Software Requirements Specification

for

Hawker Stalk

Version 1.0 approved

Prepared by

Cho Zhi Wei
Choo Zhen Ming
Chow Weng Shi
Lai Xin Yee
Pham Nguyen Vu Hoang
Swaminathan Navitraa
Tan Ming Hao

NTU CCDS – SC2006 – Team *Hawker Stalk*

5th November 2024

Table of Content

Revision History

Name	Date	Reason For Changes	Version
Choo Zhen Ming	31/10/2024	Initial Document	1.0
Chow Weng Shi	1/11/2024	Added overall description	1.1
Cho Zhi Wei	5/11/2024	Revised class diagram and sequence diagram	1.2
Lai Xin Yee	8/11/2024	Added system features	1.3
Tan Ming Hao	9/11/2024	Finalize document	1.4

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to provide a detailed description of the functionalities, requirements, and design specifications for *Hawker Stalk*. *Hawker Stalk* is an application aimed at enhancing the discovery and connection between hawkers and potential customers. This document outlines the essential features, performance requirements, and user interactions necessary for developing a robust and user-friendly platform that will streamline the process of locating, interacting with, and supporting local hawkers. This SRS serves as a foundation for developers, designers, and stakeholders to ensure a shared understanding of the application's objectives and deliverables.

1.2 Document Conventions

The SRS follows the following documentation conventions to ensure clarity and consistency.

- Font Styles: "Arial" for the body text, "Times New Roman" for headers
- Font Size: 18 for main header, 14 for sub headers, 11 for content
- Highlighting: **Bold** for key terms, *italics* for emphasis
- Requirement Prioritization: Each requirement statement is explicitly labeled with priority level (High, Medium, Low) in its respective use case description. Higher-level requirements have inherited priority levels, cascading down to detailed requirements.
- Requirement Identification: Each requirement is uniquely identified for easy reference.
- Numbering: All requirements are numbered for easy reference, with the format of **a.b.** where:
  - **x** represents the requirement number.
  - **y** represents the sub requirement number.
  - For example, 1.1 will be the first sub requirement of the first requirement.

1.3 Intended Audience and Reading Suggestions

This Software Requirement Specification (SRS) document for *Hawker Stalk* is intended for the following stakeholders:

- **Project Managers**: To oversee the development process, ensure alignment with project goals, and manage timelines based on requirements. *Projects managers* should focus on all sections starting from "Overall Description" to gain a high-level understanding of the app's purpose and essential functionalities.
- **Developers**: To use this document as a comprehensive guide for implementing the app's features, adhering to defined requirements and specifications. *Developers* should carefully review the "System Features" and "Product Functions" sections, which detail all necessary functionalities.
- **UI/UX Designers**: To understand functional requirements and design specifications that will shape the user interface and experience. *The designers* should refer to the "External

Interface Requirements" and "System Features" sections, as they outline the design expectations and user interaction flows.

- **Test Engineers**: To develop test cases and procedures aligned with the outlined functional and non-functional requirements. *Test engineers* should concentrate on the "System Features" and "Other Nonfunctional Requirements" sections, which will help in the formulation of test scenarios and quality assurance standards.

- **End-User Representatives**: To ensure that user needs and expectations are addressed, especially in terms of accessibility, usability, and feature relevance. *Representatives* should review the "Product Perspective", "Product Functions" and "Other Nonfunctional Requirements" sections to verify that the requirements align with users needs and expectations.

All readers are encouraged to review the entire document, as each section builds upon the information presented in the previous ones, offering a comprehensive understanding of *Hawker Stalk's* requirements.

1.4 Product Scope

Hawker Stalk is a webpage-based designed to connect hawker stall vendors with customers by offering a convenient platform for discovering nearby stalls and fostering connections between users and hawkers. The application enhances the visibility of hawker businesses, giving users an easy way to explore a variety of local food and merchandise options in their area. Vendors can list their available items on the platform, allowing customers to browse offering seamlessly. *Hawker Stalk* aligns with broader corporate goals of social responsibility and sustainable community development, supporting local vendors and enriching neighborhood economies.

1.5 References

UI Design Inspiration:
https://www.figma.com/community/file/996012879169900959/foodwagon-food-delivery-landing-template-by-themewagon

2. Overall Description

2.1 Product Perspective

Hawker Stalk is designed as a standalone web-based platform with the flexibility to integrate with additional applications and services in the future. Potential integrations include location-based services, such as Google Maps, to enhance location accuracy; social media platforms to boost vendor visibility; and payment gateways to enable direct transactions. The platform utilizes the official government database of registered hawker centers to ensure accurate address listings. In its initial version, however, *Hawker Stalk* focuses on providing an engaging, user-friendly platform for locating and connecting with nearby hawker stalls. A system architecture diagram below (*Figure 2.1*) illustrates the interconnections and external interfaces involved.

Figure 2.1: Architecture Diagram

Hawker Stalk requires a reliable internet connection and location permissions for optimal performance. The application will depend on third-party location services for accurate positioning and mapping. Future versions may consider additional integrations with payment and rating systems, although these are not included in the initial deployment.

2.2 Product Functions

2.2.1 Use Case Diagram

The Use Case diagram below (*Figure 2.2.1*) illustrates the main product functionalities offered by *Hawker Stalk*. There are five main types of users in *Hawker Stalk* – Hawker, Customer, MapSystem, Admin and Email API, each interacting with *Hawker Stalk* to access features specific to their roles.

Figure 2.2.1: Use Case Diagram

2.2.2 Major Product Functions

Hawker Stalk allows users to perform the following functions:

Authentication

1. Signup with Different Roles – Admin, Hawker, Customer
2. Sign Up using Email
3. Login
4. Send Confirmation Email when Sign Up

Hawker Functions

5. Set-up profile during Sign-up
6. View Fault Report
7. View Reviews
8. View Hawker Listing Profile
9. Update the Information of the Stall
10. Update the Opening Hours in 12-Hour Notation
11. Update the Operating Days
12. Update the Opening Status
13. Update the Food Menu
14. Along with the Names and Pictures of Dishes with their Corresponding Prices
15. Delete Account and Close the Stall Permanently
16. Map
17. Displays all the Registered and External API Public Hawkers

Customer Functions

18. View the Information of the Stalls
19. View the Name of the Stalls
20. Menu of the Stalls, along with the Names and Pictures of Dishes with their Corresponding Prices
21. View the Opening Status of the Stall
22. View the Opening Hours of the Stalls in 12-hr Notation
23. View the Reviews of the Stalls
24. View the Ratings of the Stalls
25. Submit Fault Report
26. Submit Reviews of the Stalls
27. Provide Reviews of the Stalls
28. Provide Ratings of the Stalls
29. Search for Desired Hawker Center
30. View Hawker Center on the Map

31. View Available Stalls at each Hawker Center on the Map

Map Functions

32. Show the Exact Location of the Hawker Centers
33. Detect the Current Exact Location of User
34. Display all the Registered Public Hawkers

Admin Functions

35. Approve or Reject Newly Created Hawker Account
36. View Customer and Hawker Account Details
37. Suspend Any Types of Accounts
38. Review and Approve Hawker’s License

Email API

39. Send Confirmation Email
40. Send Link of the Reset Password Website

Other Functions

41. Reset Password
42. View Terms and Conditions of Fault Report

2.2.3 Class Diagram

The Class Diagram (*Figure 2.2.3)* demonstrates the interactions among different classes, along with the main functionalities of *Hawker Stalk* can perform.

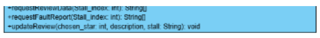

+requestReviewData(Stall_index: int): String[]
+requestFaultReport(Stall_index: int): String[]
+updateReview(chosen_star: int, description, stall: String): void

Figure 2.2.3: Class Diagram

2.3 User Classes and Characteristics

Users of the system are categorized into:

43. **Hawker**: Owner of a food stall, selling food in hawker centers. They need a more complex and functional interface.
44. **Customer**: Individuals who use this application to search for hawker centers to get food (i.e. tourists, Singapore residents).
45. **MapSystem**: Shows the exact location of all hawker centers in Singapore.
46. **Admin**: Individual who manages all types of users accounts (Hawker and Customer).
47. **Email API**: Manages account usability for new user sign-ups and password reset functions.

All five types of users are equally important for *Hawker Stalk,* forming a pipeline that ensures the *Hawker Stalk* functionality and operation.
E.g.: Hawkers update stall information and menus, and view fault reports and ratings from Customer. Customers search for hawker centers, view available stalls and menus, and submit fault reports and ratings. The MapSystem displays the correct locations of the hawker centers in Singapore, aiding all users (Customers and Hawkers). Admin ensures safety, privacy, and data protection for all types of users (Customers and Hawkers) while the Email API maintains account functionality, including new sign-ups and password resets.

2.4 Operating Environment

The platform will be a web-based application optimized for desktop, operating on the latest versions of Chrome, Firefox, and Microsoft Edge. It will be hosted on cloud infrastructure to maintain scalability and reliability, integrating external APIs (Google Map API and EmailJS API and database *storage* via Microsoft Azure SQL database. The dataset, sourced from the "List of Government Markets Hawker Centres" on data.gov.sg, will provide hawker center addresses for accurate mapping on the map.

2.5 Design and Implementation Constraints

The development of *Hawker Stalk* will be guided by several constraints below to maintain the consistency and design of the software, including:

2.5.1 Frontend Constraints

2.5.1.1 Framework and User Interface Formatting (Frontend)

The user interface will be developed using HTML and CSS to ensure a well-designed user interface.

48. **HTML**: Defines the content and structure of the webpage.
49. **CSS**: Handles the styling and layout design of the webpage.

2.5.2 Backend Constraints

2.5.2.1 Framework (Backend)

The backend server will use PHP and JavaScript for robust functionality.

50. **PHP**: Manages database connections and communication (requires setup)
51. **JavaScript**: Handles website logic and interactivity.

2.5.2.2 Database

The database will be an online SQL system on Microsoft Azure, enabling smooth collaboration with structured, relational data.

52. **SQL**: Used for robust and efficient database interactions.

2.5.3 Language Constraints

53. All software documentation, interfaces, and user interactions will be provided in English to ensure clarity and accessibility.
54. This choice is based on English being an international language, compatible with people all around the world, maintaining consistency across the platform.

2.6 User Documentation

Hawker Stalk will provide extensive user documentation to ensure developers have a clear understanding and be able to use the platform in an easy way. The documentation will include the following components:

2.6.1 README Files

A README file at the main repository will provide a project overview, including the details below:

- Brief Introduction of *Hawker Stalk*
- Project Setup Instructions with Code Requirements and Run Environment Setup
- Pre-Configured Users with Domain, Email and Password indicated
- Key Features including Authentication, Admin, Hawker and Customer
- Design Pattern that follows Model-View-Controller (MVC)
- Tech Stack of both Frontend and Backend
- External APIs used
- Project Contributors

2.6.2 Code Comments

The frontend user interface design code will include some key comments to ensure a better and clearer understanding of the code. Meanwhile, the backend code will include comments throughout the process to make it more human-readable and facilitate collaboration.

2.7 Assumptions and Dependencies

2.7.1 Assumptions

Assume that the platform is always connected to the Internet and that users have an email account before creating an account on our platform.

2.7.2 Dependencies

Hawker Stalk relies on the Google Maps API, Email API and dataset obtained from the “List of Government Markets Hawker Centres” in data.gov.sg to fetch data and ensure the functionality and availability of the webpage.

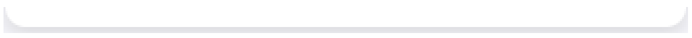
3. External Interface Requirements

3.1 User Interfaces

3.1.1 Authentication

3.1.1.1 Sign Up + Login Page

3.1.1.2 Reset Password Page (When Users Forgot their Password)



3.1.2 Admin-related Pages

3.1.2.1 Admin Home Page (View User Details + Approve, Suspend Users + View Hawker License)

hawker	hihawker@test.com	pending	View License	Approve Reject

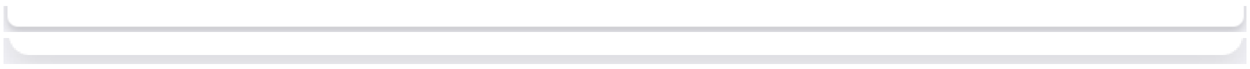
3.1.3 Customer-related Pages

3.1.3.1 Customer Home Page

3.1.3.2 Search Results (Stall Details + Stall Menu)

3.1.3.3 Reviews and Ratings (Other Customer's Reviews + Leaving Review)

3.1.3.4 Fault Reports (Reporting Issues + Terms & Conditions)



3.1.4 Hawker-related Pages

3.1.4.1 Hawker Main Page

3.1.4.2 Menu Page (Viewing + Editing + Adding Menu Items)

3.1.4.3 Opening Hours Page (Viewing + Editing Opening Dates & Time)

3.1.4.4 Stall Reviews (View Reviews + Ratings from Customers)

3.1.4.5 Stall Fault Reports (View + Delete Reported Faults by Customers)

3.1.4.6 Shutting Down Stall

3.1.5 Input Handling

3.1.5.1 Required Fields (Fields that Users Have to Fill in to Proceed)

3.1.5.2 Console Messages (Warning Messages Shown in Console)

3.2 Hardware Interfaces

Hawker Stalk is designed to operate as a web-based application, minimizing the need for direct hardware interfaces. The following outlines the logical as well as physical characteristics of the hardware and communication interfaces that support the platform's operations.

3.2.1 Device Types

Hawker Stalk is compatible with various devices, including desktop computers and laptops. Even though the application is operable on mobile devices, the user interface is not optimized for smaller screen sizes, thus certain elements may appear misaligned or difficult to interact with.

3.2.2 Data and Control Interactions

The user's device interacts with the web application through standard input and output hardware such as mouse and keyboard for input and displays for output. The user's device also communicates with the platform through network interface cards, utilizing Wi-Fi or Ethernet connections to access the internet.

3.2.3 Communication Protocols

The platform uses HTTP which currently operates on port 80 for web traffic as well as SQL Server which operates on port 1433 which allows encrypted access.

3.3 Software Interfaces

3.3.1 Operating Systems

Hawker Stalk is optimized for desktop and laptop operating systems, including Windows, macOS, and Linux. While the application may be accessible on mobile operating systems (iOS and Android), it is specifically designed for larger screens, such as those on desktops and laptops. Mobile usage may result in layout issues or misaligned elements.

3.3.2 Web Browsers

Our application supports the latest desktop versions of major web browsers that support HTML5, CSS3, and JavaScript, including:

- Google Chrome
- Mozilla Firefox
- Brave
- Internet Explorer
- Microsoft Edge

3.3.3 Database

Hawker Stalk uses an Azure SQL database to store and manage data related to hawkers, customers, hawker centers and food stalls. This database enables efficient data retrieval and secure management of user interactions.

3.3.4 Backend Tools

1. **PHP**
   - Data processing
   - CRUD operations (Create, Read, Update, Delete) with Azure SQL database

2. **XAMPP Control Panel**
   - Provides local testing environment

3.3.5 Frontend Tools

1. **HTML**
   - Content structuring

2. **CSS**
   - Provides interface styling to ensure a visually appealing experience

3. **JavaScript**
   - Enhances interactivity and responsiveness from frontend to backend
   - Handles dynamic content loading and API calls

3.3.6 API Used

1. **EmailJS API**
   - Used to send emails for automatic contact support purposes

2. **Google Map API**
   - Used to display locations of all hawker centers within Singapore

3.3.7 Data Communication

JSON format is used for data exchange between frontend and backend, ensuring consistent and structured communication. JavaScript gathers user input, formats them as JSON and sends them to PHP which then handles database interactions on Azure.

3.4 Communications Interfaces

Hawker Stalk utilizes various communication interfaces to facilitate efficient interactions between customers, hawkers and backend services:

- **HTTP Protocols**
  Data exchanges in *Hawker Stalk* currently occur over HTTP. This allows users to

interact with features like menu browsing, reviews and stall information updates. However, it does not provide data encryption. Transitioning to HTTPS in the future is considered for enhanced security and better data privacy for all users.

- **API Communication**
  Server-side operations are handled using PHP scripts hosted on XAMPP, which connects to Azure SQL database. JSON is used as the primary data format for communication between frontend and backend, facilitating structured data exchange for a more efficient processing.

- **Future Real-Time Updates**
  In the future, implementing WebSocket protocol could enable real-time updates, allowing users to instantly see changes made by hawkers without reloading the page. This allows immediate access to the latest data, thereby improving user experience.

- **Email Notifications**
  EmailJS API is integrated to handle automated email notifications, including account confirmation and password reset emails.

- **Interactive Map**
  Google Map API is used to display hawker center locations and real-time location of customers, allowing interaction with a map to view all available hawker centers. This provides a geographic context and helps customers locate stalls conveniently.

- **Network Protocols**
  Standard TCP/IP protocols support reliable data transmission. While *Hawker Stalk* is optimized for desktop and laptop use, these protocols ensure consistent network communication in line with established web standards.

These communication interfaces support *Hawker Stalk*'s core functionality, providing reliable interactions between frontend, backend and external APIs. Future updates can be implemented to improve real-time communication to make the platform more responsive and user-friendly.

4. System Features

This section will cover the core system features of *Hawker Stalk*.

4.1 Create a New Account (Customer)

4.1.1 Description and Priority

This feature allows customers to create their own account before starting to use our system. Upon the creation of a new account, the customer's domain, email address and password will be stored in the database. Any actions made by the customer, for example leaving reviews and ratings, filing fault reports will be using this created account as a reference.

Overall Priority: High
Description: A customer needs to create an account before proceeding to log in to use *Hawker Stalk.*

4.1.2 Stimulus/Response Sequences

Use Case ID:	01A
Use Case Name:	Create a new customer account

Created By:	Lai Xin Yee	Last Updated By:	Cho Zhi Wei
Date Created:	30-8-2024	Date Last Updated:	7-9-2024

Actor:	55. Customer 56. Email API
Description:	Creating a new customer account
Preconditions:	57. System must have a stable connection to the database. 58. The email address used must not already exist in the database.
Postconditions:	59. Show “Account created successfully” message. 60. Store data of customer in database
Priority:	High
Frequency of Use:	1-3 times per lifetime
Flow of Events:	61. Customer clicks on the sign-up button. 62. Customer enters a valid email address, password and password confirmation fields. 63. Customer click the create account button. 64. System validates the format of the email address. 65. System validates the validity of the email address in database 66. System validates the fulfillment of the requirement of the password. 67. System validates the password and password confirmation field are identical. 68. System displays “Thank you for signing up” message upon successful creation of a new customer account. 69. Customer ask the system to send a confirmation email to the email address.
Alternative Flows:	AF-S4: System detects the invalid email address format. 70. System displays error message “Invalid email format”. 71. System returns to Step 2. AF-S5: Email address exists in database: 72. System displays error message “Email exists, do you want to login?” 73. System returns to Step 2. AF-S6: System detects password does not fulfill the password requirements. 74. System displays error message “Invalid password”. 75. System returns to Step 2. AF-S7: System detects mismatch between password and password confirmation fields. 76. System displays error message “Password mismatch”. 77. System returns to Step 2.

Exceptions:	-
Includes:	78. Validate the account availability.
Special Requirements:	-
Assumptions:	79. Database refers to the system database.
Notes and Issues:	-

4.1.3 Functional Requirements

80. The users must be able to create an account of their respective domain.
81. The users can only create one account with one email address.
82. The users must be able to click on the sign-up button.
83. The customers must provide their valid email address and valid password to sign up for an account.
84. The system must verify all required fields have been filled out.
85. The input email address must be in a valid format.
86. The system must display an error message if the email address is not in a valid format.
87. The password must contain at least one upper case letter, one lower case letter, and one number.
88. The length of the password must be at least 8 characters.
89. The system must display an error message if the password does not match the requirement in 1.3.4. and 1.3.5..
90. The customers must input the same password twice as a confirmation stage.
91. The system must display an error message if the input passwords are not the same.
92. The system must send a confirmation email to the registrar upon successful registration of a new account.

4.2 Create a New Account (Hawker)

4.2.1 Description and Priority

This feature allows hawkers to create their own account before starting to use our system. Upon the creation of a new account, the hawker's domain, email address and password will be stored in the database. After the admin approves the creation of a hawker account, the hawker will then initialize their stall information and the opening hours, menus, stall name and location will also be stored in the database. Any actions made by the hawker, for example updating menu, opening hours, viewing fault reports, ratings and reviews will be using this created account as a reference.

Overall Priority: High
Description: A customer needs to create an account before proceeding to log in to use *Hawker Stalk.*

4.2.2 Stimulus/Response Sequences

Use Case ID:	01B		
Use Case Name:	Create a new hawker account		
Created By:	Lai Xin Yee	Last Updated By:	Cho Zhi Wei

Date Created:	30-8-2024	Date Last Updated:	7-9-2024

Actor:	93. Hawker 94. Admin 95. Email API
Description:	Creating a new hawker account
Preconditions:	96. The email address used must not already be in the database. 97. System must have a stable connection to the database.
Postconditions:	98. Show “pending” status upon submission of request for creating account. 99. Show “Account created successfully” message. 100. Store data of customer in database
Priority:	High
Frequency of Use:	1-3 times per lifetime
Flow of Events:	101. Hawker clicks on the sign-up button. 102. Hawker enters a valid email address, password and password confirmation fields. 103. Hawker uploads a valid operating license. 104. Hawker selects the create account button. 105. System validates the format of the email address. 106. System validates the validity of the email address in database 107. System validates the fulfillment of the requirement of the password. 108. System validates the password and password confirmation field are identical. 109. System display “pending” message. 110. Reviewer reviews the operating license and approves the account creation request. 111. System sends confirmation email to the registered email address.
Alternative Flows:	AF-S5: System detects invalid email address format. 112. System displays error message “Invalid email address format”. 113. System returns to Step 2. AF-S6: Email address exists in database: 114. System displays error message “Email exists, do you want to login?” 115. System returns to Step 2. AF-S7: System detects password does not fulfill the password requirements. 116. System displays error message “Invalid password”. 117. System returns to Step 2. AF-S8: System detects mismatch between password and password confirmation fields.

	118. System displays error message “Password mismatch”. 119. System returns to Step 2. AF-S10: Reviewer disapproves the account creation request. 120. System sends “License disapproved; account creation failed” notification. 121. System returns to Step 2.
Exceptions:	-
Includes:	122. Validate the account availability
Special Requirements:	-
Assumptions:	123. Database refers to the system database
Notes and Issues:	-

4.2.3 Functional Requirements

124. The users must be able to create an account of their respective domain.
125. The users can only create one account with one email address.
126. The users must be able to click on the sign-up button.
127. The hawkers must provide their valid email address, valid password and a valid operating license to sign up for an account.
128. The system must verify all required fields have been filled out.
129. The input email address must be in a valid format.
130. The system must display an error message if the email address is not in a valid format.
131. The password must contain at least one upper case letter, one lower case letter, and one number.
132. The length of the password must be at least 8 characters.
133. The system must display an error message if the password does not match the requirement in 1.3.4. and 1.3.5..
134. The hawkers must input the same password twice as a confirmation stage.
135. The system must display an error message if the input passwords are not the same.
136. Admin must be able to approve the validity of the license.
137. The system must automatically send a confirmation email to the registrar upon successful registration of a new account.

4.3 Login to Account

4.3.1 Description and Priority

This feature allows existing users to login to their account. Users are required to input their domain, email address and password in order to verify their credentials during the login process. User must login prior to accessing all the other features provided in the website.

Overall Priority: High
Description: A user must be verified with the existing database before *Hawker Stalk* can be launched.

4.3.2 Stimulus/Response Sequences

Use Case ID:	02A		
Use Case Name:	Login to account		
Created By:	Cho Zhi Wei	Last Updated By:	Tan Ming Hao
Date Created:	30-8-2024	Date Last Updated:	31-8-2024

Actor:	138. Customer 139. Hawker
Description:	Login to account using the registered email address and password
Preconditions:	140. User account must already exist in the database. 141. System must have a stable connection to the database. 142. User connects to the system.
Postconditions:	143. Users can see the main menu of the system. OR 144. Users see an error message. 145. Users can re-login to their accounts.
Priority:	High
Frequency of Use:	1 – 20 times per day
Flow of Events:	146. System requires the input of email address and password. 147. User input the email address and password in the login interface. 148. User clicks on the login button. 149. System verifies that email address and password have been filled out. 150. System retrieves the information from the database. 151. System verifies the email address and password with the information retrieved from the database. 152. If the email address and password are verified, the system displays the main menu to the user.
Alternative Flows:	AF-S5: If email address or password is not filled out. 153. System displays the message “Please fill out this field”. 154. System returns to Step 2. AF-S6: If the email address does not exist in the database. 155. System displays the message “Invalid email or password.”. 156. System returns to Step 2.

	AF-S7: If the email address and password does not match the information in the database. 157. System displays the message "Invalid email address or password.". 158. System returns to Step 2.
Exceptions:	-
Includes:	159. Verify Login Credentials.
Special Requirements:	-
Assumptions:	160. Database can be referred to System's database.
Notes and Issues:	-

4.3.3 Functional Requirements

161. The users must login to the system with the correct email address and password before performing all the tasks.
162. The system must verify that both fields have been filled out.
163. The system must display an error message if the login information is wrong.

4.4 Reset Password

4.4.1 Description and Priority

This feature allows users to reset their passwords when they forgot it. The system will send an email to the registered email address that consists of a link that direct users to the reset password page. The new password will then be stored into the database and the user can login to their account using the newly created password.

Overall Priority: Low
Description: Users will not need to use this feature, unless they forgot their password and need to reset a password in order to login to their account.

4.4.2 Stimulus/Response Sequences

Use Case ID:	02B		
Use Case Name:	Reset password when forgot		
Created By:	Lai Xin Yee	Last Updated By:	Tan Ming Hao
Date Created:	31-8-2024	Date Last Updated:	31-8-2024

Actor:	164. Customer 165. Hawker 166. Email API
Description:	To reset password when user forgot their password
Preconditions:	167. User account must already exist in the database.

	168. System must have a stable connection to the database.
Postconditions:	169. Users can re-login to their accounts.
Priority:	Low
Frequency of Use:	1-5 times per lifetime
Flow of Events:	170. User clicks on the “forgot password" button at the login page. 171. System requests user to enter the email address used to register for the account. 172. System verifies that the email address exists in the database. 173. System sends the user a link to the reset password website through email. 174. User clicks on the reset password website link. 175. User inputs a new valid password and password confirmation fields. 176. System directs the user back to the main page.
Alternative Flows:	AF-S3: Email address does not exist in the database. 177. System shows message “Invalid email address”. 178. System returns to Step 2. AF-S6: System detects password does not fulfill the password requirements. 179. System displays error message “Invalid password”. 180. System returns to Step 6. AF-S6: System detects mismatch between password and password confirmation fields. 181. System displays error message “Password mismatch”. 182. System returns to Step 6.
Exceptions:	-
Includes:	183. Verify Login Credentials.
Special Requirements:	-
Assumptions:	184. Database can be referred to System’s database.
Notes and Issues:	-

4.4.3 Functional Requirements

185. The users will be able to reset their password if they forgot their password by clicking on the “Forgot password?” button.
186. The system will send a link to the reset password website through email to the user after clicking on the “Forgot password?” button.

187. The password must contain at least one upper case letter, one lower case letter, one symbol and one number.
188. The length of the password must be at least 8 characters.
189. The users must input the same password twice as a confirmation stage.
190. The system must display an error message if the password does not match the requirement in 1.1.1. and 1.1.2.
191. The system must display an error message if the input passwords are not the same.

4.5 Update Stall Information (Hawker)

4.5.1 Description and Priority

This feature allows hawkers to update their stall information, including menu, opening hours, and also check their stall reviews from customers and if there are any fault reports. The updated data will be renewed and stored in our database.

Overall Priority: Medium
Description: Updating the stall information and ensuring that the information is accurate is important. However, it is not likely for the stall to change menus or opening hours frequently. Therefore, this feature will only be used when necessary.

4.5.2 Stimulus/Response Sequences

Use Case ID:	03A		
Use Case Name:	Hawkers update the information of the stall		
Created By:	Tan Ming Hao	Last Updated By:	Lai Xin Yee
Date Created:	30-8-2024	Date Last Updated:	30-10-2024

Actor:	192. Hawkers 193. Customers
Description:	Hawkers update information of their respective stall
Preconditions:	194. System must have a stable connection to the database. 195. Hawker account must exist in the database. 196. Hawkers want to update the information of their respective stall after seeing its information. OR 197. Hawkers must update the information of their respective stall after seeing a fault report from the customers.
Postconditions:	The system will update the information about the stalls.
Priority:	Medium
Frequency of Use:	0 – 20 times per week
Flow of Events:	198. Hawker login to their respective account. 199. The system shows the "Fault Report" icon.

	200. The system shows the "Review" icon. 201. The system shows the "Menu" icon. 202. The system shows the "Opening hours" icon. 203. The system shows "Open Shop" and "Close Shop" button. 204. The system shows "Shut Down Shop" button. 205. The system shows a red "X" icon. 206. System updates the information of the stall.
Alternative Flows:	AF-S2: The hawker clicks on the fault report icon. 207. The system displays all the reporters' fault reports. 208. The system will display an exit button and a button to delete the notification. 209. The system will remove the report if the customer clicks on the button to delete the notification. 210. The system returns to Step 3 upon exit. AF-S3: The hawker clicks on the review icon. 211. The system displays all the newly added reviews. 212. The system displays an exit button. 213. The hawker clicks on the exit button. 214. The system returns to Step 4 upon exit. AF-S4: The hawker clicks on the "Menu" button. 215. The system will show a page for the hawker to edit the menu. 216. The hawker edits the menu of the stalls. 217. The hawker clicks the save button. 218. The system returns to Step 5 upon exit. AF-S5: The hawker clicks on the "Opening hours" button. 219. The system will show a page for the hawker to edit the opening hours and operating days. 220. Hawker edits the opening time and operating days. 221. Hawker clicks on the "Save" button. 222. The system returns to Step 6 upon exit.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	223. Information of the hawker's stalls can be referred to System's database.
Notes and Issues:	-

4.5.3 Functional Requirements

224. The hawkers must be able to update the information related to the stall.

225. The hawkers must be able to update the opening hours and operating days of the stalls.
226. The system must display the opening hours in 12-hour notation with AM and PM.
227. The hawkers must be able to update the menu of the stalls.
228. The hawkers must be able to see the ratings of the stalls.
229. The ratings should be displayed in number of stars.
230. The hawkers must be able to see the reviews of the stalls.
231. The hawkers must be able to see any fault reports submitted by the customers.
232. The hawkers must be able to delete the fault reports after viewing.

4.6 Add or Delete Menu Items (Hawker)

4.6.1 Description and Priority

This feature allows hawkers to add new items to their menu or delete items from the menu. The data of the newly added items, including the photo, item name, price and description will be stored into the database. The deleted items will be removed from the database and cannot be seen by other users in the stall menu anymore.

Overall Priority: Medium
Description: Making sure that the menu items are up to date is important. However, it is not likely for hawkers to make changes to their menus frequently.

4.6.2 Stimulus/Response Sequences

Use Case ID:	03B		
Use Case Name:	Hawkers update the menu of the stalls		
Created By:	Tan Ming Hao	Last Updated By:	Tan Ming Hao
Date Created:	2-9-2024	Date Last Updated:	2-9-2024

Actor:	233. Hawkers
Description:	Hawkers update the menu of the stalls of their respective stalls
Preconditions:	234. System must have a stable connection to the database. 235. Hawker account must exist in the database. 236. Hawker clicks on the “Menu” button.
Postconditions:	The system will update the menu about the stalls.
Priority:	Medium
Frequency of Use:	0-12 per year
Flow of Events:	237. The system shows a page which includes the menu of the shop. 238. The hawker clicks on the item at the right. 239. The system shows the detail of the item. 240. The system shows an “Add menu” button, “Delete” button, “Save” button and “Exit” button.

	241. The system returns to the home page of hawker.
Alternative Flows:	AF-S4: The hawker clicks on the “Add menu” button. 242. The systems show a new row for the hawker to input a new menu item. 243. The hawker fills in the details of the new item which include picture, name, description and price. 244. The hawker clicks on the “Save” 245. The system adds the item to the database of the store. 246. The system returns to Step 4 upon exit. AF-S4: The hawker clicks on the “Delete” button. 247. The systems remove the item. 248. The hawker clicks on the “Save” button. 249. The system updates the details of the item in the menu. 250. The system returns to Step 4 upon exit. AF-S4: The hawker clicks on the “Exit” button. 251. The system returns to Step 5.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	252. The system will only show the “Edit” button if there is at least one item in the menu.
Notes and Issues:	-

4.6.3 Functional Requirements

253. The hawkers must be able to update the information related to the stall.
254. The hawkers must be able to update the menu of the stall.

4.7 Shut Down Stall (Hawker)

4.7.1 Description and Priority

In the event where the stall is no longer operating, hawkers can delete their stall from the system by deleting their account. By doing so, the information related to the stall will be deleted from the database and no longer be seen by other users.

Overall Priority: Medium
Description: This feature will only be accessed when the hawker decides to shut down their stall.

4.7.2 Stimulus/Response Sequences

Use Case ID:	03C

Use Case Name:	Hawkers remove their account and their stall.		
Created By:	Tan Ming Hao	Last Updated By:	Lai Xin Yee
Date Created:	2-9-2024	Date Last Updated:	30-10-2024

Actor:	255. Hawkers
Description:	Hawkers close their account and their stall.
Preconditions:	256. System must have a stable connection to the database. 257. Hawker account must exist in the database. 258. Hawkers click on the "Shut Down Shop" button.
Postconditions:	259. The system will remove the hawker's account and its information from the database. 260. The hawker account is removed successfully. 261. The hawker account no longer exists.
Priority:	Medium
Frequency of Use:	Once per lifetime
Flow of Events:	262. The system displays two text boxes and requires the hawker to enter their password correctly twice. 263. The hawker enters their password correctly twice. 264. The system verifies the correctness of the password. 265. The hawker clicks on the "Delete Account" button. 266. The system processes the request (delete account). 267. The system shows message ("Account delete successfully") 268. The system to Sign Up Page.
Alternative Flows:	AF-S5: If the email address and password does not match the information in the database. 269. The system displays the message "Password is incorrect". 270. The system returns to Step 1.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

4.7.3 Functional Requirements

271. The hawkers must be able to update the information related to the stall.
272. The hawkers must be able to close their respective accounts about their stalls.

273. The hawkers must input password two times correctly and click the “Confirm cancel” button to cancel their respective accounts.

4.8 View Stall Information (Customer)

4.8.1 Description and Priority

This feature allows customers to view specific information of a hawker stall, including the locations, opening hours, opening days, menu items, ratings and reviews. Customers can view these information by clicking on the desired hawker center and stall.

Overall Priority: High
Description: This is one of the main features that we are promoting, and customers will be using it frequently on our website.

4.8.2 Stimulus/Response Sequences

Use Case ID:	04A		
Use Case Name:	Customer can see the information of the stalls		
Created By:	Chow Weng Shi	Last Updated By:	Choo Zhen Ming
Date Created:	30-8-2024	Date Last Updated:	3-9-2024

Actor:	274. Customer 275. Map system
Description:	Customer can view the information of a list of stalls after logging in to the account
Preconditions:	276. Customer account must already login to the system. 277. System must have a stable connection to the database.
Postconditions:	278. The system displays a list of stalls with required information.
Priority:	High
Frequency of Use:	1 – 20 times per day
Flow of Events:	279. Customer logins to respective account successfully. 280. The system displays the location of customer and all hawker centers on the map. 281. The customer clicks on a desired hawker center. 282. The system retrieves the information of all stalls in the chosen hawker center from the database. 283. The system shows a list of stalls in the hawker center with name, opening status, opening hours in

	12-hour notation, ratings in number of stars, and a button to view the reviews.
Alternative Flows:	AF-S5: The customer clicks on the name of the desired stall. 284. The system displays the menu of the stall, with names and pictures of dishes and their corresponding prices. 285. The system returns to Step 6 upon exit. AF-S5: The customer clicks on the review button. 286. The system displays the overall ratings of the stall and reviews by other customers. 287. The system returns to Step 6 upon exit.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	288. Information about the hawker center can be referred to in the System's database.
Notes and Issues:	-

4.8.3 Functional Requirements

289. The customers must be able to see the information about the stalls.
290. The system must be able to show the location of the customer and hawker centers as a marker on the map.
291. The customers must be able to click on the hawker centre marker on the map.
292. The customers must be able to see the name of a list of stalls after clicking on the hawker center icon.
293. The customers must be able to see the opening hours and operating days of the stalls.
294. The system must display the opening hours in 12-hour notation with AM and PM.
295. The customers must be able to see the menu of the stalls after choosing the specific stall.
296. The customers must be able to see the ratings of the stalls.
297. The ratings should be displayed in number of stars.
298. The customers must be able to see the reviews of the stalls.
299. The customers must be able to click the “review” button under the stall information page.
300. The customers must see the name of the hawker center where the stall is located.
301. The customers must be able to see the exact address of the hawker center.

4.9 Submit Fault Report (Customer)

4.9.1 Description and Priority

This feature allows customers to submit a fault report to a specific stall when they found out that the information provided by the stall owner (hawker) is incorrect. Customers can also view the terms and conditions of filing a fault report before submitting it.

Overall Priority: Medium
Description: This is a sub-feature that helps us to improve the accuracy of data in *Hawker Stalk*. It is important but the frequency and likelihood of customers using it is not as high as viewing the information of stalls.

4.9.2 Stimulus/Response Sequences

Use Case ID:	05A		
Use Case Name:	Customer submits fault report		
Created By:	Cho Zhi Wei	Last Updated By:	Lai Xin Yee
Date Created:	30-8-2024	Date Last Updated:	3-9-2024

Actor:	302. Customer
Description:	Customer can submit fault report to notify the hawker if either of the opening hours, operating days, content or price in the menu is wrong
Preconditions:	303. Customer account must already exist in the database. 304. System must have a stable connection to the database.
Postconditions:	305. The record of the fault report has been stored in the database. 306. Hawker can see the submitted fault report.
Priority:	Medium
Frequency of Use:	0-5 times per day
Flow of Events:	307. Customer goes to the desired stall after searching for the information through the system. 308. Customer logins to respective account successfully. 309. Customer searches up the current hawker center and the specific stall. 310. The system displays the stall information and a report button. 311. The customer clicks on the report button. 312. The system displays “What issues have you found about the current stall?”, a text box below for the customers to elaborate as well as an information button for the terms and condition of filing a report. 313. The customer inputs the issues they found in the text box and submits the report. 314. The system displays “Fault report is submitted successfully.”. 315. The system returns to Step 4 upon exit.

Alternative Flows:	AF-S6: The customer submits the report without inputting at least 1 character in the text box. 316. The system displays message “The report must contain at least one character”. 317. The system returns to Step 7. AF-S6: The customer clicks on the information button. 318. The system displays a page with information about when to file a report, as well as the terms and conditions of filing a report. 319. The system returns to Step 6 upon exit.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	Customer realizes false information about certain stalls in the system.
Notes and Issues:	-

4.9.3 Functional Requirements

320. The customers must be able to make a fault report if there is any false information in the system.
321. The system must require the customer to input at least 1 character into the text box to submit the fault report.

4.10 Provide Reviews and Ratings (Customer)

4.10.1 Description and Priority

This feature allows customers to provide their reviews and ratings to a specific stall. Customers can access to this feature by clicking the “Review” button at the bottom of the stall information. The reviews and ratings submitted will be stored in our database together with the customer’s email address. It can also be viewed by the stall owner (hawker) and other customers that are using the website.

Overall Priority: Medium
Description: This is a sub-feature that allows customers to share their opinions on food in *Hawker Stalk*. It is important but the frequency and likelihood of customers using it is not as high as viewing the information of stalls.

4.10.2 Stimulus/Response Sequences

Use Case ID:	06A		
Use Case Name:	Customer provides stall’s reviews and ratings		
Created By:	Lai Xin Yee	Last Updated By:	Chow Weng Shi
Date Created:	30-8-2024	Date Last Updated:	31-8-2024

Actor:	322. Customer
Description:	Customer provides reviews, and ratings of the stall they visited
Preconditions:	323. System must have a stable connection to the database. 324. Customer account must exist in the database.
Postconditions:	325. System shows “review/rating uploaded” message
Priority:	Medium
Frequency of Use:	1-2 times per day
Flow of Events:	326. Customer search and tap on the stall that they want to provide the review or rating. 327. Customer writes in their review and choose the number of stars for ratings in the review or rating section. 328. Customer taps the “Upload” button to upload the review or rating. 329. System shows “review/rating uploaded” message.
Alternative Flows:	AF–S2: Customer does not write in review and choose number of stars for ratings. 330. Customer only choose the number of stars for ratings and do not input any characters into the text box. 331. System returns to Step 3.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

4.10.3 Functional Requirements

332. The customers must be able to provide their own reviews and ratings of the desired stall.
333. The system must be able to display the options of providing ratings and reviews to the customers in the same section.
334. The system must be able to display 5 empty stars on the screen when giving ratings.
335. The stars must be displayed in a horizontal straight line from left to right, with the leftmost star representing 1 star rating and the rightmost star representing 5 stars rating.
336. The customers must be able to click on the stars to select how many stars they want to rate the desired stall, with 0 stars being the lowest rating and 5 stars being the highest rating.
337. The system must automatically fill in all the empty stars prior to the star chosen by the customers, including the chosen star, with yellow color.

338. The system must be able to display a text box for the customers to input their reviews in.
339. The system must display only the ratings of the customers if they did not input any characters into the text box in their submitted reviews.
340. The system must display the customers' reviews below their ratings if they input at least 1 character into the text box in their submitted reviews.

4.11 Searches for Location (Customer)

4.11.1 Description and Priority

This feature allows customers to search for a desired location from the search bar. Relevant locations will appear in the search results and customers can click on them to view the hawker centers.

Overall Priority: High
Description: Customers are likely to search for hawker centers based on their current location in order to find the nearest hawker centers to get food.

4.11.2 Stimulus/Response Sequences

Use Case ID:	07A		
Use Case Name:	Customer searches for a location		
Created By:	Lai Xin Yee	Last Updated By:	Lai Xin Yee
Date Created:	30-8-2024	Date Last Updated:	30-8-2024

Actor:	341. Customer
Description:	To search for a desired location
Preconditions:	342. System is connected to Wi-Fi or Mobile Data.
Postconditions:	343. A list of relevant hawker centers will be displayed in the system.
Priority:	High
Frequency of Use:	0-15 times per day
Flow of Events:	344. Customer enters query in the search bar. 345. Customer taps the search button. 346. System provides a list of relevant search results based on the customer's query.
Alternative Flows:	-
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

4.11.3 Functional Requirements

347. The customers must be able to search the name of the stall for the location of the desired hawker center.
348. The customers input must contain at least 1 character but less than 512 characters for the search results.
349. The system must be able to show relevant results when the customers type in at least one character.
350. The system must be able to display the location of the hawker center on the map.

4.12 Suspends User's Account (Admin)

4.12.1 Description and Priority

This feature allows customers to create their own account before starting to use our system. Upon the creation of a new account, the user's domain, email address and password will be stored in the database. Any actions made by the user, for example leaving reviews and ratings, filing fault reports will be using this created account as a reference.

Overall Priority: High
Description: Admin have the highest privileges for accounts in the system.

4.12.2 Stimulus/Response Sequences

Use Case ID:	08A		
Use Case Name:	Admin suspends accounts of user		
Created By:	Cho Zhi Wei	Last Updated By:	Cho Zhi Wei
Date Created:	5-11-2024	Date Last Updated:	5-11-2024

Actor:	351. Admin
Description:	To suspend accounts of user
Preconditions:	352. System is connected to Wi-Fi or Mobile Data.
Postconditions:	353. Status of certain accounts change, the account cannot be used to login
Priority:	High
Frequency of Use:	0-5 times per day
Flow of Events:	354. Admin login to account 355. Admin click on "Suspend" button for the account 356. System asks admin to ensure to suspend the certain user 357. Admin click on "OK" 358. System changes the status of the user account to "suspended"
Alternative Flows:	AF-S3: Admin refuse to suspend the user account 359. Admin click on "Cancel" 360. System shows the user management page

Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

4.12.3 Functional Requirements

361. The admin must be able to manage all the accounts of user
362. The admin must be able to suspend account of certain user

4.13 Activates User's Account

4.12.4 Description and Priority

This feature allows customers to create their own account before starting to use our system. Upon the creation of a new account, the user's domain, email address and password will be stored in the database. Any actions made by the user, for example leaving reviews and ratings, filing fault reports will be using this created account as a reference.

Overall Priority: High
Description: Admin have the highest privileges for accounts in the system.

4.12.5 Stimulus/Response Sequences

Use Case ID:	08B		
Use Case Name:	Admin activates accounts of user		
Created By:	Cho Zhi Wei	Last Updated By:	Cho Zhi Wei
Date Created:	5-11-2024	Date Last Updated:	5-11-2024

Actor:	363. Admin
Description:	To activate accounts of user
Preconditions:	364. System is connected to Wi-Fi or Mobile Data.
Postconditions:	365. Status of certain accounts change, the account can be used to login
Priority:	High
Frequency of Use:	0-5 times per day
Flow of Events:	366. Admin login to account 367. Admin click on “Activate” button for the account 368. System asks admin to ensure to activate the certain user 369. Admin click on “OK” 370. System changes the status of the user account to “active”

Alternative Flows:	AF-S3: Admin refuse to activate the user 371. Admin click on “Cancel” 372. System shows the user management page
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

4.12.6 Functional Requirements

373. The admin must be able to manage all the accounts of user
374. The admin must be able to activate account of certain user

5. Other Nonfunctional Requirements

5.1 Performance Requirements

1) Each page should load within 2 seconds, including login, authentication and searching hawker stalls.
2) The web application should be scalable to support up to 30,000 concurrent users while maintaining optimal performance.
3) The web application should have high availability with 99.9% uptime, allowing users to access it reliability at any time of day.
4) The web application should automatically retry failed network requests up to 3 times to ensure a seamless user experience.

5.2 Safety Requirements

1) The application should display terms and conditions to file a fault report. This ensures guidelines for respectful conduct to prevent hawkers from harassment or abuse.
2) The system should allow the admin to suspend a customer or hawker account if they breach *Hawker Stalk's* code of conduct.
3) The application should not require all types of users to share personal information with other users, to ensure the safety and privacy of all users.

5.3 Security Requirements

1) All user data, which includes personal information and location data, should be encrypted both in transit and at rest in the database.
2) All user data should be kept confidential and should not be disclosed to other users of the app or to the public. By complying with the PDPA law on data-privacy regulation, user rights can be protected.
3) The system should enforce strong password policies that have minimum length requirements and necessitate a mix of characters.
4) The system should restrict API calls and database access to authorized sources only.

5.4 Software Quality Attributes

1) All code should be well-explained with comments. These comments should minimally explain the purpose of the code block to allow readers to interpret, reuse and maintain the code.
2) The application should perform consistently and be available when users need it, with a downtime of not more than 1%, to ensure reliability.
3) The application should be easy to update, debug, and enhance. This is achieved through a modular architecture, like MVC, to make specific parts of the system independently modifiable.
4) The application should be easy-to-use and understand for all user types: hawkers, customers and admins – with user-friendly interfaces, intuitive navigation and clear instructions.
5) The backend server's API endpoints must always include documentation detailing the endpoint URL, required request body, and expected response body.

5.5 Business Rules

The business rules establish essential functionality for the *Hawker Stalk* app, ensuring that it operates within appropriate legal and ethical guidelines.

5.5.1 Review and Rating System

375. Customers may only leave a review for a hawker after a completed transaction.
376. Reviews must follow community guidelines, and any inappropriate content may be removed.

5.5.2 Hawker Sign-up

377. Hawkers must upload their hawker license certificate to ensure legitimacy.
378. The admin must review the Hawker details and license as an additional layer to ensure the legitimacy of all hawkers present on the platform.

5.5.3 Admin User-Management Dashboard

379. The admin can view personal details and activity of all users on the app. When a customer or hawker violates community guidelines, the admin can suspend their account.

5.5.4 Function Restrictions

Each type of user in *Hawker Stalk* can perform different functions, as specified in table 5.5.4 below.

Actor	Fault Reports	Reviews & Ratings	Stall Information

Hawker	getFaultReports()
	ResolveFaultReport()
		getReviews()
		getRatings()
		respondToReview()
			setMenu()
			setOpeningHours()
			
			updateMenu()
			updateOpeningHours()

			openShop()
			closeShop()
			shutDownShop()

Customer	viewFaultReportTerms(
	)
	createFaultReport()
	submitFaultReport()
		viewReviews()
		leaveReview()
		submitReview()
		submitRating()
			viewMenu()
			viewOpeningHours()
			
			findStall()
			findHawkerCentre()

Admin	ViewFaultReports()	viewReviews()
		viewRatings()
			viewMenu()
			viewOpeningHours()

Table 5.5.4

6. Other Requirements

6.1 Internationalization Requirements

380. In the first version, *Hawker Stalk* will only support the English language. Future releases will consider multi-language support such as Chinese and Malay, to cater to a broader audience in Singapore, especially senior citizens who cannot understand English.

381. Date and time representations should be in the Singapore Time Zone (GMT +8).

6.2 Legal Requirements

382. Adherence to local food safety rules.

383. Compliance with PDPA when handling user's sensitive data.

6.3 Reuse Objectives

384. Develop a modular and reusable code structure to reduce redundancy, simplify maintenance, and enhance scalability for future updates or additional features.

1. Design external integrations to enable flexibility for adding, modifying, or replacing services without extensive code rewriting.

6.4 Accessibility Requirements

385. Avoid using content that flashed, blinks, or moves rapidly, as it can be distracting and may trigger seizures in users with photosensitive epilepsy. Animations, if used, should be smooth and subtle.

386. Avoid imposing strict time limits on tasks. If time limits are necessary, provide an option to extend them to accommodate users who may need more time.

Appendix A: Glossary

Acronym / Abbreviation	Meaning
API	Application Programming Interface A set of protocols for building and interacting with software applications (e.g. Google Map API)
PDPA	Personal Data Protection Act Provide a baseline standard of protection for personal data in Singapore
SQL	Structured Query Language A domain-specific language used in programming for managing relational databases.
Frontend	The “client side” of an application that users interact with directly It consists of everything the user experiences directly Examples: Text, Images, Sliders, Buttons, etc.
Backend	The “server side” that manage and process data behind the scenes Include server, database, and application logic
SRS	Software Requirements Specification A document that describes what the software will do and how it will be expected to perform. It includes a set of use cases that describe all the interactions the users will have with the software

Appendix B: Analysis Models

Architecture Diagram

+requestReviewData(Stall_index: int): String[]
+requestFaultReport(Stall_index: int): String[]
+updateReview(chosen_star: int, description, stall: String): void

Class Diagram

Dialog Map

Appendix C: To Be Determined List

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

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