

# CSCI334

## Assignment 2

### Software Requirements Specification



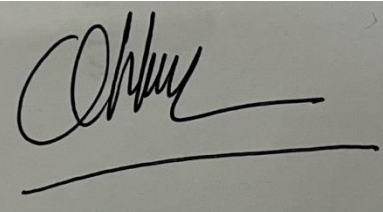
### Group 6

**S  
M  
A  
R  
T  
O  
N  
I  
O  
N**

Smart  
Monitoring  
And  
Resourceful  
Tracking  
Of  
Nutrition  
In  
Our  
Neat Fridge



## Member contribution

Name	Contribution		Signature
Myunggyun Yu	Figma design , Prototype & SRS	Contributed	
Duc Huy Tran	Figma design & SRS	Contributed	
Ashutosh Yadav	Documentation Fridge Use Case Fridge UI	Contributed	Ashutosh
Bryce Van Den Berg	SRS	Contributed	Bryce VDB
Dinh Quoc Huy Nguyen	Use case diagram & Documentation	Contributed	
Gia Bach Nhu	Use case description, SRS Documentation	Contributed	Bach

## Table of Contents

<b>1. INTRODUCTION .....</b>	<b>6</b>
1.1. PURPOSE.....	6
1.2. SCOPE .....	6
1.3. DEFINITIONS, ACRONYMS, AND ABBREVIATIONS .....	7
1.4. REFERENCES .....	7
1.5. OVERVIEW .....	7
1.6. SCORING MATRIX .....	7
1.7. PRODUCT DESIGN PRINCIPLES .....	9
<b>2. OVERALL DESCRIPTION .....</b>	<b>11</b>
2.1. PRODUCT PERSPECTIVE .....	11
2.1.1 <i>System Interfaces</i> .....	11
2.1.2 <i>User Interfaces</i> .....	12
2.1.3 <i>Hardware Interfaces</i> .....	12
2.1.4 <i>Software Interfaces</i> .....	12
2.1.4.1 <i>External System Interface</i> .....	12
2.1.4.2 <i>User Interface</i> .....	12
2.1.5 <i>Communication Interfaces</i> .....	12
2.1.6 <i>Memory Constraints</i> .....	12
2.1.7 <i>Operations</i> .....	12
2.2. PRODUCT FUNCTIONS.....	13
2.3. USER CHARACTERISTICS .....	13
2.4. CONSTRAINTS .....	13
2.5. ASSUMPTIONS AND DEPENDENCIES .....	13
<b>3. SPECIFIC REQUIREMENTS .....</b>	<b>14</b>
3.1. FUNCTIONAL REQUIREMENTS.....	14
3.1.1 <i>Users Management Subsystem</i> .....	14
3.1.2 <i>Administrator Side</i> .....	14
3.2. NON-FUNCTIONAL REQUIREMENTS .....	40
3.5 FIGMA PROTOTYPE VIDEO (MOBILE APPLICATION) .....	49
3.6 FRIDGE UI .....	50
3.7 FRIDGE PROTOTYPE .....	53

## Revision History

Date	Version	Description	Author
15/03/2024	1.0	First SRS	Myunggyun Yu Duc Huy Tran
31/03/2024	2.0	Final Draft	Myunggyun Yu Duc Huy Tran Ashutosh Yadav Bryce Van Den Berg Dinh Quoc Huy Nguyen
01/04/2024	2.1	Final	Ashutosh Yadav Gia Bach Nhu

## 1. Introduction

### 1.1. Purpose

The purpose of this document is to describe the specifications on the external behaviors of a food inventory management system. The intended audience of this document includes the prospective software development team and the potential users of the system.

### 1.2. Scope

The software to be produced is an app-based electronic food inventory management system, which will be referred to as “Smart Fridge system” through this document.

The Smart Fridge system aims to serve two groups of users - system administrator and family members of the smart fridge which is going to use the system. The need of administrators is to maintain smooth operations of the system. The purpose of family members is to use Smart Fridge system resources. To ensure client satisfaction and efficient management, the Smart Fridge system is divided into two independent yet interconnected subsystems.

- Users Management Subsystem manages profiles of all users of the Smart Fridge system.
- Activities Management Subsystem manages all activities, processes and transactions occurring within the Smart Fridge system, e.g., viewing and updating food inventory.

To view, manage, update, and search for food that is currently stored within the user's fridge, Smart Fridge also enables the use of smart refrigerators. This function greatly improves user convenience when it comes to inventory management and meal plan suggestions.

The Smart Fridge system cannot take the role of a nutritionist, even if it is designed to help consumers manage and use food. The following limitations result from the Smart Fridge system's inability to imitate or capture some normal fridge usage or nutritionist guidance:

- Family members still must purchase food directly.
- Family members can keep track of their food consumption and nutrition, but these numbers cannot be utilized to guide or reflect on their health status.

### 1.3. Definitions, Acronyms, and Abbreviations

Smart Fridge, Fridge = Refers to the Application Running on the Fridge

### 1.4. References

E-Library System Software Requirements Specifications, SRS sample (Sample - SoftwareRequirementsSpecification.doc)

### 1.5. Overview

The rest of this document can be divided into two main sections:

1. The Overall Description (section 2) outlines the overarching elements that impact the system and its requirements.
2. The Specific Requirements (section 3) includes all of the software requirements that are necessary to suit customer needs.

### 1.6. Scoring Matrix

This Scoring Matrix is for how meal suggestions will be implemented.

The Smart Fridge application, as part of the system requirements, needs to be able to suggest meals, this is to help with planning a family's meals for the week. Since every family has different eating habits, cultural backgrounds, and ways of life this quickly becomes a complex problem.

A few ways of implementing suggesting meals were discussed:

- Fixed Database – The product would have a private database of recipes. The program will choose from the recipes based on the fridges contents to make it as convenient as possible for families to create.
- Connect to Google – The product would, when fridge contents are selected, google for recipes involving those ingredients to suggest meals to the family.
- AI suggestions – The product would use an Artificial Intelligence trained on recipe data to generate potential recipes for the family's using some of the ingredients in the fridge.

To choose between these suggestions a few evaluation criteria were used:

- Time Bound – How quickly can this suggestion be implemented? Given this project needs to be completed within 2-3 months, time is an important factor for considering these potential solutions. (weighting = 25%)
- Feasibility – Can the team realistically implement this solution? Considering the team doesn't have years of experience designing and developing programs this is a reasonable factor to consider when implementing potential solutions. (weighting = 15%)
- Easy to Use – Can a user of the program easily find recipes to choose from? If users find this feature to be too much of a hassle to use when doing meal planning, then it will not get used. As such this is an important factor for implementing potential solutions. (weighting = 25%)

- Beauty – Do users like the program interface? Does it provide pictures for the recipes? Are the recipes tidy? Users are more likely to want to use this type of feature when it looks presentable and tidy, so this is a reasonable factor to consider for potential solutions. (weighting = 15%)
- Adequacy – Does this solution correctly solve the requirement of suggesting meals? Are there downsides to this approach? If the program doesn't give meals that will help with inventory management or doesn't correctly provide instructions for the recipes then this feature will not be used, it is a somewhat important factor when considering potential solutions (weighting = 20%)

	Weights	Fixed Database		Connect to Google		AI Solution	
Time Bound	0.25	5	1.25	3	0.75	2	0.5
Feasibility	0.15	5	0.75	4	0.6	2	0.3
Easy to Use	0.25	3	0.75	5	1.25	5	1.25
Beauty	0.15	2	0.3	4	0.6	3	0.45
Adequacy	0.2	3	0.6	4	0.8	4	0.8
Total:			3.65		4.0		3.3

The Scoring matrix has shown that Connecting to Google is the best potential option for this feature.



## 1.7. Product design principles

### 1. Adequacy

An adequate design is a design that meets its stakeholders needs, while subject to specified constraints. By creating detailed use case diagrams and use case descriptions we can ensure the features of the application accurately meet the stakeholders needs. For example, how meal suggestions will work for the fridge application is properly defined in Use case 12 and as such, will accurately meet the stakeholder requirements.

### 2. Beauty

Beauty in design is not only about aesthetics, but also organization, and simplification. Planning the features of the smart fridge application in depth makes it easy to organize and simplify the coding structure. In Use case 5: Add Product to Shopping List, the procedure of adding an item to the shopping list is broken down to the necessary steps, allowing for more structured and organized code for the shopping list.

Planning also helps improve the user experience as prototyping the front end design, figuring out what works and iterating on it leads to a more streamlined, enjoyable experience. In the Hi-Fi figma wireframe, when adding products to the shopping list, some main categories for items are shown, like fruit, vegetables, and meats. Within these categories common items are displayed and a search feature allows the user to easily navigate to the item they want to add to the list. In this case its beauty comes from simplicity and ease of use.

### 3. Economy

To be as economical as possible the team will be focusing on the most important features. The features of the program have been clearly defined in the use case descriptions and diagrams to aid ranking in terms of importance. For example, we can see that Use case 8: View Fridge/Inventory Contents is more necessary for the program to function than Use case 9 which manages alerts. Importantly the features of the Smart Fridge system are in line with the project scope to reduce the chance of project scope creep, as scope creep is a common problem in design which will increase expenses.

### 4. Feasibility

Product feasibility is demonstrated through clearly defining the success criteria which was created in the project scope, clearly defining the features and evaluating difficult decisions using scoring matrices. The Project Scope for the Smart Fridge system outlines what the system can do, from viewing and managing the fridge contents, to creating and storing a shopping list. This outline of the Smart Fridge system is then broken down in the Use case diagrams and descriptions till each step of constructing the system is feasible. When

difficult issues arise, such as how to suggest meals to the user, a scoring matrix is then used to evaluate each option.

## **5. Simplicity**

Creating a simple design requires a well planned structure and documentation, this is achieved by defining and iterating on the software features to keep them concise. In Use Case 12: Suggesting meal plans and recipes, there are goals that need to be achieved, users want meal suggestions based on their food preferences, dietary restrictions, and nutritional goals. These goals help simplify the problem of creating a complex application by sectioning off a small area of the program to work on. The goals are then broken down into a flow of steps that are simple to implement.

Simple design is also created by iterating on the UX design, by making lo-fi and hi-fi prototypes in figma.

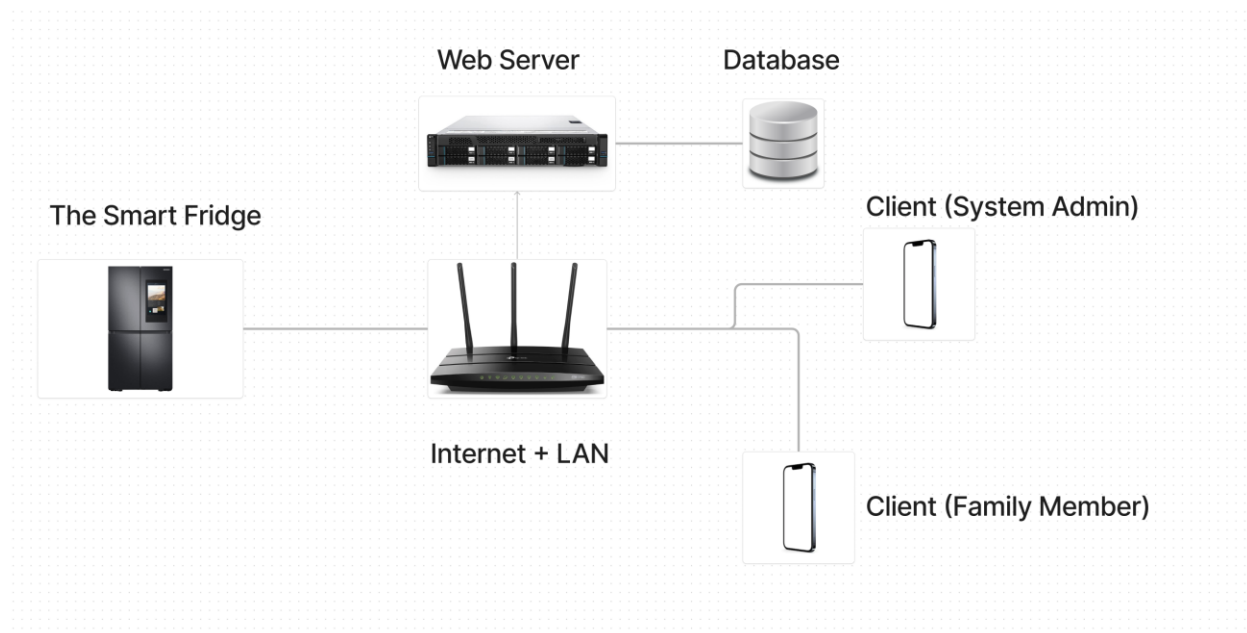
## 2. Overall Description

### 2.1. Product perspective

Nowadays, the majority of food inventory management tasks are carried out and overseen manually by family members, particularly parents. These processes are time-consuming and require significant effort, which presents a difficulty in today's fast-paced and busy lifestyle.

Given the characteristics of this situation, there is a need to develop a solution that can successfully aid family members in maintaining and utilizing their food inventory. The Smart Fridge system strives to be an ideal tool for accomplishing these objectives.

#### 2.1.1 System Interfaces



The fridge is connected to the LAN via a router, the ports should be allowed on the router to make it accessible to the clients while accessing offline. The fridge also connects to the internet, so even if the user's aren't connected to the LAN they can access to the fridge management via internet.

### 2.1.2 User Interfaces

The user interfaces provided to client machines must be GUI and must be accessible through any smart devices. The user interfaces provided to system administrators differ from those provided to family members.

All interactions with the Database, which is operated by a MySQL server, must be carried out indirectly via the graphical user interfaces (GUIs) supplied by the Smart Fridge system.

### 2.1.3 Hardware Interfaces

The backend requires 4GB of RAM and 4 CPU cores to run on the server. An Android OS with a minimum of 3GB of RAM is needed for the fridge interface. The end-user application can run on both Android and iOS.

The fridge needs to be connected to internet via, local router.

### 2.1.4 Software Interfaces

#### 2.1.4.1 External System Interface

- The system can import existing user accounts from other systems.

#### 2.1.4.2 User Interface

- The user interacts with the system through a mobile application.
- The system supports both iOS and Android devices.

### 2.1.5 Communication Interfaces

All the users need to be connected to internet in respect to communicate with their fridge when outside the LAN. Users can communicate with the fridge without requiring Internet connect when connected to the same Network as fridge. The ports need to be allowed to via the router.

### 2.1.6 Memory Constraints

Application on fridge required 250MB minimum and 2GB (Additionally after the Android OS ) max to store the application and run efficiently. Mobile application size could be varying as it depends on the mobile OS and the dependencies size being used for the time being.

### 2.1.7 Operations

Smart Fridge systems must be easy for all users to use, e.g. no specific information or skills (except knowledge on how to use a smart device) must be required to use the tool.

## 2.2. Product functions

- The main function of the Smart Fridge system is to manage food inventory and food consumption in an online manner.
- For family members, Smart Fridge system helps them to manage food inventory, as well as their food consumption through the following main functions:
- Create and manage their own profiles. Each profile contains personal information such as ages, dietary preferences, food allergies, personal calories goal per day, etc.
- View and request to add products into the shopping lists.
- Manage and track their individual food consumption and nutrition against their target.
- View various statistics and reports about their food consumption.

## 2.3. User characteristics

The users of Smart Fridge system include system administrators, family administrators, and family members.

- System administrators have strong knowledge on networks and web applications to be able to install and maintain Smart Fridge systems.
- Family administrators are familiar with the traits and demands of their family members, as well as a basic understanding of the fridge and smart devices with which they operate.
- Family members are persons who understand enough about the usage of the Internet and smart devices to use the system.

## 2.4. Constraints

- The system must follow and satisfy these following constraints:
- Authentication security: the system should ensure user authentication security.
- Access control: The system should provide appropriate access rights and user interfaces for each kind of user (for example, system administrators and family administrators can view and delete family members profiles while family members themselves cannot view and update other member profiles).
- Backup and recovery: Easy restore and backup of system databases is crucial to avoid corruption and loss hazards.
- The system must be developed within eleven weeks and must be released by the end of May 2024.

## 2.5. Assumptions and dependencies

The following assumptions and dependencies for the systems are stated:

- All potential users of Smart Fridge must have valid information (email, DOB, name).

- All potential users of Smart Fridge must be familiar with technology devices.
- Smart Fridge must have reliable internet service and proper network infrastructure.
- Smart Fridge must have access to electrical outlets and reliable electricity, as well as backup power options in case of outages.

### 3. Specific Requirements

#### 3.1. Functional Requirements

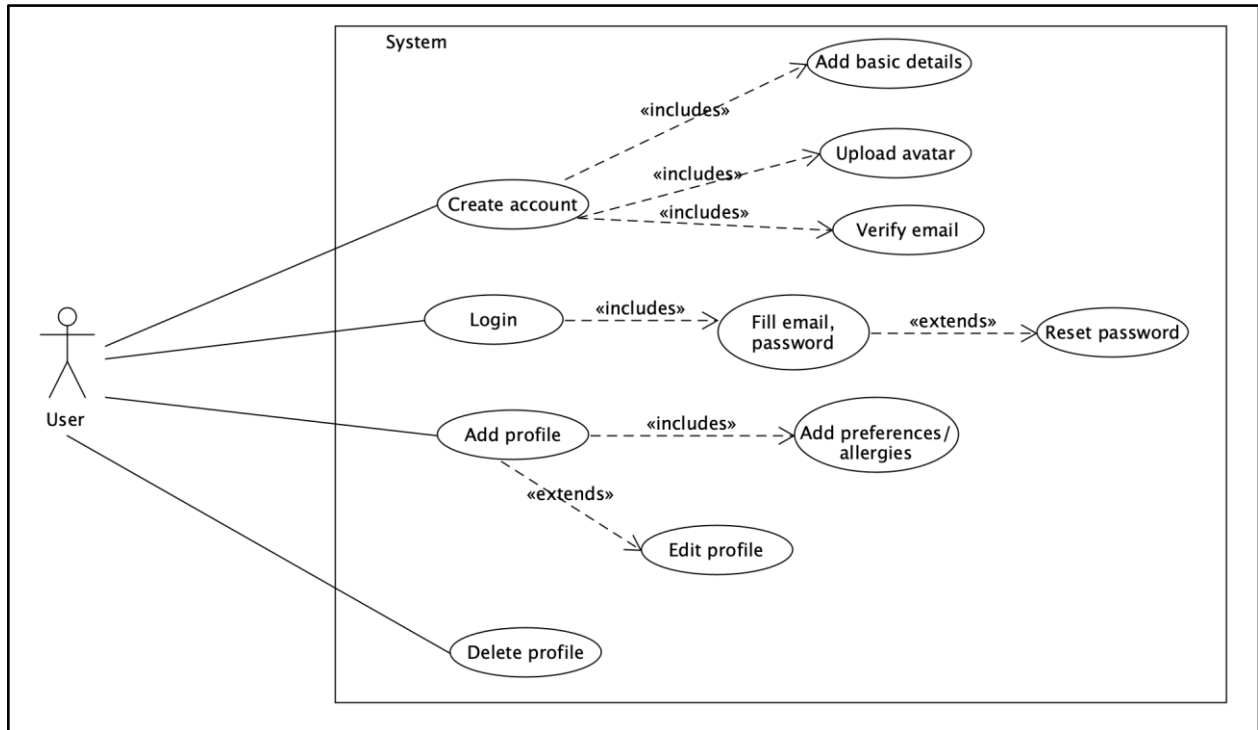
##### 3.1.1 Users Management Subsystem

- This section captures functionalities that Smart Fridge system provides to system admin, family admin and family members to manage user profiles, including adding, editing, deleting, searching and generating reports of user profiles.

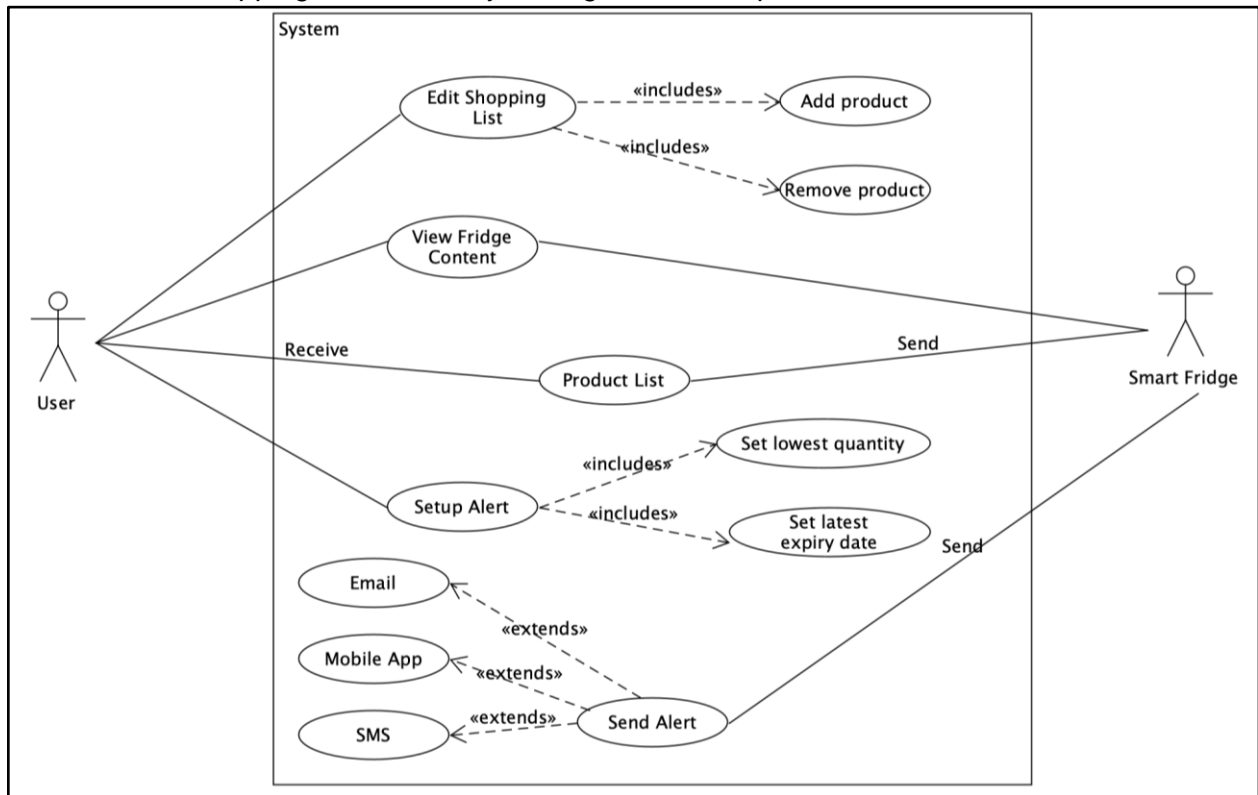
##### 3.1.2 Administrator Side

- This section includes all functions that system admin can use to manage the configuration of system, user profile and status of each user account.

## Use case #1: Create account, Login and Add profile

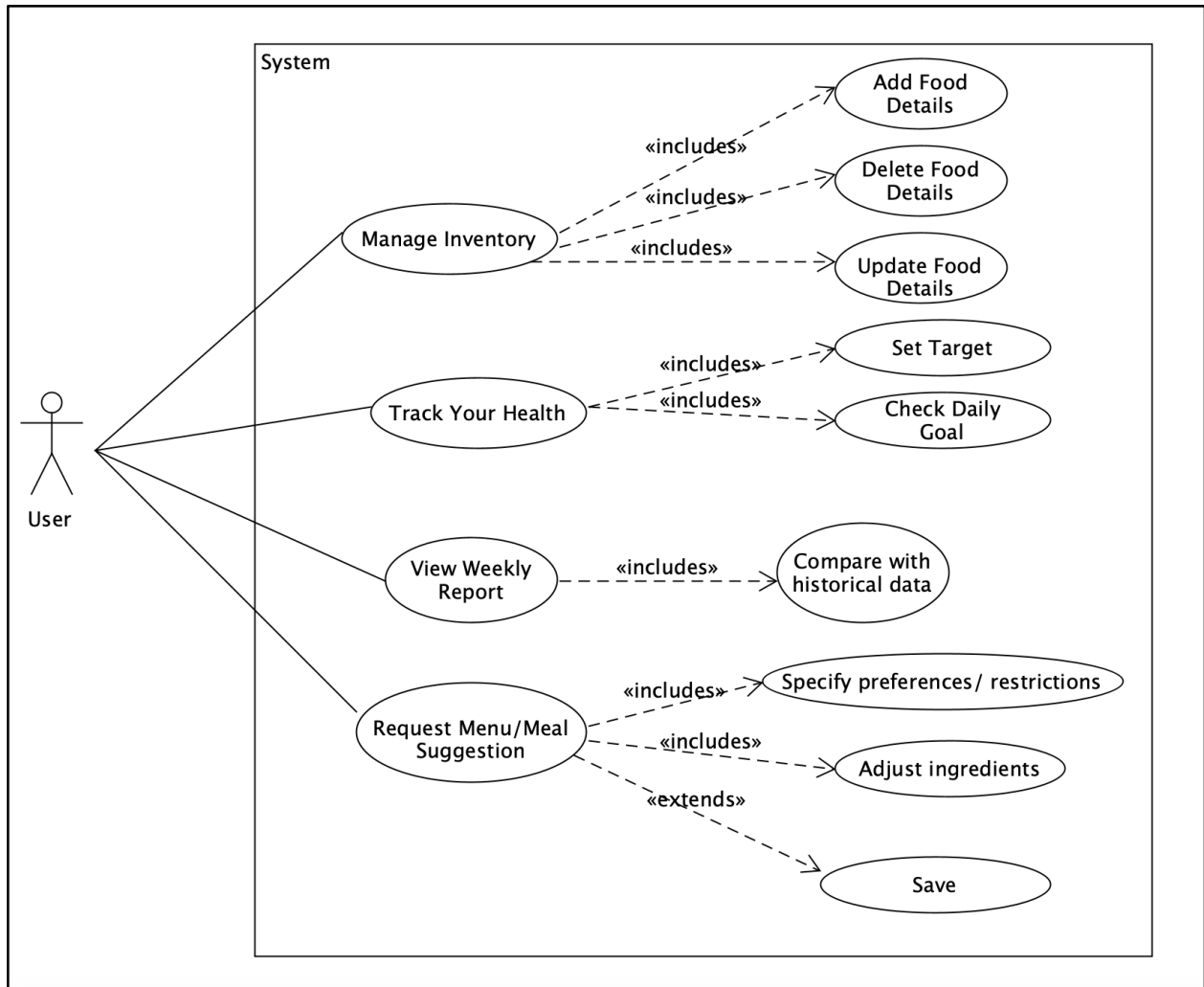


## Use case #2: Shopping List, Inventory management, Setup Alarm



## Use case #3: Track food Consumption, Generate Report, Meal Suggestion





Description: Basic User Creation and Updates
Use case 1: Create Account / Profile
Actors: User and system handler
Goals: <ul style="list-style-type: none"> <li>- Users want to create an account.</li> <li>- System will approve and save to the database</li> </ul>
Pre-condition : User has access to the app and has valid info (email, dob, name, picture, preferences, food, allergies).
Post-condition: User account is created.
Basic Flow: <ul style="list-style-type: none"> <li>- Users navigate to the app.</li> <li>- Users click on the "Create Account" button.</li> <li>- Users fill in necessary details.</li> <li>- Users submit the information.</li> <li>- System handler updates the database with user details.</li> </ul>
Alternative Flow <ol style="list-style-type: none"> <li>1. User encounters an error during submission: <ul style="list-style-type: none"> <li>- System handler attempts to update the database with user details.</li> <li>- The system detects an error in the submitted information (e.g., invalid email, dob format, missing required fields).</li> <li>- System displays an error message indicating the issue(s) encountered.</li> <li>- User corrects the errors in the form fields.</li> <li>- User resubmits the corrected information.</li> <li>- System handler updates the database with the corrected user details.</li> </ul> </li> <li>2. User attempts to create an account with existing credentials: <ul style="list-style-type: none"> <li>- System handler attempts to update the database with user details.</li> <li>- The system detects that the provided username already exists in the database.</li> <li>- System displays an error message informing the user that the username is already in use.</li> <li>- User modifies the email or username to make it unique.</li> <li>- User resubmits the corrected information.</li> <li>- System handler updates the database with the modified user details.</li> </ul> </li> </ol>

Description: Basic User Creation and Updates
Use case 2: Add/remove members
Actors: User and system handler
Goals: <ul style="list-style-type: none"> <li>- Users want to add/remove a member.</li> <li>- System will approve and save to the database.</li> </ul>
Pre-condition: User is logged in and has permission to manage members.
Post-condition: Members are added or removed and associated data is updated.
Basic Flow: <ul style="list-style-type: none"> <li>- Users have already logged in and then accessed the member section.</li> <li>- Users add/ remove member.</li> <li>- System handlers validate user inputs and store info securely</li> </ul>
Alternative Flow: <ol style="list-style-type: none"> <li>1. User encounters an error during member addition/removal: <ul style="list-style-type: none"> <li>- System handler validates user inputs and member details.</li> <li>- The system detects an error in the submitted information (e.g., invalid member information, insufficient permissions).</li> <li>- System displays an error message indicating the issue(s) encountered.</li> <li>- User corrects the errors in the form fields or adjusts permissions.</li> <li>- User resubmits the corrected information.</li> <li>- System handler updates the database with the corrected member details or permissions.</li> </ul> </li> <li>2. System encounters an error during database update: <ul style="list-style-type: none"> <li>- System handler validates user inputs and member details.</li> <li>- System attempts to update the database with the new member information or removal request.</li> <li>- The system encounters a technical issue (e.g., database connection failure, server error).</li> <li>- System displays a message informing the user that the member addition/removal process cannot be completed now due to a technical issue.</li> <li>- Users are prompted to try again later or contact customer support for assistance.</li> </ul> </li> <li>3. User cancels the member addition/removal process: <ul style="list-style-type: none"> <li>- User is logged in and accesses the member section.</li> <li>- User starts to add or remove a member but decides to cancel the action.</li> <li>- User clicks on a "Cancel" or "Back" button.</li> <li>- System handler cancels the current operation.</li> <li>- Users are redirected back to the main member management interface without any</li> </ul> </li> </ol>

changes being made to the members.
Description: Basic User Creation and Updates
Use case 3: Make changes about preferences and allergies
Actors: User and system handler
Goals: <ul style="list-style-type: none"> <li>- User defined preferences when they created their account but they change their mind and want to make changes.</li> <li>- System will approve and save to the database.</li> </ul>
Pre-condition: User is logged in and has access to the preferences section.
Post-condition: Food/drinks preferences and allergies are updated for the user.
Basic Flow: <ul style="list-style-type: none"> <li>- User is logged in and accesses their profile.</li> <li>- User navigates to the preferences section.</li> <li>- Users attempt to add, remove, or modify preferences or allergies.</li> <li>- User input changes to preferences or allergies.</li> <li>- User saves the changes.</li> <li>- System handler validates user inputs and changes.</li> </ul>
Alternative Flow: <ol style="list-style-type: none"> <li>1. User encounters an error during preference or allergy modification: <ul style="list-style-type: none"> <li>- System handler validates user inputs and changes.</li> <li>- The system detects an error in the submitted information (e.g., invalid input format, conflicting preferences).</li> <li>- System displays an error message indicating the issue(s) encountered.</li> <li>- User corrects the errors in the form fields or adjusts preferences/allergies.</li> <li>- User resubmits the corrected information.</li> <li>- System handler updates the database with the corrected user preferences and allergies.</li> </ul> </li> <li>2. System encounters an error during database update: <ul style="list-style-type: none"> <li>- System handler validates user inputs and changes.</li> <li>- System attempts to update the database with the modified preferences and allergies.</li> <li>- The system encounters a technical issue (e.g., database connection failure, server error).</li> <li>- System displays a message informing the user that the preference or allergy changes cannot be saved at the moment due to a technical issue.</li> <li>- Users are prompted to try again later or contact customer support for assistance.</li> </ul> </li> </ol>

3. User cancels the preference or allergy modification process:
  - User is logged in and accesses their profile.
  - User navigates to the preferences section.
  - User starts to add, remove, or modify preferences or allergies but decides to cancel the action.
  - User clicks on a "Cancel" or "Back" button.
  - System handler cancels the current operation.
  - User is redirected back to the main profile interface without any changes being made to the preferences or allergies.

Description: Login
Use case 4: User can login in their accounts
Actors: Actors: User, System
Goals: <ul style="list-style-type: none"> <li>- User wants to access their account within the system by providing valid credentials.</li> <li>- System verifies the user's credentials and grants access to the account if they are valid.</li> </ul>
Pre-condition: User is registered with the system and has valid login credentials.
Post-condition: User successfully gains access to their account.
Basic Flow: <ul style="list-style-type: none"> <li>- User navigates to the login page or interface within the system.</li> <li>- User enters their username or email address and password in the designated input fields.</li> <li>- User submits the login form by clicking on the "Login" button.</li> <li>- System validates the entered credentials against the records stored in the database.</li> <li>- If the entered credentials match a registered user's record: <ul style="list-style-type: none"> <li>o System authenticates the user's identity and grants access to their account.</li> <li>o User is redirected to the main dashboard or home page of their account.</li> </ul> </li> <li>- If the entered credentials do not match any registered user's record: <ul style="list-style-type: none"> <li>o System displays an error message indicating that the login credentials are invalid.</li> <li>o User is prompted to re-enter their credentials or reset their password if needed.</li> </ul> </li> </ul>
Alternative Flow: <ol style="list-style-type: none"> <li>1. Forgot Password: This will follow reset password use case (4.1)</li> </ol>

Description: Basic User Creation and Updates
Use case 4.1: Reset Password
Actors: User and system handler
Goals: <ul style="list-style-type: none"> <li>- User wants to reset their password after forgetting it.</li> <li>- System facilitates the password reset process securely.</li> </ul>
Pre-condition: User has an account registered with the system.
Post-condition: User successfully resets their password and gains access to their account.
Basic Flow: <ul style="list-style-type: none"> <li>- User navigates to the login page of the system.</li> <li>- User clicks on the "Forgot Password" link/button.</li> <li>- System redirects the user to the password reset page/form.</li> <li>- User enters their email address associated with the account.</li> <li>- User submits the email address.</li> <li>- System verifies the email address and sends a password code to the provided email address.</li> <li>- User checks their email inbox, including spam/junk folders, for the password reset email.</li> <li>- User enters the password reset code into the system.</li> <li>- System validates the reset code and directs the user to a page/form to enter a new password.</li> <li>- User enters a new password and confirms it.</li> <li>- User submits the new password.</li> <li>- System updates the user's password in the database.</li> <li>- System confirms a successful password reset to the user.</li> <li>- Users can now log in using the new password.</li> </ul>
Alternative Flow: <ol style="list-style-type: none"> <li>Invalid Email Address: <ul style="list-style-type: none"> <li>- If the email address entered by the user is invalid (e.g., incorrect format, not registered with the system): <ul style="list-style-type: none"> <li>o System displays an error message indicating that the email address is invalid.</li> <li>o User is prompted to re-enter a valid email address.</li> <li>o User follows the steps from step 4 of the basic flow.</li> </ul> </li> </ul> </li> <li>Email Not Received: <ul style="list-style-type: none"> <li>- If the user does not receive the password reset email within a reasonable amount of</li> </ul> </li> </ol>

time:

- User checks their spam/junk folder.
- If the email is not found:
  - User clicks on the "Resend Email" option provided on the password reset page.
  - System resends the password reset email.
  - User checks their email again.
- If the email is still not received:
  - User contacts customer support for assistance.

3. Expired Reset Link:

- If the user clicks on the password reset link after it has expired (typically due to a time limit set for security reasons):
  - System displays an error message indicating that the password reset link has expired.
  - User is prompted to initiate the password reset process again.



Description: Shopping list and fridge interaction
Use case 5: Add Product to Shopping List
Actors: User and system handler
Goals: <ul style="list-style-type: none"> <li>- Users want to add a product to their shopping list with preferred quantity and purchase frequency.</li> <li>- System allows the user to specify the preferred amount and whether it's a one-off or recurring purchase.</li> </ul>
Pre-condition: User is logged in and has access to their shopping list feature.
Post-condition: Product is successfully added to the shopping list with specified details.
Basic Flow: <ul style="list-style-type: none"> <li>- User opens the shopping list feature within the system.</li> <li>- User clicks on the "Add Product" button/icon.</li> <li>- System presents a form for adding a new product to the list.</li> <li>- User enters the name of the product.</li> <li>- User specifies the preferred quantity/amount for the product and manually type in the expiry date.</li> <li>- User selects whether it's a one-off purchase or recurring purchase.</li> <li>- User submits the form to add the product to the list.</li> <li>- System validates the input data.</li> <li>- System adds the product to the shopping list with the specified details.</li> <li>- System confirms successful addition of the product to the user.</li> </ul>
Alternative Flow: <ol style="list-style-type: none"> <li>Invalid Quantity/Amount: <ul style="list-style-type: none"> <li>- If the user enters an invalid quantity/amount (e.g., negative value, non-numeric input): <ul style="list-style-type: none"> <li>o System displays an error message indicating that the entered quantity/amount is invalid.</li> <li>o User corrects the input and resubmits the form.</li> </ul> </li> </ul> </li> <li>Invalid Purchase Frequency: <ul style="list-style-type: none"> <li>- If the user selects an invalid purchase frequency (e.g., neither one-off nor recurring): <ul style="list-style-type: none"> <li>o System displays an error message indicating that the selected purchase frequency is invalid.</li> <li>o User corrects the selection and resubmits the form.</li> </ul> </li> </ul> </li> <li>Cancel Adding Product: <ul style="list-style-type: none"> <li>- If the user decides to cancel adding the product to the shopping list: <ul style="list-style-type: none"> <li>o User clicks on a "Cancel" or "Back" button.</li> <li>o System cancels the operation and returns the user to the main shopping list</li> </ul> </li> </ul> </li> </ol>

interface without adding the product.

4. Duplicate Product Entry:

- If the user attempts to add a product that already exists in the shopping list:
  - System displays a warning message indicating that the product already exists in the list.
  - User can choose to update the quantity/amount or purchase frequency of the existing entry or cancel the operation.

Description: Shopping list and fridge interaction
Use case 6: Update Product in Shopping List
Actors: User and system handler
Goals: <ul style="list-style-type: none"> <li>- Users want to update the details of a product already present in their shopping list.</li> <li>- System allows the user to modify the quantity/amount or purchase frequency of the product.</li> </ul>
Pre-condition: User is logged in and has access to their shopping list feature.
Post-condition: Product details in the shopping list are successfully updated.
Basic Flow: <ul style="list-style-type: none"> <li>- User opens the shopping list feature within the system.</li> <li>- User selects the product they wish to update from the list.</li> <li>- User clicks on the "Edit" or "Update" button/icon next to the selected product.</li> <li>- System displays a form with the current details of the selected product.</li> <li>- User modifies the quantity/amount or purchase frequency of the product as desired.</li> <li>- User submits the form to update the product details.</li> <li>- System validates the input data.</li> <li>- System updates the product details in the shopping list.</li> <li>- System confirms successful updates of the product to the user.</li> </ul>
Alternative Flow: <ol style="list-style-type: none"> <li>1. Cancel Product Update: <ul style="list-style-type: none"> <li>- If the user decides to cancel updating the product: <ul style="list-style-type: none"> <li>o User clicks on a "Cancel" or "Back" button.</li> <li>o System cancels the operation and returns the user to the main shopping list interface without making any changes to the product.</li> </ul> </li> </ul> </li> <li>2. Invalid Quantity/Amount: <ul style="list-style-type: none"> <li>- If the user enters an invalid quantity/amount (e.g., negative value, non-numeric input): <ul style="list-style-type: none"> <li>o System displays an error message indicating that the entered quantity/amount is invalid.</li> <li>o User corrects the input and resubmits the form.</li> </ul> </li> </ul> </li> </ol>

Description: Shopping list and fridge interaction
Use case 7: Remove Product from Shopping List
Actors: User and system handler
Goals: <ul style="list-style-type: none"> <li>- Users want to remove a product from their shopping list.</li> <li>- System allows the user to delete the selected product from the list.</li> </ul>
Pre-condition: User is logged in and has access to their shopping list feature.
Post-condition: Product is successfully removed from the shopping list.
Basic Flow: <ul style="list-style-type: none"> <li>- User opens the shopping list feature within the system.</li> <li>- User selects the product they wish to remove from the list.</li> <li>- User clicks on the "Remove" or "Delete" button/icon next to the selected product.</li> <li>- System displays a confirmation prompt asking the user to confirm the removal.</li> <li>- User confirms the removal of the product.</li> <li>- System removes the selected product from the shopping list.</li> <li>- System confirms successful removal of the product to the user.</li> </ul>
Alternative Flow: <ol style="list-style-type: none"> <li>1. Cancel Product Removal: <ul style="list-style-type: none"> <li>- If the user decides to cancel removing the product: <ul style="list-style-type: none"> <li>o User cancels the confirmation prompt.</li> <li>o System cancels the operation and returns the user to the main shopping list interface without making any changes to the list.</li> </ul> </li> </ul> </li> </ol>

Description: Shopping list and fridge interaction
Use case 8: View Fridge/Inventory Contents
Actors: User and system handler
Goals: <ul style="list-style-type: none"> <li>- User wants to view the contents of their fridge or inventory.</li> <li>- System provides the user with a list of items currently stored in the fridge or inventory.</li> </ul>
Pre-condition: User is logged in and has access to the fridge/inventory feature.
Post-condition: User successfully views the contents of the fridge or inventory.
Basic Flow: <ul style="list-style-type: none"> <li>- User opens the fridge/inventory feature within the system.</li> <li>- System retrieves the list of items stored in the fridge or inventory.</li> <li>- System displays the list of items to the user, organized in a user-friendly format (e.g., categorized, alphabetized).</li> <li>- Users can scroll through the list to view all the items present.</li> <li>- Users can search for specific items using a search bar or filter options provided by the system.</li> <li>- Users can click on individual items to view additional details such as quantity, expiration date, or notes associated with the item.</li> <li>- Users can navigate back to the main fridge/inventory interface after viewing the contents.</li> </ul>
Alternative Flow: <ol style="list-style-type: none"> <li>1. Empty Fridge/Inventory: <ul style="list-style-type: none"> <li>- If the fridge or inventory is empty: <ul style="list-style-type: none"> <li>o System displays a message indicating that the fridge or inventory is currently empty.</li> <li>o User can choose to add items to the fridge/inventory or return to other features of the system.</li> </ul> </li> </ul> </li> <li>2. Error Retrieving Contents: <ul style="list-style-type: none"> <li>- If the system encounters an error while retrieving the contents of the fridge/inventory: <ul style="list-style-type: none"> <li>o System displays an error message indicating that there was a problem retrieving the contents.</li> <li>o User can choose to retry viewing the contents or contact customer support for assistance.</li> </ul> </li> </ul> </li> <li>3. No Search Results: <ul style="list-style-type: none"> <li>- If the user's search query returns no results:</li> </ul> </li> </ol>

- System displays a message indicating that no items matching the search criteria were found.
- User can refine their search query or clear the search to view all items again.

4. View Item Details:

- If the user clicks on an individual item to view details:
  - System displays additional information about the selected item, such as quantity, expiration date, or notes.
  - User can close the item details view to return to the list of contents.

Use case 8.1 : Add Product to Inventory
Actors: User, System
Goals: <ul style="list-style-type: none"> <li>- User wants to add a new product to the inventory.</li> <li>- System allows the user to input details of the new product and stores it in the inventory database.</li> </ul>
Pre-condition: User is logged in and has access to the inventory management system.
Post-condition: Product is successfully added to the inventory.
Basic Flow: <ul style="list-style-type: none"> <li>- User navigates to the inventory management section within the system.</li> <li>- User selects the option to add a new product to the inventory.</li> <li>- System presents a form for the user to input details of the new product, including: <ul style="list-style-type: none"> <li>- Product name</li> <li>- Product category <ul style="list-style-type: none"> <li>o Quantity</li> <li>o Description (optional)</li> <li>o Supplier information (optional)</li> <li>o Unit price</li> <li>o Expiry date (if applicable)</li> </ul> </li> </ul> </li> <li>- User fills in the required fields with accurate information about the new product.</li> <li>- User submits the form to add the new product to the inventory.</li> <li>- System validates the input data to ensure completeness and accuracy.</li> <li>- System updates the inventory database with the information provided by the user.</li> <li>- System confirms successful addition of the new product to the inventory to the user.</li> </ul>
Alternative Flow: <ol style="list-style-type: none"> <li>1. Invalid Input Data: <ul style="list-style-type: none"> <li>- If the user enters invalid or incomplete data in the form: <ul style="list-style-type: none"> <li>o System displays error messages indicating the fields that need to be corrected.</li> <li>o User corrects the input data and resubmits the form.</li> </ul> </li> </ul> </li> <li>2. Duplicate Product Entry: <ul style="list-style-type: none"> <li>- If the user attempts to add a product that already exists in the inventory: <ul style="list-style-type: none"> <li>o System detects the duplicate entry and alerts the user.</li> <li>o System suggests options such as updating the quantity or modifying the existing entry if necessary.</li> <li>o User confirms whether to proceed with adding the duplicate product or makes changes accordingly.</li> </ul> </li> </ul> </li> </ol>

Use case 8.2 : Remove Product from Inventory
Actors: User, System
Goals: <ul style="list-style-type: none"> <li>- User wants to remove a product from the inventory.</li> <li>- System allows the user to select and remove the desired product from the inventory database.</li> </ul>
Pre-condition: User is logged in and has access to the inventory management system.
Post-condition: Product is successfully removed from the inventory.
Basic Flow: <ul style="list-style-type: none"> <li>- User navigates to the inventory management section within the system.</li> <li>- User selects the option to view the list of products in the inventory.</li> <li>- System displays the list of products currently stored in the inventory.</li> <li>- User identifies the product they want to remove from the inventory.</li> <li>- User selects the option to remove the identified product.</li> <li>- System prompts the user to confirm the removal of the product.</li> <li>- User confirms the removal of the product.</li> <li>- System updates the inventory database to remove the selected product.</li> <li>- System confirms successful removal of the product to the user.</li> </ul>
Alternative Flow: <p>Alternative Flow</p> <ol style="list-style-type: none"> <li>1. Undo Removal: <ul style="list-style-type: none"> <li>- If the user accidentally removes a product or decides to undo the removal: <ul style="list-style-type: none"> <li>o User requests to undo the removal action immediately after confirming it.</li> <li>o System restores the removed product back to the inventory database.</li> <li>o System confirms the successful restoration of the product to the user.</li> </ul> </li> </ul> </li> </ol>



Description: Shopping list and fridge interaction
Use case 9: Sending Alerts for Low Quantity or Near Expiry Date
Actors: User and system handler
Goals: <ul style="list-style-type: none"> <li>- Users want to receive alerts when items in their fridge or inventory reach a low quantity or are nearing their expiry date.</li> <li>- System monitors the quantity and expiry dates of items and sends alerts to the user as needed.</li> </ul>
Pre-condition: User is logged in and has access to the fridge/inventory feature.
Post-condition: User successfully receives alerts for low quantity or near expiry date items.
Basic Flow: <ul style="list-style-type: none"> <li>- Users set up alert preferences within the system for low quantity and expiry date alerts.</li> <li>- System regularly monitors the quantity and expiry dates of items stored in the fridge or inventory.</li> <li>- When an item's quantity falls below the user-defined threshold or its expiry date approaches, the system triggers an alert.</li> <li>- System sends an alert notification to the user via their preferred communication channel (e.g., email, push notification, SMS).</li> <li>- User receives the alert and views the details of the low quantity or near expiry date item.</li> <li>- Users can take appropriate action, such as restocking the item or using it before it expires.</li> <li>- Users acknowledge the alert, marking it as read or dismissing it from their notification list.</li> </ul>
Alternative Flow: <ol style="list-style-type: none"> <li>1. User Adjusts Alert Settings: <ul style="list-style-type: none"> <li>- If the user wants to adjust their alert preferences (e.g., change the low quantity threshold, set a different lead time for expiry date alerts): <ul style="list-style-type: none"> <li>o User navigates to the alert settings section within the system.</li> <li>o User modifies the alert preferences according to their requirements.</li> <li>o System updates the user's alert settings accordingly.</li> </ul> </li> </ul> </li> <li>2. System Error Sending Alerts: <ul style="list-style-type: none"> <li>- If the system encounters an error while sending alerts: <ul style="list-style-type: none"> <li>o System logs the error for investigation.</li> <li>o System retries sending the alert at a later time.</li> </ul> </li> <li>- If the alert cannot be sent successfully after multiple attempts: <ul style="list-style-type: none"> <li>o System displays a message indicating the issue and advises the user to check</li> </ul> </li> </ul> </li> </ol>

their items manually for low quantity or expiry date.

3. User Manually Checks Items:

- If the user prefers to manually check their items instead of relying solely on alerts:
  - User navigates to the fridge/inventory feature within the system.
  - User reviews the list of items stored, paying attention to quantities and expiry dates.
  - User takes necessary actions based on their observations, such as restocking or discarding expired items.

Description: Shopping list and fridge interaction
Use case 10: Manage and Track Individual's Food Consumption/Nutrition
Actors: User and system handler
Goals: <ul style="list-style-type: none"> <li>- Users want to track their food consumption and nutrition intake for health and wellness purposes.</li> <li>- System provides tools for the user to input, manage, and track their food consumption and nutrition data.</li> </ul>
Pre-condition: User is logged in and has access to the nutrition tracking feature.
Post-condition: User successfully manages and tracks their food consumption and nutrition data.
Basic Flow <ul style="list-style-type: none"> <li>- User navigates to the nutrition tracking feature within the system.</li> <li>- User selects the option to add a new food item to their consumption log.</li> <li>- System presents a form for the user to input details of the consumed food item, including name, quantity, and meal category (e.g., breakfast, lunch, dinner, snacks).</li> <li>- User fills in the required fields with accurate information about the consumed food item.</li> <li>- User submits the form to add the food item to their consumption log.</li> <li>- System validates the input data and updates the user's consumption log with the new entry.</li> <li>- Users can view their consumption log to see a summary of their food intake for the day, week, or custom date range.</li> <li>- Users can view nutritional information associated with each consumed food item, such as calories, macronutrients (carbohydrates, proteins, fats), vitamins, and minerals.</li> <li>- Users can edit or delete individual entries in their consumption log if needed (e.g., to correct errors or remove duplicates).</li> <li>- Users can set goals or targets for their daily nutritional intake (e.g., target calories, target protein intake).</li> <li>- System provides feedback to the user on their progress towards their nutritional goals based on their consumption log data.</li> </ul>
Alternative Flow: <ol style="list-style-type: none"> <li>1. Duplicate Entry: <ul style="list-style-type: none"> <li>- If the user accidentally adds a duplicate entry to their consumption log: <ul style="list-style-type: none"> <li>o System detects the duplicate entry and alerts the user.</li> <li>o User confirms whether they want to keep both entries, merge them, or delete one of them.</li> </ul> </li> </ul> </li> <li>2. Missing Information: <ul style="list-style-type: none"> <li>- If the user forgets to input required information for a food item (e.g., name, quantity):</li> </ul> </li> </ol>

- System prompts the user to fill in all required fields before submitting the form.
3. Unrecognized Food Item:
- If the user consumes a food item not present in the system's database:
    - User can manually input the nutritional information for the custom food item.
    - System saves the custom food item to the user's consumption log for future reference.

Description: Shopping list and fridge interaction
Use case 11: Reports Generating
Actors: User and system handler
Goals: <ul style="list-style-type: none"> <li>- Users want to generate reports and statistics to gain insights into their food consumption and nutrition intake.</li> <li>- System provides tools for generating comprehensive reports and statistics based on the user's consumption log data.</li> </ul>
Pre-condition: : User is logged in and has inputted sufficient data into their consumption log.
Post-condition: User successfully generates and accesses reports and statistics on their food consumption and nutrition intake.
Basic Flow: <ul style="list-style-type: none"> <li>- Users navigate to the reporting/statistics feature within the system.</li> <li>- Users select the date range for the report or statistic they want to generate (e.g., daily summary, nutrient breakdown).</li> <li>- System retrieves the relevant data from the user's consumption log based on the selected parameters.</li> <li>- System generates the requested report or statistic based on the retrieved data.</li> <li>- System presents the report or statistic to the user in a user-friendly format, such as a chart, graph, or table.</li> <li>- Users review the generated report or statistic to gain insights into their food consumption and nutrition intake.</li> <li>- Users can customize the report or statistic by adjusting parameters such as date range, meal category, or nutrient focus.</li> <li>- Users can export the generated report or statistic in various formats (e.g., PDF, Excel) for further analysis or sharing.</li> <li>- Users can compare current reports and statistics with historical data to track progress towards their nutritional goals over time.</li> <li>- Users can set up automated report generation and delivery preferences (e.g., receive a weekly summary report via email).</li> </ul>
Alternative Flow: <ol style="list-style-type: none"> <li>Insufficient Data: <ul style="list-style-type: none"> <li>- If the user does not have enough data in their consumption log to generate meaningful reports or statistics: <ul style="list-style-type: none"> <li>o System notifies the user that there is insufficient data available for the requested report.</li> <li>o System suggests actions the user can take to input more data into their consumption log (e.g., track meals consistently, input more detailed information).</li> </ul> </li> </ul> </li> </ol>

- User acknowledges the notification and takes necessary actions to input more data.
- 2. Custom Report/Statistic:
  - If the user wants to generate a custom report or statistic with specific parameters:
    - User navigates to the custom report/statistic section within the system.
    - User selects the desired parameters and criteria for the custom report/statistic.
    - System generates the custom report/statistic based on the user's specifications.
    - System presents the custom report/statistic to the user for review and analysis.
    - Data Filtering and Sorting:
  - If the user wants to filter or sort the data in the generated report/statistic:
    - User utilizes filtering and sorting options provided by the system to refine the displayed data according to their preferences.
    - System updates the report/statistic based on the user's selected filters and sorting criteria.
- 4. Feedback and Suggestions:
  - If the user wants to provide feedback or suggestions for improving the reporting/statistics feature:
    - User submits feedback through a dedicated feedback form or feature within the system.
    - System collects and analyzes user feedback to identify areas for improvement and future enhancements to the reporting/statistics feature.

Description: Shopping list and fridge interaction
Use case 12: Suggesting meal plans and recipes
Actors: User and system handler
Goals: <ul style="list-style-type: none"> <li>- Users want suggestions for meals and new dishes based on their food preferences, dietary restrictions, and nutritional goals.</li> <li>- System provides personalized meal suggestions to the user to help them plan their meals effectively.</li> </ul>
Pre-condition: User is logged in and has provided information about their food preferences, dietary restrictions, and nutritional goals.
Post-condition: User receives personalized meal suggestions from the system.
Basic Flow: <ul style="list-style-type: none"> <li>- User navigates to the meal suggestion feature within the system.</li> <li>- User specifies any preferences or criteria for the suggested meals, such as cuisine type, dietary restrictions (e.g., vegetarian, gluten-free), or desired nutritional profile (e.g., low-carb, high-protein).</li> <li>- System retrieves user profile data, including food preferences, dietary restrictions, and nutritional goals.</li> <li>- System analyzes the user's profile data and generates personalized meal suggestions based on the specified criteria.</li> <li>- System presents the meal suggestions to the user in a user-friendly format, such as a list of recommended recipes or meal plans.</li> <li>- Users review the suggested meals and select one or more options that appeal to them.</li> <li>- Users can view detailed recipes and cooking instructions for the selected meals.</li> <li>- Users can add the selected meals to their meal planner or shopping list for easy reference and planning.</li> </ul>
Alternative Flow: <ol style="list-style-type: none"> <li>1. No Meal Suggestions Available: <ul style="list-style-type: none"> <li>- If the system cannot generate any meal suggestions based on the user's criteria: <ul style="list-style-type: none"> <li>o System notifies the user that no meal suggestions are available based on the specified preferences and criteria.</li> <li>o System provides suggestions for adjusting the criteria or preferences to generate more relevant meal suggestions.</li> <li>o User adjusts the criteria or preferences and retries the meal suggestion process.</li> </ul> </li> </ul> </li> <li>2. Customized Meal Plans: <ul style="list-style-type: none"> <li>- If the user wants to receive customized meal plans tailored to their specific goals (e.g., weight loss, muscle gain): <ul style="list-style-type: none"> <li>o User selects the option to receive personalized meal plans when requesting meal suggestions.</li> <li>o System generates meal plans with optimized nutritional profiles based on the user's goals and preferences.</li> </ul> </li> </ul> </li> </ol>

- System provides a variety of meal options for breakfast, lunch, dinner, and snacks, along with corresponding recipes and portion sizes.

#### Use Case 13: Change Fridge Temperature from Home Screen

Actors: User

Pre-condition: User need to be on the Home Screen of the Smart Fridge

Post-condition: Fridge temperature setting is successfully changed according to the user's preference.

##### Goals:

- User wants to change the temperature setting of the fridge directly from the home screen of the smart fridge interface.
- System allows the user to adjust the temperature setting easily and efficiently.

##### Basic Flow:

- User accesses the home screen of the smart fridge interface.
- User locates the temperature control widget displayed on the home screen.
- System displays the current temperature setting along with controls for adjusting the temperature.
- User adjusts the temperature setting using the provided controls, such as sliders or buttons.
- User confirms the new temperature setting.
- System updates the fridge's temperature control system with the new setting.
- System successful adjustment of the temperature setting to the user.

##### Benefits

- The user can easily monitor and adjust the temperature of their refrigerator.
- The user can optimize the energy efficiency of their refrigerator by setting the appropriate temperature.
- The user can help to prevent food spoilage by keeping food at the correct temperature.

##### Alternative Flow:

##### Temperature Range Limits:

- If the system enforces limits on the temperature adjustment range to prevent extreme settings:
- User attempts to set the temperature outside the allowed range.
- System displays a message indicating the allowed temperature range and prompts the user to choose a setting within that range.
- User selects a temperature setting within the allowed range.
- System proceeds with updating the temperature setting accordingly



### 3.2. Non-Functional Requirements

Requirement #: NREQ_01	Requirement Type: Performance	Use case #: NONE
Description: Ensure that the system responds to user actions within 2 seconds.		
Rationale: Users expect a responsive system that allows them to perform tasks without significant delays. A response time of 2 seconds or less enhances user satisfaction and productivity.		
Source: Stakeholder feedback, usability studies.		
Fit Criterion: Measure the response time for various user actions, such as clicking buttons or loading pages, and ensure that 90% of these actions are completed within 2 seconds.		
Dependencies: Dependent on the system architecture, network latency, and hardware capabilities.		
Rank of importance: High priority, as performance directly impacts user experience and satisfaction.		
Supporting Materials: Performance testing reports, user feedback regarding system responsiveness.		
History:  Created by Myunggyun Yu on 31/03/2024		

Requirement #: NREQ_02	Requirement Type: Usability	Use case #: NONE
Description: The system must be simple to use for all users, requiring no special knowledge or skills (other than the ability to utilize an application on a smart device)		
Rationale: The food inventory management system should be user-friendly and intuitive to ensure that all users, regardless of their technical expertise, can easily navigate and utilize its features. This requirement aims to enhance user adoption and minimize the learning curve.		
Source: User feedback, usability studies.		
Fit Criterion: Conduct user testing sessions to evaluate the ease of use of the system. Measure the time taken for users to perform common tasks and gather feedback on the system's intuitiveness.		
Dependencies: Dependent on the system's user interface design and the availability of clear instructions and guidance within the application.		
Rank of importance: High priority, as usability directly impacts user satisfaction and the system's overall effectiveness.		
Supporting Materials: Usability testing reports, user feedback on ease of use.		
History:  Created by Myunggyun Yu on 31/03/2024  Update by Ashutosh on 31/03/2024		

Requirement #: NREQ_03	Requirement Type: Security	Use case #: NONE
Description: The system must ensure user authentication security		
Rationale: To protect user data and prevent unauthorized access, it is crucial to implement robust security measures for user authentication. This requirement aims to safeguard sensitive information and maintain the privacy of users.		
Source: System security guidelines, stakeholder requirements.		
Fit Criterion: Conduct security audits and penetration testing to identify vulnerabilities and ensure that the system's authentication mechanisms are secure. Regularly update and patch any identified security flaws.		
Dependencies: Dependent on the system's authentication protocols, encryption methods, and adherence to security best practices.		
Rank of importance: High priority, as security breaches can have severe consequences for users and the system's reputation.		
Supporting Materials: Security audit reports, penetration testing results.		
History:  Created by Myunggyun Yu on 31/03/2024  Update by Ashutosh on 31/03/2024		

Requirement #: NREQ_04	Requirement Type: Reliability:	Use case #: NONE
Description: The system should be available 24/7 with minimal downtime for maintenance or updates. Data integrity should be maintained at all times, with appropriate backup and recovery mechanisms in place. Error handling should be robust, providing informative messages to users and logging errors for troubleshooting purposes		
Rationale: Continuous availability and data integrity are critical for ensuring user trust and satisfaction. Downtime or data loss can have significant business impact.		
Source: Service level agreements (SLAs), user requirements.		
Fit Criterion: Monitor system uptime and ensure that downtime does not exceed agreed-upon thresholds. Regularly test backup and recovery procedures to verify data integrity.		
Dependencies: Redundant infrastructure.		
Rank of importance: High priority		
Supporting Materials: Uptime reports, backup and recovery documentation.		
History: Created by Myunggyun Yu on 31/03/2024 Update by Ashutosh on 31/03/2024		

Requirement #: NREQ_05	Requirement Type: Scalability	Use case #: NONE
Description: The system architecture should be designed to scale horizontally and vertically to accommodate increasing data and user loads. Load balancing mechanisms should be implemented to distribute incoming requests evenly across multiple server instances.		
Rationale: As the user base and data volume grow, the system must be able to handle increased traffic without degradation in performance.		
Source: Project requirements, anticipated growth projections.		
Fit Criterion: Conduct scalability tests to ensure that the system can handle increased loads without performance degradation. Monitor system performance under varying load conditions.		
Rank of importance: Low		
Supporting Materials:		
History: Created by Myunggyun Yu on 31/03/2024 Update by Ashutosh on 31/03/2024		

Requirement #: NREQ_06	Requirement Type: Maintainability	Use case #: NONE
Description: The codebase should follow best practices and coding standards to facilitate code maintenance and future enhancements. Documentation should be comprehensive and up-to-date, including system architecture, API documentation, and user manuals. Modular design principles should be employed to allow for easy extension and modification of system components.		
Rationale: A well-maintained codebase and thorough documentation simplify ongoing development and support efforts, reducing the risk of technical debt.		
Source: Development team, coding standards.		
Fit Criterion: Conduct code reviews to ensure adherence to coding standards. Verify that documentation is comprehensive and up-to-date, covering all aspects of the system.		
Dependencies: Development processes, version control systems.		
Rank of importance: Important		
Supporting Materials: Code review reports, documentation review results.		
History: Created by Myunggyun Yu on 31/03/2024 Update by Ashutosh on 31/03/2024		

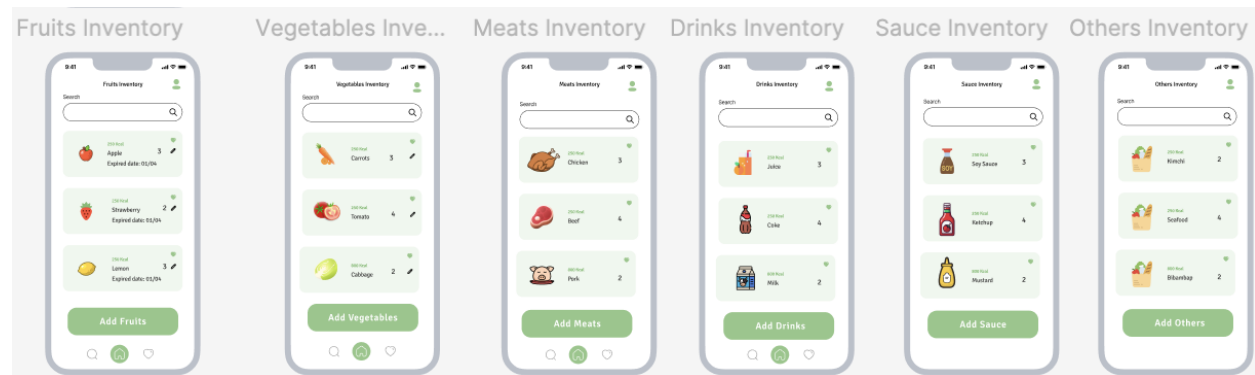
Requirement #: NREQ_07	Requirement Type: Integration	Use case #: NONE
Description: The system should support integration with external APIs, such as those provided by smart fridges, for automated data updates. Compatibility with different devices and platforms should be ensured to maximize accessibility for users.		
Rationale: Integration with external systems expands the functionality of the application and enhances user experience by automating processes and providing seamless interactions.		
Source: Project requirements, external system specifications.		
Fit Criterion: Test integration with external APIs to verify data exchange functionality. Ensure compatibility with various devices and platforms through compatibility testing.		
Dependencies: External system APIs, communication protocols.		
Rank of importance: Important		
Supporting Materials: Integration test results, compatibility test reports.		
History: Created by Myunggyun Yu on 31/03/2024 Update by Ashutosh on 31/03/2024		

Requirement #: NREQ_08	Requirement Type: Data Privacy	Use case #: NONE
Description: Personal data of users, including dietary preferences and consumption patterns, should be stored securely and handled in compliance with relevant privacy regulations (e.g., GDPR, CCPA). Data anonymization techniques should be employed where applicable to protect user privacy while still allowing for meaningful analysis and reporting.		
Rationale: Protecting user privacy is essential for maintaining trust and compliance with privacy regulations, which carry legal and reputational risks if violated.		
Source: Privacy regulations, user consent requirements.		
Fit Criterion: Conduct privacy impact assessments to identify and address privacy risks. Ensure that data storage and handling practices comply with relevant regulations.		
Dependencies: Data encryption methods, privacy compliance tools.		
Rank of importance: High priority		
Supporting Materials: Privacy impact assessment reports, compliance audit results.		
History: Created by Myunggyun Yu on 31/03/2024 Update by Ashutosh on 31/03/2024		

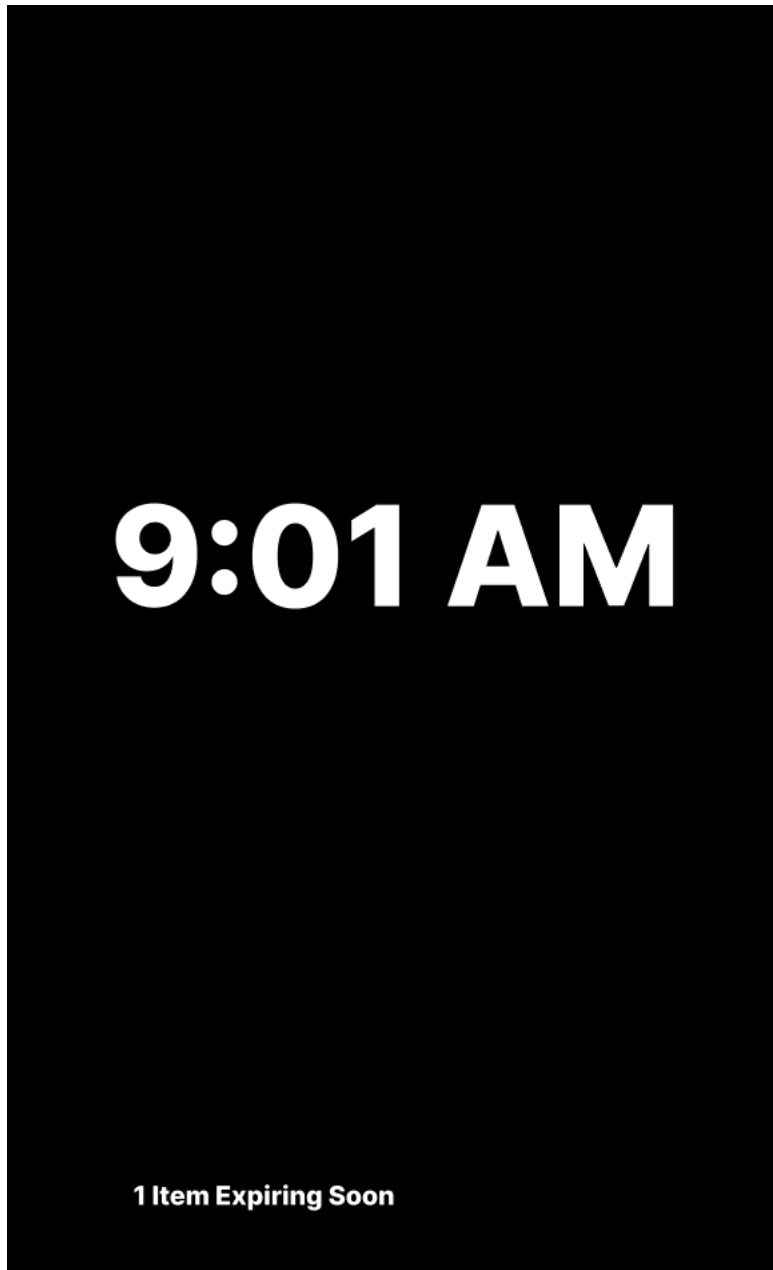


### 3.5 Figma Prototype Video (Mobile Application)

<https://drive.google.com/file/d/1MU4Ls6yHg-xGSeDh4mXzKHprh9wahU9p/view?usp=sharing>

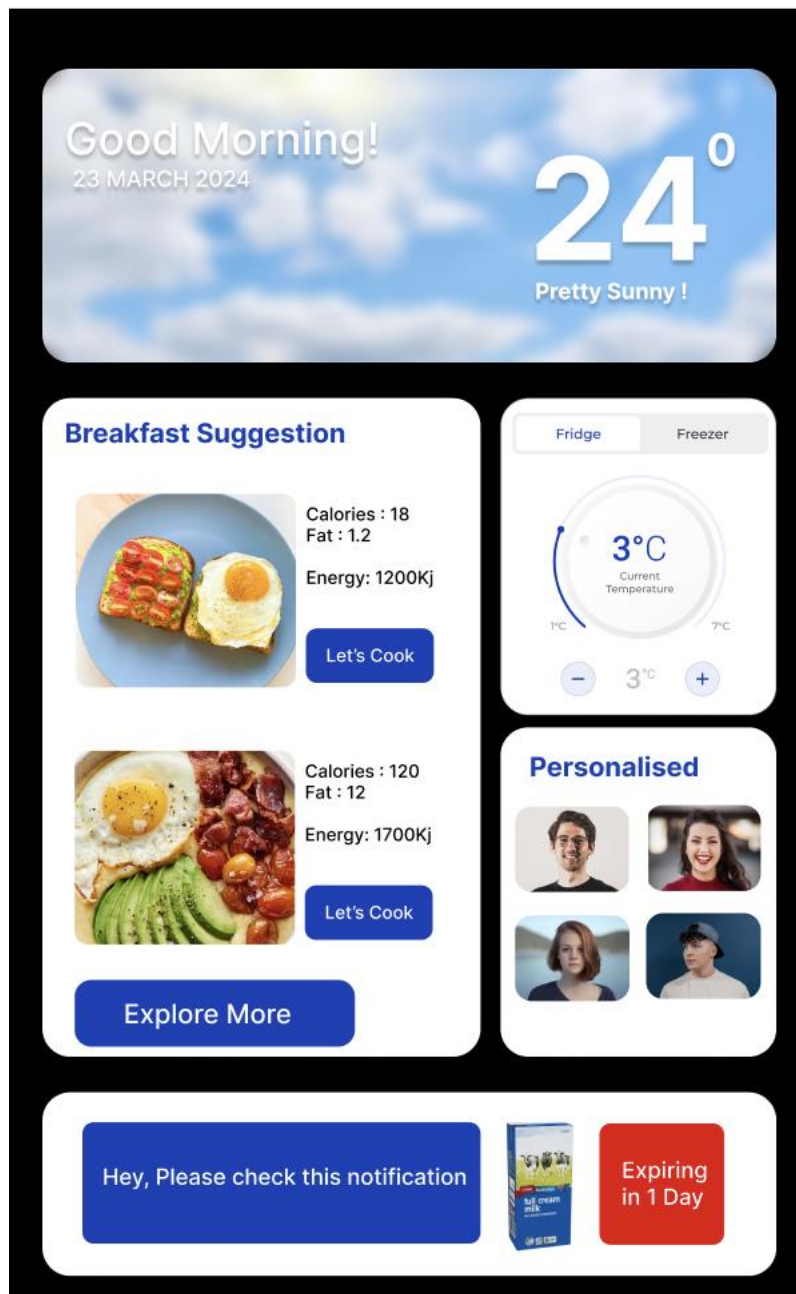


### 3.6 Fridge UI



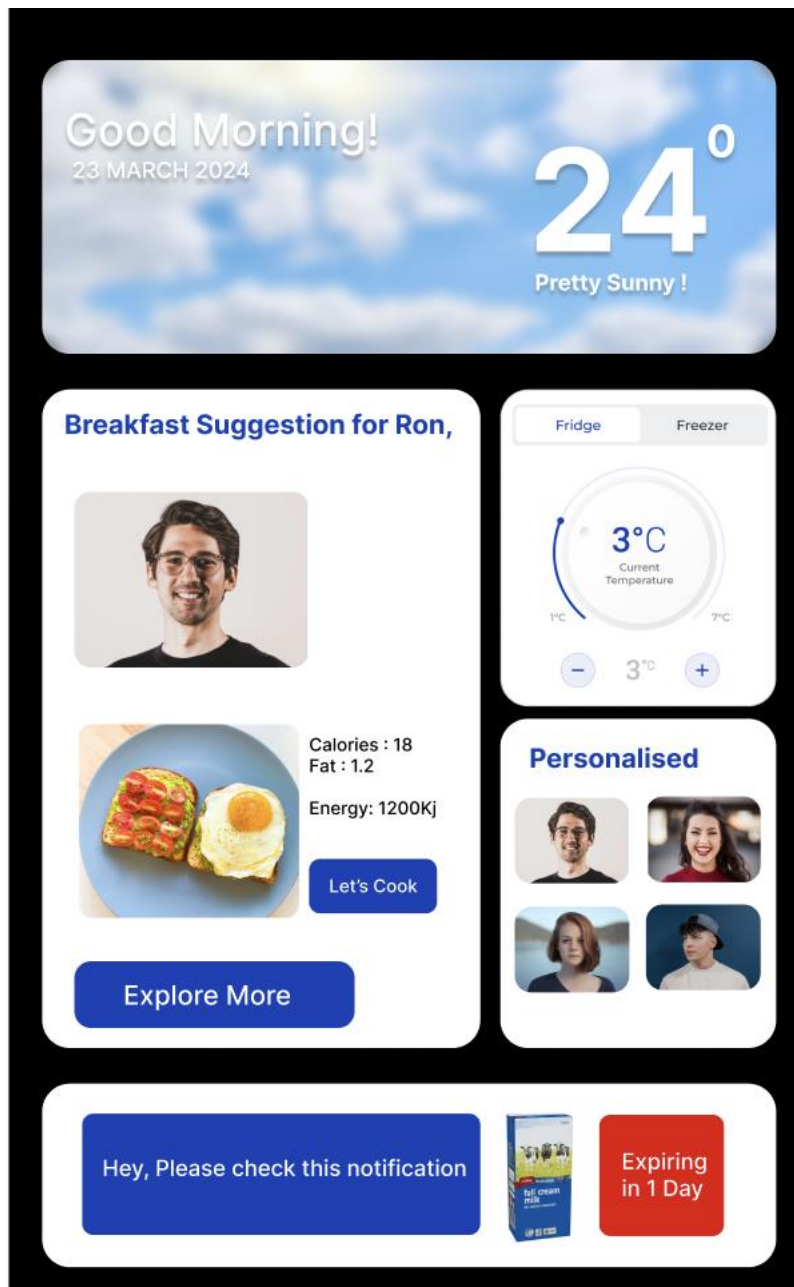
This is the screen for the fridge when the it's idle (Not performed any interaction with the fridge)

- Saves energy
- Shows the time and important notification



The Home screen for the fridge, it show the important features for the users on homepage.

- Weather
- Date & Time
- Breakfast suggestion
- Temperature Control
- Important Notification
  - Expiring Notification
  - Buy next



It shows the personalized meal suggestion based on the profiles when clicked.

### 3.7 Fridge Prototype



Tools used to create this SRS and the visuals:

Word Document – Microsoft Word

Design and Prototype - Figma.com

Diagrams - Draw.io

Image Search - Google.com

Copyright Free Images – Pixels.com

*“We believe that no system is perfect, and a good system is one that keeps changing to maintain the fulfillment of the users and according to current trends and requirements.”*

THANK YOU