Software Requirements Specification

for

ShopPeas

Version 1.2 approved

Prepared by

Eng Yi Xuan
Koh Huei Shan Winnie
Lee Jedidiah
Rachel Tan Min Zhi
Saffron Salmah Yen Lim
Yan Jun Chao

Nanyang Technological University, Team Peas Limited

10/11/2024

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Yi Xuan, Winnie, Jedidiah, Rachel, Saffron, Jun Chao	24/08/2024	Documented the purpose, target user, functional requirements, non-functional requirements, and UI interfaces.	1.0
Yi Xuan, Winnie, Jedidiah, Rachel, Saffron, Jun Chao	15/09/2024	Updated the functional requirements, UI interfaces, and data dictionary.	1.1
Yi Xuan, Winnie, Jedidiah, Rachel, Saffron, Jun Chao	10/11/2024	Final updates to the documentation based on development work.	1.2

1. Introduction

1.1 Purpose

This document provides the Software Requirements Specifications (SRS) for Version 1.0 of the ShopPeas mobile application. It serves as a comprehensive guide for all stakeholders, detailing the application's description, interface requirements, system features, and other relevant system considerations. The SRS ensures a shared understanding of the project's objectives and functionalities across all parties involved.

1.2 Document Conventions

The document follows the following conventions for clarity and consistency. Priorities for higher-level requirements are assumed to be inherited by detailed requirements.

Font: Times New Roman
Line Spacing: 1.5
Heading 1: Font Size: 18, Font Weight: bold
Heading 2: Font Size: 14, Font Weight: bold
Heading 3: Font Size: 12
Content: Font Size: 12

1.3 Intended Audience and Reading Suggestions

This document is intended for all stakeholders of the application, including three primary user groups: consumers, wholesalers, and application system administrators, as well as, application developers, project managers, testers, and the marketing staff.

Consumers: Our target users who want to buy healthy food products from wholesalers. Information relevant to consumer functions and features is outlined in the following sections:

- 1. Introduction

- 2.6 User Documentation
- 3.1 User Interfaces
- 4. System Features.

Wholesalers: Wholesalers from Singapore and Malaysia, specifically Johor Bahru, who want to sell their wholesale products on a platform. Information relevant to wholesaler functions and features is outlined in the following sections:

- 1. Introduction
- 2.6 User Documentation
- 3.1 User Interfaces
- 4. System Features

System Administrator: Manages the consumer and wholesaler accounts to ensure the smooth operation of the application. Information relevant to system administration functions and features is outlined in the following sections:

- 1. Introduction
- 2.6 User Documentation
- 3.1 User Interfaces
- 4. System Features

App Developers: To obtain a comprehensive understanding of the application, it would be beneficial to review the entire document. However, developers can focus on development-focused sections in the following sequence

- 3. External Interface Requirements
- 4. System Features
- 5. Non-Functional Requirements
- Appendix A. Data Dictionary

Project Managers: Overseeing the ShopPeas Project, project managers need a broad understanding of the application. However, they can focus on the user-focused sections in the following sequence

- 2. Overall Description

- 3. External Interface Requirements
- 4. System Features

Testers: Test the application to ensure that it fulfils the user requirements. Testers can focus on the following sections in the following sequence

- 4. System Features
- 5. Non-Functional Requirements

Marketing Staff: Responsible for marketing the application to our target users. We recommend focusing on the following sections in the sequence

- 1. Introduction
- 2. Overall Description
- 3.1 User Interfaces

1.4 Product Scope

ShopPeas is an e-commerce application built in line with the Singapore Government's Healthier SG and SmartNation initiatives. It aims to provide consumers with easy accessibility to a diverse range of healthy and cheap wholesale food products while increasing wholesaler outreach. This dual approach aligns with Peas Ltd's mission to enhance market efficiency while contributing to the community's well-being.

1.5 References

React Native Documentation: https://reactnative.dev/docs/environment-setup
Spring Boot Documentation: https://docs.spring.io/spring-boot/index.html
Firebase Documentation: https://firebase.google.com/docs
Currency Rate API: https://currencyapi.com/docs
OneMap API: https://www.onemap.gov.sg/apidocs/

2. Overall Description

2.1 Product Perspective

ShopPeas is a new application developed by Peas Ltd. Currently, wholesalers operate through their established supply chains or exclusive third-party channels such as WhatsApp or Telegram.

Through ShopPeas, we directly connect consumers in Singapore with local and regional wholesalers. By eliminating the need for intermediaries, who typically contribute to increasing product costs, ShopPeas reduces the overall expenses borne by consumers. With a focus on healthy food products, the app will also offer a comprehensive and user-friendly platform that allows consumers to browse, compare, and purchase products from various wholesalers. The platform will enable consumers to make informed and cost-efficient purchasing decisions by providing transparent pricing and detailed product information.

Other than the consumers, the platform will also serve wholesalers by expanding their reach to a broader customer base and offering valuable sales analytics to help optimise their operations. Furthermore, ShopPeas minimises the need for wholesalers to set up and maintain their own websites, allowing wholesalers to run their businesses more efficiently and effectively.

2.2 Product Functions

The ShopPeas platform is designed to serve two primary users: consumers and wholesalers. The following section summarises the key functionalities that the platform will provide to each of these user groups.

For Consumers:

1. Account Management
    - Consumers can create a new account to access the platform.
    - Consumers can log in with their existing credentials.
    - Consumers can update their personal details when required.
2. Product Browsing

- Consumers can search for products and see a listing of healthier wholesale items.
- Consumers can select the specific products which will redirect them to the respective product page.
- Consumers can search for and sort wholesalers according to their needs.

3. Purchasing and Cart Management
    - There should be a shopping cart, payment page and history page for consumers.
    - Consumers can add products to their shopping cart as well as update their cart, which includes adding views, updating, and deleting items.
    - Consumers should be able to add items directly from the respective product pages, where they will be allowed to select the quantity..
    - Consumers can check out their carts and select a payment method available (Credit/Debit Cards).
4. Orders
    - Consumers can review the history of their past purchases.
    - Consumers can leave ratings for purchased products.

For Wholesalers:

1. Account and Store Management:
    - Wholesalers can update store details when required.
2. Product Management
    - Wholesalers can show their current product listings.
    - Wholesalers can create, update and delete product listings, including prices and quantity.
3. Transaction Management
    - Wholesalers can track their pending for acceptance, to be completed, and completed transactions.

2.3 User Classes and Characteristics

Consumer:

Frequency of Use:

- Regular to frequent (daily to weekly)

Product Functions Used

- Update account details
- Search products
- View products
- Like products
- View wholesalers
- Follow wholesalers
- Searching for specific items
- View cart
- Updating items in the cart
- Making purchases

Characteristics

- Technical Expertise: Low to moderate
- Security/Privilege Level: Basic user privileges
- Educational Level: Varied
- Experience: Ranges from first-time online shoppers to experienced e-commerce users

Importance

- High-priority user class; primary target for ShopPeas platform

Wholesaler:

Frequency of Use

- Moderate to high (weekly to daily)

Product Functions to be used

- List products for sale
- Update product details
- Remove products
- Process Bulk orders
- Manage wholesaler account

Characteristics

- Technical Expertise: Moderate to high

- Security/Privilege Level: Elevated privileges for product management
- Educational Level: Generally higher, with business knowledge
- Experience: Typically experienced in B2B or B2C sales

Importance

- Medium to high priority; crucial for product supply and variety

2.4 Operating Environment

2.4.1 Production Environment of *ShopPeas*

This subsection describes the setting of which the mobile application is put into operation.

Setting	**Description**
Mobile OS Support	The mobile application requires iOS 16.4 or above, with potential expansion to Android platforms.
React Native Support	The application is built using React Native (version 0.74.5) and requires compatible mobile devices to run the compiled native code.
Screen Support	The application is optimized for modern mobile devices with the following specifications: ● Minimum supported resolution: 390x844 pixels ● Supports different pixel densities (current development on 3x density) ● Adapts to system font scaling
Connectivity requirements	The application requires an active internet connection to communicate with the backend services.

2.4.2 Development Environment of *ShopPeas*

This subsection describes the setting of which the mobile application is built and tested on during the development phase.

Setting	**Description**
Front-end development using React Native	React Native is an open-source mobile application framework maintained by Meta. The development team uses React Native to build all user interfaces of the mobile application as it features cross-platform compatibility and reusable components which drastically speed-up the development process. Edition: React Native (version 0.74.5)
Development using Expo	Expo is a framework and platform for React Native that helps you develop, build, deploy, and quickly iterate on mobile apps. The development team uses Expo to streamline the development process and access additional features. Edition: Expo SDK 51.0.28
Back-end development using Spring Boot	Spring Boot is an open-source, Java-based framework maintained by Spring that follows the MVC architectural pattern. The development team uses Spring Boot to build the web server, as well as REST APIs of the application. Edition: Spring Boot (version 3.3.3)
Database using Firebase	Firebase is a Backend-as-a-Service (BaaS) platform maintained by Google. The development team deploys the application's database using Firebase Admin SDK for real-time data synchronization and

	scalable cloud storage. Editions: ● Frontend: Firebase (version 10.13.1) ● Backend: Firebase Admin SDK (version 9.3.0)
HTTP Client using Axios	Axios is used for making HTTP requests between the front-end and back-end services. Edition: ● Axios (version 1.7.7) ● React Native Axios (version 0.17.1)
Development tools	The application development environment includes: ● Expo CLI for React Native development ● Node.js package manager (npm/yarn) ● Java Development Kit (JDK) 22 ● Environment configuration using dotenv 16.4.5

2.5 Design and Implementation Constraints

This section covers all constraints which have limited, or will be limiting, options available to the ShopPeas development team, both during development stage and post-production maintenance stage.

2.5.1 Frontend Constraints

2.5.1.1 Framework (Frontend)

The mobile application must be developed using React Native and Expo, which imposes the following constraints:

- Limited to React Native's built-in components and styling

- Limited to features supported by the Expo SDK
- Must follow Expo's build and deployment process
- Limited to React Native's cross-platform components and APIs

2.5.1.2 Code Structure

- Must use Javascript only. Typescript is not supported in this version.
- Must follow React Native's unidirectional data.

2.5.1.3 Server Dependency

- The mobile application requires a connection to the backend server to function
- Core features are not available in offline mode
- User authentication and data persistence rely on server availability
- Real-time features require constant server connection

2.5.2 Backend Constraints

2.5.2.1 Framework (Backend)

The app utilises Spring Boot 3.3.3 for server-side operations and imposes the following constraints:

- Requires Java 22 or higher runtime environment
- Limited to Spring Boot's dependency versions for compatibility

2.5.2.2 Security

- Access to specific features and data must be restricted based on user roles and permissions
- Role-based access control (RBAC) must be enforced at the API level

2.5.2.3 API Dependecies

- OneMap API key must be regenerated every 3 days

2.5.3 Database Constraints

The app utilises Firbase as our database and is constrained to:

- Limited to Firebase's NoSQL data structure
- Must follow Firebase's security rules structure
- Must handle Firebase's eventual consistency model

2.5.4 Design Principles and Patterns

The code is to follow the following design patterns to enhance code maintainability and reusability:

- Controller-Service-Repository (CSR): for separation of concerns. Controllers will handle HTTP requests from the front-end, services manage business logic, and repositories will handle database logic. This simplifies testing and manages complexity when scaling our app.
- Model-View-Controller (MVC): the code shall be organised such that the models and controllers in the backend and views in the frontend.
- Data Transfer Object (DTO) pattern: for encapsulation.

The system will also incorporate SOLID design principles such as:

- Single Responsibility Principle (SRP): each functionality has its own class with a well-defined responsibility.
- Open-Closed Principle (OCP): to allow for addition of new functionality without modifying existing code.
- Interface Segregation Principle (ISP): ensures that clients only use the interfaces they require.
- Dependency Inversion Principle (DIP): ensures that high-level modules will not depend on low-level modules through abstraction to build a loosely coupled application.

2.6 User Documentation

We have created a README.md file in our GitHub Repository to guide users on how to set up the project. The following are the main steps.

2.6.1 Front-End Set Up

1. Install an emulator on your computer. If you are using a MacOS, install the simulator with Xcode. If you are using Windows, install the emulators with Android Studio.

2. Create an `.env` file in the `frontend` folder and fill in the respective fields. The REACT_APP_BACKEND_* environment variables indicate the respective API path to use and will call the respective controllers.

   REACT_APP_API_KEY=<your-firestore-apikey>
   REACT_APP_BACKEND_API=http://localhost:8080/
   REACT_APP_BACKEND_AUTH=auth
   REACT_APP_BACKEND_CONSUMER=consumer
   REACT_APP_BACKEND_WHOLESALER=wholesaler
   REACT_APP_BACKEND_PRODUCT=products
   REACT_APP_BACKEND_WHOLESALER_PRODUCT=wholesaler_product
   REACT_APP_BACKEND_CART=cart
   REACT_APP_BACKEND_HISTORY=history
   REACT_APP_BACKEND_TRANSACTION=transaction
   REACT_APP_BACKEND_PAYMENT=payment
   REACT_APP_ONEMAP_API_KEY=<your-onemap-apikey>

3. Install required packages for the app to run.

   ```
   npm install
   ```

4. Run the front-end.

   ```
   npx expo start
   ```

2.6.2 Back-End Set Up

1. Download Java SE 22.

2. Open the `backend` folder in your IDE. We used IntelliJ IDEA.

3. In the `./backend/src/main/resources/config` folder, add these 3 JSON files that will contain the resources to communicate with external services. The first is the Currency API JSON file. Create an account with Currency API and get the API key.

   ```
   # currencyapi.json file
   {
       "apiKey": "<your-currency-api-key>"
   }
   ```

4. Create a OneMap account and get the API key.

   ```
   # onemapapi.json file
   {
       "apiKey": "<your-onemap-api-key>"
   }
   ```

5. You can generate the `firebase-service-account.json` file by selecting Service accounts for Firestore Database in your Firebase console, and generating a new private key.

   ```
   # shoppeas-firebase-service-account.json
   {
     "type": "service_account",
     "project_id": "<your-project-id>",
     "private_key_id": "<your-private-key-id>",
     "private_key": "<your-private-key>",
     "client_email": "<your-client-email>",
   ```

"client_id": "<your-client-id>",
"auth_uri": "<your-auth-uri>",
"token_uri": "<your-token-uri>",
"auth_provider_x509_cert_url": "<your-auth-provider-cert-url>",
"client_x509_cert_url": "<your-client-cert-url>",
"universe_domain": "googleapis.com"
}

6. Run the `ShoppeasApplication.java` file. The back-end should be running on http://localhost:8080/.

2.7 Assumptions and Dependencies

The ShopPeas development team developed the ShopPeas mobile application with the following Assumptions:

1. Technical assumptions
    - The user is operating the application on an IOS device
    - Users have constant strong Internet connection, as most data are delivered by the server
    - The return information of the APIs will be consistent.
    - Device has GPS capabilities enabled for location features

2. Infrastructure assumptions
    - Full-scale development will receive funding for full Firebase subscription
    - OneMap API and Currency API services will remain available and maintain current API structure
    - Spring Boot server will be hosted on a reliable cloud platform

3. User Assumptions

- Users understand English as the primary language
- Users have access to valid payment methods
- Users are willing to share their address to calculate distance and travel time to wholesaler lcoation

4. Business assumptions
    - Merchants will provide accurate product information
    - Product prices and availability will be updated regularly
    - Payment gateway services will remain operational

2.7.1 Front-End Dependencies

The dependencies can be found in the `frontend/package.json` file.

Dependency	**Version**	**Purpose**
react-native	0.74.5	Framework for building native mobile applications using JavaScript.
react-native-paper	5.12.5	Material Design component library for UI elements.
react-native-alert-notification	0.4.0	Custom alert and notification system implementation.
@react-navigation/native	6.1.18	Core navigation infrastructure for screen management.
@react-navigation/bottom-tabs	6.6.1	Implementation of bottom tab navigation pattern.
@react-navigation/stack	6.4.1	Stack navigation with gesture support and animations.
react-native-maps	1.14.0	Integration of mapping services for location

		features.
@react-native-async-storage/async-storage	1.24.0	Local data persistence and caching functionality.
fuse.js	7.0.0	Lightweight fuzzy-search functionality.
expo	51.0.28	Framework that simplifies React Native development and deployment process.
expo-network	6.0.1	Network state and connectivity management.
@expo/vector-icons	14.0.3	Comprehensive icon set for the application interface.
zustand	4.5.5	State management solution for handling application data flow.
axios	1.7.7	Handle HTTP requests and responses between frontend and backend.
firebase	10.13.1	Client SDK for Firebase services integration.
dotenv	16.4.5	Environment variable management for development configuration.

2.7.2 Back-End Dependencies

The dependencies can be found in the `backend/pom.xml` file and are managed by Maven.

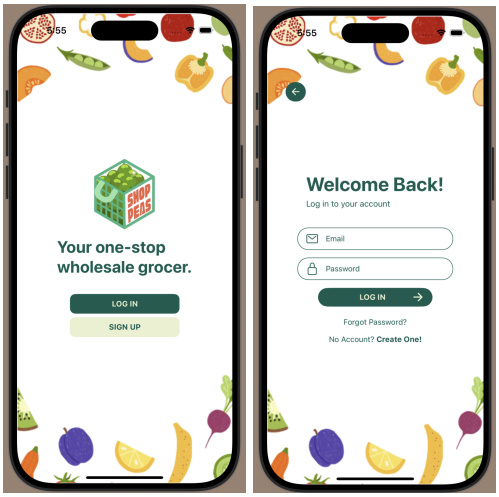
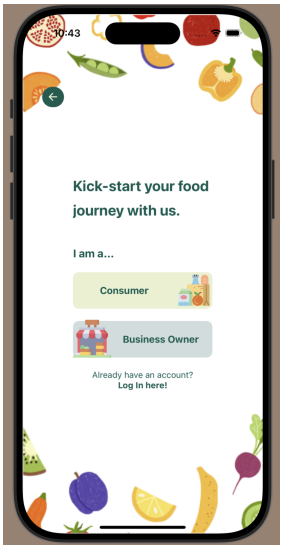
Dependency	**Version**	**Purpose**
spring-boot-starter	latest	Provides the basic functionalities for back-end development.

spring-boot-starter-web	latest	To build RESTful applications with Spring MVC.
spring-boot-starter-security	latest	Allows for role-based authorisation of APIs.
spring-boot-starter-log4j2	3.3.4	Logging of errors to files for traceability and app maintainability. The configuration can be found in the backend/src/main/resoucrces/log4j2.xml file and the output folder for the logs is backend/logs.
springdoc-openapi-starter-webmvc-ui	2.6.0	Automate the generation of API documentation with SwaggerUI.
jakarta.servlet-api	6.1.0	Handle HTTP requests and responses on the server side. We used it specifically for Firebase.
lombok	1.18.34	Minimise boilerplate code by automatically generating the getters, setters, and constructors for the application's models.
firebase-admin	9.3.0	To connect to our Firestore Database for querying, creation, modification, and deletion of our records.
junit-jupiter-api	4.13.2	To create and execute test cases.

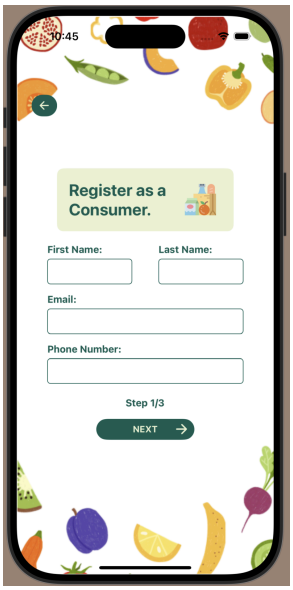
3. External Interface Requirements

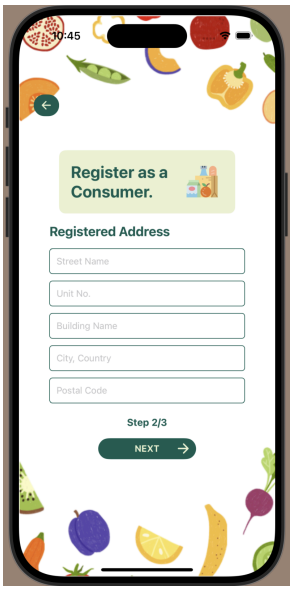
3.1 User Interfaces

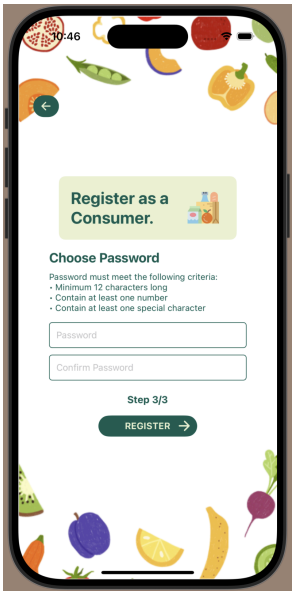
3.1.1 Guest Interfaces

	**Login / Sign-up Option** • Users will have the option of logging in with an existing account, or signing up with a new account **Login Page:** • Users will key in their email and password for their existing account • Both consumers and wholesalers will log into their accounts from this page
	**Sign-up Page:** • If users click on 'Sign Up' in the options, they will be led to this page where they can choose between being a Consumer or being a Business Owner. • A choice of logging in with an existing account is provided below, where it would direct users to the login page upon clicking on it

3.1.2 Consumer Interfaces







Consumer Registration

Shows up when consumer registers for an account with ShopPeas

First Image (Detail & Password Collection) :

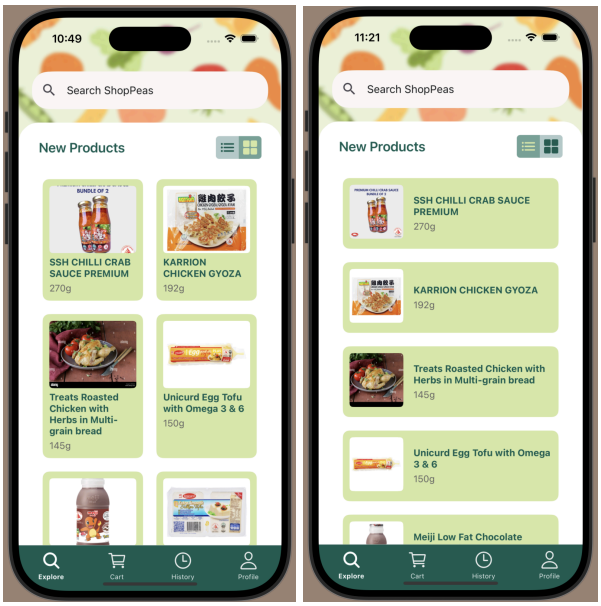
- Consumer will key in their First Name, Last Name, Email and Phone Number for contact purposes

Second Image (Address Collection) :

- Consumer will key in their address for calculation of how far the seller is away from their house

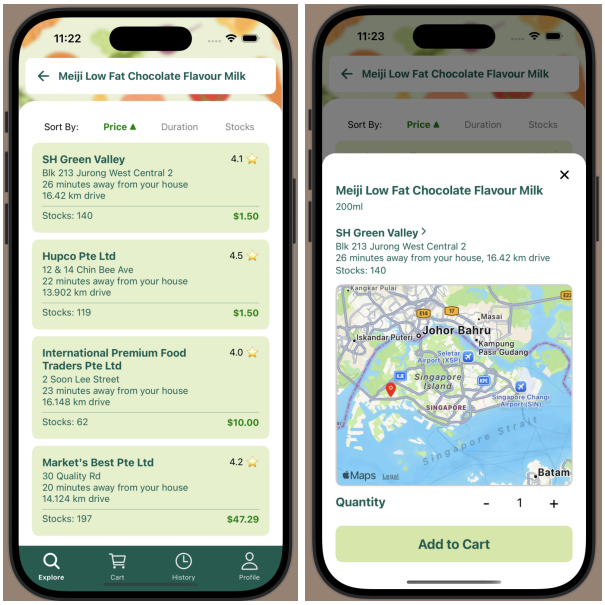
Third Image:

- Consumer will choose a password and confirm it with the following requirements
    - Min. 12 Characters
    - Contain at least 1 number
    - Contain at least 1 special character



Consumer Home Page (Explore)

- Home page shows Newest Products with different display formats (grid and list)
- User able to search for the items that they are looking for
- Bottom bar includes explore, cart, (order) history and profile
- Specific icon will light up in white according to whichever page user is on



Product Details

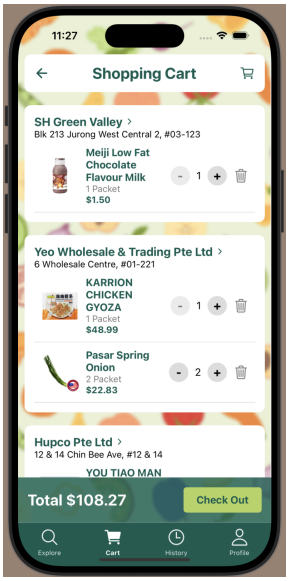
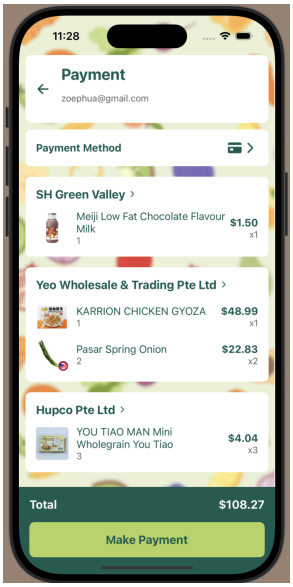
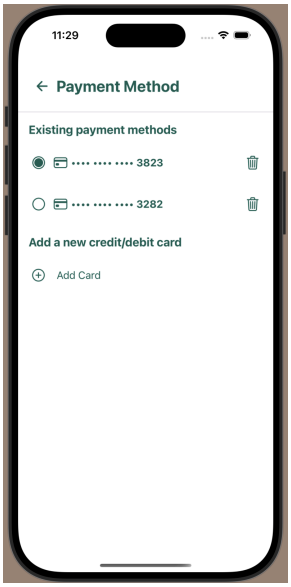
These pages will be shown after user clicks into a specific product, in this case, Meiji Milk

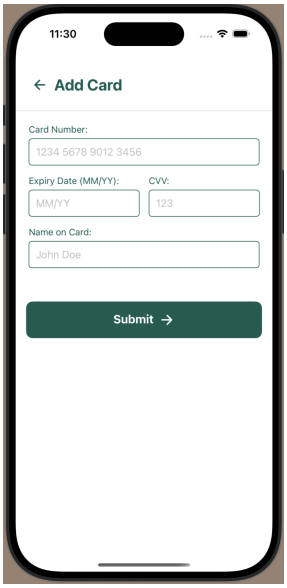
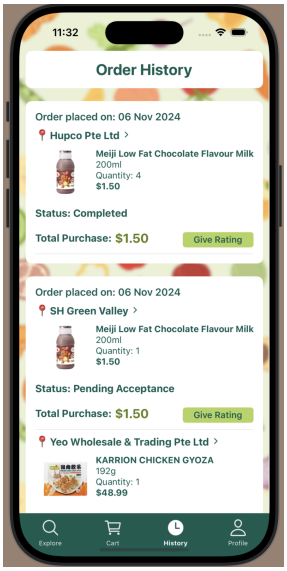
Left Image:

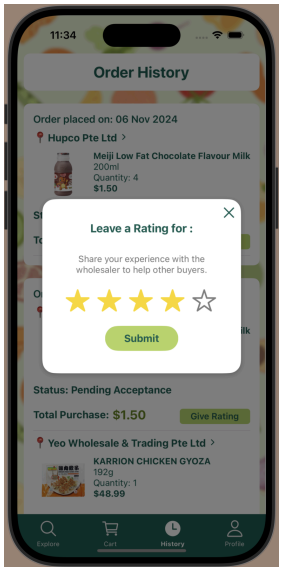
- Shows the different wholesalers that are selling this product, their location, duration away from user (distance), rating, stocks left and price
- Wholesalers can be arranged by price, duration away from user and stocks

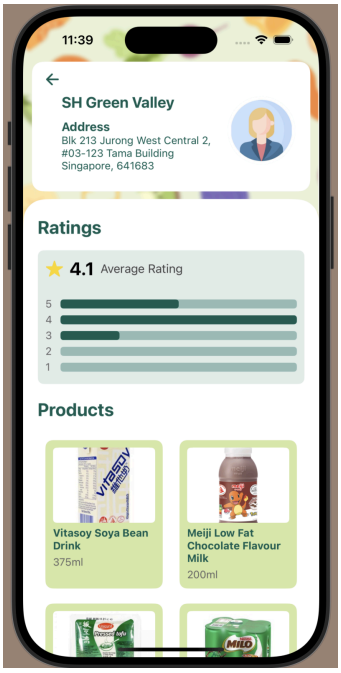
Right Image:

- When a specific wholesaler is clicked, there will be a change in colour, and a pop up will appear, showing where exactly the wholesaler is
- Users can adjust the quantity of product

	they are keen on getting and add to cart
	**Shopping Cart** • Shows the items that the user has added to cart, under each different wholesaler • User can adjust the quantity of product in this page • They can also remove items by clicking on the bin icon for that item • If they wish to checkout, they can do so from this page
 	**Payment Page** First Image: • Includes user email, payment methods, and the products they want to get, again under each wholesaler respectively • Making payment will allow for order placement Second Image: • This page appears when the user clicks on ‘payment method’ on the first page. • User can either change to an existing payment method, or add a new card • Users will click on the option they want and click ‘add’. • User can delete a payment method by clicking on the bin beside the respective

<table>
<tr>
<td> 
</td>
<td>payment method
Third Image:
• This page will appear if the user clicks on ‘Add Card’
• They will key in their details here, and upon clicking ‘submit’, their details will be saved</td>
</tr>
<tr>
<td> 


</td>
<td>Order History
• Shows the user order history
• Each box shows the details for each order, of which includes
○ Date order was placed
○ Wholesalers and the products they bought from each of them
○ Order status: ‘Pending Acceptance’, ‘Pending Completion’ or ‘Completed’.
• Consumers can rate the wholesalers through the ‘Give Rating’ button
Give Rating Popup
• A Pop-up will appear that will allow the user to leave a rating, of which they can click the number of stars they would like to leave for the wholesaler</td>
</tr>
</table>

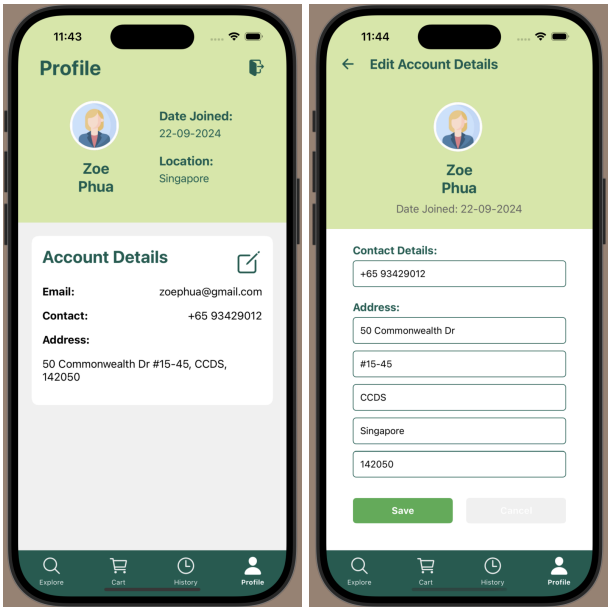


Wholesaler Page (Consumer POV)

Can be accessed by :

- Clicking on wholesaler name in wholesaler product details page (Pop up that appears after clicking on a product → On a wholesaler the consumer wants to buy from).
- Clicking on wholesaler name in cart, payment, and order history will also navigate the user to this page.

Shows Wholesaler Address, Ratings and Products

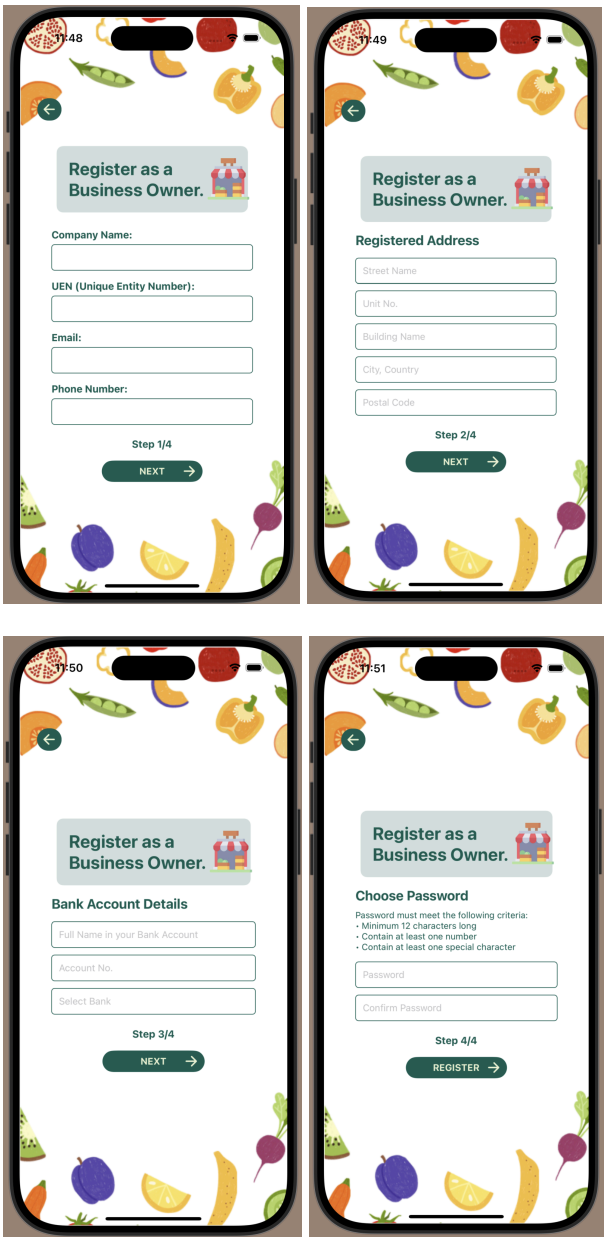


Consumer Profile

Consumers can see their account details on this page. (Left Image)

They can also click the edit icon to edit their details, of which the ‘Edit Account Details’ Page (Right Image) will appear. Consumers can change their Contact Details, Location and click ‘Save’ to save details, or ‘Cancel’ to cancel all changes and return to their profile page.

3.1.3 Wholesaler Interfaces



Business Registration

Shows up when a business registers for an account with ShopPeas

First Image (Detail Collection) :

- Business will key in their Company Name, UEN (Unique Entity Number) Email and Phone Number for contact and verification purposes

Second Image (Detail Collection Pt. 2):

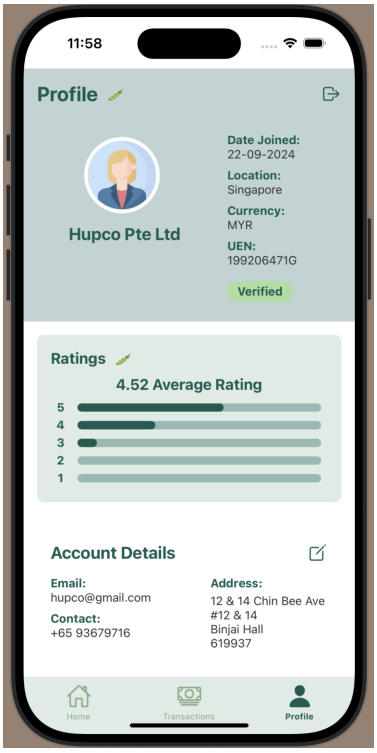
- Business wil key in the Business's Registered Address for verification purposes

Third Image (Detail Collection Pt. 3):

- Business will key in their Bank Account Details for payment purposes

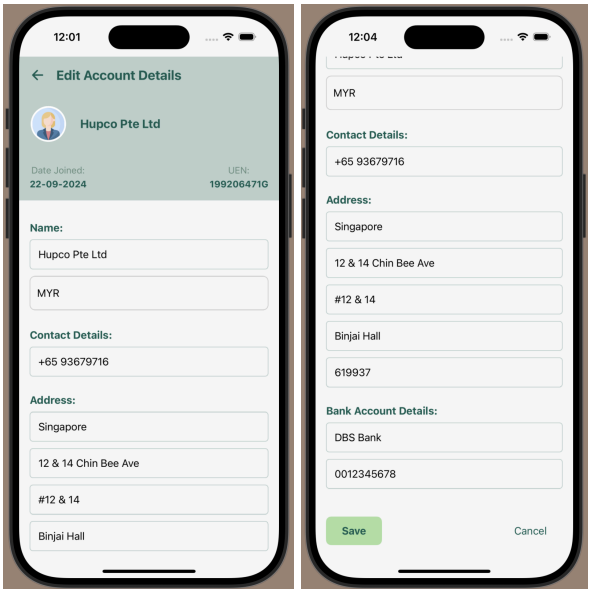
Fourth Image (Detail Collection Pt. 4):

- Business will choose a password and confirm it with the following requirements
    - Min. 12 Characters
    - Contain at least 1 number
    - Contain at least 1 special character



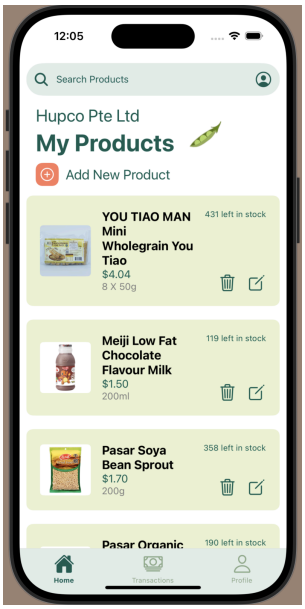
Business Profile Page

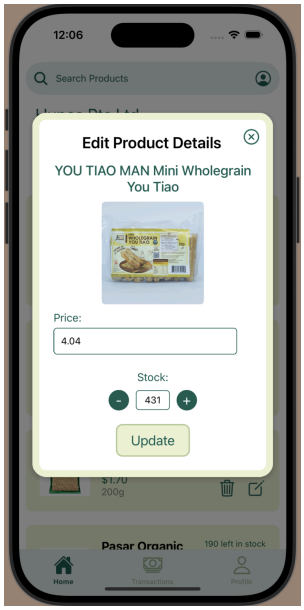
- Businesses can view their Profile, including details like the date joined, location, chosen currency, and account details (Contact details, UEN, Address)
- Businesses can also view their verification status
- From this page, Businesses can navigate to:
    - Edit account (Edit account icon)
    - Log out (Exit icon)

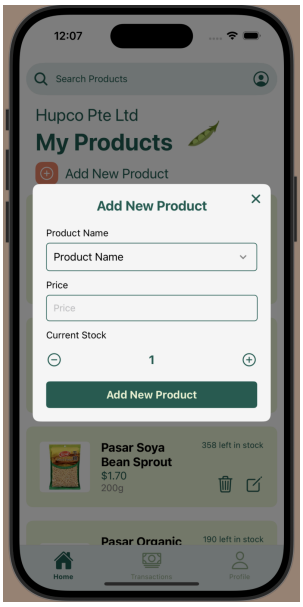


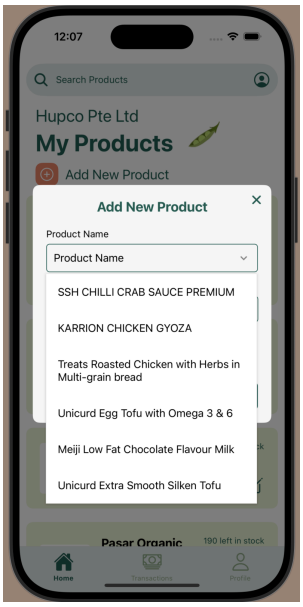
Edit Account Details

- Business Users can edit Account Details such as:
    - Location
    - Currency
    - Contact Details (Email and Phone Number)
    - Address
- Details such as UEN and business name cannot be changed after registration
- Business Users can update their details via the “Save” button, or cancel the changes via the “Cancel” button (bottom of page) or the back arrow button (top left hand corner)









My Products Page

First Page (List of Products):

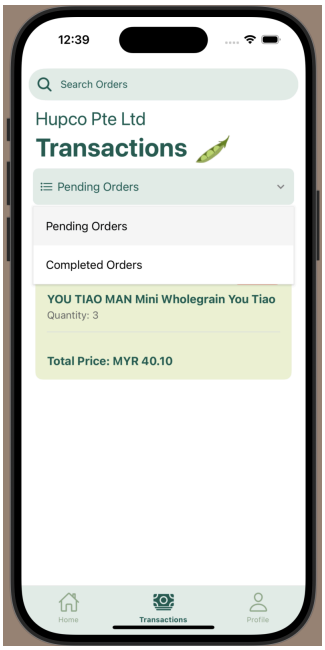
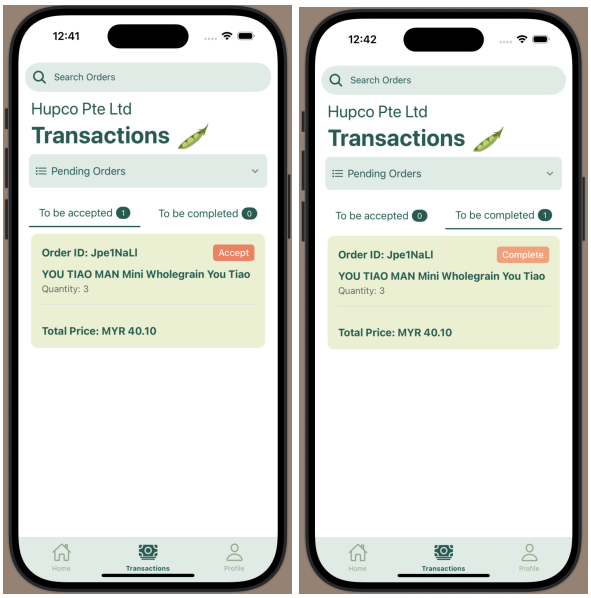
- Business Users can view all their current Products in a list format and search for specific Products via the search bar
- Each Product can be edited, deleted, or added
- New Products can be added via the "Add New Product" button (top)

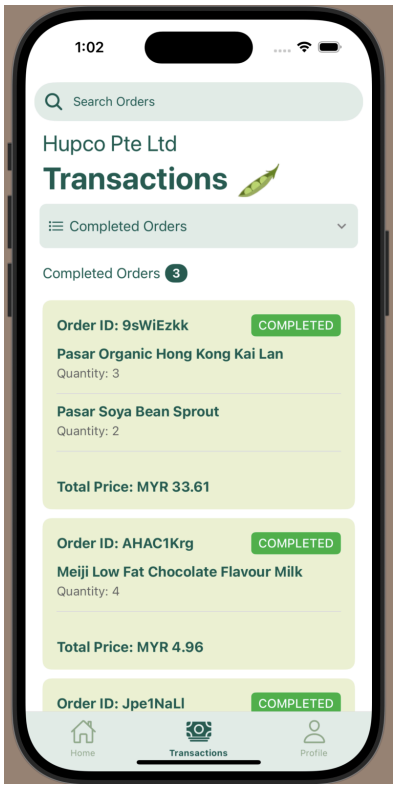
Second Page (Edit Product Details):

- The Product to be edited will be highlighted
- Product details price and current stock can be edited
- Changes can be saved via the "Update" button or cancelled via the cross button

Third and Fourth Page (Add Product):

- New Products can be added via the "Add New Product" button (top)
- Business Users will choose the product from the dropdown, of which lists all the healthy products, price and current stock.
- Changes can be saved via the "Add New Product" button or cancelled via the cross button

<table>
<tr>
<td> 
</td>
<td>Transactions Page
• The Transactions Page will display different types of Transactions:
○ Pending Orders
○ Completed Orders
• The Business User can view the different types of Transactions via the dropdown menu</td>
</tr>
<tr>
<td> 
</td>
<td>Transactions Page (Pending Orders [To be accepted / To be completed])
• Pending Orders are categorised into “To be accepted” (Left Image) and “To be completed” (Right Image). Business Users can toggle between these two categories. The current category that the Business User is viewing is highlighted in dark text
• The number of orders that have not yet been confirmed (accepted) and not yet completed will be displayed next to their respective categories
• Each entry will display the Order ID (OID), Product Names, Quantity, Total Price in MYR (converted based on user preference in profile)</td>
</tr>
</table>

	• Under the 'To be accepted' tab, wholesalers can click on the 'Accept' button to accept an order, and the order will go under the 'To be completed' tab • Under the 'To be completed' tab, wholesalers can click on the 'Complete' button to complete an order, and the order will go under the 'Completed Orders' section
	**Transactions Page (Completed Orders)** • The Business Owner can view all the Orders that have been completed under the completed orders section • The number of Orders will be displayed beside 'Completed Orders' • Entries are similar to those under the 'Pending Orders' section, except the buttons have now changed to a 'Completed' status

3.2 Hardware Interfaces

3.2.1 Client-Side Device Compatibility

ShopPeas is designed primarily to be a mobile application that is supported on iOS version 16.4 and above. Currently, ShopPeas is not compatible with laptop, tablet, or Android devices, nor does it run on web browsers. Internet connectivity is required for the operation of Shoppeas. iOS devices should meet the compatibility requirements outlined in Section 2.4.1 for a more optimal user experience.

All interactions between the user and the application are carried out via standard I/O such as display screens for visual output, keyboards for text input, and touchscreens for navigation and general inputs.

3.2.2 Server-Side Device Compatibility

The ShopPeas backend is hosted and runs on a laptop-based server, which manages database operations (CRUD), API calls, and other backend logic. The server should meet the requirements outlined in Sections 2.4.2 and 2.6.

3.3 Software Interfaces

3.3.1 APIs and Data

ShopPeas utilises two real-time APIs in its operation:

1) OneMap API (https://www.onemap.gov.sg/apidocs/):

   The OneMap API provides an real-time travel information in Singapore developed by the Singapore Land Authority. As ShopPeas currently only supports self-pickup of ordered goods, the OneMap API allows us to provide consumers with a visual of the wholesaler's location, and also handles distance or travel related operations, such as calculating the time and distance between the wholesaler's and consumer's registered locations.

2) Currency Conversion API (https://currencyapi.com/):

   The currency conversion API provides real-time currency conversion with the most updated currency exchange rates. As ShopPeas also supports cross-border sales, specifically for wholesalers in Malaysia, the currency conversion API is currently used to convert transaction prices between SGD and MYR.

ShopPeas also relies on the Healthier Choice Symbol Product List provided by the Department of Statistics, Singapore (https://data.gov.sg/datasets/d_6725eed000bf5b3c5d310eb08de0851f/view), which contains information about healthier choice product names and product quantities. Wholesalers can only sell products from this list, ensuring that ShopPeas provides consumers with healthy and fresh food.

3.3.2 Database

ShopPeas employs the use of the Firestore database in the backend for real-time CRUD operations, data persistence, and the secure storage of all data, including user information (both wholesalers and consumers), product information, current and previous orders and transactions. Firebase Authentication is used to handle user functionality such as registration, login, and updating of profile details in a secure manner.

3.3.3 Frameworks

The frameworks used in building ShopPeas are as follows:

1) Frontend:
   a) React Native (version 0.74.5)
   b) Expo SDK 51.0.28
   c) Firebase (version 10.13.1)
2) Backend:
   a) Spring Boot (version 3.3.3)
   b) Firebase Admin SDK (version 9.3.0)

c) Axios (version 1.7.7)

d) React Native Axios (version 0.17.1)

Section 2.4 provides further details regarding the operating environment of ShopPeas.

3.4 Communications Interfaces

ShopPeas uses several communication interfaces to interact with Firebase for data storage and user authentication, as well as to facilitate secure data exchange between the front and backend.

- **Protocols & Standards:** The frontend communicates with the backend through RESTful API calls over HTTPS to retrieve, update and display data stored in Firebase. The backend serves as an intermediary, receiving API requests from the frontend, processing them, and interacting with Firebase to access and modify the data. Once the backend retrieves the data from Firebase, it formats it using standardised JSON structures facilitated by Data Transfer Objects (DTOs) to ensure consistency and ease of parsing by the frontend.
- **Data Transfer Objects (DTOs):** To maintain consistency and structure in data exchanges, DTOs are used to organise and standardise the data transferred between the backend and frontend. Each DTO defines specific fields to be sent over, ensuring that only relevant information is included in the JSON payload.
- **Role-Based Authorization & Access Control:** ShopPeas implements role-based access control using Spring Security's @PreAuthorize annotations. Each API endpoint is restricted to users with specific roles, as determined by the Authorization header. Hence, ensuring that only authorised users can access certain data and functionality, minimising the risk of unauthorised access.
- **API Key Management:** To protect sensitive information, such as API keys, these keys are stored in environmental files (.env files). This keeps sensitive information out of the codebase and provides an added layer of security by managing keys separately in the deployment environment.
- **Security & Encryption:** All data exchanged between ShopPeas, the backend and Firebase is encrypted over HTTPS. Firebase Authentication is used to verify user identities, ensuring only authenticated users can access ShopPeas.

- **Error Handling & Resilience:** The backend employs robust exception handling to manage and log errors during API communication with Firebase. This ensures fault tolerance by catching and handling specific and general exceptions, ensuring a smooth user experience.
- **Data Transfer Rates:** Firebase infrastructure optimises data transfer rates, allowing ShopPeas to handle concurrent requests effectively while maintaining performance. Hence, the backend is designed to manage high volumes of requests simultaneously.
- **Synchronisation Mechanisms:** The real-time functionality of our database is used to synchronise data instantly, ensuring users are always presented with the latest updates whenever changes are made to the database.

4. System Features

4.1 Consumer Features

4.1.1 Register

4.1.1.1 Description and Priority

Description: A form to get the user's details and create a new account for them to purchase food products from wholesalers.

Priority: High

4.1.1.2 Stimulus/Response Sequences

Use Case ID: CUS01

Flow of Events

1. The user enters the app and sees the landing page.
2. The user clicks the "SIGN UP" option and is directed to a page that asks the user which account they want to create: "Consumer" or "Business Owner".
3. The user clicks the "Consumer" option and is directed to the registration form.
4. The user enters their credentials.
    a. The user provides their first name.
    b. The user provides their last name.
    c. The user provides an email.
    d. The user provides a phone number.
    e. The user clicks "NEXT" to enter details about their address.
5. The user enters their address details.
    a. The user provides their street name.
    b. The user provides their unit number.
    c. The user provides their building name.
    d. The user provides their city, country.
    e. The user provides their postal code.
6. The user chooses their password
    a. The user provides a password.

   b. The user must confirm the password by entering it again in another field.
7. The user clicks "REGISTER" to register their details in the system.
   a. The system validates that the email follows the correct format.
   b. The system checks if there is a duplicate email.
   c. The system validates that the phone number follows the correct format.
   d. The system checks if there is a duplicate phone number.
   e. The system validates that the provided password is secure enough.
   f. The system validates if the password entered is the same in both fields.
8. The system creates a new record in the database.
9. The app prompts the user to log in.

Alternate Flows:

AF01-CUS01: The user enters an invalid input at any stage

1. User inputs invalid information (e.g., incorrect email format).
2. The system displays error messages before allowing the user to register and prompts the user to fix the input values.
3. User corrects the input and resubmits, the system will return to the Login/Sign up Page.

AF02-CUS01: The password entered in the "Password" and "Confirm Password" fields do not match.

1. User keys in a different password in “Confirm Password” compared to the initial Password.
2. The system prompts the user to re-enter the password, "Passwords do not match".

Exceptions

EX01-CUS01: The user enters an email address that already exists.

1. The system checks the database and finds that the email address already exists.
   a. Display an error message, "This email belongs to an existing user. Please log in instead."

EX02-CUS01: Firebase connection cannot be established.

1. Inform the user of the connection issue and ask them to try again later.

4.1.1.3 Functional Requirements

REQ-1. The system must display four text fields for the user to input their credentials.

REQ-1.1. The first field asks for the user's first name.

REQ-1.1.1. The system must display an error message if this field is empty.

REQ-1.2. The second field asks for the user's last name.

REQ-1.2.1. The system must display an error message if this field is empty.

REQ-1.3. The third field asks for the user's email.

REQ-1.3.1. The system must display an error message if this field is empty.

REQ-1.3.2. The system must display an error message if the inputted email does not follow the correct email format (e.g., user@domain.com).

REQ-1.4. The fourth field asks for the user's phone number.

REQ-1.4.1. The system must display an error message if this field is empty.

REQ-1.4.2. The system must display an error message if the inputted number does not follow the correct phone number format: must have 8 digits, starting with either 8 or 9.

REQ-1.5. The system must display a "NEXT" button for the user to enter their address details.

REQ-2. The system must display five text fields for the user to input their address details.

REQ-2.1. The first field asks for the user's street name.

REQ-2.2. The second field asks for the user's unit number.

REQ-2.3. The third field asks for the user's building name.

REQ-2.4. The fourth field asks for the user's city, country.

REQ-2.5. The fifth field asks for the user's postal code.

REQ-3. The system must display two text fields for the user to input their password.

REQ-3.1. The first field asks for the user’s password.

REQ-3.1.1. The system must display an error message if this field is empty.

REQ-3.1.2. The system must display an error message if the inputted password does not follow the password requirements: minimum of 12

characters, contains at least one number, and at least one special character.

REQ-3.2. The second field asks for the user to re-enter the password.

REQ-3.2.1. The system must display an error message if this field is empty.

REQ-3.2.2. The system must display an error message if this field does not match the password field (REQ-3.1).

REQ-4. The system must display a "REGISTER" button for the user to click once all information has been entered, below the two text fields.

REQ-4.1. The system must validate the user's input.

REQ-4.1.1. If the credentials are valid, the user will be redirected to the Login page.

REQ-4.1.2. If the credentials are invalid, the user should not be able to access the Purchase History, and Profile pages.

REQ-5. The system must display a "LOGIN" button for the user to log into their newly created account.

4.1.2 Login

4.1.2.1 Description and Priority

Description: A form to ask for the user's email and password before they can access app functionalities that require user authentication.
Priority: High

4.1.2.2 Stimulus/Response Sequences

Use Case ID: CUS02

Flow of Events

1. The user enters their credentials - an email and password.
    a. The system validates that the email keyed in exists in the database.
    b. The system validates that the provided password matches the password tagged to the account.

2. The user is directed to the Consumer Product Page after successfully logging in.

Alternative Flow

AF01-CUS02: User wants to Register an Account

1. The user clicks the "No Account? Create One!" link.
    a. The user is redirected to the Register page to enter their chosen account details.

AF02-CUS02: User keys in the wrong password

1. User keys in wrong password
    a. System displays an error message, “Wrong password!”

Exception

EX01-CUS02: Firebase connection cannot be established.

1. Inform the user of the connection issue and ask them to try again later.

4.1.2.3 Functional Requirements

REQ-1. The system must display two text fields for the user to input their credentials.

REQ-1.1. The first field asks for the user's email.

REQ-1.1.1. The system must display an error message if this field is empty or does not follow the correct email format (e.g., user@domain.com).

REQ-1.2. The first field asks for the user's password.

REQ-1.2.1. The system must display an error message if this field is empty.

REQ-2. The system must display a "Login" button for the user to click once all required information is provided, below the two text fields.

REQ-2.1. The system must validate the user's inputs.

REQ-2.1.1. If the credentials are valid, the user will be redirected to the Product List page.

REQ-2.1.2. If the credentials are invalid, the user should not be able to access the Purchase History, and Profile pages.

REQ-3. The system must display a "Forgot Password?" link below the "Login" button.

REQ-4. The system must display a "No Account? Create One!" link centre-aligned and below the "Forgot Password?" link.

4.1.3 Product List

4.1.3.1 Description and Priority

Description: The list of healthy food products sold by various wholesalers for the consumers to purchase.
Priority: High

4.1.3.2 Stimulus/Response Sequences

Use Case ID: CUS03
Flow of Events

1. The system queries the database to get all product records.
2. The user scrolls through the products.
    a. Upon selecting a product they are interested in, the system will direct them to the product's respective Product Details page.
3. If the user has a specific product to find, they can enter the item in the search bar.
    a. The system will filter the product records based on the text query entered.
    b. The system will display the filtered list of records.

Exceptions

EX01-CUS03: System unable to retrieve information

1. The System displays the message “Unable to retrieve products, check your connection”.

4.1.3.3 Functional Requirements

REQ-1. Each product in the list should display the product information.

REQ-1.1. The information must consist of the product image.

REQ-1.1.1. The image dimensions should be consistent.

REQ-1.2. The information must consist of the product name.

REQ-1.3. The information must consist of the product package size.

REQ-2. The system must display a search bar with a search icon at the top of the page.

REQ-2.1. If there are no relevant records, the system should display a "No Records Found" message.

4.1.4 Product Details

4.1.4.1 Description and Priority

Description: Displays the details of the healthy food product the user selected, such as the available wholesalers, price, stocks, and location for them to make an informed decision before purchasing the product. Through this page, the user can add products to their shopping cart.

Priority: High

4.1.4.2 Stimulus/Response Sequences

Use Case ID: CUS04

Flow of Events

1. The system queries the database to get all wholesalers selling the product based on the product selected on the Product page.
    a. For each wholesaler, the system will call the OneMap API to get the total time and distance between the wholesaler's pickup location and the consumer's home address.
        i. The time will be displayed in each record.
2. The user reviews the list of wholesaler options.
    a. The user can sort the list based on their priorities.
        i. The system will sort the records based on the chosen field.
        ii. The system will display the sorted records.
    b. For each option, the user can click on the wholesaler's name to view more details about the wholesaler.

i. The wholesaler's name will be highlighted.
ii. The user will be directed to the respective "Wholesaler Details" page.

3. The user selects an option.
    a. The option is highlighted in a different colour.
    b. A pop-up screen appears and the user indicates the quantity they want to add to the cart. A map of the wholesaler's location is also shown using the OneMap API.
        i. If the user selects the "Add to Cart" button, the Shopping Cart page is updated with the addition of this product.
4. The user clicks the return button to continue shopping on the Product page.

Alternative flow

AF01-CUS04: User tries to add an invalid quantity to cart

1. System prompts user to add a valid quantity to cart

Exceptions

EX01-CUS04: System unable to retrieve information

1. The System displays the message “Unable to retrieve products, check your connection”.

EX02-CUS04: User did not type in a valid postal code

1. The system does not display anything on the interface

4.1.4.3 Functional Requirements

REQ-1. The product name will be displayed at the top of the page.

REQ-2. A sorter will be displayed.

REQ-2.1. There will be an option to sort the price.

REQ-2.2. There will be an option to sort the duration.

REQ-2.3. There will be an option to sort the stocks.

REQ-3. The system will display the records of wholesalers selling the selected product in a list format.

REQ-3.1. Each record will display the wholesaler's name.

REQ-3.2. Each record will display the pickup location.

REQ-3.3. Each record will display the duration and distance between the consumer's home address and the wholesaler's pickup address.

REQ-3.4. Each record will display the wholesaler's selling price for the product.

REQ-3.5. Each record will display the wholesaler's stocks.

REQ-4. The user clicks on a row to select a wholesaler to purchase.

REQ-4.1. When the user selects the row, it will be highlighted.

REQ-4.2. A pop-up will appear at the bottom.

REQ-4.2.1. The pop-up will display a number input spinner to get the user's preferred quantity.

REQ-4.2.1.1. The default number of products that consumers must purchase is 1.

REQ-4.2.1.2. There will be a deduct ("-") button.

REQ-4.2.1.2.1. If the current number displayed is 1 (minimum number of products to purchase), the deduct ("-") button will be disabled.

REQ-4.2.1.3. There will be an add ("+") button.

REQ-4.2.1.3.1. If the current number displayed is the wholesaler's current quota, the add ("+" button) will be disabled.

REQ-4.2.1.4. There will be a number between the deduct and add buttons.

REQ-4.2.2. Below the number input spinner, there will be an "Add to Cart" button.

REQ-4.2.3. Besider the ‘Add to Cart’ button, there will be a heart, of which the user can click on it to like the product + wholesaler.

4.1.5 Shopping Cart

4.1.5.1 Description and Priority

Description: Facilitates the purchasing process by allowing users to create a virtual shopping cart. Users can add desired products to their cart, view a detailed list of

items, and calculate the total cost before checkout. This feature provides a convenient way for users to organise their purchases and make informed decisions.
Priority: High

4.1.5.2 Stimulus/Response Sequences

Use Case ID: CUS05

Flow of Events

1. The system displays a list of products added to the cart, where each product detail is shown together in a list item.
2. Product details shown include quantity ordered, wholesaler ordered from, wholesaler address, and price of that specific item.
3. The system calculates the total cost of all items within the cart and displays it at the bottom panel as “Total $__”.
4. The user may complete their purchase by clicking the button to check out.
    a. The user is directed to the Payment page.

Alternate Flows

AF01-CUS05: The user deletes an item.

1. The user clicks the delete icon associated with the product item.
2. The item is removed from the shopping cart.
3. The system recalculates the total amount.

AF02-CUS05: The user updates the quantity.

1. The user uses the number input spinner to update the quantity.
2. The system updates the price within the product item row.
3. The system recalculates the total amount.

AF03-CUS05: The user clicks the wholesaler name.

1. The system directs the user to the respective View Whoelsaler page.

AF04-CUS05: The user checks out when the updated stock amount is less than the requested quantity.

1. The Shopping Cart page loads.
2. The system gets the quota to update the maximum number for the number input spinner for each product.
    a. If the stock is less than the user's selected quantity.
        i. If the stock is not 0.
            1. Change the quantity in the number input slider to the wholesaler's maximum stock.
            2. Update the price for the product.
            3. Display an alert to inform the user that the wholesaler's stock has been updated.
        ii. If the stock is 0.
            1. Change the quantity in the number input slider to 0.
            2. Disable all buttons in the number input slider.
            3. Change the price to 0 for the product.
            4. Display an alert to inform the user that the wholesaler has no more stocks and ask them to consider other wholesalers.
                a. If the user selects another wholesaler.
                    i. Update the price for the product.
                    ii. Change the quantity in the number input slider to 1.
                    iii. Enable all buttons in the number input slider.
                b. If the user does not select another wholesaler and checks out.
                    i. The system removes the product from the purchase history records.

AF05-CUS05: The user has an empty cart

1. The Shopping Cart page loads.
2. System displays a message stating “No Cart Items Yet”.

AF06-CUS05: The user tries to checkout an empty cart.

1. The Shopping Cart page loads.
2. User clicks on the checkout button.
3. System displays an error message and prevents the user from checking out the cart.

Exceptions

EX01-CUS05: Retrieval of shopping cart data for the user fails.

1. System displays an error message

4.1.5.3 Functional Requirements

REQ-1. Display each product item in a table row and divide the product information into three columns.

REQ-1.1. The left column displays the product image.

REQ-1.2. The centre column displays product information and user choices.

REQ-1.2.1. Display the product name on top.

REQ-1.2.2. Display a dropdown list with the selected wholesaler as the current option below the product name.

REQ-1.2.2.1. If the user chooses another wholesaler, the chosen option will be updated.

REQ-1.2.3. Display the total price for the product by multiplying the quantity and the unit price at the bottom left side.

REQ-1.2.4. Display a number input spinner to update the quantity on the bottom right side.

REQ-1.3. The right column displays a delete button.

REQ-1.3.1. When the delete button is clicked, the record is removed from the shopping cart.

REQ-2. Display the total price of all products at the bottom left of the screen.

REQ-3. Display a "Check Out" button on the right of the total price.

REQ-3.1. When the button is clicked, it redirects the user to the Payment page.

4.1.6 Check Out

4.1.6.1 Description and Priority

Description: For consumers to review their order details, enter payment information, and finalise their purchase.

Priority: High

4.1.6.2 Stimulus/Response Sequences

Use Case ID: CUS06

Flow of Events

1. The shopping cart data is retrieved from the database
    a. The price of each item is checked for any updates and the prices of the individual items and the total payment amount are updated and reflected accordingly in the transactions and shopping cart data records.
2. The total payment amount is calculated and displayed.
3. The user clicks the Check out button.
4. The user is directed to the payment page where the payment methods and the item he is purchasing is displayed.
5. The user clicks into Payment Method to choose the payment method.
6. The user's current payment methods are displayed (if any)
    a. If the user has not selected a payment method, they can click a button to add a payment method which will direct them to a page to add a new credit/debit card.
7. The user selects a payment method.
8. The user clicks the "Make Payment" button to transfer the payment to the wholesaler.
9. The interface shows "Success! Card Successfully added".
    a. The amount is converted to the wholesaler's preferred currency (SGD or MYR).
10. Upon successful payment: the following changes occur:

a. The status of the transaction data is changed from “IN-CART” to “PENDING-ACCEPTANCE”.
b. The current shopping cart is emptied
c. A new record is added to the Order History table for data persistence.

Alternate Flows

AF01-CUS06: Data entered into the form fields are in the wrong format.

1. An error message is displayed to notify the user of the incorrect data format and prompt them to re-enter.

AF02-CUS06: User selects a new payment method.

1. User clicks the "Payment Method" button in the Payment page which directs them to the Payment Method page.
2. The user selects a new payment method listed and is directed to an Add Card page for them to enter their card details.
3. The user clicks "Submit" after filling in the form.
4. The form is validated.
5. The system directs the user back to the Payment page.
6. The new card details will be updated as an existing payment method.

AF03-CUS06: Shopping cart is empty.

1. After selecting the “Cart” section, the system is unable to find any cart records for the user.
2. The system displays a message “No Cart Items”.
3. Users can navigate to other sections of the app.

Exceptions

EX01-CUS06: Input data contains invalid wholesaler details.

1. System displays an error message

EX02-CUS06: Input data contains invalid order details.

1. System displays an error message

EX03-CUS06: Retrieval of payment method data for user fails

1. System displays an error message

4.1.6.3 Functional Requirements

REQ-1. Display the user's payment information and cart items.

REQ-2. Display a list of the consumer's existing payment methods and allow the user to select one.

REQ-2.1. If the user does not have any payment methods added to the application, they may add in a new payment method.

REQ-2.2. When adding a new payment method, dynamically display the form to collect information about the selected payment method.

REQ-2.2.1. Display a field to ask for the card number.

REQ-2.2.2. Display a field to ask for the expiry date in the format "MM/YY".

REQ-2.2.3. Display a field to ask for the CVV.

REQ-2.2.4. Display a field to ask for the cardholder's name.

REQ-3. Once a payment method has been selected, display a "Check Out" button below the form for the user to pay the wholesaler.

REQ-4. Once the items in the cart have been checked out, display a confirmation message and bring the user to the Order History Page.

4.1.7 Order History

4.1.7.1 Description and Priority

Description: To aid the user experience, the purchase history will allow users to be able to view what they had previously ordered and rate their orders.

Priority: Medium

4.1.7.2 Stimulus/Response Sequences

Use Case ID: CUS07

Flow of Events

1. The user selects the “History” option from the bottom tab navigation.
2. The system retrieves the user order data from the database and displays it.
3. For each order, the system will display the relevant information such as
    a. Name of Wholesaler(s).
    b. Date of order purchase.
    c. List of items purchased.
    d. Quantity of items.
    e. Price of individual item.
    f. Status of the order.
    g. Total Amount paid.
    h. Status on whether the order has been rated.
4. The user can rate an item by clicking onto the “Give Rating” button and give the appropriate rating out of 5 stars for the wholesaler.
5. The wholesaler rating will be updated in the frontend UI and backend database.
6. The interface shows the “Give Rating” icon turning dark green “Rated” to indicate success.
7. The user can navigate to other sections of the app using the navigation menu.

Alternate Flows

AF01- CUS07: User has no purchase history

1. After selecting the “History” section, the system is unable to find past orders for the user.
2. The system displays a message “No past orders”.
3. Users will be given the option to start shopping or view other sections.

4.1.7.3 Functional Requirements

REQ-1. The system shall display a “History” section in the user’s bottom navigation.

REQ-2. The system should retrieve all past orders associated with the user’s account.

REQ-3. When the user goes to a specific order entry, the system shall display the relevant information.

REQ-3.1. Name of wholesaler(s).

REQ-3.2. Date of order purchase.

REQ-3.3. A list of products purchased in that order.

REQ-3.4. Product names, quantities and individual prices.

REQ-3.5. Status on whether the order has been rated.

REQ-3.6. The total amount paid for the order.

REQ-3.7. Rat

REQ-4. The system shall display a "Give Rating" button for unrated orders which allows the user to rate that order.

REQ-5. The system shall display that an unclickable "Rated" button for orders that have already been rated.

REQ-6. The system shall allow the user to navigate to other sections of the app.

4.1.8 Wholesaler Details

4.1.8.1 Description and Priority

Description: Users will be allowed search and view detailed information about different wholesalers on ShopPeas. This enhances the user experience by enabling consumers to conveniently find wholesalers that meet their needs, compare offerings and make informed purchasing decisions.

Priority: High

4.1.8.2 Stimulus/Response Sequences

Use Case ID: CUS08

Flow of Events

1. The user is viewing products on the Product Details page.
2. The user clicks on the wholesaler's name.

3. The system directly retrieves the wholesaler's information.
4. The user is redirected to the wholesaler's details page to view more detailed information including:
    a. Wholesaler name and logo.
    b. Location and address.
    c. List of products offered.
    d. Ratings.

Alternate Flows

AF01-CUS08: View Wholesaler from the Shopping Cart Page

1. The user is viewing their shopping cart.
2. The user clicks on the wholesaler's name.
3. The system directly retrieves the information of the wholesaler.
4. The user is redirected to the wholesaler's details page.

AF02-CUS08: View Wholesaler from the Order History Page

1. The user is viewing their order history.
2. The user clicks on the wholesaler's name.
3. The system directly retrieves the information of the wholesaler.
4. The user is redirected to the wholesaler's details page.

4.1.8.3 Functional Requirements

REQ-1. The consumer can click on the wholesaler name in the Product Details, Shopping Cart, and Order History pages, and they will be directed to the View Wholesaler page.

REQ-2. The system shall retrieve and display a list of wholesalers based on the user's search criteria.

REQ-2.1. Each wholesaler entry shall include the following relevant information.

REQ-2.1.1. Wholesaler name

REQ-2.1.2. Products offered

REQ-2.1.3. Location

REQ-2.1.4. Wholesaler rating

REQ-3. When a user selects a wholesaler, the system shall display detailed information about the wholesaler.

REQ-3.1. Wholesaler name and logo.

REQ-3.2. Location and address.

REQ-3.3. List of products offered.

REQ-3.4. Ratings.

REQ-4. The system shall provide a "Back" button to allow the user to return to the page they navigated from.

4.1.9 Profile

4.1.9.1 Description and Priority

Description: The profile feature allows users to view and manage their personal information within ShopPeas platform. Users can access their profiles to review current details and make updates as needed. This feature enhances user experience by providing a centralised location for managing account information.

Priority: Low

4.1.9.2 Stimulus/Response Sequences

Use Case ID: CUS09

Flow of Events

1. The user clicks on the "Profile" button in the bottom tab navigation menu.
2. The system displays the user's profile page with current information such as Name, Date Joined, Location, Email, Contact, and Address.
3. The user selects the "Edit Profile" option
4. The system presents editable fields for the user to update their information and fill in with current information.
    a. Users can change their contact details, street name, unit number, building name, city and postal code by editing the fields.

5. The user makes changes and clicks the “Save” button.
6. The system validates the input, saves the changes and displays a confirmation message “Success! Profile has been updated”.
7. The system navigates back to the Explore page.
8. The updated information will be stored in the backend Firestore database.
9. Updated profile information is shown to the user if the user accesses the profile page.

Alternate Flows:

AF01-CUS09: Invalid Contact Details

1. The user enters an invalid contact details format
2. The system displays error messages when the user tries to save details and prompts users to re-enter relevant fields.
3. The user corrects the input and resubmits

AF04-CUS09: No Changes Made

1. The user clicks “Save” without making any changes
2. The system detects no changes and returns back to the Explore page.

AF05-CUS09: Cancels editing

1. The user clicks on the “cancel” button.
2. The system brings the user back to the Profile page.

Exceptions:

EX01-CUS09: Phone number already exists.

1. Prompt the user to enter another phone number.

4.1.9.3 Functional Requirements

REQ-1. The system shall provide a "Profile" button in the main navigation menu to access the user's profile.

REQ-2. The system shall display the user's profile information when accessed.

REQ-2.1. The profile display shall include the following information:

REQ-2.1.1. User's name

REQ-2.1.2. Email address

REQ-2.1.3. Shipping address(es)

REQ-2.1.4. Phone number

REQ-2.1.5. Date joined

REQ-3. The system shall provide an edit profile button to create a form that allows users to modify their information.

REQ-3.1. When in edit mode, the system shall allow users to update the following information:

REQ-3.1.1. Email address

REQ-3.1.2. Location

REQ-3.1.3. Phone number

REQ-3.2. The system shall validate all user inputs before saving changes.

REQ-3.3. The system shall display error messages for any invalid inputs.

REQ-3.4. The system shall provide a "Save" button to update the profile information.

REQ-3.5. The system shall provide a “Cancel” button to return the profile page.

REQ-3.6. The system shall display a confirmation message upon successful update of profile information.

4.2 Wholesaler Features

4.2.1 Register

4.2.1.1 Description and Priority

Description: The registration feature allows wholesalers to create an account on the ShopPeas. This multi-step process collects essential information about the wholesaler, including company details, address, bank account information, and account credentials. The registration process is designed to be straightforward while ensuring that all necessary data is collected for proper vetting and account setup.

Priority: High

4.2.1.2 Stimulus/Response Sequences

Use Case ID: WH01

Flow of Events:

1. The user enters the app and sees the landing page.
2. The user clicks the "SIGN UP" option and is directed to a page that asks the user which account they want to create: "Consumer" or "Business Owner".
3. The user clicks the "Business Owner" option and is directed to the registration form.
4. The user enters their credentials.
    a. The user provides their company name
    b. The user provides an email.
    c. The user provides a phone number.
    d. The user clicks "NEXT" to enter details about their address.
5. The user enters their address details.
    a. The user provides their street name.
    b. The user provides their unit number.
    c. The user provides their building name.
    d. The user provides their city, country.
    e. The user provides their postal code.
6. The user enters their address details.
    a. The user provides their street name.
    b. The user provides their unit number.
    c. The user provides their building name.
    d. The user provides their city, country.
    e. The user provides their postal code.
7. The user enters their bank account details
    a. The user provides their bank account name.
    b. The user provides their bank account number
    c. The user provides the bank.
8. The user chooses their password
    a. The user provides a password.
    b. The user must confirm the password by entering it again in another field.

9. The user clicks "REGISTER" to register their details in the system.
    a. The system validates that the email follows the correct format.
    b. The system checks if there is a duplicate email.
    c. The system validates that the phone number follows the correct format.
    d. The system checks if there is a duplicate phone number.
    e. The system validates that the provided password is secure enough.
    f. The system validates if the password entered is the same in both fields.
10. The system creates a new record in the database.
11. The app prompts the user to log in.

Alternate Flows:

AF01-WH01: The user enters an invalid input at any stage

1. The wholesaler inputs invalid information (e.g., incorrect email format).
2. The system displays error messages before allowing the user to register and prompts the user to fix the input values.
3. The wholesaler corrects the input and resubmits, the system will return to the Login/Sign up Page.

AF02-WH01: The password entered in the "Password" and "Confirm Password" fields do not match.

1. User keys in a different password in “Confirm Password” compared to the initial Password.
2. The system prompts the user to re-enter the password, "Passwords do not match".

Exceptions

EX01-WH01: Existing email or UEN

1. The wholesaler inputs an existing email or UEN.
2. The system checks with the database and confirms the existing email/UEN.
3. The system displays specific error messages that the email or UEN is already registered.
4. The wholesaler inputs a different valid email or UEN.

4.2.1.1 Functional Requirements:

REQ-1. The system shall provide a registration option specifically for wholesalers.

REQ-2. The system shall implement a multi-step registration process for wholesalers.

REQ-3. The system shall collect and validate company information:

- REQ-3.1. The system shall require the following inputs:
  - REQ-3.1.1. Company Name
  - REQ-3.1.2. Unique Entity Number (UEN)
  - REQ-3.1.3. Email address
- REQ-3.2. The system shall validate the input of this information:
  - REQ-3.2.1. Company Name
  - REQ-3.2.2. Unique Entity Number (UEN)
  - REQ-3.2.3. Email
- REQ-3.3. The system shall check for existing accounts using the provided email or UEN.

REQ-4. The system shall collect and validate address information:

- REQ-4.1. The system shall require the input of the wholesaler's registered address.
  - REQ-4.1.1. Street Name
  - REQ-4.1.2. Unit No.
  - REQ-4.1.3. Building Name
  - REQ-4.1.4. City, Country
  - REQ-4.1.5. Postal Code
- REQ-4.2. The system shall validate the address format for completeness and accuracy.

REQ-5. The system shall collect and validate bank account information:

- REQ-5.1. The system shall require the input of bank account details.
- REQ-5.2. The system shall validate the format of bank account details.

REQ-6. The system shall manage password creation:

- REQ-6.1. The system shall require the wholesaler to choose a password.
- REQ-6.2. The system shall require the wholesaler to confirm the chosen password.
- REQ-6.3. The system shall enforce password strength requirements.

REQ-6.4. The system shall validate that the confirmed password matches the chosen password.

REQ-7. The system shall provide navigation controls:

REQ-7.1. The system shall provide a "NEXT" button to progress through registration steps.

REQ-7.2. The system shall provide a "REGISTER" button on the final step to complete registration.

REQ-8. The system shall implement error handling and validation:

REQ-8.1. The system shall display specific error messages for invalid inputs.

REQ-8.2. The system shall prevent progression to the next step if any required field is invalid or empty.

REQ-8.3. The system shall display an error message if an email or UEN is already registered.

REQ-8.4. The system shall display an error message if the password does not meet the requirements.

REQ-8.5. The system shall display an error message if the confirmed password does not match the chosen password.

4.2.2 Product Management

4.2.2.1 Description and Priority

Description: The Product Management feature allows wholesalers to add, edit, delete, and manage their product listings on the ShopPeas platform. This feature enables wholesalers to maintain an up-to-date catalogue of their offerings, including product details, pricing, inventory levels, and product images.

Priority: High

4.2.2.2 Stimulus/Response Sequences

Use Case ID: WH02

Flow of Events:

1. The wholesaler logs into ShopPeas.
    a. The system displays the wholesaler's current products, with options for adding new products, viewing existing products, deleting products and editing product information.
2. The wholesaler adds a new product.
    a. The wholesaler clicks on the “Add New Product” button
    b. The system displays a form with several required information:
        i. Wholesaler clicks on the selected Product Name based on the existing list given.
        ii. Wholesaler input the Price
        iii. Wholesaler input the Current Stock
    c. The wholesaler fills in the input fields and clicks on the “Add New Product” Button.
    d. The system validates the input, saves the new product and adds the product to the top of the wholesaler’s “My Products” page.
3. The wholesaler edits an existing product.
    a. The wholesaler selects a product from the list and clicks on the edit product icon button.
    b. The system displays a form with several required information:
        i. Price
        ii. Current Stock
    c. The wholesaler fills in the input fields and clicks on the “Update” Button.
    d. The system validates the input
    e. The system displays “ Success Product Edited!”
    f. System saves the changes and displays an updated product list.
4. The wholesaler deletes a product.
    a. The wholesaler selects a product from the list and clicks on the delete product icon button.
    b. The system displays a confirmation dialogue.
    c. The wholesaler confirms the deletion.

d. The system displays “Success Product Removed!”.

e. The system removes the products and updates the product list.

Alternate flows:

AF01-WH02: Invalid Product information

1. The wholesaler enters invalid or incomplete product information.
2. The system does not update the product listing.
3. The wholesaler resubmits the form to add new products.

4.2.2.3 Functional Requirements

REQ-1. The system shall provide a Product Management page for wholesalers.

REQ-2. The system shall allow wholesalers to add new products that they do not currently offer within our list of healthy products:

REQ-2.1. The system shall provide a form to add a new product:

REQ-2.1.1. A dropdown field of other healthy products they do not currently sell

REQ-2.1.2. Price

REQ-2.1.3. Current Stock

REQ-2.2. The system shall validate all entered product information before saving.

REQ-3. The system shall enable wholesalers to edit existing products:

REQ-3.1. The system shall display current product information in an editable form.

REQ-3.2. The system shall allow the updating of all product fields including images.

REQ-4. The system shall allow wholesalers to delete products:

REQ-4.1. The system shall require confirmation before deleting a product.

REQ-5. The system shall implement error handling and validation:

REQ-5.1. The system shall display specific error messages for invalid inputs.

REQ-6. The system shall ensure data consistency:

REQ-6.1. The system shall update the database in real-time when changes are made on the frontend.

4.2.3 Order Management (Pending Orders)

4.2.3.1 Description and Priority

Description: Wholesaler will be able to view and manage orders placed by consumers that have not yet been completed (Pending Orders). Pending Orders are categorised into “To be accepted” and “To be completed”. Orders can also be searched by Order ID or product.

Priority: High

4.2.3.2 Stimulus/Response Sequences

Use Case ID: WH03

Flow of Events:

1. The wholesaler accesses the Pending Orders page.
    a. The wholesaler clicks on the “Transactions” tab at the bottom navigation menu and then selects “Pending Orders” from the drop-down menu.
    b. By default, the System will display Orders (old to new) that are “To be accepted”. Information displayed includes:
        i. Order ID (OID)
        ii. Product Name
        iii. Order Quantity
        iv. Total Price according to the wholesaler’s currency preference

2. The wholesaler Accepts an order
    a. The wholesaler navigates to the Pending Orders, “To be accepted” page.
    b. The wholesaler clicks on the “Accept” button for the Order that the wholesaler wants to Accept.
        i. The status of the order (transaction) is updated in Firebase.
    c. The Order that was Accepted can now be found under the “To be completed” page.

3. The wholesaler accesses the Pending Orders, “To be completed” page
    a. The wholesaler clicks on the “Transactions” tab at the bottom navigation menu and then selects “Pending Orders” from the drop-down menu. The Wholesaler then selects the “To be completed” tab.
    b. The number of Orders that are “To be completed” are displayed.
    c. The System displays all the orders that have yet to be completed. Information displayed includes:
        i. Order ID (OID)
        ii. Product Name
        iii. Order Quantity
        iv. Total Price according to the wholesaler’s currency preference

4. The wholesaler Completes an Order
    a. The wholesaler navigates to the Pending Orders, “To be completed” page.
    b. The wholesaler clicks on the “Complete” button for the Order that the wholesaler wants to Complete.
        i. The status of the order (transaction) is updated in Firebase.
    c. The Order that was Completed can now be found under the “Completed Orders” page.

5. The wholesaler searches for an Order
    a. The wholesaler navigates to the Pending Orders page.
    b. The wholesaler enters either the Order ID or the product name in the search bar.
    c. The System displays the search result.

Alternative Flows

AF01-WH03: No search results based on specific products

1. The system is unable to find specific products matching the wholesalers's search criteria.
2. The system displays no results.

AF02-WH03: No Pending Orders to be Accepted

1. The system is unable to find any Orders that have not been Accepted.
2. The system displays the message "No results found".

AF03-WH03: No Pending Orders to be Completed

1. The system is unable to find any Orders that have not been Completed.
2. The system displays the message "No results found".

Exceptions

EX01-WH03: Unable to Retrieve Transactions by Status

1. The system displays an error message.

4.2.3.3 Functional Requirements

REQ-1.1. The system shall have a search bar for the searching of Orders.

REQ-1.2. The system shall a dropdown menu with the options Pending Orders and Completed Orders

REQ-1.3. The system shall have two tabs under the Pending Orders Page: "To be Accepted" and "To be Completed".

REQ-1.4. The System shall display, for each "To be Accepted" Order, the following information:

REQ-1.4.1. Order ID (OID)

REQ-1.4.2. Product Name

REQ-1.4.3. Order Quantity

REQ-1.4.4. Total Price according to the wholesaler's currency preference

REQ-1.5. The System shall display, for each “To be Accepted” Order, an “Accept” button.

REQ-1.6. The System shall display, for each “To be Completed” Order, the following information:

REQ-1.6.1. Order ID (OID)

REQ-1.6.2. Product Name

REQ-1.6.3. Order Quantity

REQ-1.6.4. Total Price according to the wholesaler’s currency preference

REQ-1.7. The System shall display, for each “To be Accepted” Order, a “Complete” button.

4.2.4 Order Management (Completed Orders)

4.2.4.1 Description and Priority

Description: Wholesalers will be able to view and manage orders placed by consumers that have been completed (Completed Orders). Orders can also be searched by Order ID or product.

Priority: High

4.2.4.2 Stimulus/Response Sequences

Use Case ID: WH04

Flow of Events:

1. The wholesaler accesses the Completed Orders page.
    a. The wholesaler clicks on the “Transactions” tab at the bottom navigation menu and then selects “Completed Orders” from the drop-down menu.
    b. System will display Completed Orders (old to new). Information displayed for each Order includes:
        i. Order ID (OID)

ii. Product Name
iii. Order Quantity
iv. Total Price according to the wholesaler's currency preference.

2. The wholesaler searches for an Order
    a. The wholesaler navigates to the Completed Orders page.
    b. The wholesaler enters either the Order ID or the product name in the search bar.
    c. The System displays the search result.

Alternative Flows

AF01-WH04: No search results based on specific products

1. The system is unable to find specific products matching the wholesalers's search criteria.
2. The system displays no results.

AF02-WH04: No Completed Orders

1. The system is unable to find any Orders that have been Completed.
2. The system displays the message "No results found".

Exceptions

EX01-WH04: Unable to Retrieve Completed Transactions

1. The system displays an error message.

4.2.4.3 Functional Requirements

REQ-1.1. The system shall have a search bar for the searching of Orders.

REQ-1.2. The System shall display, for each Completed Order, the following information:

REQ-1.2.1. Order ID (OID)

REQ-1.2.2. Product Name

REQ-1.2.3. Order Quantity

REQ-1.2.4. Total Price according to the wholesaler's currency preference

REQ-1.3. The System shall display, for each Completed Order, a "Completed" tag.

4.2.5 Profile Management

4.2.5.1 Description and Priority

Description: Wholesalers will be able to view their Profile details, including account details and products. The wholesaler may edit their account details.

Priority: High

4.2.6.2 Stimulus/Response Sequences

Use Case ID: WH05

Flow of Events:

1. The wholesaler accesses the Profile page.
    a. The wholesaler clicks on the Profile button in the bottom navigation menu.
    b. The System displays the following information:
        i. Wholesaler Profile Image
        ii. Wholesaler Business Name
        iii. Date Joined
        iv. Location
        v. Currency
        vi. Unique Entity Number (UEN)
        vii. Account details:
            1. Contact number
            2. Email
            3. Address

4. Payment details

c. From the Profile page, the wholesaler may navigate to the "My Products" page or the "Settings" page.

2. The wholesaler edits their account details
    a. The wholesaler navigates to the "Profile" page and clicks on the "edit" button.
    b. The wholesaler can then amend the following items. The form field will display the original values.
        i. Select Preferred Currency (SGD/MYR) using the drop-down panel
        ii. Contact Details
        iii. Address
            1. City
            2. Street name
            3. Unit Number
            4. Building Name
            5. Postal Code
        iv. Bank Account Details
            1. Bank
            2. Bank Number
    c. The wholesaler can press the "Save" button to save their changes.
    d. The system will validate the input values and prompt the user if there are invalid input values. The system will not allow the user to save their changes until the input has been corrected.
    e. The wholesaler can press the "Cancel" button to cancel their changes.

3. The wholesaler accesses the My Products page
    a. The wholesaler navigates to the "Profile" page and clicks on the "My Products" button.

b. The System will display the “My Products” page.

Alternative Flows

AF01-WH05: Invalid Contact Details

1. The user enters an invalid contact details format.
2. The system displays error messages when the user tries to save details and prompts users to re-enter relevant fields.
3. The user corrects the input and resubmits

AF02-WH05: No Changes Made

1. The user clicks “Save” without making any changes
2. The system detects no changes and returns to the Profile page.

AF03-WH05: Cancels editing

1. The user clicks on the “cancel” button.
2. The system brings the wholesaler back to the Profile page.

4.2.5.3 Functional Requirements

REQ-1.1. The system display the following information:

REQ-1.1.1. Wholesaler Profile Image

REQ-1.1.2. Wholesaler Business Name

REQ-1.1.3. Date joined

REQ-1.1.4. Location

REQ-1.1.5. Currency

REQ-1.1.6. Unique Entity Number (UEN)

REQ-1.1.7. Account details:

REQ-1.1.7.1. Contact number

REQ-1.1.7.2. Email

REQ-1.1.7.3. Address

REQ-1.1.7.4. Payment details

REQ-1.2. The System shall not allow the User to edit the Business Name and the UEN.

REQ-1.3. The System shall allow the User to edit the following Account Details:

REQ-1.3.1. Location

REQ-1.3.2. Currency

REQ-1.3.3. Account details:

REQ-1.3.3.1. Contact number

REQ-1.3.3.2. Email

REQ-1.3.3.3. Address

REQ-1.3.3.4. Payment details

REQ-1.4. The System shall allow the User to Save or Cancel any changes made to the Account Details

REQ-1.5. The System shall include buttons to allow the User to navigate to the Settings Page and the My Products Page.

REQ-1.6. The System shall provide a “Back” button to allow the User to return to the Home Page

5. Nonfunctional Requirements

For a mobile app focused on selling healthy groceries wholesale, with three user groups (consumers and wholesalers), we'll need to consider several non-functional requirements. These requirements ensure the app's quality, performance, and usability. Here are some key non-functional requirements to consider:

5.1 Performance Requirements

REQ-PERF-1. Each page should take no more than 3 seconds to load under normal operating conditions with 80% concurrent users.

REQ-PERF-2. The system must display the contents of the Product Details page within 3 seconds with 80% concurrent users.

REQ-PERF-3. The system shall ensure that the time to add an item to the shopping cart does not exceed 2 seconds with 80% concurrent users.

REQ-PERF-4. The system must be able to calculate the total price whenever there are updates to the cart within 2 seconds with 80% concurrent users.

5.2 Safety Requirements

REQ-SAFE-1. The system should ensure that wholesalers are legitimate and the pickup locations are safe for the consumers.

REQ-SAFE-2. The system should ensure that the wholesaler's grocery products follow e-commerce and food safety regulations.

5.3 Security Requirements

REQ-SEC-1. User authentication and authorisation must be implemented. This will restrict access to information based on individual user roles and permissions.

REQ-SEC-2. To adhere to data protection regulations such as the Personal Data Protection Act (PDPA) and Payment Card Industry Data Security Standard (PCI DSS), the consumers'

private details such as address and payment details must be kept confidential and cannot be disclosed to other users.

REQ-SEC-3. The system must ensure the encryption of sensitive data such as passwords and payment information in data storage and transfer.

REQ-SEC-4. Database access should be limited to only the system administrator and application developers.

5.4 Software Quality Attributes

1. Scalability
    a. The system shall scale to accommodate an increasing number of concurrent users and transactions without degradation in performance.
    b. The application shall efficiently manage and display large product catalogues, ensuring seamless navigation and quick response times.
2. Reliability
    a. The system shall achieve an uptime of 99.9% or higher, ensuring consistent availability.
    b. The application shall maintain stable performance during peak usage times, such as major sales events.
    c. Robust error handling and recovery mechanisms shall be in place to handle unexpected failures and ensure continuity of service.
    d. Regular and automated data backups shall be implemented to prevent data loss, with quick recovery processes in case of any data corruption.
3. Usability
    a. The application shall feature an intuitive and user-friendly interface designed for easy navigation by both novice and experienced users.
    b. The user experience shall be consistent across all devices and screen sizes, ensuring responsiveness, accessibility, and ease of use on different mobile device sizes.
4. Compatibility
    a. The application shall support major mobile operating systems, ensuring broad accessibility for users.

b. Compatibility shall extend to all modern web browsers, providing a seamless experience across different platforms.

5. Data Integrity
    a. The system shall ensure data accuracy and consistency across all components of the application, preventing discrepancies in user and transaction data.
    b. Regular backups and data recovery mechanisms shall be in place to protect against data loss and ensure the integrity of the data stored within the system.
    c. The system shall ensure payment verification by validating whether an order is successful and the accuracy of the payment amount.
6. Localization
    a. The application shall support multiple currencies, allowing users from different regions to interact with the system in their preferred currency.
7. Maintainability
    a. The system shall be designed for ease of maintenance, with a modular architecture that facilitates updates and enhancements.
    b. Comprehensive documentation shall be provided, covering both the codebase and system architecture, to support ongoing development and maintenance efforts.

5.5 Business Rules

REQ-BUS-1. Unauthenticated users can access the login, register, and product page. If they try to access any other pages, they will be prompted to log in.

REQ-BUS-2. Authenticated consumers can access all pages in section 4.1 Consumer Features.

REQ-BUS-3. Authenticated wholesalers can access all pages in section 4.2 Wholesaler Features.

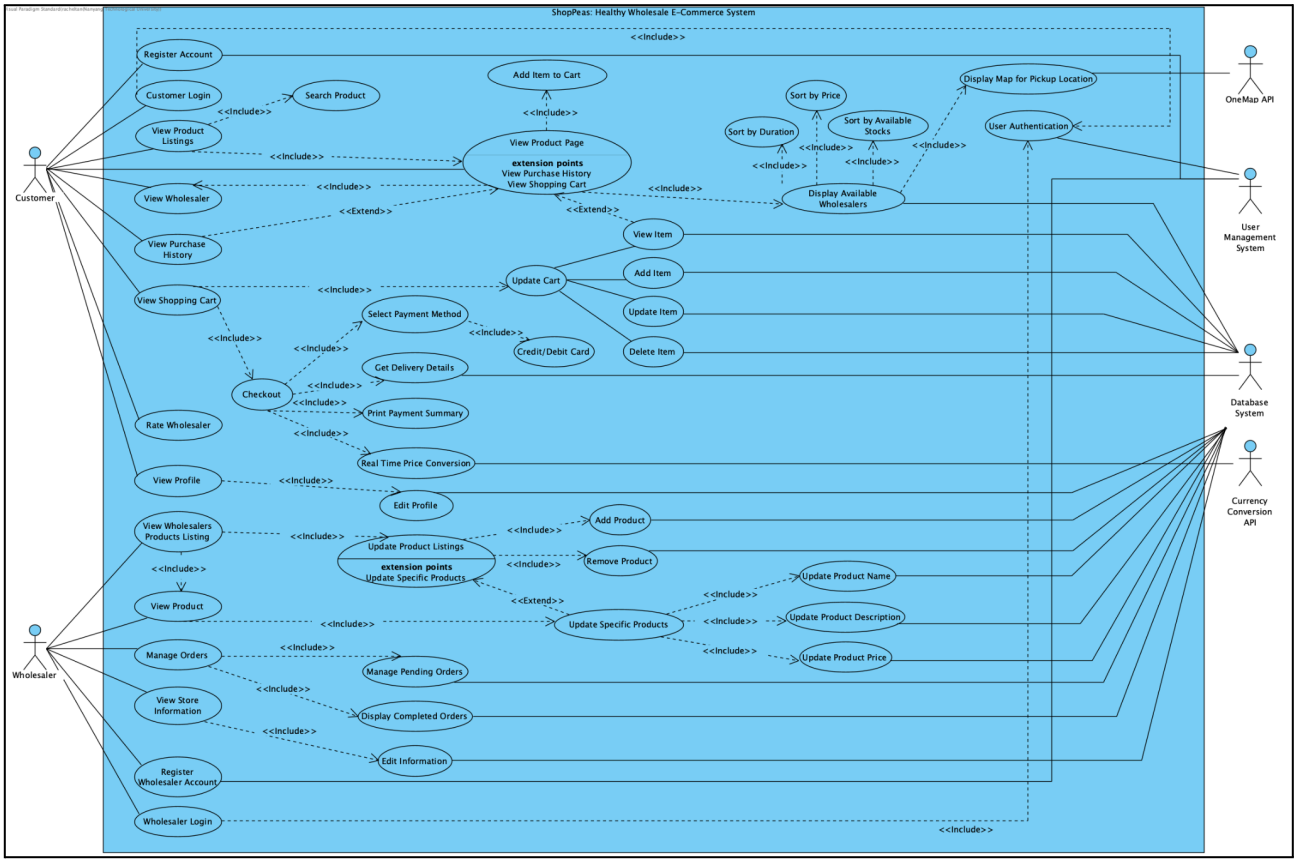
Appendix A: Data Dictionary

Term	Definition
System	
Application Programming Interface (API)	An intermediary that facilitates communication and data exchange between the front end and the back end of ShopPeas.
Authentication	The process of verifying a user's identity to grant access to ShopPeas
Encryption	The process of encoding sensitive data, such as payment information, to protect it from unauthorised access
Session	The duration of time a user is actively using ShopPeas, from login to logout
User	Any individual who interacts with or uses ShopPeas. They can belong to different categories such as Consumers and Wholesalers.
User Interface (UI)	The visual part of the application through which users interact with the system
User Experience (UX)	The overall experience and satisfaction a user has when interacting with ShopPeas
General System	
Account	A personal profile that allows users to access the features and services on ShopPeas. The account includes user credentials and personal information.
Home Page	The main interface that users (Consumers and Wholesalers) see when they log into the system.
Consumer System	

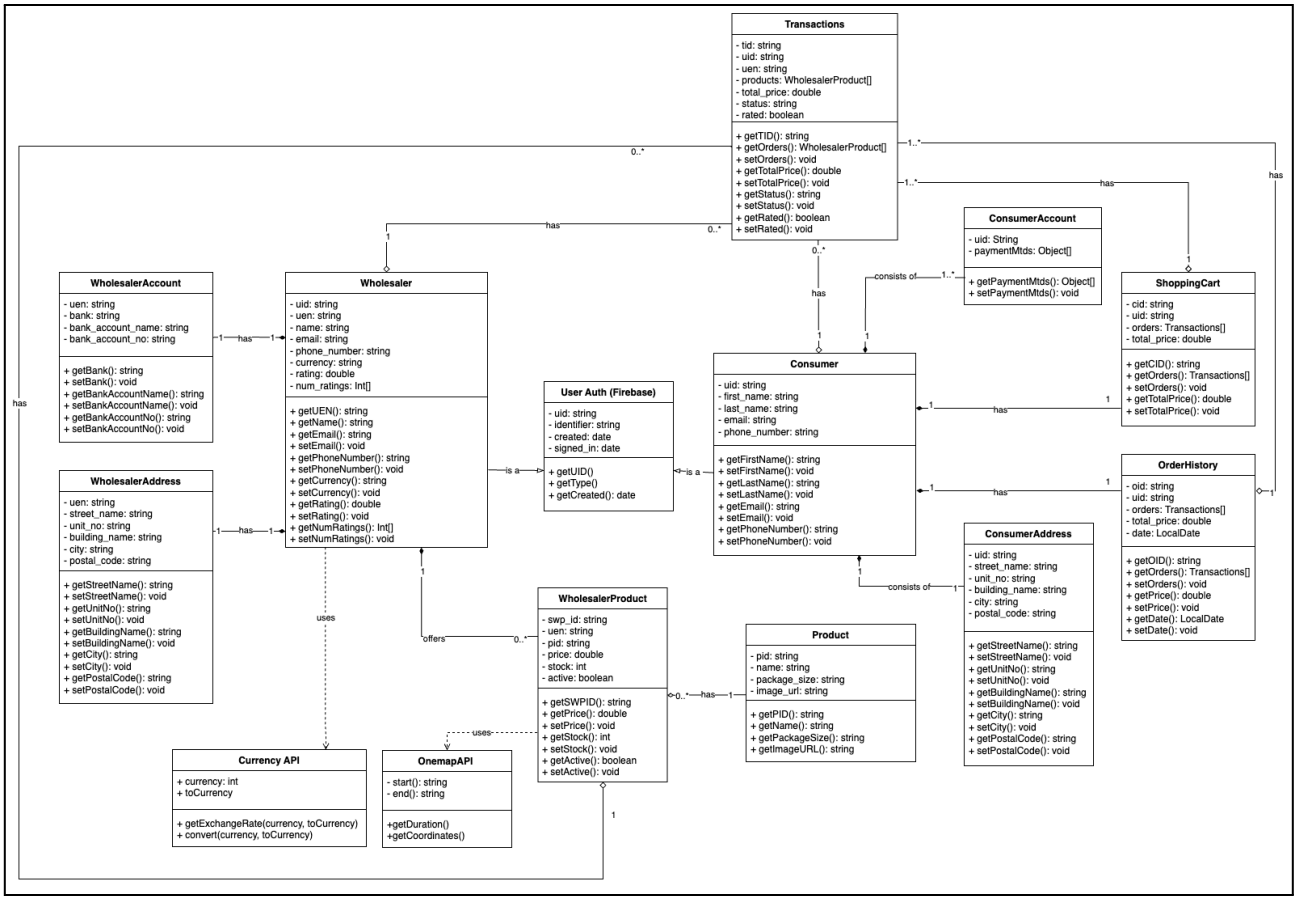
Consumer	A user who uses the ShopPeas platform to browse, select and purchase wholesale food products. Also known as “Customer”.
Product List	A dynamic list of products available to consumers for purchasing
Product Page	A detailed page for each product listed, with comprehensive information such as their description, prices, availability and wholesaler details
Wholesaler Details	Detailed information about a specific wholesaler
Location	Refers to the physical location of wholesalers.
Shopping cart	A virtual cart where consumers can store products they intend to purchase and proceed to checkout.
Checkout	The process whereby consumers review their shopping carts, enter payment information and finalise their purchase.
Ratings	Consumer's assessments on products they have purchased.
Wholesaler System	
Wholesaler	Located in either Singapore or Johor Bahru, wholesalers are users who use ShopPeas to sell their products.
Wholesaler Registration	The process by which businesses can apply to become wholesalers on ShopPeas

Appendix B: Analysis Models

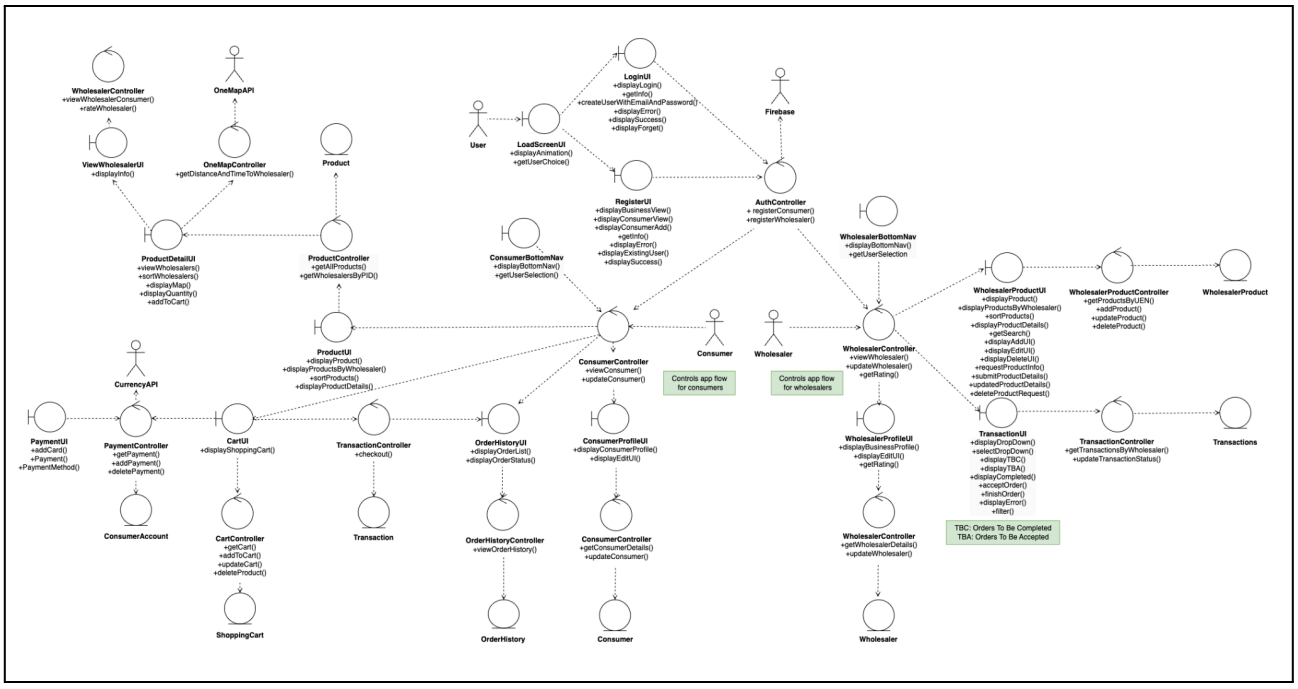
Please refer to our GitHub repository for a clearer view of the diagrams.



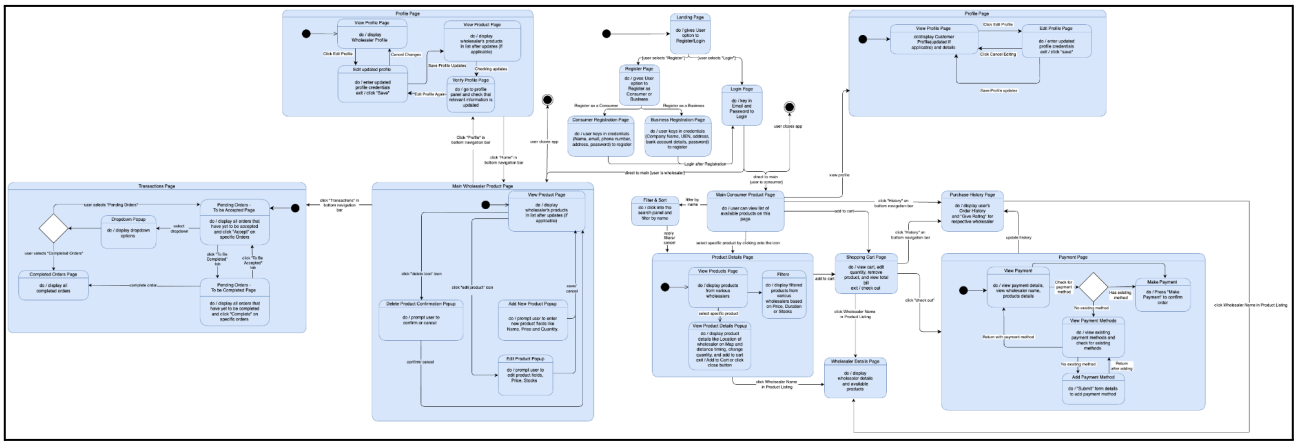
Use Case Diagram



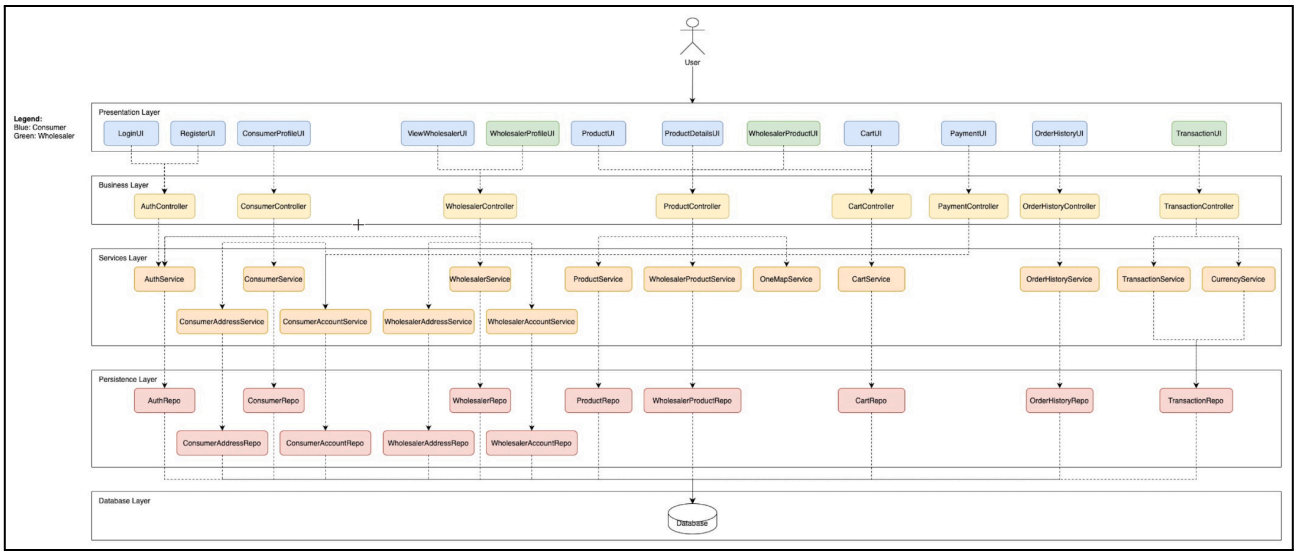
Entity Class Diagram



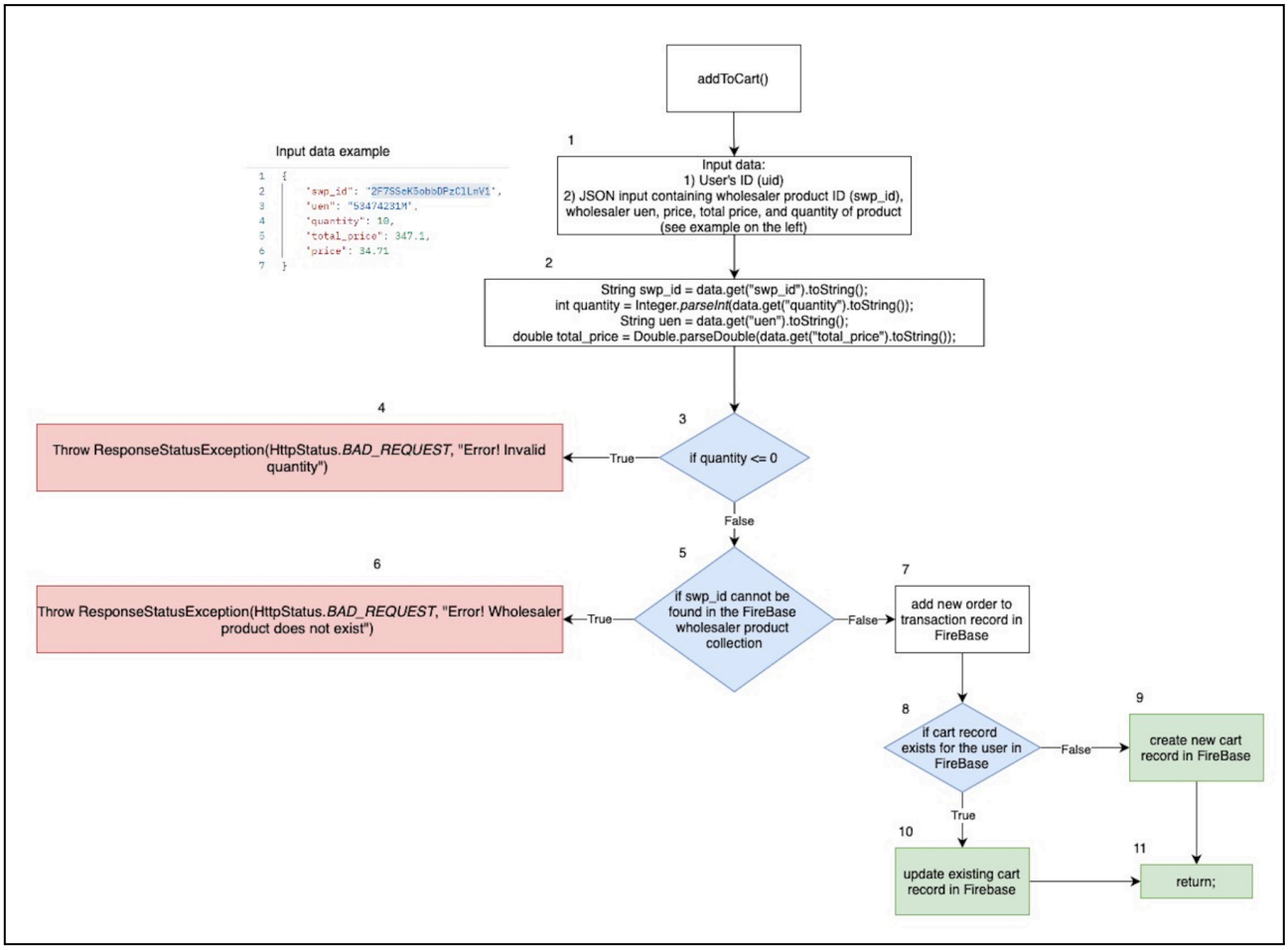
Controller Boundary Diagram



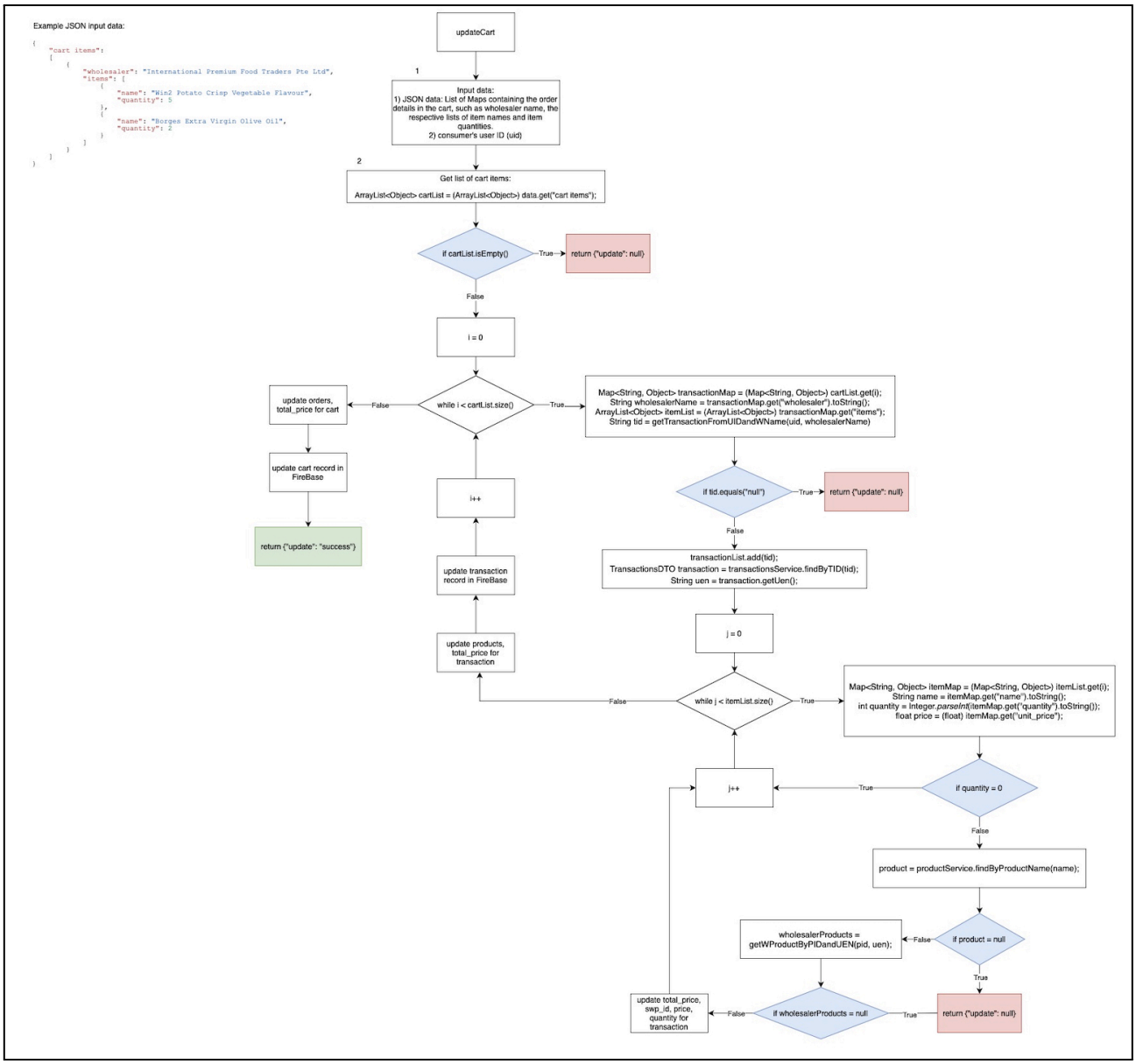
Dialog Map



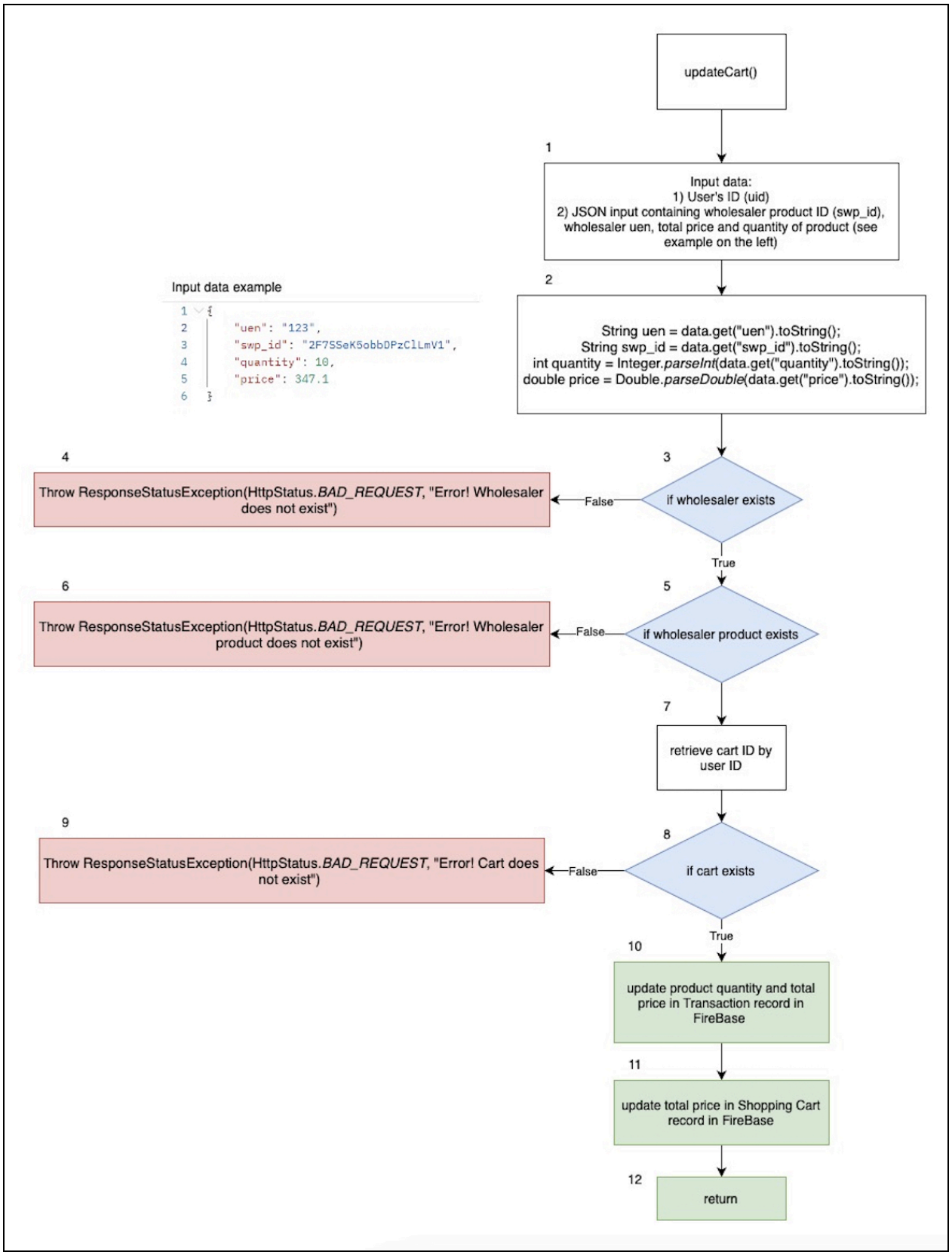
System Architecture Diagram



Basis Path Test for addToCart Testing



Basis Path Test for updateCart Testing



Basis Path Test for updateCart Testing (Detailed)