

Overview of Project:

The UniFood mobile application is a map-centered, student-focused grocery/food app that will provide innovative advancements to the Google Maps software and will make students' lives more convenient. The main purpose of this app is to allow students to easily identify grocery stores and restaurants that are selling products at the best price and to be able to share with other users the most current prices of products. UniFood will also allow users to input their grocery lists, and these lists will be used to filter out locations best suited for students purchasing budgets and needs. Users will also be able to upload images and reviews to certain map locations, but reviews will require users to input pricing and deals to allow for better recommendations by the app to other users. Other features that will be added include an expiry date tracker/notifier, the user's current pantry inventories and nutritional summaries of their grocery lists. Other than the food map page, there will be 4 other pages including a pantry page with expiry dates, a grocery list page with nutritional information, a profile page with a friends list and a list of user's reviews and a sign-in/create account page.

List of Group Members and Their Responsibilities

1. Malhar Singh - Code Design and Coding:

- Development of user's current pantry page
- Development of expiry date tracker
- Development of sign-in page
- Development of grocery list scanner and input into grocery list
- Code integration and troubleshooting/testing
- Code documentation

2. Shian Li Chen - Code Design and Coding:

- Development of food map using integration of Google Maps API
- Development of reviews input forms and application to food map
- Development of filtering options for food map
- Development of grocery list scanner and input into grocery list
- Code integration and troubleshooting/testing
- Code documentation

3. Johnny Liang - Code Design and Coding:

- Development of grocery list input page
- Development of nutritional summaries of grocery lists
- Development of user's profile/posted reviews page
- Development of user's friend list
- Code integration and troubleshooting/testing
- Code documentation

List of Features and Functional Requirements

1. Sign In Page: Users will be able to sign in using their email account(s) allowing them to see their profile as well as their own posted reviews, grocery lists, friends list and pantries. Will require Firebase cloud authentication to store and verify user's emails, usernames and passwords.

2. Food Map: Displays a map similar to that of Google Maps, with just grocery stores and restaurant locations being the only highlighted locations. Users will be able to click on an area of the map to add a review to a registered location or register a new location. Will require the use of HTTP requests from the Google Maps API, Google Maps Flutter package, firebase authentication and Firestore for review and location information storage.

3. Filter Options: Filters the map locations based on different features like grocery stores, restaurants, pricing, rating, highly reviewed, etc. Will only show certain locations that match the filter requirements. Helps to streamline user's search options.

4. Reviews: Users are able to write reviews about certain locations. These reviews will be stored locally and in the cloud. Mandatory parts of the review will be item purchases and the current pricing details to allow for easy data comparisons. Reviews will be stored in cloud storage using Firebase authentication and cloud Firestore to allow for reviews to show up for all users.

5. User's Profile/Reviews Page: Shows a list of reviews made by the user. It will also show the profile of the user, user account details and their friends list. The account information and friends list will be stored in the cloud using cloud Firestore to allow for easy sharing and real-time synchronization.

6. Grocery List Creation: Lets the user make a grocery list. The user has the option to checkmark items they find. Users will be able to add and remove items from their grocery list. Items will be added using user-completed forms. Grocery lists will be stored in local storage using SQLite to allow for data persistence and will allow users to access and view their grocery lists while offline.

7. Grocery List Nutritional Summary: Shows a nutritional summary of the items within the grocery list (Total calories, proteins, carbohydrates, etc...). Allows users to make more informed decisions on what they should and shouldn't be buying. Nutritional information will be retrieved from grocery and food APIs (e.g., Open Food Facts API, Nutritionix API or Spoonacular API) using HTTP requests. The nutrition information provided by the APIs will be used to calculate nutritional summaries for grocery lists.

8. User's Current Pantry: Users will be able to make a list of items in their pantry and add and/or reduce the amount in their pantry according to the purchases and uses. The data in the pantry will be able to provide notifications to the user of what they need and of anything they add to the grocery list, also the user will be notified if the added item is already existing in the list. Users will add items to the pantry list using a user input form. The pantry list will be stored in local storage using SQLite to allow users to be able to access their pantry information while not connected to the internet and for faster data access.

9. Expiry Date Tracker: Allows the user to manually add the expiration dates to items in their pantry. The user can add a notification that will go off at a user input time before the expiration date. The expiry dates will be added using a form that will be provided when the user adds an item to their pantry list and will be regulated to make sure that they are in the form: mm/dd/year. Users will also be provided a written date update in the form similar to October 20, 2023, to allow users to verify that their inputted dates are correct. Expiry dates will be stored in local storage using SQLite along with the pantry lists in order to allow users to see the dates while offline and to allow for notifications to be provided to the users even while offline.

10. Page Navigation Bar: Allows users to easily switch between the pantry, map, grocery list and profile pages. Will contain easily understood icons and labels located in a bar at the bottom of the screen to allow for easy navigation between pages.

11. Grocery List Scanner: This will allow users to scan already saved photos of handwritten grocery lists on the user's device or will access the camera to scan the handwritten grocery lists in real-time. Will take the scanned text, process the data and will input the items into the grocery list page. Users will be able to see the information scanned and added to a form so that they can make any edits before the form contents are added to the grocery list page. Will require the use of the android camera permissions, the image picker package and the Optical Character Recognition (OCR) library: Google ML kit.

Non-functional requirements:

1. Tutorial: A tutorial with a walkthrough of the app will be provided to first-time users after their account has been created. It will have a step-through, including highlighting of specific areas of each page and detailed steps, indicating to the user how to add reviews to the food map, add/delete items from the grocery list and pantry, add friends to their friends list, update their profile details and will guide users on how to scan their handwritten grocery lists so that they can be added to the grocery list page.

2. Welcome page: The sign-up / login page will be the welcome page.

The signup page will have, as the mock-up shows, forms for the user to input their email, username, and password. The password shows the requirements for a valid password and another input field for password confirmation.

3. App Documentation: Each page will have a clickable title at the top of the page in the navigation bar that will allow the user to see a step-by-step walkthrough of the page just in case users forget how to operate specific features of the page. Documentation will also be provided in a full-page write-up that can be accessed on the profile page, in case users would prefer to read the how-to instructions instead of having to go through the full walkthrough of each page.

4. Code Documentation: Code documentation will be provided as comments in each dart file but will also be compiled into one full PDF page that will be accessible on the homepage of the repository.

5. Storage: Local and cloud storage will be required in order to make sure that the app data persists. For local storage, SQLite will be used whereas Google's Firebase and Cloud Firestore will be used for cloud storage. The food map information and users' reviews will be stored in cloud storage along with the user's login information (i.e., username, email address and password). User's grocery lists and pantry will be stored in local storage to allow for the lists to be quickly accessible for users even if they are not connected online.

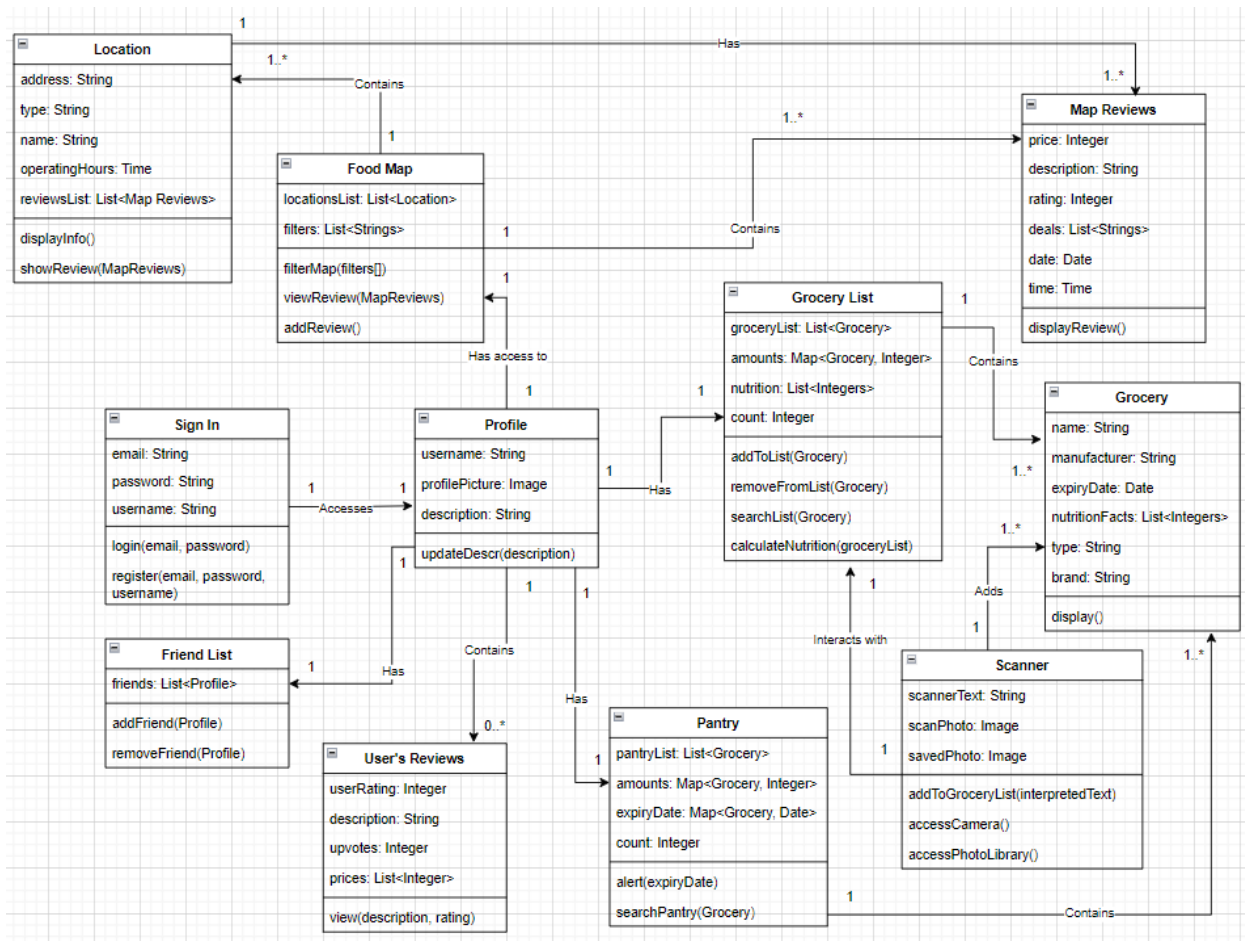
6. APIs/HTTP Requests: A number of APIs will be used to allow for the functionality of the application. For the food map page of the app, the Google Maps API will be used to have access to local maps, access to location information, geocoding and geolocation. APIs like Open Food Facts API, Nutritionix API or Spoonacular API will be used to retrieve the food product information, including name, brand, quantity and nutritional information. These food APIs will be used to help calculate the nutritional information in order to provide an accurate nutritional summary of user's grocery lists allowing them to make more informed decisions relating to their health. The information from the API will also be used to help users input the correct product names in their grocery list and pantry list.

7. Universality: Once all functionality has been completed and the application is fully functional, the intl package and arb files will be created to allow for internationalization of the app. Google's Cloud Translation API may be also used to help achieve internationalization. At the very least the app will be available in English and French, to accommodate the official languages of Canada.

8. Security: The security of the user's app and profile information will be preserved using Google's Firebase and Cloud Firestore. As Cloud Firestore is built around the idea of preserving data security, users' emails, passwords and usernames will be safely stored in the Google Cloud Firestore. Food map reviews will also be stored in the cloud Firestore to ensure security and prevent manipulation of reviews.

Code Design

Design Class Diagram Overview:



This Design Class Diagram illustrates the high-level overview of the entire app. This includes the following pages: **Sign-in page**, **Map page**, **Grocery List page**, **Pantry page**, **Camera/Scanner page**, and the **User's profile page**. In this UML diagram, classes, along with their methods and attributes, are shown. The general purpose of this diagram is to show how the different pages/classes interact with each other and the relationships between them. Since this is a design class diagram, it does not encompass the services (APIs, local storage, cloud storage, Scanner, etc.), which we

further explained in each page's Use Case Diagram. Since this app does not allow guests and the user has to log in before viewing any additional page within the app, most of the pages/classes are connected to the **Profile class**.

The first page shown to the user is the sign-up/login page. This page has three attributes **username, email, and password**. It also has two methods **login()** and **register()**. The **login** method would verify if the user exists (using **email and password**) and would **validate** it with the **database**. The **register** is for first-time users which on successful registration (**valid username, email, and password**), stores the **new user** information in the **database** (**relationship: accesses info from user profile if applicable**).

After logging in or signing up, the user can **navigate** to any **page** they like. The **Map page** allows users to use the **features** of **Google Maps** with **additional features** suited to their needs. The **Food Map class** has a list of **locations** and **filters**. The **filterMap()** method shows the information on the map after applying the **filters** the user selected. The **addReview()** allows the user to **add a review** for a selected location. The Food Map Class is also split into two separate classes for cleaner code, one for **location information** and one for **reviews**.

The **location class** includes **displayInfo()** which displays the following: **address, type (restaurant, grocery store), and operating hours**. It also includes **showReviews()** which shows a **list of reviews** for that specific place.

The **Map Reviews class** has a **displayReview()** which displays the following: **price, rating, description, date, time, and deals (a list of deals, not just one)**.

Each User would also have their own **profile**. Within this page, the User can **view** and **edit** their (**username, profile picture, and/or description**). This page also has a **Friends List** which allows the user to **view, add, and/or remove friends**.

User Reviews are also contained within this page which displays information such as **user rating, description, upvotes and price** with the use of **view()**.

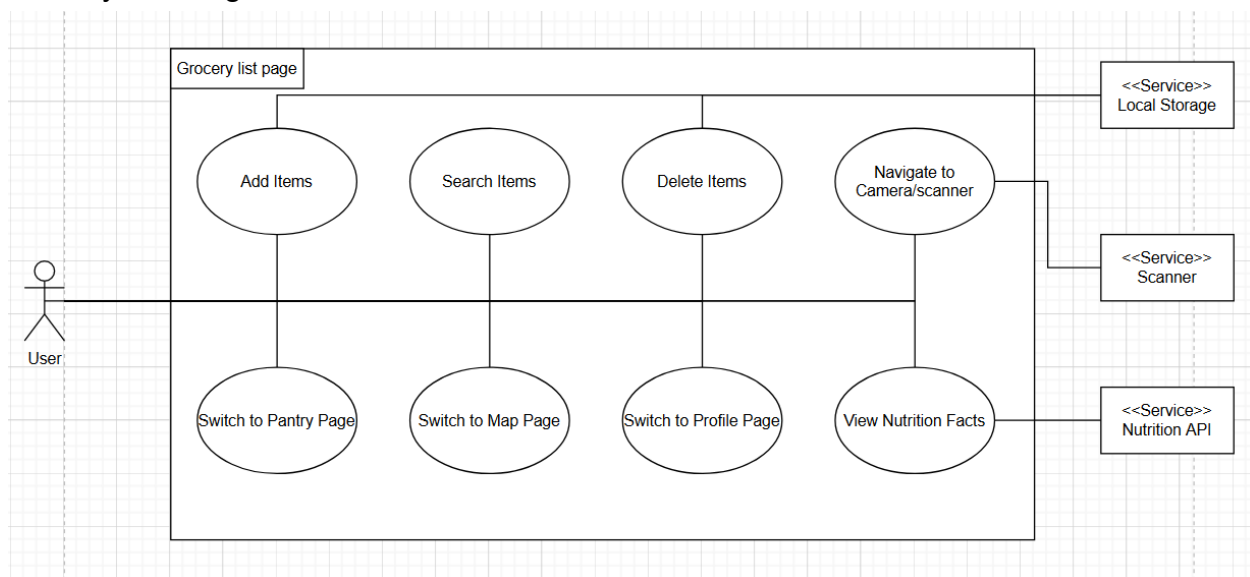
There are also two additional pages connected to the user, **Grocery List** and **Pantry**, which both utilize the **Grocery class**. The **Grocery class** has a **display()** to display general information about each product. This includes things such as the product's **name, manufacturer, expiration date, nutritional information, type, and brand**.

The **Grocery List** contains a **list of type grocery, the amount of each item (not cost, quantity), and nutritional information (as a Map<Grocery, integer>)**. It also contains methods to **add, remove, search items** or **calculate nutritional information (calculates it for the combination of all items)** within the **grocery list**.

The **Scanner class** also interacts with **Grocery List** because users can scan items to add items to their grocery list, instead of just typing. This class will have methods to **add items** to the grocery list (**which will use the attribute: scannerText**), **access the camera (save scanned photo as scanPhoto)**, and/or access the phone's **photo library (save selected photo as savedPhoto)**.

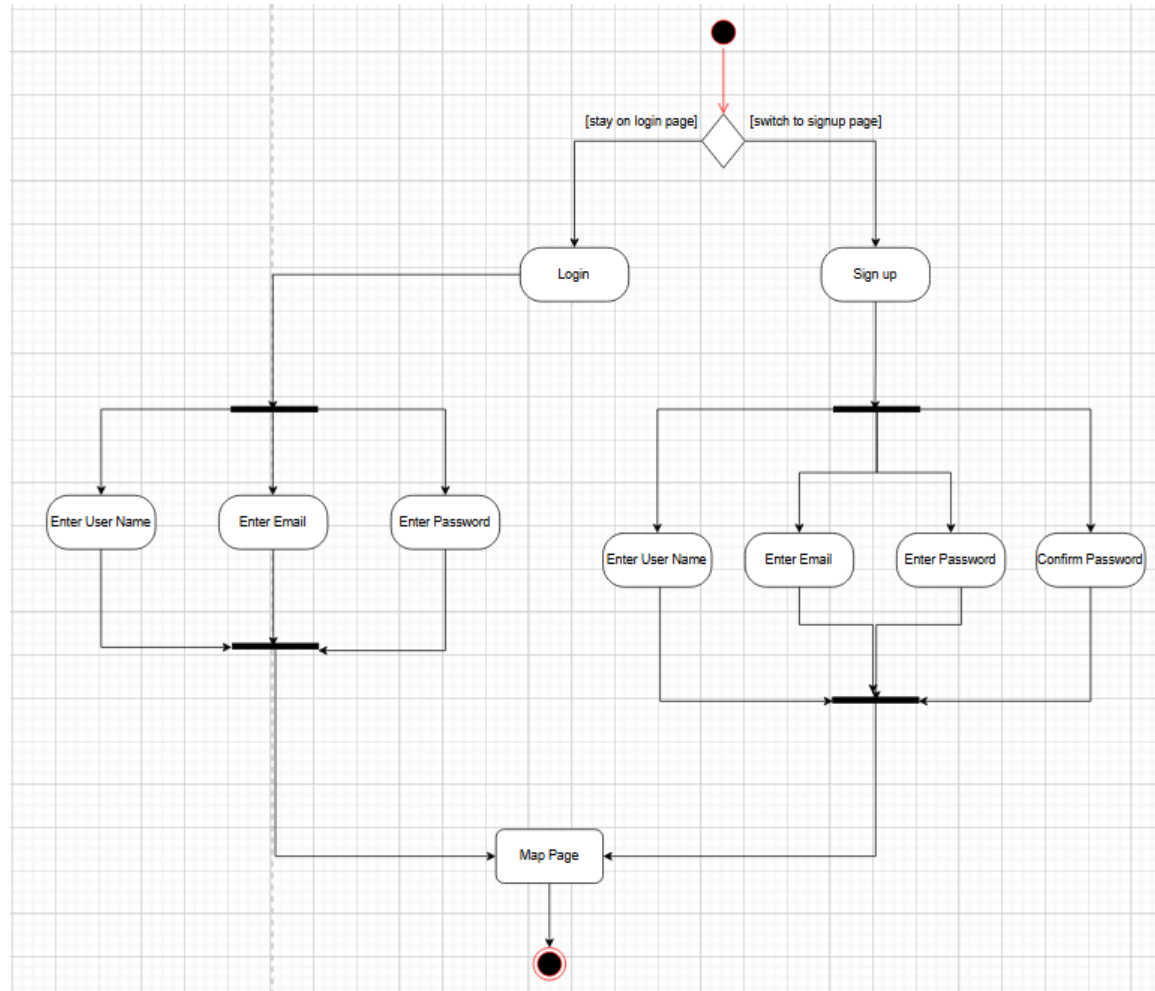
The **Pantry** has a **list of type grocery**, the **amount of each item (as a Map<Grocery, integer>)**, and the **expiry date (as a Map<Grocery, date>)**. It also has two methods **alert()** and **searchPantry()**. The alert method is for sending the user **alerts/notifications** if one or more of their items are about to expire. The search method allows the user to search for items in their pantry.

Grocery List Page:



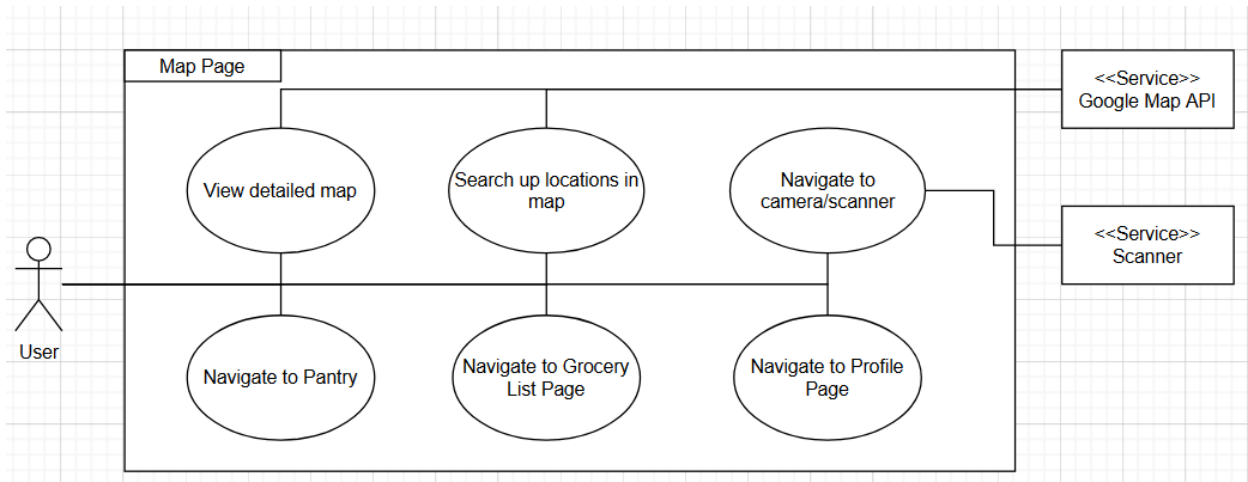
- This use case diagram shows the functionalities the user can use and the service utilized.
- **Add Items**, and **Delete Items** are both work with **<<Service>> Local Storage**.
- **Local Storage** stores the added items to the phone's local storage to later compare with the user's pantry.
- **View Nutrition Facts** is works with **<<Service>> Nutrition API**.
- **Nutrition API** is an API with information about the nutrition of the items.
- **Navigate to Camera/Scanner** utilizes the scanner services provided by flutter.
- **Switch to Pantry Page**, **Switch to Map Page**, **Switch to Profile Page**, and **Camera/Scanner page** are used to move to the mentioned page.

Login/Sign Up Page:



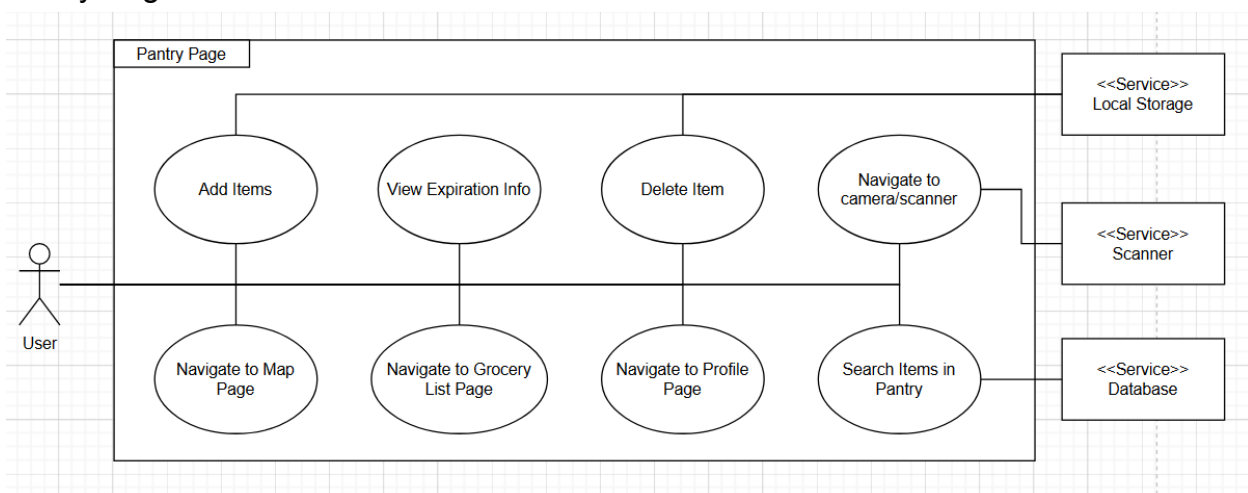
This activity diagram shows the workflow of the user interaction with the application. It starts by giving the option to Login or Sign up. If Login is chosen, it will prompt the user to Enter UserName, Email, and Password, and end in Map Page. On the other hand, if Sign up is chosen, the user is prompted to Enter Username, Email, Password and to confirm the Password and from that go to Map Page.

Food Map Page:



- This use case diagram shows the functionalities of Food Map page and the services utilized.
- **View detailed map** and **Search up locations in map** work with <<Service>> **Google Map API**. They use the API to produce a very detailed map with some additional features.
- **Navigate to Camera/Scanner** utilizes the scanner services provided by flutter.
- Also, from this page, the user is able to navigate to **Pantry**, **Grocery List**, **Profile page** and **Camera/Scanner** page.

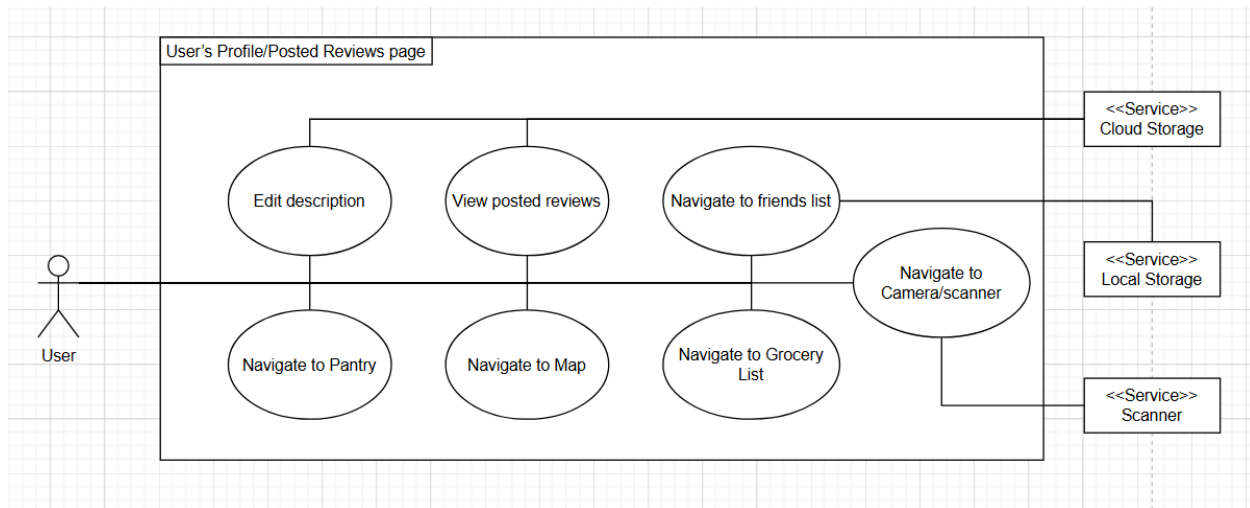
Pantry Page:



- This use case diagram shows the functionalities of the Pantry page and the services utilized.
- **Add Items** and **Delete Item** work with the <<Service>> **Local Storage**.
- **Local Storage** stores the added/deleted items to the phone's local storage to later compare with the user's Grocery list.

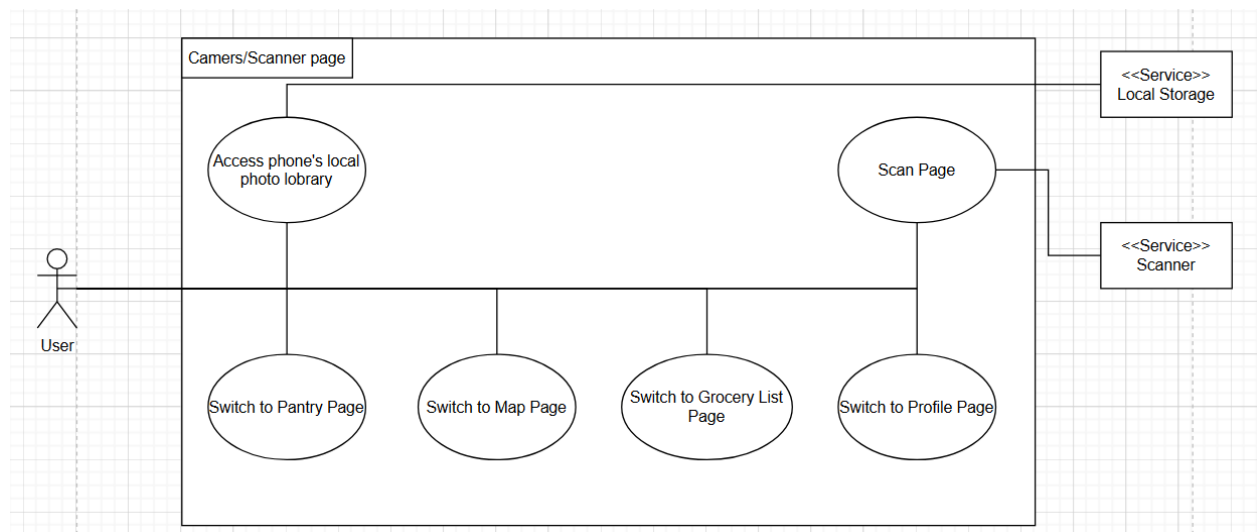
- **Navigate to Camera/Scanner** utilizes the scanner services provided by flutter.
- The User is also able to **Search Items in Pantry and View Expiration Info.**
- Also, they are able to navigate to **Map, Grocery List, Profile and Camera/Scanner page.**

User's Profile/Posted Reviews page:



- This use case diagram shows what functionality is within the Profile/Posted Reviews Page and the services utilized.
- **Edit description** and **View posted reviews** both work with the **<<Service>> Cloud Storage**.
- **Cloud Storage** is used for storing the description of the user and the reviews in cloud storage.
- **Navigate to friends list** works with **<<Service>> Local Storage**.
- **Local Storage** is used to store the information of friends.
- **Navigate to Camera/Scanner** utilizes the scanner services provided by flutter.
- Also, the User will be able to navigate to the **Pantry, Map, Grocery List page,** and **Camera/Scanner page.**


Camera/Scanner:



- The **Camera/Scanner page** allows users to scan their grocery list directly using the phone's camera or access their photo library on the phone.
- The **<<Service>> local Storage** stores the users photo(s) in local storage.
- The **<<Service>> Scanner** is used to allow for scanning pages/
- Also, From the bottom navigation bar, the User will be able to navigate to the **Pantry, Map, Grocery List page, and Profile page.**


Mockup of User Interface

Sign in and create user account page




[Login](#) | [Sign Up](#)

or



[Log In](#)



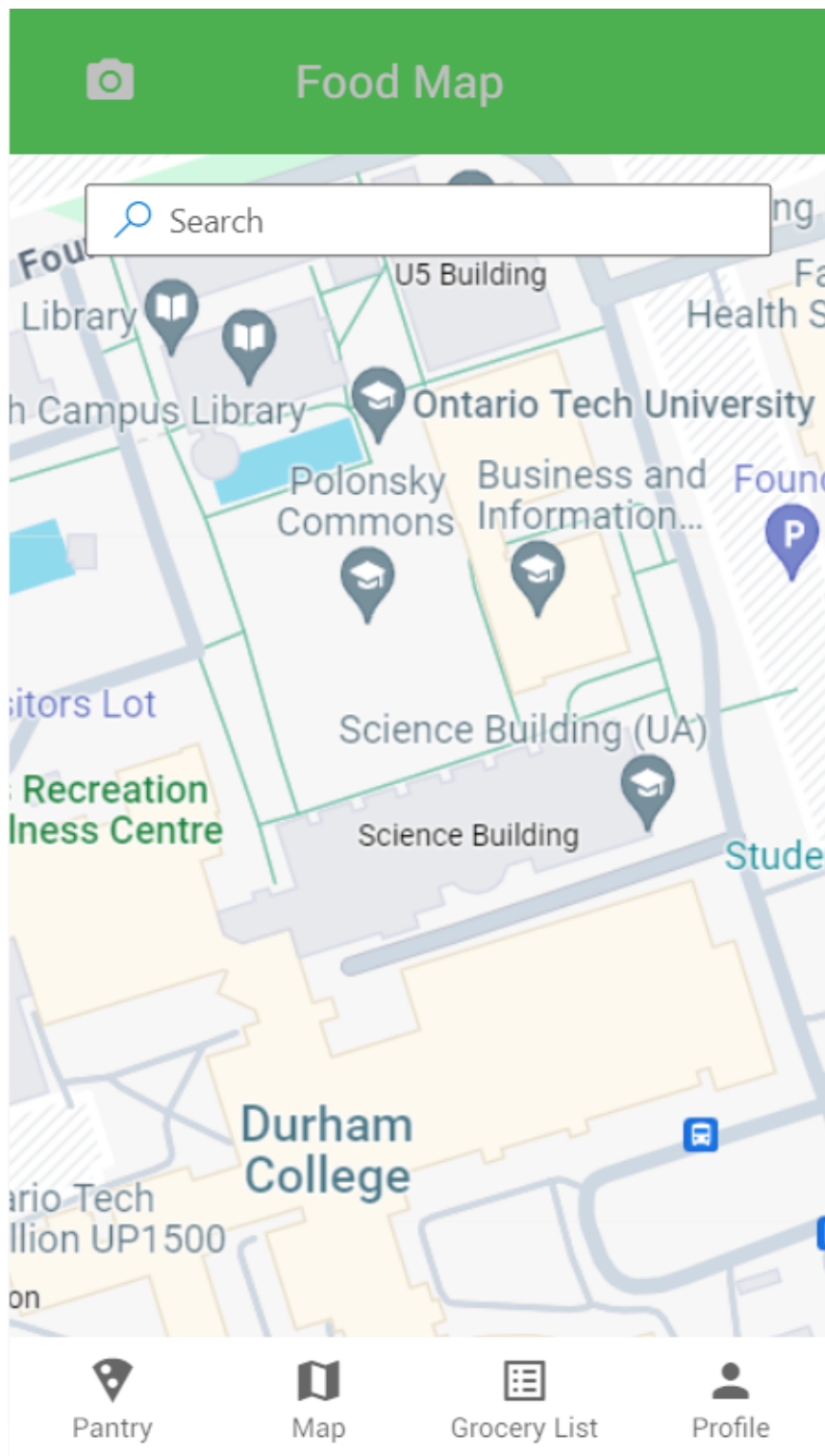
[Login](#) | [Sign Up](#)


Must be 8-15 characters, contain at least 1 number and one special character (!@#\$\$%^&*()_)

[REGISTER](#)

[Already a user? Login](#)


Food Map page





Grocery List

DELETE



Search

1

Eggs
Amount: 1

☐

2

Milk
Amount: 1

☐

3

Bread
Amount: 1

☐

4

Butter
Amount: 3


☐

Add new item


+

Nutritional Summary:


4000 cal, 250 g fat, 100 g protein, 1000 g carbs




Pantry




Map



Grocery List




Profile



Pantry

DELETE



Search

1

Eggs
Amount: 1

EXP. Date
10/20/2023

!

2

Milk
Amount: 1

EXP. Date
10/20/2023

!

3

Bread
Amount: 1

EXP. Date
10/20/2023

!

4

Butter
Amount: 3

EXP. Date
12/01/2023


✓

Add new item


+

Number of Items Expiring:


3




Pantry



Map



Grocery List



Profile

User's Profile/Posted Reviews page

