

# SC2006 Software Engineering

## Lab3 Deliverables

Group 26

**Hawker Stalk**

# **Table of Contents**

Target users	3
Functional Requirements	3
Non-functional Requirements	7
Data dictionary	8
Use Case Diagram	9
Use Case Description	10
UI Mockup Diagram	31
Sequence Diagrams	36
Dialog Map	49
System Architecture	50
Application Skeleton	51
Design Pattern and Considerations	52

## Target users

- Residents in Singapore
- Tourists to Singapore
- Hawkers in Singapore

## Functional Requirements

1. The users and hawkers must be able to create an account of their respective domain.
  - 1.1. The users and hawkers can only create one account with one email address.
  - 1.2. The users and hawkers must be able to click on the sign-up button.
  - 1.3. The users must provide their valid email address and valid password to sign up for an account.
    - 1.3.1. The system must verify all required fields have been filled out.
    - 1.3.2. The input email address must be in a valid format.
    - 1.3.3. The system must display an error message if the email address is not in a valid format.
    - 1.3.4. The password must contain at least one upper case letter, one lower case letter, **one special character** and one number.
    - 1.3.5. The length of the password must be at least 8 characters.
    - 1.3.6. The system must display an error message if the password does not match the requirement in 1.3.4. and 1.3.5..
    - 1.3.7. The users must input the same password twice as a confirmation stage.
    - 1.3.8. The system must display an error message if the input passwords are not the same.
    - 1.3.9. The system must automatically send a confirmation email to the registrar upon successful registration of a new account.
  - 1.4. The hawkers must provide their valid email address, valid password and a valid operating license to sign up for an account.
    - 1.4.1. The system must verify all required fields have been filled out.
    - 1.4.2. The input email address must be in a valid format.
    - 1.4.3. The system must display an error message if the email address is not in a valid format.

- 1.4.4. The password must contain at least one upper case letter, one lower case letter, one **special character** and one number.
- 1.4.5. The length of the password must be at least 8 characters.
- 1.4.6. The system must display an error message if the password does not match the requirement in 1.4.4. and 1.4.5..
- 1.4.7. The hawkers must input the same password twice as a confirmation stage.
- 1.4.8. The system must display an error message if the input passwords are not the same.
- 1.4.9. Reviewer must approve the validity of the license.
- 1.4.10. The system must automatically send a confirmation email to the registrar upon successful registration of a new account.
- 2. The users and hawkers must login to the system with the correct email address and password before performing all the tasks.
  - 2.1. The system must verify that both fields have been filled out.
  - 2.2. The system must display an error message if the login information is wrong.
  - 2.3. The user will be able to reset their password if they forgot their password by clicking on the “Forgot password?” button.
    - 2.3.1. The system will send a link to the reset password website through email to the user after clicking on the “Forgot password?” button.
      - 2.3.1.1. The password must contain at least one upper case letter, one lower case letter, one symbol and one number.
      - 2.3.1.2. The length of the password must be at least 12 characters.
      - 2.3.1.3. The users must input the same password twice as a confirmation stage.
      - 2.3.1.4. The system must display an error message if the password does not match the requirement in 2.3.1.1. and 2.3.1.2.
      - 2.3.1.5. The system must display an error message if the input passwords are not the same.
      - 2.3.1.6. The system must automatically send a confirmation email to the user upon successful reset of their password.
- 3. The hawkers must be able to update the information related to the stall.
  - 3.1. The hawkers must be able to update the opening hours and operating days of the stalls.

- 3.1.1. The system must display the opening hours in 12-hour notation with AM and PM.
  - 3.2. The hawkers must be able to update the menu of the stalls.
  - 3.3. The hawkers must be able to see the ratings of the stalls.
    - 3.3.1. The ratings should be displayed in number.
  - 3.4. The hawkers must be able to see the reviews of the stalls.
  - 3.5. The hawkers must be able to see any fault reports submitted by the users.
  - 3.6. The hawkers must be able to update their opening status at any time.
  - 3.7. The hawkers must be able to close their respective accounts about their stalls.
    - 3.7.1. The hawkers must input password two times correctly and click the “Confirm cancel” button to cancel their respective accounts.
- 4. The users must be able to see the information about the stalls.
  - 4.1. The users must be able to see the hawker centers nearby.
    - 4.1.1. The system must be able to show the hawker centers in the icon on the map.
    - 4.1.2. The user must be able to click on the hawker center icon on the map.
  - 4.2. The users must be able to see the name of the stalls after clicking the hawker center icon.
  - 4.3. The users must be able to see the opening hours and operating days of the stalls.
    - 4.3.1. The system must display the opening hours in 12-hour notation with AM and PM.
  - 4.4. The users must be able to see the menu of the stalls.
  - 4.5. The users must be able to see the ratings of the stalls.
    - 4.5.1. The ratings should be displayed in number.
  - 4.6. The users must be able to see the reviews of the stalls.
    - 4.6.1. The users must be able to click the “review” button under the stall information page.
  - 4.7. The users must be able to see the crowd situation of the stalls.
  - 4.8. The users must be able to see the opening status of the stalls.
  - 4.9. The users must see the name of the hawker center where the stall is located.
  - 4.10. The users must be able to see the exact address of the hawker center.
- 5. The users must be able to make a fault report if there is any false information in the system.

- 5.1. The system must require the user to input at least 1 character into the text box to submit the fault report.
- 6. The users must be able to view the real-time crowd level of a specific stall.
  - 6.1. The system must be able to display 5 small human icons horizontally on the screen to show the crowd situation.
  - 6.2. The users must be able to update the crowd situation after clicking the update button.
    - 6.2.1. The user must be able to click on the small human icons to provide the crowd situation of the stalls.
    - 6.2.2. The system must automatically fill up the human icons prior to the icon chosen by the user, including the chosen icon, with red color.
- 7. The users must be able to provide their own reviews and ratings of the desired stall.
  - 7.1. The system must be able to display the options of providing ratings and reviews to the user in the same section.
    - 7.1.1. The system must lock the option of providing reviews to prevent the user from giving reviews.
    - 7.1.2. The system must require the user to provide the ratings before unlocking the review section for the user to access freely.
    - 7.1.3. The user must be able to submit their ratings without having to provide their reviews.
  - 7.2. The system must be able to display 5 empty stars on the screen when giving ratings.
    - 7.2.1. 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.
    - 7.2.2. The user 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.
    - 7.2.3. The system must automatically fill in all the empty stars prior to the star chosen by the user, including the chosen star, with yellow color.
  - 7.3. The system must be able to display a text box for the users to input their reviews in.
    - 7.3.1. The system must display only the ratings of the users if they did not input any characters into the text box in their submitted reviews.
    - 7.3.2. The system must display the users' reviews below their ratings if they input at least 1 character into the text box in their submitted reviews.

8. The users must be able to search the name of the stall for the location of the desired hawker center.
  - 8.1. The user input must contain at least 1 character but less than 512 characters for the search results.
  - 8.2. The users must be able to see the distance between their current location and hawker center.
  - 8.3. The system must be able to show relevant results when the users type in at least one character.
  - 8.4. The system must be able to show the nearby hawker center of the user based on the user's current position.
  - 8.5. The system must be able to display the location of the hawker center on the map.
  - 8.6. The system must be able to filter the search results based on the user's requirements.
    - 8.6.1. The system must be able to filter search results based on the distance between the user and the hawker center.
    - 8.6.2. The system must be able to filter the search results based on ratings.
9. The system must be able to detect the real-time position of the users.
  - 9.1. The system must be able to show the current exact location of the users.
  - 9.2. The system must be able to direct the user to the correct location by using the map function.

## **Non-functional Requirements**

1. Security
  - 1.1. Personal and financial data (identification details, contact information, device information) of the stalls' owner should be encrypted to prevent unauthorized access.
  - 1.2. User data, including location information, search history, and personal details, should be safeguarded and not disclosed to unauthorized entities, ensuring compliance with relevant data protection regulations.
2. Reliability
  - 2.1. The system must be able to return accurate results for all hawker centers matching the user's search queries, including partial matches and variations.
  - 2.2. Information provided on the app, such as stall details should be up-to-date, accurate and consistent.

- 2.3. The search results should reflect real-time data, including hawkers' operating hours and the menu availability.
- 2.4. The system should reliably locate and display the user's accurate location on the system.
- 3. Performance
  - 3.1. The search results should be reflected with minimal response time for users.
- 4. Scalability
  - 4.1. The system should be designed to scale efficiently, accommodating an increasing number of users as the platform grows without compromising performance.
  - 4.2. The system should be able to include and manage the growing volume of data, including new stalls without much performance degradation.
- 5. Usability
  - 5.1. The system should be usable by being intuitive and simple design.
  - 5.2. 80% of first-time users must be able to enter a simple search query within 2 minutes of starting to use the system.

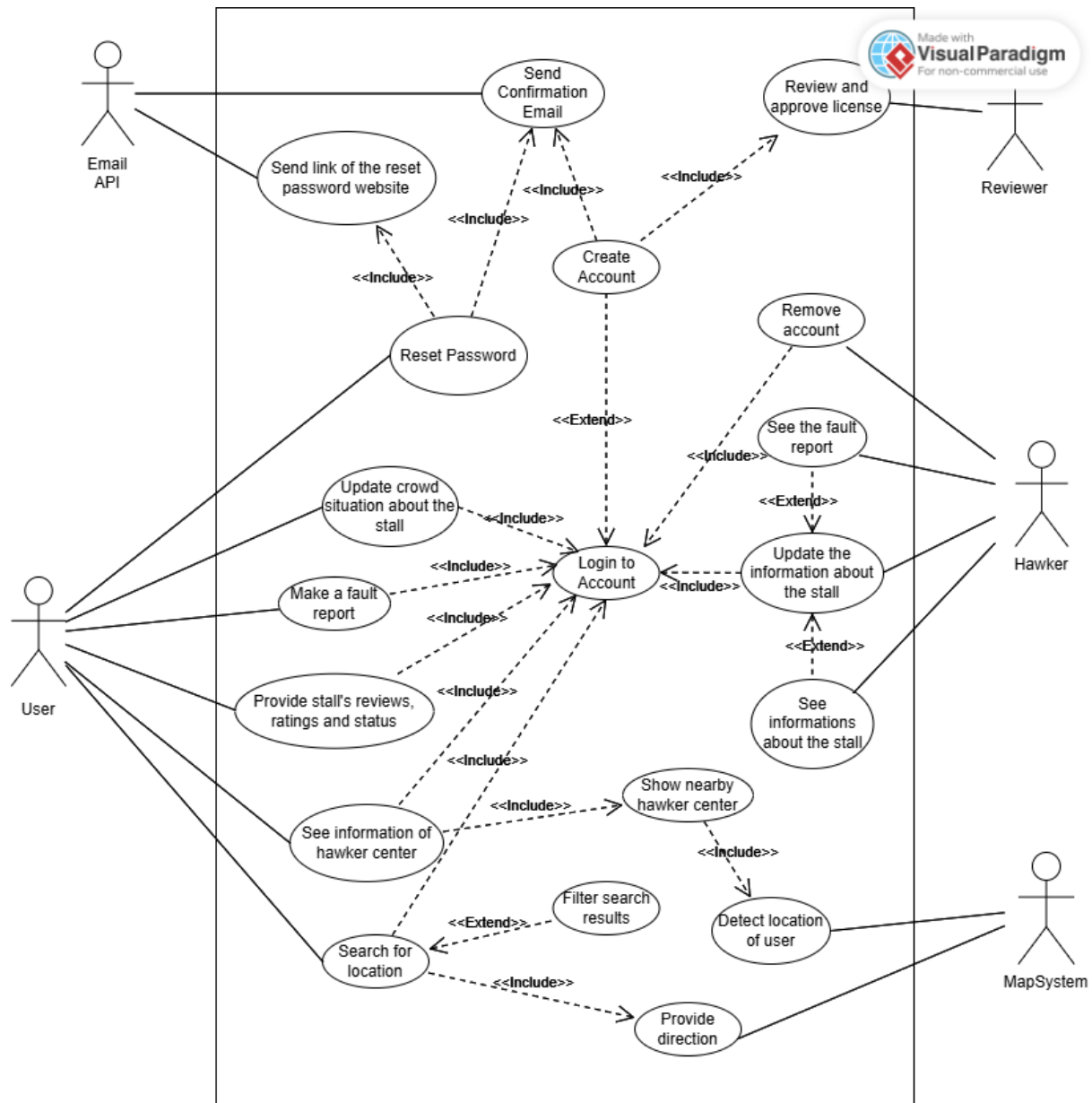
## Data dictionary

Term	Definition
Users	All the target users who are using this application to search for hawker centers to get food (i.e. tourists, Singapore residents)
System	The application
Hawker center	The place that consists of stalls that sell food
Hawker	The owner of the stall
Stalls	The unit selling food in a hawker center
Menu	A list of information showing the name, details, and the price of the food
Ratings	A numerical score given by the users to show the satisfaction to the stall
Reviews	Comments given by the users
Empty	Indicate the icons that are filled with white color, indicating their non-chosen status



# Use Case Diagram

If the picture is unclear, please refer to the picture attached in the same file



# Use Case Description

Use Case ID:	01A		
Use Case Name:	Create a new user account		
Created By:	Lai Xin Yee	Last Updated By:	Cho Zhi Wei
Date Created:	30-8-2024	Date Last Updated:	7-9-2024

Actor:	<ul style="list-style-type: none"><li>• User</li><li>• Email API</li></ul>
Description:	Creating a new user account
Preconditions:	<ul style="list-style-type: none"><li>• System must have a stable connection to the database.</li><li>• The email address used must not already exist in the database.</li></ul>
Postconditions:	<ul style="list-style-type: none"><li>• Show “Account created successfully” message.</li></ul>
Priority:	High
Frequency of Use:	1-3 times per lifetime
Flow of Events:	<ol style="list-style-type: none"><li>1. User clicks on the sign-up button.</li><li>2. User enters a valid email address, password and password confirmation fields.</li><li>3. User click the create account button.</li><li>4. System validates the format of the email address.</li><li>5. System validates the validity of the email address in database</li><li>6. System validates the fulfillment of the requirement of the password.</li><li>7. System validates the password and password confirmation field are identical.</li><li>8. System displays “account created successfully” message upon successful creation of a new user account.</li></ol>

	9. System automatically sends confirmation email to the registered email address.
Alternative Flows:	<p>AF-S4: System detects the invalid email address format.</p> <ol style="list-style-type: none"> <li>1. System displays error message “Invalid email format”.</li> <li>2. System returns to Step 2.</li> </ol> <p>AF-S5: Email address exists in database:</p> <ol style="list-style-type: none"> <li>1. System displays error message “Email exists, do you want to login?”</li> <li>2. System returns to Step 2.</li> </ol> <p>AF-S6: System detects password does not fulfill the password requirements.</p> <ol style="list-style-type: none"> <li>1. System displays error message “Invalid password”.</li> <li>2. System returns to Step 2.</li> </ol> <p>AF-S7: System detects mismatch between password and password confirmation fields.</p> <ol style="list-style-type: none"> <li>1. System displays error message “Password mismatch”.</li> <li>2. System returns to Step 2.</li> </ol>
Exceptions:	-
Includes:	<ul style="list-style-type: none"> <li>• Validate the account availability.</li> </ul>
Special Requirements:	-
Assumptions:	<ul style="list-style-type: none"> <li>• Database refers to the system database.</li> </ul>
Notes and Issues:	-

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:	<ul style="list-style-type: none"> <li>• Hawker</li> <li>• Reviewer</li> <li>• Email API</li> </ul>
Description:	Creating a new hawker account
Preconditions:	<ul style="list-style-type: none"> <li>• The email address used must not already be in the database.</li> <li>• System must have a stable connection to the database.</li> </ul>
Postconditions:	<ul style="list-style-type: none"> <li>• Show “pending approval” status upon submission of request for creating account.</li> <li>• Send a notification once the account registration is approved.</li> </ul>
Priority:	High
Frequency of Use:	1-3 times per lifetime
Flow of Events:	<ol style="list-style-type: none"> <li>1. Hawker clicks on the sign-up button.</li> <li>2. Hawker enters a valid email address, password and password confirmation fields.</li> <li>3. Hawker uploads a valid operating license.</li> <li>4. Hawker selects the create account button.</li> <li>5. System validates the format of the email address.</li> <li>6. System validates the validity of the email address in database</li> <li>7. System validates the fulfillment of the requirement of the password.</li> <li>8. System validates the password and password confirmation field are identical.</li> <li>9. System display “pending approval” message.</li> </ol>

	<p>10. Reviewer reviews the operating license and approves the account creation request.</p> <p>11. System sends “account created successfully” notification upon successful creation of a new user account.</p> <p>12. System automatically sends confirmation email to the registered email address.</p>
Alternative Flows:	<p>AF-S5: System detects invalid email address format.</p> <ol style="list-style-type: none"> <li>1. System displays error message “Invalid email address format”.</li> <li>2. System returns to Step 2.</li> </ol> <p>AF-S6: Email address exists in database:</p> <ol style="list-style-type: none"> <li>3. System displays error message “Email exists, do you want to login?”</li> <li>4. System returns to Step 2.</li> </ol> <p>AF-S7: System detects password does not fulfill the password requirements.</p> <ol style="list-style-type: none"> <li>1. System displays error message “Invalid password”.</li> <li>2. System returns to Step 2.</li> </ol> <p>AF-S8: System detects mismatch between password and password confirmation fields.</p> <ol style="list-style-type: none"> <li>1. System displays error message “Password mismatch”.</li> <li>2. System returns to Step 2.</li> </ol> <p>AF-S10: Reviewer disapproves the account creation request.</p> <ol style="list-style-type: none"> <li>1. System sends “License disapproved, account creation failed” notification.</li> <li>2. System returns to Step 2.</li> </ol>

Exceptions:	-
Includes:	<ul style="list-style-type: none"> <li>Validate the account availability</li> </ul>
Special Requirements:	-
Assumptions:	<ul style="list-style-type: none"> <li>Database refers to the system database</li> </ul>
Notes and Issues:	-

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:	<ul style="list-style-type: none"> <li>User</li> <li>Hawker</li> </ul>
Description:	Login to account using the registered email address and password
Preconditions:	<ul style="list-style-type: none"> <li>User account must already exist in the database.</li> <li>System must have a stable connection to the database.</li> <li>User connects to the system.</li> </ul>
Postconditions:	<ul style="list-style-type: none"> <li>Users can see the main menu of the system.</li> </ul> OR <ul style="list-style-type: none"> <li>Users see an error message.</li> <li>Users can re-login to their accounts.</li> </ul>
Priority:	High
Frequency of Use:	1 – 20 times per day
Flow of Events:	<ol style="list-style-type: none"> <li>System requires the input of email address and password.</li> </ol>

	<ol style="list-style-type: none"> <li>2. User input the email address and password in the login interface.</li> <li>3. User clicks on the login button.</li> <li>4. System verifies that email address and password have been filled out.</li> <li>5. System retrieves the information from the database.</li> <li>6. System verifies the email address and password with the information retrieved from the database.</li> <li>7. If the email address and password are verified, the system displays the main menu to the user.</li> </ol>
Alternative Flows:	<p>AF-S5: If email address or password is not filled out.</p> <ol style="list-style-type: none"> <li>1. System displays the message “Please input all the required information”.</li> <li>2. System returns to Step 2.</li> </ol> <p>AF-S6: If the email address does not exist in the database.</p> <ol style="list-style-type: none"> <li>1. System displays the message “Invalid email address or password. Do you want to create an account?”.</li> <li>2. System returns to Step 2.</li> </ol> <p>AF-S7: If the email address and password does not match the information in the database.</p> <ol style="list-style-type: none"> <li>1. System displays the message “Invalid email address or password. Do you want to create an account?”.</li> <li>2. System returns to Step 2.</li> </ol>
Exceptions:	-
Includes:	<ul style="list-style-type: none"> <li>• Verify Login Credentials.</li> </ul>
Special Requirements:	-
Assumptions:	<ul style="list-style-type: none"> <li>• Database can be referred to System’s database.</li> </ul>

Notes and Issues:	-
-------------------	---

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:	<ul style="list-style-type: none"> <li>• User</li> <li>• Hawker</li> <li>• Email API</li> </ul>
Description:	To reset password when user or hawker forgot their password
Preconditions:	<ul style="list-style-type: none"> <li>• User or hawker account must already exist in the database.</li> <li>• System must have a stable connection to the database.</li> </ul>
Postconditions:	<ul style="list-style-type: none"> <li>• Users or hawkers can re-login to their accounts.</li> </ul>
Priority:	Low
Frequency of Use:	1-5 times per lifetime
Flow of Events:	<ol style="list-style-type: none"> <li>1. User or hawker clicks on the "forgot password" button at the login page.</li> <li>2. System requests user to enter the email address used to register for the account.</li> <li>3. System verifies that the email address exists in the database.</li> <li>4. System sends the user a link to the reset password website through email.</li> <li>5. User clicks on the reset password website link.</li> <li>6. User inputs a new valid password and password confirmation fields.</li> </ol>



	7. System automatically sends a confirmation email to the user upon successful reset of their password.
Alternative Flows:	<p>AF-S3: Email address does not exist in the database.</p> <ol style="list-style-type: none"> <li>1. System shows message “Invalid email address”.</li> <li>2. System returns to Step 2.</li> </ol> <p>AF-S6: System detects password does not fulfill the password requirements.</p> <ol style="list-style-type: none"> <li>1. System displays error message “Invalid password”.</li> <li>2. System returns to Step 6.</li> </ol> <p>AF-S6: System detects mismatch between password and password confirmation fields.</p> <ol style="list-style-type: none"> <li>1. System displays error message “Password mismatch”.</li> <li>2. System returns to Step 6.</li> </ol>
Exceptions:	-
Includes:	<ul style="list-style-type: none"> <li>• Verify Login Credentials.</li> </ul>
Special Requirements:	-
Assumptions:	<ul style="list-style-type: none"> <li>• Database can be referred to System’s database.</li> </ul>
Notes and Issues:	-

Use Case ID:	03A		
Use Case Name:	Hawkers update the information of the stall		
Created By:	Tan Ming Hao	Last Updated By:	Chow Weng Shi
Date Created:	30-8-2024	Date Last Updated:	3-9-2024

Actor:	<ul style="list-style-type: none"> <li>• Hawkercs</li> </ul>
--------	--

	<ul style="list-style-type: none"> <li>• Users</li> </ul>
Description:	Hawkers update information of their respective stall
Preconditions:	<ul style="list-style-type: none"> <li>• System must have a stable connection to the database.</li> <li>• Hawker account must exist in the database.</li> <li>• Hawkercan want to update the information of their respective stall after seeing its information.</li> </ul> <p>OR</p> <ul style="list-style-type: none"> <li>• Hawkercan must update the information of their respective stall after seeing a fault report from the users.</li> </ul>
Postconditions:	The system will update the information about the stalls.
Priority:	Medium
Frequency of Use:	0 – 20 times per week
Flow of Events:	<ol style="list-style-type: none"> <li>1. Hawker login to their respective account.</li> <li>2. The system shows the number of fault reports on "Fault Report" icon.</li> <li>3. The system shows the "Review" icon.</li> <li>4. The system shows the "Menu" icon.</li> <li>5. The system shows the "Opening hours" icon.</li> <li>6. The system shows "Open Shop" and "Close Shop" button.</li> <li>7. The system shows "Shut Down Shop" button.</li> <li>8. The system shows a red "X" icon.</li> <li>9. System updates the information of the stall.</li> </ol>
Alternative Flows:	<p>AF-S2: The hawker clicks on the fault report icon.</p> <ol style="list-style-type: none"> <li>1. The system displays all the reporters' names and their corresponding fault reports.</li> <li>2. The system will display an exit button and a button to delete the notification.</li> </ol>

	<ol style="list-style-type: none"> <li>3. The system will remove the report if the user clicks on the button to delete the notification.</li> <li>4. The system returns to Step 3 upon exit.</li> </ol> <p>AF-S3: The hawker clicks on the review icon.</p> <ol style="list-style-type: none"> <li>1. The system displays all the newly added reviews.</li> <li>2. The system displays an exit button and a button to delete the notification.</li> <li>3. The system will remove the review notifications if the user clicks on the button to delete the notification.</li> <li>4. The system returns to Step 4 upon exit.</li> </ol> <p>AF-S4: The hawker clicks on the “Menu” button.</p> <ol style="list-style-type: none"> <li>1. The system will show a page for the hawker to edit the menu.</li> <li>2. The hawker edits the menu of the stalls.</li> <li>3. The hawker clicks the save button.</li> <li>4. The system returns to Step 5 upon exit.</li> </ol> <p>AF-S5: The hawker clicks on the “Opening hours” button.</p> <ol style="list-style-type: none"> <li>1. The system will show a page for the hawker to edit the opening hours and operating days.</li> <li>2. Hawker edits the opening time and operating days.</li> <li>3. Hawker clicks on the “Save” button.</li> <li>4. The system returns to Step 6 upon exit.</li> </ol> <p>AF-S7: The hawker clicks on the “Shut Down” button.</p> <ol style="list-style-type: none"> <li>1. The system will show a confirmation page to delete the hawker account.</li> <li>2. Hawker inputs two times his correct password.</li> </ol>
--	--

	<ol style="list-style-type: none"> <li>Hawker clicks on the confirmation button.</li> <li>The system deletes the account of the respective hawkers.</li> </ol>
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	<ul style="list-style-type: none"> <li>Information of the hawker's stalls can be referred to System's database.</li> </ul>
Notes and Issues:	-

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:	<ul style="list-style-type: none"> <li>Hawkers</li> </ul>
Description:	Hawkers update the menu of the stalls of their respective stalls
Preconditions:	<ul style="list-style-type: none"> <li>System must have a stable connection to the database.</li> <li>Hawker account must exist in the database.</li> <li>Hawker clicks on the "Menu" button.</li> </ul>
Postconditions:	The system will update the menu about the stalls.
Priority:	Medium
Frequency of Use:	0-12 per year
Flow of Events:	<ol style="list-style-type: none"> <li>The system shows a page which includes the menu of the shop.</li> <li>The hawker clicks on the item at the right.</li> <li>The system shows the detail of the item.</li> </ol>

	<ol style="list-style-type: none"> <li>The system shows an “Add menu” button, “Delete” button, “Save” button and “Exit” button.</li> <li>The system returns to the home page of hawker.</li> </ol>
Alternative Flows:	<p>AF-S4: The hawker clicks on the “Add menu” button.</p> <ol style="list-style-type: none"> <li>The systems show a new row for the hawker to input a new menu item.</li> <li>The hawker fills in the details of the new item which include picture, name, description and price.</li> <li>The hawker clicks on the “Save”</li> <li>The system adds the item to the database of the store.</li> <li>The system returns to Step 4 upon exit.</li> </ol> <p>AF-S4: The hawker clicks on the “Delete” button.</p> <ol style="list-style-type: none"> <li>The systems remove the item.</li> <li>The hawker clicks on the “Save” button.</li> <li>The system updates the details of the item in the menu.</li> <li>The system returns to Step 4 upon exit.</li> </ol> <p>AF-S4: The hawker clicks on the “Exit” button.</p> <ol style="list-style-type: none"> <li>The system returns to Step 5.</li> </ol>
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	The system will only show the “Edit” button if there is at least one item in the menu.
Notes and Issues:	-

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:	9-9-2024

Actor:	<ul style="list-style-type: none"> <li>Hawkers</li> </ul>
Description:	Hawkers close their account and their stall.
Preconditions:	<ul style="list-style-type: none"> <li>System must have a stable connection to the database.</li> <li>Hawker account must exist in the database.</li> <li>Hawkers click on the red “X” button.</li> </ul>
Postconditions:	<ul style="list-style-type: none"> <li>The system will remove the hawker’s account and its information from the database.</li> <li>The hawker account is removed successfully.</li> <li>The hawker account no longer exists.</li> </ul>
Priority:	Medium
Frequency of Use:	Once per lifetime
Flow of Events:	<ol style="list-style-type: none"> <li>The system displays the message “Are you sure you want to delete your account?”</li> <li>Hawker clicks “Yes”.</li> <li>The system displays two text boxes and requires the hawker to enter their password correctly twice.</li> <li>The hawker enters their password correctly twice.</li> <li>The system verifies the correctness of the password.</li> <li>The hawker clicks on the “Delete Account” button.</li> <li>The system processes the request (delete account).</li> <li>The system shows message (“Account delete successfully”)</li> <li>The system to Sign Up Page.</li> </ol>

Alternative Flows:	<p>AF-S5: If the email address and password does not match the information in the database.</p> <ol style="list-style-type: none"> <li>1. The system displays the message “Password is incorrect”.</li> <li>2. The system returns to Step 1.</li> </ol>
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	04A		
Use Case Name:	User 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:	<ul style="list-style-type: none"> <li>• User</li> <li>• Map system</li> </ul>
Description:	User can view the information of a list of stalls after logging in to the account
Preconditions:	<ul style="list-style-type: none"> <li>• User account must already exist in the database.</li> <li>• System must have a stable connection to the database.</li> </ul>
Postconditions:	<ul style="list-style-type: none"> <li>• The system displays a list of stalls with required information.</li> </ul>
Priority:	High
Frequency of Use:	1 – 20 times per day
Flow of Events:	<ol style="list-style-type: none"> <li>1. User logs in to respective account successfully.</li> </ol>

	<ol style="list-style-type: none"> <li>The system displays the location of all hawker centers on the map.</li> <li>The user clicks on a desired hawker center.</li> <li>The system retrieves the information of all stalls in the chosen hawker center from the database.</li> <li>The system shows a list of stalls in the hawker center with name, opening status, opening hours in 12-hour notation, ratings in number, crowd situation of the stalls, and a button to view the reviews.</li> </ol>		
Alternative Flows:	<p>AF-S5: The user clicks on the name of the desired stall.</p> <ol style="list-style-type: none"> <li>The system displays the menu of the stall, with names and pictures of dishes and their corresponding prices.</li> <li>The system returns to Step 6 upon exit.</li> </ol> <p>AF-S5: The user clicks on the review button.</p> <ol style="list-style-type: none"> <li>The system displays the overall ratings of the stall and reviews by other users.</li> <li>The system returns to Step 6 upon exit.</li> </ol>		
Exceptions:	-		
Includes:	-		
Special Requirements:	-		
Assumptions:	<ul style="list-style-type: none"> <li>Information about the hawker center can be referred to in the System's database.</li> </ul>		
Notes and Issues:	-		

Use Case ID:	05A		
Use Case Name:	User 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:	<ul style="list-style-type: none"> <li>User</li> </ul>
Description:	User 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:	<ul style="list-style-type: none"> <li>User account must already exist in the database.</li> <li>System must have a stable connection to the database.</li> </ul>
Postconditions:	<ul style="list-style-type: none"> <li>The record of the fault report has been stored in the database.</li> <li>Hawker can see the submitted fault report.</li> </ul>
Priority:	Medium
Frequency of Use:	0-5 times per day
Flow of Events:	<ol style="list-style-type: none"> <li>1. User goes to the desired stall after searching for the information through the system.</li> <li>2. User logs in to respective account successfully.</li> <li>3. User searches up the current hawker center and the specific stall.</li> <li>4. The system displays the stall information and a report button.</li> <li>5. The user clicks on the report button.</li> <li>6. The system displays “What issues have you found about the current stall?”, a text box below for the users to elaborate as well as an information button for the terms and condition of filing a report.</li> <li>7. The user inputs the issues they found in the text box and submits the report.</li> <li>8. The system displays “Fault report is submitted successfully.”.</li> <li>9. The system returns to Step 4 upon exit.</li> </ol>
Alternative Flows:	AF-S6: The user submits the report without inputting at least 1 character in the text box.

	<ol style="list-style-type: none"> <li>1. The system displays message “The report must contain at least one character”.</li> <li>2. The system returns to Step 7.</li> </ol> <p>AF-S6: The user clicks on the information button.</p> <ol style="list-style-type: none"> <li>1. The system displays a page with information about when to file a report, as well as the terms and conditions of filing a report.</li> <li>2. The system returns to Step 6 upon exit.</li> </ol>
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	User realizes false information about certain stalls in the system.
Notes and Issues:	-

Use Case ID:	06A		
Use Case Name:	User updates crowd situation about the store		
Created By:	Tan Ming Hao	Last Updated By:	Lai Xin Yee
Date Created:	31-8-2024	Date Last Updated:	2-9-2024

Actor:	<ul style="list-style-type: none"> <li>• User</li> </ul>
Description:	User provides the information about the crowd situation about the stalls
Preconditions:	<ul style="list-style-type: none"> <li>• System must have a stable connection to the database.</li> <li>• User account must exist in the database.</li> <li>• User logs in to the respective account.</li> </ul>
Postconditions:	<ul style="list-style-type: none"> <li>• System updates the crowd situation of the store.</li> </ul>
Priority:	Medium

Frequency of Use:	0-2 times per day
Flow of Events:	<ol style="list-style-type: none"> <li>1. User searches and clicks on a certain stall.</li> <li>2. User clicks on the “Update crowd situation” button.</li> <li>3. User selects the number of icons that represent the crowd situation of the stalls.</li> <li>4. User clicks on the “Update” button.</li> <li>5. System displays message “Update Successfully”.</li> </ol>
Alternative Flows:	-
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	User realizes that the crowd situation is different from what is shown the system.
Notes and Issues:	-

Use Case ID:	07A		
Use Case Name:	User 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:	<ul style="list-style-type: none"> <li>• User</li> </ul>
Description:	User provides reviews, and ratings of the stall they visited
Preconditions:	<ul style="list-style-type: none"> <li>• System must have a stable connection to the database.</li> <li>• User account must exist in the database.</li> </ul>
Postconditions:	<ul style="list-style-type: none"> <li>• System shows “review/rating uploaded” message</li> </ul>
Priority:	Medium
Frequency of Use:	1-2 times per day

Flow of Events:	<ol style="list-style-type: none"> <li>1. User search and tap on the stall that they want to provide the review or rating.</li> <li>2. User writes in their review and choose the number of stars for ratings in the review or rating section.</li> <li>3. User taps the “Upload” button to upload the review or rating.</li> <li>4. System shows “review/rating uploaded” message.</li> </ol>
Alternative Flows:	<p>AF-S2: User does not write in review and choose number of stars for ratings.</p> <ol style="list-style-type: none"> <li>1. User only choose the number of stars for ratings and do not input any characters into the text box.</li> <li>2. System returns to Step 3.</li> </ol>
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	08A		
Use Case Name:	User 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:	<ul style="list-style-type: none"> <li>• User</li> </ul>
Description:	To search for a desired location
Preconditions:	<ul style="list-style-type: none"> <li>• System is connected to Wi-Fi or Mobile Data.</li> </ul>
Postconditions:	<ul style="list-style-type: none"> <li>• A list of relevant hawker centers will be displayed in the system.</li> </ul>

Priority:	High
Frequency of Use:	0-15 times per day
Flow of Events:	<ol style="list-style-type: none"> <li>1. User enters query in the search bar.</li> <li>2. User taps the search button without adding filters.</li> <li>3. System provides a list of relevant search results based on the user's query.</li> </ol>
Alternative Flows:	-
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	09A		
Use Case Name:	Direct the user to a desired location		
Created By:	Lai Xin Yee	Last Updated By:	Lai Xin Yee
Date Created:	30-8-2024	Date Last Updated:	31-8-2024

Actor:	<ul style="list-style-type: none"> <li>• User</li> <li>• Map System</li> </ul>
Description:	To direct the user from their current location to the desired location
Preconditions:	<ul style="list-style-type: none"> <li>• System must have a stable connection to the database.</li> <li>• User has searched for their desired location.</li> </ul>
Postconditions:	<ul style="list-style-type: none"> <li>• Close the Google Map directory interface.</li> </ul>
Priority:	High
Frequency of Use:	3-5 times per day

Flow of Events:	<ol style="list-style-type: none"> <li>1. User tabs on the “Show Direction” button beside their desired location.</li> <li>2. System directs the user to Google Map.</li> <li>3. System detected that the user has arrived at the location.</li> <li>4. System closes Google Map and directs the user back to the last interface.</li> </ol>
Alternative Flows:	<p>AF-S3: User closes Google Map before arriving at the destination.</p> <p>Return to Step 4.</p>
Exceptions:	<ul style="list-style-type: none"> <li>• Hawker does not include the Google Map link of their stall when updating the information of their stall in the system.</li> </ul>
Includes:	-
Special Requirements:	-
Assumptions:	<ul style="list-style-type: none"> <li>• The “Show Direction” button consists of a hyperlink to Google Map which can then direct the user to the location.</li> </ul>
Notes and Issues:	-

# UI Mockup Diagram

Login Page

(1) Login Page

Domain

Email

Password

**Login**

[Sign Up](#)  
[Forgot Password](#)

Load Button

Sign Up Page

(2a) Signup Page

Enter Domain

Email

Password

Re-enter Password

**Sign up**

Sign Up Page (with Hawker Verification)

(2b) Sign-up page (hawkers verification)

Domain

Email

Password

Re-enter Password

Upload license

**Submit**

Forgot Password Page

(3a) Reset Password Page (Request)

**Forgot your password?**

Domain

Email

**Submit**

Reset Password Page

(3b) Reset Password Page (Reset)

**Reset Password**

Enter New Password

Re-Enter New Password

**Reset password**

Hawker Main Page

(4) Home page (hawkers)

**Western Stall**

**UPDATE** **STATISTICS**

**Menu** **Review**

**Opening hours** **Fault Report**

**OPEN SHOP** **CLOSE SHOP** **SHUT DOWN SHOP**

Fault Report Page (Hawker)

(5) Fault Report Page (hawkers)

Western Stall

X

Fault Reports

☐ Spaghetti price more expensive
 ☐ Menu not updated
 ☐ Opening time differs
 ☐ Fish and Chips not available anymore
 ☐ Location incorrect
 ☐

Opening Hours Page (Hawker)

(6) Opening hours page (Hawkers)

Western Stall

X

Original Opening Hours

Opening Hours:

Opening Days:

Update Opening Hours

Mon

Tue

Wed

Thu

Fri

Sat

Sun

e.g. 09:00am-09:00pm

Save

Hawker – Add new item

(7) Hawkers -- Add new menu item

Spaghetti

Burger

Fish & Chips

+

Western Stall

X

Enter Name

Save

Enter Description

Price

\$ Enter Price

Hawker – Edit Menu

(8) Hawkers -- Edit Menu Page

Spaghetti

Burger

Fish & Chips

+

Western Stall

X

Spaghetti

Edit

Tomato sauce with creamy spaghetti noodles

Price

\$ 4.50

Hawker – Delete Account

(9) Hawkers -- Delete Account Page

Western Stall

X

DELETE ACCOUNT

Are you sure you want to delete account?

Enter Password

Re-Enter Password

Delete Account

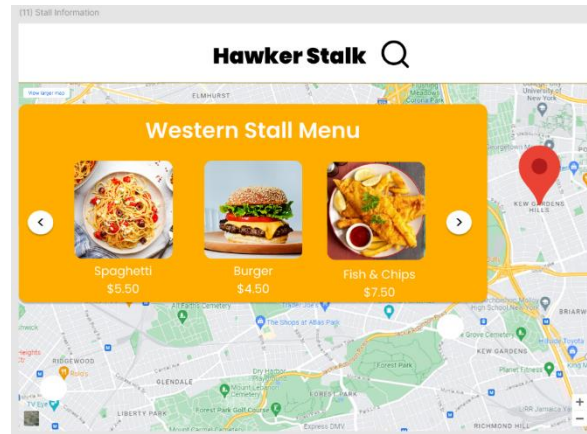
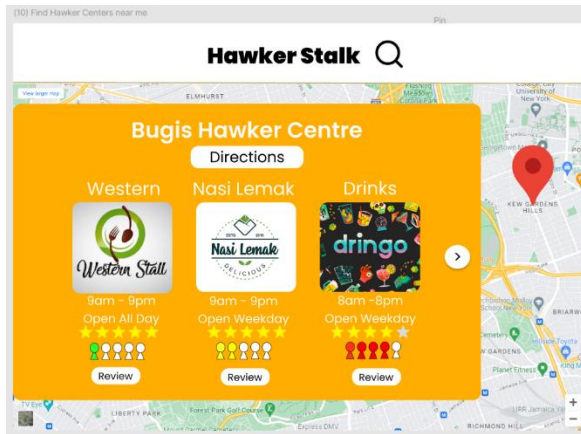
Customers – Main Page

(10) Home Page -- Customers

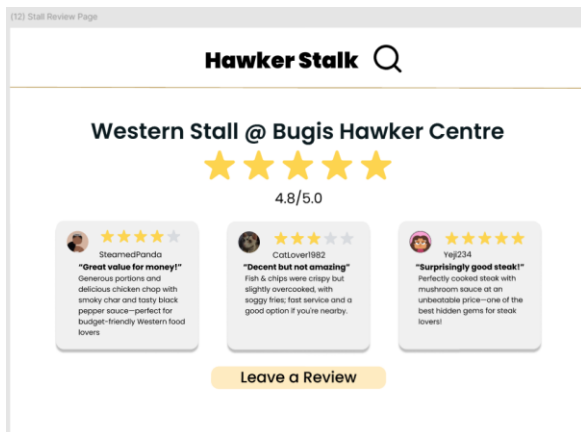
Find Hawker Center Near Me Page

Stall Information Page

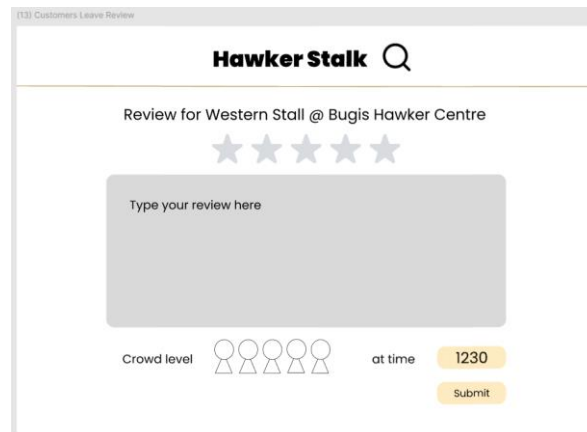




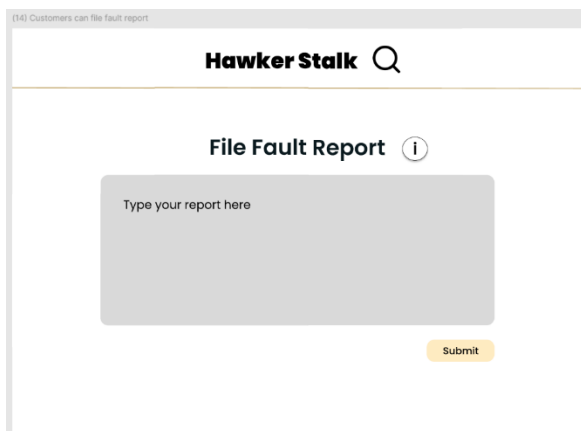
Stall Review Page



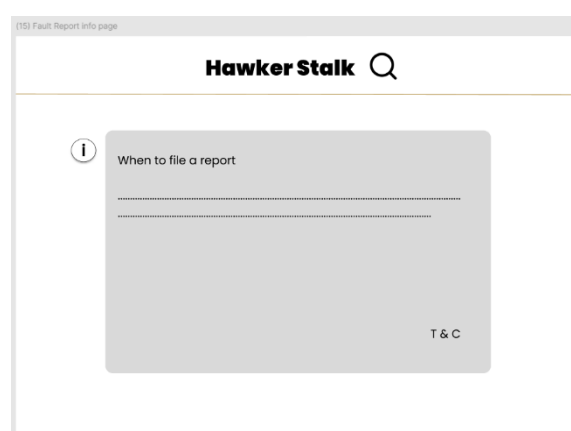
Customer Leave Review Page



Hawker File Fault Report Page

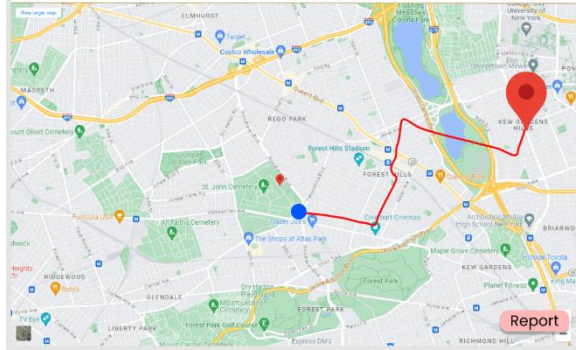


Fault Report Information Page

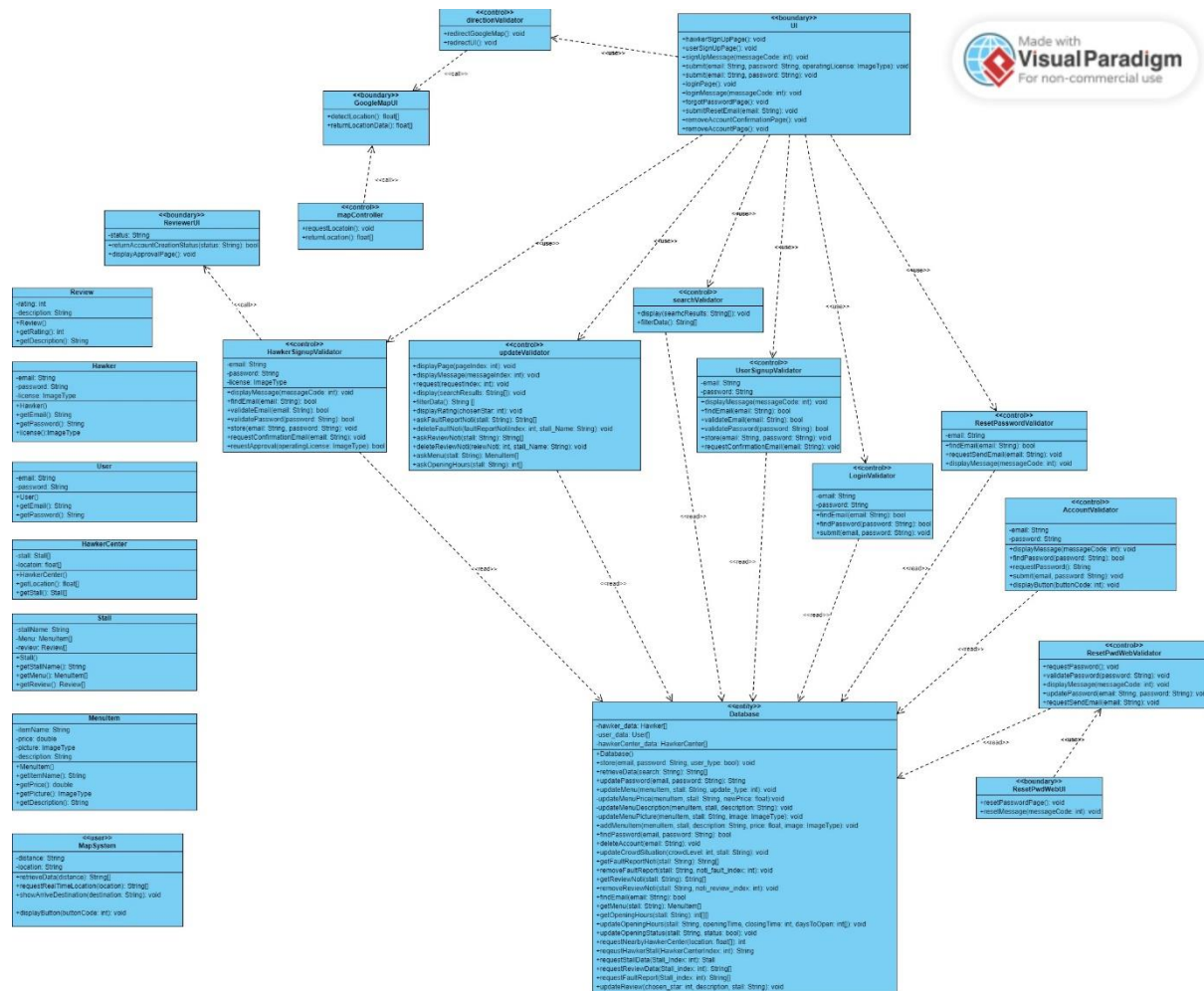


Direction to Hawker Stall Page

## Hawker Stalk 🔍



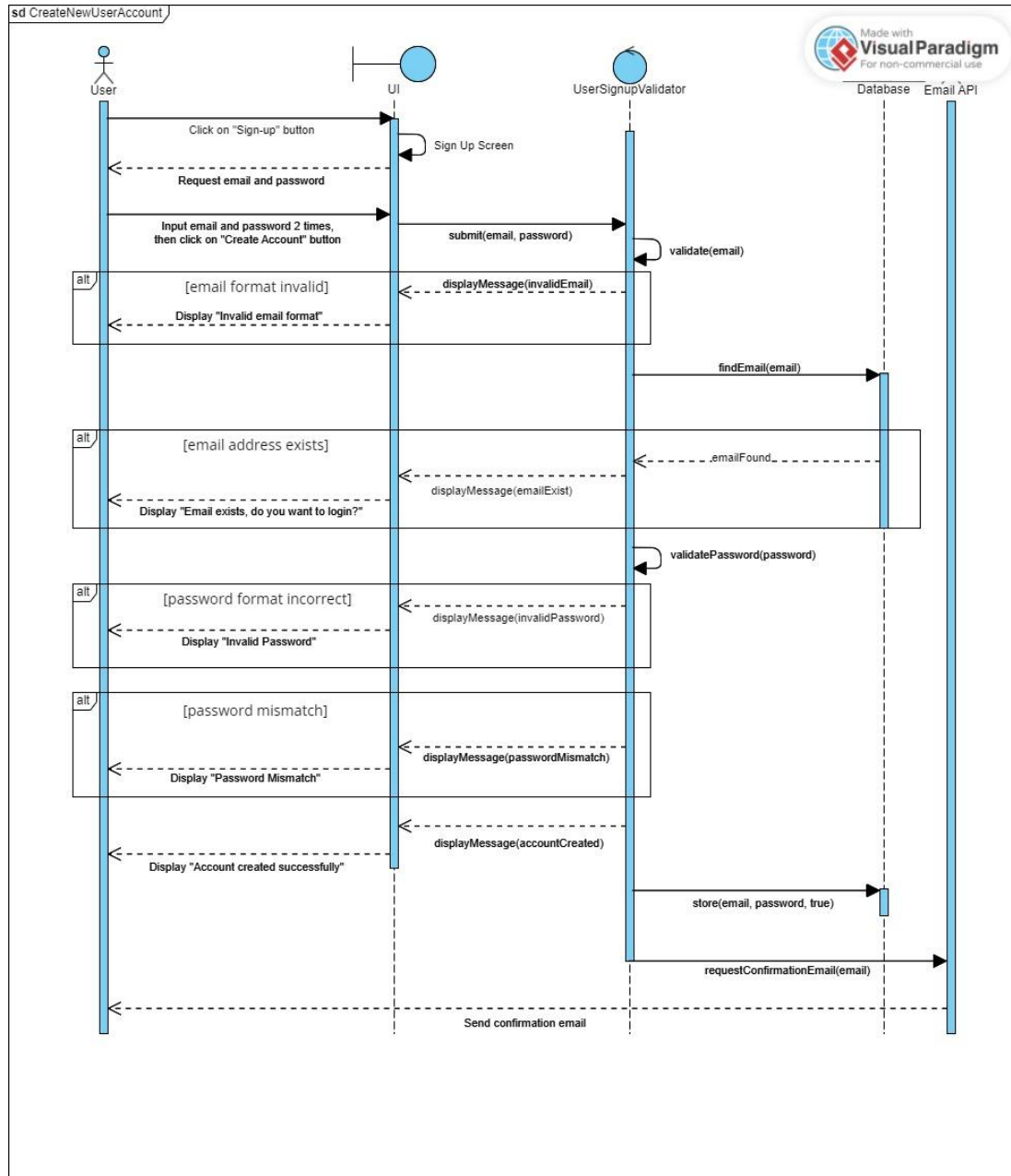
If the picture is unclear, please refer to the picture attached in the same file



# Sequence Diagrams

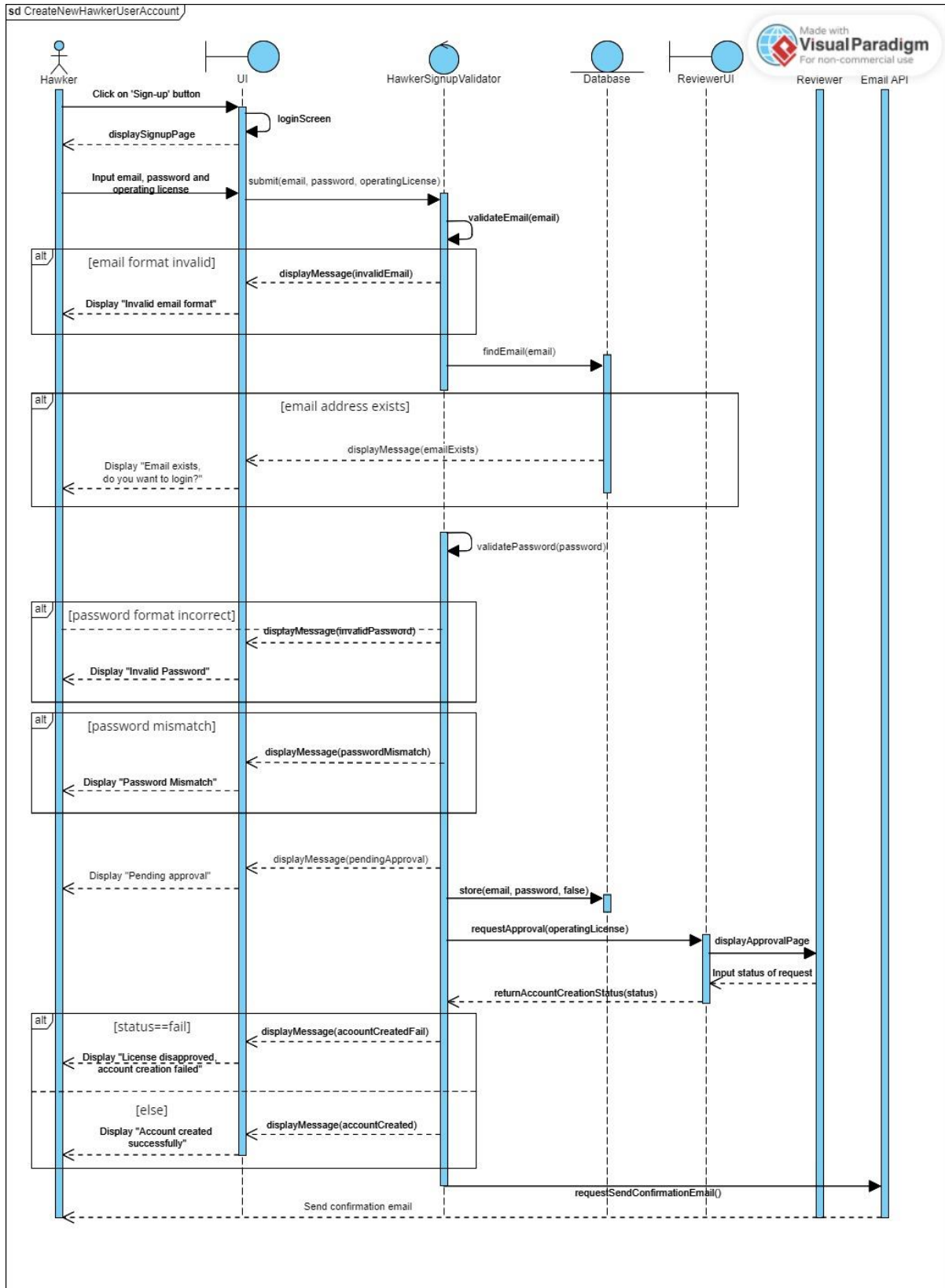
If the picture is unclear, please refer to the picture attached in the same file

Use Case ID: 01A

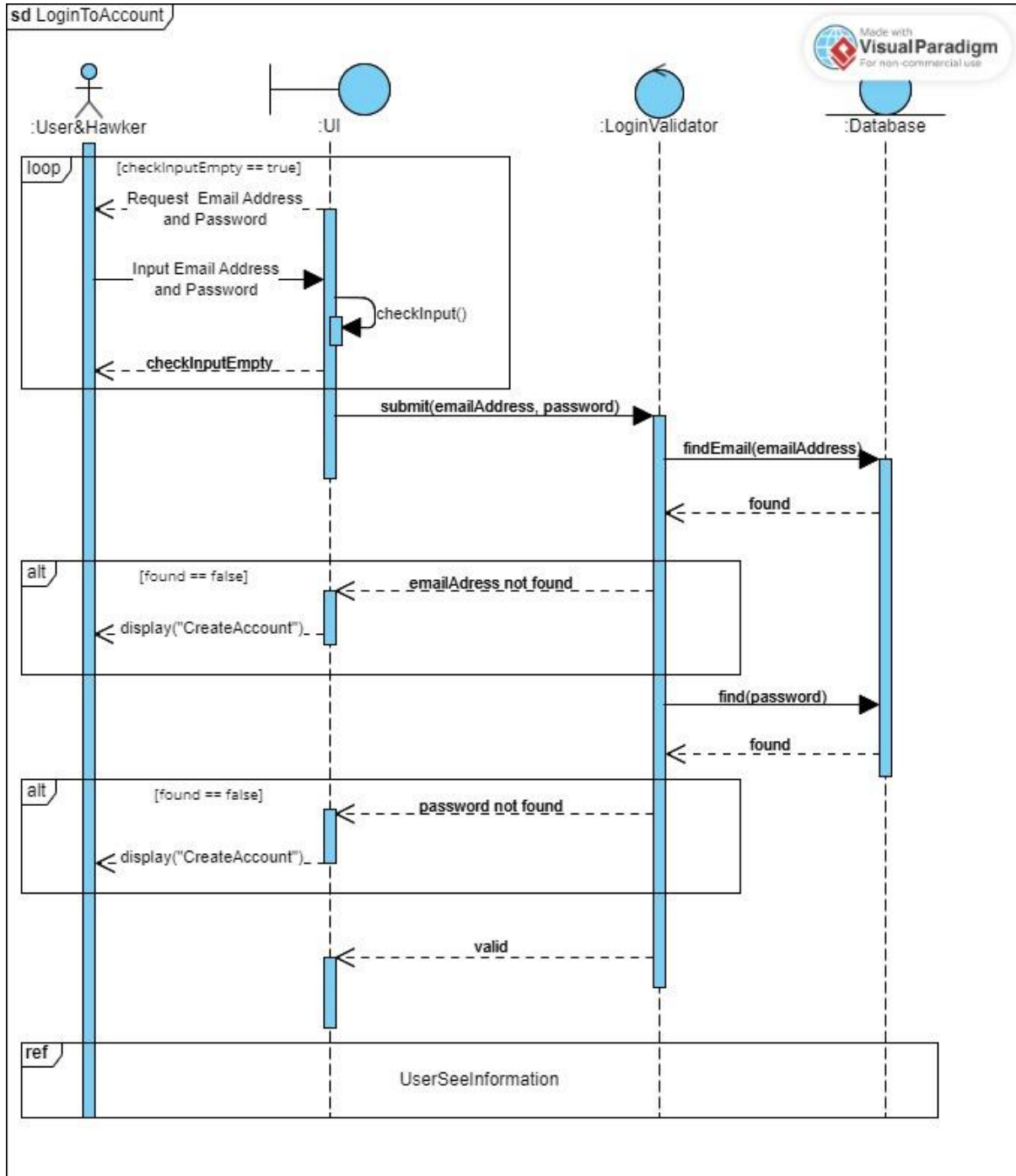


## Use Case ID: 01B

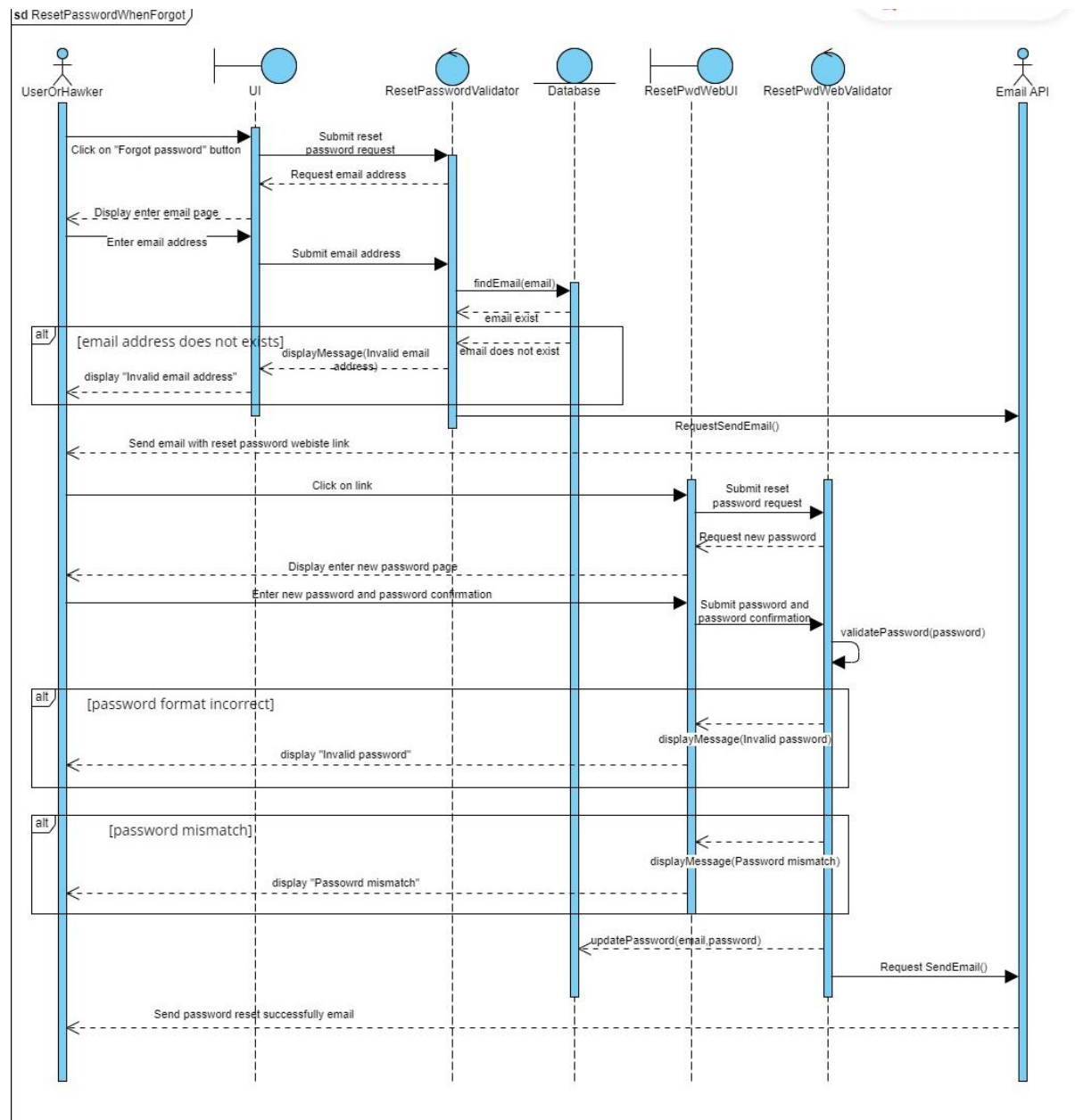
sd CreateNewHawkerUserAccount



## Use Case ID: 02A

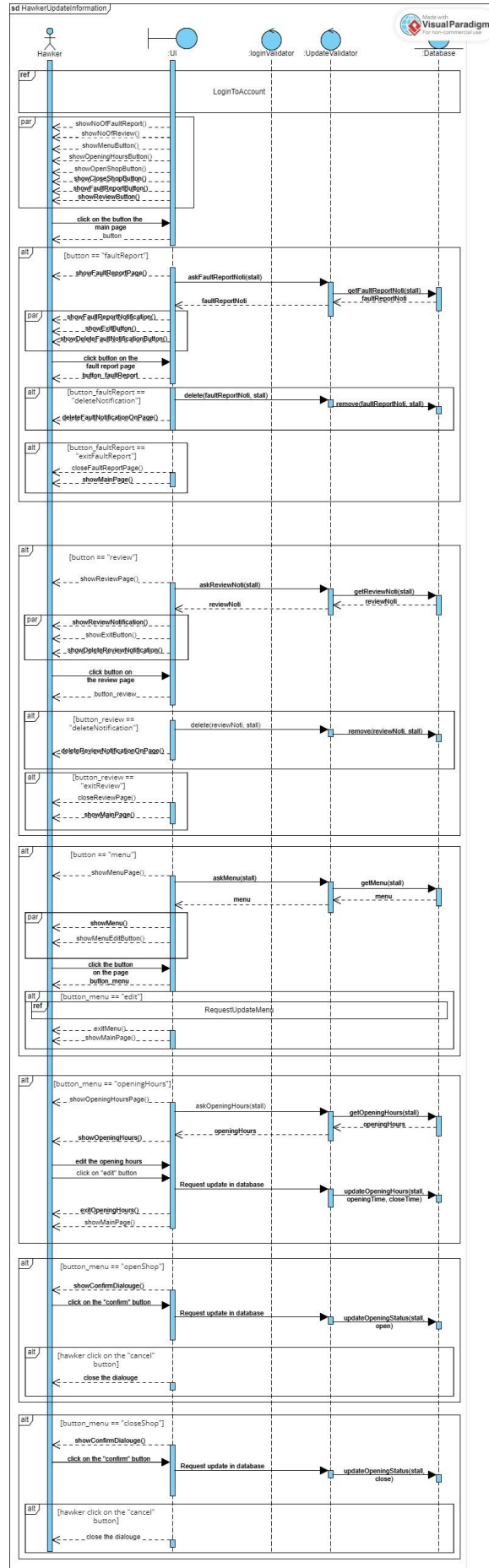


## Use Case ID: 02B





# Use Case ID: 03A



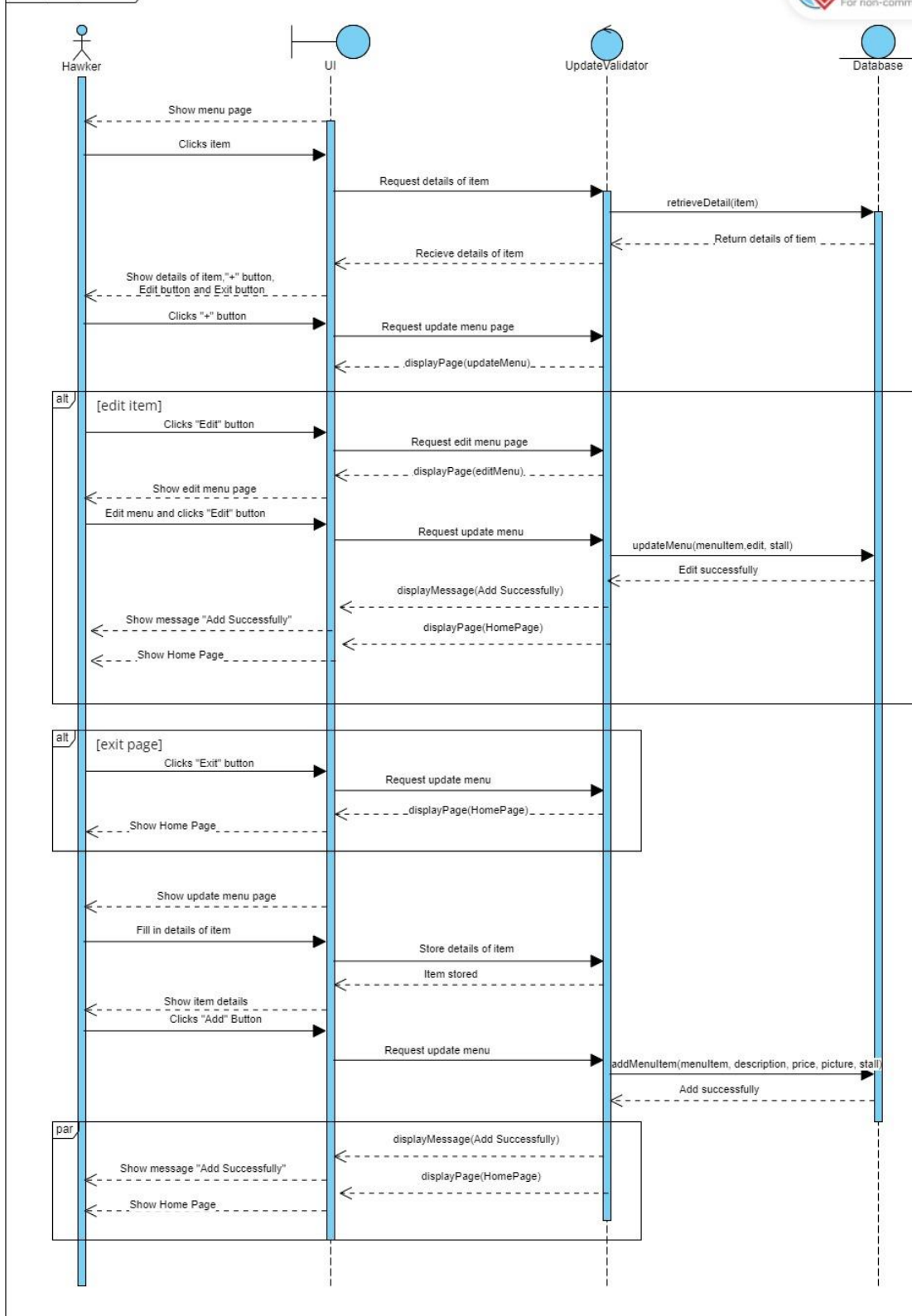


## Use Case ID: 03B

### 03B Hawker Update Menu

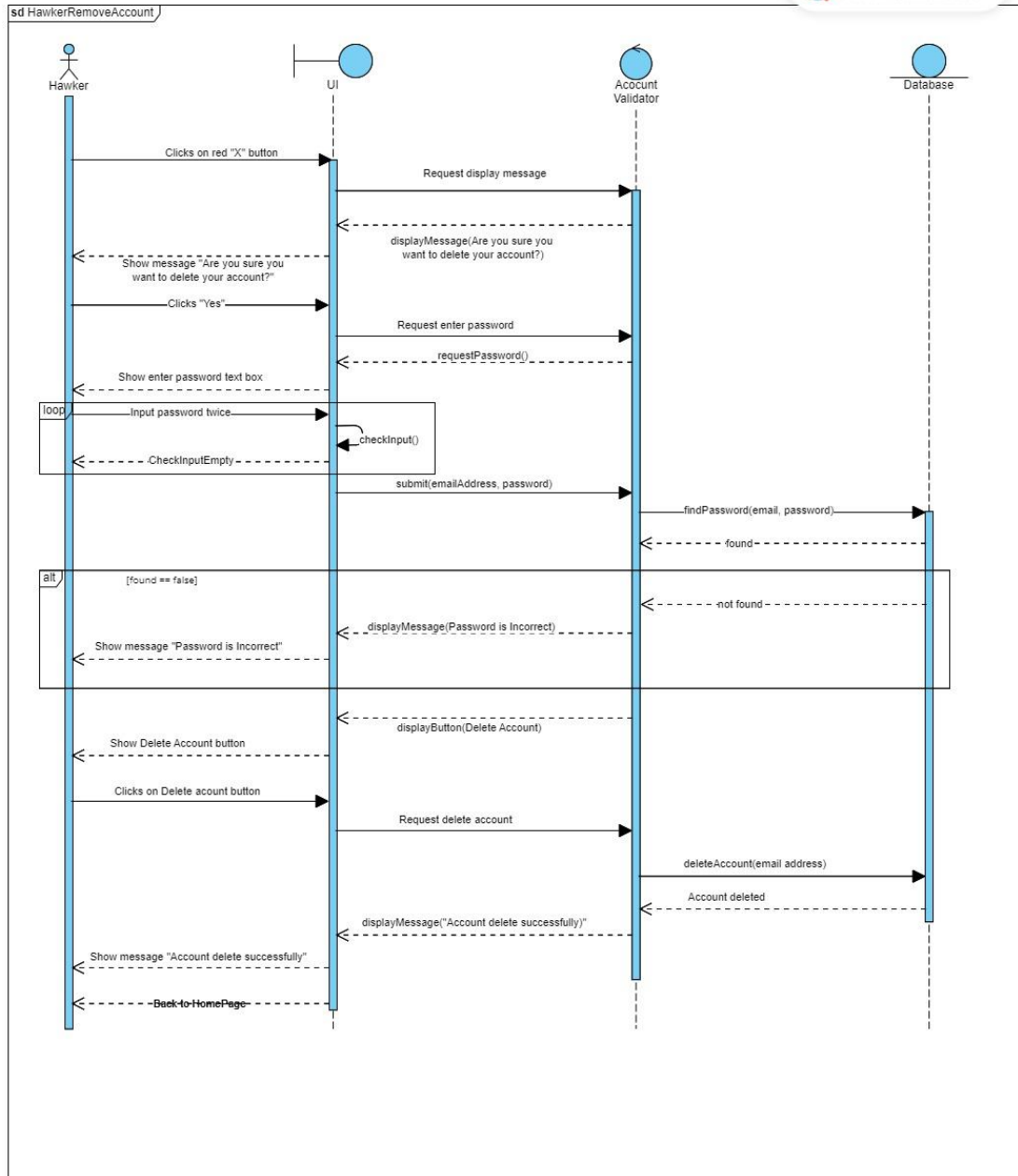


sd RequestUpdateMenu

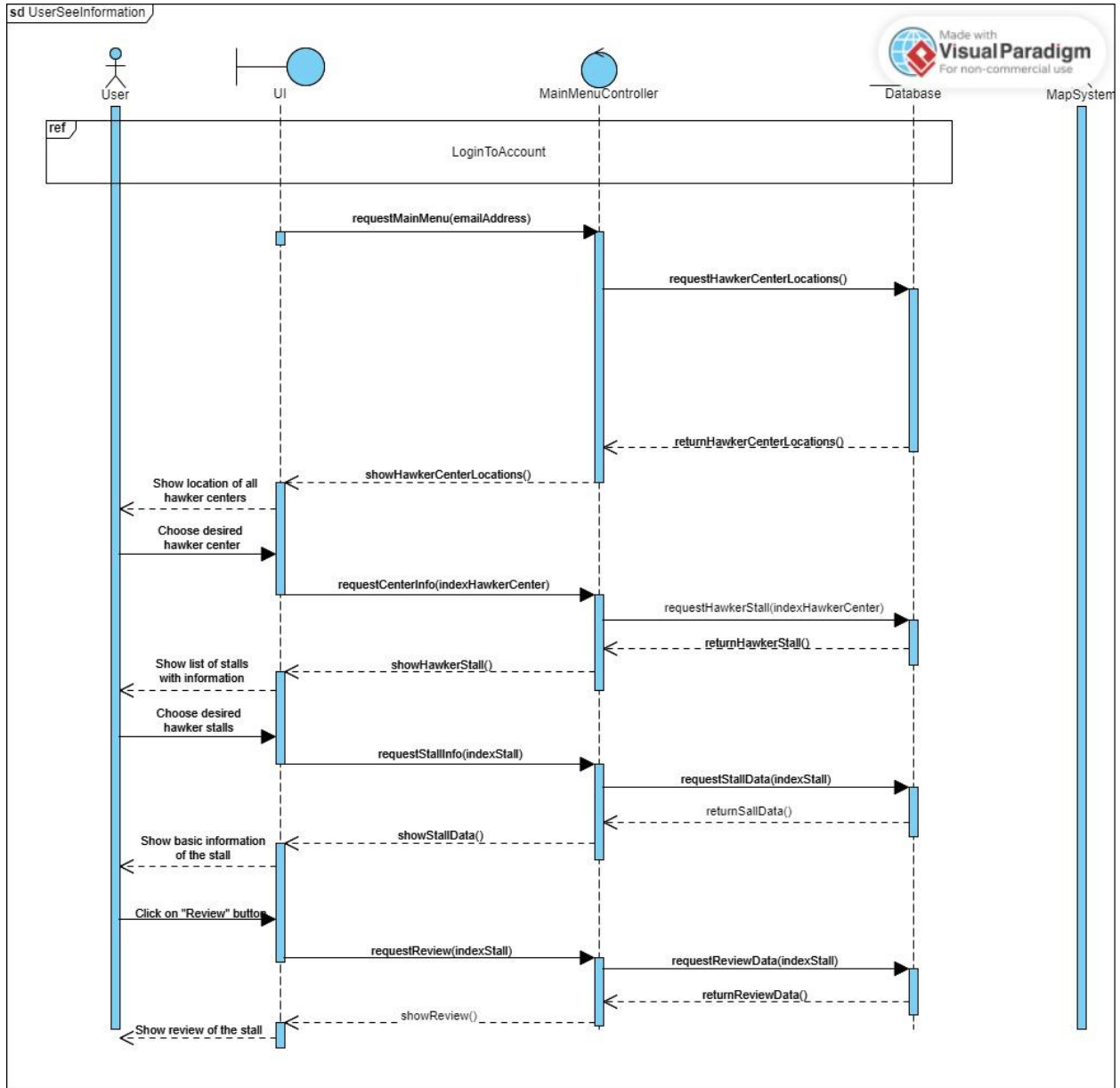


## Use Case ID: 03C

03C Hawker Remove Their Account and Their Stall

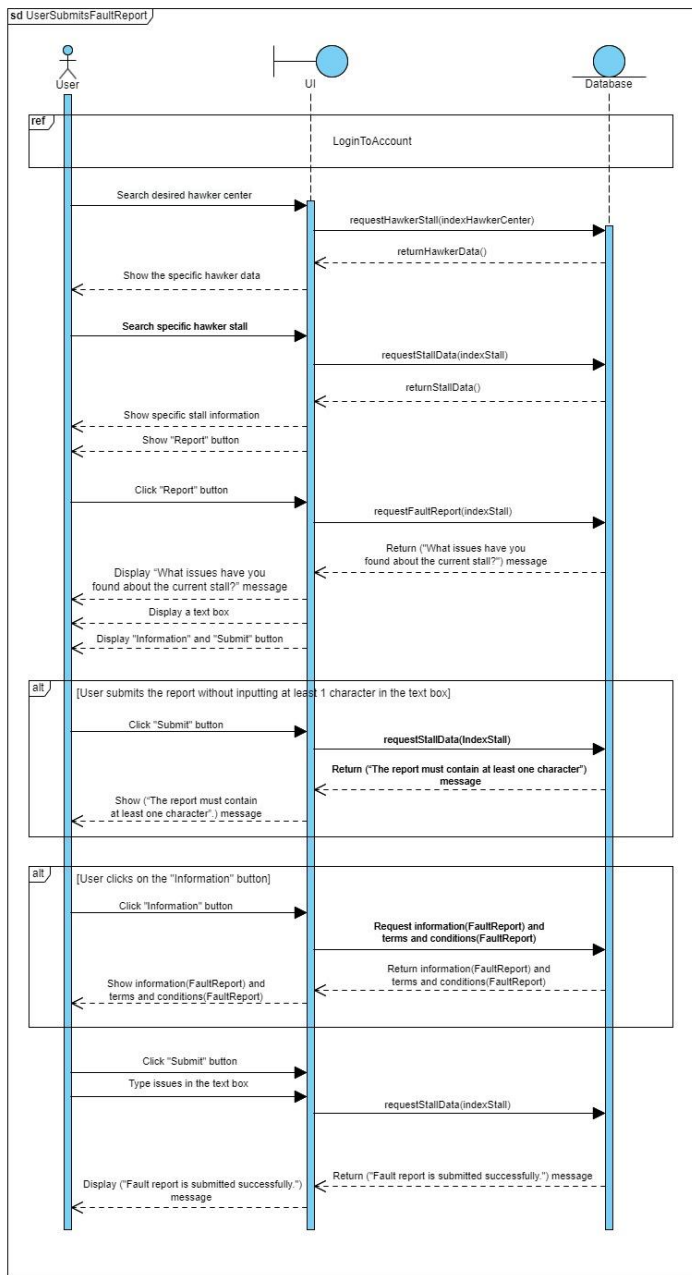


## Use Case ID: 04A



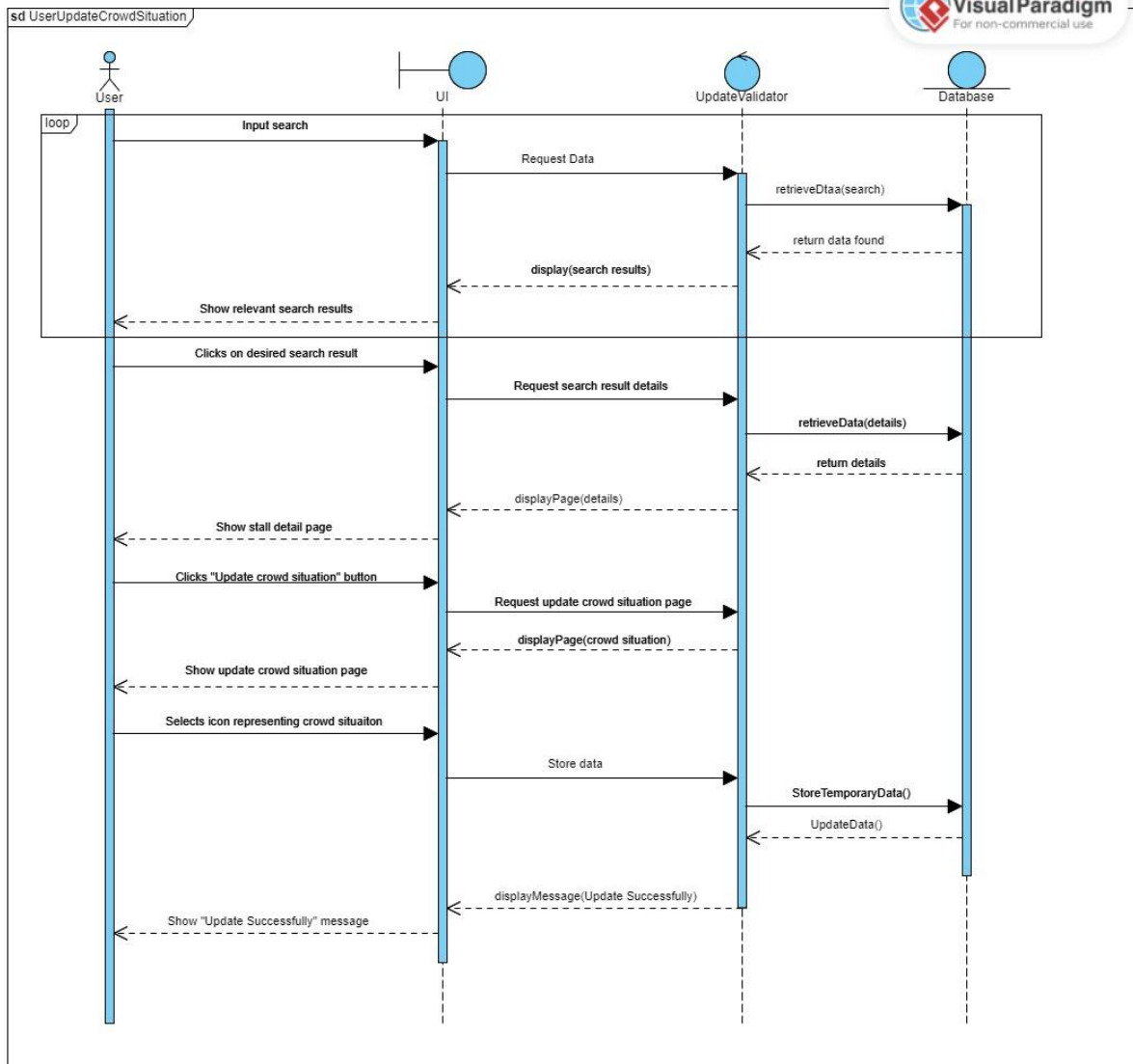
## Use Case ID: 05A

05A User Submits Fault Report

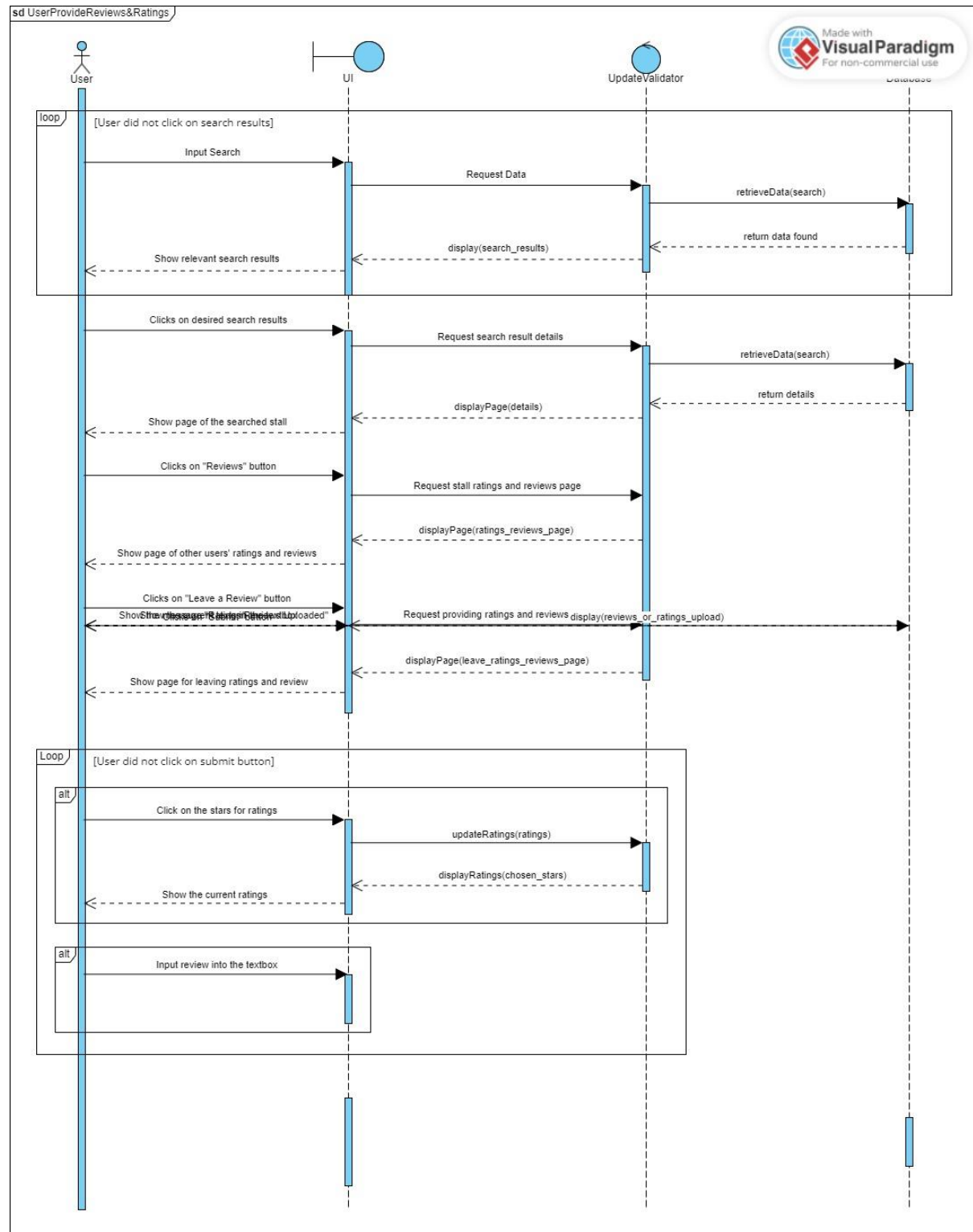


## Use Case ID: 06A

### 06A User Update Crowd Situation

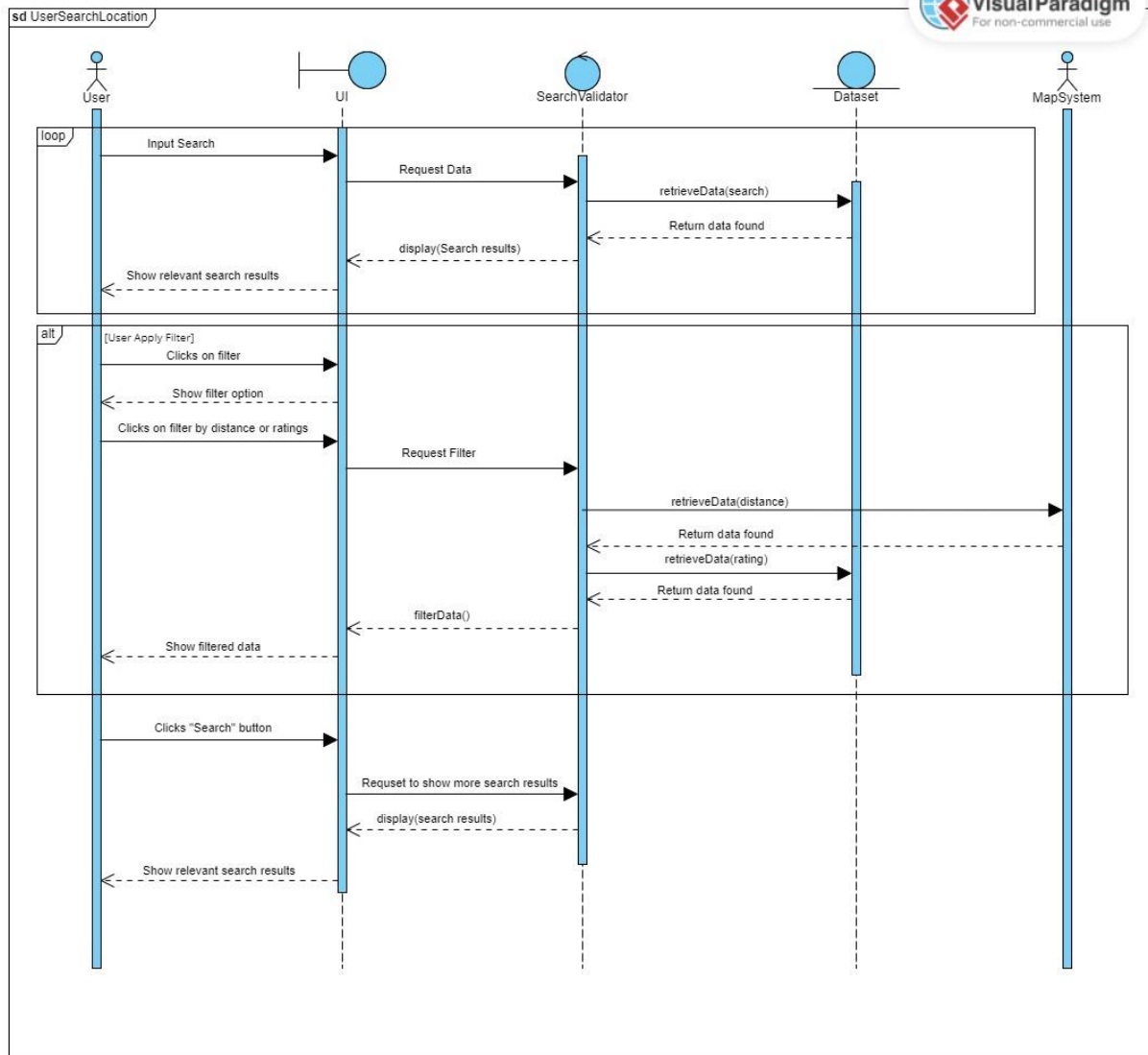


## Use Case ID: 07A



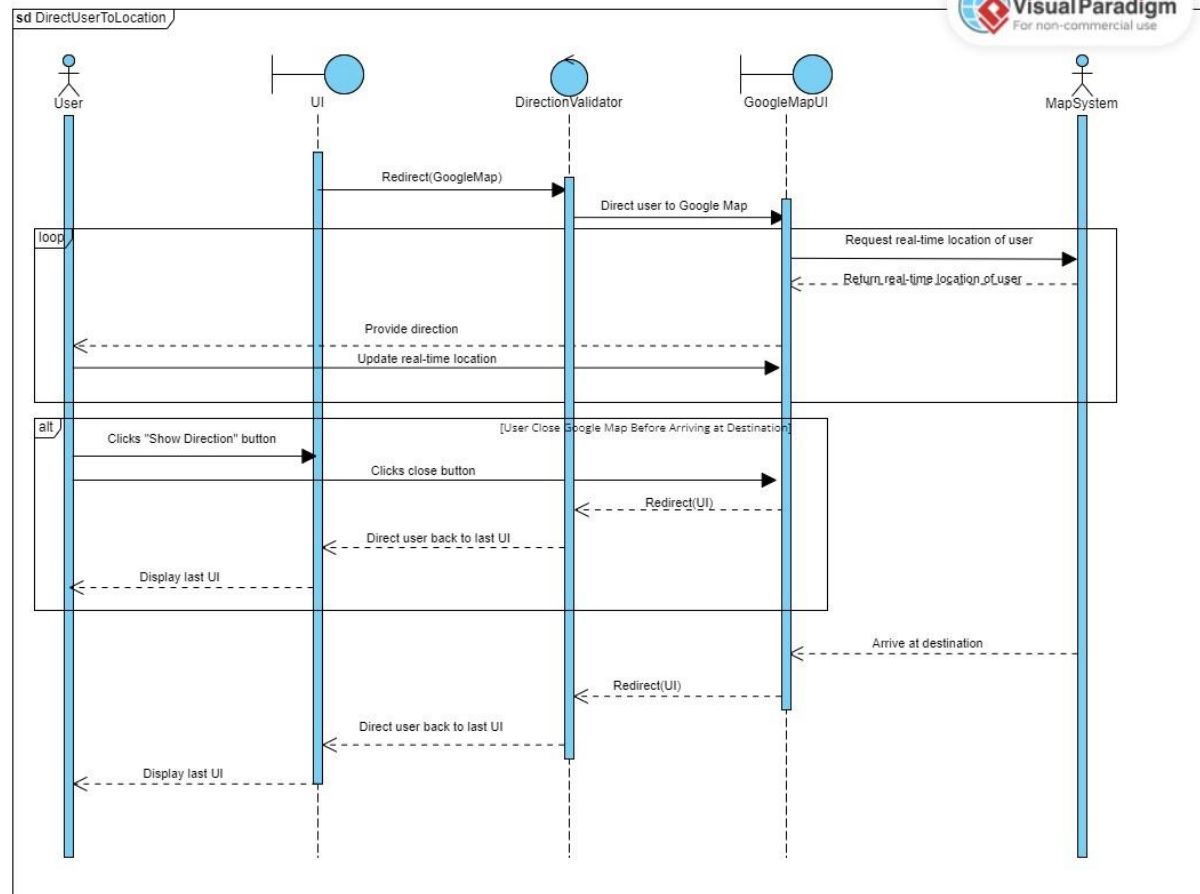
## Use Case ID: 08A

### 08A User Searches for a Location



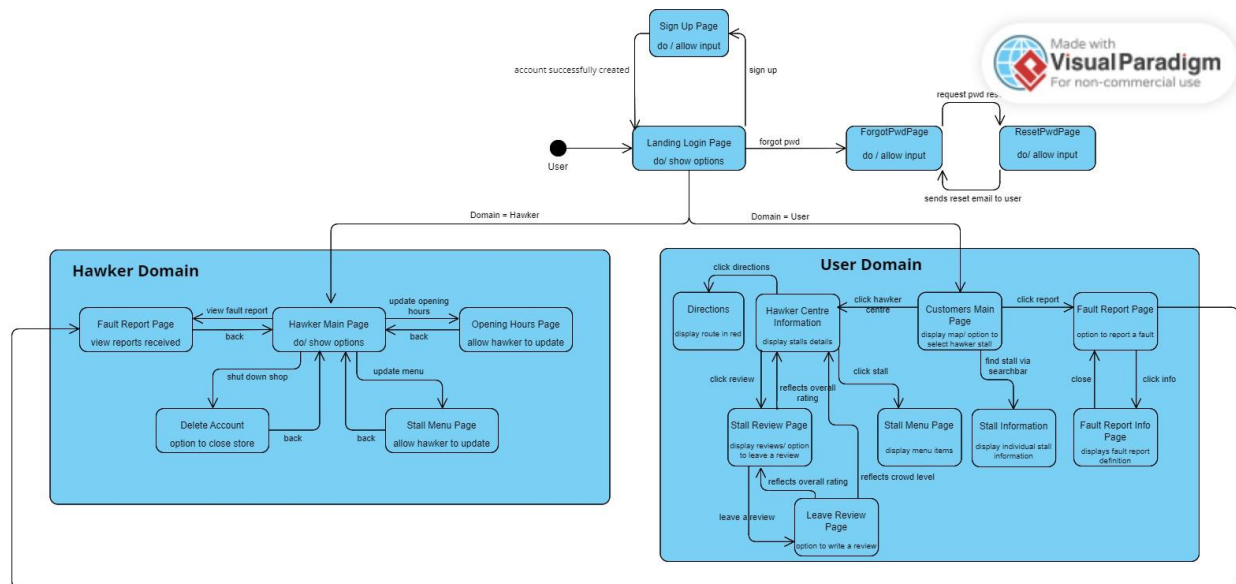
## Use Case ID: 09A

### 09A Direct User to a Desired Location

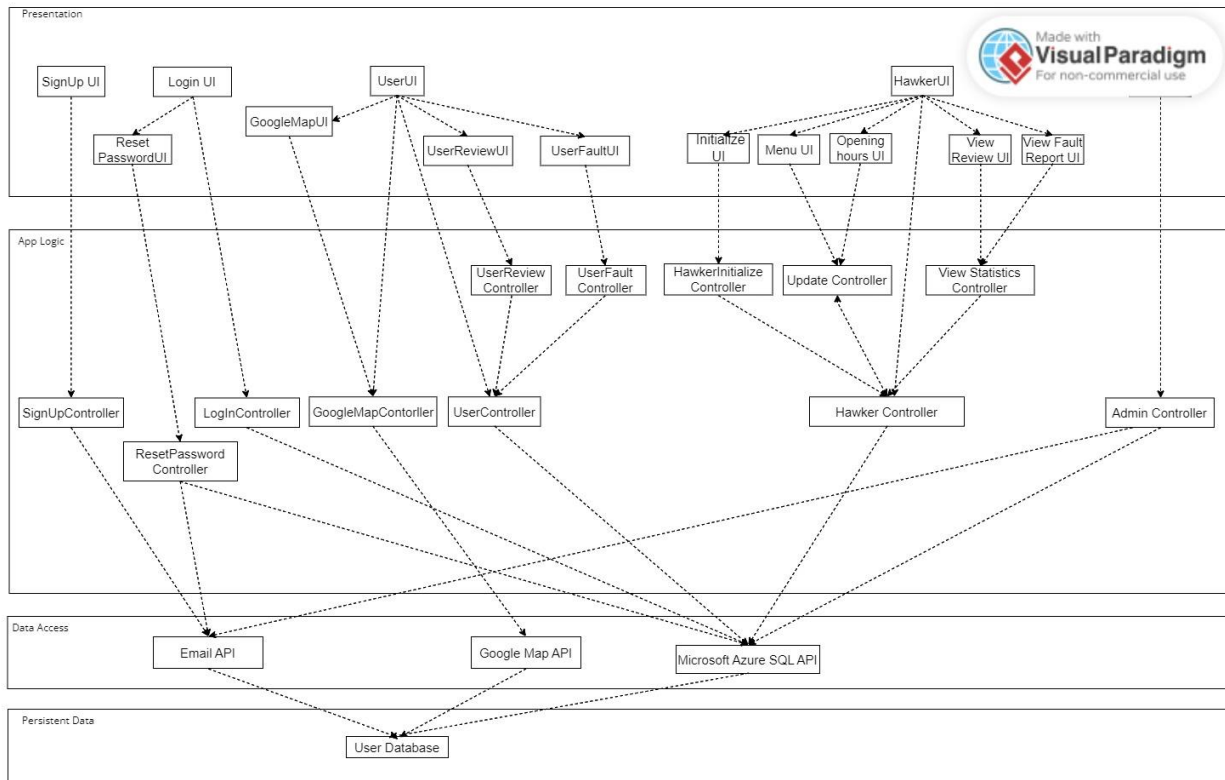




# Dialog Map



# System Architecture



# Application Skeleton

## 1. Frontend

- We use HTML to construct the layout of our system, and CSS to design our system.

## 2. Backend

- We use Azure SQL Database as database of our system.
- We connect to Azure SQL Database with PHP using *sqlsrv*.
- We use JavaScript to make the logic behind our system.

## 3. API used

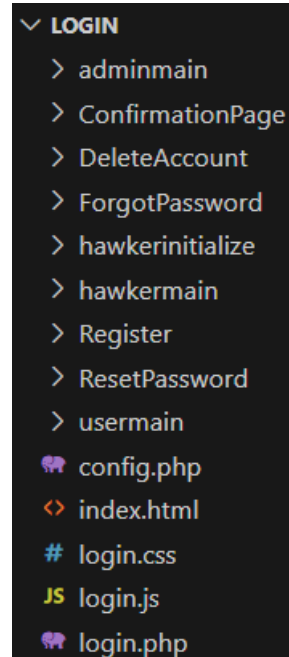
- Google Map API: Use in main page of user to check location of the hawker center
- EmailJS API: Use for sending email to users and hawkers after account registration and 'reset password when forgot' function

## 4. Data use

- We use the data "List of Government Markets Hawker Centres" in [data.gov.sg](https://data.gov.sg) to get the address of hawker centres and mark it on our map

## 5. Folder hierarchy

- Each folder is considered a webpage with both frontend and backend
- "Login/index.html" is the entry point of the frontend of our system



```
▼ LOGIN
  > adminmain
  > ConfirmationPage
  > DeleteAccount
  > ForgotPassword
  > hawkerinitialize
  > hawkermain
  > Register
  > ResetPassword
  > usermain
  🐘 config.php
  <> index.html
  # login.css
  JS login.js
  🐘 login.php
```

1 Folder hierarchy of our system

# Design Pattern and Considerations

The MVC pattern is ideal for HawkerStalk because our app needs to handle user interactions, manage complex data, and display dynamic content such as reviews, fault reports, and hawker stall details. The MVC design pattern allows the app to manage complex data, diverse user interactions, and scalability demands. It creates a clear, maintainable structure that allows us to scale the functionalities of our app while keeping our code clean, organized, and easy to debug.

The MVC design pattern separates HawkerStalk's functionalities into three key concerns:

Concern	Details
Model	<p>The model primarily contains HawkerStalk's business logic. It manages data like user accounts, hawker stalls, reviews, and fault reports. In HawkerStalk, this layer will handle (but not limited to):</p> <ul style="list-style-type: none"><li>• Verifying account creation for users and hawkers</li><li>• Storing reviews and fault reports</li><li>• Managing hawker stall data such as opening hours, menus, and crowd levels</li></ul>
View	<p>The view manages how data is presented to users. HawkerStalk, this layer will handle (but not limited to):</p> <ul style="list-style-type: none"><li>• Displays user-friendly interfaces, such as the review submission page, stall information pages, and fault reporting screens.</li><li>• Show dynamic data like crowd levels, ratings, and reviews in an intuitive and visually appealing manner.</li></ul>
Controller	<p>The controller bridges the gap between the view and model. It handles user input, processes data from the model, and updates the view accordingly. In HawkerStalk, the controller would (but not limited to):</p> <ul style="list-style-type: none"><li>• Process form submissions (sign up, login, reviews, fault reports)</li><li>• Validate data such as email format, password strength, and fault reports before passing it to the model for storage.</li><li>• Respond to user actions (e.g., clicking on a hawker stall icon to view details) by updating the view with relevant data.</li></ul>

Our decision to implement the MVC design pattern was due to its ability to streamline data flow, manage complex user interactions, and support the evolving needs of our platform. In the following sections, we will explain the key reasons why we chose the MVC pattern and how it benefits HawkerStalk in terms of scalability, real-time data management, and multi-user functionality.

Firstly, the MVC pattern allows for excellent scalability and maintainability. As HawkerStalk grows and more features are introduced, like notifications or payment integrations, the structure provided by MVC allows us to add new components without disturbing the existing functionality. Each core part—model, view, and controller—can be modified independently, making the app easier to maintain and extend. The enhanced flexibility would reduce the risk of introducing bugs and makes it easier for developers to collaborate and manage different sections of the app.

Secondly, the MVC pattern can handle real-time data efficiently. HawkerStalk relies on up-to-date information, like crowd levels at hawker stalls and the operating status of the stalls. With MVC, the model can retrieve and update real-time data from the database, while the controller ensures this data is delivered to the view immediately, allowing the app to stay dynamic and responsive. This means that users can always see the latest updates on stall activity, ensuring an interactive and engaging experience without delay.

Thirdly, the MVC pattern allows us to provide our two distinct user groups (customers and hawkers) distinct experiences in the app. This is necessary as both groups interact with the app in different ways. For example, hawkers can log in to update their stall's information, review customer feedback, or submit fault reports, while customers can search for stalls, submit reviews, and see real-time updates on stall activity. The controller in the MVC structure helps manage these various actions, ensuring that the right data is processed for each role. This flexibility allows HawkerStalk to cater to both user groups while maintaining a consistent and organized codebase.