



SC2006: Software Engineering

Software Requirements Specifications

Lab Group	ACDA2
Team	Hawkar
Members	Thant Htoo Aung (U2220809L)
	Cao Jun Ming (U2310254A)
	Kow Zi Ting (U2310485B)
	Muhammad Aliff Amirul Bin Mohammed Ariff (U2322581A)
	Saeng-nil Natthakan (U2220832B)

1. Introduction	4
1.1. Purpose	4
1.2. Scope	4
1.3. Definitions, Acronyms and Abbreviations	4
1.3.1. Data Dictionary	4
1.4. References	5
2. Overall Description	6
2.1. Product Perspective	6
2.2. Product Functions	6
2.3. User Classes and Characteristics	7
2.4. Operating Environment	7
2.5. Design and Implementation Constraints	8
2.5.1. Frontend Constraints	8
2.5.1.1. Framework (Frontend)	8
2.5.1.2. Code Formatting (Frontend)	8
2.5.2. Backend Constraints	8
2.5.2.1. Framework (Backend)	8
2.5.2.2. Database	8
2.5.2.3. Code Formatting (Backend)	8
2.6. System Architecture	9
2.7. Application Skeleton	11
2.8. User Documentation	12
2.8.1. README Files	12
2.8.1.1. Main Repository	12
2.8.1.2. Frontend Repository	12
2.8.1.4. Code Comments	12
2.8.1.5. Backend API Documentation	12
2.9. Assumptions and Dependencies	13
2.9.1. Assumptions	13
2.9.2. Dependencies	13
2.9.2.1. Frontend Libraries	13
2.9.2.2. Backend Libraries	13
2.9.2.3. External Services	14
2.9.2.4. Continuous Integration and Deployment (CI/CD)	14
3. Specific Requirements	15
3.1. Functional Requirements	15
3.2. Non-functional Requirements	20
3.3. External Interface Requirements	21
3.3.1. User Interfaces	21
3.3.1.1. UI Mockups	21
3.3.2. Hardware Interfaces	29

3.3.3. Software Interfaces	30
4. System Features	31
4.1. User(base class) Features	31
4.2. Consumer Features	33
4.4. Hawker Features	39
4.5. Admin Features	43
5. System Architecture and Design	46
5.1. Class Diagram	46
5.2. Stereotype Class Diagram	47
5.3. Dialog Map	48
6. Appendices	48

1.Introduction

1.1. Purpose

This document delineates the software requirements for “Hawkar, Version 1.0”. This SRS serves to document the functionalities, features, design considerations and constraints of the application. Following the IEEE 830 standard, it is intended to guide future development, testing and maintenance of Hawkar to ensure alignment with its core objectives.

1.2. Scope

Hawkar is a platform of discovery and recommendation that connects hawkers and consumers based upon their needs, preferences and requirements. By integrating search filters, wait-time predictions, real-time notifications, and interactive maps, the project benefits both consumers seeking to find their ideal hawker choice and hawkers looking to attract and retain customers by enhancing consumer convenience and experience whilst helping to improve the visibility of hawkers to potential customers. Hawkar does not handle financial transactions or delivery logistics.

1.3. Definitions, Acronyms and Abbreviations

1.3.1. Data Dictionary

The technical terms we will use in this document include the following:

Term	Definition
Account	A user profile containing credentials and personal details, allowing access to the specific parts of the application based on the user roles (e.g. Consumer, Hawker, Admin)
Admin	A privileged user role responsible for managing the application, overseeing the reported reviews from the consumers.
Application	The web platform that facilitates interactions between consumers, hawkers and administrators for discovering, reviewing and managing hawker-related data.
Consumer	A user of the application who searches for hawker centers and stalls, leaves reviews, and rates the stalls.
Hawker	A vendor or stall owner who operates within a Hawker Stall and provides food or beverages to consumers. Hawkers have the special privilege to update stall details

Map	A visual representation of the geographical locations of Hawker Centres, allowing users to find nearby stalls and navigate efficiently.
Rating	A numerical score provided by consumers to evaluate the quality of food, service or overall experience at a hawker stall on a scale of 1 to 5.
Review	A textual or multimedia-based feedback entry left by a consumer about their experience at a hawker stall, often accompanied by a rating.
Hawker Centre	A food complex housing multiple Hawker stalls, providing a variety of affordable local cuisine options for consumers.
Hawker Stall	An individual food stall within a Hawker Centre, operated by a Hawker, offering special dishes or beverages for sale.

1.4. References

Next.js Documentation: <https://nextjs.org/docs>

Render Deployment Guide: <https://render.com/docs>

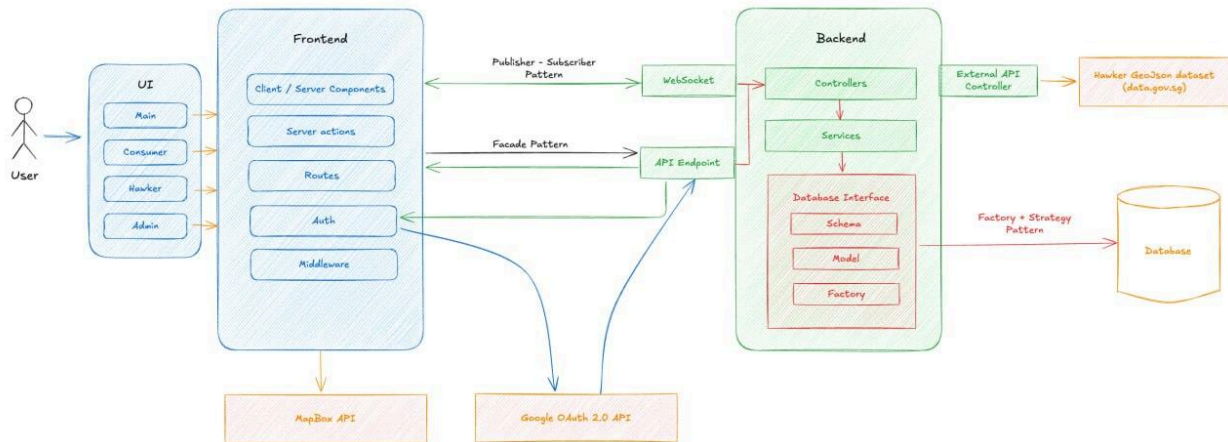
Vercel Deployment Guide: <https://vercel.com/docs>

Google OAuth: <https://developers.google.com/identity/protocols/oauth2>

2. Overall Description

2.1. Product Perspective

Hawkar is a self-contained product designed to facilitate consumers' discovery of hawker stalls and hawkers' promotion of their stalls. As an effort to enrich the existing information available about hawker stalls located around Singapore, Hawkar functions independently and integrates existing external APIs to make the platform a success. The following system architecture diagram is provided to illustrate the interconnections between Hawkar and external interfaces.



2.2. Product Functions

Hawkar will allow users to perform the following functions:

- **Authentication**
 - Account Creation
 - Profile Creation (into different roles: Consumer, Hawker, Admin)
 - Login
 - Display Password
- **Consumer Functions**
 - View and Search for Hawker Stalls
 - Save Favourite Stalls
 - Filter Stalls by Operating Hours, Price Range, Location or Hygiene Ratings
 - Add, Edit, Delete and Report Reviews
 - Sort Reviews by Recency or Rating Values
- **Hawker Functions**
 - Add, Edit and Delete Dishes, Stalls and Promotions
 - Report Reviews
- **Admin Functions**
 - Verify Hawkers
 - Process Reviews
- **Profile Image Uploads**
 - Upload avatars

- **Stall/Dish Images**
 - Stall and dishes can be displayed with images

2.3. User Classes and Characteristics

Hawkar's users fall within the following categories:

- **Consumers:** Individuals or families looking for new eating spots. They may vary in technical expertise.
- **Hawkers:** Vendors operating food and beverage within hawker centres. Often older in age, they may require a straightforward interface.
- **Admins:** Staff regulating interactions between consumers and hawkers. They are assumed to have a strong technical capability and require extensive functions for user management.

These three groups of users share a symbiotic relationship where Hawkers provide comprehensive information about their stalls, Consumers support hawkers by patronising the stalls and leaving reviews, and Admins ensure that the platform remains fair and reliable.

2.4. Operating Environment

The “Hawkar” application is a web-based platform developed using Next.js (a React framework) and styled with Tailwind CSS. It is designed to be platform-independent and runs on modern web browsers. The operating environment can be categorized into two main contexts: development, deployment and user environment.

1. Development Environment
 - Operating System: Window 10/11, macOS or any Unix-based system
 - Node.js: v20.x or later
 - Package Manager: npm or yarn
 - Code Editor: Visual Studio Code
 - Version Control: Git and GitHub
2. Deployment Environment
 - Hosting platform: Vercel (for frontend), Render (for backend)
 - Build Tool: Next.js Build (SSR/ Static Generation)
 - Runtime Requirements:
 - Node.js runtime on the server
 - CDN support for static assets
3. User Environment
 - Supported Browsers: Latest versions of Chrome, Firefox, Edge and Safari
 - Device Compatibility: Desktop, tablet and mobile responsive
 - Internet Connection: Required for full functionality

2.5. Design and Implementation Constraints

2.5.1. Frontend Constraints

2.5.1.1. Framework (Frontend)

The user interface for our Hawkar application is developed using Next.js which is a popular React framework for building optimised web applications and styled with Tailwind CSS. Next.js was chosen for its ability to perform server-side data revalidation and UI updates within a single network request. Its middleware enables request interception for implementing routing and authentication rules. Additionally, Next.js supports both client-side and server-side rendering with configurable caching strategies for optimized performance. Tailwind CSS was chosen for its utility-first approach, enabling rapid, responsive styling through configurable inline class utilities.

2.5.1.2. Code Formatting (Frontend)

Prettier will be used to ensure consistent code formatting and adherence to coding best practices.

2.5.2. Backend Constraints

2.5.2.1. Framework (Backend)

FastAPI will be used to develop the backend server in Python.

2.5.2.2. Database

We will be using SQL-based databases, SQLite and PostgreSQL.

We will designate SQL as the language for database queries as this will allow for the efficient and secure manipulation of data.

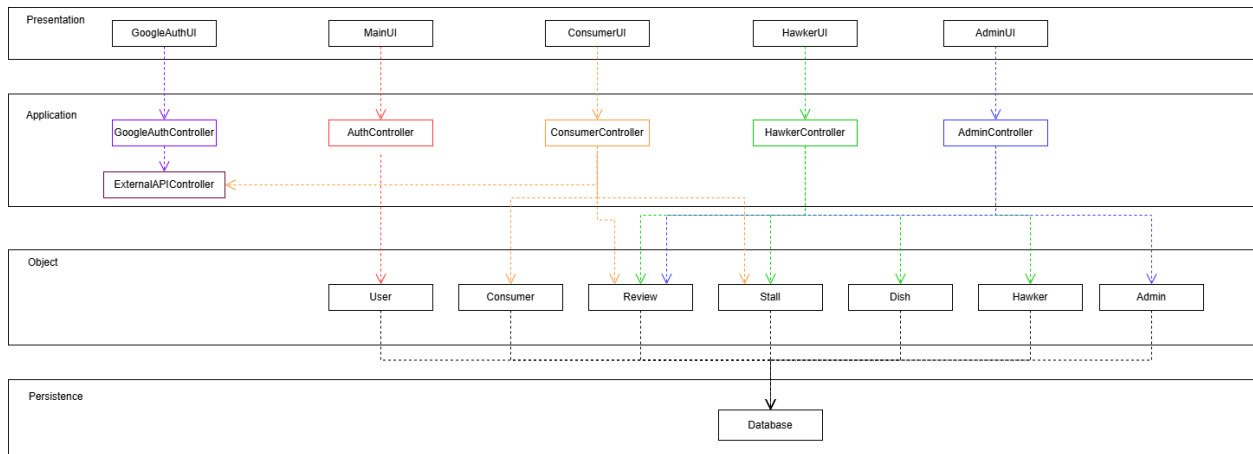
2.5.2.3. Code Formatting (Backend)

autopep8 which is a Python code formatting tool from Microsoft is used to ensure consistent code formatting for Python files

2.6. System Architecture

Our application implements a layered system architecture. Such a design allows us to segment our application into more manageable layers.

If the image is unclear, please refer to the accompanying raw PNG file uploaded with this document.



Presentation Layer

The Presentation Layer governs the interactions between the User and our application. It is also responsible for displaying data. It consists of our UI components which call relevant Controllers in the Logic Layer. The components of this layer include:

1. **MainUI**

MainUI facilitates account creation, profile creation and login processes by calling AuthController.

2. **AdminUI**

AdminUI facilitates Admins in performing their job tasks by calling AdminController.

3. **HawkerUI**

HawkerUI facilitates Hawkers in the maintenance of their stall pages, menus and stall reviews by calling HawkerController.

4. **ConsumerUI**

ConsumerUI facilitates Consumers in searching for stalls and leaving reviews by calling ConsumerController.

Application Layer

Involving the Controller classes, the Application Layer bridges the Presentation and Object Layers. It implements the use cases by calling entity classes in the Object Layer and retrieves processed results. The components of this layer include:

1. **AuthController**

AuthController is called by MainUI for User account creation and login operations. Its logic includes the login and sign up methods.

2. ConsumerController

ConsumerController is called by ConsumerUI for Consumer-specific actions, including: searching and favouriting stalls, modifying and reporting reviews. Its logic includes the methods to perform the aforementioned operations.

3. HawkerController

HawkerController is called by HawkerUI for Hawker-specific actions, including: modifying stalls and dishes. Its logic encompasses the methods to perform these operations.

4. AdminController

AdminController is called by AdminUI for Admin-specific actions, including: processing reported reviews and verifying Hawker profiles. Its logic includes the methods to perform the aforementioned operations.

Object Layer

The Object Layer houses the entity classes and their business logic, which are called by the Application layer. The components of this layer include:

1. User

User contains the name, email address, account password and photo of the registered User.

2. Consumer

Consumer contains the address, contact number, dietary preferences, preferred cuisines, ambulatory status and unique consumerID of each Consumer.

3. Review

Review contains the rating, content (as text) and unique reviewID of each submitted Review.

4. Stall

Stall contains the name, hawker centre, unit number, operating hours, hygiene rating, cuisine types, estimated wait time and unique stallID of each Stall.

5. Dish

Dish contains the dish name, price, photo and unique dishID of each dish.

6. Hawker

Hawker contains the SFA food license number and contact number of each Hawker.

7. Admin

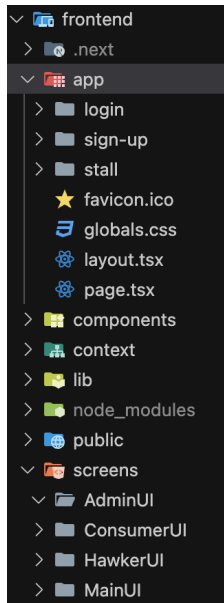
Admin contains the unique adminID of each Admin.

Persistence Layer

The Persistence Layer manages interactions with the databases that store entity data.

2.7. Application Skeleton

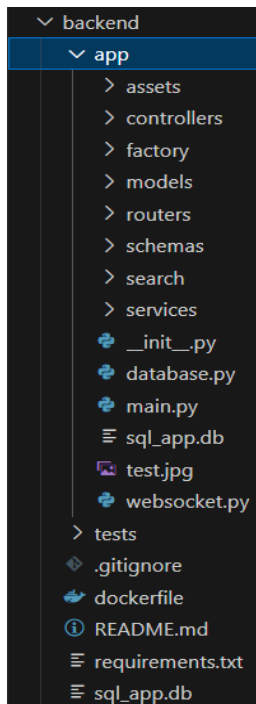
Please refer to the source code uploaded in the github repository for the application skeleton.



Frontend

The frontend is built with Next.js, TypeScript and Next.js, using TailwindCSS for styling. It is organized into role-based interfaces like MainUI, ConsumerUI, HawkerUI and AdminUI, each handling specific user tasks. The entry point is page.tsx with shared layout and styles defined globally.

Reusable components, utilities and static assets are structured into separate folders for clarity. This modular design improves maintainability, supports collaboration and also ensures consistent UI across the app.



Backend

The backend uses FastAPI with Python, structured into routers, controllers, services, models and schemas. It connects to a MySQL database using SQLAlchemy with a factory pattern for session handling.

The logic is modular and separated by feature domains. Validation is handled with Pydantic, and trie-based search improves performance. WebSocket support and test coverage are also included for real-time features and reliability

2.8. User Documentation

Hawkar will compile robust user documentation to enhance alignment and ensure ease of use for developers. The documentation will comprise the following:

2.8.1. README Files

2.8.1.1. Main Repository

A README file for the main repository that outlines the overall project. It will cover the system architecture, steps to set up the development environment, deployment procedures and a broad description of the repository's structure.

2.8.1.2. Frontend Repository

A separate README file will be provided for the frontend repository. It will explain how to set up the frontend, describe the project structure, and offer guidance on using React.js, TypeScript and TailwindCSS. Instructions for running and building the frontend will also be included

2.8.1.3. Backend Repository

The backend repository will also feature its own README, detailing how to set up the FastAPI server, configure the Python environment, manage dependencies and integrate the SQL database.

2.8.1.4. Code Comments

- Both frontend and backend code will be thoroughly commented to clarify complex logic, functions, components, classes, and methods. Tags such as TODO and FIXME will be used where appropriate.
- Inline comments will be used to explain the functionality and purpose of different code sections, modules, and libraries

2.8.1.5. Backend API Documentation

- The backend will leverage FastAPI's built-in interactive documentation, which is generated directly from the source code and stays up to date automatically. This will serve as a comprehensive reference and testing tool for the API.
- It will provide detailed information on API endpoints, including input parameters, authentication requirements, and response formats. This will be especially helpful for frontend developers and any third-party integrations.
- FastAPI provides built-in API documentation via Swagger UI, which is automatically generated based on the defined endpoints and type annotations.

2.9. Assumptions and Dependencies

2.9.1. Assumptions

It is assumed that all users of Hawkar will have access to the internet when using the application.

2.9.2. Dependencies

2.9.2.1. Frontend Libraries

Library	Purpose
Next.js	Used as the React framework for building a fast, scalable web application with support for server-side rendering and routing.
React.js	Provides the component-based architecture for building dynamic and interactive user interfaces.
Typescript	Adds static typing to JavaScript, improving code reliability, maintainability, and developer productivity.
TailwindCSS	Enables rapid and responsive UI development using utility-first CSS classes.
Shadcn	Provides accessible and customizable UI components built on top of Radix UI and Tailwind CSS for consistent design.
react-oauth/google	Handles Google OAuth integration for secure and streamlined user authentication.

2.9.2.2. Backend Libraries

Library	Purpose
FastAPI	To build a high-performance, asynchronous RESTful API with automatic documentation with Swagger UI.
uvicorn	Serves as the Asynchronous Server Gateway Interface (ASGI) to run the FastAPI application efficiently in production
SQLAlchemy	Provides an ORM for interacting with the relational database using Python Objects.
httpx	Enables asynchronous HTTP requests for internal or external API communication.

bcrypt	Handles secure password hashing and verification for user authentication.
python-multipart	Supports multipart/form-data parsing, essential for file uploads.
pydantic	Used for data validation, including email formats, via type-safe models.
minio	Connects the backend to an object storage service for handling media or file uploads.
google-auth	Manages verification and validation of Google OAuth tokens for authentication.
requests	Facilitates synchronous HTTP requests for non-async operations.

2.9.2.3. External Services

Hawkar depends on the following external services:

- Third-party APIs
 - MapBox API
 - Google OAuth 2.0 API
 - Hawker GeoJSON API Dataset (data.gov.sg)
- Cloud hosting services
 - Vercel
 - Render

2.9.2.4. Continuous Integration and Deployment (CI/CD)

To streamline the development workflow, a CI/CD pipeline has been configured. This setup ensures that whenever code changes are pushed to the main branch of the repository, the system will automatically:

1. Run built and test processes to verify the integrity of the application (Continuous Integration).
2. Upon successful completion, deploy the latest version of the application to the designated cloud hosting platform (Continuous Deployment)

This automated workflow enables faster release cycles, reduces the risk of human error and ensures the applications always up to date with the latest changes

3. Specific Requirements

3.1. Functional Requirements

1. Users must be authenticated to use our application.
 - 1.1. Users in general must be able to create an account in our application.
 - 1.1.1. Users must be able to create an account in our application using their emails (key), and their password. They can also optionally create one with their google accounts.
 - 1.1.2. System must be able to verify whether or not the email is already associated with an already existing account.
 - 1.2. Users must be able to create a unique profile assigned to their account, with the profile including the account role (Consumer, Hawker, Admin) alongside relevant information specific to the selected role. This is a separate operation that does not affect the existence of the account.
 - 1.2.1. Assuming the User already has an account, the User should be able to create a profile for his or her account, the profile will include the account role and all relevant information specific to the role.
 - 1.3. Hawker profile creation must require a contact number, personal address and the hawker's SFA license number for profile creation. Additionally, the profile will require admin verification before it is created and tied to the account.
 - 1.3.1. For admin verification, the system will send a verification request to Admins for verification.
 - 1.4. Admin profile creation must require a unique admin ID.
 - 1.4.1. The User Management System will only create the account if the key is recorded in the database.
 - 1.5. Users must be able to sign in using the accounts they have created previously.
 - 1.5.1. Users must be able to input their email address and password to login to the application or log in via Google.
 - 1.5.2. The application system must mask the user password by replacing actual characters with dots, unless the Users choose to unmask it.
 - 1.5.3. If the email address and password entered by the Users do not match, the application system shall display "Invalid email or password" to the Users.

- 1.5.4. If the email address and password entered by the Users match, the application shall log the User in and navigate the User to the dashboard of the application.
 - 1.5.5. System must also be able to verify the existence of an account given the input email.
- 2. Consumers must be able to view the list of hawker centers and stalls on the dashboard.
 - 2.1. Consumers must be able to view map views of hawker center locations.
 - 2.1.1. The application shall display a map where Consumers can see the locations of hawker centers.
 - 2.2. Consumers must be able to view a list of all hawker stalls registered with the application.
 - 2.3. Consumers must be able to search for hawker stalls.
 - 2.3.1. The application shall allow consumers to search by name of the stalls.
- 3. Consumers must be able to filter hawker centers and stalls based on various criteria.
 - 3.1. Consumers must be able to filter hawker centers and stalls by their operating hours.
 - 3.1.1. The application shall provide options for Consumers to specify a time range to find hawker centers and stalls that are open during that period
 - 3.2. Consumers must be able to filter hawker centers and stalls by location.
 - 3.3. Consumers must be able to filter stalls based on food preferences, price range, hygiene ratings and hawker centre.
 - 3.3.1. The application shall allow consumers to filter stalls based on food preferences (cuisines).
 - 3.3.1.1. The application shall display a picture, name, cuisine and price range for each hawker stall.
 - 3.3.2. The application shall allow consumers to filter stalls based on price range, using predefined categories of different prices.
 - 3.3.2.1. The application shall display a picture, name, cuisine and price range for each hawker stall.
 - 3.3.3. The application shall allow consumers to filter stalls based on hygiene ratings assigned by the National Environment Agency (NEA).
 - 3.3.3.1. The application shall display a picture, name, cuisine and price range for each hawker stall.
- 4. Consumers must be able to modify ratings and reviews for stalls.

- 4.1. Consumers must be able to submit ratings and reviews for stalls.
 - 4.1.1. Consumer must be able to rate stalls on a scale of 1 to 5 stars
 - 4.1.2. Consumers must be able to write text reviews for stalls.
 - 4.1.3. The application must be able to provide an option for Consumers to submit their reviews and ratings.
- 4.2. Consumers must be able to edit their ratings and reviews for stalls.
 - 4.2.1. Consumers must be able to modify both the rating (1-5 stars) and the text of their review.
 - 4.2.2. Consumers must be able to edit their review only within 48 hours of submission.
- 4.3. Consumers must be able to delete their ratings and reviews for stalls.
 - 4.3.1. The application shall allow Consumers to delete their reviews within 48 hours of submission.
 - 4.3.2. The application shall notify Consumers if their review has been successfully deleted.
5. The application must aggregate reviews for stalls to provide a consolidated rating.
 - 5.1. The application shall calculate the average rating of a stall based on Consumer review.
 - 5.2. The application shall display the average rating for each stall.
6. The application must display reviews and ratings in a way that helps Consumers make better decisions.
 - 6.1. The application shall allow Consumers to sort reviews by rating (high to low) and recency.
7. Consumers and Hawkers must be able to report reviews for stalls.
 - 7.1. The application shall provide a set of predefined categories for reporting, such as “Spam”, “Offensive” or “Irrelevant to food”.
 - 7.2. The application shall notify the Admin when a review is reported.
 - 7.3. The application shall show a confirmation message after a review is reported.
8. Consumers must be able to save and delete their favourite stalls for quick access.
 - 8.1. The application shall display the saved hawker stalls under the Consumer’s profile.
 - 8.2. The application shall allow consumers to remove hawker stalls from their favorites list.
9. The application shall be able to notify Consumers on matters of various purposes.

- 9.1. The system shall notify Consumers on new stalls added to nearby hawker centres.
- 9.2. The system shall notify Consumers on promotions or discounts at hawker centres.
- 9.3. The system shall notify Consumers on successful actions done within the application.
- 10. The application shall allow Hawkers to perform hawker-specific tasks.
 - 10.1. The hawker shall be able to modify the stall information associated with the hawker centre.
 - 10.1.1. The hawker must be able to add the stall information associated with the hawker centre.
 - 10.1.1.1. The name of the stall
 - 10.1.1.2. The associated hawker center of the stall
 - 10.1.1.3. The unit number of the stall
 - 10.1.1.4. The operating hours of the stall
 - 10.1.1.5. The hygiene rating of the stall
 - 10.1.1.6. The cuisine type of the stall
 - 10.1.1.7. The price range of the stall
 - 10.1.1.8. The photo of the stall
 - 10.1.2. The hawker must be able to edit the stall information associated with the hawker centre.
 - 10.1.2.1. The name of the stall
 - 10.1.2.2. The associated hawker center of the stall
 - 10.1.2.3. The unit number of the stall
 - 10.1.2.4. The operating hours of the stall
 - 10.1.2.5. The hygiene rating of the stall
 - 10.1.2.6. The cuisine type of the stall
 - 10.1.2.7. The price range of the stall
 - 10.1.2.8. The photo of the stall
 - 10.1.3. The hawker must be able to delete the stall information associated with the hawker centre.
 - 10.2. The hawker shall be able to view the reviews and ratings left by consumers for their stalls
 - 10.3. The hawker shall be able to modify dishes on the menu.

- 10.3.1. The hawker shall be able to add the following information for each dish:
 - 10.3.1.1. The name of the dish
 - 10.3.1.2. The price of the dish
 - 10.3.1.3. The photo of the dish
 - 10.3.1.4. The promotional status of the dish
 - 10.3.1.4.1. The discounted price of the dish
 - 10.3.1.4.2. The start date of the promotion
 - 10.3.1.4.3. The end date of the promotion
 - 10.3.2. The hawker shall be able to edit the following information for each dish:
 - 10.3.2.1. The name of the dish
 - 10.3.2.2. The price of the dish
 - 10.3.2.3. The photo of the dish
 - 10.3.2.4. The promotional status of the dish
 - 10.3.2.4.1. The discounted price of the dish
 - 10.3.2.4.2. The start date of the promotion
 - 10.3.2.4.3. The end date of the promotion
 - 10.3.3. The hawker shall be able to delete each dish.
- 11. The application shall allow Admins to perform admin-specific tasks.
 - 11.1. The admin shall be able to verify Hawkers' accounts.
 - 11.2. The admin shall be able to process reviews reported as irrelevant by Hawkers and Consumers.
 - 11.2.1. The admin shall be able to view the report details when a hawker reports a review, including the predefined reporting categories.

The admin shall be able to delete reported reviews that violate the application's policies or guidelines.

3.2. Non-functional Requirements

Usability	Different languages <ul style="list-style-type: none">- The web page must automatically adjust its language based on its user's locale, supporting English, Chinese and Malay
	Responsive User Interface <ul style="list-style-type: none">- The web page must provide a seamless user experience with minimal delays.- The web page must provide common interactions and navigation actions with a response time of less than 5 seconds for each action.
	Mobile Responsive <ul style="list-style-type: none">- Users must be able to see 100% of the contents of the web page clearly even from the mobile browsers
Reliability	Guaranteed Uptime <ul style="list-style-type: none">- The web application must maintain 99.9% uptime to ensure continuous availability for users.
	Data Integrity <ul style="list-style-type: none">- The web application must consistently store and retrieve user's data from the database without corruption and loss.
Performance	Response time <ul style="list-style-type: none">- The web application must load and respond to user interactions within 3 seconds for standard pages and 5 seconds for complex operations.
Supportability	The database solution that the application uses must be interchangeable with any commercial solutions that support both standard SQL or NoSQL queries

3.3. External Interface Requirements

3.3.1. User Interfaces

3.3.1.1. UI Mockups

Account Creation

Logo

Create an account

Already have an account? [Login](#)



Create account

Or register with

Sign up with Google

Profile Creation - Consumer

Logo



Click on the profile picture to upload your photo

Consumer



Get Started

Complete Your Profile to Get Started!

Just a few more details, and you'll be all set. Once completed, we'll take you to your home page!

Profile Creation - Hawker

Logo

Complete Your Profile to Get Started!

Just a few more details, and you'll be all set.
Once completed, we'll take you to your home page!



Click on the profile picture to upload your photo

Hawker

Business name

Business UEN

SFA licence num

Address

Contact Number

Hawker center

Cuisines type

Operating Hours

Get Started

Profile Creation - Admin

Logo

Complete Your Profile to Get Started!

Just a few more details, and you'll be all set.
Once completed, we'll take you to your home page!



Click on the profile picture to upload your photo

Admin

Admin unique key

Contact Number

Get Started

Login

Logo

Welcome Back!

Don't have an account? [Register](#)

Email Address

Enter your password

Log in

Or sign in with

Sign in with Google

Consumer - Dashboard

Hawkar



Hello, Consumer!

Search Hawker Stalls

Filters

Hawker Stalls



Big Bites 4.9 stars
Estimated Wait time: 10 mins



Big Bites 4.9 stars
Estimated Wait time: 10 mins

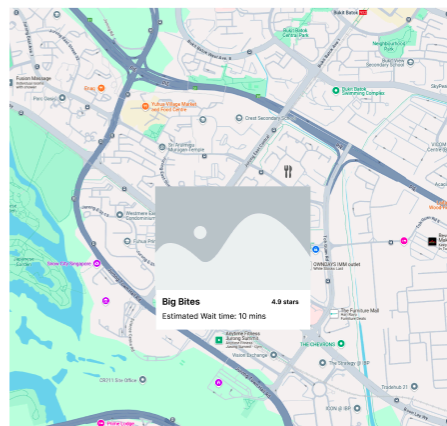


Big Bites 4.9 stars
Estimated Wait time: 10 mins



Big Bites 4.9 stars
Estimated Wait time: 10 mins

Sort By: Wait time



Filters Dialog - Consumer

Filters

×

Operating Hours:

Starting Hours

Closing Hours

Amenities

Washroom

Parking Space

Accessibility

Wheelchair Access

Railing Support

Tactile Paving

Food Preferences

Western

Indian Food

Korean Food

Price Range

LowHigh

\$3\$10

Location

Clementi

▼

Hygiene Ratings

A

▼

Apply Filters

Notifications Page - Consumer

Hawkar

Hello, Consumer!

Notifications

A new stall is added to Yuhua Food Centre

Today at 12.20 PM

Big Bites has added a new promotion to the stall

Today at 12.20 PM

Yuhua Food Centre is closed for cleaning

Today at 12.20 PM

Hawker Stall Information - Consumer

Hawkar
🌐
🔔
👤 Hello, Consumer!







Big Bites
 Western | Yuhua Food Centre
 Estimated Wait time: 10 mins
 Operating Hours: 10am to 4pm
 Hygiene Rating: A

Stall Menu


Food Name 1
Food type

Price


Food Name 1
Food type

Price


Food Name 1
Food type

Price

Ratings & Reviews

4.9 ★
480 Ratings

5 ★
4 ★
3 ★
2 ★
1 ★

Add Review

Sort by: Ratings


Consumer1
★★★★★

If you're on the hunt for a solid plate of Hainanese chicken rice, Ah Huat is a gem tucked away in Maxwell Food Centre. The poached chicken is tender and juicy, with just the right amount of gelatinous skin to give it that silky texture.


Consumer1
★★★★★

If you're on the hunt for a solid plate of Hainanese chicken rice, Ah Huat is a gem tucked away in Maxwell Food Centre. The poached chicken is tender and juicy, with just the right amount of gelatinous skin to give it that silky texture.


Consumer
★★★★★

If you're on the hunt for a solid plate of Hainanese chicken rice, Ah Huat is a gem tucked away in Maxwell Food Centre. The poached chicken is tender and juicy, with just the right amount of gelatinous skin to give it that silky texture.

Modify Review + Report Review

Add Review

Add Review

Ratings ★★★★★

Review

Submit Review

Edit Review

Edit Review

Ratings ★★★★★

This is the review

Submit Review

Report Review

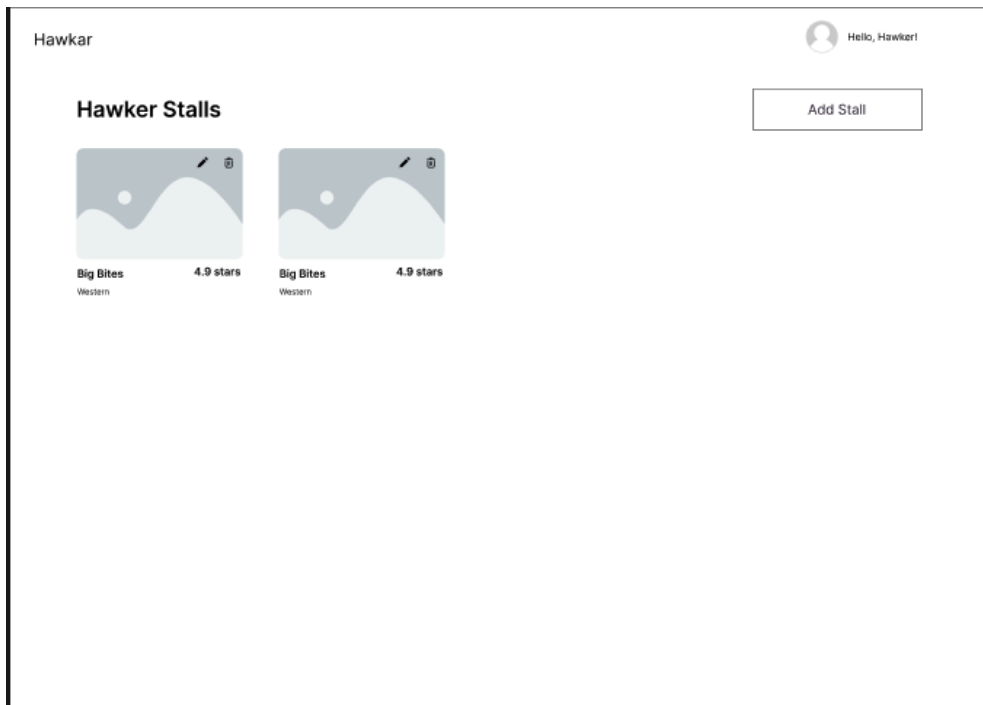
Report Review

Report
Inadequate to Food
offensive content

Why is it irrelevant?

Report Review

Hawker - Dashboard





Add Stall - Hawker

Logo

Add a Hawker Stall

Help expand our hawker directory by adding stall details.

Name
Hawker Center
Operating Hours
Hygiene Rating
Cuisine Type
<u>Upload Images</u>
 
Add Stall

Update Stall - Hawker

Logo

Edit Hawker Stall Details

Keep stall information accurate and up to date.

Name

Hawker Center

Operating Hours

Hygiene Rating

Cuisine Type

Upload Images

Update Stall

Stall Information View for Hawker

Hawker Stalls > Big Bites

Big Bites

Woolston | Yunus Food Centre

Estimated Wait time: 10 mins

Operating Hours: 10am to 4pm

Hygiene Rating: A

Stall Menu

Food Name 1

Price

Food Name 1

Price

Food Name 1

Price

Food type

Food type

Food type

Add Food

Ratings & Reviews

4.9 ★

480 Ratings

5 ★

4 ★

3 ★

2 ★

1 ★

Sort by: Ratings

Consumer1

★★★★★

If you're on the hunt for a solid plate of Hainanese chicken rice, Ah Huat is a gem tucked away in Maxwell Food Centre. The poached chicken is tender and juicy, with just the right amount of gelatinous skin to give it that silky texture.

Consumer1

★★★★★

If you're on the hunt for a solid plate of Hainanese chicken rice, Ah Huat is a gem tucked away in Maxwell Food Centre. The poached chicken is tender and juicy, with just the right amount of gelatinous skin to give it that silky texture.

Consumer

★★★★★

If you're on the hunt for a solid plate of Hainanese chicken rice, Ah Huat is a gem tucked away in Maxwell Food Centre. The poached chicken is tender and juicy, with just the right amount of gelatinous skin to give it that silky texture.

27

Add Dish - Hawker



Logo

Add a Dish

Help others discover great food by adding a dish to this stall.

☐ Promotion

[Upload Images](#)



Edit Dish - Hawker



Logo

Edit a Dish

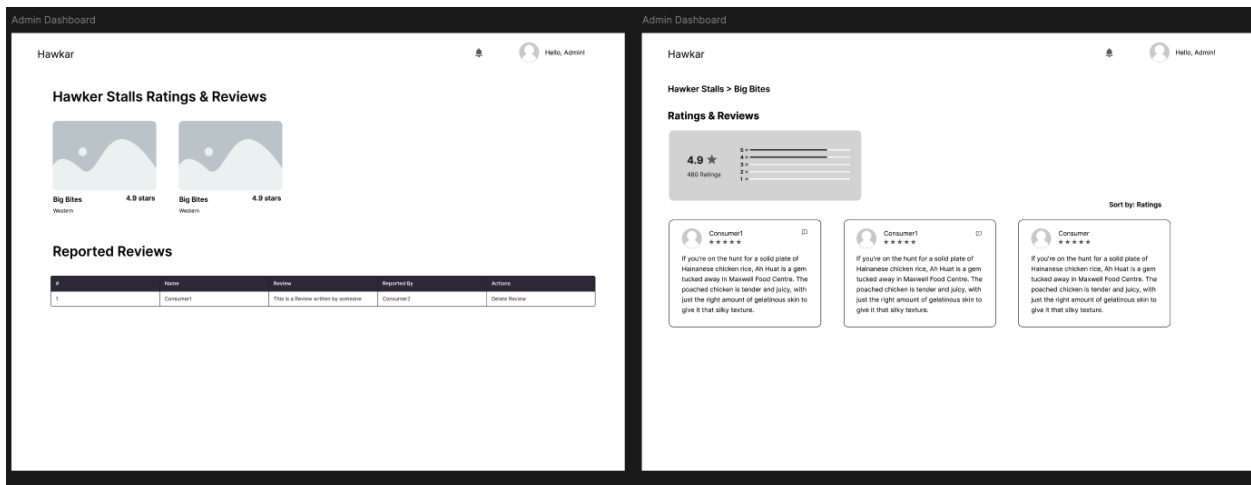
Keep dish information accurate and up to date.

☐ Promotion

[Upload Images](#)



Admin Dashboard



3.3.2. Hardware Interfaces

Hawkar primarily operates as a web-based application, thus it does not require any specialised hardware interfaces. It is designed to run on standard consumer devices, including smartphones, tablets, desktops and laptops with internet access.

Device types

The software is built to work across multiple device types, such as desktops, laptops, tablets, and mobile phones.

Data and Control Interactions

Users interact with the software through input/output devices such as touchscreens and standard peripheral devices like keyboards and mice, as well as network interface cards or built-in networks for internet connectivity.

3.3.3. Software Interfaces

Operating System

This application is developed to run on Android, iOS and any operating system (Windows, macOS, Linux).

Web Browsers

Latest versions of Chrome, Edge, Firefox, Safari or any modern web browsers

Databases

SQLite or PostgreSQL, accessed through backend services.

Frontend Tools

- Next.js
- React.js
- Typescript
- TailwindCSS
- Shadcn
- react-oauth/google

Backend Tools

- FastAPI
- uvicorn
- SQLAlchemy
- httpx
- bcrypt
- python-multipart
- pydantic
- minio
- google-auth
- Request

4. System Features

4.1. User(base class) Features

4.1.1. Create Account

4.1.1.1. *Description and Priority*

Description:

Allows new users to create an account using their email and password, and then proceed to set up their role-specific profile.

Priority:

- High

4.1.1.2. *Stimulus/Response Sequences*

1. User launches the app and selects Sign Up.
2. System prompts for:
 - a. Name
 - b. Email address
 - c. Password
3. System validates input and creates a new account.
4. User is redirected to the profile creation page.

4.1.1.3. *Functional Requirements*

- REQ-1: System must provide a Sign-Up interface
- REQ-2: System must validate:
 - Email format
 - Password strength
- REQ-3: On success, accounts must be created and stored securely.
- REQ-4: System must redirect user to role selection and profile creation.

4.1.2. Select Role

4.1.2.1. *Description and Priority*

Description:

Allows users to select a role (Consumer, Hawker, Admin) during account setup to determine the features they will have access to.

Priority:

- High

4.1.2.2. Stimulus/Response Sequences

1. After account creation, the user is prompted to choose a role.
2. System displays available roles with brief descriptions.
3. User selects a role and proceeds to the corresponding profile creation.

4.1.2.3. Functional Requirements

- REQ-1: System must display available roles with descriptions.
- REQ-2: System must allow user to select one role.
- REQ-3: System must redirect to appropriate role-specific profile form.

4.1.3. Login

4.1.3.1. Description and Priority

Description:

Allows existing users to log in to their account using their email and password.

Priority:

- High

4.1.3.2. Stimulus/Response Sequences

1. User opens the app and selects Log In.
2. System prompts for email and password.
3. System verifies credentials and logs the user in.
4. User is directed to their role-based dashboard.

4.1.3.3. Functional Requirements

- REQ-1: System must provide a login screen with email and password fields.
- REQ-2: System must authenticate credentials against stored records.
- REQ-3: On success, User must be directed to their respective dashboard.
- REQ-4: System must show an error message on failure.

4.1.4. Google Login

4.1.4.1. Description and Priority

Description:

Allows Consumers to sign up and log in using their Google account, streamlining access without needing to create a separate password.

Priority:

- High

4.1.4.2. *Stimulus/Response Sequences*

1. Consumer selects Continue with Google on the login or signup screen.
2. System redirects to Google authentication flow.
3. Upon successful authentication, Consumer is signed in and redirected to the dashboard or profile setup.

4.1.4.3. *Functional Requirements*

- REQ-1: System must display Continue with Google as an option on login and signup pages.
- REQ-2: System must use Google OAuth for authentication.
- Description: Allows users to securely log out of the application.

4.1.5. **Logout**

4.1.5.1. *Description and Priority*

Description:

Allows users to securely log out of the application.

Priority:

- High

4.1.5.2. *Stimulus/Response Sequences*

1. User clicks the Logout button in the app menu.
2. System ends the session and returns to the login screen.

4.1.5.3. *Functional Requirements*

- REQ-1: System must provide a logout button in the navigation menu.
- REQ-2: Logout must terminate session and redirect to login screen.

4.2. **Consumer Features**

4.2.1. **Create Consumer Profile**

4.2.1.1. *Description and Priority*

Description:

Allows a User to assign the Consumer role to their newly created account. This involves uploading a profile picture and filling in Consumer-specific preferences.

Priority:

- High

4.2.1.2. *Stimulus/Response Sequences*

1. User signs up and is directed to the profile creation page.
2. System prompts User to upload a profile photo.
3. User selects the Consumer role from the role selection screen.
4. System prompts User to fill in:
 5. Address
 6. Contact number
 7. Dietary preference
 8. Preferred cuisine
 9. Ambulatory status
10. User clicks "Get Started".
11. System saves the Consumer profile and grants access to the Consumer dashboard.

4.2.1.3. *Functional Requirements*

- REQ-1: System must prompt the user to upload a profile photo.
- REQ-2: System must allow role selection (Consumer, Hawker, Admin).
- REQ-3: If Consumer is selected, system must prompt for:
 - Address
 - Valid contact number (starts with 8 or 9, and 8 digits long)
 - Dietary preference
 - Preferred cuisine
 - Ambulatory status
- REQ-4: System must validate inputs and display errors for invalid or missing fields.
- REQ-5: Upon successful submission, the Consumer profile must be saved and assigned.

4.2.2. **View Hawker Centres On Map**

4.2.2.1. *Description and Priority*

Description:

Allows Consumers to view the locations of hawker centres as pins on an interactive map.

Priority:

- Normal

4.2.2.2. *Stimulus/Response Sequences*

1. Consumer logs in and is directed to the Dashboard.
2. Map automatically displays pins representing hawker centres.

4.2.2.3. *Functional Requirements*

- REQ-1: System must display a map on the Dashboard.
- REQ-2: System must display hawker centre pins on the map.
- REQ-3: Pins must correspond to real hawker centre locations.

4.2.3. **View List of Hawker Stalls**

4.2.3.1. *Description and Priority*

Description:

Allows Consumers to view a list of all hawker stalls registered with Hawkar.

Priority:

- Normal

4.2.3.2. *Stimulus/Response Sequences*

1. Consumer logs in and lands on the Dashboard.
2. System displays a list of all registered hawker stalls beside the interactive map.

4.2.3.3. *Functional Requirements*

- REQ-1: System must display a scrollable list of hawker stalls.
- REQ-2: List must include stall name, cuisine type, hygiene rating, and price range.

4.2.4. **Search Hawker Stalls**

4.2.4.1. *Description and Priority*

Description:

Allows Consumers to search for hawker stalls by name using the search bar.

Priority:

- Normal

4.2.4.2. *Stimulus/Response Sequences*

1. Consumer types into the search bar.
2. System displays stalls that match the search input.
3. If no results are found, an empty message is shown.

4.2.4.3. *Functional Requirements*

- REQ-1: System must allow keyword search by stall name.
- REQ-2: Search must support partial and case-insensitive matches.
- REQ-3: Search results must update in real time.

4.2.5. **Filter Hawker Stalls**

4.2.5.1. *Description and Priority*

Description:

Allows Consumers to filter hawker stalls based on operating hours, location, price range, and hygiene rating.

Priority:

- Normal

4.2.5.2. *Stimulus/Response Sequences*

1. Consumer clicks the "Filters" button.
2. Consumer selects filters (e.g., operating hours, location, price, hygiene).
3. Consumer clicks "Apply Filters".
4. System displays a filtered list of hawker stalls.

4.2.5.3. *Functional Requirements*

- REQ-1: System must provide filter options for:
 - Operating hours
 - Location
 - Price range
 - Hygiene rating
- REQ-2: System must update stall list based on selected filters.
- REQ-3: If no matches are found, show the appropriate message.

4.2.6. **Modify Reviews (Add, Edit, Delete)**

4.2.6.1. *Description and Priority*

Description:

Allows Consumers to add, edit, or delete reviews they have submitted on hawker stalls.

Priority:

- Normal

4.2.6.2. *Stimulus/Response Sequences*

- Add: Consumer clicks "Add Review", fills rating and comment, then submits.
- Edit: Consumer clicks pen icon beside existing review, modifies content, then updates.
- Delete: Consumer clicks bin icon beside existing review, confirms deletion.

4.2.6.3. *Functional Requirements*

- REQ-1: Consumers must be able to add a review with rating and comment.
- REQ-2: Consumers must be able to edit their own reviews.
- REQ-3: Consumers must be able to delete their own reviews.
- REQ-4: System must validate ownership and provide confirmation before deletion.

4.2.7. **Sort and View Reviews**

4.2.7.1. *Description and Priority*

Description:

Allows Consumers to sort stall reviews by recency or rating.

Priority:

- Normal

4.2.7.2. *Stimulus/Response Sequences*

1. Consumer selects sorting option from dropdown.
2. System reorders reviews according to selection.

4.2.7.3. *Functional Requirements*

- REQ-1: System must support sorting by:
 - Most Recent
 - Highest Rating
 - Lowest Rating
- REQ-2: System must re-render reviews based on selected sort order.

4.2.8. **Report Review**

4.2.8.1. *Description and Priority*

Description:

Allows Consumers to report reviews that are spam, offensive, or irrelevant.

Priority:

- Normal

4.2.8.2. Stimulus/Response Sequences

1. Consumer clicks the flag icon on a review.
2. Consumer selects a report reason and enters an optional comment.
3. Consumer clicks "Submit Report".
4. System forwards the report to Admins for review.

4.2.8.3. Functional Requirements

- REQ-1: System must allow flagging of individual reviews.
- REQ-2: System must provide predefined report reasons.
- REQ-3: System must forward report to Admin dashboard.
- REQ-4: System must confirm report submission to Consumer.

4.2.9. Save and Delete Favourite Stalls

4.2.9.1. Description and Priority

Description:

Allows Consumers to save hawker stalls to their favorites for easy access, and remove them when no longer relevant.

Priority:

- Normal

4.2.9.2. Stimulus/Response Sequences

- Save: Consumer taps heart icon on a stall to save it.
- Delete: Consumer taps red heart icon to remove from favorites.
- System updates the profile's Saved Stalls list accordingly.

4.2.9.3. Functional Requirements

- REQ-1: System must allow saving stalls via heart icon.
- REQ-2: System must allow unsaving from the same interface.
- REQ-3: Saved stalls must be accessible via profile section.
- REQ-4: System must reflect changes immediately.

4.3. Notify Consumer

4.3.1. Description and Priority

Description:

Allows System to notify Consumers about new stalls, promotions, and confirmation messages.

Priority:

- High

4.3.2. Stimulus/Response Sequences

1. Consumer logs in and taps the bell icon.
2. System displays notifications in chronological order.
3. If no notifications exist, the system displays: "Great, you're all caught up!"

4.3.3. Functional Requirements

- REQ-1: System must display a bell icon for notifications.
- REQ-2: Notifications must be stored and shown in a mailbox interface.
- REQ-3: System must support push-style updates for promotions, confirmations, and activity logs.
- REQ-4: If no notifications, the system must display a friendly empty state.

4.4. Hawker Features

4.4.1. Create Hawker Profile

4.4.1.1. Description and Priority

Description:

Allows a User to register as a Hawker by submitting business-related details such as their SFA license, stall name, and contact information. This profile creation is required to manage stalls and dishes within the system.

Priority:

- High

4.4.1.2. Stimulus/Response Sequences

1. User signs up and is directed to the profile creation page.
2. System prompts the User to upload a profile photo.
3. User selects the Hawker role from the role selection screen.
4. System prompts User to fill in:
 - a. Stall name
 - b. Address

- c. SFA License number
 - d. Contact number
 - e. Email address
5. User clicks "Get Started".
 6. System saves the Hawker profile and routes it for admin verification.

4.4.1.3. Functional Requirements

- REQ-1: System must prompt for a profile photo.
- REQ-2: System must require SFA license number, valid contact, and business details.
- REQ-3: System must store Hawker profile in a pending verification state.
- REQ-4: System must notify Admin for verification.

4.4.2. Modify Dishes

4.4.2.1. Description and Priority

Description:

Allows verified Hawkers to add, edit, or delete dishes under their stall profile. Each dish includes a name, description, price, and image.

Priority:

- High

4.4.2.2. Stimulus/Response Sequences

1. Hawker logs into the dashboard.
2. Hawker selects their stall and accesses the "Dishes" section.
3. Hawker adds a new dish or edits/deletes an existing one.
4. System saves updates in real-time and reflects changes immediately.

4.4.2.3. Functional Requirements

- REQ-1: System must allow dish creation with:
 - Name
 - Price
 - Image
 - Optional description
- REQ-2: System must allow editing and deleting dishes.
- REQ-3: Changes must be reflected in the stall profile immediately.

4.4.3. Modify Stalls

4.4.3.1. Description and Priority

Description:

Allows Hawkers to update their stall details including name, cuisine type, pricing, hygiene rating, and description.

Priority:

- High

4.4.3.2. Stimulus/Response Sequences

1. Hawker logs into the dashboard and clicks on their stall card.
2. Hawker edits stall fields and saves the form.
3. System updates the stall information and confirms success.

4.4.3.3. Functional Requirements

- REQ-1: System must allow updating of:
 - Stall name
 - Description
 - Cuisine type
 - Price range
 - Hygiene rating
 - Image
- REQ-2: System must validate fields and apply updates in real time.

4.4.4. Modify Promotions

4.4.4.1. Description and Priority

Description:

Allows Hawkers to create and manage promotional offers to increase visibility and attract Consumers.

Priority:

- Normal

4.4.4.2. Stimulus/Response Sequences

1. Hawker navigates to the Promotions tab.
2. Hawker fills in the promotion name, description, and time window.
3. Hawker clicks "Submit" and the system updates the stall profile.

4.4.4.3. Functional Requirements

- REQ-1: System must allow creation of promotions with:
 - Title
 - Description
 - Start and end time/date
- REQ-2: System must allow editing and deleting active promotions.
- REQ-3: Promotions must appear on the Consumer-facing stall page.

4.4.5. Report Review

4.4.5.1. Description and Priority

Description:

Allows Hawkers to report reviews that are inappropriate, spam, or harmful to their business.

Priority:

- Normal

4.4.5.2. Stimulus/Response Sequences

1. Hawker navigates to the reviews section under their stall.
2. Hawker clicks the flag icon beside the offending review.
3. System opens a report modal where reason and comments can be provided.
4. Hawker submits the report, and the Admin is notified.

4.4.5.3. Functional Requirements

- REQ-1: System must allow Hawkers to flag reviews from their stall page.
- REQ-2: System must provide a reason selector and optional comment field.
- REQ-3: Report must be forwarded to Admin for further action.
- REQ-4: System must confirm submission to Hawker.

4.4.6. Notify Hawker

4.4.6.1. Description and Priority

Description:

System sends Hawkers real-time notifications regarding account approval, stall activity, and Consumer interactions.

Priority:

- High

4.4.6.2. *Stimulus/Response Sequences*

1. Hawker logs in and clicks the notification bell.
2. System displays any unread notifications.
3. Clicking a notification redirects the Hawker to the relevant content.

4.4.6.3. *Functional Requirements*

- REQ-1: System must store notifications in a dedicated Hawker inbox.
- REQ-2: System must support real-time updates for events such as:
 - Admin approval
 - Dish or stall updates
 - Consumer reviews and reports
- REQ-3: Notifications must link directly to related content.

4.5. Admin Features

4.5.1. Create Admin Profile

Description:

Allows a User to register as an Admin by providing their contact information and uploading a profile photo. Admins have elevated privileges to manage the system.

Priority:

- High

4.5.1.1. *Description and Priority*

1. User signs up and is directed to the profile creation page.
2. System prompts the User to upload a profile photo.
3. User selects the Admin role.
4. System prompts for:
 - a. Admin key for verification
 - b. Address
 - c. Contact Number
5. Upon verification, the system creates an Admin profile.

4.5.1.2. *Stimulus/Response Sequences*

1. User signs up and is directed to the profile creation page.
2. System prompts the User if they want to upload a profile photo.
3. User selects the Admin role.
4. System prompts for:
 - a. Contact number
 - b. Admin key for verification
5. Upon verification, the system creates an Admin profile.

4.5.1.3. *Functional Requirements*

- REQ-1: System must prompt for a profile photo.
- REQ-2: System must require contact number and admin key.
- REQ-3: System must validate the admin key before assigning the role.
- REQ-4: System must create and store the Admin profile.

4.5.2. **Verify Hawkers**

4.5.2.1. *Description and Priority*

Description:

Allows Admins to view pending Hawker accounts and approve or reject them based on submitted documentation.

Priority:

- High

4.5.2.2. *Stimulus/Response Sequences*

1. Admin logs in and navigates to the Hawker Approval screen.
2. System displays a list of pending Hawker profiles.
3. Admin reviews each profile and clicks Approve or Reject.
4. System updates the status and notifies the Hawker.

4.5.2.3. *Functional Requirements*

- REQ-1: System must display all pending Hawker applications.
- REQ-2: Admin must be able to view submitted SFA license and contact info.
- REQ-3: Admin must be able to approve or reject each application.
- REQ-4: System must update profile status and notify the Hawker.

4.5.3. **Manage Reported Reviews**

4.5.3.1. *Description and Priority*

Description:

Allows Admins to review, investigate, and take action on reviews reported by Consumers or Hawkers.

Priority:

- High

4.5.3.2. *Stimulus/Response Sequences*

1. Admin navigates to the Reported Reviews dashboard.
2. System displays a list of reported reviews with details and reason.
3. Admin can choose to Keep or Delete the review.
4. System updates the review state accordingly.

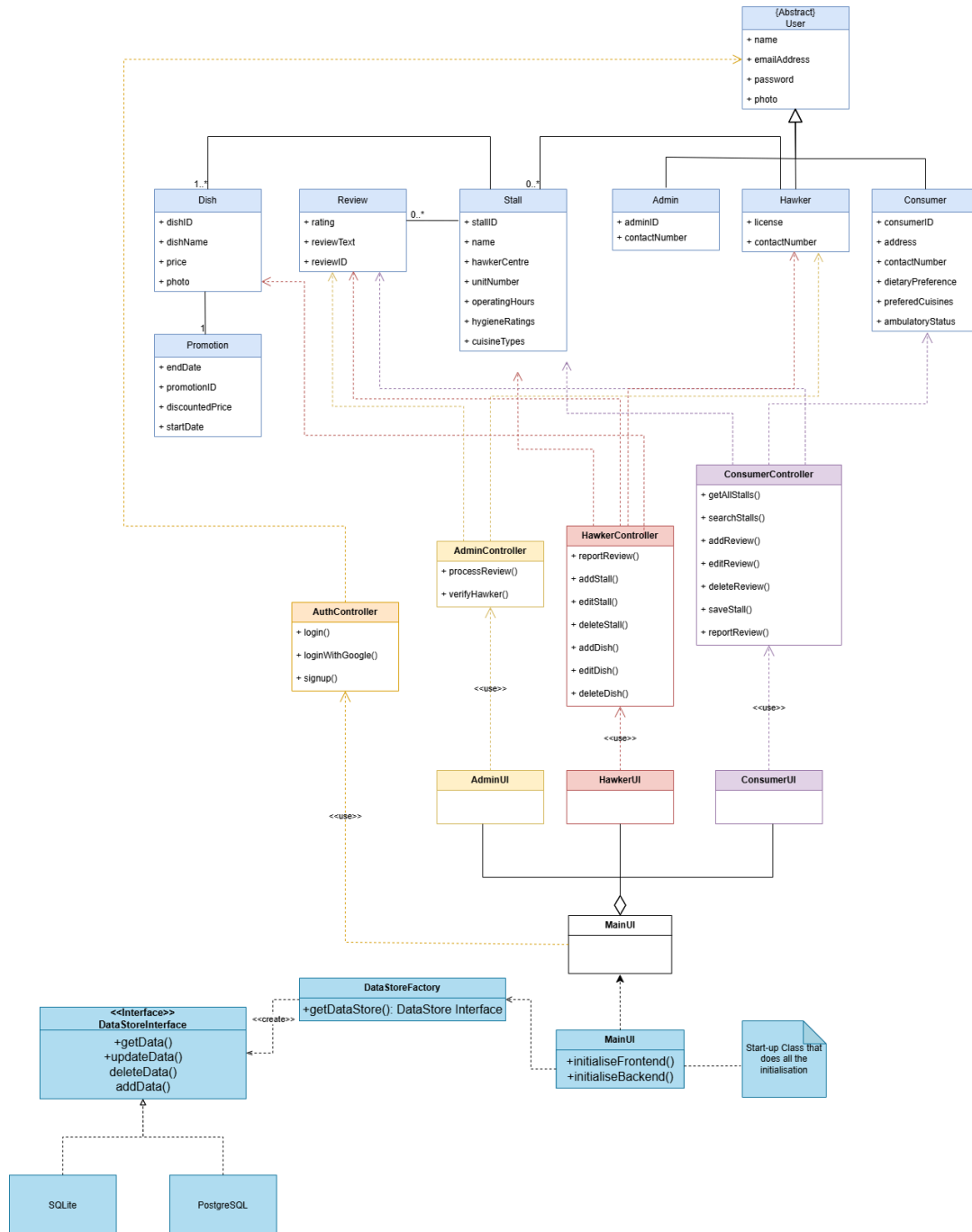
4.5.3.3. *Functional Requirements*

- REQ-1: System must list all reported reviews and associated metadata.
- REQ-2: Admin must be able to see who reported it and why.
- REQ-3: Admin must be able to keep or delete the review.
- REQ-4: System must update and reflect changes immediately.

5. System Architecture and Design

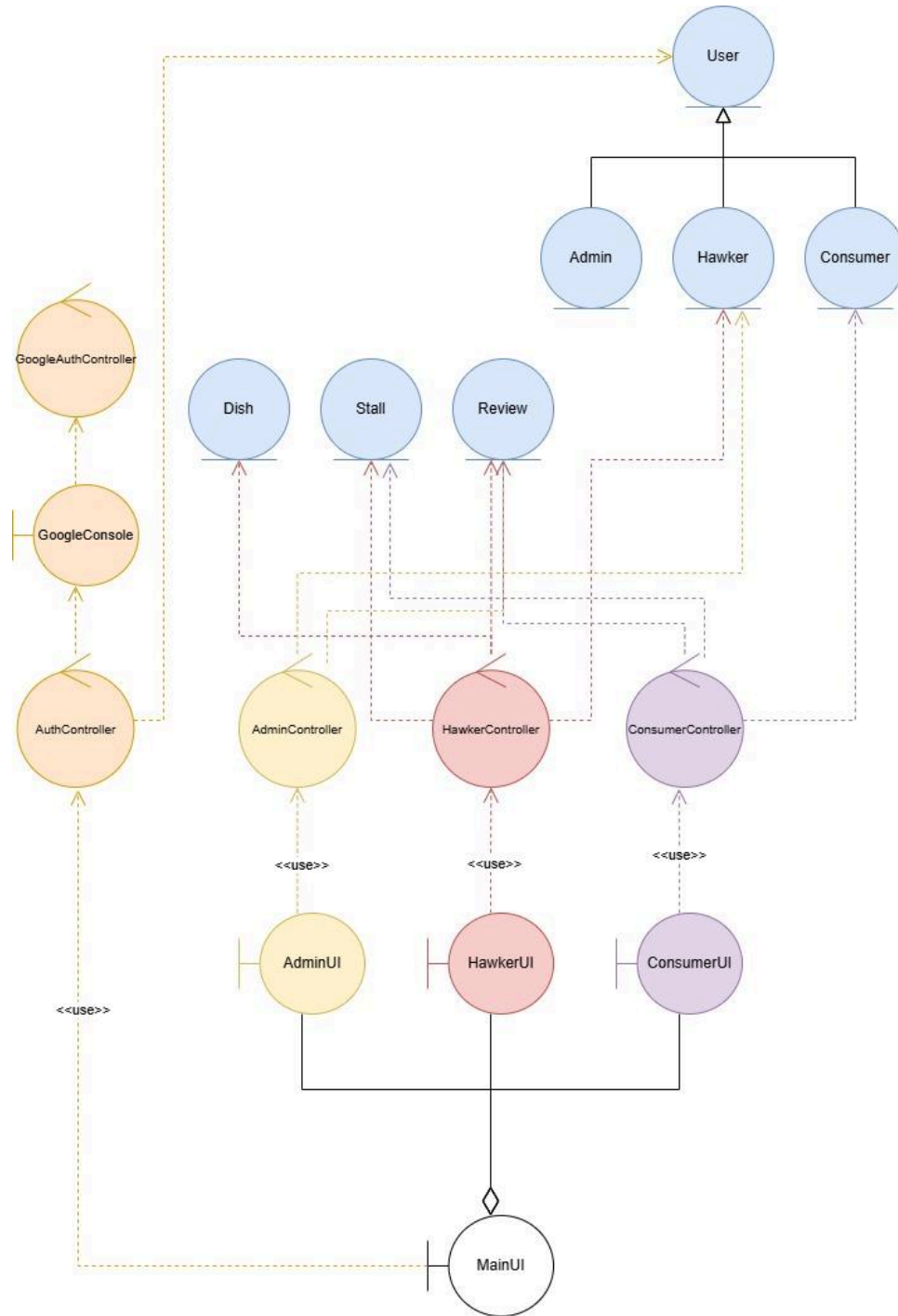
5.1. Class Diagram

If the image is unclear, please refer to the accompanying raw PNG file uploaded with this document.



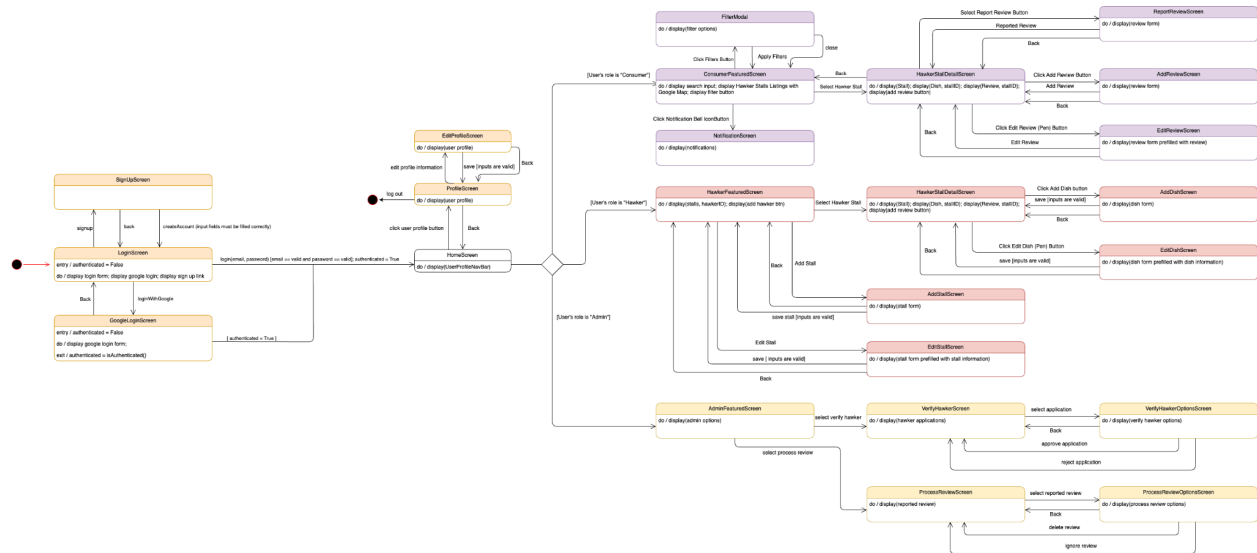
5.2. Stereotype Class Diagram

If the image is unclear, please refer to the accompanying raw PNG file uploaded with this document.



5.3. Dialog Map

If the image is unclear, please refer to the accompanying raw PNG file uploaded with this document.



6. Appendices

Appendix A: Glossary

Abbreviation	Meaning
API	A set of protocols and tools that enable different software applications to communicate with each other.
SQL	SQL stands for Structured Query Language. It is a standardised, domain-specific programming language used to manage and modify data stored in relational databases.
Front-end	Front-end encompasses all elements that users see and interact with on a website. It includes the design, layout, text, images, buttons and the overall user interface (UI).
Back-end	Back-end refers to the server side of a web application and involves manipulating data processing, server logic and integration with other systems.
SRS	Software Requirement Specifications is a document detailing the expected functionality and performance of software, including user interactions and system behavior.