Expected	User successfully logs in.				

Test ID	TC13	**Category**	Functional	**Severity**	Medium
Objective	Verify that users cannot log in with incorrect credentials				
Steps	1. Open the app 2. Enter invalid credentials 3. Click "Giriş Yap"				
Expected	Error message is displayed.				

Test ID	TC14	**Category**	Functional	**Severity**	High
Objective	Verify that users can reset their password				
Steps	1. Click "Şifremi Unuttum" 2. Enter email 3. Check email for reset link				
Expected	Password reset email is received.				

Test ID	TC15	**Category**	Functional	**Severity**	Medium
Objective	Verify that users can log out				
Steps	1. Open settings 2. Click "Çıkış Yap"				
Expected	User is logged out and returned to login screen.				

Test ID	TC16	**Category**	Functional	**Severity**	High
Objective	Verify that users can update their profile information				
Steps	1. Open profile settings 2. Edit details and save				
Expected	Profile updates successfully.				

Test ID	TC17	**Category**	Functional	**Severity**	High
Objective	Verify that the database updates in real-time				
Steps	1. Open the pantry on two devices 2. Add an item on Device 1 3. Check Device 2				
Expected	The new item appears instantly on Device 2.				

Test ID	TC18	**Category**	Non-Functional	**Severity**	Medium
Objective	Verify that the app works properly in offline mode				
Steps	1. Disable internet connection 2. Open the app 3. View the “Mutfağımdakiler”				
Expected	The app displays locally stored data.				

Test ID	TC19	**Category**	Functional	**Severity**	High
Objective	Verify that users can sort pantry items by name				
Steps	1. Open the pantry 2. Click on the sort button 3. Select "İsme Göre Sırala"				
Expected	Items are sorted alphabetically.				

Test ID	TC20	**Category**	Functional	**Severity**	Medium
Objective	Verify that users can view a history of consumed items				
Steps	1. Open the app 2. Navigate to "Tüketim Geçmişi"				
Expected	A list of previously used items is displayed.				

Test ID	TC21	**Category**	Non-Functional	**Severity**	Medium
Objective	Verify that the app does not drain battery excessively				
Steps	1. Use the app continuously for an hour 2. Check battery consumption				
Expected	Battery usage remains within acceptable limits.				

Test ID	TC22	**Category**	Non-Functional	**Severity**	High
Objective	Verify that the app works on different screen sizes				
Steps	1. Open the app on multiple devices				
Expected	The UI adjusts correctly for all screen sizes.				

Test ID	TC23	**Category**	Non-Functional	**Severity**	Medium
Objective	Verify that the app functions properly in dark mode				
Steps	1. Enable dark mode in device settings 2. Open the app				
Expected	The app appears correctly in dark mode.				

Test ID	TC24	**Category**	Functional	**Severity**	Medium
Objective	Verify that pantry data only visible for users in same household				
Steps	1. Log in with User u1 that is part of Household h1 2. Add an item to the pantry 3. Log out and log in with User u1 that is not a part of Household h1				
Expected	Pantry data from User u1 is not visible in User u2.				

Test ID	TC25	**Category**	Functional	**Severity**	Medium
Objective	Verify that users can clear all pantry data				
Steps	1. Open settings 2. Click "Mutfağımdakileri sil"				
Expected	All pantry items are removed.				

Test ID	TC26	**Category**	Non-Functional	**Severity**	High
Objective	Verify that the app handles large amounts of data efficiently				
Steps	1. Add 10,000 items to the pantry				
Expected	The app remains responsive and does not crash.				

Test ID	TC27	**Category**	Functional	**Severity**	High
Objective	Verify that the app extracts product details accurately from a clear receipt image				
Steps	1. Open the app and 2. Select the "Fiş Okut" option 3. System processes the receipt and extracts product details 4. System displays extracted data for user confirmation 5. User confirms the data.				
Expected	The inventory is updated.				

Test ID	TC28	**Category**	Functional	**Severity**	Medium
Objective	Verify that users receive an error for an invalid receipt scan				
Steps	1. Open the app 2. Click "Fiş Okut" 3. Scan an invalid receipt				
Expected	An error message is displayed.				

Test ID	TC29	**Category**	Functional	**Severity**	Medium
Objective	Verify that the system correctly identifies and adds low-stock items to the shopping list				
Steps	1.Ensure some products in the inventory are running low. 2.Open the app and navigate to the "Alışveriş Listesi" section. 3.Verify if low-stock items appear automatically in the list.				
Expected	The shopping list includes low-stock items with the correct quantities.				

Test ID	TC30	**Category**	Functional	**Severity**	Medium
Objective	Verify that users can remove incorrectly suggested items				
Steps	1.Open the "Alışveriş Listesi" section. 2.Select an item from the list. 3.Click on the remove option.				
Expected	The item is successfully removed from the list.				

Test ID	TC31	**Category**	Functional	**Severity**	High
Objective	Verify that the OCR function works with receipts of different layouts and font styles				
Steps	1.Collect receipts from different stores with varying formats 2.Scan each receipt using the app				
Expected	The OCR extracts and interprets text accurately across different receipt formats.				

Test ID	TC32	**Category**	Functional	**Severity**	High
Objective	Verify the OCR can handle lengthy receipts without truncation				
Steps	1.Scan a long grocery receipt with more than 20 items				
Expected	The system extracts all items correctly without missing entries.				

Test ID	TC33	**Category**	Functional	**Severity**	High
Objective	Verify that the system correctly distinguishes between kitchen-related and non-kitchen items				
Steps	1.Scan a receipt containing both kitchen and non-kitchen products				
Expected	The system filters and only adds kitchen-related items to inventory.				

Test ID	TC34	**Category**	Functional	**Severity**	High
Objective	Verify that the system updates quantities instead of duplicating item				
Steps	1.Scan a receipt containing items already in the inventory				
Expected	The system updates existing product quantities instead of adding duplicates.				

Test ID	TC35	**Category**	Functional	**Severity**	Medium
Objective	Verify OCR performance on faded or low-contrast receipts				
Steps	1.Scan a faded receipt with light ink.				
Expected	The system accurately extracts text or prompts the user for manual verification.				

Test ID	TC36	**Category**	Functional	**Severity**	High
Objective	Verify that extracted product names are matched correctly with known kitchen product datasets				
Steps	1.Scan a receipt and allow the system to process it				
Expected	The system matches product names accurately with its kitchen product database.				

Test ID	TC37	**Category**	Functional	**Severity**	High
Objective	Verify that the system recommends recipes using current inventory items				
Steps	1.Navigate to the “Tarifler” section 2.Check suggested meals				
Expected	Recipes use available ingredients.				

Test ID	TC38	**Category**	Functional	**Severity**	Medium
Objective	Verify users can join a household using the household ID				
Steps	1.Navigate to "Eve Katıl." 2.Enter the household ID				
Expected	The user joins the household.				

Test ID	TC39	**Category**	Functional	**Severity**	High
Objective	Verify household creation works correctly				
Steps	1.Navigate to "Household Management." 2.Click "Create Household." 3.Enter household name				
Expected	A new household is created.				

Test ID	TC40	**Category**	Functional	**Severity**	High
Objective	Verify that users in the same household have a shared pantry				
Steps	1. User A logs into the app and adds an item to the pantry 2. User B logs into the same household account on a different device 3. User B checks the pantry				
Expected	The pantry data is identical for both users, and the item added by User A is visible to User B				

Test ID	TC41	**Category**	Functional	**Severity**	High
Objective	Verify that users can delete their accounts successfully				
Steps	1. User logs into the app 2. User navigates to the account settings page 3. User selects the "Hesabı Sil" option 4. User confirms account deletion 5. User attempts to log in again using the deleted account credentials				
Expected	The account is deleted, and the user is unable to log in with the deleted credentials. All associated data is removed as per the app's policy				

Test ID	TC42	**Category**	Functional	**Severity**	High
Objective	Verify that users in the same household can add items to the shopping list and that it is visible to all household members				
Steps	1. User A logs into the app. 2. User A navigates to the shopping list section. 3. User A adds an item to the shopping list. 4. User B (who belongs to the same household) logs into their account. 5. User B navigates to the shopping list section.				
Expected	The item added by User A is visible in the shopping list for User B. All household members can see and update the shared list.				

Test ID	TC43	**Category**	Functional	**Severity**	Medium
Objective	Verify that users in the same household can add notes to the shopping list and that other household members can see them				
Steps	1. User A logs into the app. 2. User A navigates to the notes section. 4. User A adds a note (e.g., "Get whole wheat"). 5. User B (who belongs to the same household) logs into their account. 6. User B navigates to the notes section.				
Expected	The note ("Get whole wheat") are visible to User B and other household members.				

Test ID	TC44	**Category**	Functional	**Severity**	Medium
Objective	Verify that items in the pantry are placed in the correct category based on their type				
Steps	1. User logs into the app. 2. User adds various items to the pantry. 3. User navigates to the pantry section and checks the category labels.				
Expected	Items are automatically categorized correctly.				

Test ID	TC45	**Category**	Functional	**Severity**	Medium
Objective	Verify that users can leave a household successfully.				
Steps	1.Navigate to settings and select "Aile Hesabından Ayrıl” 2.Confirm the action when prompted. 3.System removes the user from the household.				
Expected	User successfully leaves the household and is redirected to the household management screen.				

Test ID	TC46	**Category**	Functional	**Severity**	Medium
Objective	Verify that duplicate email registration is not allowed.				
Steps	1.Enter an email already used for another account. 2.Fill in the remaining fields and attempt to sign up.				
Expected	The app displays an error message for duplicate email				

Test ID	TC47	**Category**	Functional	**Severity**	Low
Objective	Verify that invalid email formats are rejected.				
Steps	1.Enter an incorrectly formatted email (e.g., "user@com" or "user#email.com"). 2.Click "Sign Up."				
Expected	The app displays an error message for user to enter a valid email				

Test ID	TC48	**Category**	Functional	**Severity**	Medium
Objective	Verify that duplicate products are prevented in the inventory.				
Steps	1.Attempt to add a product with an identical name.				
Expected	System prompts to update quantity instead.				

Test ID	TC49	**Category**	Functional	**Severity**	Medium
Objective	Verify that users are prompted to provide missing product details.				
Steps	1.Try adding a product without a name or quantity.				
Expected	System highlights missing fields and prevents submission.				

Test ID	TC50	**Category**	Functional	**Severity**	Medium
Objective	Verify that users can edit products in the shopping list.				
Steps	1.Navigate to the "Alışveriş Listem" section. 2.Select an item and modify its name, quantity, or category. 3.Save the changes.				
Expected	The edited product details are updated in the shopping list successfully.				

Test ID	TC51	**Category**	Functional	**Severity**	Medium
Objective	Verify that users can delete products from the shopping list				
Steps	1.Navigate to the "Alışveriş Listem" section. 2.Select an item and choose the delete option. 3.Confirm deletion.				
Expected	The selected item is removed from the shopping list.				

6. Consideration of Various Factors in Engineering Design

6.1 Constraints

The following can be listed as constraints:

Aesthetics

- The user interface will emphasize a clean and visually appealing design, ensuring an intuitive and clutter-free experience.
- Colors and icons will follow universal design principles to enhance attractiveness and ease of use.

Code Standards

- The app will adhere to secure coding standards for mobile development, ensuring reliability and protection against vulnerabilities.

Constructability

- A single Flutter codebase will ensure seamless deployment on both iOS and Android platforms.

User Experience & Ergonomics

- The app will minimize user effort by incorporating automation features like OCR for receipt scanning.
- Notifications will be designed to be simple and non-intrusive to avoid overwhelming users.

Scalability & Extensibility

- The system architecture will support the easy integration of new features for future improvements.

Core Functionality

- The app will deliver key features as outlined in the functional requirements, including inventory tracking and recipe recommendations.

Cross-Device Compatibility

- Users will be able to sync their data across multiple devices under the same account for a seamless experience.

Legal Compliance

- The app will adhere to data privacy regulations such as KVKK, ensuring user information is protected.

Marketability

- The app will address key consumer challenges, such as reducing food waste and simplifying grocery management, to attract a wide user base.

Policy Alignment

- The app will be designed in accordance with sustainability policies and responsible consumption guidelines.

Regulatory Compliance

- Deployment on app stores will follow regional guidelines for both Google Play Store and Apple App Store.

Development Timeline

- The project will adhere to a structured development schedule, including key milestones for testing, feedback collection, and final release.

Industry Standards

- The app will be built in compliance with established mobile application development standards.

Sustainability

- The app will encourage sustainable practices by promoting reduced food waste and better resource management.
- Energy-efficient design principles will be implemented to minimize power consumption during usage.

Minimum Hardware & Software Requirements

iOS

- OS Requirement: iOS 13 or later (for compatibility with Core ML and Vision frameworks).
- Hardware Requirement: Devices with an A11 Bionic chip or newer (iPhone 8 and above) to support efficient OCR processing.

Android

- OS Requirement: Android 8.0 or later.
- Hardware Requirement: Devices with a minimum of 2GB RAM.

Global Accessibility

- Cross-platform availability on both iOS and Android will maximize global reach.

Cultural Adaptability

- Recipe Recommendations: The app will provide region-specific recipes, considering cultural dietary preferences.
- Shared Accounts: Sub-account functionality will support shared grocery responsibilities in family-oriented cultures.

Social Impact

- Food Waste Reduction: The app will contribute to solving a major social issue by minimizing food waste and encouraging responsible consumption.
- Inclusivity: Features such as accessibility settings and dietary customization will cater to diverse user needs.

Environmental Responsibility

- Sustainability Focus: Encouraging users to reduce food waste will have a positive environmental impact.
- Energy Efficiency: The app will be designed to consume minimal resources, reducing energy consumption.

6.2 Standards

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

7. Teamwork Details

7.1 Contributing and functioning effectively on the team

Effective teamwork is essential for the successful development of the **On the Shelf** application. The team members are responsible for contributing to various aspects of the project, including system design, development, documentation, and testing. Responsibilities are distributed based on expertise to ensure an efficient workflow. Regular meetings, brainstorming sessions, and code reviews help maintain alignment across different roles. Clear communication and task delegation allow the team to meet deadlines while ensuring the quality and functionality of the system.

7.2 Helping creating a collaborative and inclusive environment

A collaborative and inclusive environment is fostered through open communication and teamwork. Team members actively participate in discussions, share insights, and provide constructive feedback to improve the system design. The use of collaborative tools, such as version control systems, shared documentation platforms, and real-time messaging applications, ensures transparency and accessibility for all members. Encouraging diverse perspectives helps refine ideas and leads to a more user-centered and innovative solution.

7.3 Taking lead role and sharing leadership on the team

Leadership within the team is shared, with different members taking the lead in various aspects of the project. Some focus on defining system architecture, while others oversee documentation, user interface design, or testing strategies. Decision-making is a collective process, where team members contribute ideas and collaborate to refine project objectives. Leadership roles shift depending on the task at hand, allowing for flexibility and ensuring that all aspects of the project receive adequate attention. By distributing leadership responsibilities, the team maintains a balanced workflow and efficiently addresses challenges as they arise.

8. Glossary

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

9. References

[1] "Whisk" [Online]. Available: https://www.whiskapp.net/. [Accessed 09 March 2025].

[2] "KitchenPal" [Online]. Available: http://kitchenpalapp.com/en/. [Accessed 09 March 2025].

[3] "Paprika" [Online]. Available: https://www.paprikaapp.com/. [Accessed 09 March 2025].

[4] "What's Left" [Online]. Available: https://tellmewhatsleft.de/index_en.html. [Accessed 09 March 2025].