Software Requirements Specification

for

<FeedItForward>

Version 1.0 approved

Prepared by <Toh Jing Qiang, Toh Jing Hua,

Denise, Minze, Tommy, Pinyang>

<SC2006 - Team FeedItForward>

<19/11/2023>

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This document outlines the software requirements for the "FeedItFoward, Version 1.0". This SRS outlines the functionalities, features, and constraints of the platform, aiming to provide a clear and comprehensive framework for development. The scope of this SRS encompasses the entire digital platform, detailing the subsystems for user management, food donation coordination, and distribution logistics. This SRS is integral for guiding the development, testing, and maintenance of the platform, ensuring alignment with the project's objectives.

1.2 Document Conventions

This SRS follows standard documentation conventions to ensure clarity and consistency. Key conventions include:

- Font Styles: 'Arial' for body text, 'Times New Roman' for headers.
- Highlighting: Bold for key terms, italics for emphasis.
- Requirement Prioritization: Higher-level requirements have inherited priority levels, cascading down to detailed requirements. Each requirement statement is explicitly labeled with a priority level (High, Medium, Low).
- Requirement Identification: Each requirement is uniquely identified for easy reference.
- Versioning: Changes in requirements through different versions are tracked for historical reference and future revisions.

1.3 Intended Audience and Reading Suggestions

This document is structured to cater to a wide array of stakeholders and should be read differently depending on the reader’s role:

- **Project Managers and Stakeholders:** Begin with the *Introduction* to understand the purpose and scope, then proceed to the *Overall Description* for a high-level overview of the system.
- **Developers:** After the introductory sections, delve into the *System Features* for detailed descriptions of the platform's functionalities, and the *External Interface Requirements* for integration specifics.
- **Testers:** Focus on the *System Features* for test case development, and refer to the *Nonfunctional Requirements* for performance and security testing guidelines.

- **User Experience Designers:** The *User Interface* sections within the *External Interface Requirements* will be of particular interest, providing details on the interaction between the system and its users.
- **Technical Writers:** The entire document is relevant, with emphasis on *User Documentation* and *Assumptions and Dependencies* to ensure accurate and comprehensive user guides and help documentation.
- **Quality Assurance:** Refer to the *System Features* for functionality, *Other Nonfunctional Requirements* for performance benchmarks, and the *Other Requirements* for additional specifications.

All readers are encouraged to review the document in its entirety, as each section builds on the information presented in the preceding ones, providing a complete understanding of the FeedItFoward's requirements.

1.4 Product Scope

FeedItForward is a digital solution designed to bridge the gap between hawker food surplus and food scarcity in needy communities. Its primary goal is to connect local food vendors with surplus produce to families and individuals in need, thereby reducing food waste and enhancing community welfare. The platform will enable vendors to list available surplus food, allow recipients to request donations, and facilitate volunteers in logistics. This aligns with broader corporate objectives of social responsibility and sustainable community development.

1.5 References

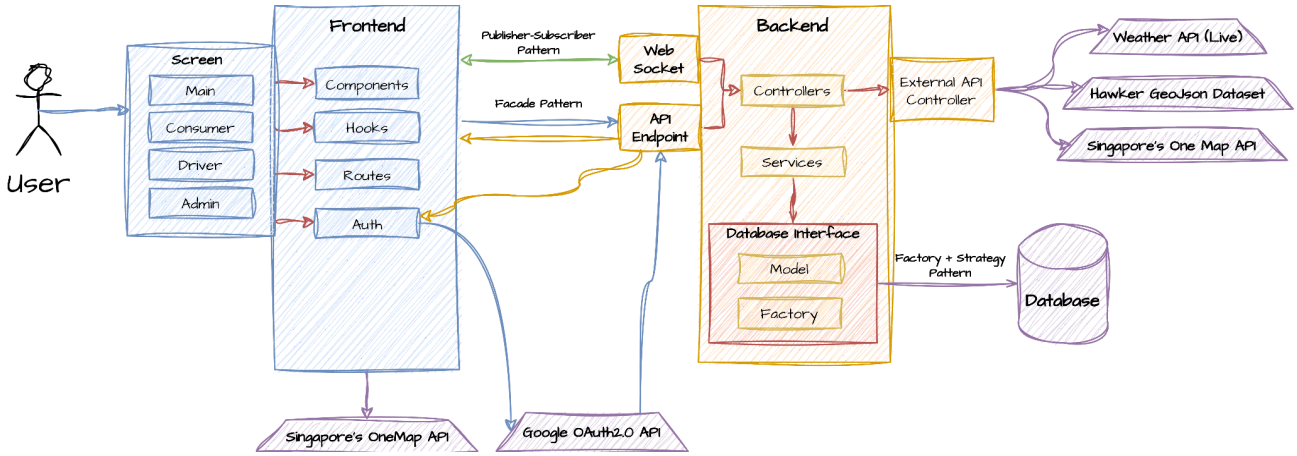
UI Inspiration: *https://dribbble.com/shots/20463041-Food-Delivery-App-Foodoo*

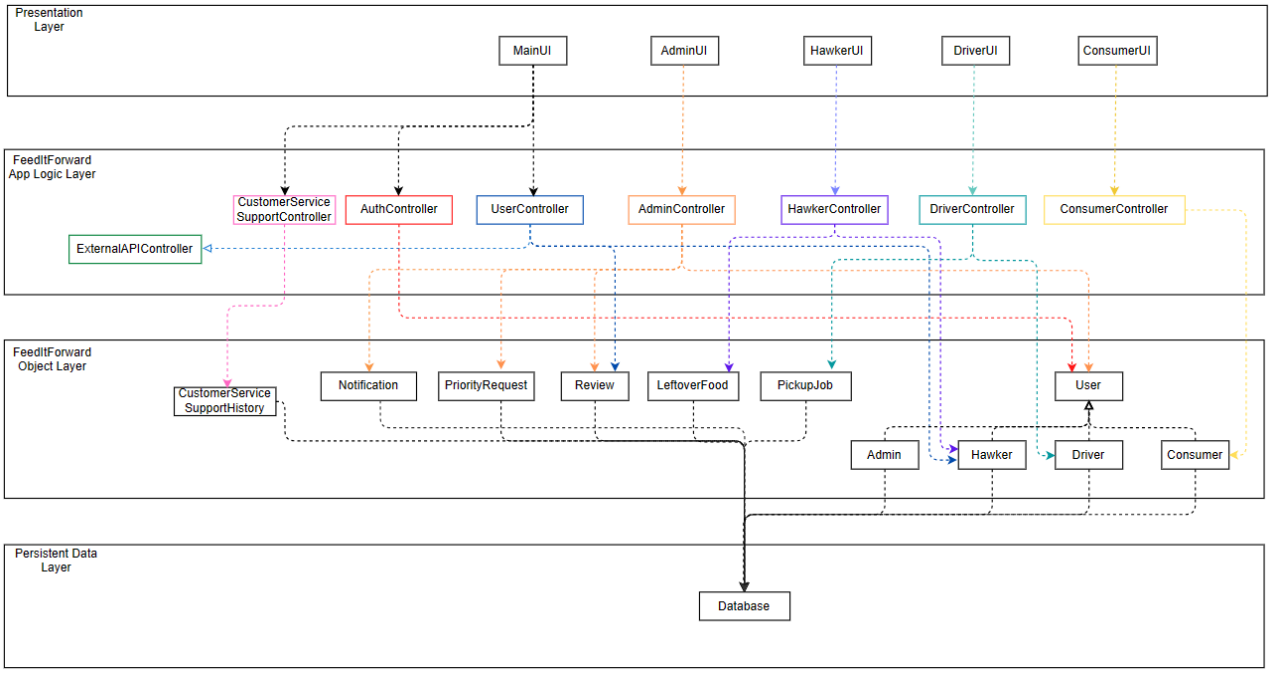
OOP Principles: *https://khalilstemmler.com/articles/object-oriented/programming/4-principles/*

2. Overall Description

2.1 Product Perspective

This SRS specifies the requirements for the *FeedItForward*, a new, self-contained product designed to create an ecosystem for food distribution. It is not a follow-on member of a product family nor a replacement for existing systems but is a pioneering effort to reduce food waste and hunger through technology. The platform functions independently and also integrates with existing technologies via APIs. A system architecture diagram (as shown below) is provided to illustrate the interconnections and external interfaces.





2.2 Product Functions

2.2.1 Use Case Diagram

The following use case diagram (Figure 2.2.1) depicts the key product functionalities that FeedItForward includes. There are 4 primary users of FeedItForward – Admin, Hawker, Consumer, and Driver. Each type of user interacts with FeedItForward to use their role-specific functions.

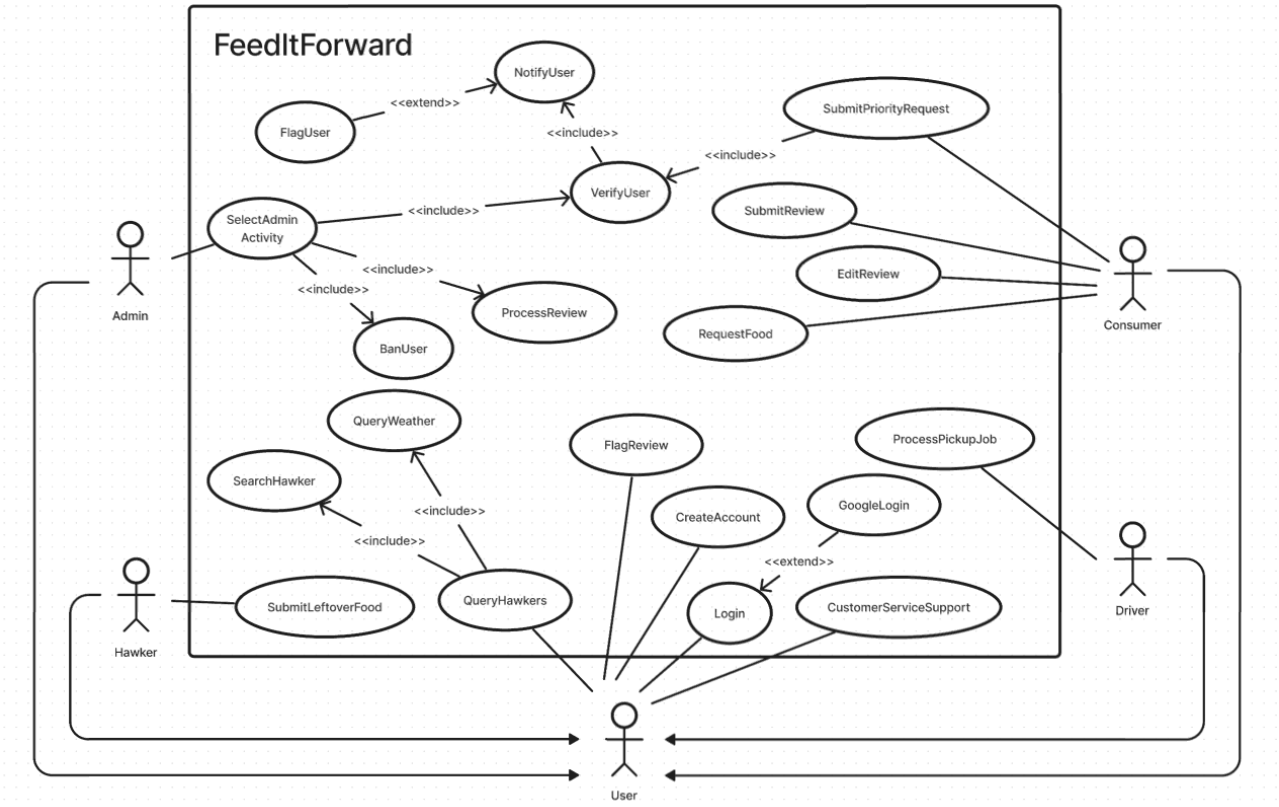


Figure 2.2.1: Use Case Diagram

2.2.2 Major Product Functions

FeedItFoward will allow users to perform the following functions:

1. **Authentication**
    - Signup (with different roles – Admin, Hawker, Consumer, Driver)
    - Login
    - Login with Google
2. **Hawker Functions**
    - View Hawker Listing Profile
    - Submit Leftover Food
    - **Map**
        - Displays all the registered and external api public hawkers
3. **Consumer Functions**
    - **Hawkers**
        - View and Search for Hawkers
        - View Hawker Listing
    - **Leftover Food**
        - View and Search for Leftover Food
        - Request for Leftover Food
    - **Reviews**
        - View, Add, and Edit Hawker Reviews
        - Flag Reviews
    - **Map**
        - View Hawkers
        - Search Hawkers
    - **Food Priority Request** (to gain priority when requesting for food)
4. **Driver Functions**
    - Accept/Ignore Pickup Job
    - Submit photo proofs to complete Pickup Job
    - **Map**
        - Weather Forecast
5. **Admin Functions**
    - Verify User (Verify Consumers' Food Priority Request)
    - Ban User
    - Process Flagged Reviews
    - Notify User (Admin can send custom notifications)
6. **Notifications**
    - System-wide alerts and user-specific notifications
7. **Customer Service Support**

- Real-time Messaging between Users and Admins

8. Other Functions

- Settings
- Edit Profile

2.2.3 Class Diagram

The following Class Diagram (Figure 2.2.3) shows how different classes interact with each other, as well as the key functionalities that FeedItForward needs to perform (refer to the controllers in the middle row section).

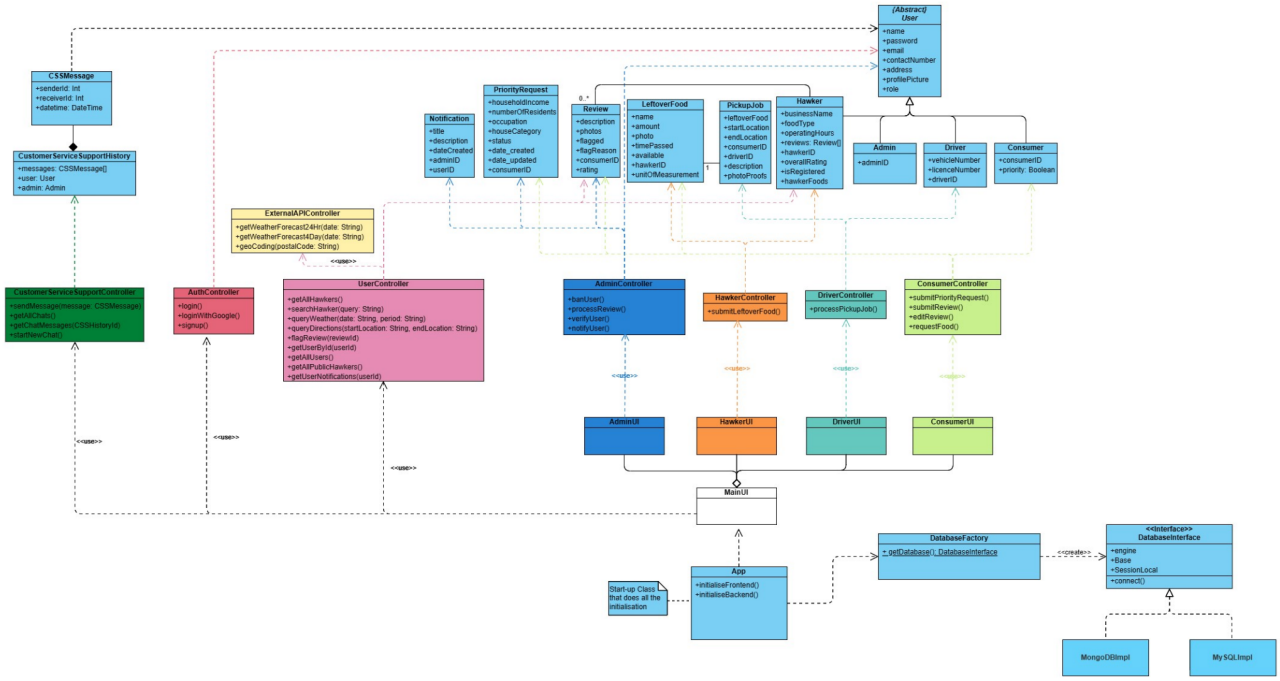


Figure 2.2.3: Class Diagram

2.3 User Classes and Characteristics

Users of the platform are categorized into:

- **Hawkers:** Food vendors with surplus to donate. They require a simple, straightforward interface.
- **Consumers:** Individuals or families in need of food. They may vary in technical expertise.
- **Drivers:** Volunteers or paid drivers who deliver food. They will use the platform frequently and need a reliable mapping and scheduling system.
- **Admins:** Staff who oversee platform operations. They are technically skilled and need comprehensive tools for user management.

All four types of users are equally important for FeedItForward as they form a pipeline that makes FeedItForward functional and operational. E.g.: Hawkers submit leftover surplus food, Consumers request for the leftover food, Drivers accept pickup jobs and deliver the leftover food from Hawkers to Consumers, while Admins ensure that FeedItForward is fair and safe for its users.

2.4 Operating Environment

The platform will be web-based, optimized for both desktop and mobile use, operating on the latest versions of Chrome, Firefox, Safari, and Edge. It will be hosted on cloud services for scalability and reliability and must integrate with external APIs (OneMap, Google OAuth, Weather, Hawker) and database storage (SQL).

2.5 Design and Implementation Constraints

The development of FeedItForward will be subject to several constraints that will guide the design and implementation of the software:

2.5.1 Frontend Constraints

2.5.1.1 Framework (Frontend)

The user interface will be developed using React.js, TypeScript, and TailwindCSS.
(TypeScript was chosen for its strong typing system which enhances code quality and maintainability.)

2.5.1.2 Code Formatting

Code formatting and consistency will be enforced using Prettier and ESLint so as to adhere to the best coding practices.

2.5.2 Backend Constraints

2.5.2.1 Framework (Backend)

The frontend will be developed using React.The backend server will be developed using FastAPI in Python.

2.5.2.2 Database

The database will be in SQL as our data is structured and relational.

SQL will be the language for database interactions, with an emphasis on robustness and the ability to handle complex queries efficiently.

2.5.2.3 Code Formatting

Code formatting and consistency will be enforced using the Black formatter.

The Black formatter will be used to maintain uniform code formatting on the backend, streamlining the development process and facilitating code reviews.

2.5.3 Language Constraints

- All software documentation, interfaces, and user interactions will be provided in English.
- The choice of English is due to its prevalence as a primary or secondary language among the anticipated user demographics and to maintain consistency across the platform.

2.6 User Documentation

FeedItForward will include a comprehensive set of user documentation to ensure proper understanding and ease of use for developers. The documentation will consist of the following components:

2.6.1 README Files

2.6.1.1 Main Repository

A README file for the main repository will provide an overview of the project, including the architecture, how to set up the development environment, deployment instructions, and a high-level guide to the repository's contents.

2.6.1.2 Frontend Repository

The frontend repository will have its own README file detailing the setup process, the structure of the frontend application, guidelines for using React.js, TypeScript, and TailwindCSS within the project, and instructions for running and building the application.

2.6.1.3 Backend Repository

Similarly, the backend repository will contain a README file with instructions specific to setting up the FastAPI server, configuring Python environments, handling dependencies, and information about the SQL database integration.

2.6.2 Code Comments

- The source code for both frontend and backend will be well-commented, providing clarity on complex logic, functions, components, classes, and methods. Comments will also include tagging for TODOs and FIXMEs where necessary.
- Inline documentation will be present to explain the purpose and usage of various code blocks, modules, and libraries.

2.6.3 Backend API Documentation

- For the backend API documentation, FastAPI's automatic interactive API documentation will be utilised. This API documentation is generated from the backend source code and is updated automatically. This will provide comprehensive documentation and testing capabilities for the API endpoints.
- The documentation will include detailed information about the API routes, expected input parameters, authentication requirements, and response models. This will be an invaluable resource for frontend developers who need to integrate with the backend services and for any third-party services that may need to interact with the platform.
- **Note: A total of 100 API routes will be documented.*

2.7 Assumptions and Dependencies

2.7.1 Assumptions

It is assumed that the platform will have access to the internet at all times.

2.7.1 Dependencies

2.7.1.1 Front-end Libraries

The user interface of FeedItForward depends on numerous front-end libraries:

Library	Purpose
React.js	To create dynamic single page application
Material UI	To provide a base user interface that can be further customised
react-oauth/google	To provide authentication via Google OAuth
react-leaflet	To provide an interactive map interface
react-modal	To provide an accessible modal dialog component
react-dropzone	To enable file uploads using drag and drop.
react-hot-toast	To provide a notification component
react-icons	To provide icons
react-websocket	To provide a component for web socket communication
typescript	To provide strong typing for easier maintainability

2.7.1.2 Back-end Libraries

The backend of FeedItForward depends on numerous back-end libraries:

Library	Purpose
fastapi	To provide a framework for developing the backend server
uvicorn	To serve the backend server
sqlalchemy	To provide a python interface to SQL
httpx	To provide a more reliable http

bcrypt	To provide hashing functions
python-multipart	To enable the server to accept file uploads

2.7.1.3 External Services

FeedItForward depends on external services:

- Third-party APIs
    - OneMap API
    - Google OAuth
    - Weather API
    - Hawker GeoJSON API Dataset
- Cloud hosting services
    - GitHub Pages

Hence, FeedItForward relies on the availability and reliability of external services.

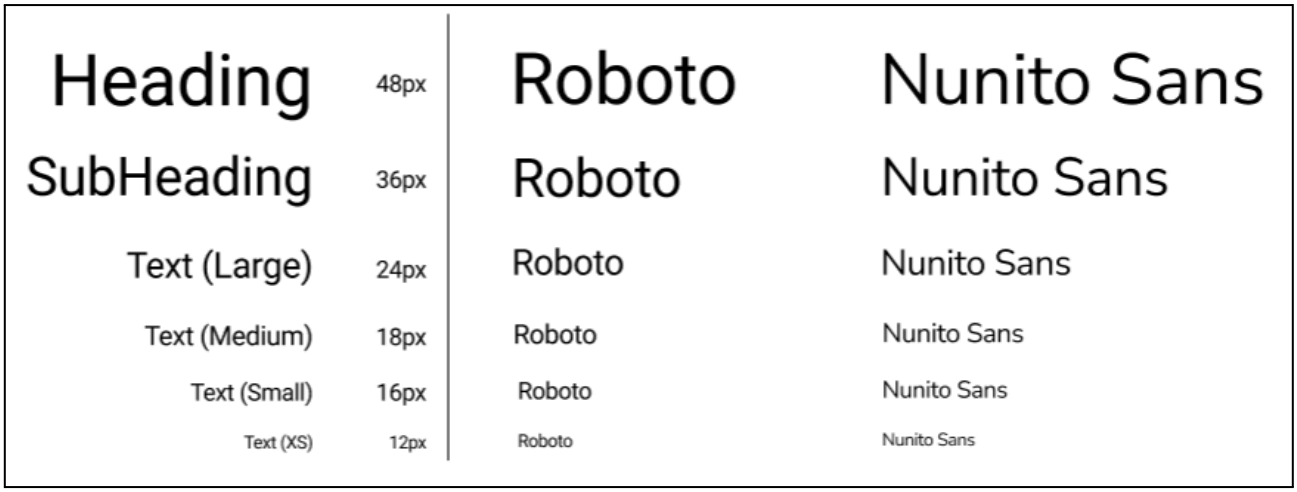
3. External Interface Requirements

3.1 User Interfaces

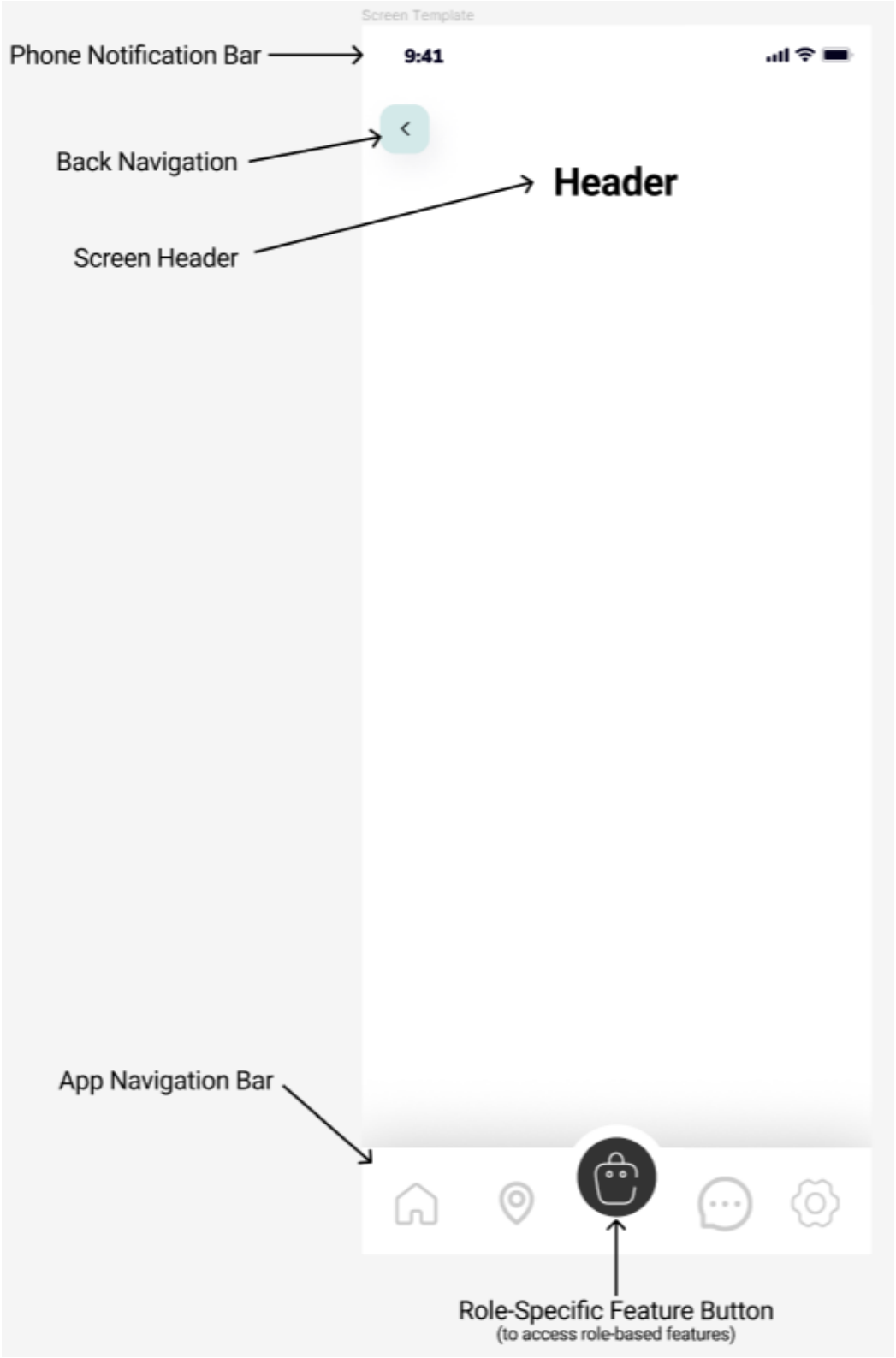
3.1.1 Style Guides

3.1.1.1 Theme Colors

3.1.1.2 Typography

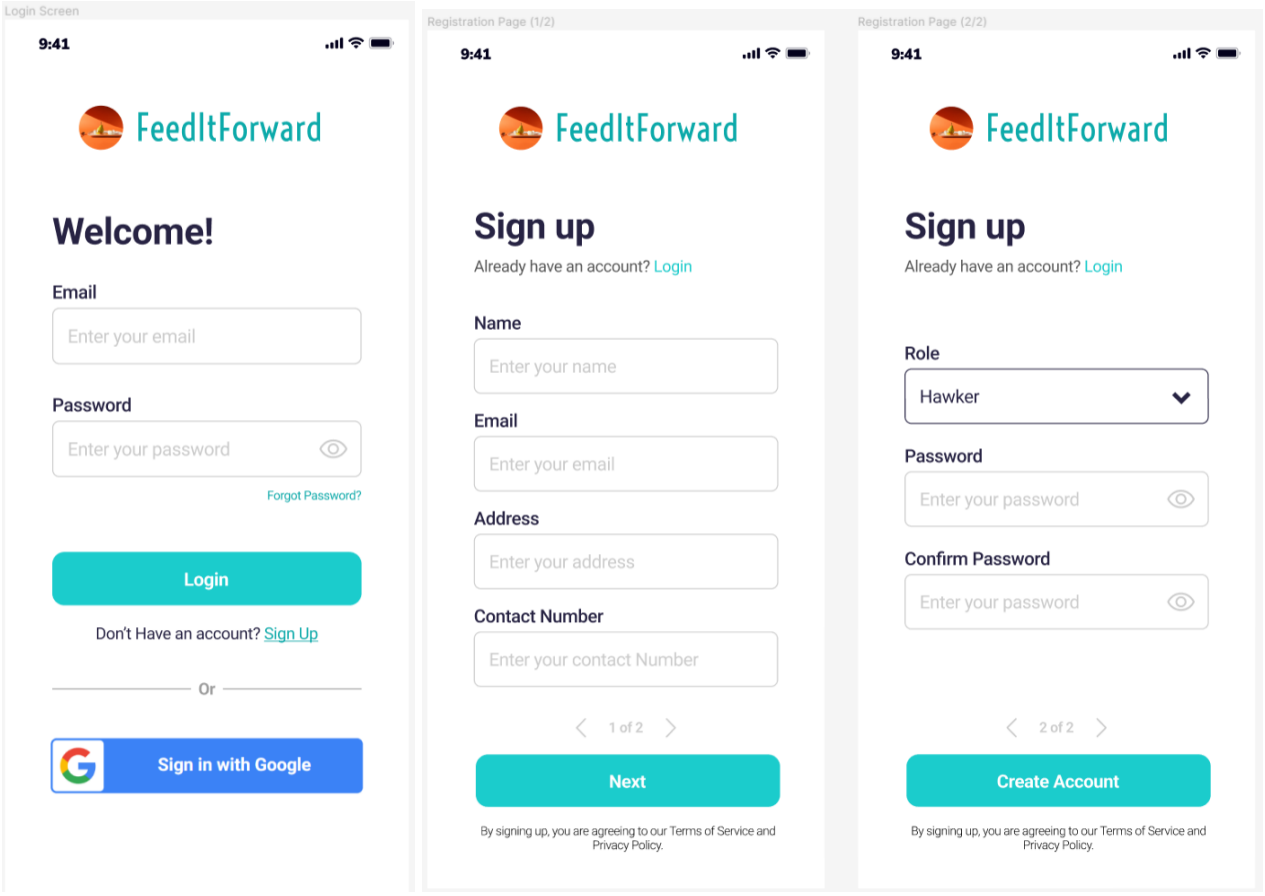


3.1.1.3 App Screen Template

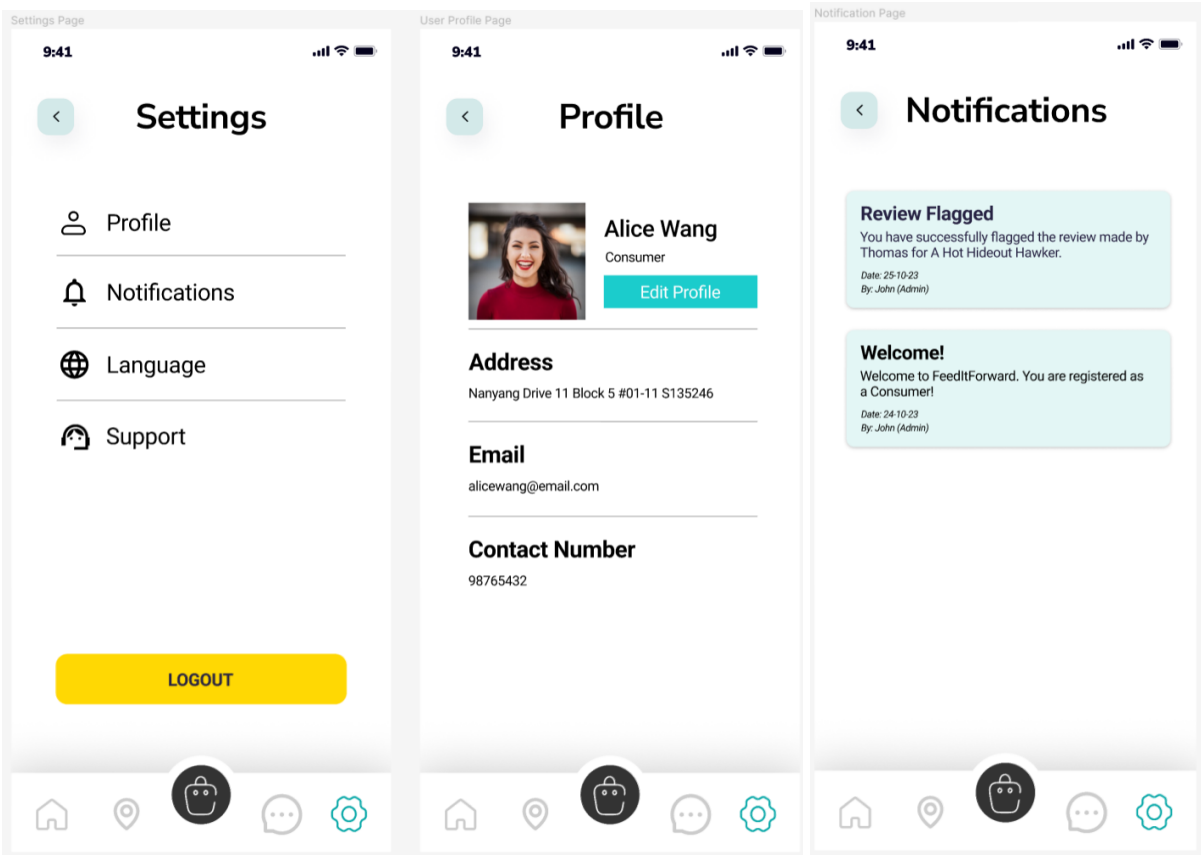


3.1.2 UI Mockups (Figma)

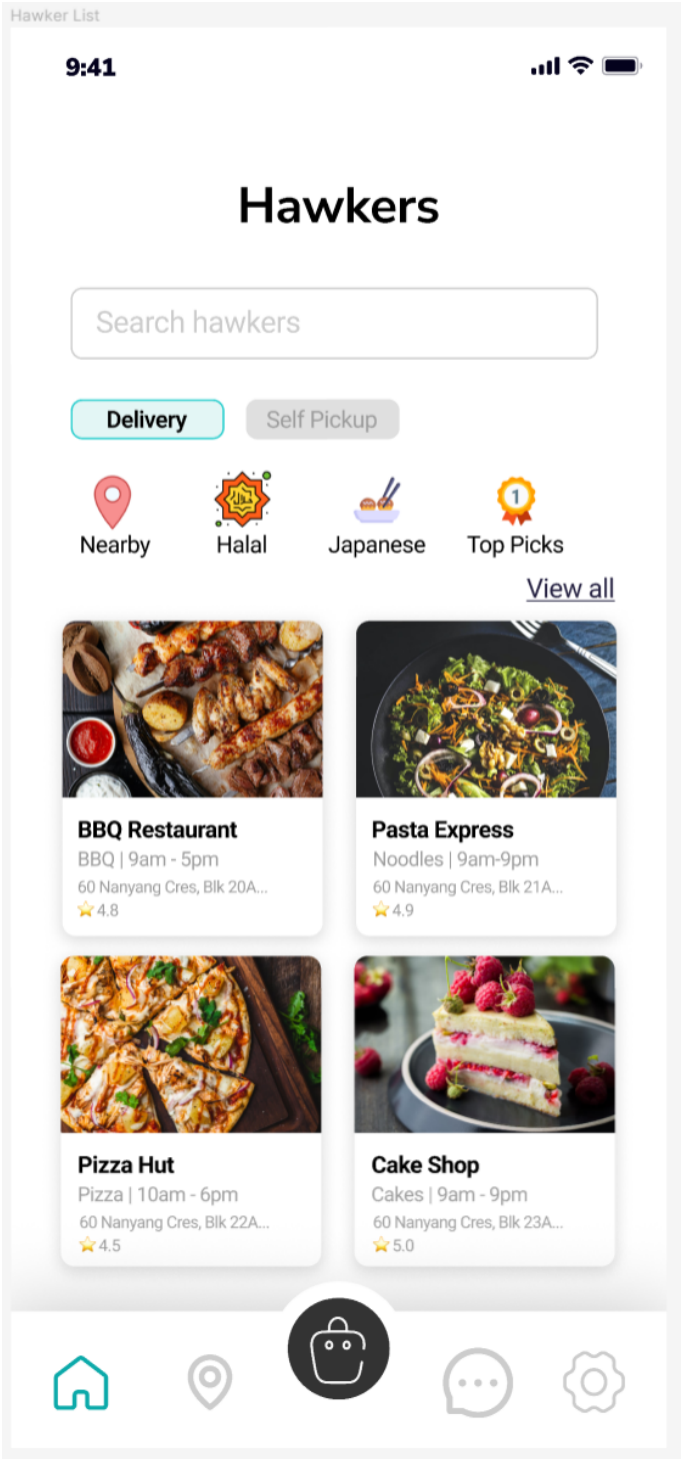
4.1 Authentication (Login + Google Login + Signup)



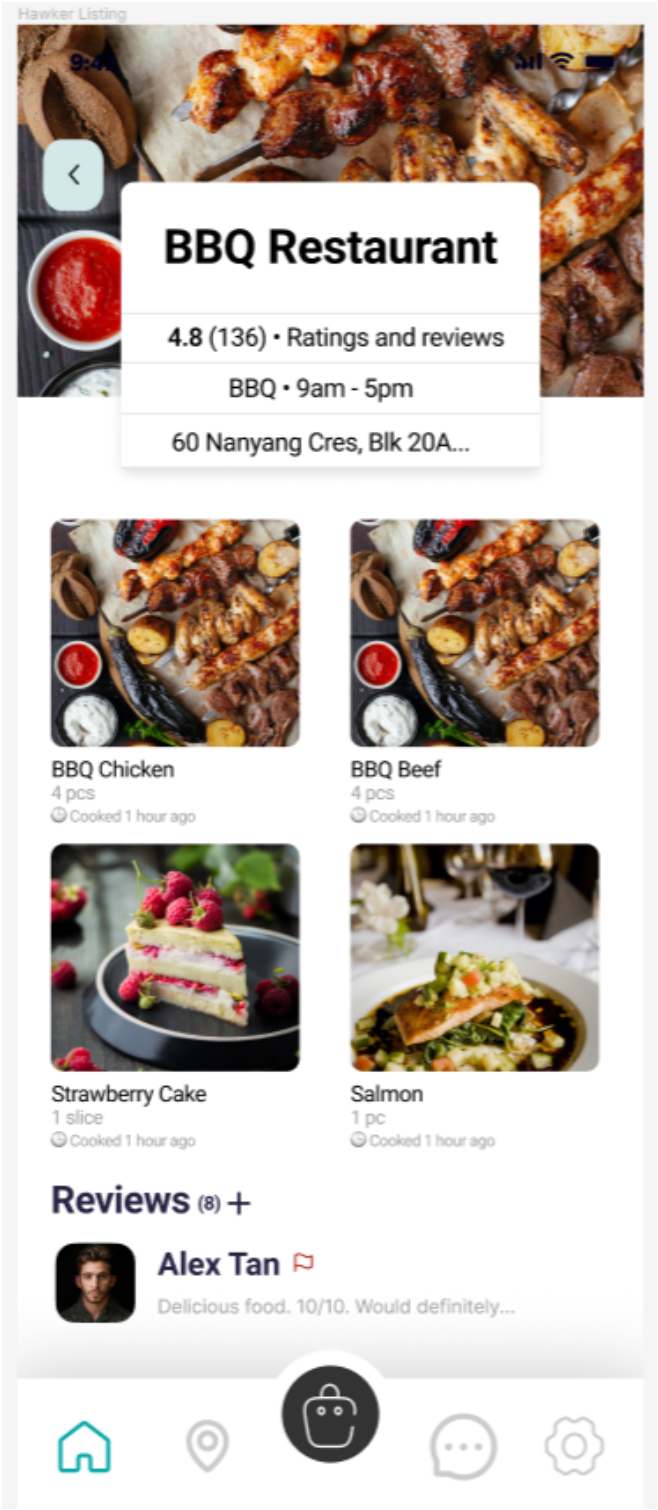
4.2 Settings + User Profile + Notifications



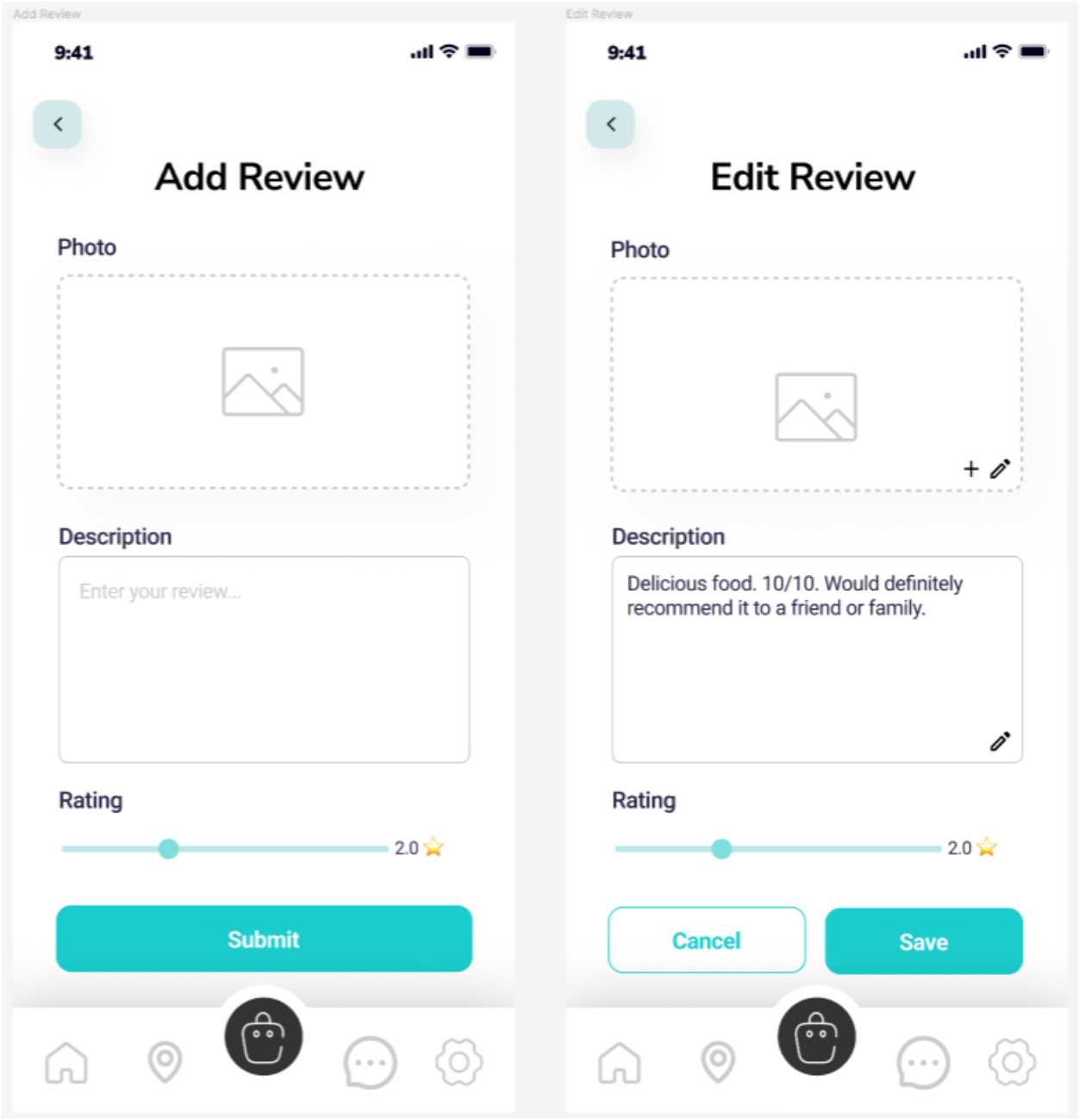
4.3 Home Screen



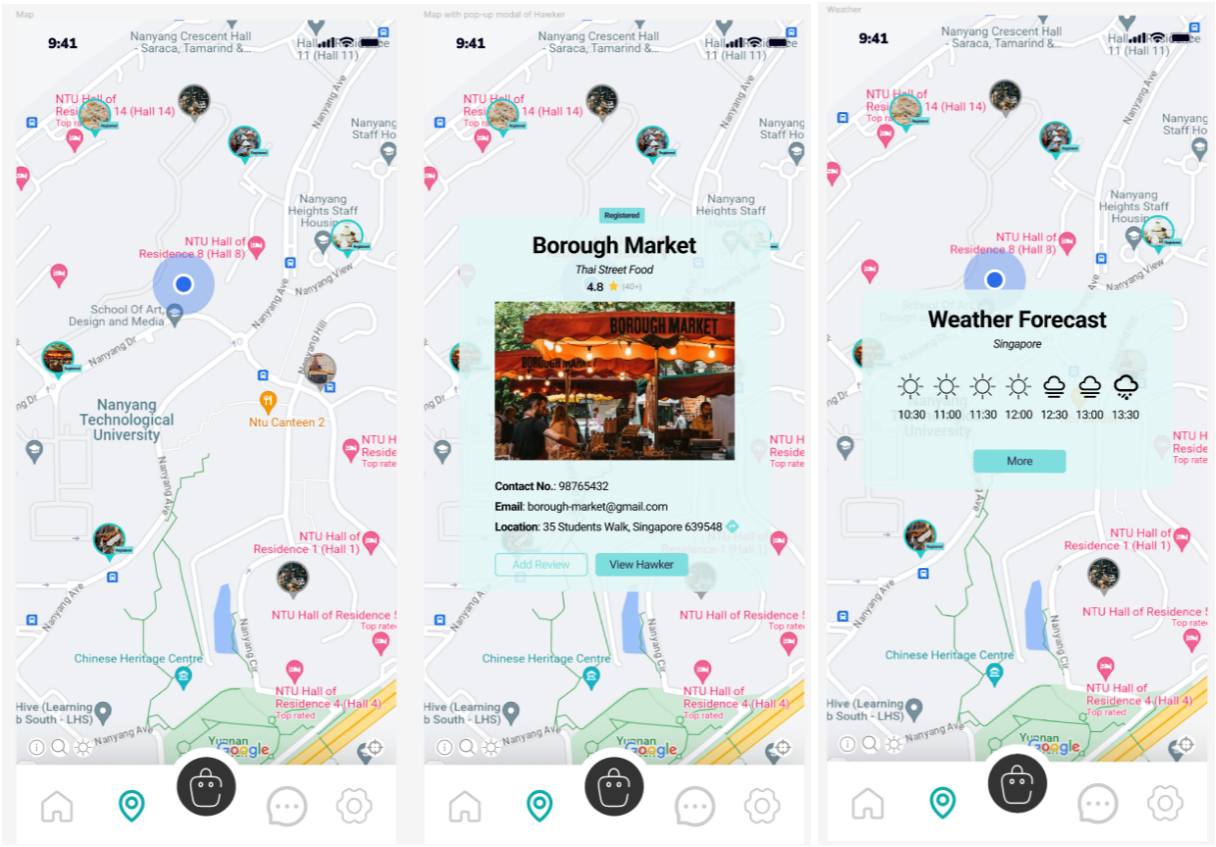
4.4 Hawker Listing



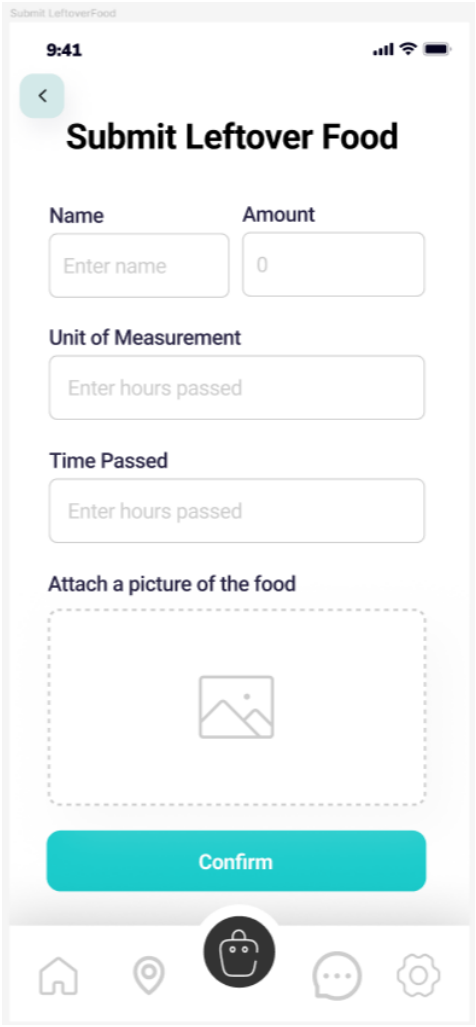
4.5 Add Review + Edit Review



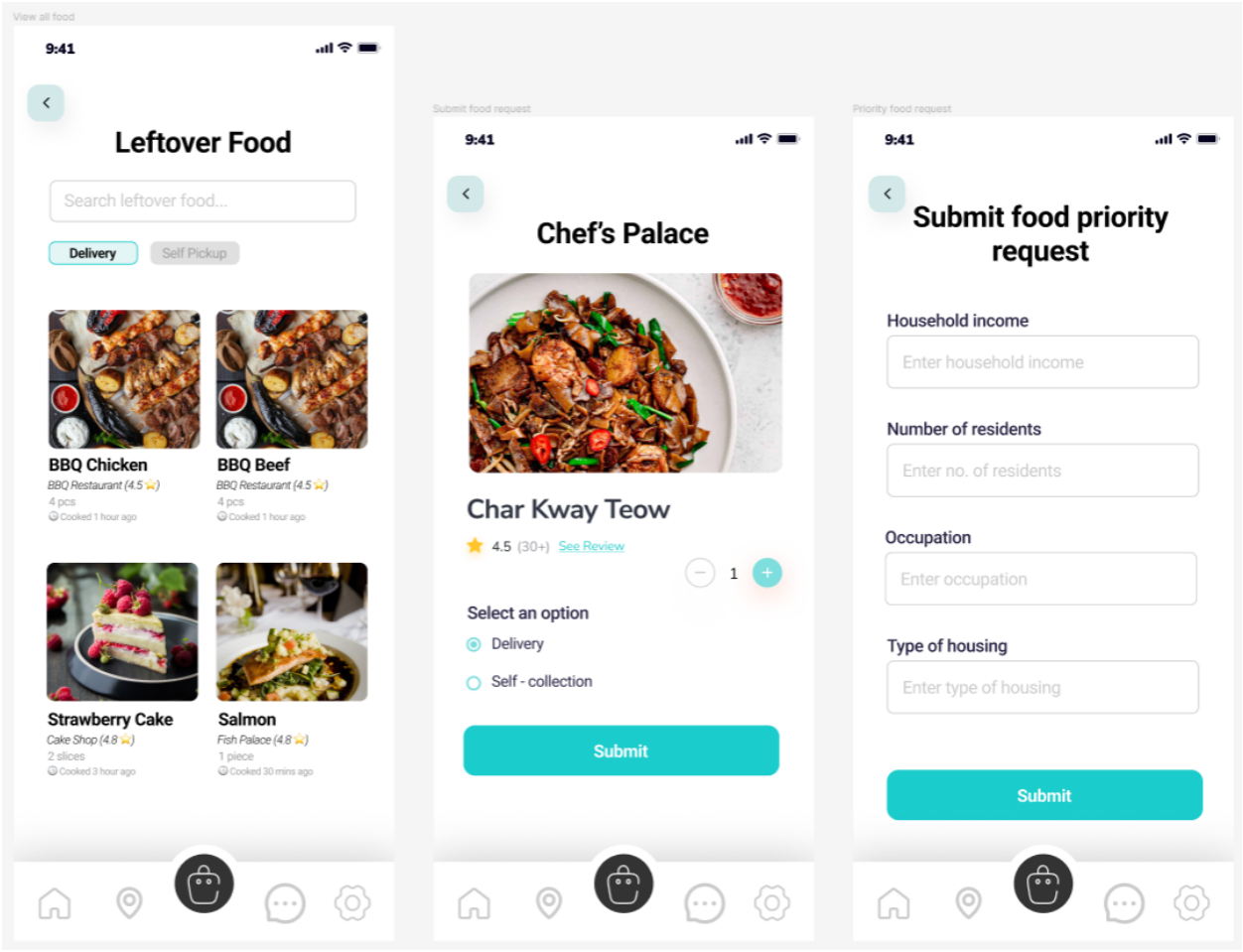
4.6 Map – Interactive Map + Directions + Weather + Hawker Pop-up Modal



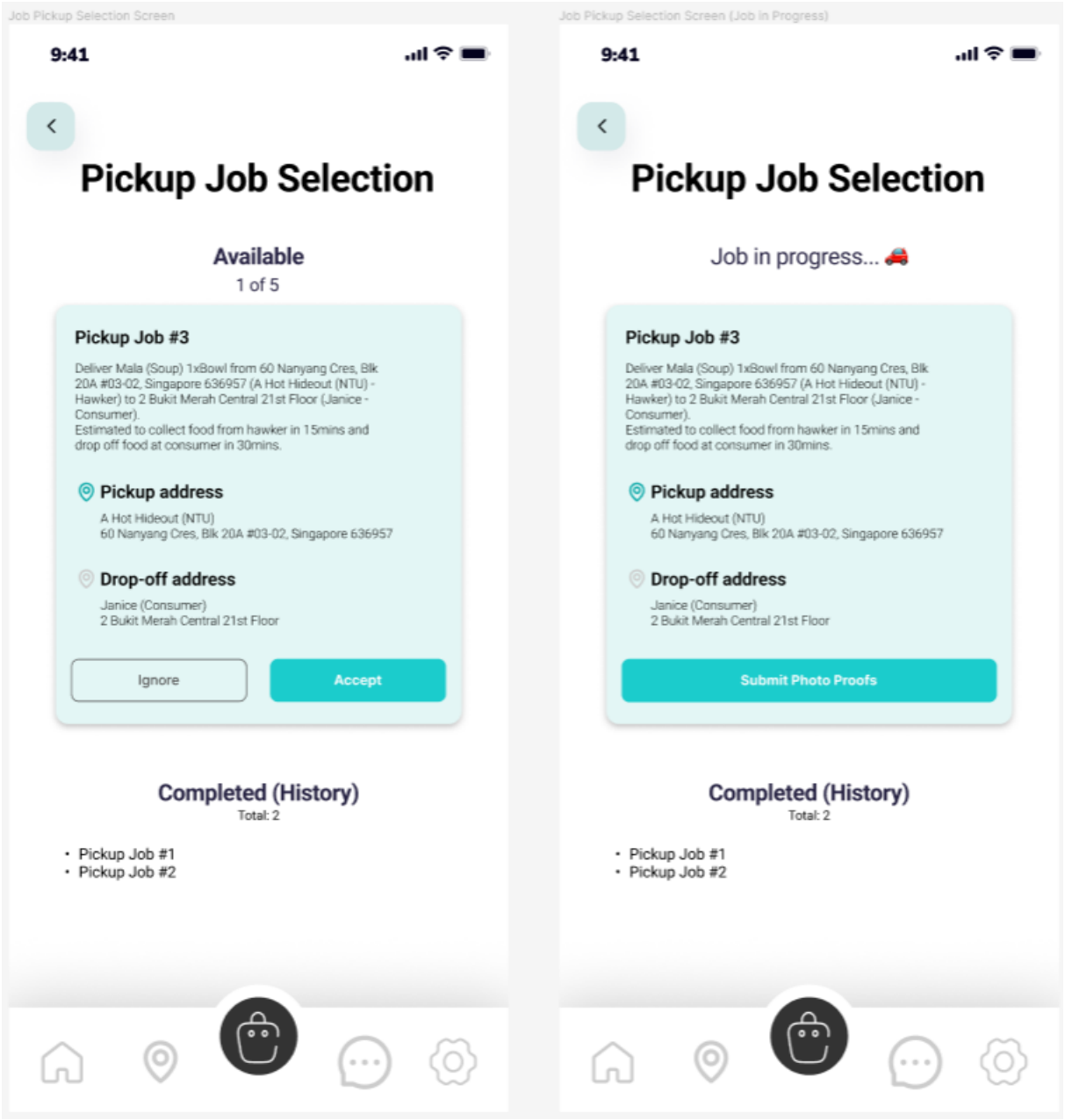
4.7 Hawker-related – Submit Leftover Food

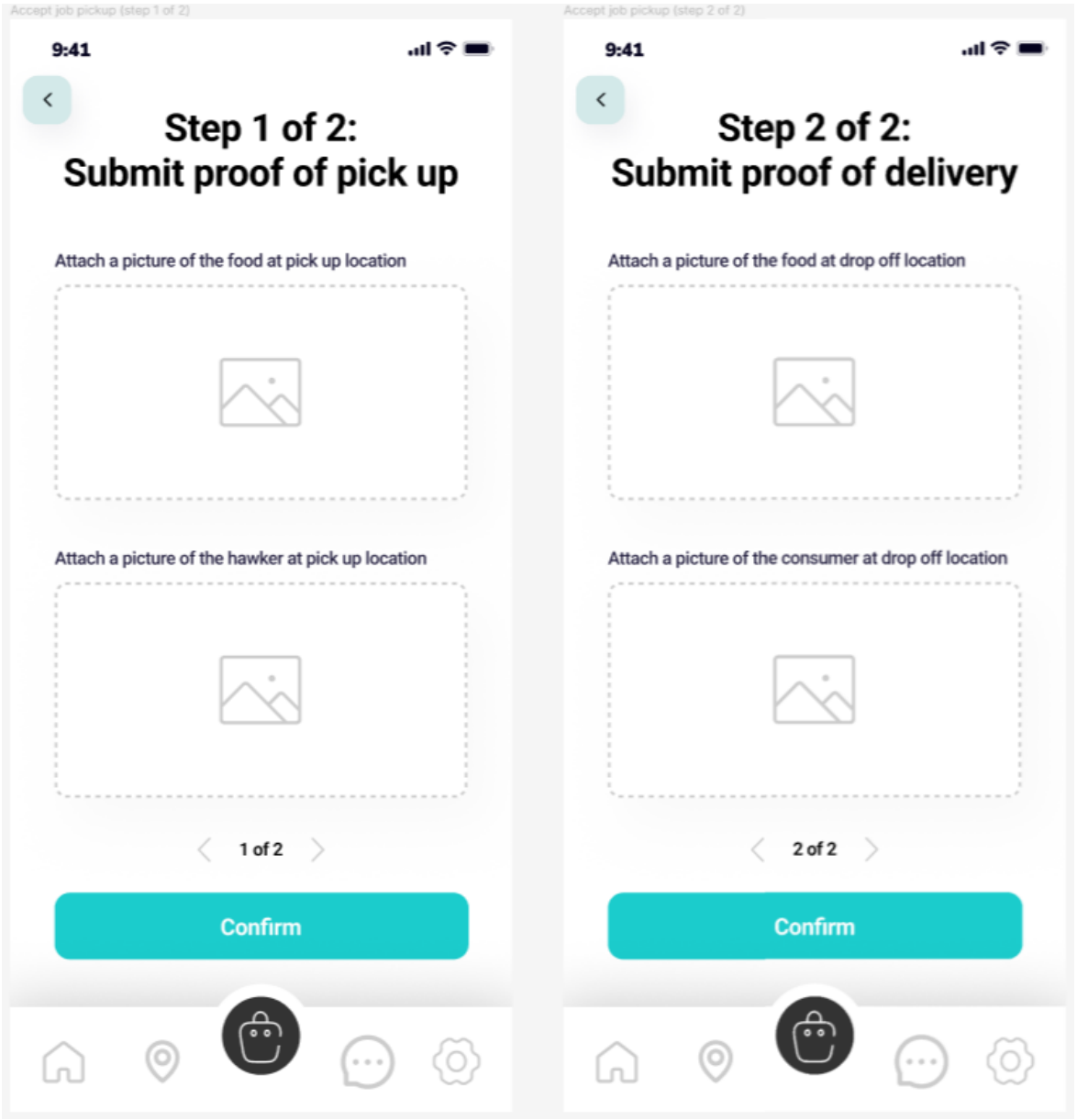


4.8 Consumer-related – View all leftover food + Submit food request + Priority Food Request

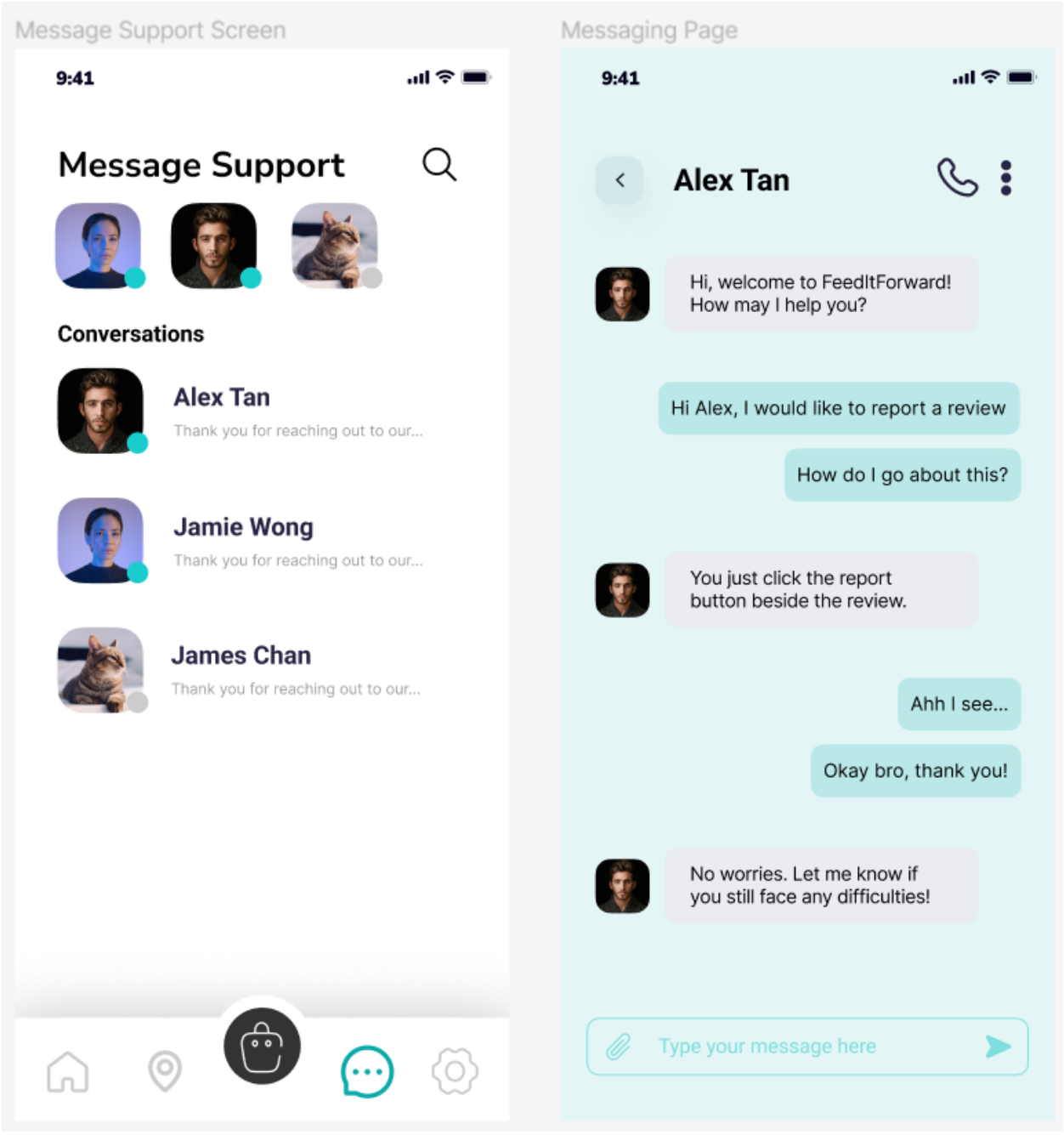


4.9 Driver-related – Pickup Job Selection + Process Pickup Job (Submit photo proofs)

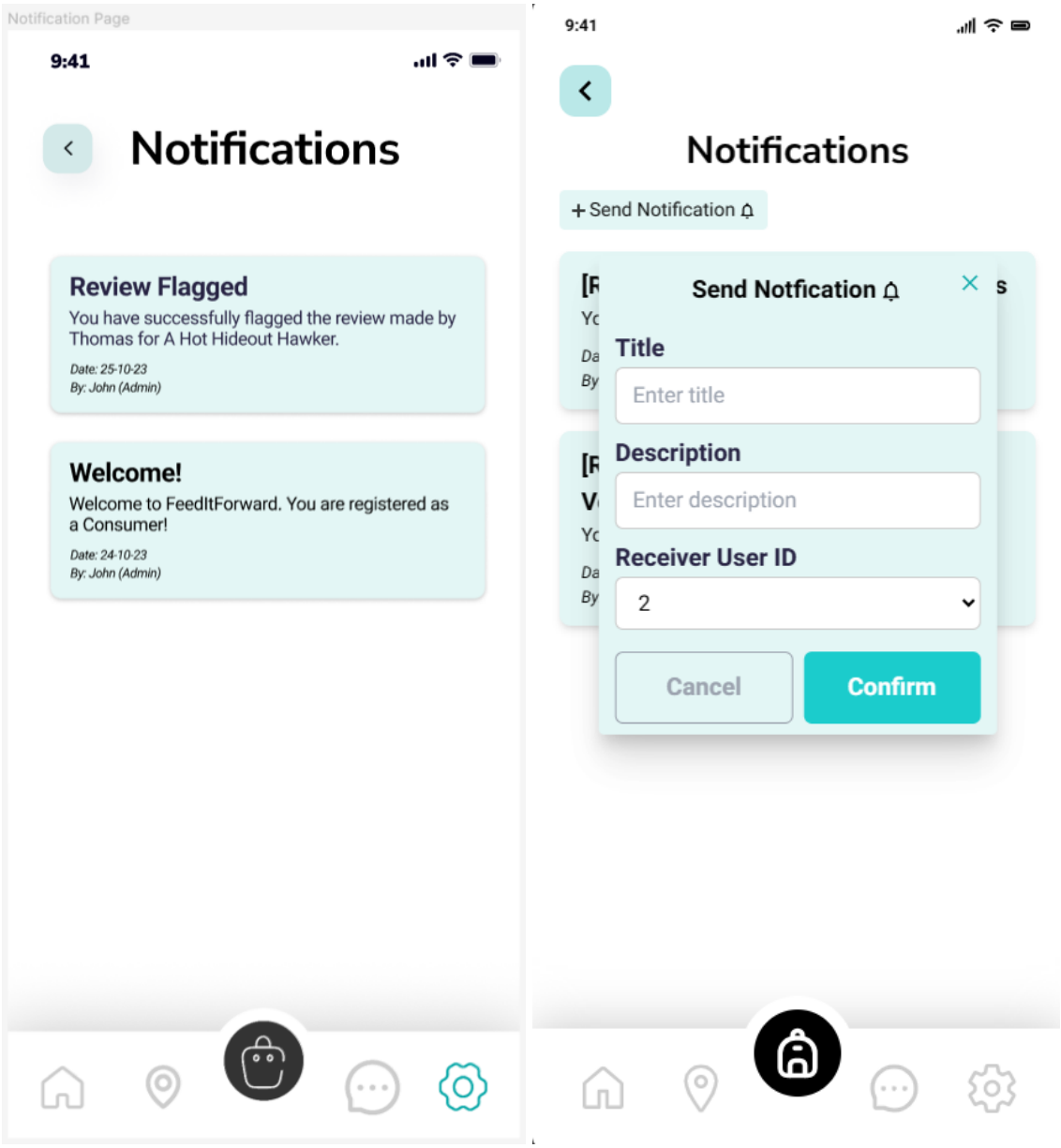



Accept job pickup (step 1 of 2)
9:41
Step 1 of 2:
Submit proof of pick up
Attach a picture of the food at pick up location
Attach a picture of the hawker at pick up location
1 of 2
Confirm
Accept job pickup (step 2 of 2)
9:41
Step 2 of 2:
Submit proof of delivery
Attach a picture of the food at drop off location
Attach a picture of the consumer at drop off location
2 of 2
Confirm

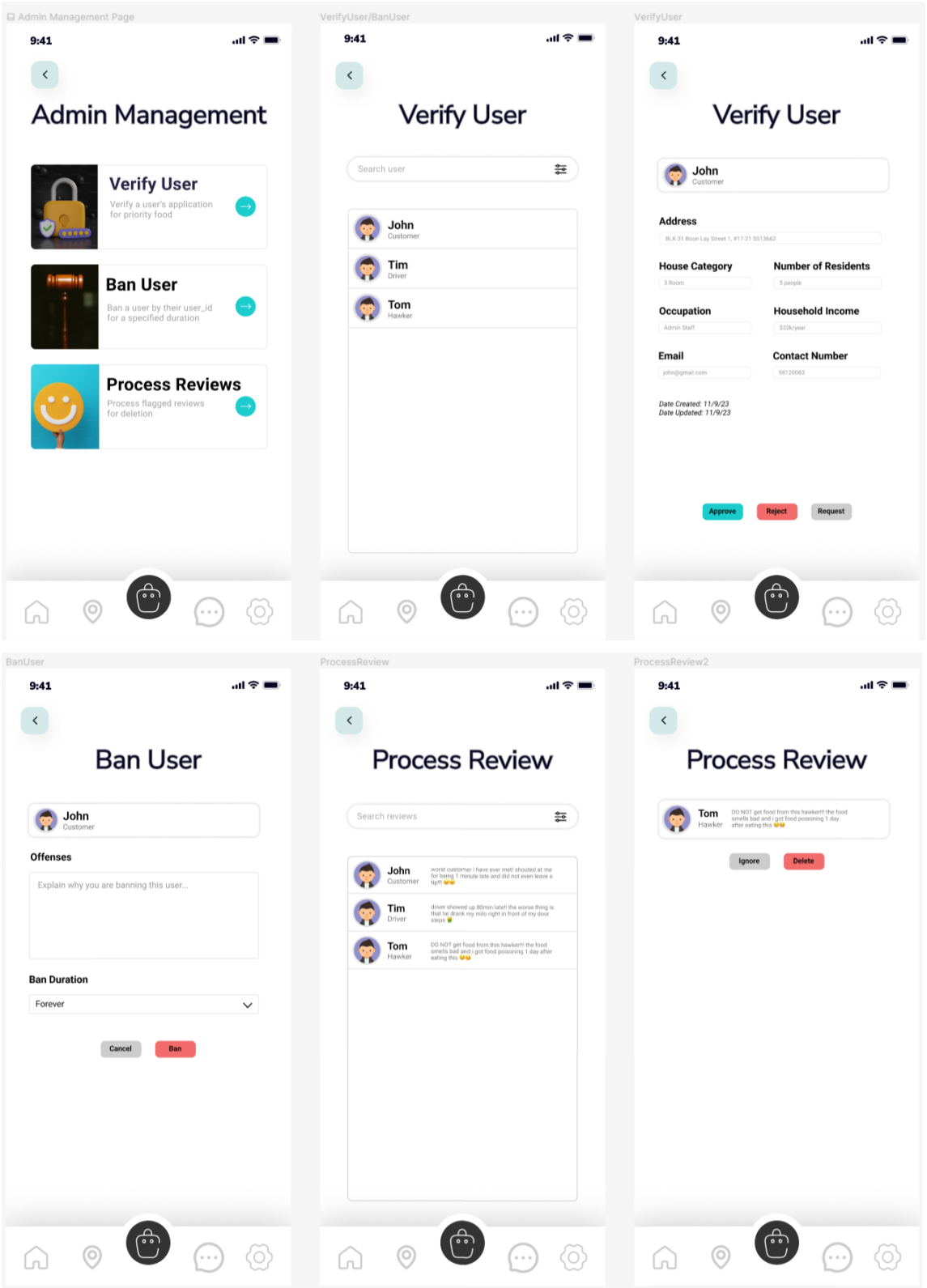
4.10 Customer Service Support – Messaging



4.11 App Notifications + Send Notification

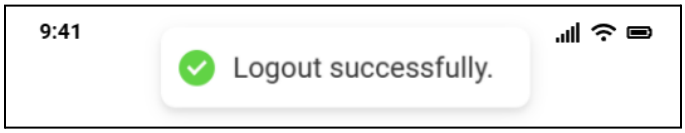


4.12 Admin-related – Admin Management + Ban User + Verify User + Process Reviews Screens

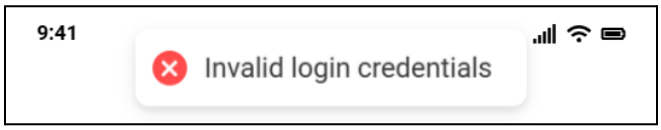


3.1.3 Success/Error Message Display (Toast Notification)

The toast notification of a success/error message will be used whenever a success/error event is triggered. For example, on a successful logout (Fig. 3.1.3.a), or a failed login attempt (Fig. 3.1.3.b).



3.1.3.a: Success Message



3.1.3.b: Error Message

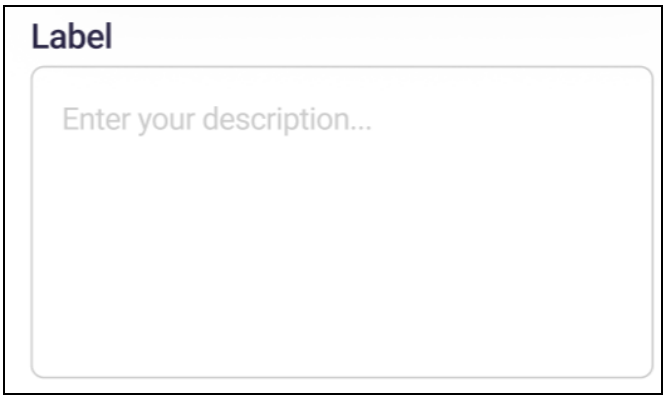
3.1.4 Common Components

3.1.4.1 Form Inputs

Form Text Input

Form Submit Button

Form Image Upload Input

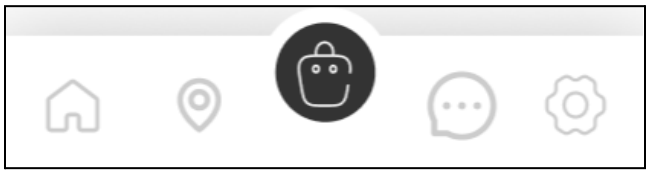


Form Textarea Input

Form Number Range Picker Input

3.1.4.2 Misc

Screen Header



App Navigation Bar

3.2 Hardware Interfaces

FeedItForward is designed to operate primarily as a web-based application, minimizing the need for direct hardware interfaces. However, the following outlines the logical and physical characteristics of the interfaces between the software and hardware:

3.2.1 Device Types

The software is designed to be compatible with various devices, including desktop computers, laptops, tablets, and smartphones.

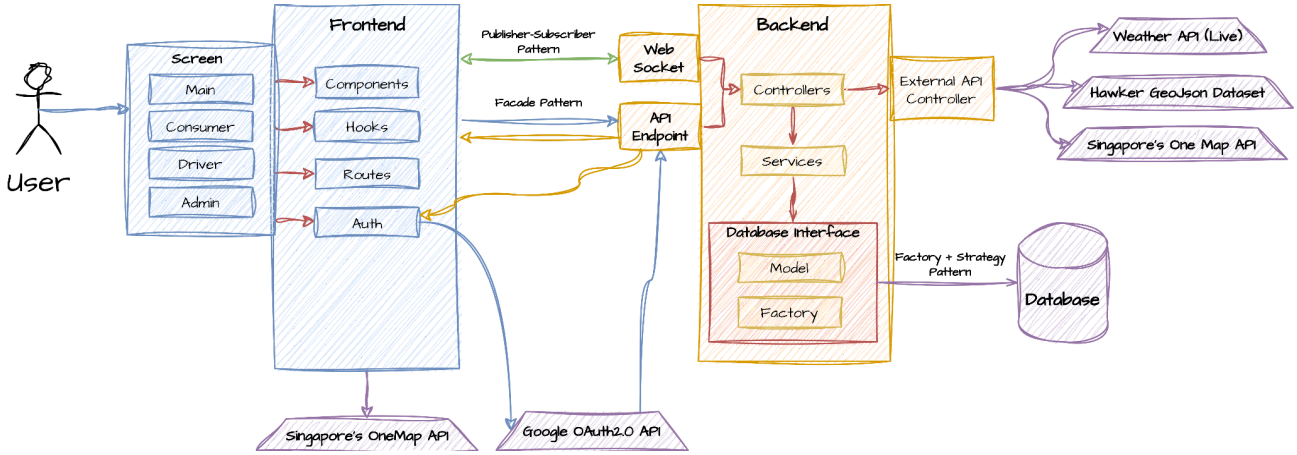
3.2.2 Data and Control Interactions

The user's device interacts with the software through standard input/output hardware such as keyboards, touchscreens, and mice for input; displays for output; and network interface cards for internet connectivity.

3.2.3 Communication Protocols

The platform uses HTTPS for secure web traffic. The APIs, websocket connection, and services will communicate over standard web ports (80 for HTTP and 443 for HTTPS).

3.3 Software Interfaces



FeedItForward interfaces with several software components:

3.3.1 Operating System

Any OS capable of running a modern web browser (Windows, macOS, Linux, iOS, Android).

3.3.2 Web Browsers

Latest versions of Chrome, Firefox, Safari, Edge, and others capable of supporting HTML5, CSS3, and JavaScript.

3.3.3 Databases

SQL-based database systems, accessed through backend services.

3.3.4 Backend Tools

1) FastAPI (version 0.104.1) for server-side logic
2) Python (version 3.10) for scripting
3) Black for code formatting

3.3.5 Frontend Tools

1) React.js (version 18.2.0)

2) TypeScript (version 4.9.5)
3) TailwindCSS (version 3.3.3)
4) Prettier and ESLint for code formatting

3.3.6 RESTful APIs

RESTful API interfaces for communication between frontend and backend, with detailed documentation (required request body and expected response body) provided via FastAPI's interactive tools.

3.3.7 Data Sharing

JSON format is used for all data exchange between frontend and backend. Data persistence is maintained through the SQL database with specific defined data models.

3.4 Communications Interfaces

For FeedItForward, various communication interfaces are integral to ensuring a seamless and responsive user experience:

- **Web Socket:** Implementation of Web Socket protocol to facilitate real-time bi-directional messaging between clients and the server. This will be especially crucial for customer service support features, allowing for instant communication between users and support personnel.
- **HTTP/HTTPS:** All interactions with the web application will occur over HTTP/HTTPS protocols, ensuring compatibility with modern web browsers and secure data transmission.
- **API Communication:** RESTful APIs will be used for most server communications, with JSON as the primary data interchange format. For the documentation and testing of these APIs, FastAPI's Swagger UI will be employed.
- **Electronic Forms:** Forms submitted through the platform will use secure HTTPS transactions to protect user-submitted data.
- **Network Protocols:** Standard network protocols such as TCP/IP will be used for network communication, ensuring adherence to established internet communication standards.
- **Security and Encryption:** TLS (Transport Layer Security) will be enforced for all communications to safeguard data integrity and privacy. OAuth may be used for integrating third-party services.
- **Message Formatting:** Standardized message formats will be used for API responses and WebSocket communications to ensure consistency and ease of parsing.

These communication interfaces are designed to provide robust, efficient, and secure interactions within the platform, aligning with the goal of delivering a reliable service for connecting food surplus with the needy while supporting real-time customer service capabilities.

4. System Features

Please refer to the 'Use Case Diagram and Descriptions, and Sequence Diagrams.pdf' for the full use case diagram, detailed use case descriptions, and sequence diagrams for each use case.

4.1 Admin Features

4.1.1 SelectAdminActivity

4.1.1.1 Description and Priority

Description:

- This feature enables an Admin to select and perform various administrative activities within the system.

Priority:

- High; plays a central role in maintaining system integrity and user management.

4.1.1.2 Stimulus/Response Sequences

1. After the admin successfully logs in and is authenticated, admin can go to the SelectAdminActivity page.
    a) The system displays the activity options available to admin: Ban user, Verify User, Process Review.
2. Admin can now select an option.
    a) Admin selects the ban user option.

        i) System carries out the BanUser feature, allowing the Admin to ban a user.

    b) Admin selects the verify user option.

        i) System carries out the VerifyUser feature, allowing the Admin to approve or reject user applications for priority.

    c) Admin selects the process review option.

        i) System carries out the ProcessReview feature, allowing the Admin to decide on the removal of a review.

4.1.1.3 Functional Requirements

REQ-1: App must authenticate the Admin before allowing access to administrative functions.

REQ-2: App must display the options for Admin to select from the following available activities: Ban user, Verify User, Process Review.

REQ-2.1: App must integrate the BanUser, VerifyUser and ProcessReview features.

REQ-2.2: App must navigate admin to the activity page based on their selected activity.

4.1.2 BanUser

4.1.2.1 Description and Priority

Description:

- Allows Admin to ban a User by their user ID for a specified duration.

Priority:

- High; important for enforcing user conduct and maintaining the integrity of the system.

4.1.2.2 Stimulus/Response Sequences

1. After admin selects the Ban User option from SelectAdminActivity page, admin is directed to the BanUser page.
2. App displays ban user related information.
    a) Admin inputs the user ID of the user to be banned.
        i) The system validates the user ID
        ii) The system cannot validate the user ID, and shows an error message to prompt admin to enter user ID again.
    b) Admin enters the duration of the ban.
        i) The system processes the duration for the ban.
    c) Admin confirms the ban by selecting confirm.
        i) The system implements the ban and provides confirmation to the Admin via a toast notification.
    d) Admin cancels the ban by selecting cancel.
        i) The system cancels the ban process and returns to the SelectAdminActivity page

4.1.2.3 Functional Requirements

REQ-1: The system must allow Admin to input user ID, ban duration, and allow Admin to confirm or cancel ban user action.

REQ-1.1: The system must validate the user ID entered by the Admin to ensure it corresponds to an existing user.

REQ-1.1.1: If user ID cannot be validated, the system must display an error message to prompt admin to enter user ID again.

REQ-1.2: If admin chooses to confirm ban action, the system must provide admin confirmation that ban is successful.

REQ-1.3: If admin chooses to cancel ban action, the system must direct admin back to SelectAdminActivity page.

4.1.3 VerifyUser

4.1.3.1 Description and Priority

Description:

- Allows Admin to verify a User's request for food priority.

Priority:

- High

4.1.3.2 Stimulus/Response Sequences

1. After admin selects the Verify User option from SelectAdminActivity page, admin is directed to the VerifyUser page.
    a) App displays a list of consumers that have submitted a Priority Food Request application and allows admin to select a consumer from the list.
2. Admin selects a consumer
    b) App displays all information related to the Priority Food Request and allows admin to select the following actions: Approve, Reject, Request.
    c) Admin selects Approve
        i) System processes approval of priority requests, and updates the consumer to a priority consumer.
        ii) Consumer is notified of approved priority request through the NotifyUser feature
    d) Admin selects Reject
        i) System processes rejection of the priority request.
        ii) Consumer is notified of rejected priority request through the NotifyUser feature
    e) Admin selects Request
        i) System processes request, and updates the priority request status to pending
        ii) The Consumer is notified of the pending priority request, and is required to provide additional details through the NotifyUser feature.

4.1.3.3 Functional Requirements

REQ-1: App must display list of consumers and allow admin to select a consumer.

REQ-1.1: After admin has selected a consumer, app must display consumer information as stated in the SubmitPriorityFood feature, and allow admin to perform the following actions: Approve, Reject, Request

REQ-1.2: Upon selection of an action, the app must process the action and notify consumers of the status of their priority food application.

4.1.4 Process Review

4.1.4.1 Description and Priority

Description:

- Allows Admin to process reviews that have been flagged and decide whether to delete or keep it.

Priority:

- Medium

4.1.4.2 Stimulus/Response Sequences

1. After admin selects the Process Review option from SelectAdminActivity page, admin is directed to the ProcessReview page.
    a) App displays a list of reviews that have been flagged and allows admin to select a review from the list.
2. Admin selects a review to process.
    a) App displays the selected review and allows the admin to select from the following actions: Delete, Ignore.
    b) Admin selects delete
        i) System processes deletion of review, and review is deleted and removed.
    c) Admin selects ignore
        i) System removes the review from the list of flagged reviews.

4.1.4.3 Functional requirements

REQ-1: App must display list of flagged reviews and allow admin to select a flagged review.
REQ-1.1: After the admin has selected a review, app must display review information as stated in the SubmitReview feature, and allow admin to perform the following actions: Delete, Ignore.
REQ-1.2: Upon selection of an action, the app must process the action.

4.1.5 Notify User

4.1.5.1 Description and Priority

Description:

- Allows Admin to send a notification to a user

Priority:

- Medium

4.1.5.2 Stimulus/Response Sequences

1. After the admin has successfully logged in, Admin can send notifications to users.
2. Admin can input notification content: Title, Description, and specify recipient user ID.
3. Admin is allowed to select from the following options: Confirm, Cancel.
    a. If admin selects Confirm
        i. System will send the notification to the specified recipient user.
    b. If admin selects Cancel
        i. System will cancel the action and redirect the admin back to the previous page.

4.1.5.3 Functional requirements

REQ-1: App must allow admin to send notification to a specified recipient user.

REQ-1.1: After admin has entered necessary inputs (title, description, recipient userID), app must verify inputs.

REQ-1.1.1: If the system cannot verify recipient User ID, an error message is displayed and admin is prompted to enter recipient user ID again.

REQ-1.2: System will send the notification to the user.

4.1 Hawker Features

4.2.1 SubmitLeftOverFood

4.2.1.1 Description and Priority

Description:

- Hawkers can submit their leftover food and list it onto the app.

Priority:

- High

4.2.1.2 Stimulus/Response Sequences

1. After hawkers have successfully logged in, hawker can submit leftover food unto the FeedfItForward platform.
2. Hawkers can input the amount of leftover food, unit of measurement (of the leftover food), name, amount of time passed (in hours) since the food was cooked, and upload a picture of the leftover food.
3. After entering all the information related to the leftover food, hawkers can confirm the action.
4. System will process submission, and consumers can now view the newly added listing.

4.2.1.3 Functional requirements

REQ-1: App must allow Hawkers to submit a priority request.
REQ-2: App must direct Hawkers to the Submit Leftover Food page.
REQ-2.1: App must allow Hawkers to input the name of food.
REQ-2.2: App must allow Hawkers to input the amount of food.
REQ-2.3: App must allow Hawkers to input the unit of measurement of food.
REQ-2.4: App must allow Hawkers to input time passed since food was cooked.
REQ-2.5: App must allow Hawkers to attach a photo of the food.
REQ-2.6: App must allow Hawkers to confirm the submission.
REQ-2.6.1: App to process submission and add food unto food listing screen.

4.2 Consumer Features

4.3.1 Submit Review

4.3.1.1 Description and Priority

Description:

- Consumers can submit a review on a Hawker by uploading pictures, writing a review text and giving a rating score.

Priority:

- High

4.3.1.2 Stimulus/Response Sequences

1. After Consumers have successfully logged in, Consumers select to submit a review on a hawker.
2. Consumers can input a picture, review text and give a rating score.

4.3.1.3 Functional Requirements

REQ-1: App must allow Consumers to select Review for any Hawker.
REQ-2: App must direct Consumers to the Submit Review page.
REQ-2.1: App must allow Consumers to upload a picture, enter text review and give a rating score.

4.3.2 Edit Review

4.3.2.1 Description and Priority

Description:

- Consumers can edit their previously submitted review on a Hawker.

Priority:

- Medium

4.3.2.2 Stimulus/Response Sequences

1. After Consumers have successfully logged in, Consumers select “Edit” beside his/her review on a Hawker.
2. Consumers can upload a new picture, edit the text review and update the rating score given.
3. Consumers can select “Save” to save changes or “Cancel” to go back to the previous screen.

4.3.2.3 Functional Requirements

REQ-1: App must allow Consumers to select Edit on their previously submitted review on a Hawker. If no review has been made, “Edit” button will not be available.
REQ-2: App must direct Consumers to the Edit Review page.
REQ-2.1: App must allow Consumers to upload a picture, edit text review and update rating score.

REQ-2.2: App must allow Consumers to select “Save” to save changes or “Cancel” to be redirected back to the previous screen.

4.3.3 Request Food

4.3.3.1 Description and Priority

Description:

- Consumers can choose from a list of food available for request. System will display the amount of food, unit of measurement of food, picture of food, amount of time (in hours) that has passed. Consumers can submit a request for the chosen food.

Priority:

- High

4.3.3.2 Stimulus/Response Sequences

1. After Consumers have successfully logged in, Consumers will be redirected to the Home Screen. Home Screen will display a list of food available for request.
2. Consumers select food to submit a request.
3. App will redirect consumers to the Request Food page. Consumers can select “Delivery” or “Self-Collection” and indicate the number of servings.
4. Consumer to select “Submit” to submit the request.
    a. Hawker associated with the food request will be notified by the system.
    b. If the Consumer chooses the food delivery option, the system will assign a Driver to pick up the food.

4.3.3.3 Functional Requirements

REQ-1: App must display the list of food available and allow Consumers to select.
REQ-2: App must direct Consumers to the Submit Request page.
REQ-2.1: App must allow Consumers to select between “Delivery” or “Self Collection”
REQ-2.2: App must allow Consumers to indicate number of servings.
REQ-2.3: App must allow Consumers to submit the request.
REQ-2.3.1: If Consumer chooses the food delivery option, App to assign a Driver to pick up the food.

4.3.4 Submit Priority Request

4.3.4.1 Description and Priority

Description:

- Consumers can submit a request to gain priority for food request acceptance.

Priority:

- High

4.3.4.2 Stimulus/Response Sequences

1. After Consumers have logged in, Consumers can select to submit a food priority request.
2. Consumers input their personal information like household income, number of residents in household, occupation and type of housing.
3. Consumers to select “Submit” to submit the request.
    a. System will send the request to Admins for Admins to verify the information.

4.3.4.3 Functional Requirements

REQ-1: App must allow Consumers to submit a priority request.
REQ-2: App must direct Consumers to the Submit Priority Request page.
REQ-2.1: App must allow Consumers to input household income.
REQ-2.2: App must allow Consumers to input the number of residents residing in the household.
REQ-2.3: App must allow Consumers to input occupation.
REQ-2.4: App must allow Consumers to input house category.
REQ-2.5: App must allow Consumers to submit the request.
REQ-2.5.1: App to send the request to Admins for Admins to Accept/Reject/Request.

4.3 Driver Features

4.4.1 Process Pick Up Job

4.4.1.1 Description and Priority

Description:

- This feature allows drivers to select whether to accept or ignore a pick up job allocated to them. If drivers choose to accept the job, drivers will have to submit proof of pick and delivery. If drivers choose to ignore the job, other jobs will be displayed. Completed job history will be shown as well.

Priority:

- High

4.4.1.2 Stimulus/Response Sequences

1. After Drivers have successfully logged in, Drivers can go to the pick up job page.
2. System display job available for Driver and history of completed jobs. Drivers can select to accept or ignore a pickup job allocated to them.
3. If a job is accepted, the system will display “Job in Progress”.
    a. System will require Driver to submit proof of pick up, which includes a picture of food at pick up location and a picture of hawker at pick up location.
    b. System will require Driver to submit proof of delivery, which includes a picture of food at the drop off location and a picture of the consumer at the drop off location.
4. If a job is ignored, the system will display other jobs.

4.4.1.3 Functional Requirements

REQ-1: App must allow Drivers to select Pick Up Job from HomeScreen.
REQ-2: App must direct Drivers to the Pick Up Job page.
REQ-2.1: App must display completed job history, if any.
REQ-2.2: App must allow Drivers to accept or ignore a job.
REQ-2.3: If Drivers accept a job, App must allow Drivers to submit proof of pick and proof of delivery.
REQ-2.4: If Drivers ignore a job, App must display other jobs available for pick up.

4.4 Customer Service Support

4.5.1 Customer Service Support

4.5.1.1 Description and Priority

Description:

- Real-time text messaging communication between Consumer/Hawker/Driver and Admin, and vice versa.

Priority:

- High

4.5.1.2 Stimulus/Response Sequences

1. After users have successfully logged in, users can go to the customer service support page.
    a. Users can select to chat and the chat display will be shown. If there is chat history, chat history will be shown.
    b. Users can send text messages.

4.5.1.3 Functional Requirements

REQ-1: App must allow Users to select Customer Service Support from HomeScreen.
REQ-1.1: App must direct Users to the Customer Service Support page.
REQ-1.1.1: App must display chat history, if any.
REQ-1.1.2: App must allow users to send text messages.

4.5 User Features

4.6.1 Query Hawkers

4.6.1.1 Description and Priority

Description:

1) System displays a list of all registered Hawkers at the HomeScreen, with their detailed information.
2) Using the map feature, Users can see all registered and unregistered Hawkers
    a) For registered Hawkers, detailed information such as Name, Cuisine, Ratings, Contact number, Email, and Address, are displayed.
    b) For unregistered Hawkers, only Name and Address are displayed

Priority:

- High

4.6.1.2 Stimulus/Response Sequences

1. After Users have successfully logged in, Users will be directed to HomeScreen, where the list of all registered Hawkers will be displayed, with their information.
    a. Users can select specific Hawkers to view more detailed information.

2. From the HomeScreen, User can select the map tab, which will direct User to MapScreen.
    a. Users can scroll around and select specific registered or unregistered Hawkers.
        i. The system will display their corresponding information.

4.6.1.3 Functional Requirements

REQ-1: App must display detailed information of registered Hawkers in HomeScreen.
REQ-1.1: App must display Hawker's photo.
REQ-1.2: App must display Hawker's name.
REQ-1.3: App must display Hawker's operating hours.
REQ-1.4: App must display Hawker's address.
REQ-1.5: App must display Hawker's ratings.

REQ-1.6: App must allow Users to select specific Hawkers from HomeScreen.
REQ-1.6.1: App must direct Users to the selected Hawker's page.
REQ-1.6.1.1: App must display Hawker's name.
REQ-1.6.1.2: App must display Hawker's ratings and reviews.
REQ-1.6.1.3: App must display Hawker's operating hours.
REQ-1.6.1.4: App must display Hawker's address
REQ-1.6.1.5: App must display Hawker's leftover food.

REQ-2: App must allow Users to select map tab.
REQ-2.1: App must allow Users to scroll around the interactive map.
REQ-2.2: App must display icons of all registered and unregistered Hawkers.
REQ-2.2.1: App must allow Users to select specific icons.
REQ-2.2.1.1: For registered Hawkers,
REQ-2.2.1.1.1: App must display "Registered" tag.

REQ-2.2.1.1.2: App must display Hawker's name.
REQ-2.2.1.1.3: App must display Hawker's cuisine.
REQ-2.2.1.1.4: App must display Hawker's ratings.
REQ-2.2.1.1.5: App must display Hawker's photo.
REQ-2.2.1.1.6: App must display Hawker's contact number.
REQ-2.2.1.1.7: App must display Hawker's email.
REQ-2.2.1.1.8: App must display Hawker's address.
REQ-2.2.1.1.9: App must display a button to allow Users to be directed to a Hawker's individual page.

REQ-2.2.1.2: For unregistered Hawkers,
REQ-2.2.1.2.1: App must display "Public" tag.
REQ-2.2.1.2.2: App must display Hawker's name.
REQ-2.2.1.2.3: App must display Hawker's photo.
REQ-2.2.1.2.4: App must display Hawker's address.

4.6.2 Search Hawkers

4.6.2.1 Description and Priority

Description:

- Allows Users to search for Hawkers by their name and food type via a search bar and get a filtered list of Hawkers.

Priority:

- High

4.6.2.2 Stimulus/Response Sequences

1. After Users have successfully logged in, Users will be directed to HomeScreen, where there will be a search bar.
    a. Users can key in specific Hawkers' name or food type. (either part of whole)
        i. The system will return a filtered list of Hawkers for Users.
            1. The system will display the Hawkers' information
                a. Users can select a specific Hawker to view more detailed information.

2. From the HomeScreen, User can select the map tab, which will direct User to MapScreen.
    a. Users can select the search icon at the bottom left, to search for Hawkers.
        i. The system will then display a search bar.
            1. Users can key in specific Hawkers' name (either part of whole).
                a. The system will return a filtered list of Hawkers for Users.
                    i. Users can select a specific Hawker to view more detailed information.

4.6.2.3 Functional Requirements

REQ-1: App must display search bar in HomeScreen.
REQ-1.1: App must allow Users to key in characters in the search bar .
REQ-1.1.1: App must display a filtered list of Hawkers in HomeScreen, with their detailed information.
REQ-1.1.1.1: App must display Hawker's photo.
REQ-1.1.1.2: App must display Hawker's name.
REQ-1.1.1.3: App must display Hawker's operating hours.
REQ-1.1.1.4: App must display Hawker's address.
REQ-1.1.1.5: App must display Hawker's ratings.

REQ-1.1.1.6: App must allow Users to select specific Hawkers from HomeScreen.
REQ-1.1.1.6.1: App must direct Users to the selected Hawker's page.
REQ-1.1.1.6.1.1: App must display Hawker's name.
REQ-1.1.1.6.1.2: App must display Hawker's ratings and reviews.
REQ-1.1.1.6.1.3: App must display Hawker's operating hours.
REQ-1.1.1.6.1.4: App must display Hawker's address
REQ-1.1.1.6.1.5: App must display Hawker's leftover food.

REQ-2: App must allow Users to select map tab.
REQ-2.1: App must allow Users to select search icon.
REQ-2.1.1: App must display search bar.
REQ-2.1.1.1: App must allow Users to key in characters in the search bar.
REQ-2.1.1.1: App must display a filtered list of Hawkers.
REQ-2.1.1.1: App must allow users to select specific Hawker.
REQ-2.1.1.1.1: For registered Hawkers,
REQ-2.1.1.1.1.1: App must display "Registered" tag.
REQ-2.1.1.1.1.2: App must display Hawker's name.
REQ-2.1.1.1.1.3: App must display Hawker's food type.
REQ-2.1.1.1.1.4: App must display Hawker's ratings.
REQ-2.1.1.1.1.5: App must display Hawker's photo.
REQ-2.1.1.1.1.6: App must display Hawker's contact number.
REQ-2.1.1.1.1.7: App must display Hawker's email.
REQ-2.1.1.1.1.8: App must display Hawker's address.
REQ-2.1.1.1.1.9: App must display a button to allow Users to be directed to a Hawker's individual page.
REQ-2.1.1.1.2: For unregistered Hawkers,
REQ-2.1.1.1.2.1: App must display "Public" tag.
REQ-2.1.1.1.2.2: App must display Hawker's name.
REQ-2.1.1.1.2.3: App must display Hawker's photo.
REQ-2.1.1.1.2.4: App must display Hawker's address.

4.6.3 Query Weather

4.6.3.1 Description and Priority

Description:

- The system to display the current weather and Singapore's weather forecasted 1 day into the future at a 6-hours interval, and 4 days into the future at a 1 day interval.

Priority:

- Medium

4.6.3.2 Stimulus/Response Sequences

1. After Users have successfully logged in, Users will be directed to HomeScreen, where the system will display a map tab at the navigation bar.
    a. Users can select the map tab, which will direct them to MapScreen.
        i. The system will then display an interactive map, with a cloud icon at the bottom left.
            1. Users can select the icon to query weather conditions.

4.6.3.3 Functional Requirements

REQ-1: App must display cloud icon in MapScreen.

REQ-1.1: App must allow Users to select cloud icon in MapScreen.

REQ-1.1.1: App must display the current weather and Singapore's weather forecasted 1 day into the future at a 6-hours interval, and 4 days into the future at a 1 day interval.

4.6.4 Flag Review

4.6.4.1 Description and Priority

Description:

- Allows User to flag a review for the Admin to process for deletion.

Priority:

- Medium

4.6.4.2 Stimulus/Response Sequences

1. Users call view all the reviews for specific Hawkers.
    a. Users can select a specific review to flag.
    b. System will prompt the user input reason.
    c. User to select “Confirm” to flag the review
        i. Review will be sent to Admins to process.

4.6.4.3 Functional Requirements

REQ-1: App must display flag review option beside every review.
REQ-2: App must direct Users to the flag review page.
REQ-2.1: App must allow Users to input reason.
REQ-2.2: App must allow Users to confirm flagging of review.
REQ-2.2.1: App to send the review to Admins to process.

4.6 Authentication

4.6.1 Create account

Description and Priority

Description:

- Allows a person to create an Account in the FeedItForward to become a User. An Account with a specific role (Admin, Hawker, Consumer or Rider) is created for the User.

Priority:

- High

Stimulus/Response Sequences

1. At the LoginScreen, people without an Account with FeedItForward can click the “Sign Up” link to create an Account with FeedItForward.
    a. The system will direct the users to the SignupScreen, and prompt them to enter required information.
        i. The users can then key in all the required information, and select the “Create Account” button to confirm the inputs.
            1. The system will then create an Account with a specific role (Admin, Hawker, Consumer or Rider) for Users.

Functional Requirements

REQ-1: App must allow the users to select the “Sign Up” link.
REQ-1.1: App must prompt the users to enter required information.
REQ-1.1.1: App must display input field for “Name”.
REQ-1.1.1.1: App must display an error message if this field has no input.
REQ-1.1.2: App must display input field for “Email”.
REQ-1.1.2.1: App must display an error message if this field has no input.
REQ-1.1.3: App must display input field for “Address”.
REQ-1.1.3.1: App must display an error message if this field has no input.
REQ-1.1.4: App must display input field for “Contact Number”.
REQ-1.1.4.1: App must display an error message if this field has no input.
REQ-1.1.5: App must display upload field for “Profile Picture”.
REQ-1.1.5.1: App must display an error message if this field has no upload, or a wrong file type is uploaded.
REQ-1.1.6: App must display input field for “Contact Number”.
REQ-1.1.6.1: App must display an error message if this field has no input.
REQ-1.1.7: App must display a drop down button for the users to select their roles (Admin,. Consumer, Driver, or Hawker).
REQ-1.1.8: App must display input field for “Password”.
REQ-1.1.8.1: App must display an error message if this field has no input, or the length of the input is less than 8 characters.
REQ-1.1.9: App must display input field for “Confirm Password”.
REQ-1.1.9.1: App must display an error message if this field has no input, or the length of the input is less than 8 characters, or the input does not match with the above “Password” field.

REQ-1.1.10: App must prompt the users who selected (Driver, or Hawker) for additional inputs.

- REQ-1.1.10.1: For “Driver”,
  - REQ-1.1.10.1.1: App must display input field for “Vehicle Number”.
    - REQ-1.1.10.1.1.1: App must display an error message if this field has no input.
  - REQ-1.1.10.1.2: App must display input field for “License Number”.
    - REQ-1.1.10.1.2.1: App must display an error message if this field has no input.

- REQ-1.1.10.2: For “Hawker”,
  - REQ-1.1.10.2.1: App must display input field for “Business Name”.
    - REQ-1.1.10.1.1.1: App must display an error message if this field has no input.
  - REQ-1.1.10.2.2: App must display input field for “Operating Hours”.
    - REQ-1.1.10.1.2.1: App must display an error message if this field has no input.
  - REQ-1.1.10.2.3: App must display input field for “Food Type”.
    - REQ-1.1.10.1.1.1: App must display an error message if this field has no input.
  - REQ-1.1.10.2.4: App must display input field for “Postal Code”.
    - REQ-1.1.10.1.2.1: App must display an error message if this field has no input, or if the postal code is invalid.

REQ-1.1.11: App must display a “Create Account” button for the users to select once all input has been keyed in.

REQ-1.1.12: App must create an Account for the users once all information have been approved and verified.

4.6.2 Login

Description and Priority

Description:

- Allows User to log into his/her FeedItForward Account using his/her email and password.

Priority:

- High

Stimulus/Response Sequences

1. Users who choose to login with email and password, will have to key in their email address and password into the input fields.
    a. Users will then click the “Login” button.
        i. The system will then validate the input data.
            1. Once authentication is successful, Users will be directed to the HomeScreen.

Functional Requirements

REQ-1: App must display the option to log in with email and password, or Google.
REQ-1.1: App must display input fields to prompt Users for required information.
REQ-1.1.1: App must display input field for “Email”.
REQ-1.1.1.1: App must display an error message if this field has no input, or is of the wrong email type.
REQ-1.1.2: App must display input field for “Password”.
REQ-1.1.2.1: App must display an error message if this field has no input, or the length of the input is less than 8 characters.
REQ-1.2: App must display a “Login” button for Users to click once all required information has been entered.
REQ-1.2.1: App must then validate user input data.
REQ-1.2.1: App must log Users in successfully if User has an existing FeedItForward Account, and the input data are all correct.
REQ-1.2.1.1: App must direct Users to HomeScreen after successful login.
REQ-1.2.1: App must display error message if User does not have an existing FeedItForward Account, or the input data are incorrect.

4.6.3 Login with Google

Description and Priority

Description:

- Allows User to log into his/her FeedItForward Account using his/her Google account.

Priority:

- Medium

Stimulus/Response Sequences

1. Users who choose to login with google, will be redirected to Google Authentication User Interface.
    b. Users will have to sign in to their existing Google account.
        i. Once Google authentication is successful, User will be directed back to FeedItForward.

Functional Requirements

REQ-1: App must display the option to log in with email and password, or Google.
REQ-1.1: App must redirect Users to Google Auth User Interface to provide verification.
REQ-1.1.1: App must direct Users to FeedItForward's HomeScreen, if the Google Auth verification is successful.
REQ-1.1.2: App must redirect User to the FeedItForward's LoginScreen and display an error message, if Google Auth verification was unsuccessful.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

1) Each page should load in no more than 3 seconds, including logging in and authentication.
2) Errors that could arise from synchronization issues like two or more consumers request for the same leftover food at the same time, or two or more drivers accepting the same pick-up job, should be kept to a maximum of no more than 100 times per month.
3) The mobile application should be scalable enough to support at least 300,000 users at the same time while maintaining optimal performance.
4) The total error rate of hawkers submitting leftover food, consumers submitting leftover food requests, and drivers completing their pick-up jobs, should be kept below 1 percent.

5.2 Safety Requirements

1) The system shall not impose a time limit for delivery jobs to ensure the safety of drivers on the roads.
2) The system shall allow hawkers to input time elapsed since the food was cooked and display such information to consumers, to allow consumers make informed choices and to prevent any food safety or health hazards.
3) The application will not require the sharing of any personal information between non-admin users even for the messaging/chat functions to ensure the safety of all users.

5.3 Security Requirements

1) The user's residential address and other personal information shall be kept confidential and shall not be disclosed to other users of the application or the public, in line with the PDPA.
2) User authentication will always be necessary before accessing other features of the app.
3) The system should implement password hashing for storing the users' passwords to protect users' accounts.
4) There should be zero tolerance for data leaks or compromise and the database should be stored securely by limiting access to the database to key developers and admins of the application.

5.4 Software Quality Attributes

1) The code will always be accompanied with the relevant code comments indicating at least the purpose of the following block of code, to allow for interpretability, code reusability and future maintenance.
2) The API endpoints implemented by the backend server shall always be accompanied with a documentation to indicate the endpoint url, required request body, and expected response body.
3) The system will display the appropriate error or success messages when actions are done by users, to improve testability of the product.
4) The application should be able to run on both iOS and Android operating systems without any major differences in behavior and performance, for portability.
5) To ensure reliability, the application should only crash or fail to start at most 1% of the time.

5.5 Business Rules

The business rules provide a framework for the necessary functionality that the "Community FoodLink Platform" will require, and they help to ensure that the system operates within the intended legal and ethical boundaries.

5.5.1 Review and Rating System

1) Users can only leave a review for a hawker or driver after a completed food transaction.
2) Reviews must adhere to community guidelines, and inappropriate content is subject to removal.

5.5.2 Transactional Integrity

1) All transactions, including food listings, requests, and deliveries, must be logged and time-stamped for traceability.
2) Any financial transactions must be processed through secure, encrypted channels and comply with PCI DSS standards.

5.5.3 Operational Hours

1) The platform will operate 24/7, but customer service support via messaging will only be available during standard business hours unless otherwise stated.

5.5.4 Function Restrictions

Table 5.5.4.1 and 5.5.4.2 showcases the functions that different types of FeedItForward users are able to perform.

Actors	**User**	**Food**	**Food Request**
Hawker	createUser() updateUser()	createFood() updateFood() updateAvailableFood()	cancelRequest()
Driver	createUser() updateUser()		denyRequest()
Admin	createUser() updateUser() getAllUser() banUser()	createFood() updateFood() getAvailableFood() updateAvailableFood() getAllFood()	cancelRequest() denyRequest() setFulfilled() setDriver()
Customer	createUser() updateUser()	getAvailableFood()	cancelRequest()

Table 5.5.4.1

Actors	**Reviews**	**Customer service**	**Priority Request**
Hawker	getReviews() flagReview()	getHistory() clearHistory()	
Driver	getReviews() flagReview()	getHistory() clearHistory()	
Admin	getReviews() getFlaggedReviews() deleteReview() clearFlaggedReview()	getHistory() clearHistory()	getAllDocuments() getDocument() deleteDocument() updateDocument() approve() reject() request_info()
Customer	getReviews() createReview() flagReview()	getHistory() clearHistory()	getOwnDocument() deleteOwnDocument() updateOwnDocument()

Table 5.5.4.2

6. Other Requirements

6.1 Internationalisation Requirements

1) Initially, FeedItForward will support only the English language. Future releases will consider multi-language support such as dialects (e.g. Hokkien, Cantonese) to cater to a broader audience in Singapore, especially senior citizens who cannot understand english.
2) Date and time representations should be in the Singapore timezone (GMT+8).

6.3 Legal Requirements

1) Compliance with PDPA when handling user's sensitive data.
2) Adherence to local food safety and handling regulations for the distribution of edible goods.

6.4 Reuse Objectives

1) Components developed for this platform should be modular and reusable for within the project and for potential future projects.
2) APIs should be designed with industry standards (REST API) to facilitate external usage of our platform through APIs.

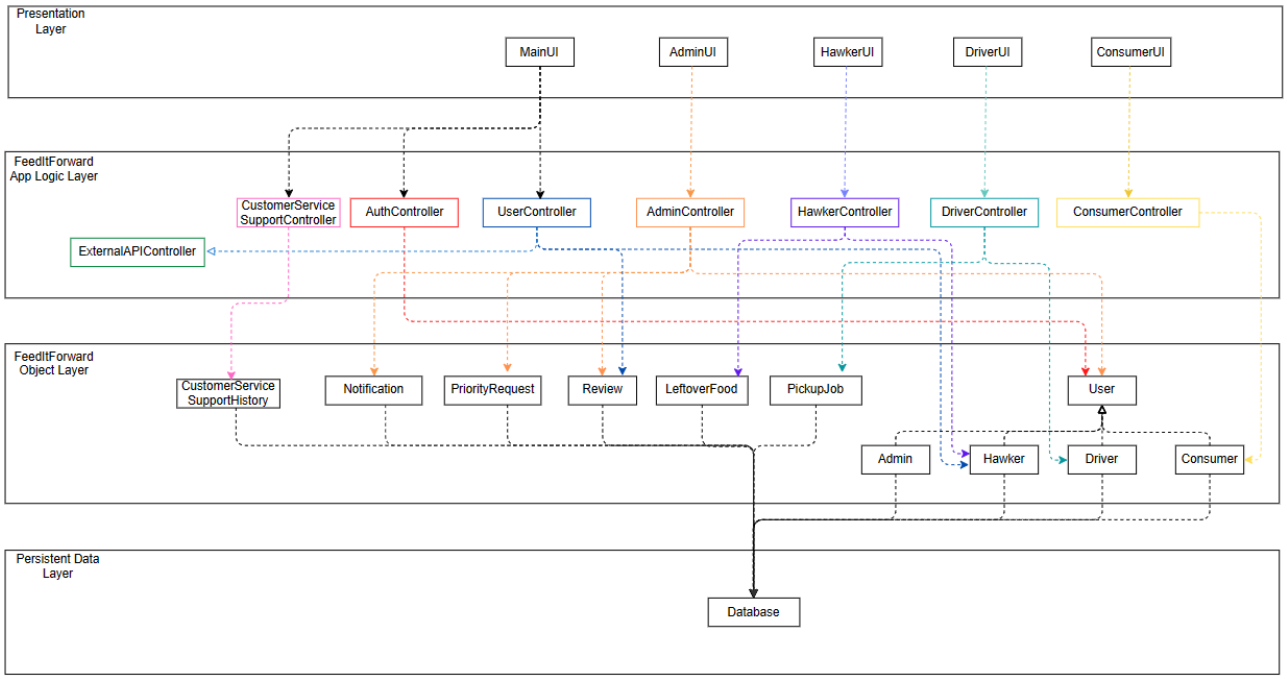
6.5 Accessibility Requirements:

1) Must meet WCAG 2.1 Level AA accessibility standards to ensure it is usable by people with disabilities.

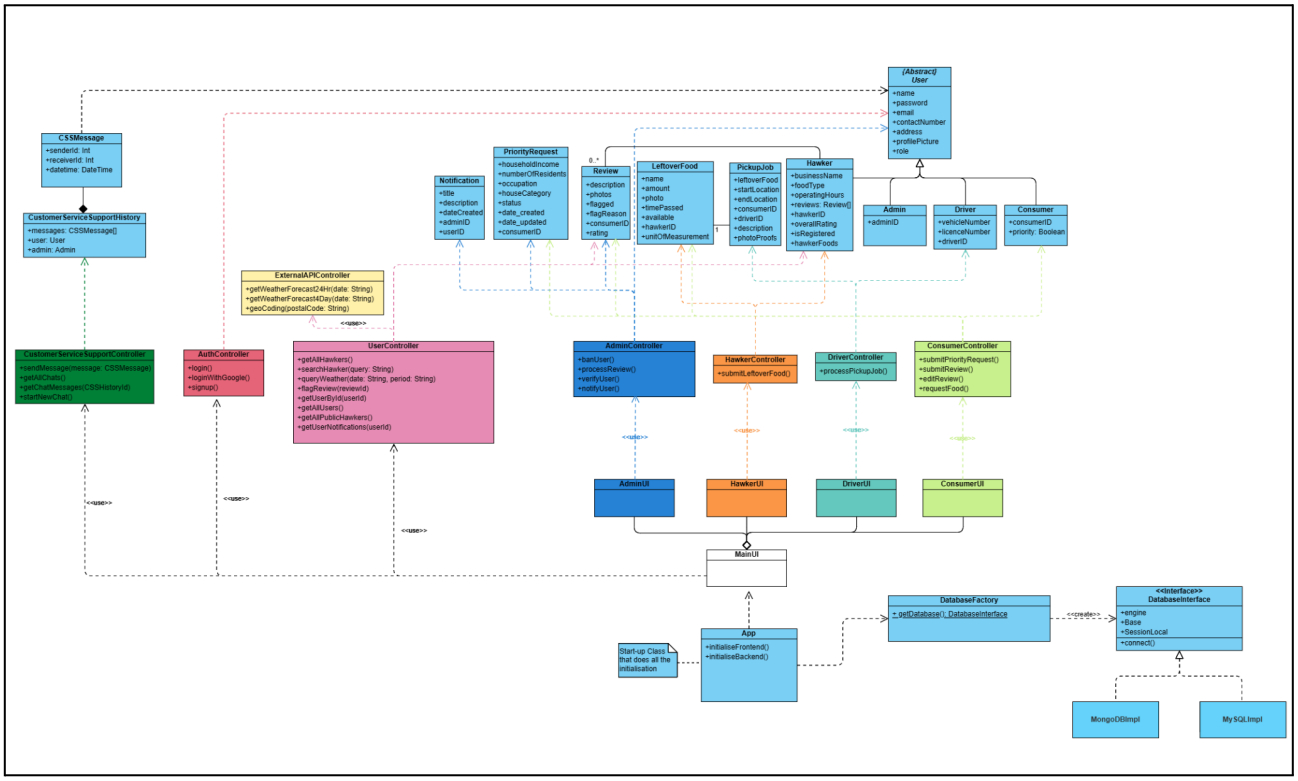
Appendix A: Glossary

Acronym / Abbreviation	Meaning
API	Application Programming Interface - A set of protocols for building and interacting with software applications.
PDPA	Personal Data Protection Act - Provides a baseline standard of protection for personal data in Singapore.
SQL	Structured Query Language - A domain-specific language used in programming for managing relational databases.
WCAG	Web Content Accessibility Guidelines - Part of a series of web accessibility guidelines published by the W3C.
Front-end	The client-side part of a web application that users interact with directly - It consists of everything the user experiences directly: from text and images to sliders and buttons.
Back-end	The server-side of a web application that handles database interactions, server logic, and application integration.
SRS	Software Requirements Specification - A document that describes what the software will do and how it will be expected to perform. It includes a set of use cases that describe all the interactions the users will have with the software.
PCI DDS	Payment Card Industry Data Security Standard - A set of security standards designed to ensure that all companies that accept, process, store, or transmit credit card information maintain a secure environment.

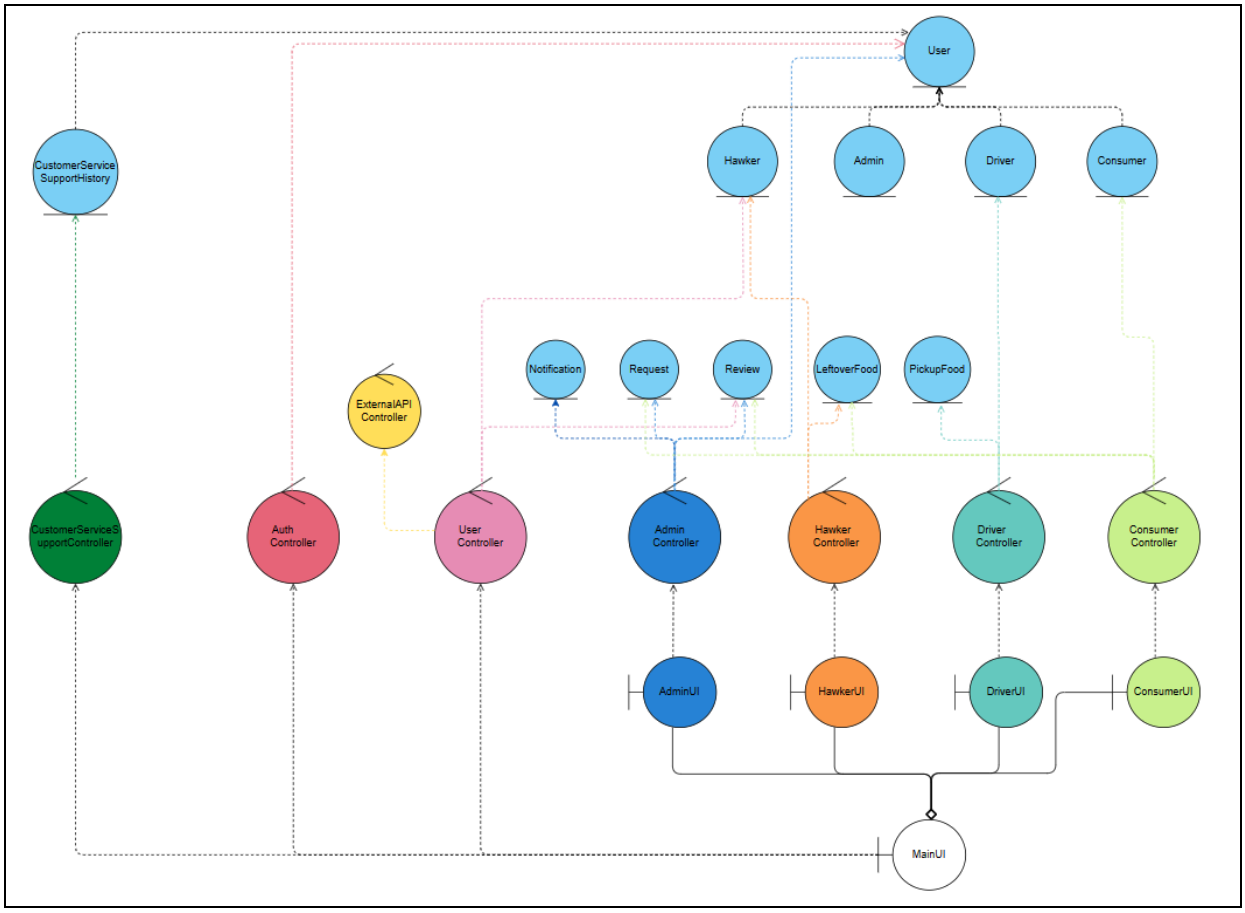
Appendix B: Analysis Models



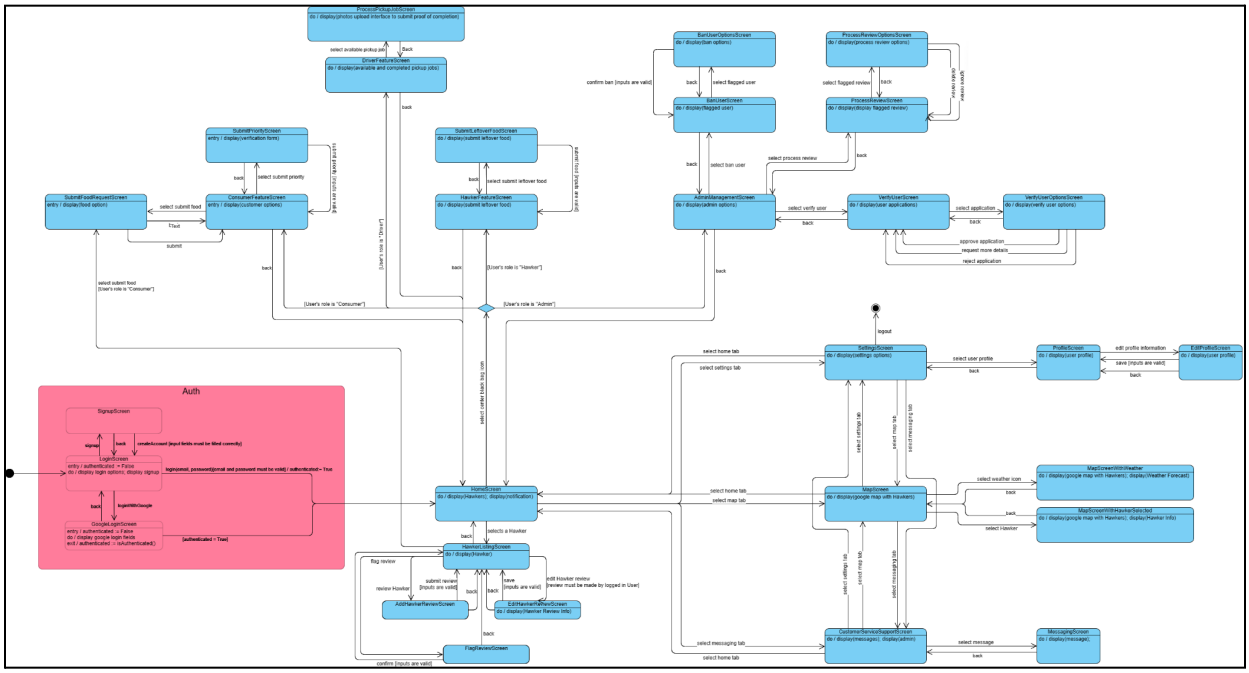
Architecture Diagram



Class Diagram



Stereotype Class Diagram



Dialog Map

Appendix C: To Be Determined List

There are no TBD items for this version of Software Requirement Specification.

Source: http://www.frontiernet.net/~kwiegers/process_assets/srs_template.doc