SC2006 Software Engineering

Lab5 Final Deliverables

Group 26

Hawker Stalk

Done By:

Tan Ming Hao (U2320580D)

Choo Zhen Ming (U2320950J)

Cho Zhi Wei (U2320530C)

Chow Weng Shi (U2320760B)

Lai Xin Yee (U23206540G)

Swaminathan Navitraa (U231255K)

Pham Nguyen Vu Hoang (U2323430D)

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	30
Sequence Diagrams	35
Dialog Map	47
System Architecture	48
Application Skeleton	49
Design Pattern and Considerations	51
Black Box Testing	53
White Box Testing	59

Target users

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

Functional Requirements

1. The users must be able to create an account of their respective domain.
    1.1. The users can only create one account with one email address.
    1.2. The users must be able to click on the sign-up button.
    1.3. The customers 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, 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 customers 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 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, 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. Admin must be able to 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 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 users 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 8 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.
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 of stars.
    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 customers.
    3.5.1. The hawkers must be able to delete the fault reports after viewing.
3.6. The hawkers must be able to close their respective accounts about their stalls.
    3.6.1. The hawkers must input password two times correctly and click the "Confirm cancel" button to cancel their respective accounts.

4. The customers must be able to see the information about the stalls.
    4.1. The system must be able to show the location of the customer and hawker centers as a marker on the map.
    4.2. The customers must be able to click on the hawker center marker on the map.
    4.3. The customers must be able to see the name of a list of stalls after clicking on the hawker center icon.
    4.4. The customers must be able to see the opening hours and operating days of the stalls.
        4.4.1. The system must display the opening hours in 12-hour notation with AM and PM.
    4.5. The customers must be able to see the menu of the stalls after choosing the specific stall.
    4.6. The customers must be able to see the ratings of the stalls.
        4.6.1. The ratings should be displayed in number of stars.
    4.7. The customers must be able to see the reviews of the stalls.
        4.7.1. The customers must be able to click the "review" button under the stall information page.
    4.8. The customers must see the name of the hawker center where the stall is located.
    4.9. The customers must be able to see the exact address of the hawker center.
5. The customers must be able to make a fault report if there is any false information in the system.
    5.1. The system must require the customer to input at least 1 character into the text box to submit the fault report.
6. The customers must be able to provide their own reviews and ratings of the desired stall.
    6.1. The system must be able to display the options of providing ratings and reviews to the customers in the same section.
    6.2. The system must be able to display 5 empty stars on the screen when giving ratings.

6.2.1. 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.

6.2.2. 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.

6.2.3. 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.

6.3. The system must be able to display a text box for the customers to input their reviews in.

6.3.1. 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.

6.3.2. 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.

7. The customers must be able to search the name of the stall for the location of the desired hawker center.
    7.1. The customers input must contain at least 1 character but less than 512 characters for the search results.
    7.2. The system must be able to show relevant results when the customers type in at least one character.
    7.3. The system must be able to display the location of the hawker center on the map.
8. The admin must be able to manage all the accounts of user
    8.1. The admin must be able to suspend account of certain user
    8.2. The admin must be able to activate account of certain user

Non-functional Requirements

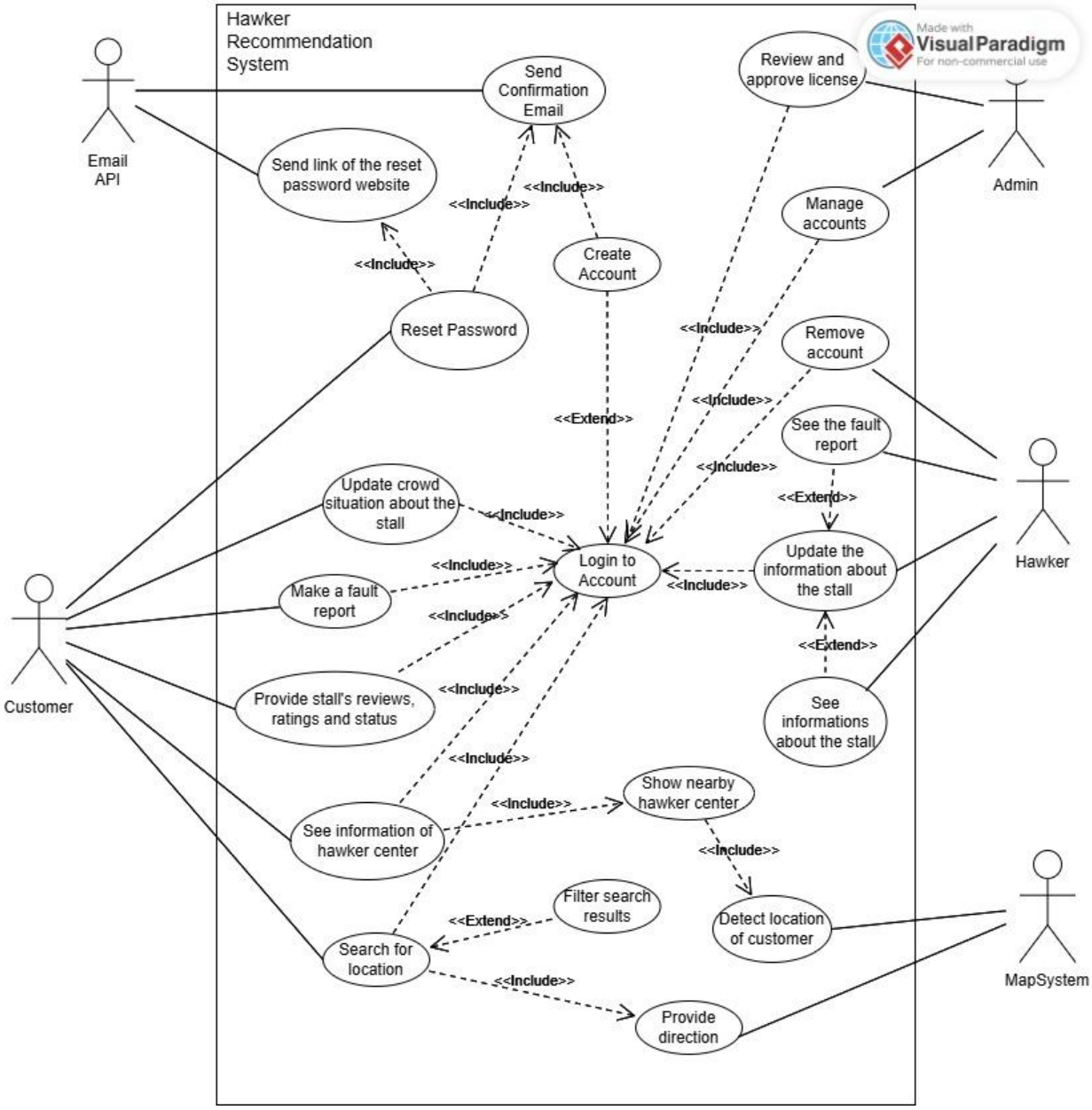
1. Security
    1.1. Personal and financial data (identification details, contact information, device information) of the stalls' owner should be encrypted to prevent unauthorized access.
    1.2. User data, including location information, search history, and personal details, should be safeguarded and 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)
Customers	The users who use the system to find food of hawker
System	The application
Hawker center	The place that consists of stalls that sell food
Hawker	The owner of the stall
Admin	The person with the highest permissions
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 customers to show the satisfaction to the stall
Reviews	Comments given by the customers
Empty	Indicate the icons that are filled with white color, indicating their non-chosen status

Use Case Diagram

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



Use Case Description

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:	• Customer • Email API
Description:	Creating a new customer account
Preconditions:	• System must have a stable connection to the database. • The email address used must not already exist in the database.
Postconditions:	• Show “Account created successfully” message. • Store data of customer in database
Priority:	High
Frequency of Use:	1-3 times per lifetime
Flow of Events:	1. Customer clicks on the sign-up button. 2. Customer enters a valid email address, password and password confirmation fields. 3. Customer click the create account button. 4. System validates the format of the email address. 5. System validates the validity of the email address in database 6. System validates the fulfillment of the requirement of the password. 7. System validates the password and password confirmation field are identical.

	8. System displays “Thank you for signing up” message upon successful creation of a new customer account. 9. Customer ask the system to send a confirmation email to the email address.
Alternative Flows:	AF-S4: System detects the invalid email address format. 1. System displays error message “Invalid email format”. 2. System returns to Step 2. AF-S5: Email address exists in database: 1. System displays error message “Email exists, do you want to login?” 2. System returns to Step 2. AF-S6: System detects password does not fulfill the password requirements. 1. System displays error message “Invalid password”. 2. System returns to Step 2. AF-S7: System detects mismatch between password and password confirmation fields. 1. System displays error message “Password mismatch”. 2. System returns to Step 2.
Exceptions:	-
Includes:	• Validate the account availability.
Special Requirements:	-
Assumptions:	• Database refers to the system database.
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:	• Hawker • Admin • Email API
Description:	Creating a new hawker account
Preconditions:	• The email address used must not already be in the database. • System must have a stable connection to the database.
Postconditions:	• Show “pending” status upon submission of request for creating account. • Show “Account created successfully” message. • Store data of customer in database
Priority:	High
Frequency of Use:	1-3 times per lifetime
Flow of Events:	1. Hawker clicks on the sign-up button. 2. Hawker enters a valid email address, password and password confirmation fields. 3. Hawker uploads a valid operating license. 4. Hawker selects the create account button. 5. System validates the format of the email address. 6. System validates the validity of the email address in database 7. System validates the fulfillment of the requirement of the password.

	8. System validates the password and password confirmation field are identical. 9. System display “pending” message. 10. Reviewer reviews the operating license and approves the account creation request. 12. System sends confirmation email to the registered email address.
Alternative Flows:	AF-S5: System detects invalid email address format. 1. System displays error message “Invalid email address format”. 2. System returns to Step 2. AF-S6: Email address exists in database: 3. System displays error message “Email exists, do you want to login?” 4. System returns to Step 2. AF-S7: System detects password does not fulfill the password requirements. 1. System displays error message “Invalid password”. 2. System returns to Step 2. AF-S8: System detects mismatch between password and password confirmation fields. 1. System displays error message “Password mismatch”. 2. System returns to Step 2. AF-S10: Reviewer disapproves the account creation request. 1. System sends “License disapproved, account creation failed” notification. 2. System returns to Step 2.

Exceptions:	-
Includes:	• Validate the account availability
Special Requirements:	-
Assumptions:	• Database refers to the system database
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:	• Customer • Hawker
Description:	Login to account using the registered email address and password
Preconditions:	• User account must already exist in the database. • System must have a stable connection to the database. • User connects to the system.
Postconditions:	• Users can see the main menu of the system. OR • Users see an error message. • Users can re-login to their accounts.
Priority:	High
Frequency of Use:	1 – 20 times per day
Flow of Events:	1. System requires the input of email address and password. 2. User input the email address and password in the login interface. 3. User clicks on the login button. 4. System verifies that email address and password have been filled out. 5. System retrieves the information from the database.

	6. System verifies the email address and password with the information retrieved from the database. 7. 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. 1. System displays the message “Please fill out this field”. 2. System returns to Step 2. AF-S6: If the email address does not exist in the database. 1. System displays the message “Invalid email or password.”. 2. System returns to Step 2. AF-S7: If the email address and password does not match the information in the database. 1. System displays the message “Invalid email address or password.”. 2. System returns to Step 2.
Exceptions:	-
Includes:	• Verify Login Credentials.
Special Requirements:	-
Assumptions:	• Database can be referred to System’s database.
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:	• Customer • Hawker • Email API

Description:	To reset password when user forgot their password
Preconditions:	• User account must already exist in the database. • System must have a stable connection to the database.
Postconditions:	• Users can re-login to their accounts.
Priority:	Low
Frequency of Use:	1-5 times per lifetime
Flow of Events:	1. User clicks on the “forgot password" button at the login page. 2. System requests user to enter the email address used to register for the account. 3. System verifies that the email address exists in the database. 4. System sends the user a link to the reset password website through email. 5. User clicks on the reset password website link. 6. User inputs a new valid password and password confirmation fields. 7. System directs the user back to the main page.
Alternative Flows:	AF-S3: Email address does not exist in the database. 1. System shows message “Invalid email address”. 2. System returns to Step 2. AF-S6: System detects password does not fulfill the password requirements. 1. System displays error message “Invalid password”. 2. System returns to Step 6. AF-S6: System detects mismatch between password and password confirmation fields.

	1. System displays error message “Password mismatch”. 2. System returns to Step 6.
Exceptions:	-
Includes:	• Verify Login Credentials.
Special Requirements:	-
Assumptions:	• Database can be referred to System’s database.
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:	Lai Xin Yee
Date Created:	30-8-2024	Date Last Updated:	30-10-2024

Actor:	• Hawkers • Customers
Description:	Hawkers update information of their respective stall
Preconditions:	• System must have a stable connection to the database. • Hawker account must exist in the database. • Hawkers want to update the information of their respective stall after seeing its information. OR • 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:	1. Hawker login to their respective account.

	2. The system shows the "Fault Report" icon. 3. The system shows the "Review" icon. 4. The system shows the "Menu" icon. 5. The system shows the "Opening hours" icon. 6. The system shows "Open Shop" and "Close Shop" button. 7. The system shows "Shut Down Shop" button. 8. The system shows a red "X" icon. 9. System updates the information of the stall.
Alternative Flows:	AF-S2: The hawker clicks on the fault report icon. 1. The system displays all the reporters' fault reports. 2. The system will display an exit button and a button to delete the notification. 3. The system will remove the report if the customer clicks on the button to delete the notification. 4. The system returns to Step 3 upon exit. AF-S3: The hawker clicks on the review icon. 1. The system displays all the newly added reviews. 2. The system displays an exit button. 3. The hawker clicks on the exit button. 4. The system returns to Step 4 upon exit. AF-S4: The hawker clicks on the "Menu" button. 1. The system will show a page for the hawker to edit the menu. 2. The hawker edits the menu of the stalls. 3. The hawker clicks the save button.

	4. The system returns to Step 5 upon exit. AF-S5: The hawker clicks on the “Opening hours” button. 1. The system will show a page for the hawker to edit the opening hours and operating days. 2. Hawker edits the opening time and operating days. 3. Hawker clicks on the “Save” button. 4. The system returns to Step 6 upon exit. AF-S7: The hawker clicks on the “Shut Down” button. 1. The system will show a confirmation page to delete the hawker account. 2. Hawker inputs two times his correct password. 3. Hawker clicks on the confirmation button. 4. The system deletes the account of the respective hawkers.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	• Information of the hawker’s stalls can be referred to System’s database.
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:	• Hawkers

Description:	Hawkers update the menu of the stalls of their respective stalls
Preconditions:	• System must have a stable connection to the database. • Hawker account must exist in the database. • 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:	1. The system shows a page which includes the menu of the shop. 2. The hawker clicks on the item at the right. 3. The system shows the detail of the item. 4. The system shows an “Add menu” button, “Delete” button, “Save” button and “Exit” button. 5. The system returns to the home page of hawker.
Alternative Flows:	AF-S4: The hawker clicks on the “Add menu” button. 1. The systems show a new row for the hawker to input a new menu item. 2. The hawker fills in the details of the new item which include picture, name, description and price. 1. The hawker clicks on the “Save” 5. The system adds the item to the database of the store. 6. The system returns to Step 4 upon exit. AF-S4: The hawker clicks on the “Delete” button. 1. The systems remove the item. 2. The hawker clicks on the “Save” button. 3. The system updates the details of the item in the menu. 4. The system returns to Step 4 upon exit.

	AF-S4: The hawker clicks on the “Exit” button. 1. The system returns to Step 5.
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:	30-10-2024

Actor:	• Hawkers
Description:	Hawkers close their account and their stall.
Preconditions:	• System must have a stable connection to the database. • Hawker account must exist in the database. • Hawkers click on the "Shub Down Shop" button.
Postconditions:	• The system will remove the hawker’s account and its information from the database. • The hawker account is removed successfully. • The hawker account no longer exists.
Priority:	Medium
Frequency of Use:	Once per lifetime

Flow of Events:	1. The system displays two text boxes and requires the hawker to enter their password correctly twice. 2. The hawker enters their password correctly twice. 3. The system verifies the correctness of the password. 4. The hawker clicks on the “Delete Account” button. 5. The system processes the request (delete account). 6. The system shows message (“Account delete successfully”) 7. The system to Sign Up Page.
Alternative Flows:	AF-S5: If the email address and password does not match the information in the database. 1. The system displays the message “Password is incorrect”. 2. The system returns to Step 1.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

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:	• Customer • Map system
Description:	Customer can view the information of a list of stalls after logging in to the account

Preconditions:	• Customer account must already login to the system. • System must have a stable connection to the database.
Postconditions:	• The system displays a list of stalls with required information.
Priority:	High
Frequency of Use:	1 – 20 times per day
Flow of Events:	1. Customer logins to respective account successfully. 2. The system displays the location of customer and all hawker centers on the map. 3. The customer clicks on a desired hawker center. 4. The system retrieves the information of all stalls in the chosen hawker center from the database. 5. 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. 1. The system displays the menu of the stall, with names and pictures of dishes and their corresponding prices. 2. The system returns to Step 6 upon exit. AF-S5: The customer clicks on the review button. 1. The system displays the overall ratings of the stall and reviews by other customers. 2. The system returns to Step 6 upon exit.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	• Information about the hawker center can be referred to in the System's database.

Notes and Issues:	-

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:	• 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:	• Customer account must already exist in the database. • System must have a stable connection to the database.
Postconditions:	• The record of the fault report has been stored in the database. • Hawker can see the submitted fault report.
Priority:	Medium
Frequency of Use:	0-5 times per day
Flow of Events:	1. Customer goes to the desired stall after searching for the information through the system. 2. Customer logins to respective account successfully. 3. Customer searches up the current hawker center and the specific stall. 4. The system displays the stall information and a report button. 5. The customer clicks on the report button. 6. 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. 7. The customer inputs the issues they found in the text box and submits the report. 8. The system displays “Fault report is submitted successfully.”. 9. 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. 1. The system displays message “The report must contain at least one character”. 2. The system returns to Step 7. AF-S6: The customer clicks on the information button. 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. 2. 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:	-

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:	• Customer
Description:	Customer provides reviews, and ratings of the stall they visited
Preconditions:	• System must have a stable connection to the database. • Customer account must exist in the database.
Postconditions:	• System shows “review/rating uploaded” message
Priority:	Medium
Frequency of Use:	1-2 times per day
Flow of Events:	1. Customer search and tap on the stall that they want to provide the review or rating. 2. Customer writes in their review and choose the number of stars for ratings in the review or rating section. 3. Customer taps the “Upload” button to upload the review or rating. 4. System shows “review/rating uploaded” message.
Alternative Flows:	AF–S2: Customer does not write in review and choose number of stars for ratings. 1. Customer only choose the number of stars for ratings and do not input any characters into the text box. 2. System returns to Step 3.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

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:	• Customer
Description:	To search for a desired location
Preconditions:	• System is connected to Wi-Fi or Mobile Data.
Postconditions:	• 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:	1. Customer enters query in the search bar. 2. Customer taps the search button. 3. System provides a list of relevant search results based on the customer's query.
Alternative Flows:	-
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

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:	• Admin
Description:	To suspend accounts of user

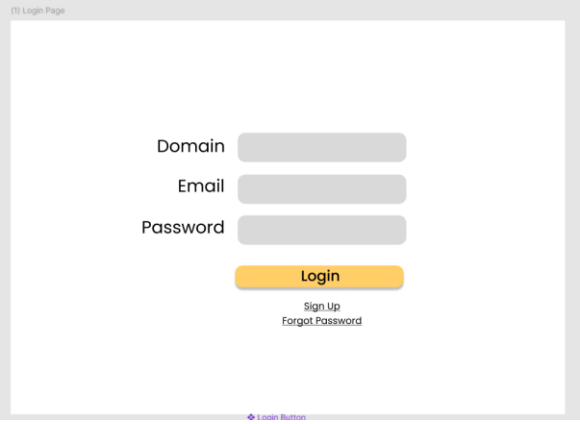
Preconditions:	• System is connected to Wi-Fi or Mobile Data.
Postconditions:	• Status of certain accounts change, the account cannot be used to login
Priority:	High
Frequency of Use:	0-5 times per day
Flow of Events:	1. Admin login to account 2. Admin click on “Suspend” button for the account 3. System asks admin to ensure to suspend the certain user 4. Admin click on “OK” 5. System changes the status of the user account to “suspended”
Alternative Flows:	AF-S3: Admin refuse to suspend the user account 1. Admin click on “Cancel” 2. System shows the user management page
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

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

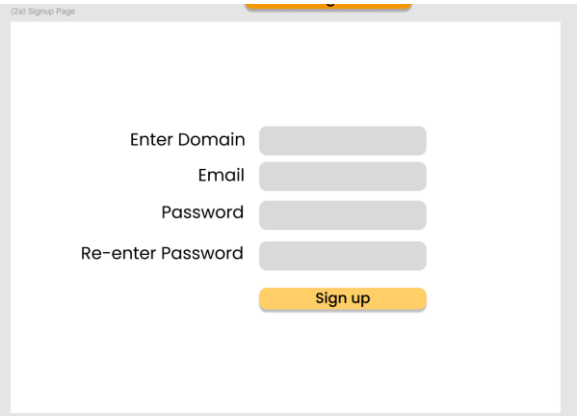
Actor:	• Admin
Description:	To activate accounts of user
Preconditions:	• System is connected to Wi-Fi or Mobile Data.
Postconditions:	• Status of certain accounts change, the account can be used to login
Priority:	High
Frequency of Use:	0-5 times per day
Flow of Events:	1. Admin login to account 2. Admin click on “Activate” button for the account 3. System asks admin to ensure to activate the certain user 4. Admin click on “OK” 5. System changes the status of the user account to “active”
Alternative Flows:	AF-S3: Admin refuse to activate the user 1. Admin click on “Cancel” 2. System shows the user management page
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

UI Mockup Diagram

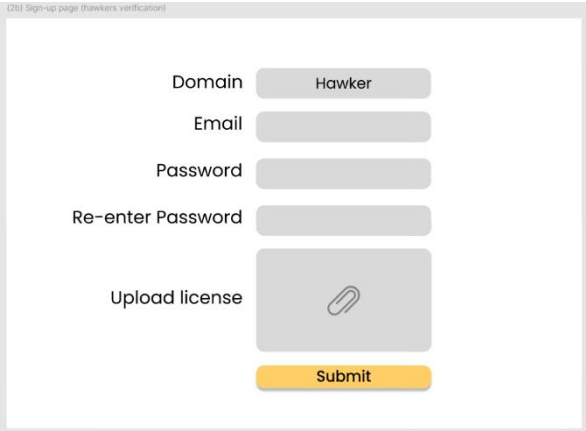
Login Page



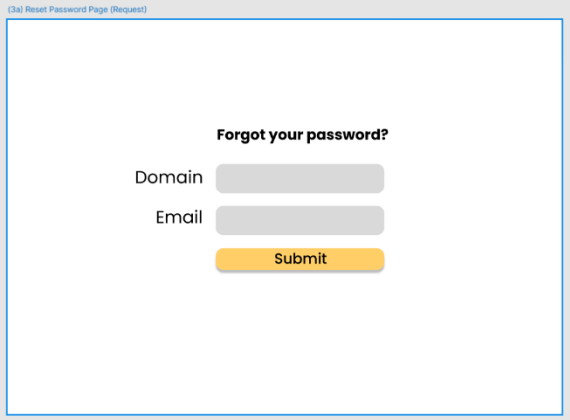
Sign Up Page



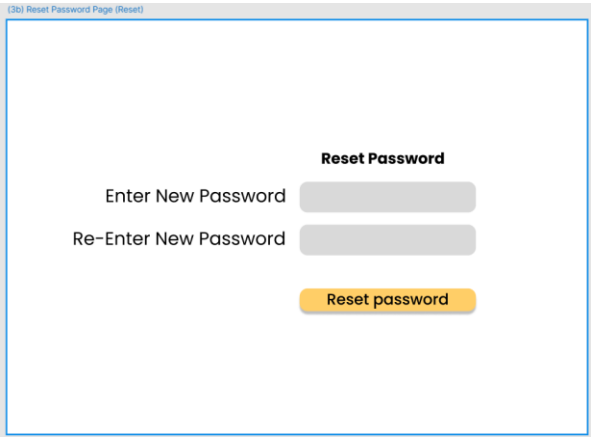
Sign Up Page (with Hawker Verification)



Forgot Password Page



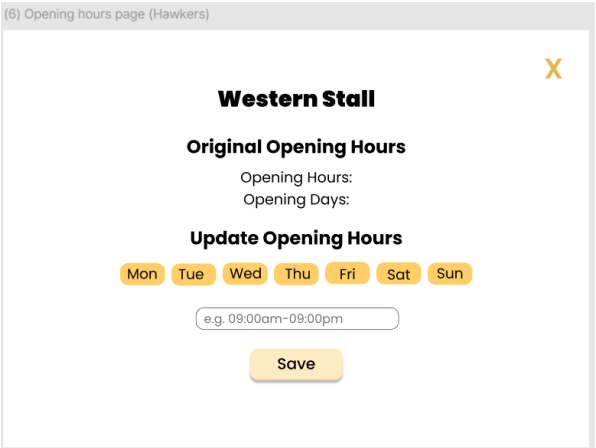
Reset Password Page



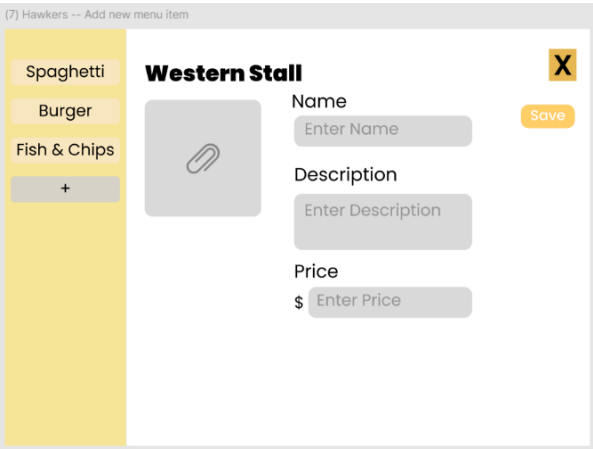
Hawker Main Page

Fault Report Page (Hawker)

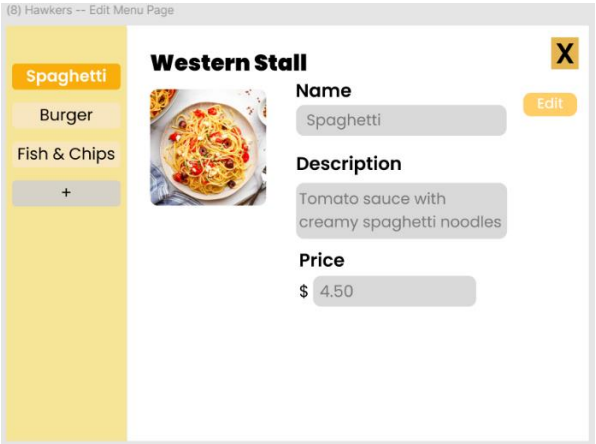
Opening Hours Page (Hawker)



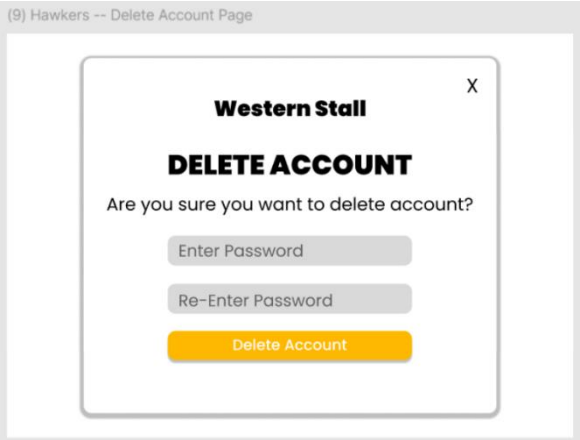
Hawker – Add new item



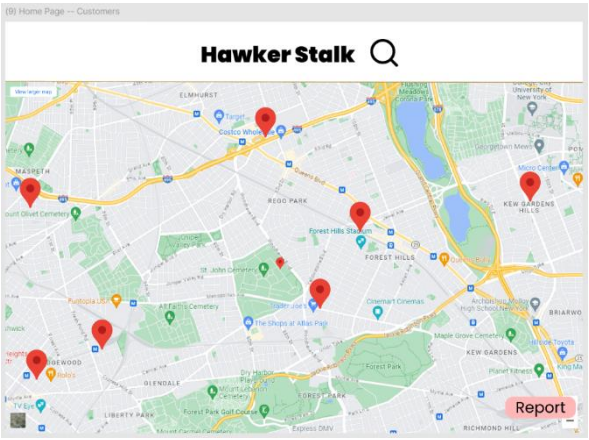
Hawker – Edit Menu



Hawker – Delete Account

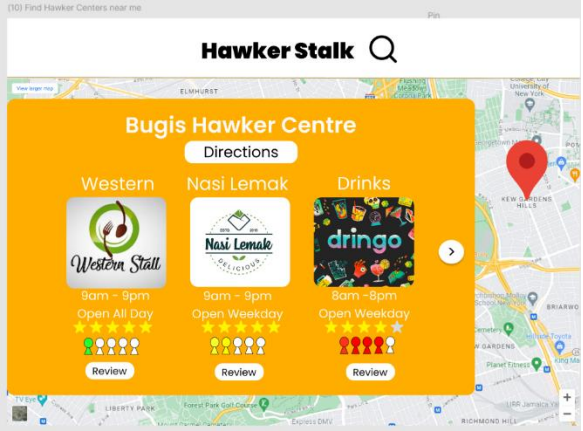


Customers – Main Page

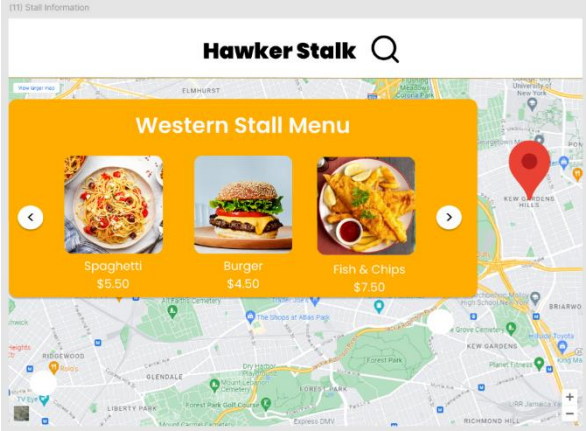


\

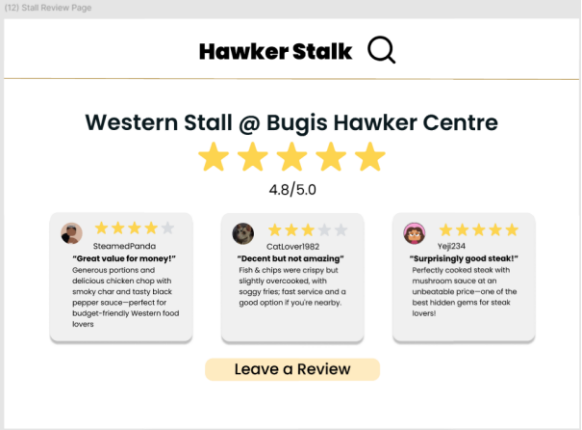
Find Hawker Center Near Me Page



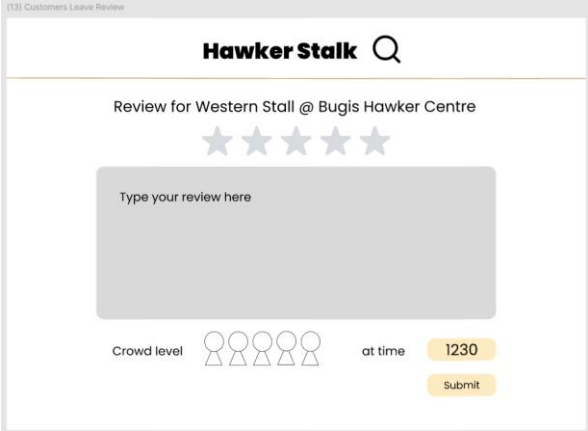
Stall Information Page



Stall Review Page

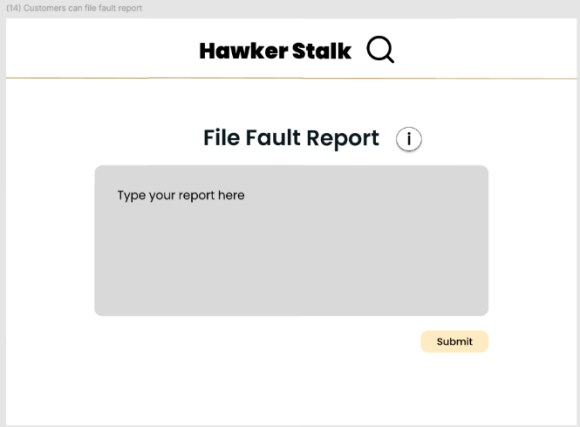


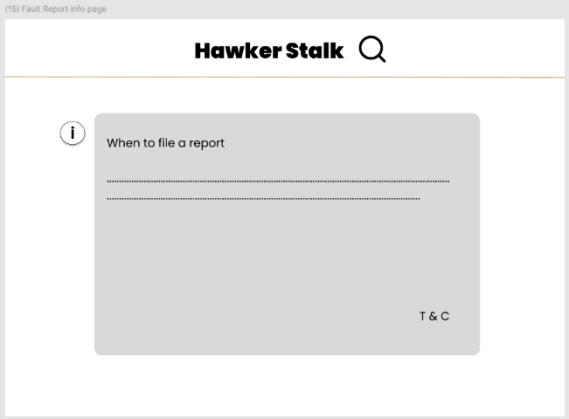
Customer Leave Review Page



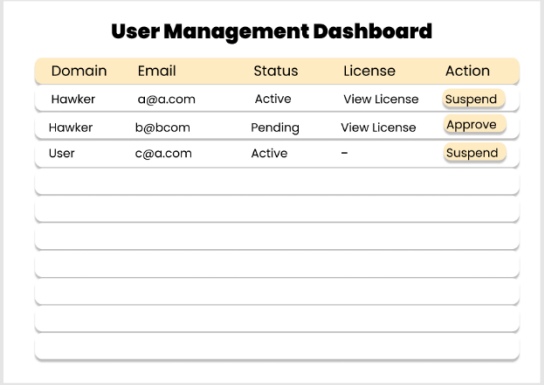
Hawker File Fault Report Page

Fault Report Information Page



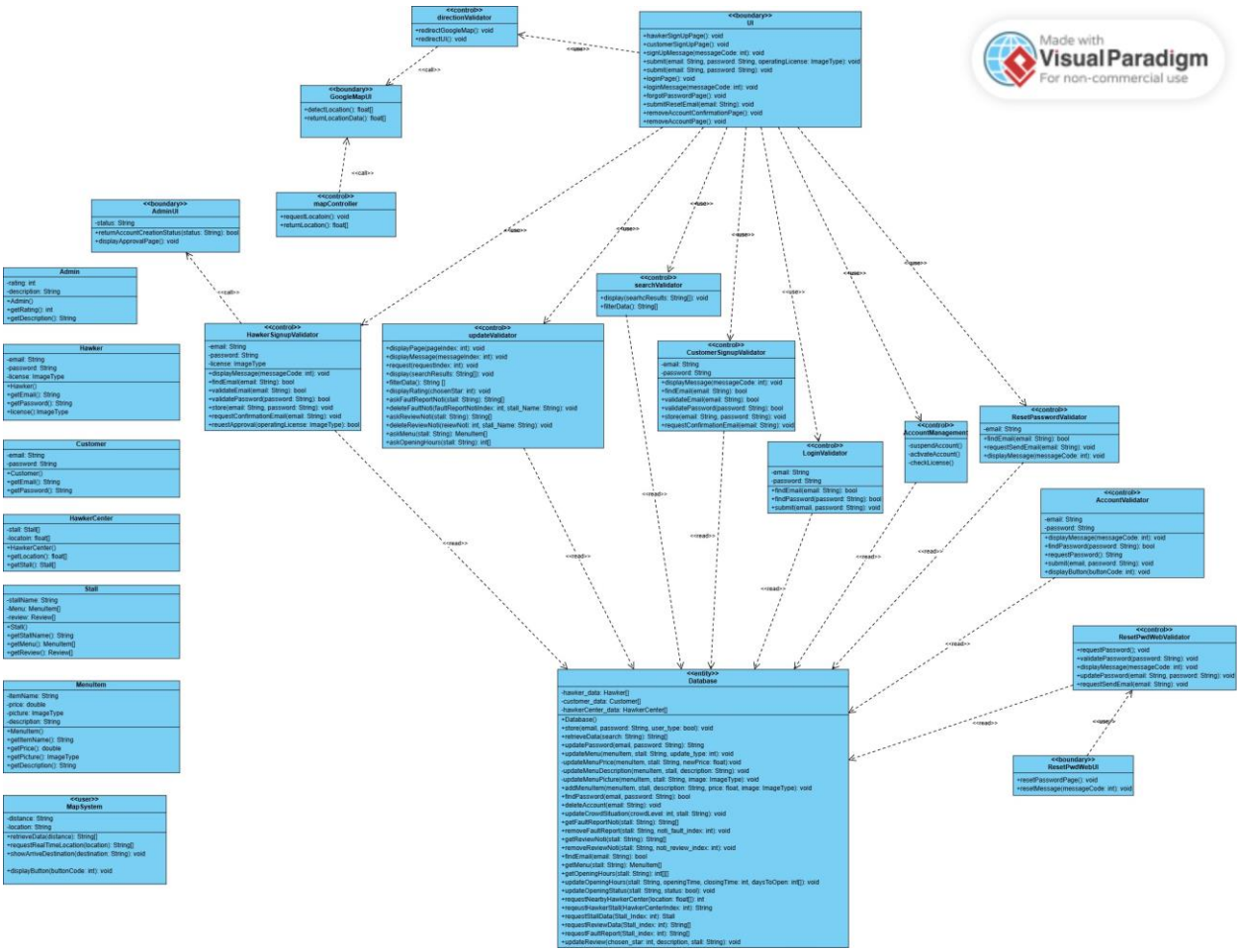


User Management Page (Admin)



Class Diagram

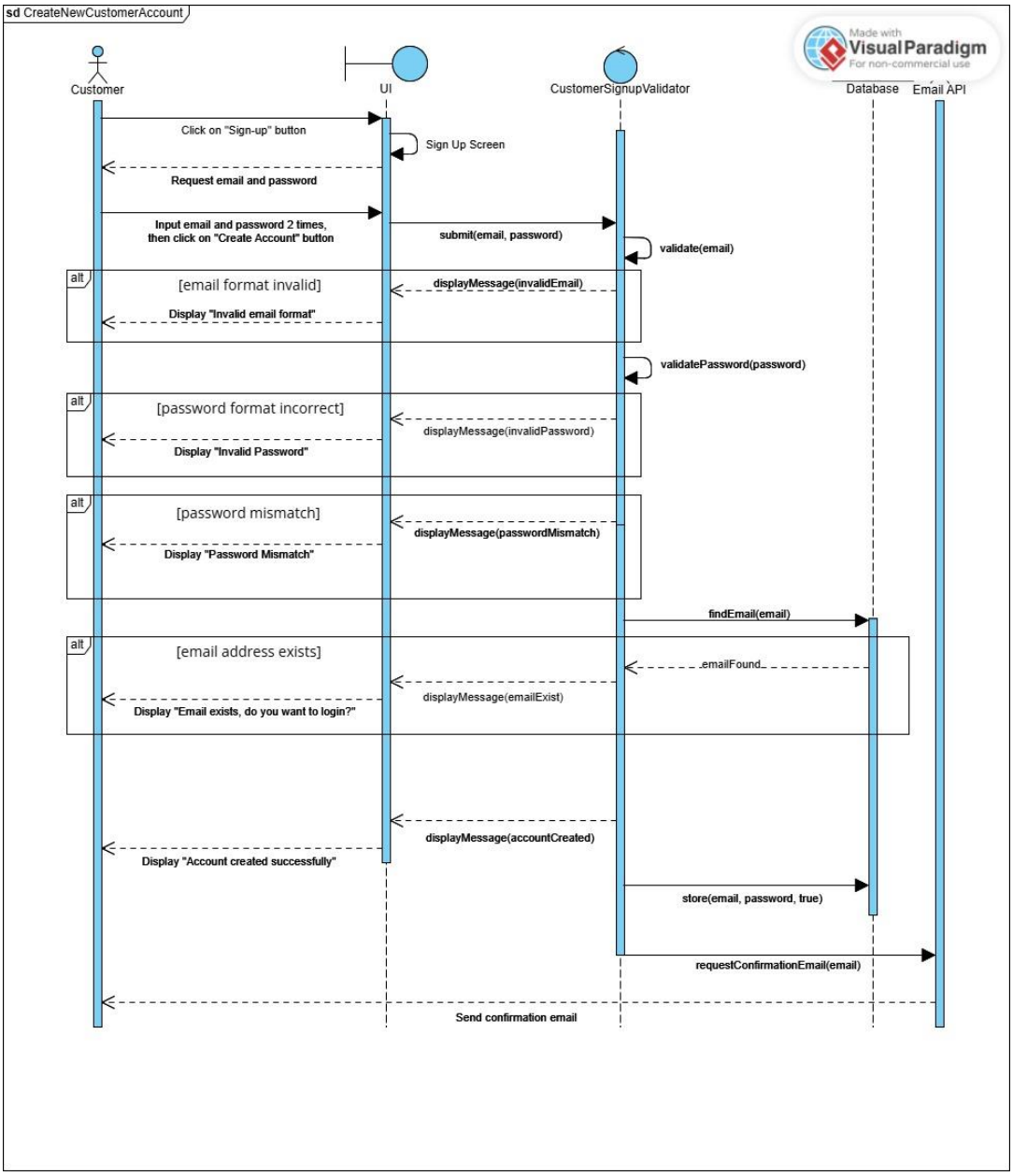
If the picture is uncleared, please refer to the picture attached in the same file



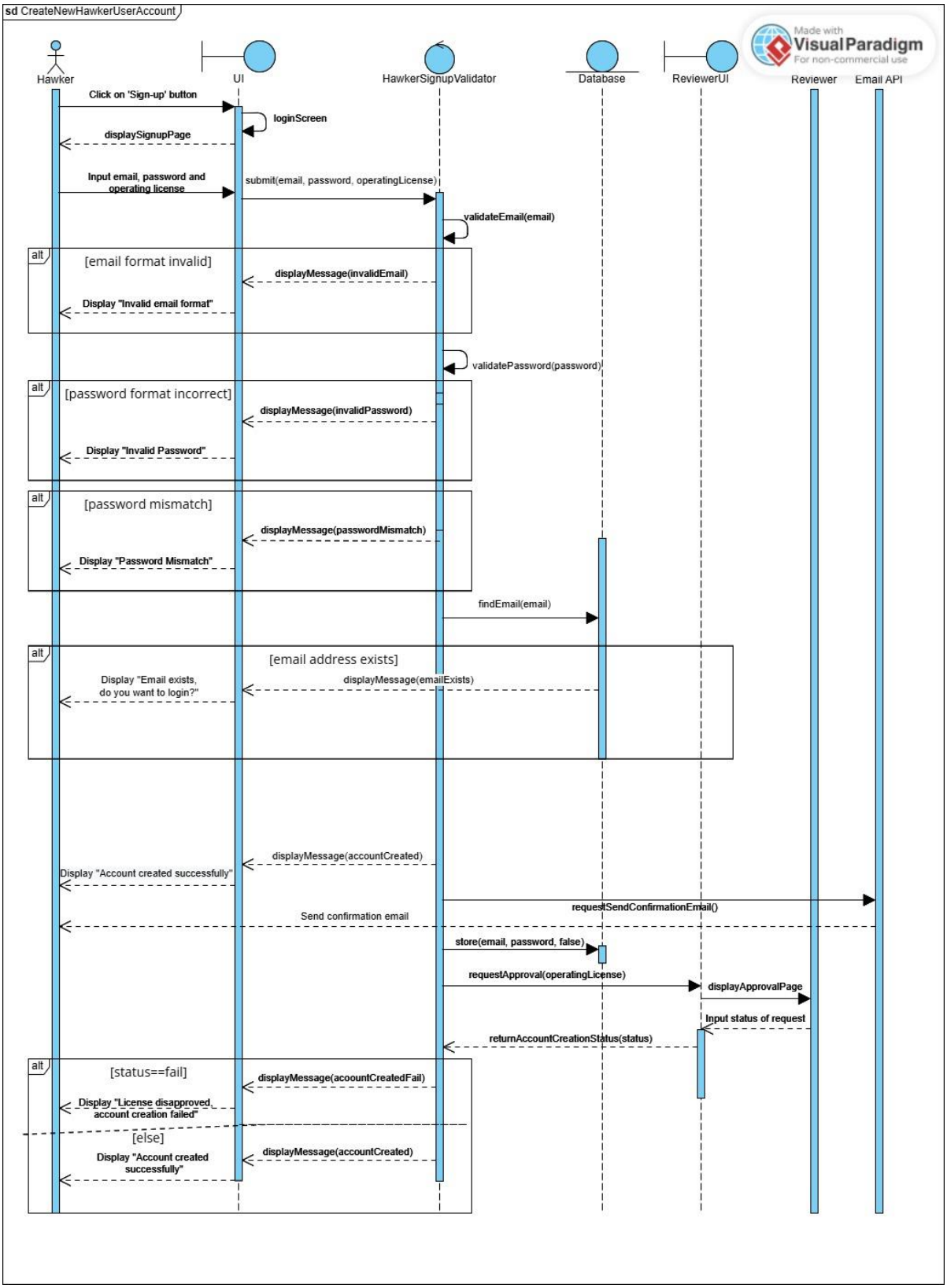
Sequence Diagrams

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

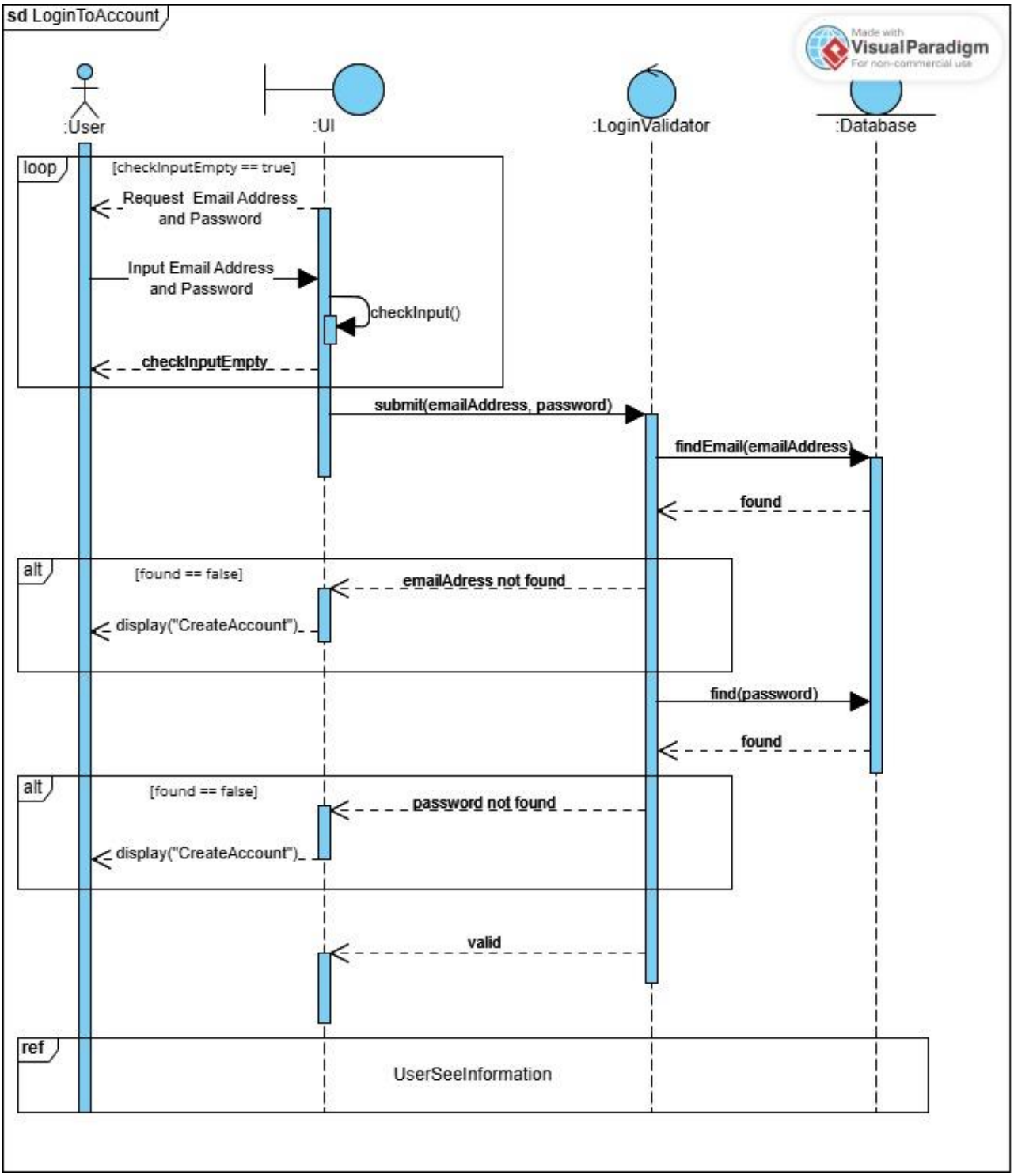
Use Case ID: 01A



Use Case ID: 01B

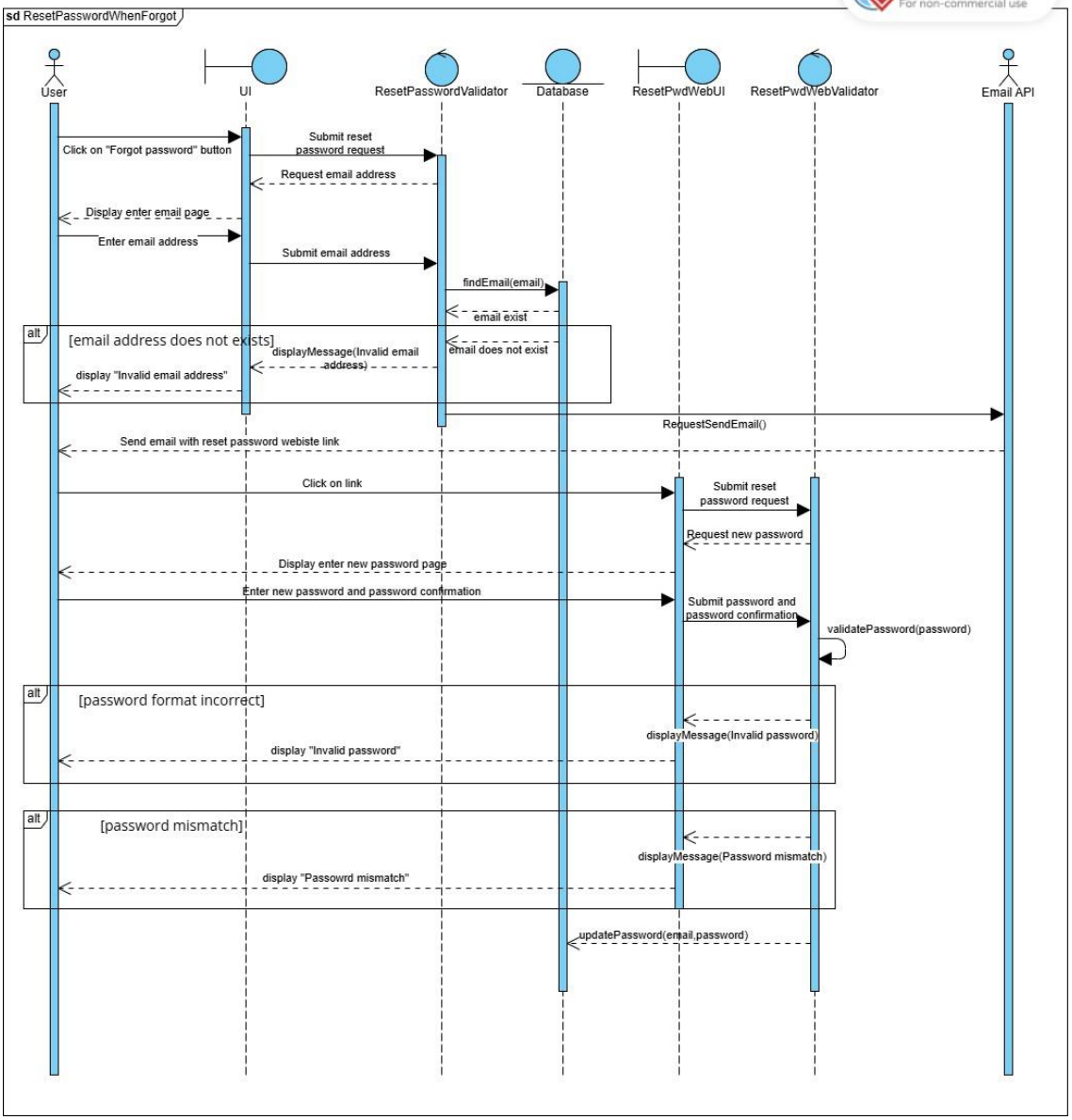


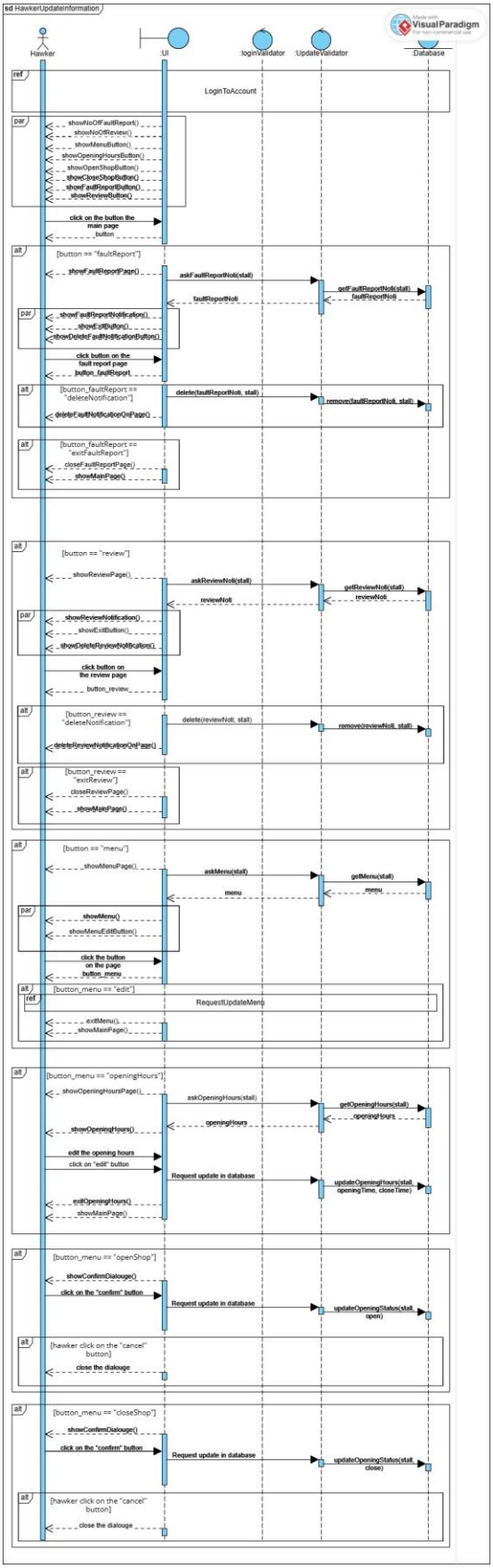
Use Case ID: 02A



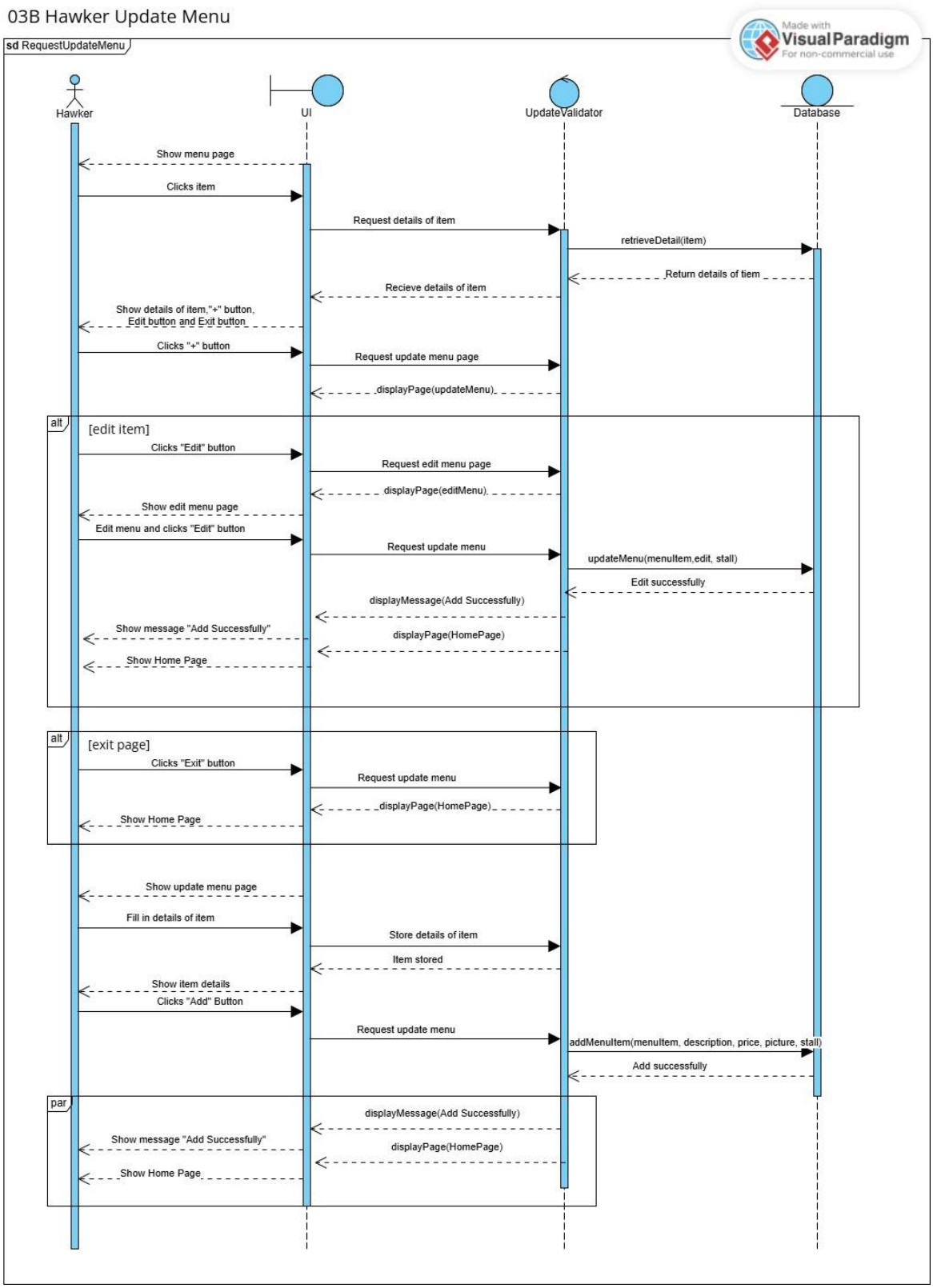
Use Case ID: 02B

02B Reset Password when Forgot



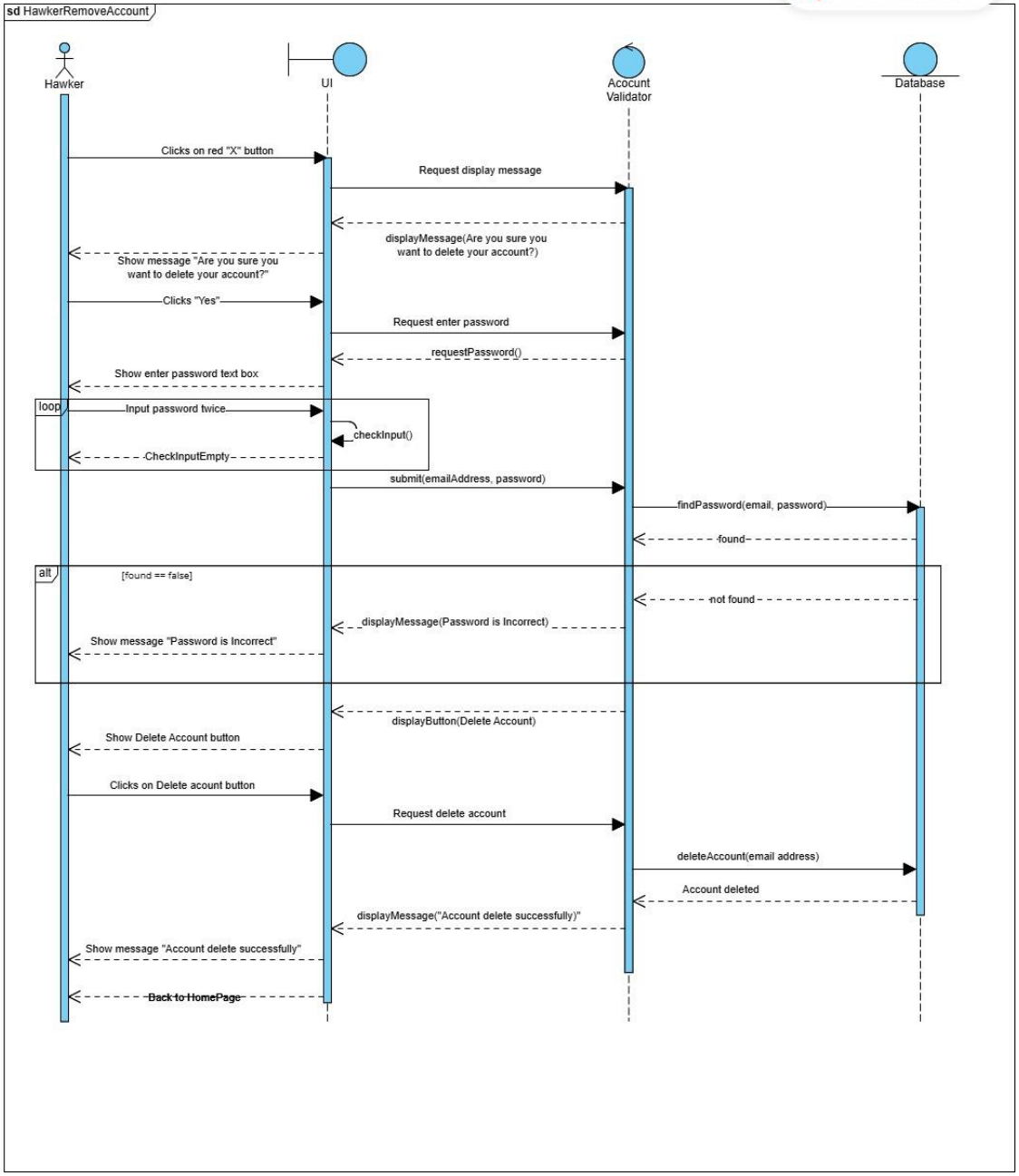

sd HawkerUpdateInformation
Hawker
:UI
:loginValidator
:UpdateValidator
Database
ref
LoginToAccount
par
showNoOfFaultReport()
showNoOfReview()
showMenuButton()
showOpeningHoursButton()
showOpenShopButton()
showCloseShopButton()
showFaultReportButton()
showReviewButton()
click on the button the main page
button
alt
[button == "faultReport"]
showFaultReportPage()
askFaultReportNoti(stall)
getFaultReportNoti(stall)
faultReportNoti
faultReportNoti
par
showFaultReportNotification()
showExitButton()
showDeleteFaultNotificationButton()
click button on the fault report page
button_faultReport
alt
[button_faultReport == "deleteNotification"]
delete(faultReportNoti, stall)
remove(faultReportNoti, stall)
deleteFaultNotificationOnPage()
alt
[button_faultReport == "exitFaultReport"]
closeFaultReportPage()
showMainPage()
alt
[button == "review"]
showReviewPage()
askReviewNoti(stall)
getReviewNoti(stall)
reviewNoti
reviewNoti
par
showReviewNotification()
showExitButton()
showDeleteReviewNotification()
click button on the review page
button_review
alt
[button_review == "deleteNotification"]
delete(reviewNoti, stall)
remove(reviewNoti, stall)
deleteReviewNotificationOnPage()
alt
[button_review == "exitReview"]
closeReviewPage()
showMainPage()
alt
[button == "menu"]
showMenuPage()
askMenu(stall)
getMenu(stall)
menu
menu
par
showMenu()
showMenuEditButton()
click the button on the page
button_menu
alt
[button_menu == "edit"]
ref
RequestUpdateMenu
exitMenu()
showMainPage()
alt
[button_menu == "openingHours"]
showOpeningHoursPage()
askOpeningHours(stall)
getOpeningHours(stall)
openingHours
openingHours
showOpeningHours()
edit the opening hours
click on "edit" button
Request update in database
updateOpeningHours(stall, openingTime, closeTime)
exitOpeningHours()
showMainPage()
alt
[button_menu == "openShop"]
showConfirmDialouge()
click on the "confirm" button
Request update in database
updateOpeningStatus(stall, open)
alt
[hawker click on the "cancel" button]
close the dialouge
alt
[button_menu == "closeShop"]
showConfirmDialouge()
click on the "confirm" button
Request update in database
updateOpeningStatus(stall, close)
alt
[hawker click on the "cancel" button]
close the dialouge

Use Case ID: 03B

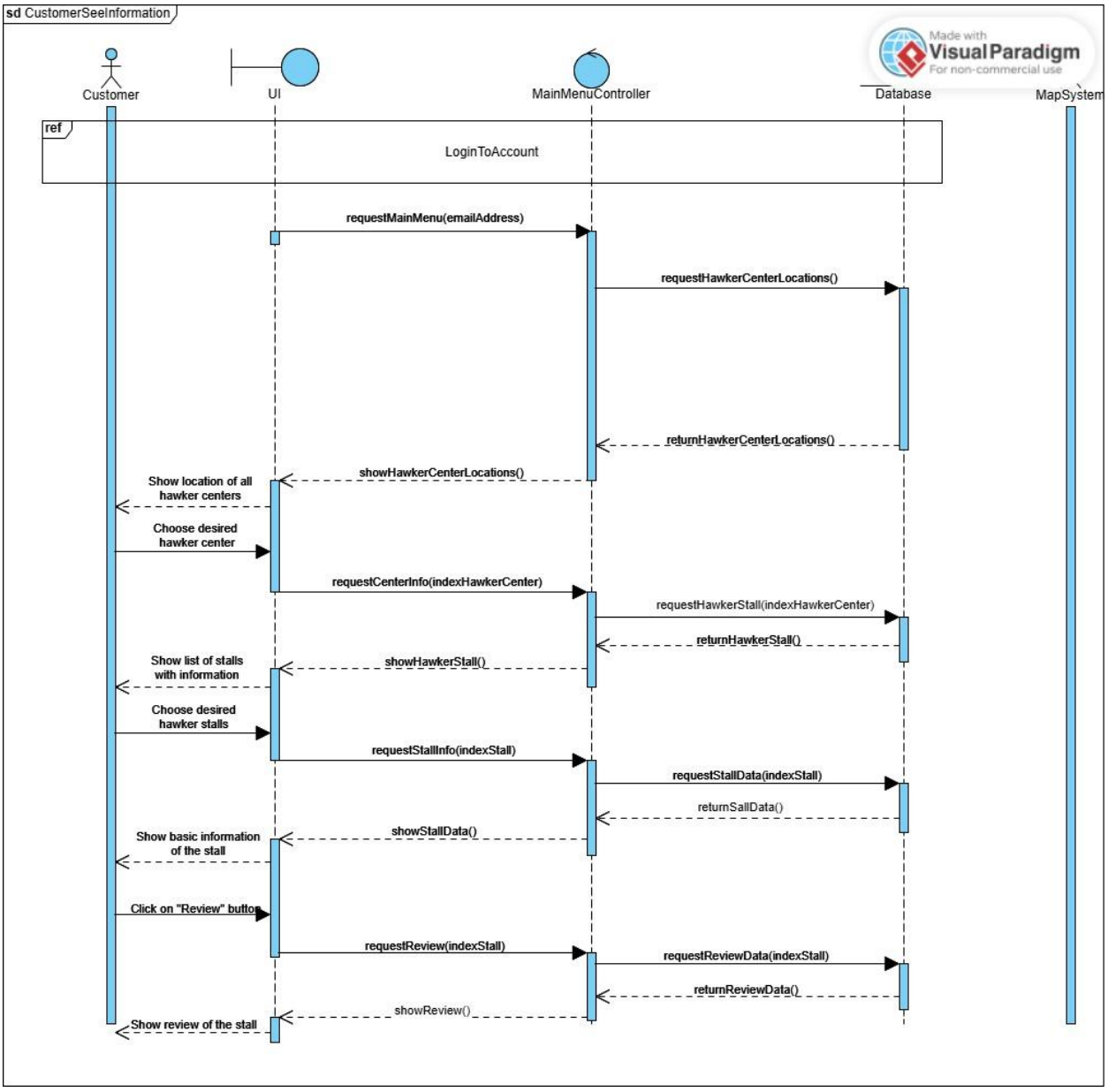


Use Case ID: 03C

03C Hawker Remove Their Account and Their Stall

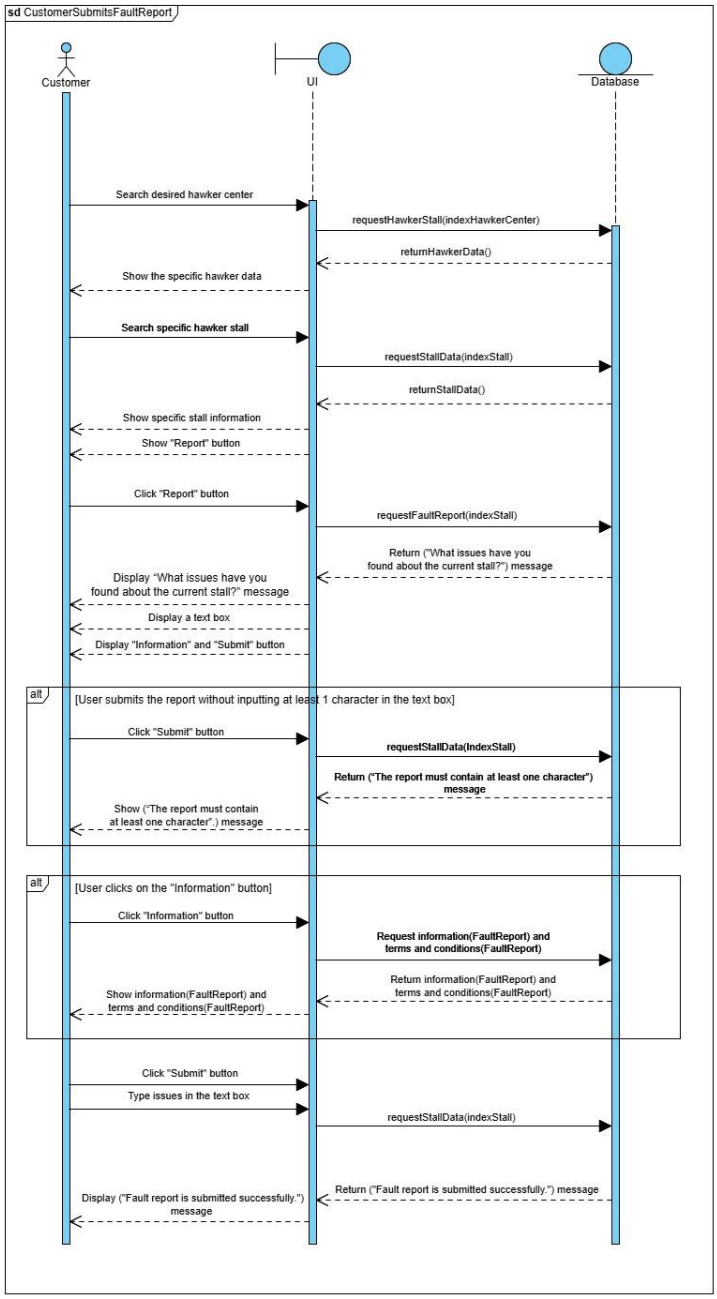


Use Case ID: 04A



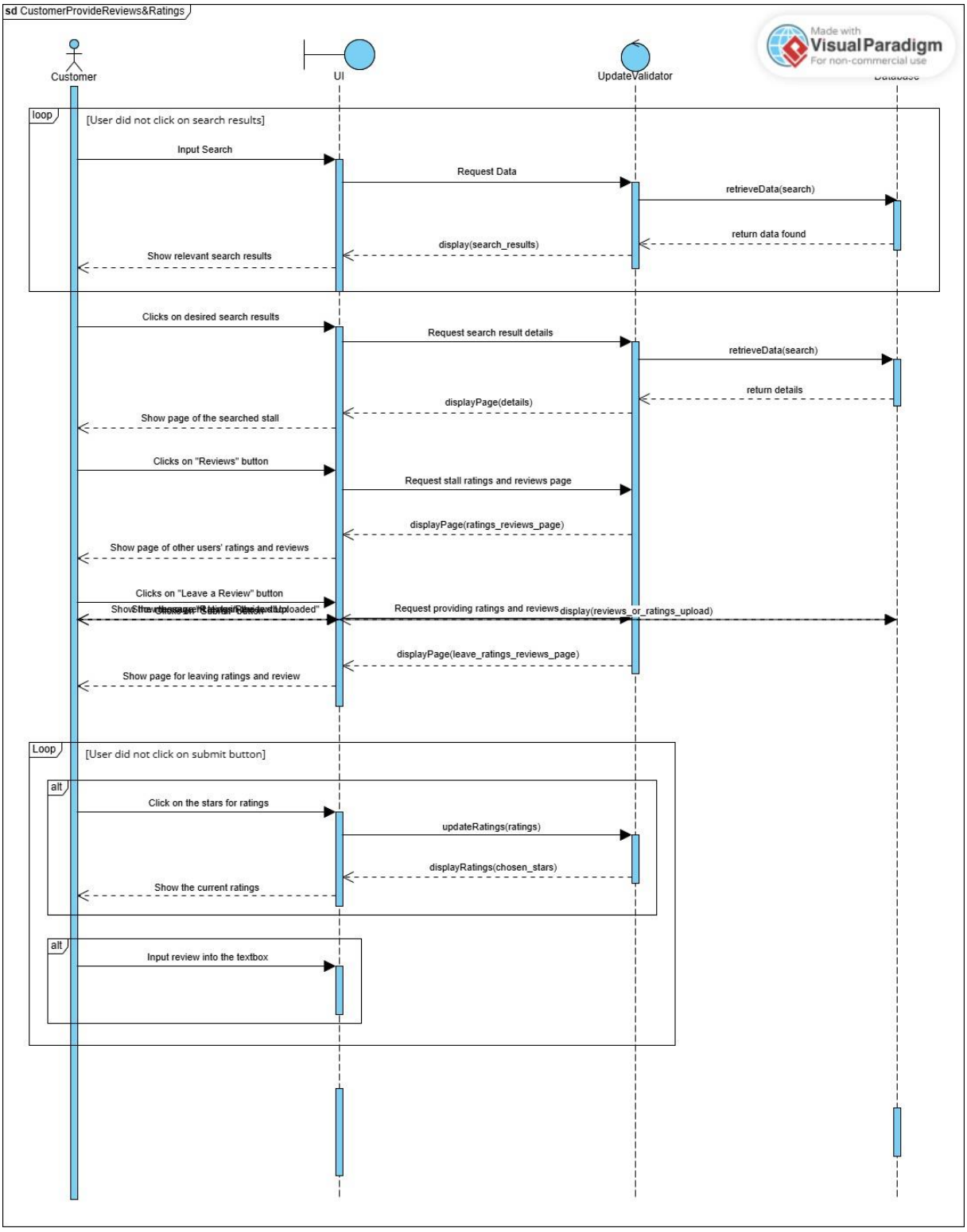
Use Case ID: 05A

05A Customer Submits Fault Report



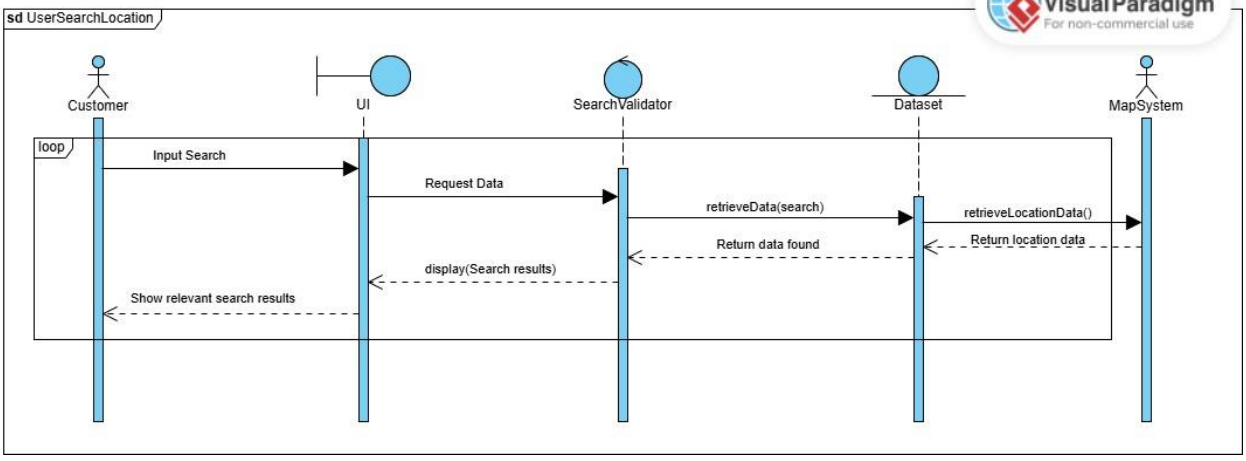
Made with VisualParadigm For non-commercial use

Use Case ID: 06A

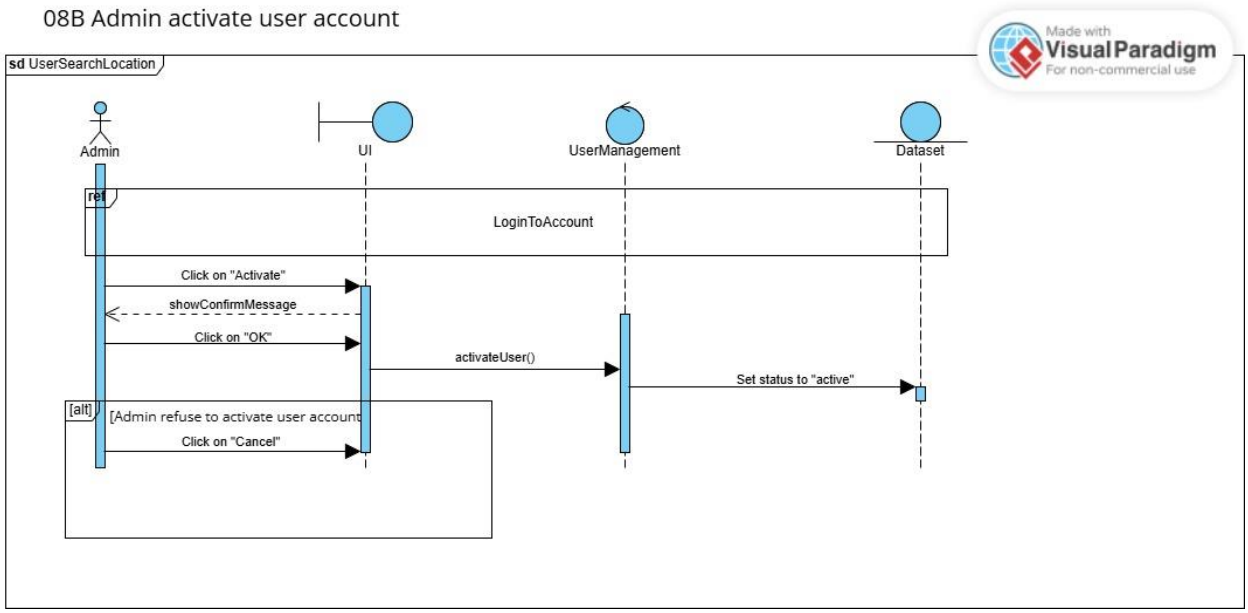


Use Case ID: 07A

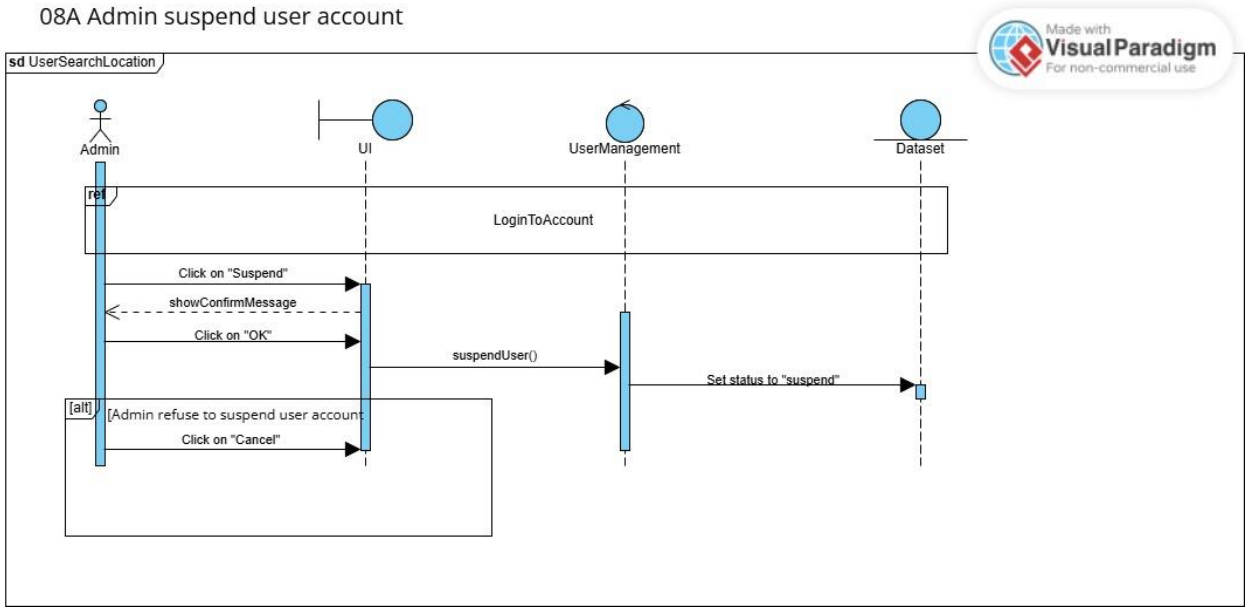
07A Customer Searches for a Location



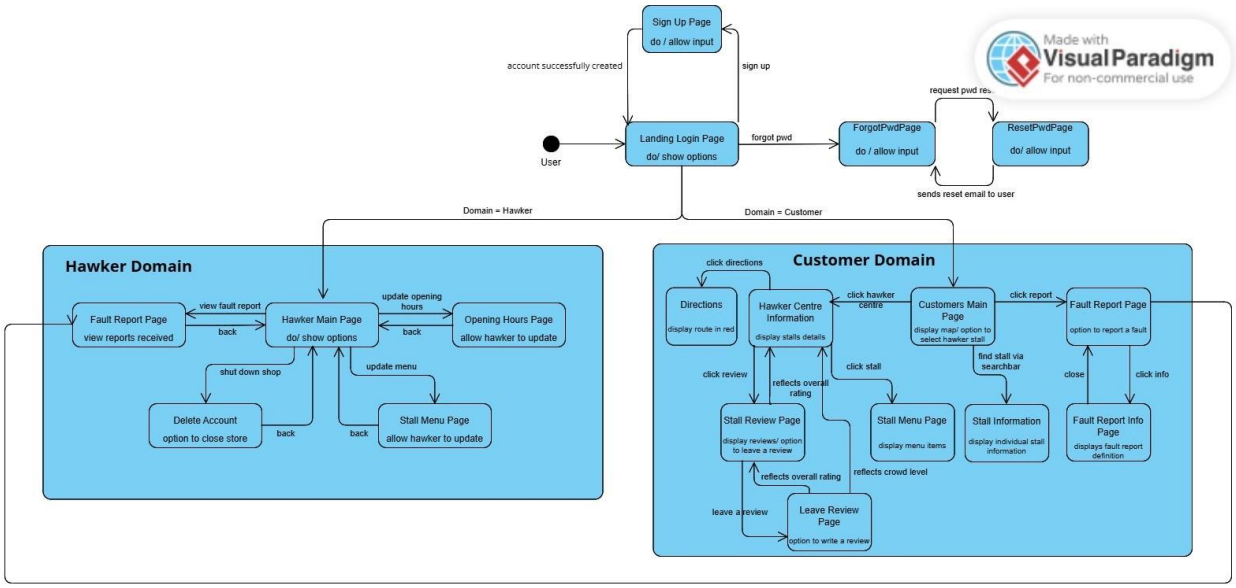
Use Case ID: 08A



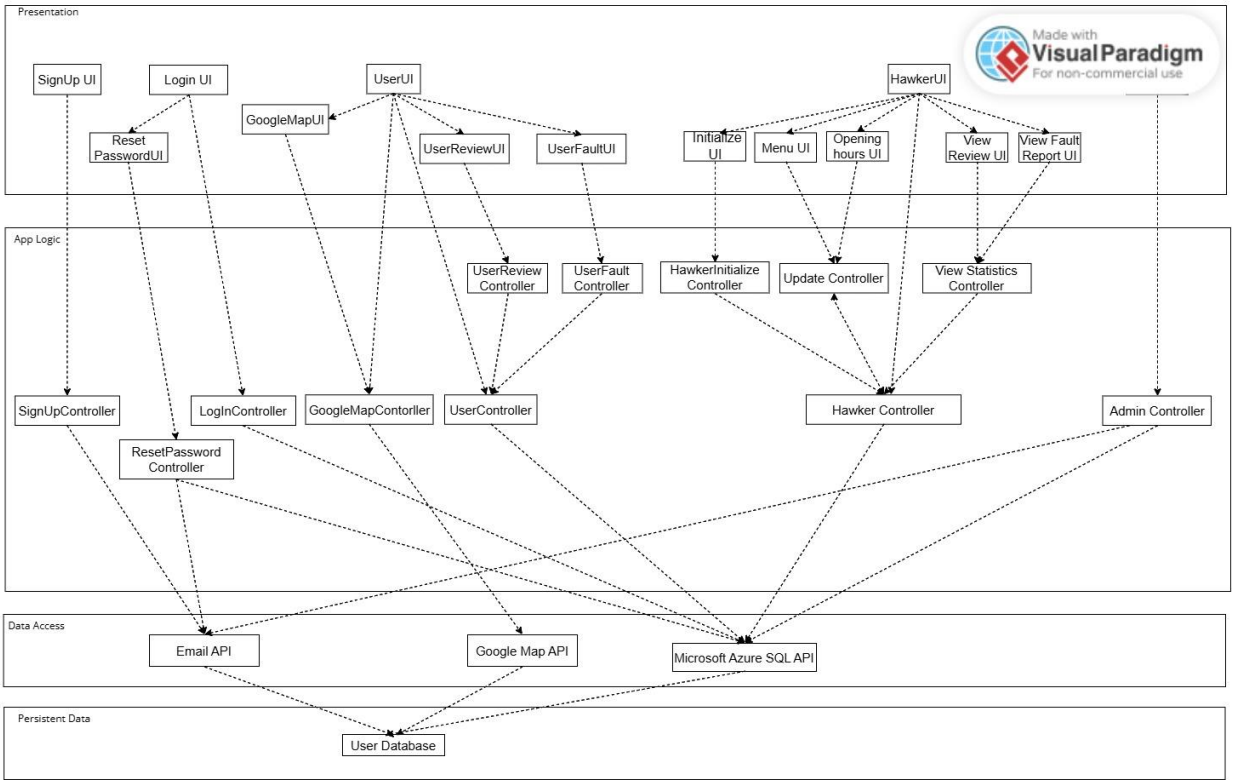
Use Case ID: 08B



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

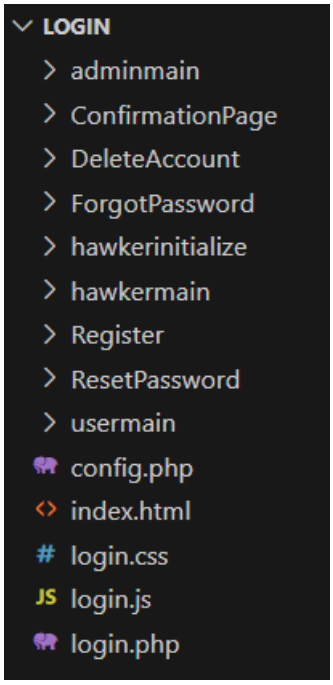
- Google Map API: Use in main page of customer to check location of the hawker center
- EmailJS API: Use for sending email to customers and hawkers after account registration and ‘reset password when forgot’ function

4. Data use

- We use the data “List of Government Markets Hawker Centres” in data.gov.sg 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



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	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): • Verifying account creation for customers and hawkers • Storing reviews and fault reports • Managing hawker stall data such as opening hours, menus, and crowd levels
View	The view manages how data is presented to users. HawkerStalk, this layer will handle (but not limited to): • Displays user-friendly interfaces, such as the review submission page, stall information pages, and fault reporting screens. • Show dynamic data like crowd levels, ratings, and reviews in an intuitive and visually appealing manner.
Controller	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): • Process form submissions (sign up, login, reviews, fault reports) • Validate data such as email format, password strength, and fault reports before passing it to the model for storage. • Respond to user actions (e.g., clicking on a hawker stall icon to view details) by updating the view with relevant data.

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.

Black Box Testing

1. AuthController

Control class to test - **AuthController**

An AuthController (short for Authentication Controller) is a crucial component of the application's controller layer responsible for handling authentication-related tasks. The AuthController typically manages customer login, registration, logout, password reset, and customer verification processes.

- **Customer Registration:** Handles customer sign-up by validating and saving new customer information to the database, ensuring security practices like password hashing are followed.
- **Customer Login:** Manages customer login by verifying the customer credentials (email and password) and issuing a session token or JSON Web Token (JWT) upon successful authentication.
- **Logout:** Logs the customer out by invalidating the customer session or token, typically by deleting or marking it as expired.
- **Password Reset:** Initiates password reset procedures, often involving sending a reset link or token to the user's registered email address and allowing them to create a new password.
- **Email Verification:** Confirms the user's email address by sending a verification link, which the customer must click to activate their account.

This unique identifier becomes a key element in distinguishing and managing customer data. Additionally, if the role of the customer selected is Hawker, the system will require additional inputs such as Operating License, Operating Hours, Food Type.

2. Equivalence Class and Boundary Value Testing

Equivalence Class Testing

Equivalence Class Testing (ECT), also known as Equivalence Partitioning, is a Black Box Testing technique used in software testing to reduce the number of test cases. The goal of ECT is to divide input data into partitions, or equivalence classes, such that any test case within an equivalence class is expected to yield the same results. This method assumes that if one test case in a partition

passes, then all other cases in that partition will also pass; similarly, if one fails, others in that partition are likely to fail.

- **Identify Input Data:** Determine the input domain for the software or system under test.
- **Partition the Input:** Divide the input data into equivalence classes. Each partition should represent a set of **valid** or **invalid** inputs.
- **Select Representatives:** Choose one representative from each equivalence class as the test case. This saves time and effort, as it is assumed that all values within a partition behave similarly.
- **Design Tests:** Create test cases using these representatives to cover all identified equivalence classes.

Valid equivalence classes describe valid situations, and the system should handle them normally.

Invalid equivalence classes describe invalid situations, and the system should reject them.

Boundary Value Testing

Boundary Value Testing (BVT), or Boundary Value Analysis, is a software testing technique focused on testing the boundaries of input ranges. The idea behind BVT is that errors often occur at the "edges" of input ranges, so testing the boundaries provides a higher chance of finding defects.

1. Login functions:

   Valid Equivalence Class: Emails and passwords with correct formats.

   Invalid equivalence Class: Emails and passwords with incorrect formats or missing information.

2. Sign- up functions:

   Valid Equivalence Class: Email, Password, and Confirm Password with correct formats

   Invalid Equivalence Class: Email, Password, and Confirm Password with incorrect formats or missing information

   If domain selected is a Hawker:

Valid Equivalence Class: Email, Password, Confirm Password, and Operating Licences with correct formats

Invalid Equivalence Class: Email, Password, Confirm Password, and Operating Licences with incorrect formats or missing information

3. Test Cases and Results

a. **Login**

Input parameters: Domain, Email and Password

No.	Test Input	Expected Output	Actual Output	Pass? (Y/N)
1	**(Valid) Domain:** "Customer" **(Valid)** Email: "halo@gmail.com" **(Valid)** Password: "111111aA"	"Log in successfully"	"Log in successfully"	Y
2	**(Invalid) Domain:** "" **(Valid)** Email: "halo@gmail.com" **(Valid)** Password: "111111aA"	"Invalid domain"	"Invalid domain"	Y
3	**(Valid) Domain:** "Customer" **(Invalid)** Email: "" **(Valid)** Password: "111111aA	"Please fill out this field"	"Please fill out this field"	Y
4	**(Valid) Domain:** "Customer" **(Valid)** Email: "halo@gmail.com" **(Invalid)** Password: ""	"Please fill out this field"	"Please fill out this field"	Y
5	**(Valid) Domain:** "Customer" **(Invalid)** Email: "halo" **(Valid)** Password: "111111aA	"Invalid email address"	"Invalid email address"	Y

6	**(Valid) Domain:** "Customer" **(Valid)** Email: "halo@gmail.com" **(Invalid)** Password: "1111aA"	"The password must be at least 8 characters long"	"The password must be at least 8 character long"	Y
7	**(Valid) Domain:** "Customer" **(Valid)** Email: "halo@gmail.com" **(Invalid)** Password: "111111aa"	"The password must contain at least one uppercase character, one lowercase character and one number"	"The password must contain at least one uppercase character, one lowercase character and one number"	Y
7	**(Valid) Domain:** "Customer" **(Valid)** Email: "halo@gmail.com" **(Invalid)** Password: "111111AA"	"The password must contain at least one uppercase character, one lowercase character and one number"	"The password must contain at least one uppercase character, one lowercase character and one number"	Y
8	**(Valid) Domain:** "Customer" **(Valid)** Email: "halo@gmail.com" **(Invalid)** Password: "aaaaAAAA"	"The password must contain at least one uppercase character, one lowercase character and one number"	"The password must contain at least one uppercase character, one lowercase character and one number"	Y

b. **Sign up**

Input parameters: Domain, Email, Password, and Confirm Password

No.	Test Input	Expected Output	Actual Output	Pass? (Y/N)

1	**(All Valid Inputs)** Domain: “Customer” Email: halo@gmail.com Password: “111111aA” Confirm Password: “111111aA”	“Sign up successfully”	“Sign up successfully”	Y
2	**(All Valid Inputs except Domain)** Domain: “”	"Please select domain”	"Please select domain”	Y
3	**(All Valid Inputs except Email)** Email: “”	“Please fill out this field”	“Please fill out this field”	Y
4	**(All Valid Inputs except Email)** Email: “halo”	“Invalid email address”	“Invalid email address”	Y
5	**(All Valid Inputs except Password)** Password: “”	“Please fill out this field”	“Please fill out this field”	Y
6	**(All Valid Inputs except Password)** Password: “1111aA”	“The password must be at least 8 characters long”	“The password must be at least 8 character long”	Y
7	**(All Valid Inputs except Password)** Password: “111111aa”	“The password must contain at least one uppercase character, one lowercase character and one number”	“The password must contain at least one uppercase character, one lowercase character and one number”	Y
8	**(All Valid Inputs except Password)** Password: “111111AA”	“The password must contain at least one uppercase character, one lowercase character and one number”	“The password must contain at least one uppercase character, one lowercase character and one number”	Y
9	**(All Valid Inputs except Password)**	“The password must contain at	“The password must contain at	Y

	Password: “aaaaAAAA”	least one uppercase character, one lowercase character and one number”	least one uppercase character, one lowercase character and one number”	
10	**(All Valid Inputs except Confirm Password)** Confirm Password: “”	“Please confirm password”	“Please confirm password”	Y
11	**(All Valid Inputs except Confirm Password)** Password: “111111aA” Confirm Password: “222222aA”	“Password mismatch!”	“Password mismatch!”	Y

c. **Sign up – If domain selected is Hawker**

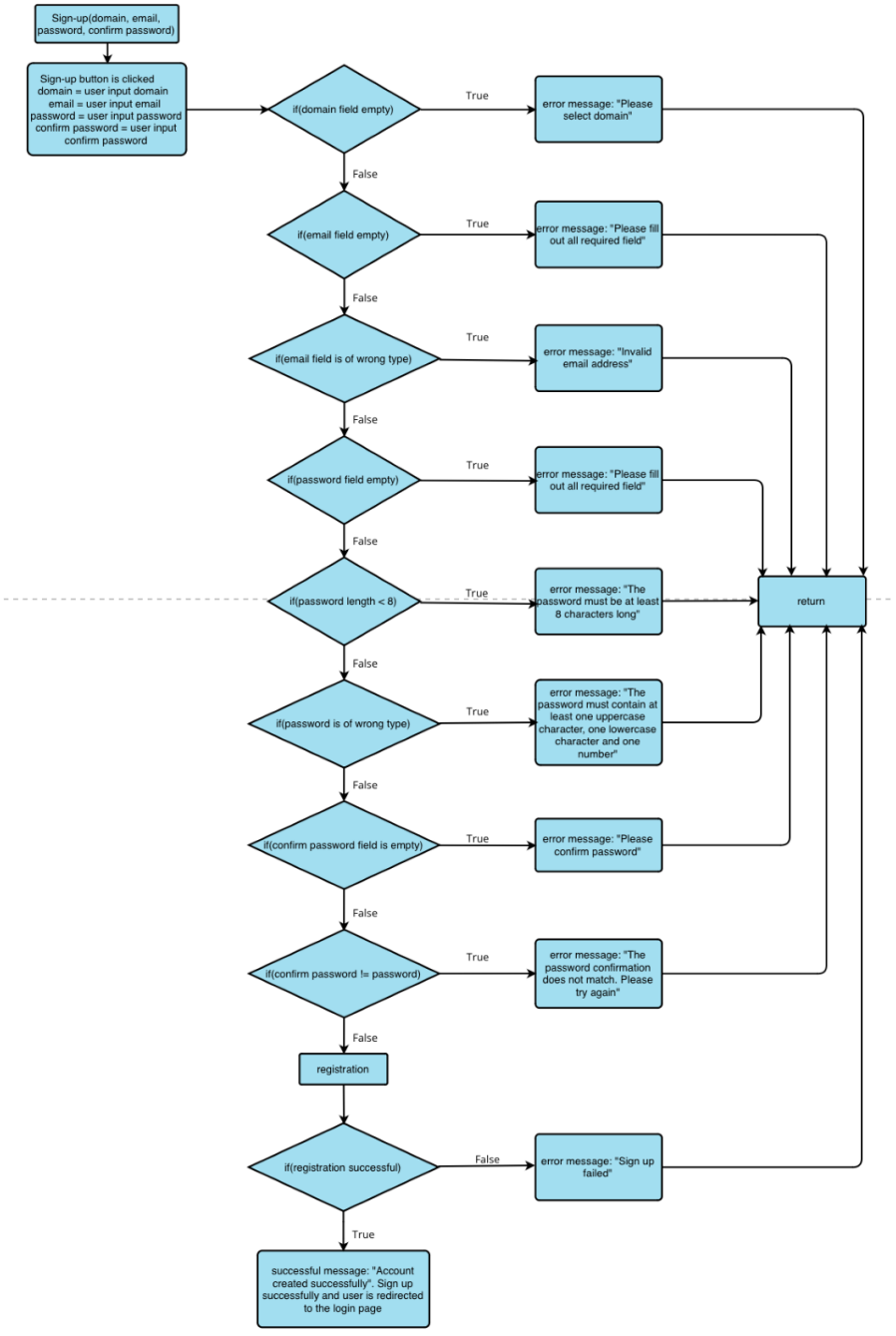
Input parameters: Domain, Email, Password, Confirm Password, and Licences

No.	Test Input	Expected Output	Actual Output	Pass? (Y/N)
1	**(All Valid Inputs)** Domain: “Hawker” Email: abc123@gmail.com Password: “Password123” Confirm Password: “Password123” Licence: “Submitted”	“Account created successfully”	“Account created successfully”	Y
2	**(All Valid Inputs except Licence)** Licence: “”	“Please upload a valid operating licence”	“Please upload a valid operating licence”	Y

White Box Testing

1. Create New Account (Customer)

a. Control Flow Path



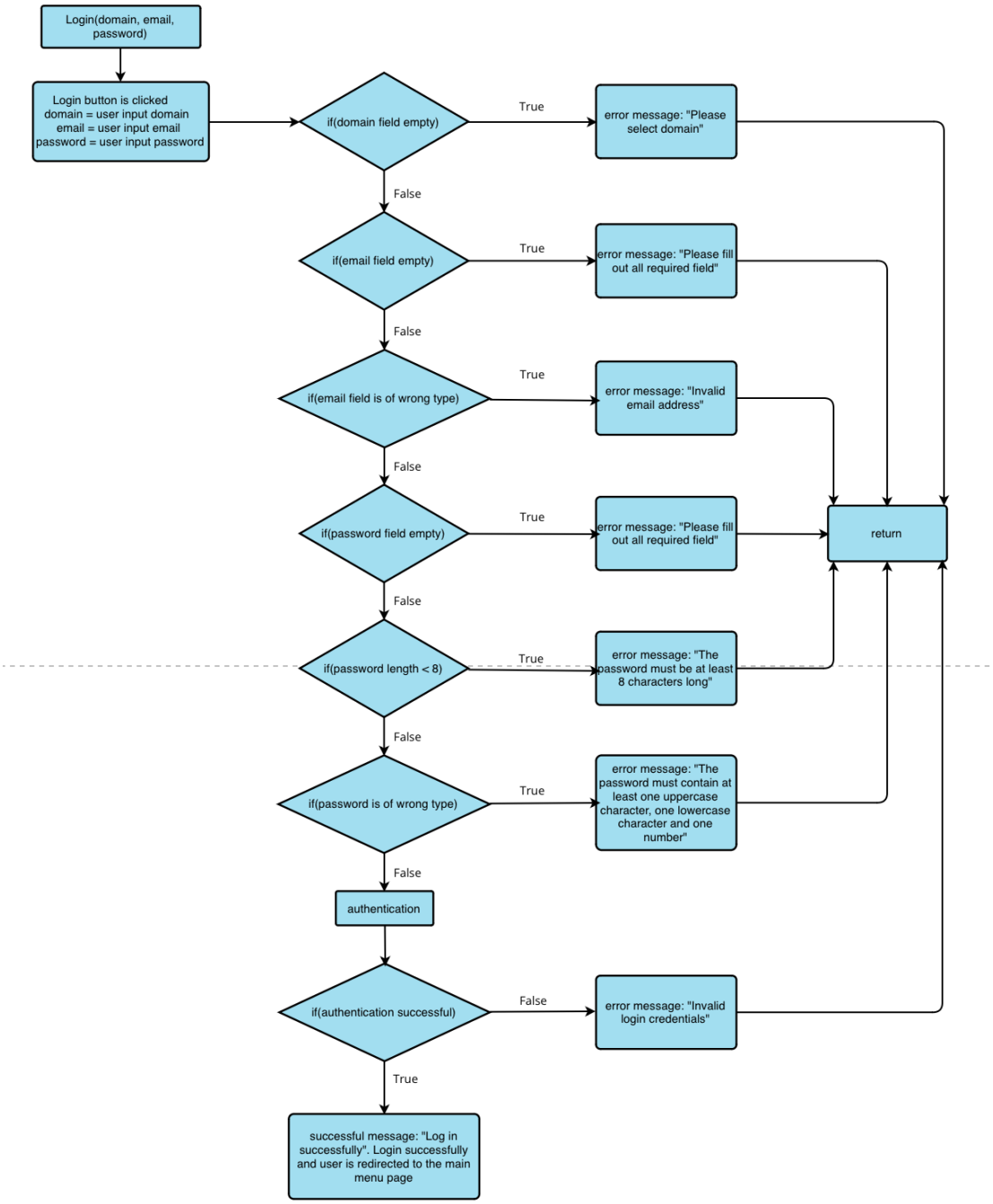
b. Test Cases and Results

SignUp(email, password, confirmpwd)

No.	Test Input	Expected Output	Actual Output	Pass?
1	Email = "halo@gmail.com" Password = "111111aA" Confirmpwd = "111111aA"	"Sign up successfully"	"Sign up successfully"	Y
2	Email = "halo" Password = "111111aA" Confirmpwd = "111111aA"	"Please include an '@' in the email address. 'halo' is missing an '@'"	"Please include an '@' in the email address. 'halo' is missing an '@'"	Y
3	Email = halo@gmail.com Password = "111111aA" Confirmpwd = "222222aA"	"Password mismatch!"	"Password mismatch!"	Y
4	Email = "" Password = "111111aA" Confirmpwd = "111111aA"	"Please fill out this field"	"Please fill out this field"	Y
5	Email = "halo@gmail.com" Password = "" Confirmpwd = "111111aA"	"Please fill out this field"	"Please fill out this field"	Y
6	Email = "halo@gmail.com" Password = "111111aA" Confirmpwd = ""	"Please fill out this field"	"Please fill out this field"	Y
7	Email = "halo@gmail.com" Password = "1111aA" Confirmpwd = "1111aA"	"Password must be at least 8 characters long"	"Password must be at least 8 characters long"	Y
8	Email = "halo@gmail.com" Password = "111111aa" Confirmpwd = "111111aa"	"Password must contain both uppercase and lowercase letters"	"Password must contain both uppercase and lowercase letters"	Y
9	Email = *exsisting email* Password = "111111aA" Comfirmpwd = "111111aA"	"Email exists, do you want to login?"	"Email exists, do you want to login?"	Y

2. Login

a. Control Flow Path



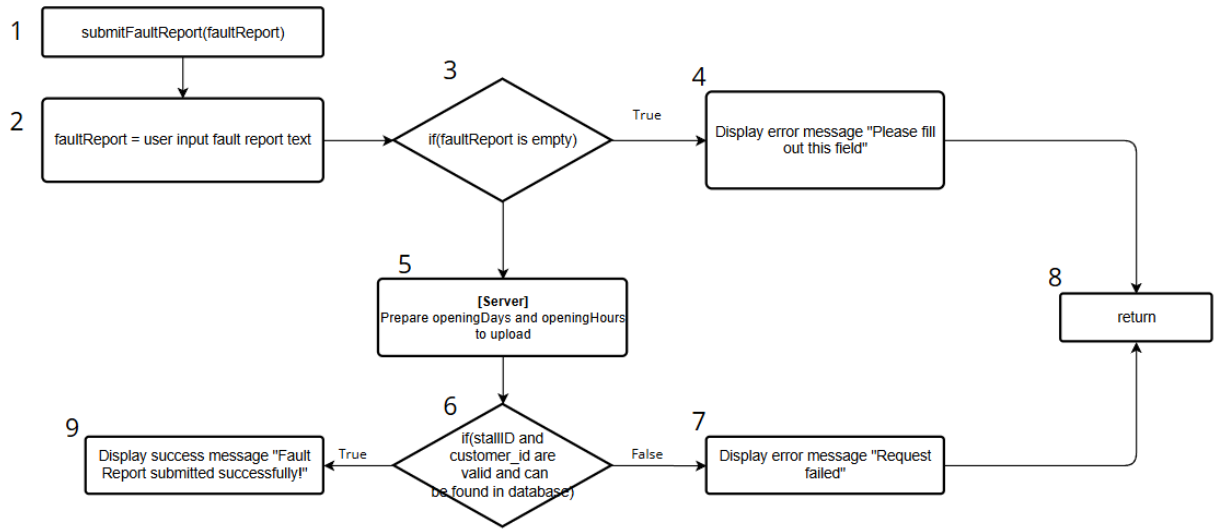
b. Test Cases and Results

login(domain, email, password)

No.	Test Input	Expected Output	Actual Output	Pass?
1	domain = “Customer” email = "halo@gmail.com” password = “111111aA”	“Login successfully”	“Login successfully”	Y
2	domain = “Hawker” email = “hawker123@test.com” password = “Hawker123#”	“Login successfully”	“Login successfully”	Y
3	domain = “” email = “halo@gmail.com” password = “111111aA”	“Invalid domain”	“Invalid domain”	Y
4	domain = “Customer” email = “” password = “111111aA”	“Please fill out this field”	“Please fill out this field”	Y
5	domain = “Customer” email = “halo@gmail.com” password = “”	“Please fill out this field”	“Please fill out this field”	Y
6	domain = “Customer” email = “halo” password = “111111aA”	“Please include an ‘@’ in the email address. ‘halo’ is missing an ‘@’”	“Please include an ‘@’ in the email address. ‘halo’ is missing an ‘@’”	Y
7	domain = “Customer” email = “hawker123@test.com” password = “Hawker123#”	“Invalid email or password”	“Invalid email or password”	Y
8	domain = “Hawker” email = “hawker123@test.com” password = “Hawker456#”	“Invalid email or password”	“Invalid email or password”	Y

3. Customer Submit Fault Report

a. Control Flow Path



b. Test Cases and Results

submitFaultReport(faultReport)

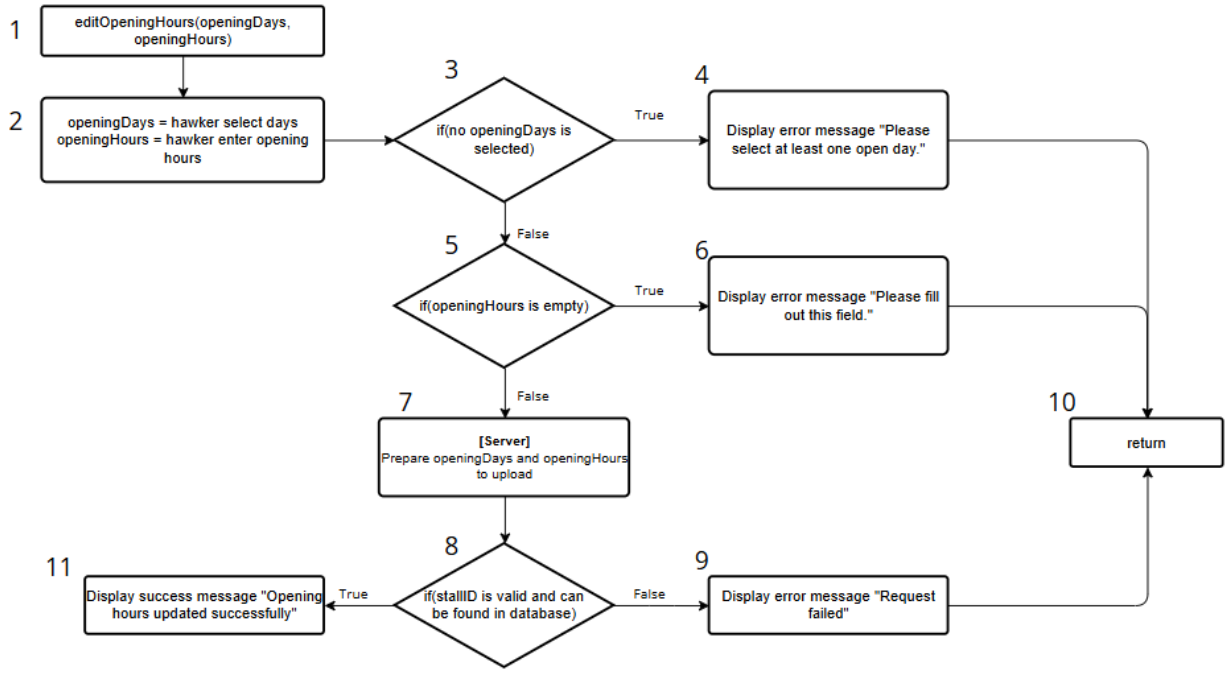
- Input: faultReport
- Implied input: stallID (passed automatically by inferring from the current stall selected by customer), user_id (passed automatically by inferring from the current current logged in customer)

No.	Test Input	Expected Output	Actual Output	Pass?
1	faultReport = “Wrong opening hours” stallID = 4 user_id = 1	“Fault Report submitted successfully”	“Fault Report submitted successfully!”	Y
2	faultReport = “” stallID = 4 user_id = 1	“Please fill out this field”	“Please fill out this field”	Y
3	faultReport = “Wrong opening hours” stallID = -1 (invalid and not found in database) user_id = 1	“Request Failed”	“Request failed”	Y
4	faultReport = “Wrong opening hours”	“Request Failed”	“Request Failed”	Y

	stallID = 4 user_id = -1 (invalid and not found in database)			

Hawker Edit Opening Hours

a. Control Flow Path



b. Test Cases and Results

editOpeningHours(openingDays, openingHours)

- Inputs: openingDays, openingHours
- Implied input: StallID (passed automatically by inferring from the current current logged in hawker)

No.	Test Input	Expected Output	Actual Output	Pass?
1	openingDays != “0000000” openingHours = “9AM - 9PM” StallID = 4	“Opening hours updated successfully.”	“Opening hours updated successfully.”	Y

2	openingDays = “0000000” openingHours = “9AM - 9PM” StallID = 4	“Please select at least one open day.”	“Please select at least one open day."	Y
3	openingDays != “0000000” openingHours = “” StallID = 4	“Please fill out this field”	“Please fill out this field”	Y
4	openingDays != “0000000” openingHours = “9AM - 9PM” StallID = -1(invalid and not found in database)	“Request Failed”	“Request Failed”	Y