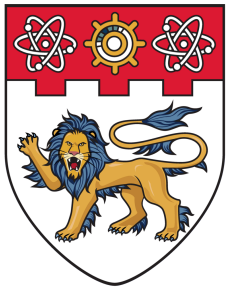


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC2006 - Software Engineering

Lab 4 Deliverables

Lab Group: SCSD

Team: FeedEmGreens

Members: Rushaidy, Rajath, Hanzhi, Girija, Yohesh

Table of Contents

Table of Contents.....1
1. Project Mission Statement.....2
2. Documentation of functional and non-functional requirements.....2
A. Functional Requirements..... 2
B. Non-Functional Requirements.....9
3. Data Dictionary..... 11
4. Use Case Diagram..... 14
A. Use Case Descriptions.....15
5. UI Prototype..... 49
6. Boundary, Control & Entity Model……61
7. Class Diagrams……..63
8. Sequence Diagrams……..64
9. Initial Dialog Map……71
10. Traceability Matrix……72

1. Project Mission Statement

FeedEmGreens' intention is to develop an application which will help ease health-oriented individuals or people with dietary restrictions to locate and direct them towards cheap, hygienic and healthy eateries based on their personalization.

2. Documentation of functional and non-functional requirements

A. Functional Requirements

1. The system shall perform user authentication to allow Users to use the application.
    1.1. The system shall allow Users to create an account
        1.1.1. Users shall enter at minimum: username, email, and password.
        1.1.2. An account created shall be tagged with the “User” role.
    1.2. The system shall allow Administrators to create an account with Admin privileges.
        1.2.1. Administrators shall enter username, email, and password.
        1.2.2. The “Admin” role shall be tagged to the account upon creation.
    1.3. The system shall allow Users to sign in with the account they created.
        1.3.1. Users shall enter their username and password to log into the application
        1.3.2. The system shall mask passwords with asterisks unless the unhide option is selected.
        1.3.3. If the username or password is incorrect, the system shall display:*“The username or password is incorrect. Please try again.”*
        1.3.4. After 3 unsuccessful attempts within 5 minutes, the system shall lock the account for 30 minutes.
2. The system shall allow Users to discover the location of healthier eateries
    2.1. The system shall display healthier hawker centres and stalls on login.
        2.1.1. The system shall display results in list views.
        2.1.2. The system shall allow Users to search by stall name, hawker centre name, or location keyword.
        2.1.3. The system shall display for each result: name, address (with map pin), food categories, operating hours, and price indicator.

2.1.4. If any information is missing, the system shall display “Unavailable” for that field.

2.1.5. In the list form, the system shall sort search results according to distance.

2.2. The system shall allow Users to sort results.

2.2.1. Sorting options shall include distance, average price, and popularity.

2.3. The system shall allow Users to filter results

2.3.1. Users can filter eateries based on dietary tags, maximum budget, distance from current or chosen location and healthy options

2.3.2. The system shall output filtered results in a form of lists sorted according to distance, price or popularity

3. The system shall provide crowd estimation for Users.

3.1. The system shall display a **crowd indicator** for the area around the eatery

3.2. The crowd indicator shall classify levels as Low (< 5 mins), Medium (5-15 mins) , or High (>15 mins).

3.3. The system shall perform crowd estimation using contextual factors.

3.3.1. The system shall use time of day to refine estimation (e.g., peak lunch and dinner hours).

3.3.2. The system shall use day of week to refine estimation (e.g., weekends vs weekdays).

3.3.3. The system shall combine contextual factors with real-time or historical data to refine queue and crowd estimates

3.4. The system shall handle unavailable data.

3.4.1. If neither real-time nor historical data is available, the system shall display “Crowd data unavailable.”

4. The system can direct Users to the location of their desired eatery.

4.1. The system shall provide a “Get Directions” option on each hawker centre detail page

4.1.1. The system shall generate optimal routes to the selected hawker centre or stall using the OneMap API.

4.1.2. The system shall request location access to detect the User's current location for route calculation.

4.1.3. If location access is denied by the User, the system shall allow the User to manually enter a starting point.

4.1.4. The system shall display multiple modes of transportation supported by OneMap (e.g., walking, bus, train, driving).

4.1.5. For each mode, the system shall display estimated travel time and distance.

4.1.6. If the OneMap API is unavailable, the system shall display the notification: **"Directions are not available at this time."**

5. The system shall suggest food options based on preferences.

5.1. The system shall display recommendations in the home page to authenticated Users only.

5.1.1. If location access is denied and a distance filter is active, the system shall display default results based on distance.

5.2. The system shall generate a recommendation candidate only if stall data meets all constraint requirements.

5.2.1. The system shall exclude any stall that does not have valid distance/location data.

5.2.2. The system shall exclude any stall that does not have valid dietary tag data.

5.2.3. The system shall exclude any stall that is not currently open.

6. The system enables users to share feedback and see aggregated ratings.

6.1. The system shall restrict review and rating actions to authenticated Users only.

6.1.1. The system shall not allow non-authenticated Users to submit, edit, or delete reviews.

6.1.2. The system shall prompt non-authenticated Users to log in when attempting a review action.

6.2. The system shall allow a User to submit one active review per stall.

6.2.1. A review shall include a **Health Score** (1–5 scale, integers only).

6.2.2. A review shall include a **Hygiene Score** (1–5 scale, integers only).

6.2.3. A review may include optional text feedback.

6.2.4. A review may include optional photos.

6.2.5. If the User has an existing review for the stall, the system shall open the existing review for editing instead of creating a new one.
6.2.6. On successful submission, the system shall confirm with “Review posted.”

6.3. The system shall allow a User to edit their own review.
6.3.1. The system shall allow editing of Health Score, Hygiene Score, text, and photos.
6.3.2. On successful update, the system shall confirm with “Review updated.”

6.4. The system shall allow a User to delete their own review.
6.4.1. The system shall display a confirmation prompt before deletion.
6.4.2. On deletion, the system shall remove the review and confirm with “Review deleted.”

6.5. The system shall allow Users to flag a review as inappropriate.
6.5.1. A flag shall include a selected reason (e.g., spam, offensive, false information).
6.5.2. On successful flag submission, the system shall confirm with “Review reported.”
6.5.3. The system shall add flagged reviews to the Admin moderation queue.

6.6. The system shall display reviews on the stall details page.
6.6.1. Each review shall show Health Score, Hygiene Score, text (if any), photos (if any), author alias, and timestamp.
6.6.2. The system shall label the User’s own review as “Your review.”
6.6.3. The system shall sort reviews by Most Recent by default.
6.6.4. The system shall allow sorting by Highest Health Score and Highest Hygiene Score.
6.6.5. If no reviews exist, the system shall display “No reviews yet.”

6.7. The system shall display aggregated scores for each stall.
6.7.1. The system shall display the **average Health Score** (to one decimal place).
6.7.2. The system shall display the **average Hygiene Score** (to one decimal place).
6.7.3. The system shall display the total number of reviews used in each average.
6.7.4. The system shall update averages immediately after a review is added, edited, or deleted.

6.8. The system shall prevent review spam and duplicates.
6.8.1. The system shall restrict each User to one active review per stall.

6.8.2. The system shall enforce a minimum interval of 7 days between new review submissions for the same stall by the same User (edits are allowed anytime).

6.9. The system shall handle validation and error states for review actions.

6.9.1. If required fields (Health Score, Hygiene Score) are missing, the system shall display an inline validation message and prevent submission.

6.9.2. If photo upload fails, the system shall display “Photo upload failed” and allow resubmission without photos.

6.9.3. If a network error occurs, the system shall display “Unable to submit review. Please try again.”

6.10. The system shall expose review management actions via clear UI controls.

6.10.1. The system shall provide **Write review**, **Edit**, **Delete**, and **Report** buttons where applicable.

6.10.2. The system shall disable **Edit** and **Delete** for reviews not authored by the current User.

6.11. The system shall ensure consistency between review list and details.

6.11.1. After adding, editing, or deleting a review, the system shall refresh the review list and aggregated Health/Hygiene scores shown on the stall details page.

7. The system shall allow users to redeem rewards for using the application.

7.1. The system shall award points to authenticated Users for specific actions.

7.1.1. The system shall award points when a User submits a Health & Hygiene review.

7.1.2. The system shall prevent awarding duplicate points for the same action within a 24-hour period.

7.2. The system shall display the User’s points balance.

7.2.1. The points balance shall be displayed on the User’s profile page.

7.3. The system shall allow Users to redeem rewards with points.

7.3.1. The system shall display a list of available rewards and their required point values.

7.3.2. The system shall allow a User to confirm redemption with a “Redeem” button.

7.3.3. On successful redemption, the system shall deduct the corresponding points from the User’s balance.

7.3.4. The system shall confirm redemption with a message “Reward redeemed successfully.”

7.4. The system shall handle insufficient points for redemption.

7.4.1. If a User attempts redemption without enough points, the system shall display “Not enough points.”
7.4.2. The system shall disable the “Redeem” button for rewards where the User’s points are below the required threshold.
7.5. The system shall define reward expiration.
7.5.1. Each redeemed reward shall display an expiry date, if applicable.
7.5.2. The system shall prevent use of rewards after their expiry date.
7.6. The system shall ensure fairness in point awarding.
7.6.1. The system shall not allow manual modification of points by Users.
7.6.2. The system shall log all point transactions (award, redemption, expiry) for audit.
8. The system shall allow authenticated Admins to manage content quality and verify sensitive information.
8.1. The system shall restrict moderation actions to authenticated Admins only.
8.1.1. The system shall not allow non-Admin Users to access moderation functions.
8.1.2. The system shall prompt login if a non-authenticated User attempts to access moderation.
8.2. The system shall provide an Admin dashboard.
8.2.1. The dashboard shall display flagged reviews requiring moderation.
8.2.2. The dashboard shall display reported hygiene/health reviews requiring moderation.
8.2.3. The dashboard shall provide access to dietary tag management for stalls.
8.3. The system shall allow Admins to act on flagged reviews.
8.3.1. The system shall allow Admins to approve a flagged review as valid.
8.3.2. The system shall allow Admins to hide a review containing offensive or inappropriate content.
8.3.3. The system shall allow Admins to delete a review entirely.
8.3.4. The system shall require Admins to provide a reason for any removal.
8.3.5. The system shall notify the User who authored the review of the moderation outcome.
8.4. The system shall allow Admins to modify dietary tags in the database.
8.4.1. The system shall allow Admins to add a dietary tag (e.g., Halal, Vegetarian, Healthy) to a stall.
8.4.2. The system shall allow Admins to remove a dietary tag from a stall.
8.4.3. The system shall allow Admins to edit existing dietary tags for accuracy.

8.4.4. Changes to dietary tags shall be updated in the User-facing app immediately after Admin confirmation.

8.5. The system shall ensure all Admin actions are logged.

8.5.1. Each moderation action shall log: Admin ID, action type, target (review/tag/user), timestamp, and outcome.

8.5.2. Logs shall be viewable only by Admins and system auditors.

8.6. The system shall handle error and edge cases.

8.6.1. If a moderation action fails due to network or database error, the system shall display “Action could not be completed. Please try again.”

8.6.2. If an Admin attempts to act on a review or tag that has already been resolved, the system shall display “This item has already been moderated.”

B. Non-Functional Requirements

1. **Performance**
    - The system shall load search results and recommendations within 2 seconds under normal network conditions.
    - The system shall handle at least 500 concurrent users with no noticeable degradation in response time.
    - Queue and crowd estimation data shall be updated at least every 5 minutes.
    - Application startup time shall not exceed **5 seconds** on a **mid-range smartphone (e.g. iPhone 12, Samsung Galaxy S20, or equivalent with 6GB RAM and 4G connection)**.

2. **Scalability**
    - The system shall support scaling to 10,000 registered users without requiring major architectural changes.
    - The system design shall allow easy integration of additional hawker centres and stalls without downtime.
3. **Usability**
    - The system shall allow a User to locate a stall (search + filters) within 3 interactions or fewer
    - The system shall provide an intuitive and mobile-friendly interface, with at least 90% of test users rating usability ≥ 4/5 during usability testing.
    - The system shall allow smooth map zoom and pan with a delay of less than **2 seconds**.
    - Users should have the ability to view, access, edit, share, or opt out of sharing specific aspects of their data.
4. **Reliability & Availability**
    - The system shall achieve 99.5% uptime per month, excluding scheduled maintenance.
    - The system shall ensure no more than 1 minute of downtime per week due to unexpected failures.
    - If the map API is unavailable, the system shall display a fallback message (e.g. “Map currently not available”) within 2 seconds.
5. **Security**
    - All user credentials shall be stored securely using hashed and salted passwords.
    - The system shall enforce multi-factor authentication (MFA) for admin
      The system shall request explicit User consent before accessing location data.

- Only authorised Admin accounts shall have access to moderation and management features.

6. **Data Accuracy**
   - Nutritional information and stall tags (e.g., “Healthier Choice”) shall be verified quarterly to ensure data reliability.
7. **Legal & Compliance**
   - The system shall comply with Singapore’s PDPA for handling personal data.
   - All geolocation data shall be anonymized when stored or analyzed for recommendations.
8. **Gamification & Incentives**
   - Points and rewards shall be processed within 2 seconds of meal logging or redemption.
   - Reward redemption systems shall prevent duplicate or fraudulent claims through transaction logging and validation.
9. **Supportability**
   - The database must be replaceable with any commercial product supporting standard SQL queries.

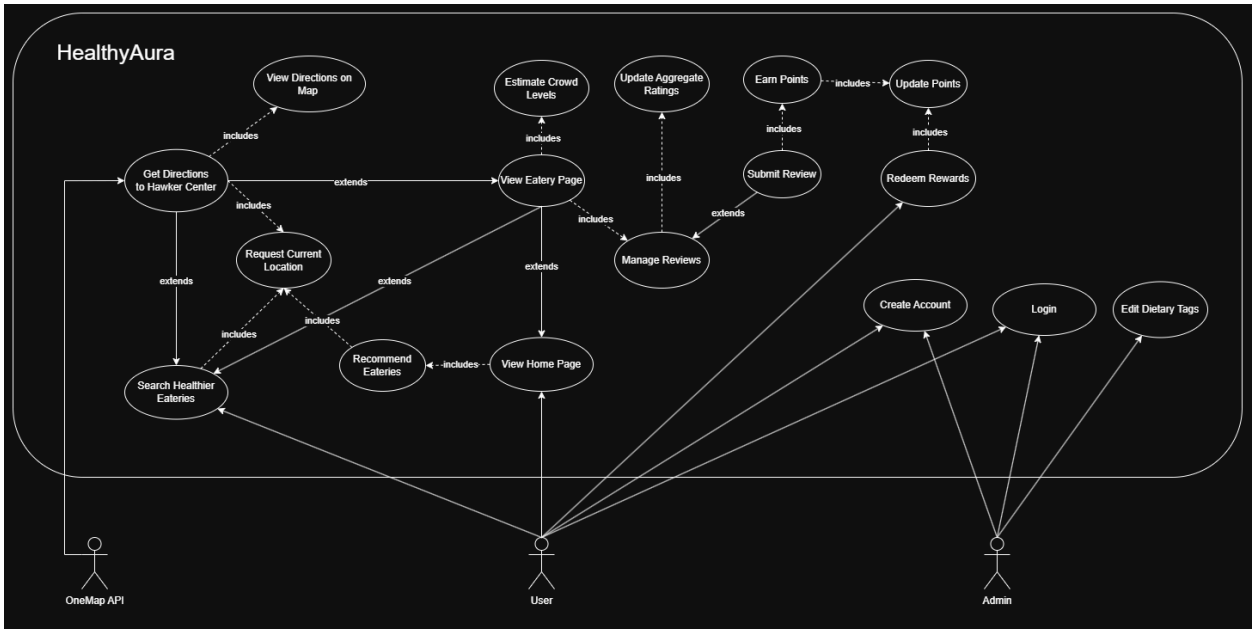
3. Data Dictionary

TERM	DEFINITION
Account	A registered User profile containing login credentials and optional personal data (e.g., dietary preferences, profile picture, review history).
Admin	A User with special privileges for moderation, managing reviews, editing dietary tags, and resolving support tickets.
Application (App)	The mobile software application that enables Users to discover healthier food options in Singapore hawker centres and stalls
Consumer	A person that uses HealthyAura to view Hawker centre's review, search for healthier food places, and perform consumer-specific tasks.
Crowd Indicator	A measure of busyness in the area around a hawker centre, classified as Low, Medium, or High
Dietary preference	A User's chosen food requirements (e.g., Halal, vegetarian, healthier choice) that the system uses to filter and personalise recommendations.
Dietary Tag	A label applied to a stall to indicate suitability for specific diets (e.g., Halal, vegetarian, vegan, healthier choice). Dietary tags are managed by Admins
Food place / Stall	Any hawker centre or stall displayed in the app with location, dietary tags, and hygiene rating.
Hawker centre	A food court in Singapore containing multiple stalls.
Healthier choice	A certification under Health Promotion

	Board (HPB) guidelines for healthier meals.
Hygiene rating	Cleanliness grading from user reviews and displayed in the app to help users choose cleaner and safer food outlets.
Map	An interactive feature that displays hawker centres and stalls, filtered by distance, dietary tags, and hygiene rating.
Points	A numerical value earned by Users for completing defined actions (e.g., healthy food choice, review submission), which can be redeemed for reward
Queue Indicator	An estimate of waiting time at a specific stall, shown to the User
Recommendation	A personalised suggestion for a stall or hawker centre, shown to the User based on dietary preferences, distance, and health criteria
Review	User-generated feedback on food quality, hygiene, and service at a food place, consisting of Health and Hygiene scores, optional text, and optional photos.
Reward	A benefit (e.g., points, vouchers, or discounts) granted to a user upon completing specific actions in the system, such as successful purchases,submitting reviews etc
Search	A User-submitted request for assistance, tracked with unique ID and status
User	An individual who has created an account in the application. A User may hold one of the following roles: ● **Admin** – has privileges to manage reviews, dietary tags, support tickets, and User accounts. ● **Standard User** – can search, filter,

	view, and review eateries, as well as receive recommendations.

4. Use Case Diagram



A. Use Case Descriptions

1. Use Case 1 - Create Account

Use Case ID:	UC-1		
Use Case Name:	Create Account		
Created By:	Rushaidy	Last Updated By:	Rushaidy
Date Created:	30th August 2025	Date Last Updated:	2nd September 2025

Actor:	● User (primary) ● Admin (for admin account creation)
Description:	This use case allows a new User / Admin to register an account in the application with required and optional details.
Preconditions:	● System is online and registration service available. ● User is not currently authenticated.
Postconditions	● A new account with the correct role is created. ● If unsuccessful, no account is created and entered data is discarded or saved as draft.
Priority:	High
Frequency of Use:	High during initial rollout; moderate thereafter.
Flow of Events:	1. The User selects Sign Up. 2. The system prompts for required fields: username, email, password. 3. The User enters the required information. 4. The system prompts for optional fields: mobile number, diet preferences, profile picture. 5. The User may enter optional information. 6. The system validates uniqueness of username/email and password complexity. 7. If valid, the system creates a new account tagged with “User” role for Users and “Admin” roles for admin. 8. The system confirms successful account creation to the User / Admin.

Alternative Flows:	**AF-1:** Email already registered 1. During validation, the system detects that the email is already in use. 2. System displays “Email already in use” and prompts user to either provide another email or recover account. 3. User chooses a path and the system continues accordingly **AF-2:** Username unavailable 1. System detects the chosen username is taken. 2. System suggests available variants. 3. User selects a suggested name or enters a new one; system revalidates. **AF-3:** Skipping optional fields 1. If the user skips optional details, the system creates the account with a minimal profile.
Exceptions:	**EX-1:** Weak password ● If the password does not meet policy, the system blocks account creation and displays password rules. **EX-2:** Unauthorized admin creation ● If a non-admin tries to create an Admin account, the system blocks the attempt, logs it, and informs the user. **EX-3:** Technical failure ● If there is a network or database error, the system saves the entered data temporarily and prompts the user to retry.
Includes:	None
Special Requirements:	1. Passwords must be hashed and salted before storage. 2. Real-time validation for email/username/password to reduce retries. 3. Captcha or equivalent must be implemented to prevent automated sign-ups.

Assumptions:	Email verification service is active.

2. Use Case 2 - Login

Use Case ID:	UC-2		
Use Case Name:	Login		
Created By:	Rushaidy	Last Updated By:	Rushaidy
Date Created:	30th August 2025	Date Last Updated:	2nd September 2025

Actor:	User or Admin
Description:	This use case authenticates user / admin with username and password.
Preconditions:	● The user has a registered account. ● The account is active and not locked.
Postconditions	● On success: Session token issued, last login updated. ● On failure: No session created, attempt logged.
Priority:	High
Frequency of Use:	Very high (daily).
Flow of Events:	1. User / Admin selects **Login**. 2. System prompts for username/email and password. 3. User / Admin enters credentials. 4. System masks password by default. 5. User may unhide password. 6. System validates credentials. 7. If valid, grants access.
Alternative Flows:	**AF-1:** Login with email instead of username 1. User enters registered email instead of username. 2. System maps email to account and continues authentication.

Exceptions:	**EX-1:** Invalid credentials ● If details are incorrect, the system denies access, displays an error, and increments the failed login counter. **EX-2:** Account locked ● If the account is locked or suspended, the system blocks access and displays a message to contact support. **EX-3:** Auth service unavailable ● If the authentication service is down, the system shows “Login unavailable. Please try again later.”
Special Requirements:	1. System should support optional MFA 2. Failed login attempts must be logged.

3. Use Case 3 - Search Healthier Eateries

Use Case ID:	UC-3		
Use Case Name:	Search Healthier Eateries		
Created By:	Rajath Krishna	Last Updated By:	Rajath Krishna
Date Created:	30th August 2025	Date Last Updated:	2nd September 2025

Actor:	● User (primary) ● Device Location Services (supporting)
Description:	The User searches for and views a list of healthier hawker centres or stalls. The System presents results with basic details and filters.
Preconditions:	● User authenticated. ● Eatery catalog accessible.
Postconditions	● Matching eateries are displayed in list. ● User can select an eatery to view details
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User opens the Search screen after login. 2. User is prompted to give current location using included use case **UC-3.I1 Request Current Location** 3. System displays a list of eateries with basic details (name, address, categories, hours, price indicator). 4. User enters a search keyword (e.g., stall name, hawker centre, or location). 5. System updates the list with matching results. 6. User selects an eatery → System executes UC-12 View Eatery Page 7. User selects Get Directions → System executes UC-5 Get Directions to Hawker Centre
Alternative Flows:	**AF-1:** Location access denied 1. User rejects location request.

	2. System switches to manual search (area or postal code). **AF-2:** No results found 1. Search query returns no matches. 2. System displays “No results found” and suggests broadening filters or nearby areas.
Exceptions:	**EX-1:** Catalog API failure ● If the system cannot fetch eatery data, it shows an error message and a Retry option. **EX-2:** Location fetch timeout ● If the system cannot get location within a set time, it prompts the user to enter location manually. **EX-3:** Offline mode ● If internet connectivity is unavailable, cached results are displayed with the label “Data may be outdated.”
Includes:	● UC-3.I1 Request Current Location ● UC-11 Estimate Queue and Crowd Levels
Extended by:	● UC-5 Get Directions to Hawker Centre ● UC-12 View Eatery Page
Special Requirements:	1. Search results should update in under 0.5 seconds.
Assumptions:	

3.1. Use Case 3.I1 - Request Current Location

Use Case ID:	UC-3.I1		
Use Case Name:	Request Current Location		
Created By:	Rajath Krishna	Last Updated By:	Rajath Krishna
Date Created:	30th August 2025	Date Last Updated:	2nd September 2025

Actor:	● User ● Device Location Services
Description:	The System requests permission to access the User's current location to support location-based search and map features.
Preconditions:	● Device location services are enabled.
Postconditions	● User's coordinates retrieved and returned to UC-3.
Priority:	Medium
Frequency of Use:	Once per session or when re-requested.
Flow of Events:	1. System prompts the User for location permission. 2. User accepts or denies the request. 3. If accepted, System retrieves coordinates and returns them to the parent use case..
Alternative Flows:	**AF-1:** One-time permission 1. If user grants permission only for this session, the system re-prompts on the next attempt. **AF-2:** Manual location entry 1. If the user chooses not to allow access, the system provides an option to input postal code or area.
Exceptions:	**EX-1:** Permission denied at OS level ● The system records the denial and continues UC-3 without location. **EX-2:** GPS error

	● The system displays “Unable to detect location” and offers retry or manual entry.
Includes:	None
Special Requirements:	1. Must respect platform-specific permission handling (Android/iOS).
Assumptions:	None

4. Use Case 4 — View Home Page

Use Case ID:	UC-4		
Use Case Name:	View Home Page		
Created By:	Rajath	Last Updated By:	Rajath
Date Created:	3rd October 2025	Date Last Updated:	3rd October 2025

Actor:	● User (primary) ● System (supporting) ● Data Provider APIs / Sensors (optional, supporting)
Description:	When the User opens the Home page, the system displays recommended eateries based on proximity and profile tags. Recommendations are presented as the main list on the Home page
Preconditions:	● User authenticated ● Eatery catalog accessible.
Postconditions	● The Home page is displayed with a list of recommended eateries. ● The User can select an eatery to view details or take further actions.
Priority:	High
Frequency of Use:	Very high (every Home page visit).
Flow of Events:	1. User navigates to the Home page. 2. The system includes UC-6 Recommend Eateries to generate a curated list. 3. The system displays the recommended eateries on the Home page. 4. The User browses the recommendations. 5. The User may select an eatery which causes System to execute UC-12 View Eatery Page
Alternative Flows:	None

Exceptions:	None
Includes:	● UC-6 Recommend Eateries
Extended by:	● UC-12 View Eatery Page
Special Requirements:	Home page should render results in under 0.5 seconds after UC-6 completes.
Assumptions:	

5. Use Case 5 — Get Directions to Hawker Centre (as <<extend>> UC-3, UC-12)

Use Case ID:	UC-5		
Use Case Name:	Get Directions to Hawker Centre		
Created By:	Rajath Krishna	Last Updated By:	Rajath Krishna
Date Created:	30th August 2025	Date Last Updated:	2nd September 2025

Actor:	● User ● OneMap API (supporting system) ● Device Location Services (supporting system, optional)
Description:	This use case allows a User to request directions to a selected hawker centre or stall. The system determines the starting point using the device's location (with permission) or allows manual input. The OneMap API then returns optimal routes with multiple modes of transport, ETA and distance to the user.
Preconditions:	● Destination is known. ● Routing API (OneMap) is available.
Postconditions	● Optimal route is displayed with mode, ETA, and distance.
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. User selects **Get Directions** on the stall or hawker centre detail page or the search page. 2. System prompts the User to enter a starting address (using UC-3.I1 Request Current Location) or postal code. 3. User enters a valid starting point. 4. System calls the OneMap API to generate optimal route to the selected destination 5. OneMap API returns the optimal route. 6. System displays that route using **UC5.I1 - View Directions on Map**

Alternative Flows:	**AF-1:** Current location used as start 1. User grants location permission. 2. System autofills current location as origin. **AF-2:** Mode of transport changed 1. User switches travel mode (e.g., walking → bus). 2. System recalculates and shows a new route.
Exceptions:	**EX-1:** Invalid starting point ● If the provided address is invalid, the system prompts the user to correct it. **EX-2:** Routing service error ● If OneMap fails, the system displays “Directions unavailable” and suggests retrying later.
Includes:	● UC-5.I1 View Directions on Map ● UC 3.I1 - Request Current Location
Extends:	● UC-3 Search Healthier Eateries ● UC-12 View Eatery Page
Special Requirements:	1. Must respect OneMap’s API usage limits. 2. Cached routes should be reused to reduce calls.
Assumptions:	OneMap API consistently provides valid route data.

5.1. Use Case 5.I1 — View Directions on Map

Use Case ID:	UC-5.I1		
Use Case Name:	View Directions on Map		
Created By:	Rajath Krishna	Last Updated By:	Rajath Krishna
Date Created:	30th August 2025	Date Last Updated:	2nd September 2025

Actor:	● User (primary) ● System (supporting)
Description:	After the optimal route is generated, the system displays this route on a map. The user sees start-to-destination path with ETA and distance
Preconditions:	● A route is already generated in UC-5.
Postconditions	● User can see the selected route plotted on the map.
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. UC-5 generates the optimal route. 2. System opens a lightweight MapUI with the route overlaid. 3. User can view ETA and distance.
Alternative Flows:	None
Exceptions:	**EX-1:** Route overlay failure ● If the route cannot be drawn on the map, the system shows error.
Includes:	None
Special Requirements:	1. Route display must render in under 1 second after selection.
Assumptions:	None

6. Use Case 6 — Recommend Eateries

Use Case ID:	UC-6		
Use Case Name:	Recommend Eateries		
Created By:	Hanzhi	Last Updated By:	Rajath
Date Created:	31st August 2025	Date Last Updated:	3rd October 2025

Actor:	● User (primary) ● System (supporting)
Description:	The system generates a list of recommended eateries tailored to the User. Recommendations are based on proximity (if location access is granted), dietary tags, and fallback popularity when preferences or location are unavailable.
Preconditions:	● User is authenticated. ● Eatery catalog is accessible.
Postconditions	● A curated list of recommended eateries is returned to the calling use case (UC-4).
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The system requests current location if not requested already (UC-3.I1 Request Current Location). 2. The system filters eateries based on proximity to the User's location. 3. The system applies the User's dietary tags or preferences to further refine the list. 4. The system ranks the filtered list based on: ○ Distance (closest first) ○ Relevance to tags/preferences ○ Default weights for healthy choice status 5. The system compiles the final curated list. 6. The system returns this list to UC-4 (View Home Page).

Alternative Flows:	**AF-1: Location access denied** 1. User denies location permission. 2. The system falls back to showing a default set of trending or popular eateries. **AF-2: No matching eateries** 1. The system finds no eateries that match dietary tags/preferences. 2. The system displays “No recommendations found” and defaults to trending eateries. **AF-3: Location fetch timeout** 1. The system cannot obtain location within a set time. 2. The system prompts the User to manually enter a postal code or area.
Exceptions:	**EX-1: Eatery catalog unavailable** ● If the catalog cannot be accessed, the system shows “Recommendations unavailable” and logs the error.
Includes:	● UC-3.I1 Request Current Location
Special Requirements:	1. Recommendations must be generated in under 0.5 seconds. 2. Must work even with no preferences (fallback = default list).
Assumptions:	Basic scoring model (dietary tags + distance + popularity) is implemented

7. Use Case 7 — Manage Reviews

Use Case ID:	UC-7		
Use Case Name:	Manage Reviews		
Created By:	Hanzhi	Last Updated By:	Hanzhi
Date Created:	31th August 2025	Date Last Updated:	2nd September 2025

Actor:	● User ● Admin ● System
Description:	Allow users to submit, edit, delete, and flag reviews for stalls, while displaying aggregated ratings.
Preconditions:	● User authenticated. ● UC-12 View Eatery Page is running
Postconditions	● Reviews are submitted, edited, or deleted as requested. ● Aggregate rating is updated.
Priority:	High
Frequency of Use:	Medium–High
Flow of Events:	1. User opens a stall's details page. 2. System shows existing reviews and current aggregate rating. 3. User chooses one of: **Submit Review, Edit Review, Delete Review, or View Reviews**. 4. System validates the chosen action. 5. System updates reviews and recalculates aggregates. 6. System confirms the outcome (posted/updated/deleted) and refreshes the list.
Alternative Flows:	**AF-1:** Review flagged 1. User or admin flags a review. 2. System hides it until moderation is complete.

Exceptions:	**EX-1:** Duplicate review ● If the user attempts to submit another review for the same stall, the system blocks it and suggests editing the existing review. **EX-2:** Inappropriate content ● If the review includes disallowed material, the system blocks submission and requests revision. **EX-3:** Technical error ● If saving fails, the system keeps a draft locally and prompts the user to retry.
Includes:	● UC-7.I1 – Update Aggregate Ratings
Extended by:	● UC-7.E1 – Submit Review ● UC-7.E2 – Edit Review ● UC-7.E3 – Delete Review
Special Requirements:	1. Content policy must be enforced automatically (basic profanity filter) 2. Ratings must use a consistent scale across the application.
Assumptions:	One review per user per stall is allowed.

7.1. Use Case 7.I1 — Update Aggregate Ratings

Use Case ID:	UC-7.I1		
Use Case Name:	Update Aggregate Ratings		
Created By:	Hanzhi	Last Updated By:	Hanzhi
Date Created:	30th August 2025	Date Last Updated:	2nd September 2025

Actor:	System
Description:	System recalculates and updates stall's aggregate rating after changes.
Preconditions:	● A review has been added, edited, or deleted.
Postconditions	● Aggregate rating and review count are updated.
Priority:	High
Frequency of Use:	Every review update
Flow of Events:	1. System recomputes aggregate rating and review count. 2. System updates the stall details page.
Alternative Flows:	**AF-1:** Weighted average 1. If the system is configured for weighted averages (e.g., recent reviews weigh more), the calculation is adjusted accordingly.
Exceptions:	**EX-1:** Computation failure ● If the calculation fails, the previous aggregate is retained and the error is logged.
Includes:	None
Extends:	● UC-7 Manage Reviews
Special Requirements:	1. Calculation must be atomic to ensure no inconsistencies during updates.
Assumptions:	Review data is always available in the database.

7.2. Use Case 7.E1 — Submit Review (as <<extend>> of UC-7)

Use Case ID:	UC-7.E1		
Use Case Name:	Submit Review		
Created By:	Hanzhi	Last Updated By:	Hanzhi
Date Created:	31th August 2025	Date Last Updated:	2nd September 2025

Actor:	● User ● System
Description:	The user submits a review for a stall with required scores (Health, Hygiene) and optional text/photos.
Preconditions:	● User is logged in. ● User has not already reviewed the stall.
Postconditions	● Review is stored in the database. ● Aggregate rating updated.
Priority:	High
Frequency of Use:	Medium
Flow of Events:	1. User selects **Submit Review**. 2. System prompts for required inputs (e.g., rating). 3. User provides required rating (and optionally adds text or a photo). 4. System validates the input. 5. On success, System saves the review and invokes the included use case **UC-7.I1 Update Aggregate Ratings** 6. System shows “Review posted.”
Alternative Flows:	**AF-1:** Save as draft 1. User chooses to save review as draft. 2. System stores draft for later posting. **AF-2:** Text-only review 1. User submits only ratings and text without photos. 2. System accepts and posts review.

Exceptions:	**EX-1:** Image upload failure ● If photo upload fails, the review is saved without images, and the user is prompted to retry uploading later.
Includes:	● UC-7.I1 – Update Aggregate Ratings
Extends:	● UC-7 Manage Reviews
Special Requirements:	1. System must support multimedia reviews (images, text).
Assumptions:	Users submit genuine reviews (manual moderation handles exceptions).

7.3. Use Case 7.E2 — Edit Review (as <<extend>> of UC-7)

Use Case ID:	UC-7.E2		
Use Case Name:	Edit Review		
Created By:	Hanzhi	Last Updated By:	Hanzhi
Date Created:	31th August 2025	Date Last Updated:	2nd September 2025

Actor:	● User ● System
Description:	User updates their existing review.
Preconditions:	● Review exists and belongs to the user.
Postconditions	● Review is updated and aggregate rating recalculated.
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. User selects **Edit Review** 2. System retrieves existing Review 3. User updates rating/text/photo. 4. System validates and saves changes. 5. System retains older version in history 6. System invokes **UC-7.I1 Update Aggregate Ratings** and shows “Review updated.”
Alternative Flows:	None
Exceptions:	**EX-1:** Edit window expired ● If editing is only allowed within a time window, the system blocks changes and suggests posting a new review.
Includes:	● UC-7.I1 – Update Aggregate Ratings
Extends:	● UC-7 Manage Reviews
Special Requirements:	1. Edits must be logged with timestamp for transparency.
Assumptions:	Editing rights are limited to the original author.

7.4. Use Case 7.E3 — Delete Review (as <<extend>> of UC-7)

Use Case ID:	UC-7.E3		
Use Case Name:	Delete Review		
Created By:	Hanzhi	Last Updated By:	Hanzhi
Date Created:	31th August 2025	Date Last Updated:	2nd September 2025

Actor:	• User • System
Description:	User removes their review.
Preconditions:	• Review exists and belongs to the user.
Postconditions	• Review is updated and aggregate rating recalculated.
Priority:	Medium
Frequency of Use:	Low
Flow of Events:	1. User selects **Delete Review** and confirms. 2. System hides review from public but retains it for audit. 3. System invokes **UC-7.I1 Update Aggregate Ratings** and shows “Review deleted.”
Alternative Flows:	None
Exceptions:	**EX-1:** Legal hold • If a review is under investigation or flagged for compliance, deletion is blocked.
Includes:	• UC-7.I1 – Update Aggregate Ratings
Extends:	• UC-7 Manage Reviews
Special Requirements:	1. Delete must require explicit confirmation to prevent accidental removal.
Assumptions:	Only the review owner or an admin can delete.

8. Use Case 8 — Edit Dietary Tags

Use Case ID:	UC-8		
Use Case Name:	Edit Dietary Tags		
Created By:	Girija	Last Updated By:	Girija
Date Created:	31th August 2025	Date Last Updated:	2nd September 2025

Actor:	● Admin (primary) ● System (supporting)
Description:	The Admin updates or corrects dietary tags (e.g., Halal, vegetarian, healthy choice) for a stall to maintain accuracy in search and recommendations.
Preconditions:	● Admin is logged in.
Postconditions	● Dietary tags are updated and applied to searches and recommendations.
Priority:	Medium
Frequency of Use:	Low-Medium
Flow of Events:	1. Admin logs in with elevated privileges. 2. Admin selects a stall record. 3. Admin edits dietary tags (add/remove/update). 4. System saves the changes 5. System updates the stall listing and recommendations that rely on these tags.
Alternative Flows:	**AF-1:** Bulk edit 1. Admin selects multiple stalls and applies the same tag update to all. **AF-2:** Pending approval 1. Admin updates tags. 2. Changes are queued for secondary approval before publishing.

Exceptions:	**EX-1:** Conflicting tags ● If incompatible tags are selected (e.g., “Halal” + “Contains Pork”), system blocks the change and requests correction. **EX-2:** Save error ● If the update cannot be saved, the system rolls back to the last stable state.
Includes:	None
Special Requirements:	1. All edits must be logged with timestamp, admin ID, and old/new tag values.
Assumptions:	Tag vocabulary is standardized.

9. Use Case 9 — Earn Points

Use Case ID:	UC-9		
Use Case Name:	Earn Points		
Created By:	Girija	Last Updated By:	Girija
Date Created:	31th August 2025	Date Last Updated:	2nd September 2025

Actor:	● System
Description:	The User earns points when performing eligible actions (e.g., submitting a review, and more to be added actions). The System updates the User's points balance.
Preconditions:	● User account exists. ● Points rules are configured.
Postconditions	● Points are added to user's balance and logged.
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. An eligible action completes (e.g., **UC-7.E1 Submit Review**). 2. System determines the points to award based on the type of action completed. 3. System includes **UC-9.I1 Update Points** to add points to the User's balance. 4. System stores a simple points transaction record into the database 5. System optionally notifies the User (e.g., "+10 points earned").
Alternative Flows:	None
Exceptions:	**EX-1:** Balance update error ● If the update fails due to concurrency or server issues, the system retries; if persistent, the event is queued. **EX-2:** Missing rule

	● If no rule matches the action, no points are awarded and the event is logged. **EX-3:** Duplicate action ● If User repeats same action quickly, System blocks duplicate points and logs attempt.
Includes:	● UC-9.I1 Update Points
Extends:	● UC-7.E1 Submit Review
Special Requirements:	1. Balance updates must be atomic and concurrency-safe. 2. Every transaction must have a unique ID for audit.
Assumptions:	Points system uses a single currency.

9.1. Use Case 9.I1 — Update Points

Use Case ID:	UC-9.I1		
Use Case Name:	Update Points		
Created By:	Girija	Last Updated By:	Girija
Date Created:	31th August 2025	Date Last Updated:	2nd September 2025

Actor:	● System
Description:	System updates the User's balance by either adding or deducting points, depending on the calling UC
Preconditions:	● Valid user account exists.
Postconditions	● User's balance is updated and transaction logged.
Priority:	High
Frequency of Use:	Each time points are awarded or deducted.
Flow of Events:	1. System retrieves current balance from the database 2. System adds or subtracts points as specified. 3. System validates the new balance (rejects if negative). 4. System saves the updated balance into database and logs the transaction.
Alternative Flows:	**AF-1:** Scheduled update (delayed transaction) 1. If the points system is temporarily unavailable, the system queues the update request. 2. The queued transaction is processed once the service is back online. 3. User's balance is updated, and the transaction is logged with a "delayed" status.
Exceptions:	**EX-1:** Negative balance attempt ● If deduction would make balance negative, the system blocks the transaction.
Includes:	None
Special Requirements:	1. Transaction history must be immutable.
Assumptions:	None

10. Use Case 10 — Redeem Reward

Use Case ID:	UC-10		
Use Case Name:	Redeem Rewards		
Created By:	Yoheshvaran	Last Updated By:	Yoheshvaran
Date Created:	31th August 2025	Date Last Updated:	2nd September 2025

Actor:	● User (primary) ● System (supporting)
Description:	The User redeems accumulated points for discounts or vouchers. The System deducts points and issues the reward.
Preconditions:	● User is authenticated. ● User has sufficient points.
Postconditions	● Reward is issued and points deducted. ● If unsuccessful, no deduction occurs.
Priority:	High
Frequency of Use:	Medium
Flow of Events:	1. User opens the Rewards section. 2. System shows balance and available rewards. 3. User selects a reward. 4. System verifies sufficient points. 5. System includes **UC-9.I1 Update Points** to deduct points. 6. System issues the reward and updates balance.
Alternative Flows:	**AF-1:** Insufficient points 1. User lacks points for selected reward. 2. System blocks redemption 3. System prompts user to select different reward **AF-2:** Reward Unavailable 1. Reward is out of stock / expired 2. System blocks redemption 3. System prompts user to select different reward.

Exceptions:	**EX-1:** Insufficient points ● Redemption is blocked and system displays how many more points are needed. **EX-2:** Reward unavailable ● If reward is out of stock or expired, the system blocks redemption and restores the user's state. **EX-3:** Deduction success but reward issue fails ● If points are deducted but reward issuance fails, the system automatically rolls back the deduction and logs the error.
Includes:	● UC-9.I1 Update Points (deduct from balance)
Special Requirements:	1. Deduction and reward issuance must succeed together (all-or-nothing). 2. Rewards must have unique identifiers (e.g., voucher codes).
Assumptions:	Rewards catalog is regularly updated by Admin.

11. Use Case 11 — Estimate Crowd Levels

Use Case ID:	UC-11		
Use Case Name:	Estimate Crowd Levels		
Created By:	Yoheshvaran	Last Updated By:	Rajath
Date Created:	30th August 2025	Date Last Updated:	3rd November 2025

Actor:	● User (primary) ● System (supporting) ● Data Provider APIs / Sensors (optional, supporting)
Description:	The User views estimated crowd levels for a hawker centre or stall. Since live data is unavailable, the System generates estimates using predefined rules based on time, day, and weather.
Preconditions:	● Eatery selected. ● Estimation rules configured.
Postconditions	● Estimated crowd level are displayed.
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. User selects a hawker centre or stall from the list. 2. System checks contextual factors (time of day, day of week, weather). 3. System applies estimation rules (e.g., peak hours = higher crowd). 4. System generates: Overall crowd level (Low / Medium / High) 5. System displays estimates with a label “Estimated based on patterns.”
Alternative Flows:	None
Exceptions:	**EX-1:** Rule engine error ● If estimation rules cannot be applied, the system hides the estimates and shows “No estimates available.” **EX-2:** Missing context data

	• If time or day information is missing, the system defaults to “Medium” crowd level.
Includes:	None
Special Requirements:	1. Estimates must include a timestamp and note that they are approximations.
Assumptions:	Historical data patterns reflect actual user traffic.

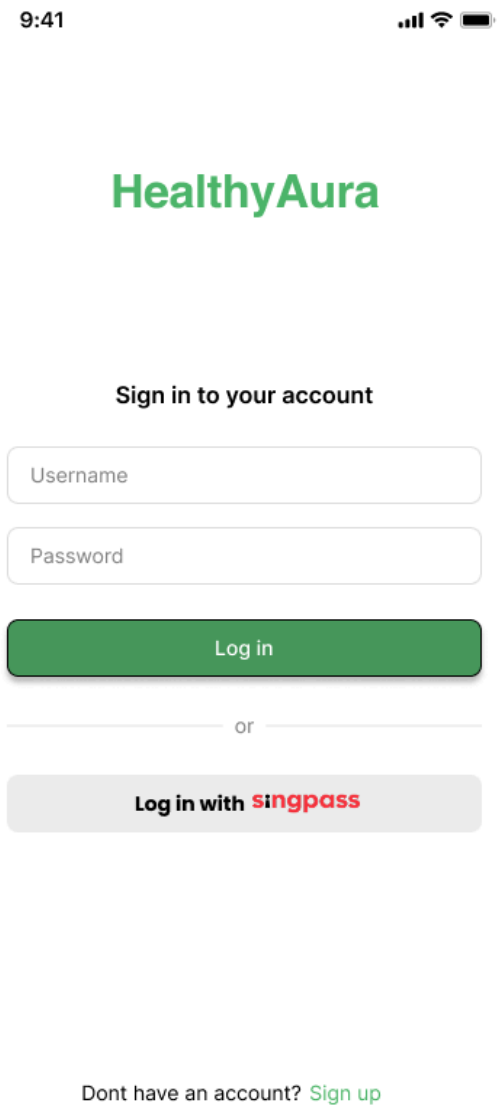
12. Use Case 12 — View Eatery Page

Use Case ID:	UC-12		
Use Case Name:	View Eatery Pages		
Created By:	Rajath	Last Updated By:	Rajath
Date Created:	26th September 2025	Date Last Updated:	3rd November 2025

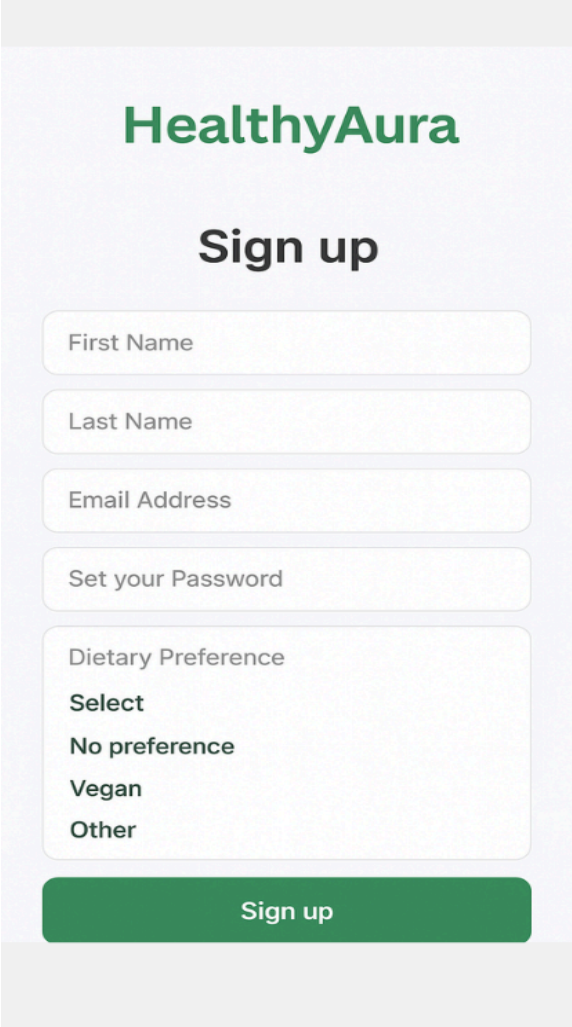
Actor:	● User (primary) ● System (supporting)
Description:	The User views detailed information about a specific eatery, including its name, address, tags (e.g., Halal, Vegan, Healthy), operating hours, reviews, and estimated crowd levels. This page serves as the main interface for users to explore an eatery in detail, access reviews, and get directions. The data displayed is fetched dynamically from the system's database and public APIs.
Preconditions:	● The user has selected an eatery from either the Search Healthier Eateries (UC-3) or View Home Page (UC-4). ● Eatery details are available in the system database.
Postconditions	● The system successfully displays detailed information about the selected eatery. ● The user may proceed to: ○ View crowd level estimates. ○ Read or submit reviews. ○ Request directions to the eatery.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User selects an eatery from the home page or search results. 2. System loads detailed eatery information (name, address, category, tags, image). 3. System retrieves additional data such as opening hours. 4. System executes UC-11 Estimate Crowd Levels to display the current crowd estimate. 5. System executes UC-7 Manage Reviews to display user reviews and ratings. 6. User can scroll through the eatery page to view reviews, crowd status, and other info. 7. User can click "Get Directions" to initiate UC-5 Get Directions to Hawker Centre.

Alternative Flows:	None
Exceptions:	**EX-1:** Data fetch failure ● The system displays “Unable to load details at the moment. Please try again later.” **EX-2:** Network connectivity loss ● The system caches previous eatery details if available and shows a warning banner (“Offline Mode: Data may be outdated”).
Includes:	● UC-11 Estimate Crowd Levels ● UC-7 Manage Reviews
Extended by:	● UC-5 Get Directions to Hawker Centre
Extends:	● UC-4 View Home Page ● UC-3 Search Healthier Eateries
Special Requirements:	Page should load in under 2 seconds for standard broadband connections.
Assumptions:	Eatery data (name, tags, address, etc.) exists and is regularly synchronized with external APIs.

5. UI Prototype (Initial Mockups)

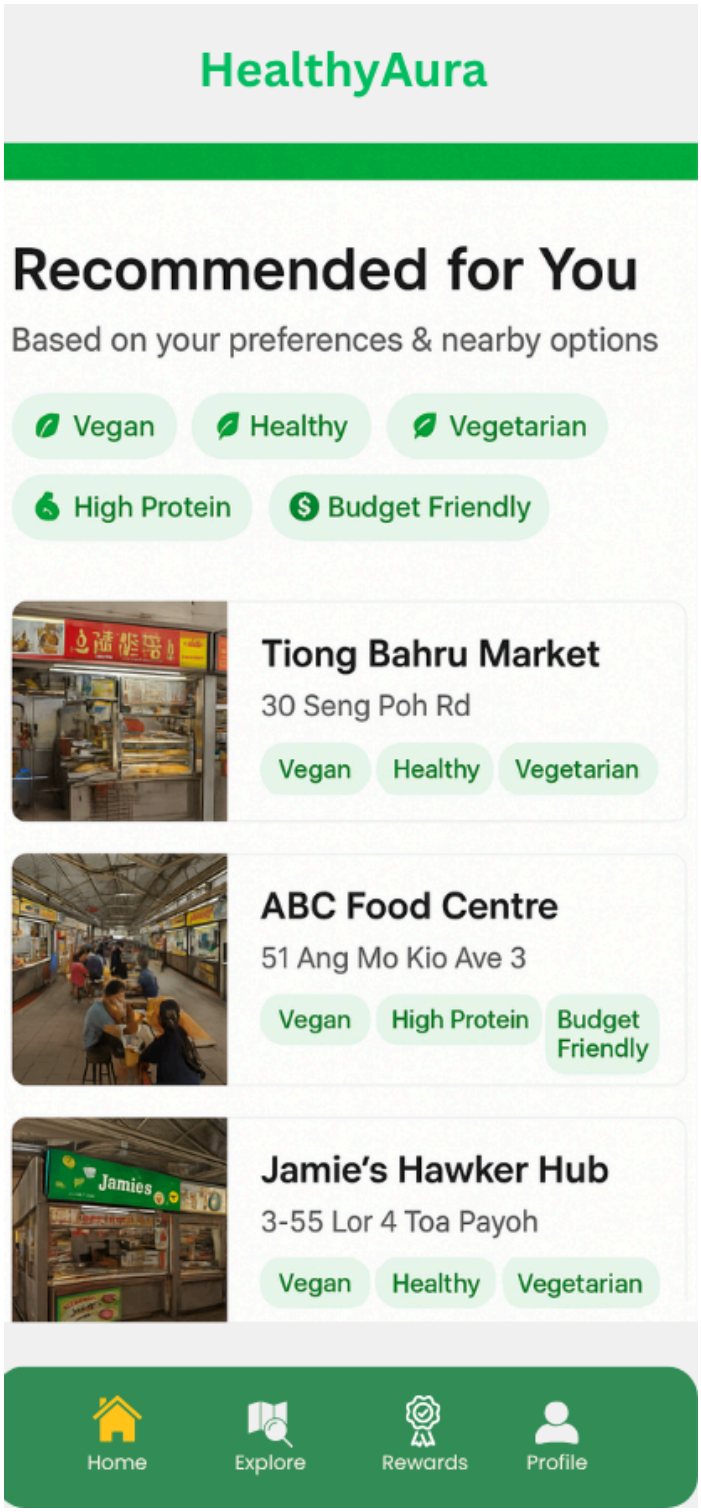


Description: Upon opening the app, the user will be displayed with the login screen. He or she can sign in with credentials or Singpass. If there is no account, they can create a new one by clicking on the sign button.

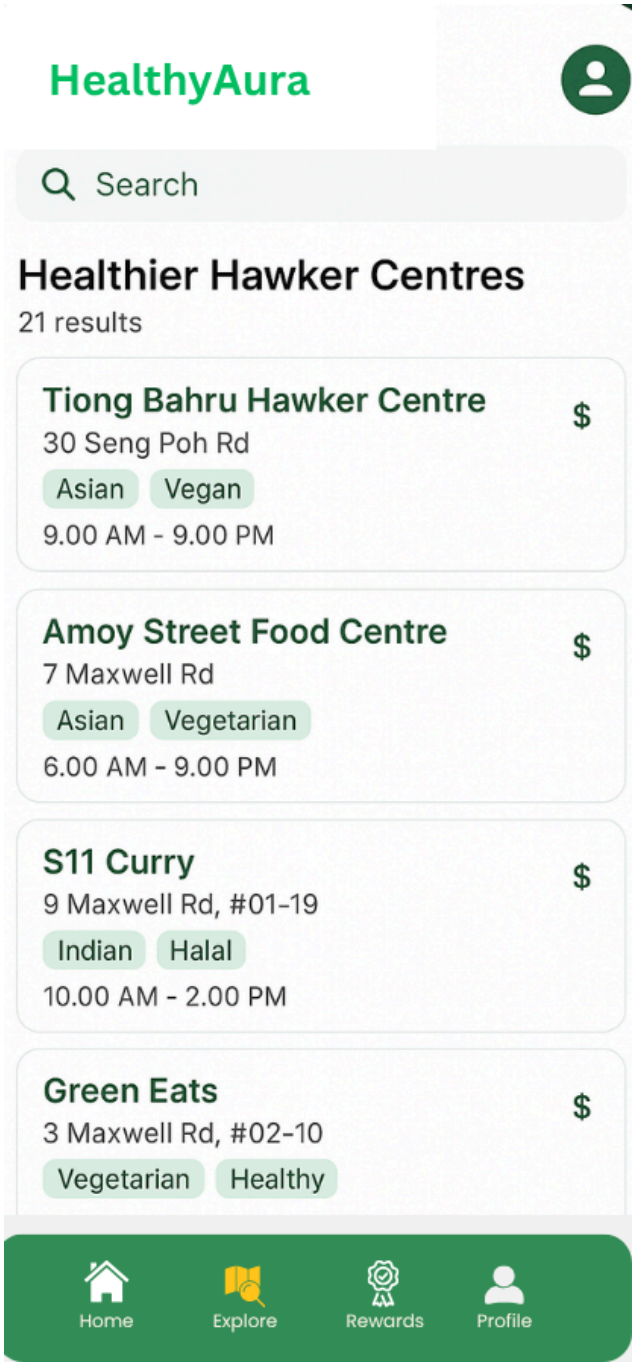


(INSERT SIGN UP UI HERE)

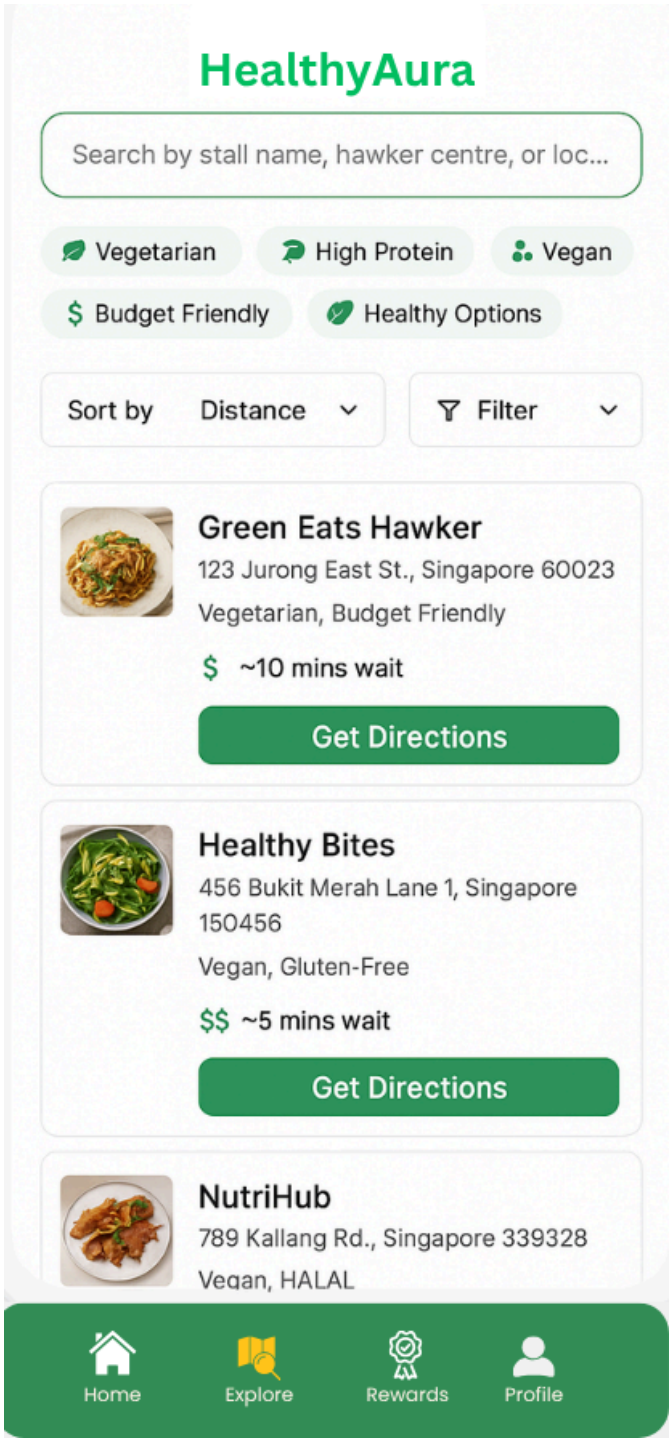
Description: Once “sign up" button is pressed, the user will be redirected to a sign up form where they can type in their personal information to create their profile including their preferences”



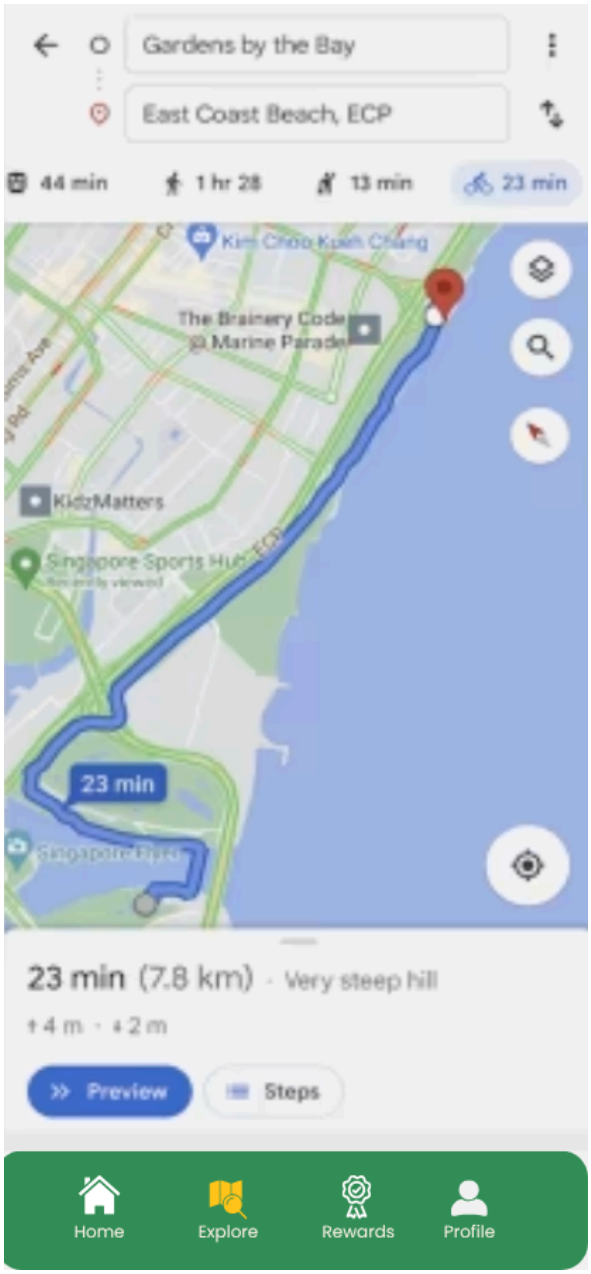
Description: The “home” page is the first to be displayed after logging in. It will be filled with a list of eateries recommended specially for the user based on their profile.



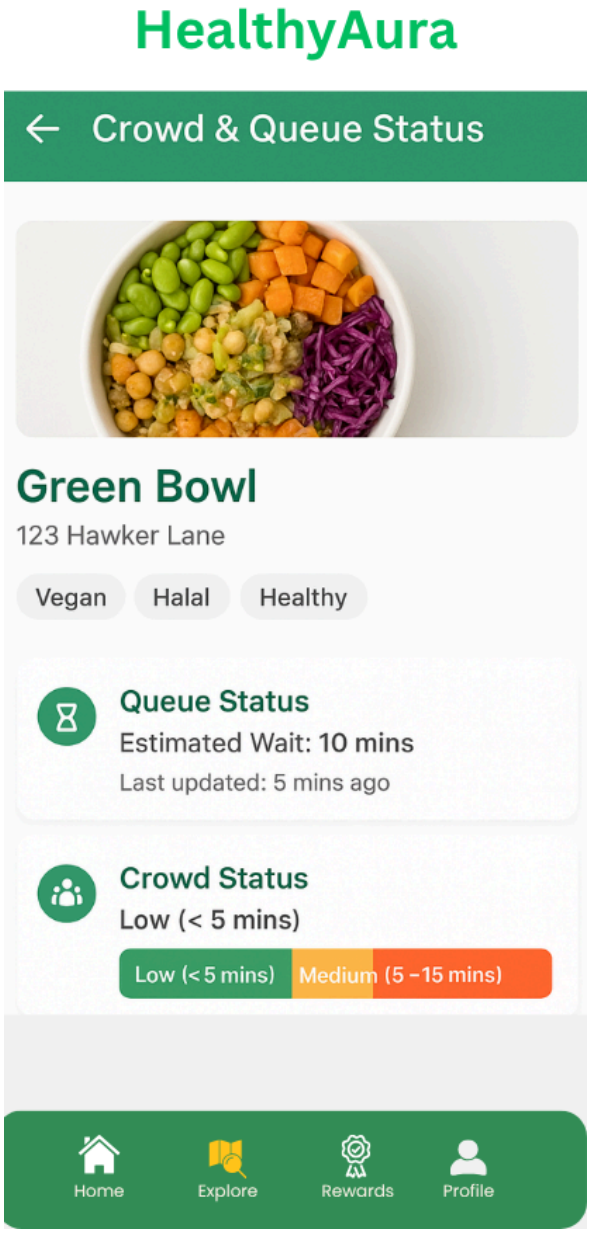
Description: If the user clicks on “explore”, the explore page will then be displayed with a search bar.This page allows the user to search a specific eatery that they are looking for and filter based on tags or distance.



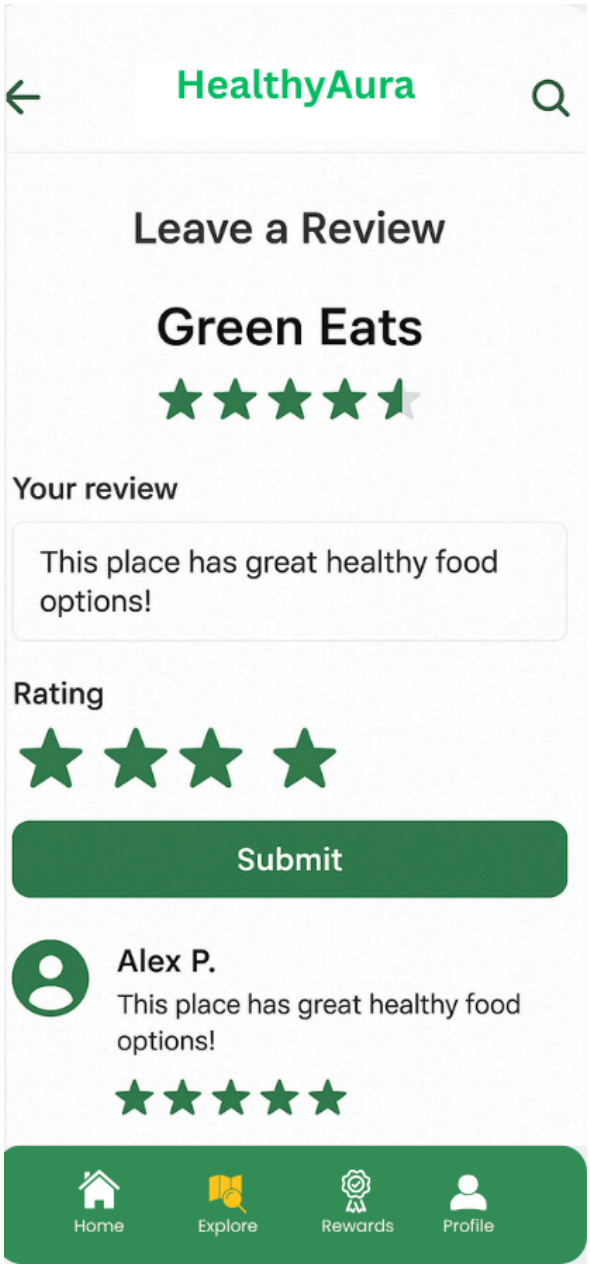
Description: The explore page also allows the user to click on the “get directions” button to see how to get to their selected destination or see how far it is.



Description: Once the “get directions” button is pressed, the application will display the map to the user with all the different routes available to go to their chosen destination.

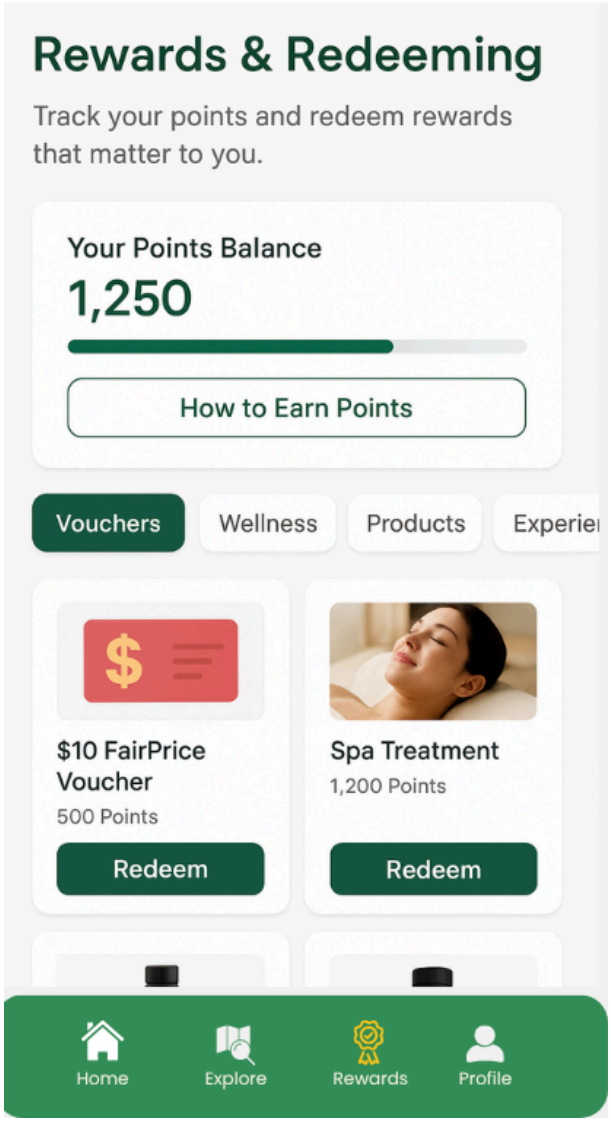


Description: If a user decides to click onto the details of the eatery, the app will display the address, eatery tags, queue status and crowd status of the chosen eatery.



Description: When the user clicks onto the “submit a review” displayed in the details page, they will be redirected to a submission form for them to fill up their review and give ratings. Their reviews will be displayed at the bottom upon successful submission.

HealthyAura



Description: The “Reward” page displays the catalog of rewards available for the user to redeem. It also displays the points balance for the user along with a guideline on how to earn points.

HealthyAura

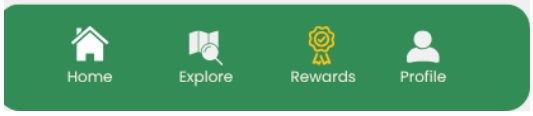
How to Earn Points

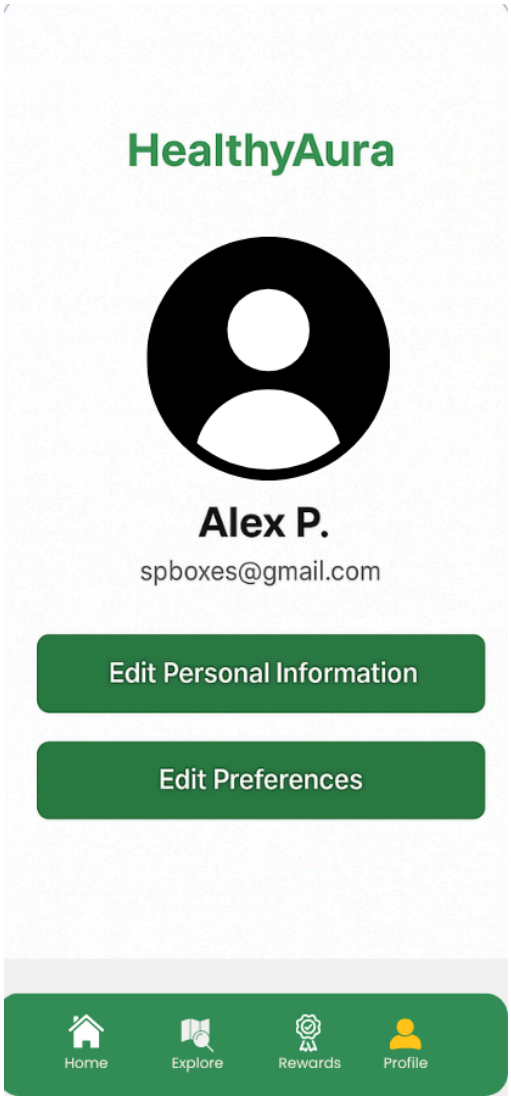
Make a booking

Complete your profile

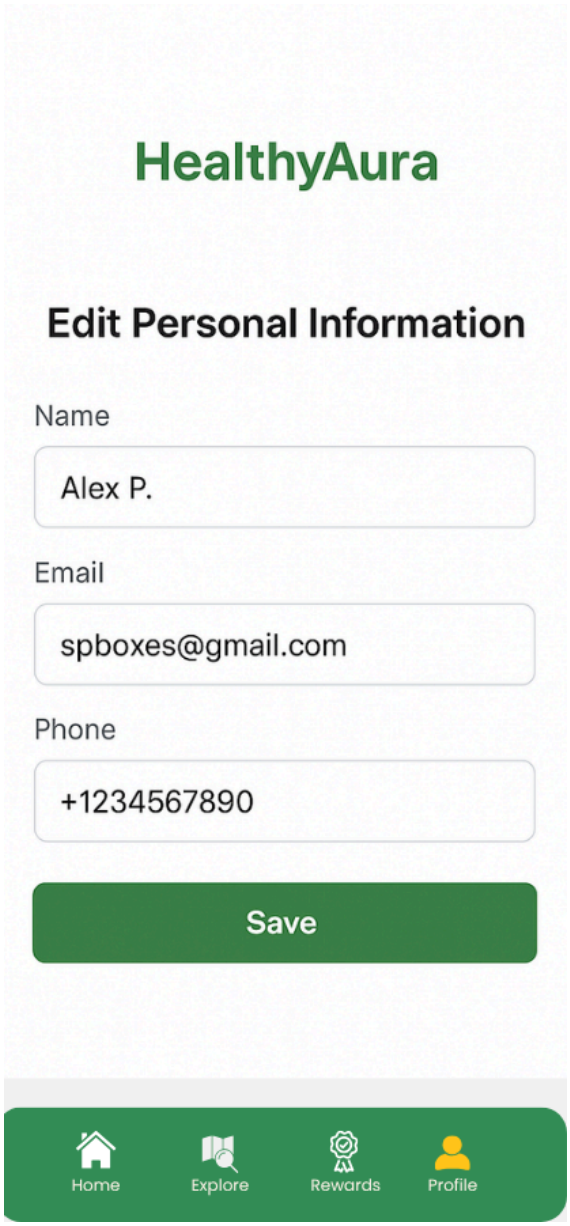
Leave a review

Refer a friend

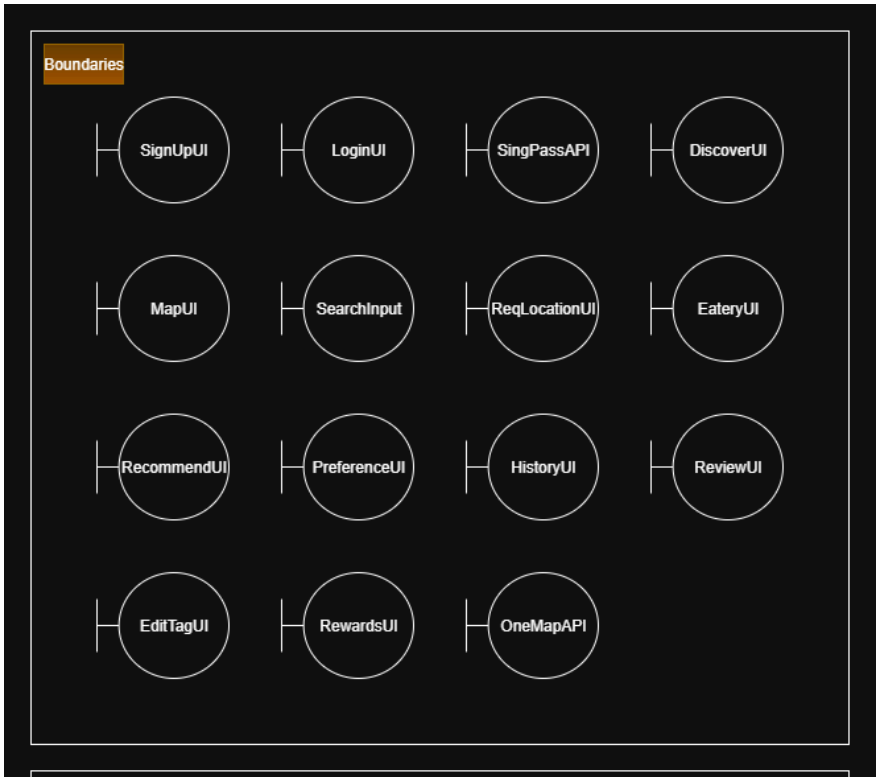


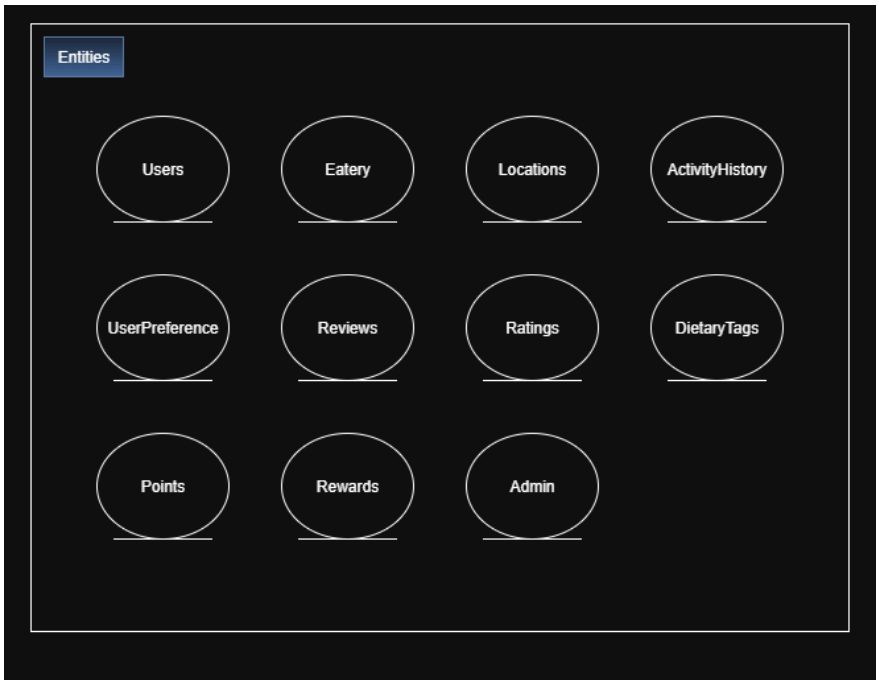


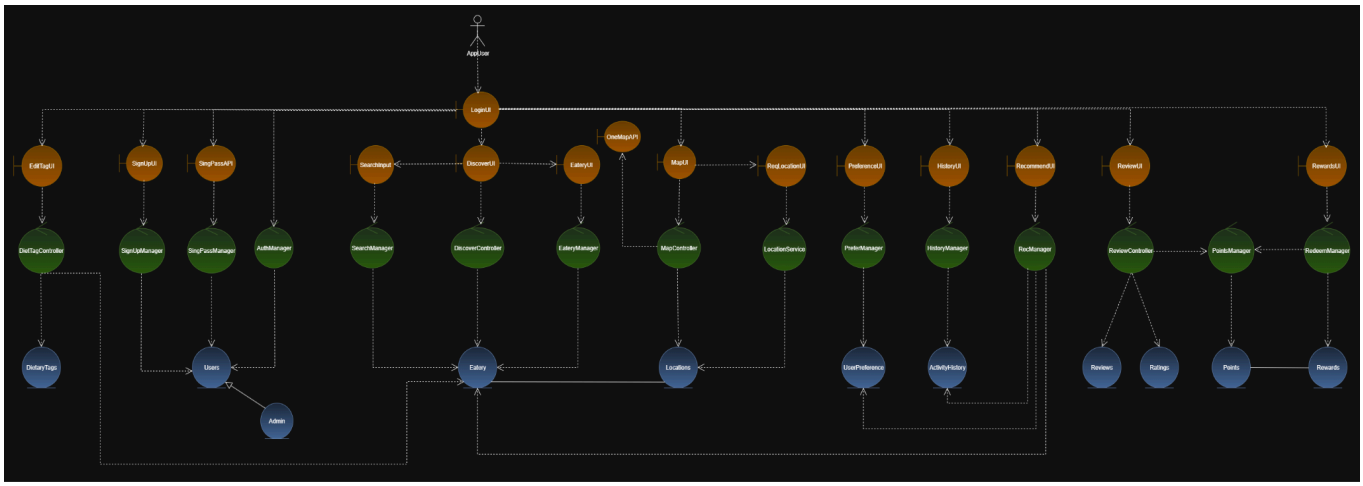
Description: The profile page will display current user profile information, he or she is able to edit their own personal information and their own preferences.


HealthyAura
Edit Personal Information
Name
Alex P.
Email
spboxes@gmail.com
Phone
+1234567890
Save
Home
Explore
Rewards
Profile

6. Boundary, Control and Entity Diagrams







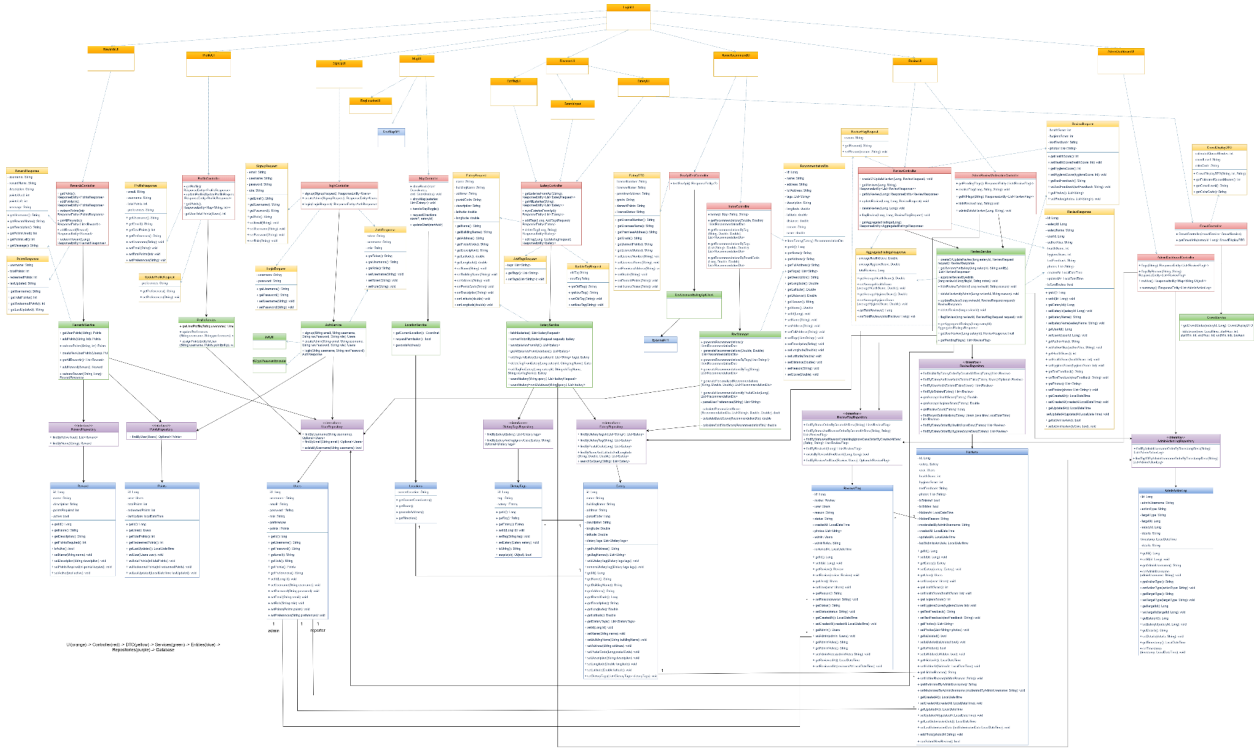
Key

Orange - Boundary Classes

Green - Control Classes

Blue - Entity Classes

7. Class Diagram

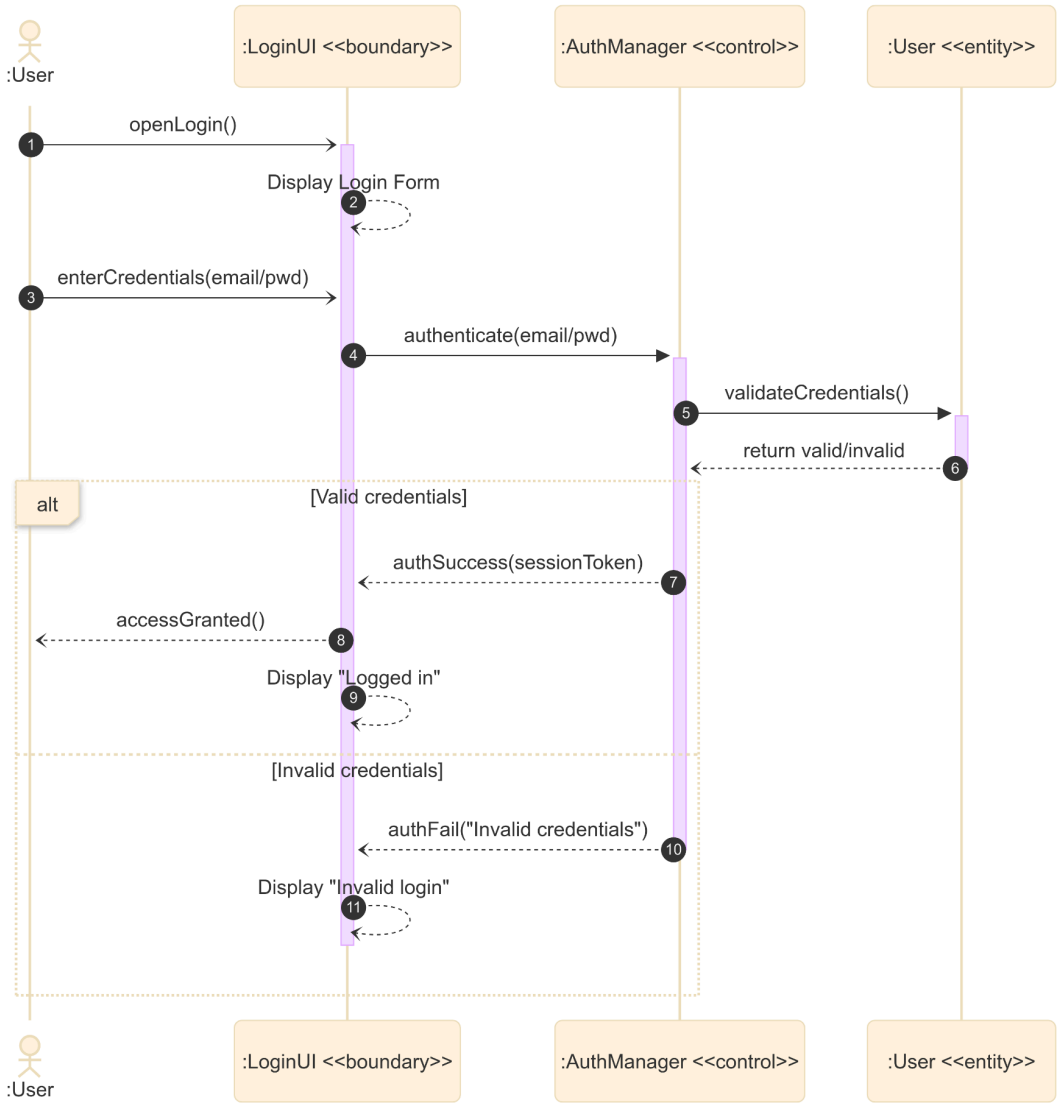


Key

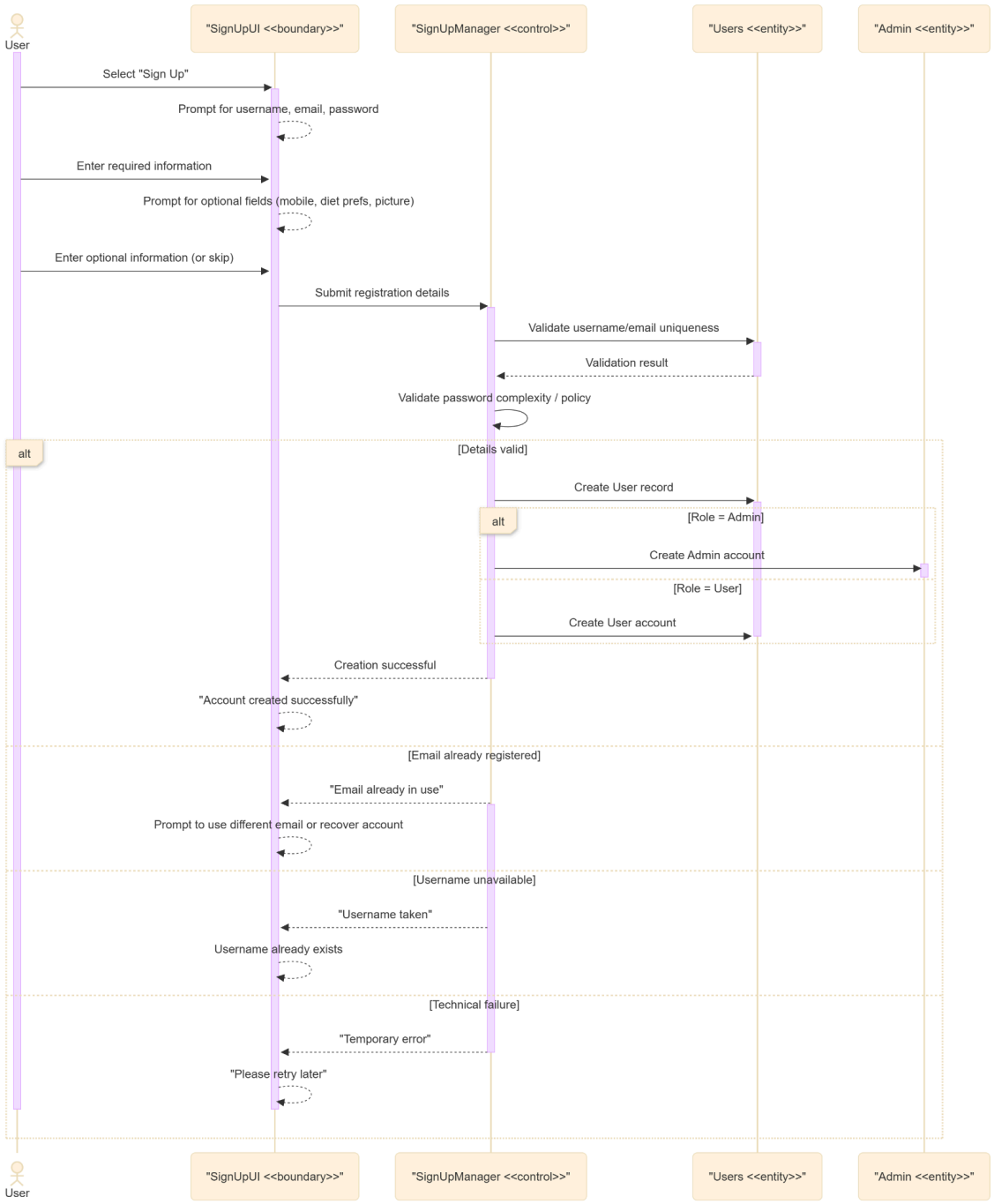
Orange - Boundary Classes
Red - Control Classes
Yello - DTO
Green - Services
Purple - Repositories
Blue - Entity Classes
Link to the diagram:
https://drive.google.com/file/d/1vxNG4teby5Paj8RcU6ncjTbrxJev1Bcv/view?usp=sharing

8. Sequence Diagrams

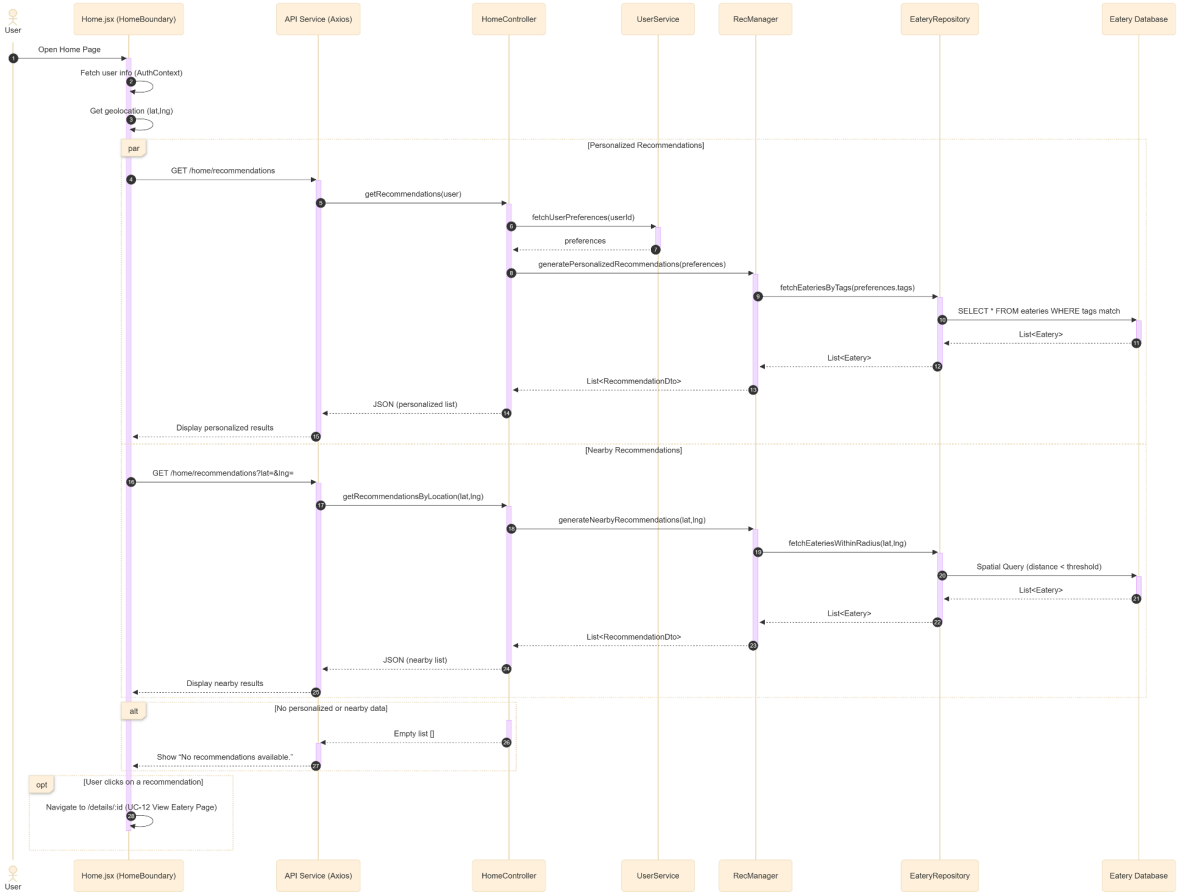
A. Login Page (UC-2)



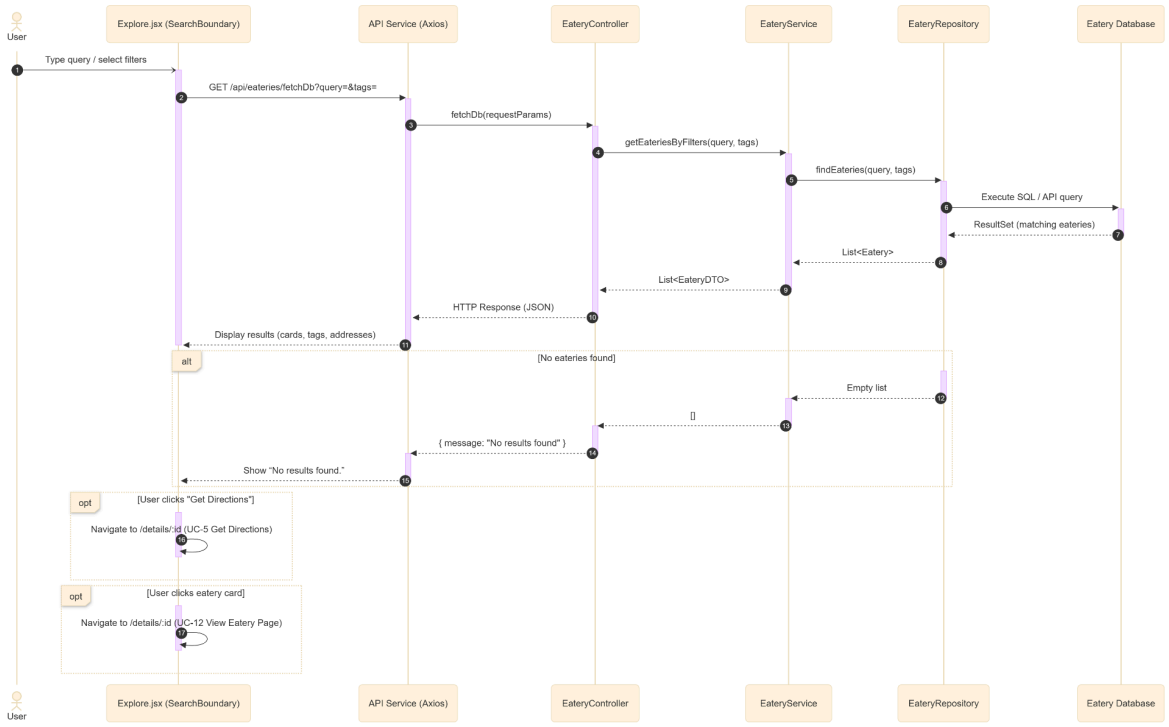
B. Sign Up Page (UC-1)



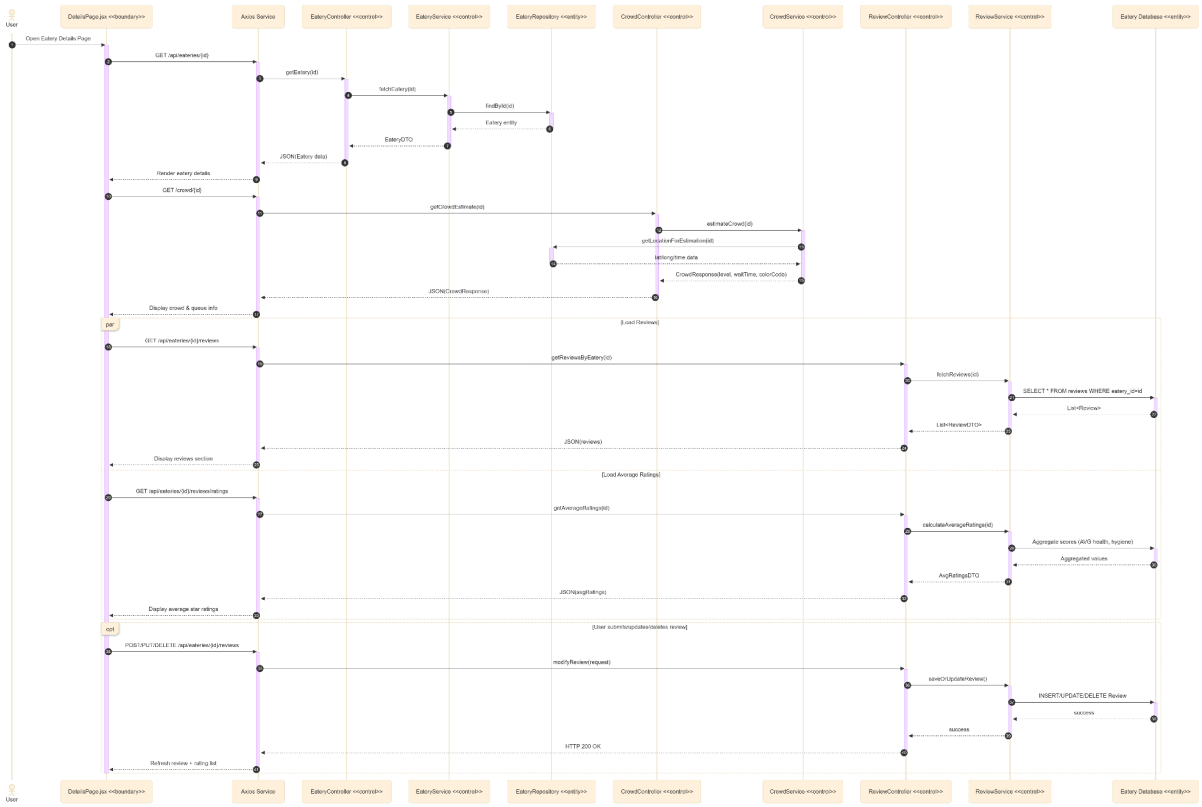
C. Home Page (UC-4)



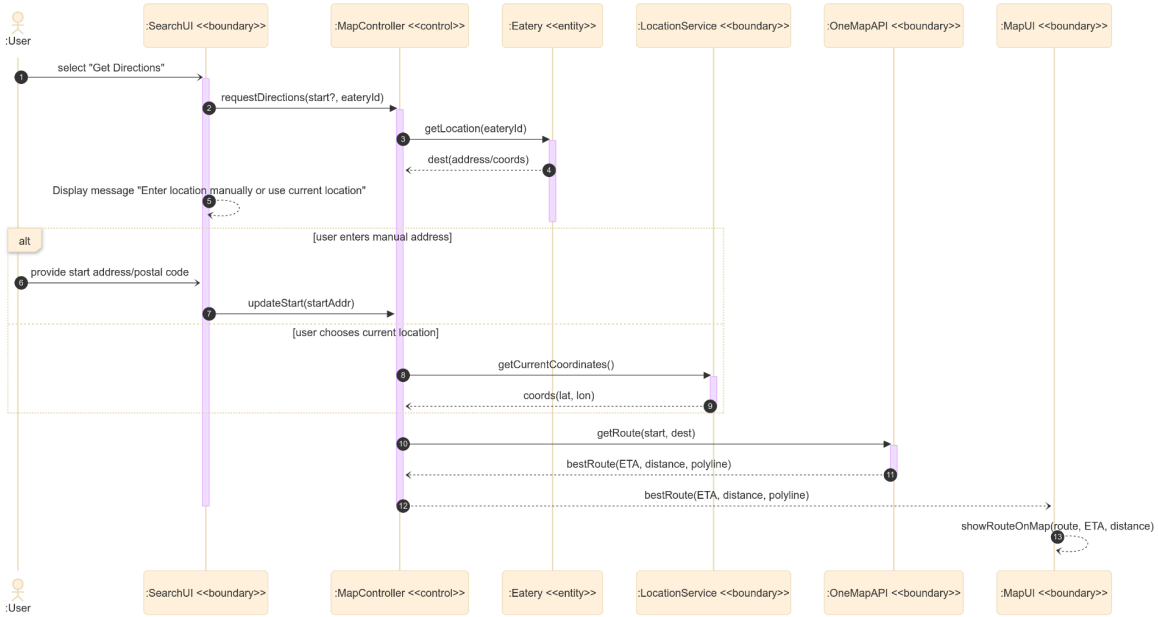
D. Search Page (UC-3)



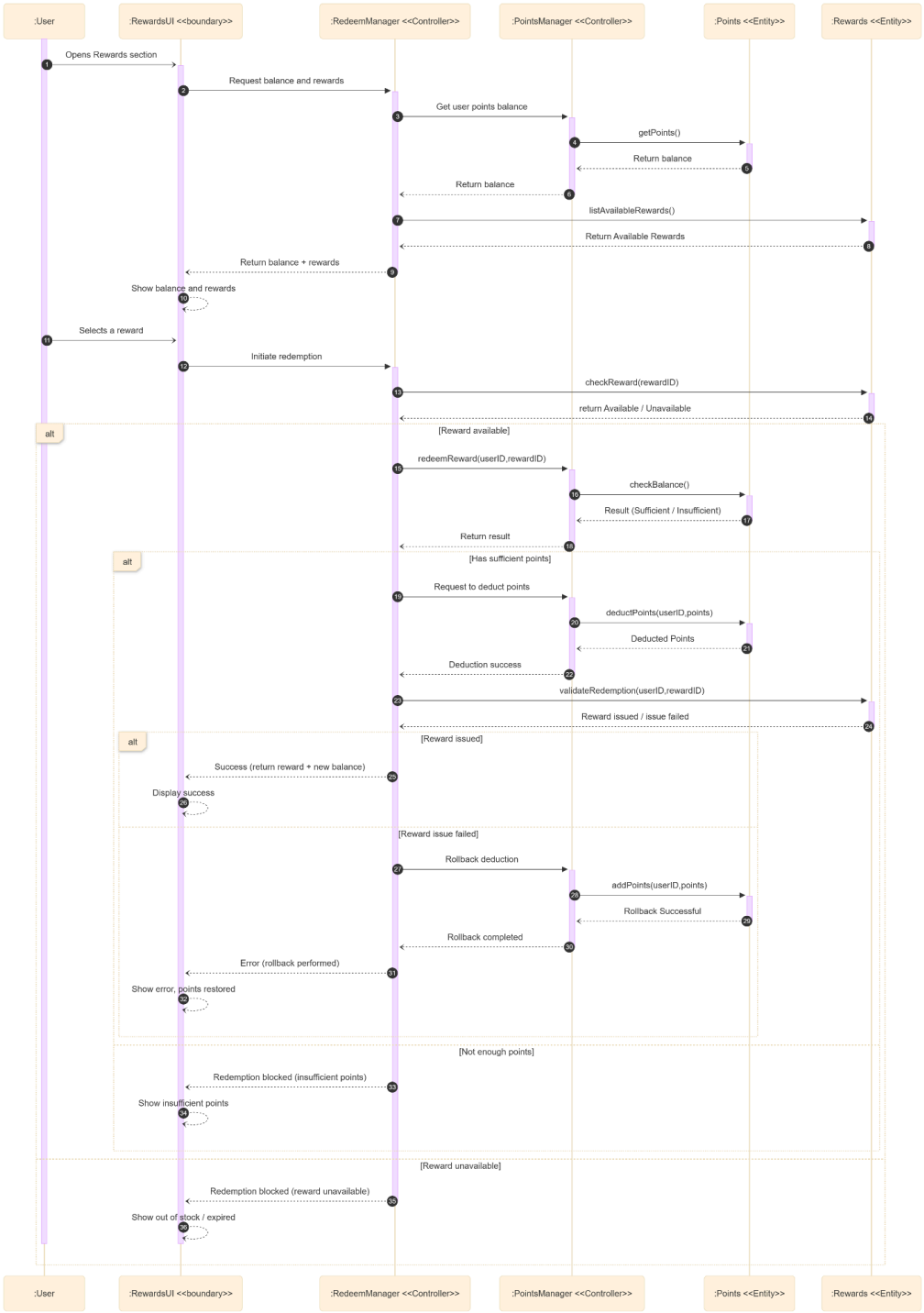
E. Eatery Page (UC-12 with UC-11and UC-7 included)



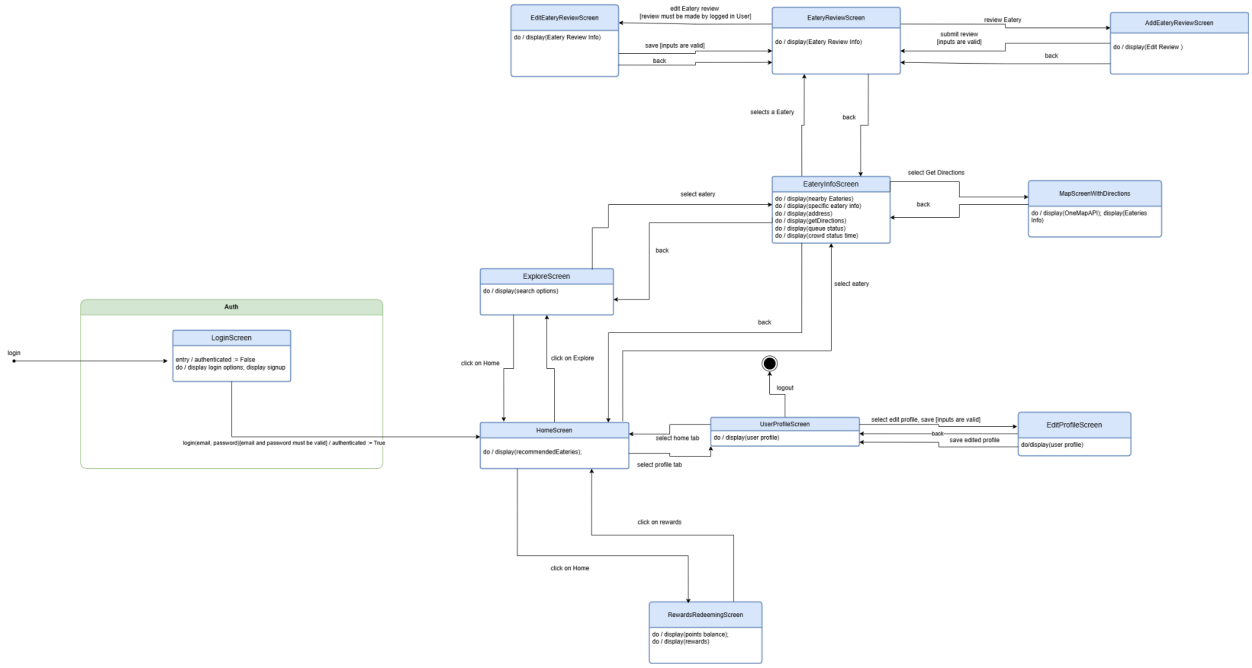
F. Get Directions (UC-5 with UC-5.I1 included)



G. Redeem Rewards Flow (UC-10)



9. Initial Dialog Map


EditEateryReviewScreen
do / display(Eatery Review Info)
edit Eatery review
[review must be made by logged in User]
save [inputs are valid]
back
EateryReviewScreen
do / display(Eatery Review Info)
review Eatery
submit review
[inputs are valid]
back
AddEateryReviewScreen
do / display(Edit Review)
selects a Eatery
back
EateryInfoScreen
do / display(nearby Eateries)
do / display(specific eatery info)
do / display(address)
do / display(getDirections)
do / display(queue status)
do / display(crowd status time)
select Get Directions
back
MapScreenWithDirections
do / display(OneMapAPI); display(Eateries Info)
select eatery
back
ExploreScreen
do / display(search options)
select eatery
back
Auth
LoginScreen
entry / authenticated := False
do / display login options; display signup
login
login(email, password)[email and password must be valid] / authenticated := True
click on Home
click on Explore
logout
HomeScreen
do / display(recommendedEateries);
select home tab
select profile tab
UserProfileScreen
do / display(user profile)
select edit profile, save [inputs are valid]
back
save edited profile
EditProfileScreen
do/display(user profile)
click on rewards
click on Home
RewardsRedeemingScreen
do / display(points balance);
do / display(rewards)

10. Traceability Matrix

Functional Requirement (ID)	Description	Related Use Case ID	Sequence Diagram	Implemented In (Code / Class)
FR 1	The system shall perform user authentication to allow Users to use the application. Includes account creation, login, and role assignment.	UC-1 Create Account UC-2 Login	Sequence Diagram – Login Page (UC-2) and Signup Page (UC-1)	`Auth.jsx`, `Profile.jsx`, `LoginController.java`, `ProfileController.java` `AuthService.java` `AuthContext.js`, `UserService.java`,
FR 2	The system shall allow Users to discover the location of healthier eateries through search, sorting, and filtering.	UC-3 Search Healthier Eateries (+ UC-3.I1 Request Current Location)	Sequence Diagram – Search Page (UC-3)	`Explore.jsx`, `EateryController.java`, `EateryService.java`, `EateryRepository.java`
FR 3	The system shall provide crowd estimation for Users based on contextual factors such as time and day.	UC-11 Estimate Crowd Levels	Sequence Diagram – UC-12 (View Eatery Page) with UC-11 included	`CrowdController.java`, `CrowdService.java`
FR 4	The system can direct Users to the location of their desired eatery using OneMap API.	UC-5 Get Directions (+ UC-5.I1 View Directions on Map)	Sequence Diagram – Get Directions (UC-5)	`Explore.jsx`, `MapUI` `EateryController.java`

FR 5	The system shall suggest food options based on user history and preferences.	UC-6 Recommend Eateries (<<include>> UC-4 View Home Page)	Sequence Diagram – Home Page (UC-4)	`Home.jsx`, `HomeController.java`, `RecManager.java`, `UserService.java`
FR 6	The system enables Users to share feedback and see aggregated ratings.	UC-7 Manage Reviews (+ UC-7.E1/E2/E3, UC-7.I1)	Sequence Diagram – UC-12 (View Eatery Page) with UC-7 included	`DetailsPage.jsx`, `ReviewController.java`, `ReviewService.java`
FR 7	The system shall allow Users to redeem rewards for using the application.	UC-9 Earn Points, UC-10 Redeem Rewards	Sequence Diagram – Rewards Flow (UC-10)	`Rewards.jsx`, `RewardController.java`, `RewardService.java`, `PointsRepository.java`
FR 8	The system shall allow authenticated Admins to manage content quality and verify information.	UC-8 Edit Dietary Tags	NA	`AdminDashboard.jsx` `AdminAuth.jsx`, `AdminModeration.jsx`, `AdminTagManager.jsx`, `AdminDashboardController.java`, `AdminLogController.java` , `AdminReviewModerationController.java`,