HawkerStalk

Singapore's Hawker Gems at your Fingertips!

Done by:
Tan Ming Hao (U2320580D)
Choo Zhen Ming (U2320950J)
Cho Zhi Wei(U2320530C)
Chow Weng Shi (U2320760B)
Lai Xin Yee (U2320650G)
Swaminathan Navitraa (U2321255K)
Pham Nguyen Vu Hoang (U2323430D)

Outline

1. Introduction
Problem statement, solution and live demo!

2. Good SWE Practices
Proper documentation, good code practices, reusability & refactoring!

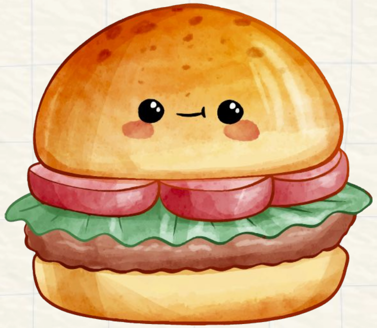
3. System Design
Illustrated through architecture, class and dialog diagrams!

4. Design Patterns
Explanation of our choice and implementation of design pattern!

5. Traceability
Walk-through of specific user test cases and different testing methods!

6. Future Plans
Additional features we would like to implement!

1. Introduction

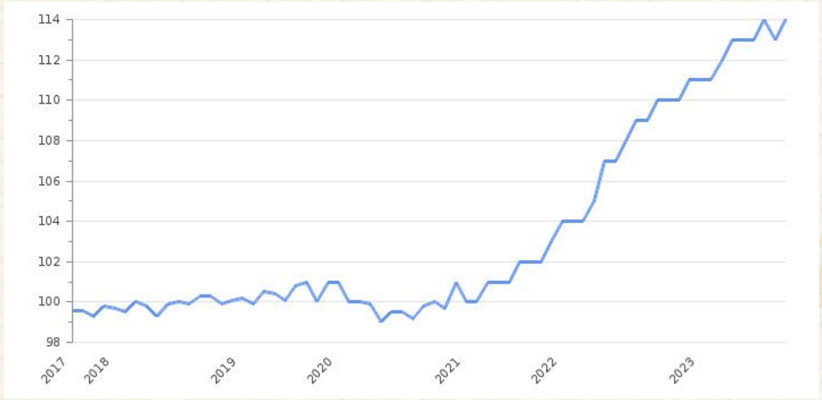


Problem

Hawkers are unable to keep up with rising costs!

- Rising operation costs - rent, table cleaning fees, dishwashing, plates returned to hawker cost, and laundry list of fines, can come up to over $4,000 a month.
- **Rising inflation - 5.5%**
- **GST Hike - 8%**
- Difficult to increase price as hawker food's key value proposition is "cheap"

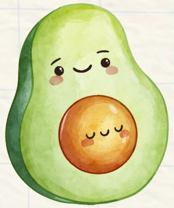
Inflation Indicated by Consumer Price Index (CPI) in Singapore



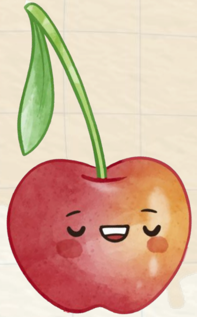
Singapore Inflation Rate (2024), Take-Profit.org

Singapore must preserve its Hawker Culture!

- Singapore prides itself on its world-renowned hawker culture
- Singapore featured on the **UNESCO Representative List of the Intangible Cultural Heritage of Humanity in 2020.**

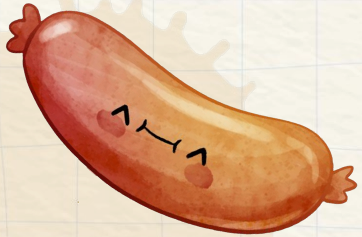


Therefore, we need to **increase demand** for Hawker food by **increasing accessibility** to keep Hawkers alive.

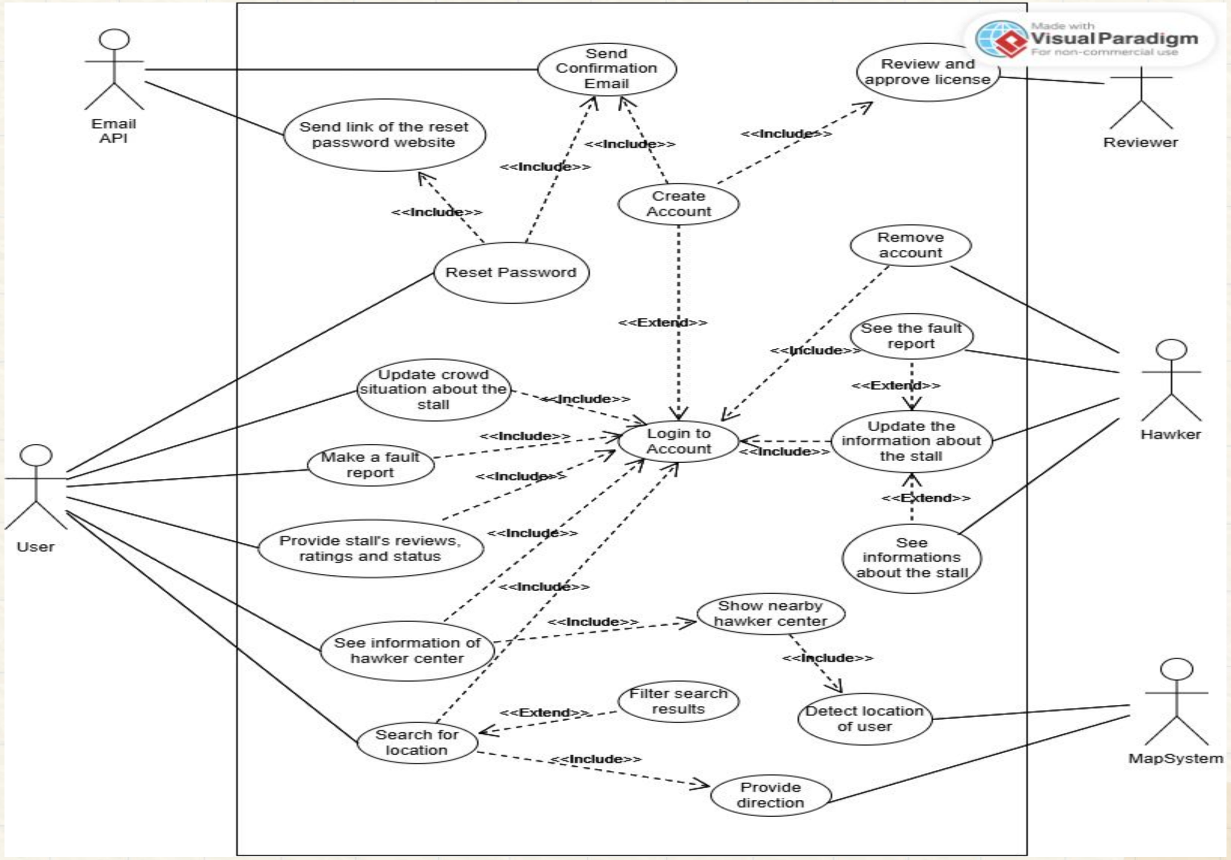


Introducing HawkerStalk!

A one-stop platform that connects tourists, residents and hawkers to preserve Singapore’s unique Hawker culture!



Use Case Diagram



Feature List

1. Authentication

a. Signup (depending on user, hawker or admin domain)
b. Verification email sent to inbox during sign-up
c. Login

2. Admin

d. Approve or reject new hawker account
e. Suspend any account
f. View user and hawker account details

Feature List

3. Hawker

a. Initialise profile during sign-up
b. Update menu
c. Update opening hours and days
d. View reviews
e. View fault reports
f. Delete account and close store permanently

Feature List

4. User

a. View hawker centre on map
b. Search for hawker centre
c. See stalls at hawker centre
d. View menu of hawker stall
e. View opening hours and days of hawker stall
f. View ratings & reviews of hawker stall
g. Submit stall review
h. Submit fault report for stall

Non-functional Requirements

1. **User-friendly interface**
   a. Use of simple, intuitive design with quick access to search and review functionalities
2. **Security and data privacy**
   a. Sensitive data, such as user information and hawker license details are encrypted for security
3. **Scalability**
   a. The app is designed to handle increasing numbers of users and data without compromising performance

External Data & APIs Used

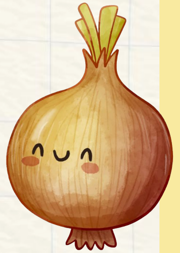
APIs

- **Google Map API** –> used in user's main page to check location of hawker centre
- **EmailJS API** –> used to send email to users and hawkers upon registration

Dataset

- **Source**: "List of Government Markets Hawker Centres" in data.gov.sg
- **Use case**: To access the address of hawker centres and mark them on our map

Tech Stack



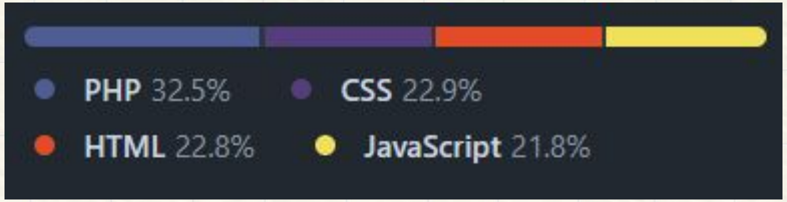
Frontend

- Html
- CSS

Backend

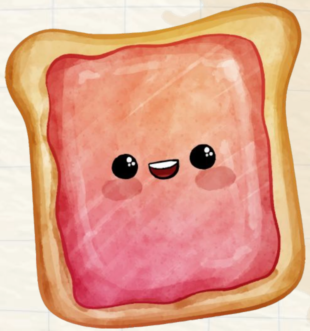
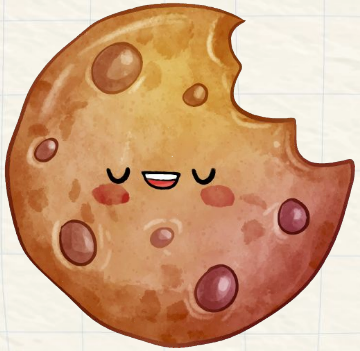
- PHP
- Javascript
- Microsoft Azure SQL database

Github Repository Language Breakdown



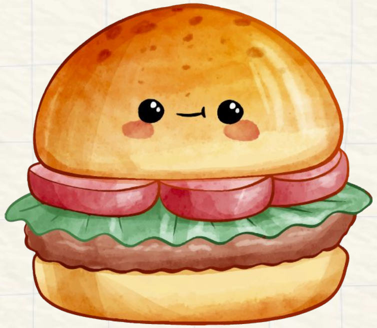
● PHP 32.5% ● CSS 22.9%
● HTML 22.8% ● JavaScript 21.8%

Live Demo!

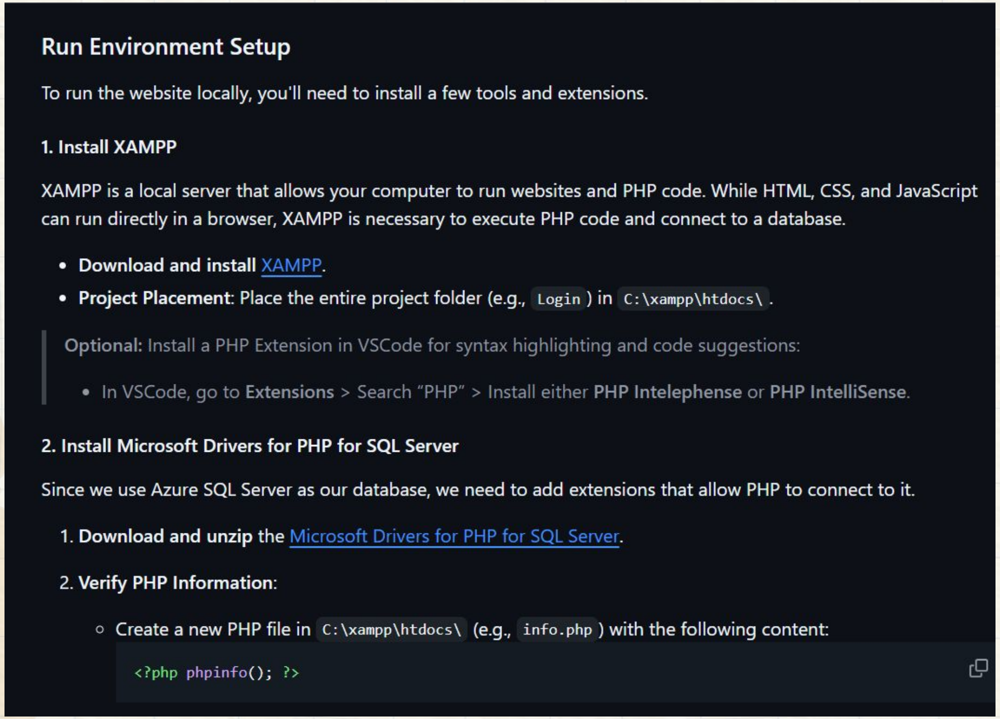



2.

Good SWE Practices



Implementation of README



Run Environment Setup

To run the website locally, you'll need to install a few tools and extensions.

1. Install XAMPP

XAMPP is a local server that allows your computer to run websites and PHP code. While HTML, CSS, and JavaScript can run directly in a browser, XAMPP is necessary to execute PHP code and connect to a database.

- **Download and install** XAMPP.
- **Project Placement**: Place the entire project folder (e.g., `Login`) in `C:\xampp\htdocs\`.

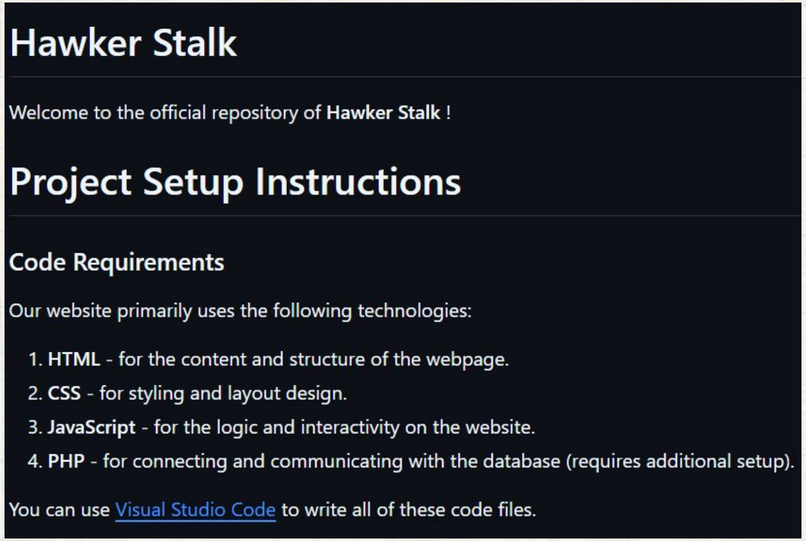
> **Optional:** Install a PHP Extension in VSCode for syntax highlighting and code suggestions:
>
> - In VSCode, go to **Extensions** > Search "PHP" > Install either **PHP Intelephense** or **PHP IntelliSense**.

2. Install Microsoft Drivers for PHP for SQL Server

Since we use Azure SQL Server as our database, we need to add extensions that allow PHP to connect to it.

1. **Download and unzip** the Microsoft Drivers for PHP for SQL Server.
2. **Verify PHP Information**:
   - Create a new PHP file in `C:\xampp\htdocs\` (e.g., `info.php`) with the following content:

```
<?php phpinfo(); ?>
```



Hawker Stalk

Welcome to the official repository of **Hawker Stalk** !

Project Setup Instructions

Code Requirements

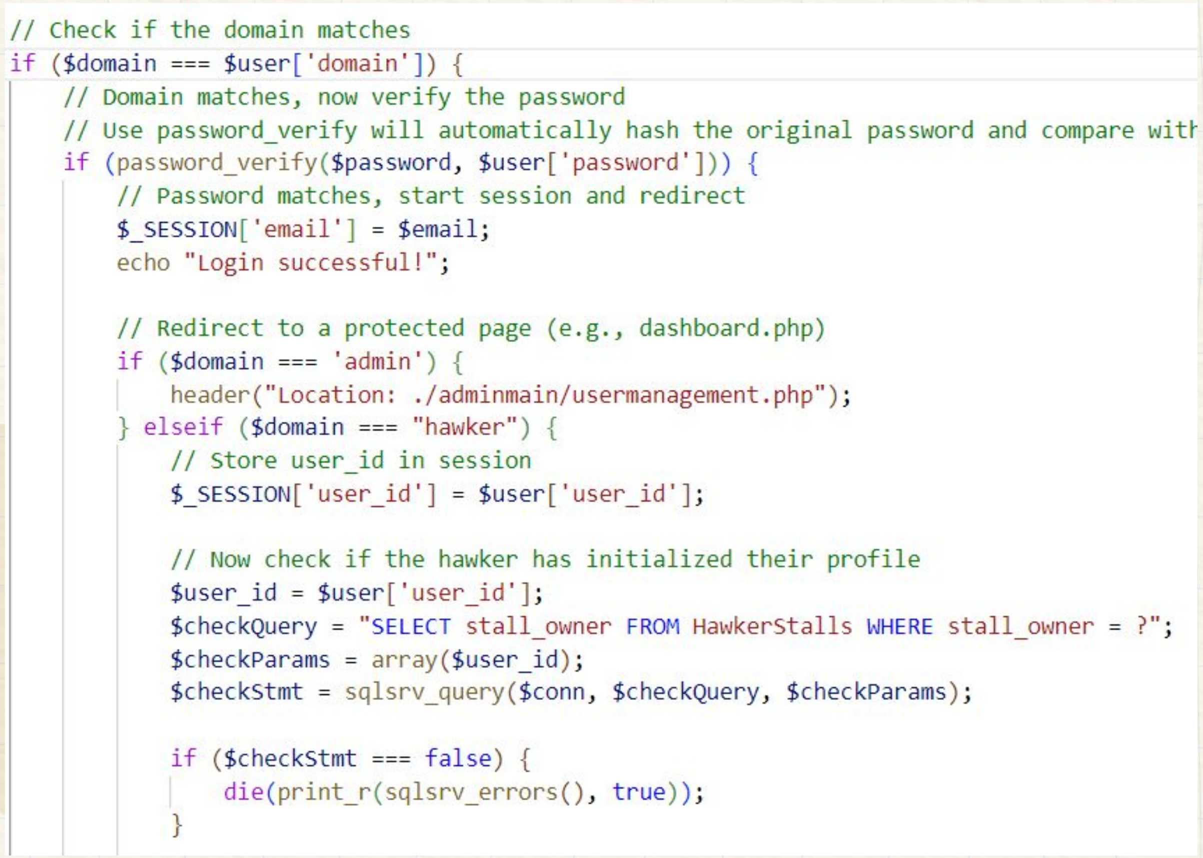
Our website primarily uses the following technologies:

1. **HTML** - for the content and structure of the webpage.
2. **CSS** - for styling and layout design.
3. **JavaScript** - for the logic and interactivity on the website.
4. **PHP** - for connecting and communicating with the database (requires additional setup).

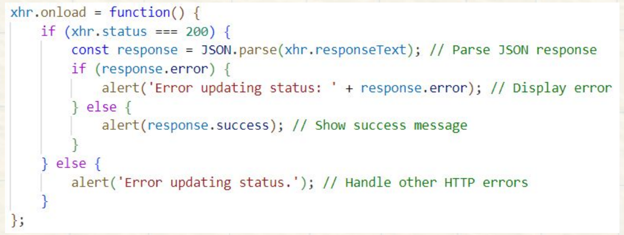
You can use Visual Studio Code to write all of these code files.

Giving instruction to all developer to maintain consistency

Code with Comments throughout whole process



```
// Check if the domain matches
if ($domain === $user['domain']) {
    // Domain matches, now verify the password
    // Use password_verify will automatically hash the original password and compare wit
    if (password_verify($password, $user['password'])) {
        // Password matches, start session and redirect
        $_SESSION['email'] = $email;
        echo "Login successful!";

        // Redirect to a protected page (e.g., dashboard.php)
        if ($domain === 'admin') {
            header("Location: ./adminmain/usermanagement.php");
        } elseif ($domain === "hawker") {
            // Store user_id in session
            $_SESSION['user_id'] = $user['user_id'];

            // Now check if the hawker has initialized their profile
            $user_id = $user['user_id'];
            $checkQuery = "SELECT stall_owner FROM HawkerStalls WHERE stall_owner = ?";
            $checkParams = array($user_id);
            $checkStmt = sqlsrv_query($conn, $checkQuery, $checkParams);

            if ($checkStmt === false) {
                die(print_r(sqlsrv_errors(), true));
            }
```

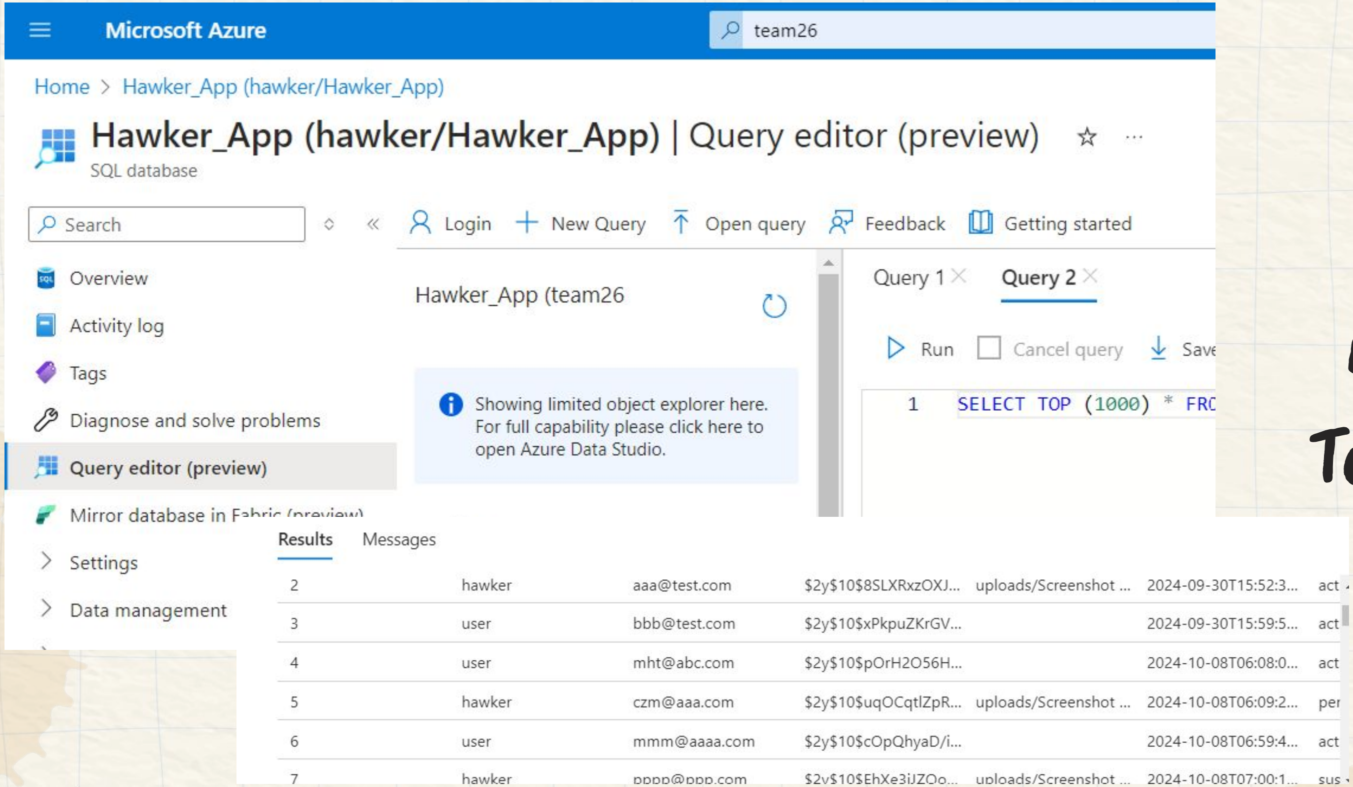


```
xhr.onload = function() {
    if (xhr.status === 200) {
        const response = JSON.parse(xhr.responseText); // Parse JSON response
        if (response.error) {
            alert('Error updating status: ' + response.error); // Display error
        } else {
            alert(response.success); // Show success message
        }
    } else {
        alert('Error updating status.'); // Handle other HTTP errors
    }
};
```

Human Readable

Cooperation

Online SQL Database - Microsoft Azure



Microsoft Azure

team26

Home > Hawker_App (hawker/Hawker_App)

Hawker_App (hawker/Hawker_App) | Query editor (preview)

SQL database

Hawker_App (team26

Showing limited object explorer here. For full capability please click here to open Azure Data Studio.

```
SELECT TOP (1000) * FRO
```

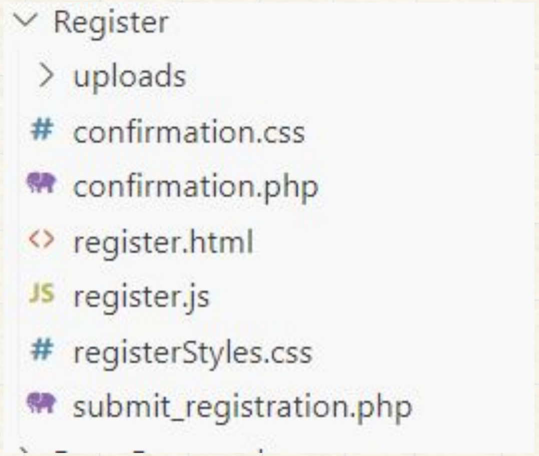
Results   Messages

2	hawker	aaa@test.com	$2y$10$8SLXRxzOXJ...	uploads/Screenshot ...	2024-09-30T15:52:3...	act
3	user	bbb@test.com	$2y$10$xPkpuZKrGV...		2024-09-30T15:59:5...	act
4	user	mht@abc.com	$2y$10$pOrH2O56H...		2024-10-08T06:08:0...	act
5	hawker	czm@aaa.com	$2y$10$uqOCqtlZpR...	uploads/Screenshot ...	2024-10-08T06:09:2...	per
6	user	mmm@aaaa.com	$2y$10$cOpQhyaD/i...		2024-10-08T06:59:4...	act
7	hawker	pppp@ppp.com	$2y$10$EhXe3iJZOo...	uploads/Screenshot ...	2024-10-08T07:00:1...	sus

Allow Backend Team Corporate Smoothly

Consistent File Naming Convention & Combination

*Using the similar name



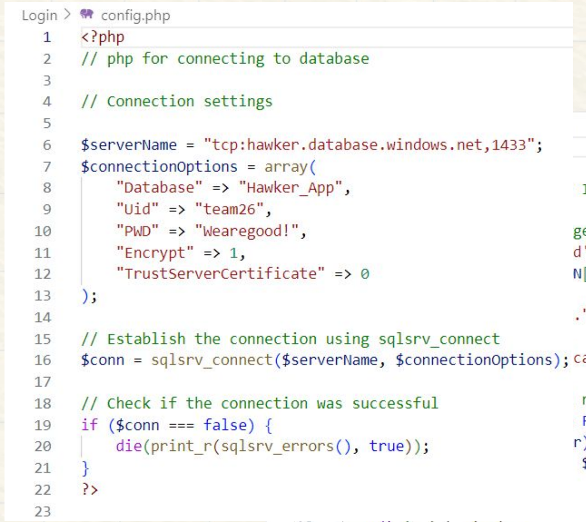
```
⌄ Register
  › uploads
  # confirmation.css
    confirmation.php
  <> register.html
  JS register.js
  # registerStyles.css
    submit_registration.php
```

Separate File into four parts

- Html (Webpage)
- Css (Design)
- Js (Frontend Design and Backend)
- Php (Dealing with Database)

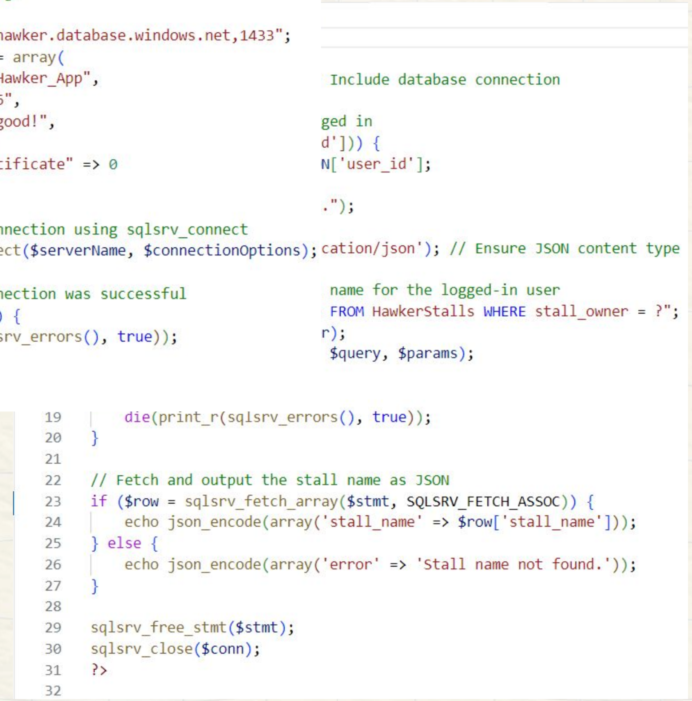
Readable, Maintainable, Extendable

Reusability & Refactoring



Login > config.php

```php
<?php
// php for connecting to database

// Connection settings

$serverName = "tcp:hawker.database.windows.net,1433";
$connectionOptions = array(
    "Database" => "Hawker_App",
    "Uid" => "team26",
    "PWD" => "Wearegood!",
    "Encrypt" => 1,
    "TrustServerCertificate" => 0
);

// Establish the connection using sqlsrv_connect
$conn = sqlsrv_connect($serverName, $connectionOptions);

// Check if the connection was successful
if ($conn === false) {
    die(print_r(sqlsrv_errors(), true));
}
?>
```



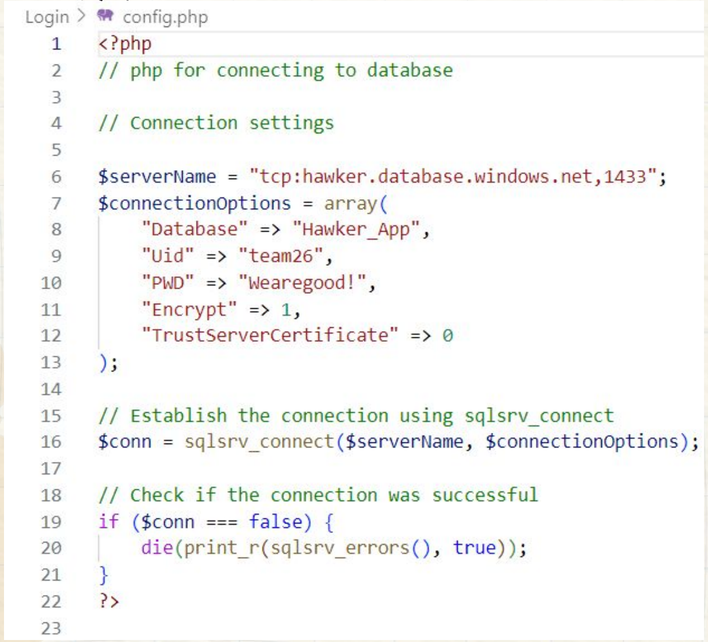
```php
 Include database connection

ged in
d'])) {
N['user_id'];

.");

cation/json'); // Ensure JSON content type

 name for the logged-in user
 FROM HawkerStalls WHERE stall_owner = ?";
r);
 $query, $params);

    die(print_r(sqlsrv_errors(), true));
}

// Fetch and output the stall name as JSON
if ($row = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_ASSOC)) {
    echo json_encode(array('stall_name' => $row['stall_name']));
} else {
    echo json_encode(array('error' => 'Stall name not found.'));
}

sqlsrv_free_stmt($stmt);
sqlsrv_close($conn);
?>
```

Avoid Redundancy

Improve Maintainability

Reusability & Refactoring

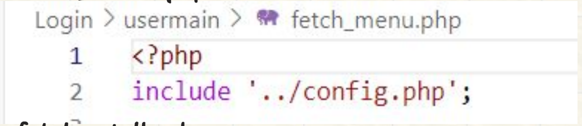
config.php



Login > config.php

```php
<?php
// php for connecting to database

// Connection settings

$serverName = "tcp:hawker.database.windows.net,1433";
$connectionOptions = array(
    "Database" => "Hawker_App",
    "Uid" => "team26",
    "PWD" => "Wearegood!",
    "Encrypt" => 1,
    "TrustServerCertificate" => 0
);

// Establish the connection using sqlsrv_connect
$conn = sqlsrv_connect($serverName, $connectionOptions);

// Check if the connection was successful
if ($conn === false) {
    die(print_r(sqlsrv_errors(), true));
}
?>
```

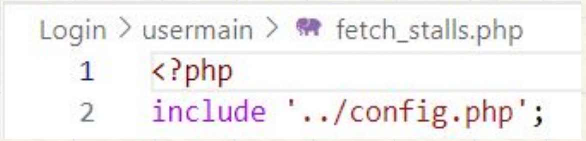
fetch_menu.php



Login > usermain > fetch_menu.php

```php
<?php
include '../config.php';
```

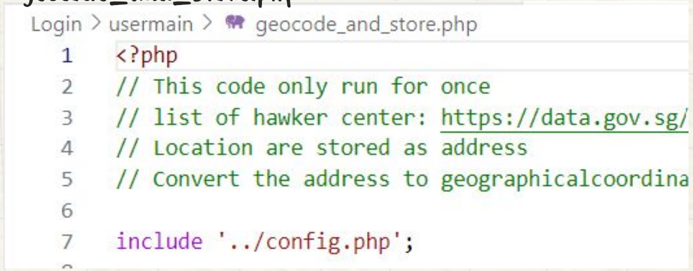
fetch_stalls.php



Login > usermain > fetch_stalls.php

```php
<?php
include '../config.php';
```

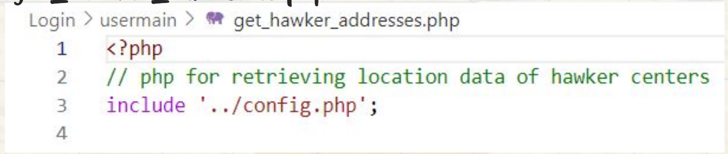
geocode_and_store.php



Login > usermain > geocode_and_store.php

```php
<?php
// This code only run for once
// list of hawker center: https://data.gov.sg/
// Location are stored as address
// Convert the address to geographicalcoordina

include '../config.php';
```

get_hawker_addresses.php

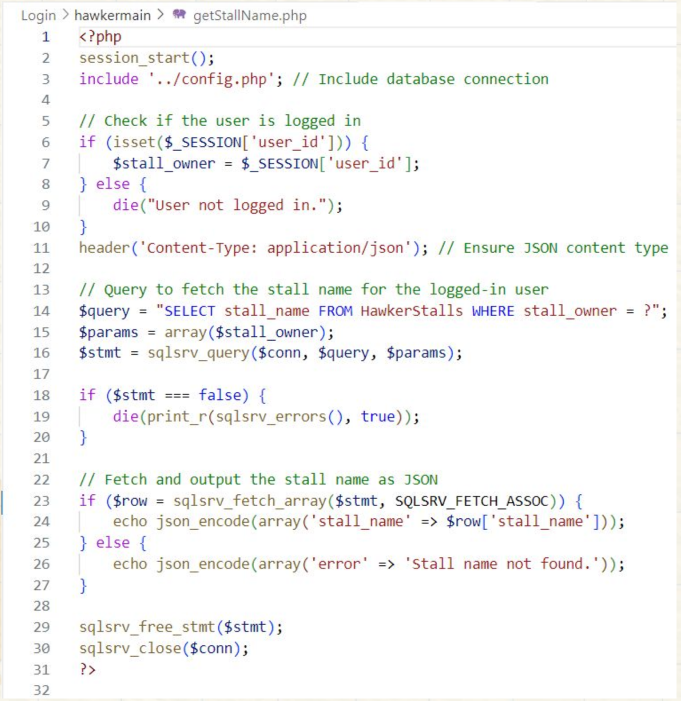


Login > usermain > get_hawker_addresses.php

```php
<?php
// php for retrieving location data of hawker centers
include '../config.php';
```

Reusability & Refactoring

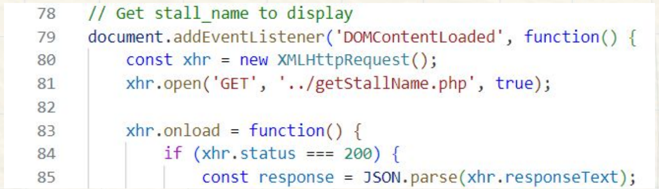
getStallName.php



Login > hawkermain > getStallName.php

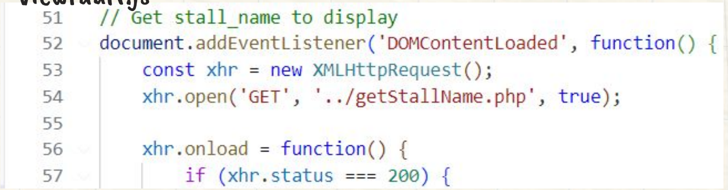
```php
<?php
session_start();
include '../config.php'; // Include database connection

// Check if the user is logged in
if (isset($_SESSION['user_id'])) {
    $stall_owner = $_SESSION['user_id'];
} else {
    die("User not logged in.");
}
header('Content-Type: application/json'); // Ensure JSON content type

// Query to fetch the stall name for the logged-in user
$query = "SELECT stall_name FROM HawkerStalls WHERE stall_owner = ?";
$params = array($stall_owner);
$stmt = sqlsrv_query($conn, $query, $params);

if ($stmt === false) {
    die(print_r(sqlsrv_errors(), true));
}

// Fetch and output the stall name as JSON
if ($row = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_ASSOC)) {
    echo json_encode(array('stall_name' => $row['stall_name']));
} else {
    echo json_encode(array('error' => 'Stall name not found.'));
}

sqlsrv_free_stmt($stmt);
sqlsrv_close($conn);
?>

```

hawkeropeninghours.js



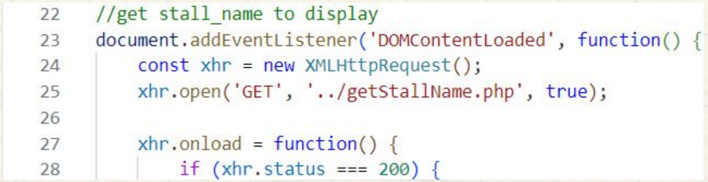
```javascript
// Get stall_name to display
document.addEventListener('DOMContentLoaded', function() {
    const xhr = new XMLHttpRequest();
    xhr.open('GET', '../getStallName.php', true);

    xhr.onload = function() {
        if (xhr.status === 200) {
            const response = JSON.parse(xhr.responseText);
```

viewfault.js



```javascript
// Get stall_name to display
document.addEventListener('DOMContentLoaded', function() {
    const xhr = new XMLHttpRequest();
    xhr.open('GET', '../getStallName.php', true);

    xhr.onload = function() {
        if (xhr.status === 200) {
```

hawkerreview.js

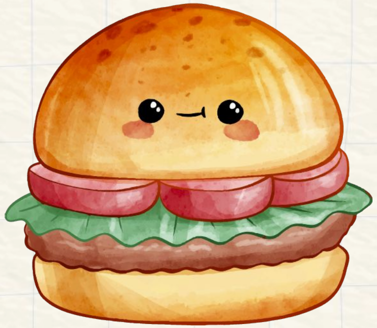


```javascript
//get stall_name to display
document.addEventListener('DOMContentLoaded', function() {
    const xhr = new XMLHttpRequest();
    xhr.open('GET', '../getStallName.php', true);

    xhr.onload = function() {
        if (xhr.status === 200) {
```

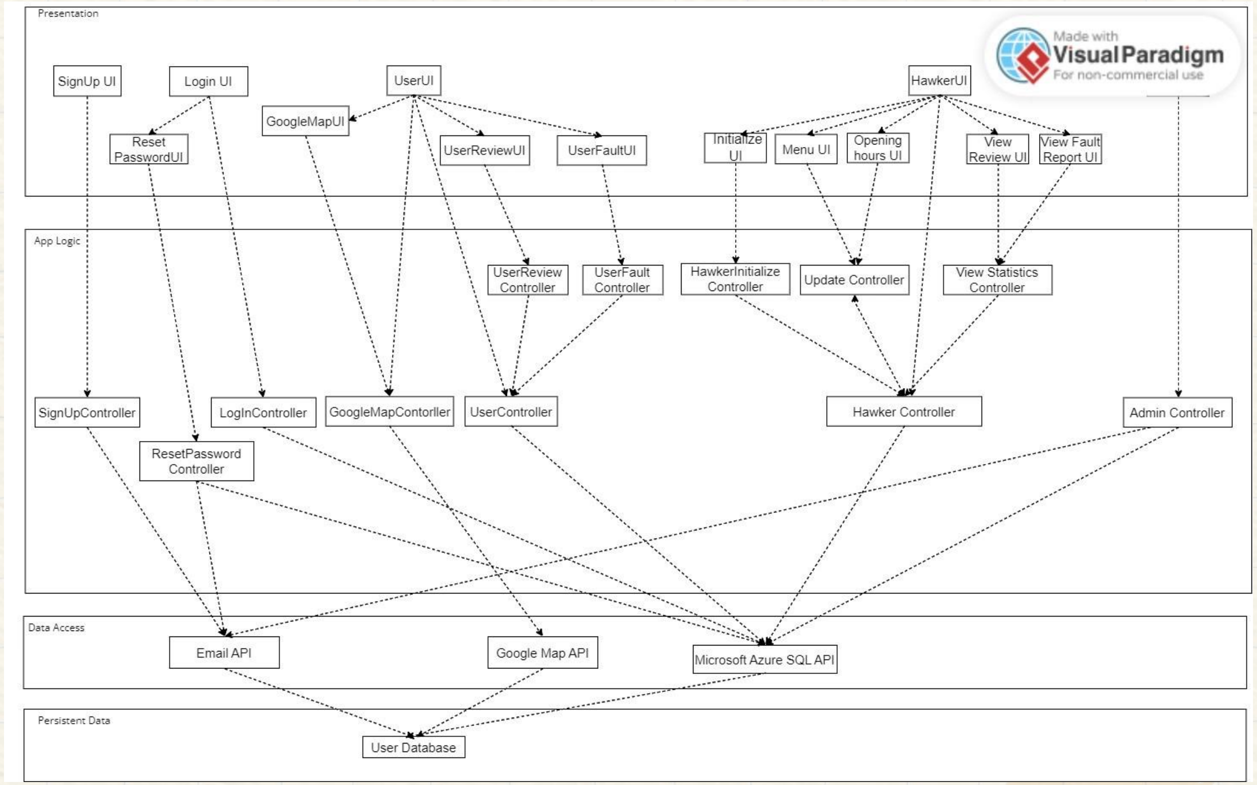
hawkerupdatestatus.js

3.

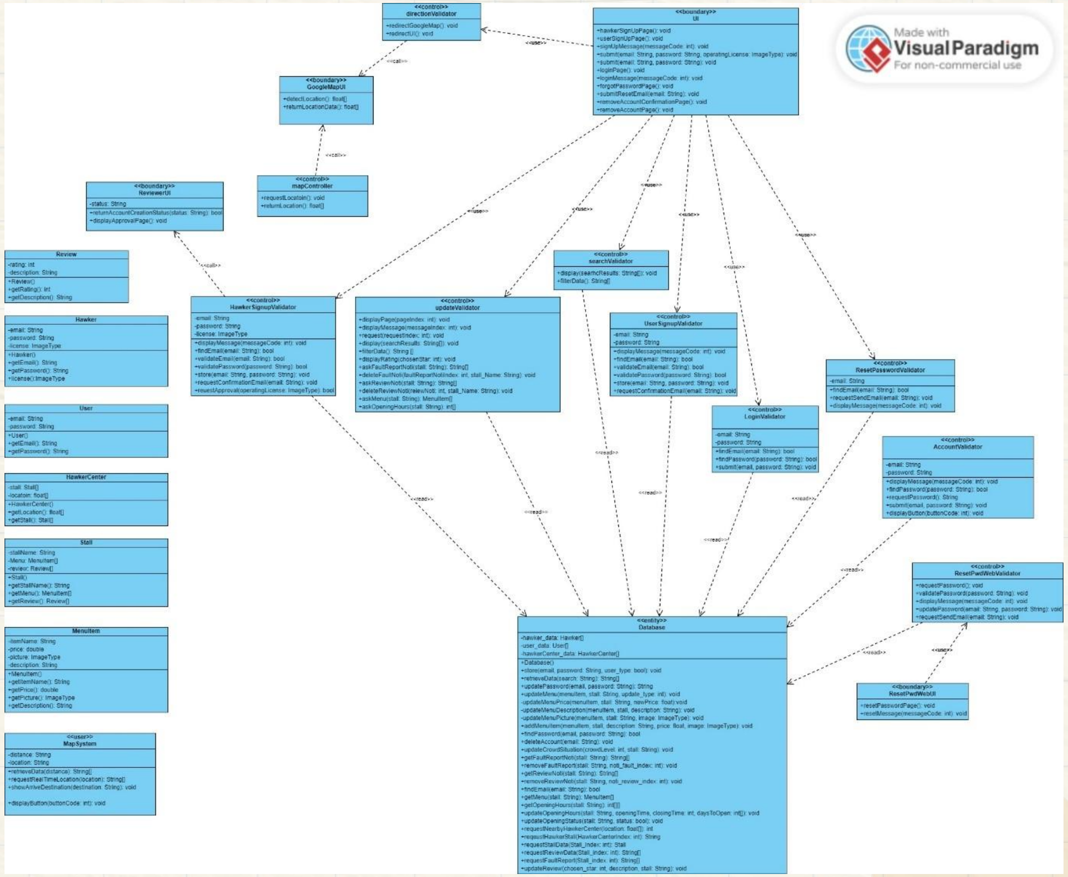
System Design



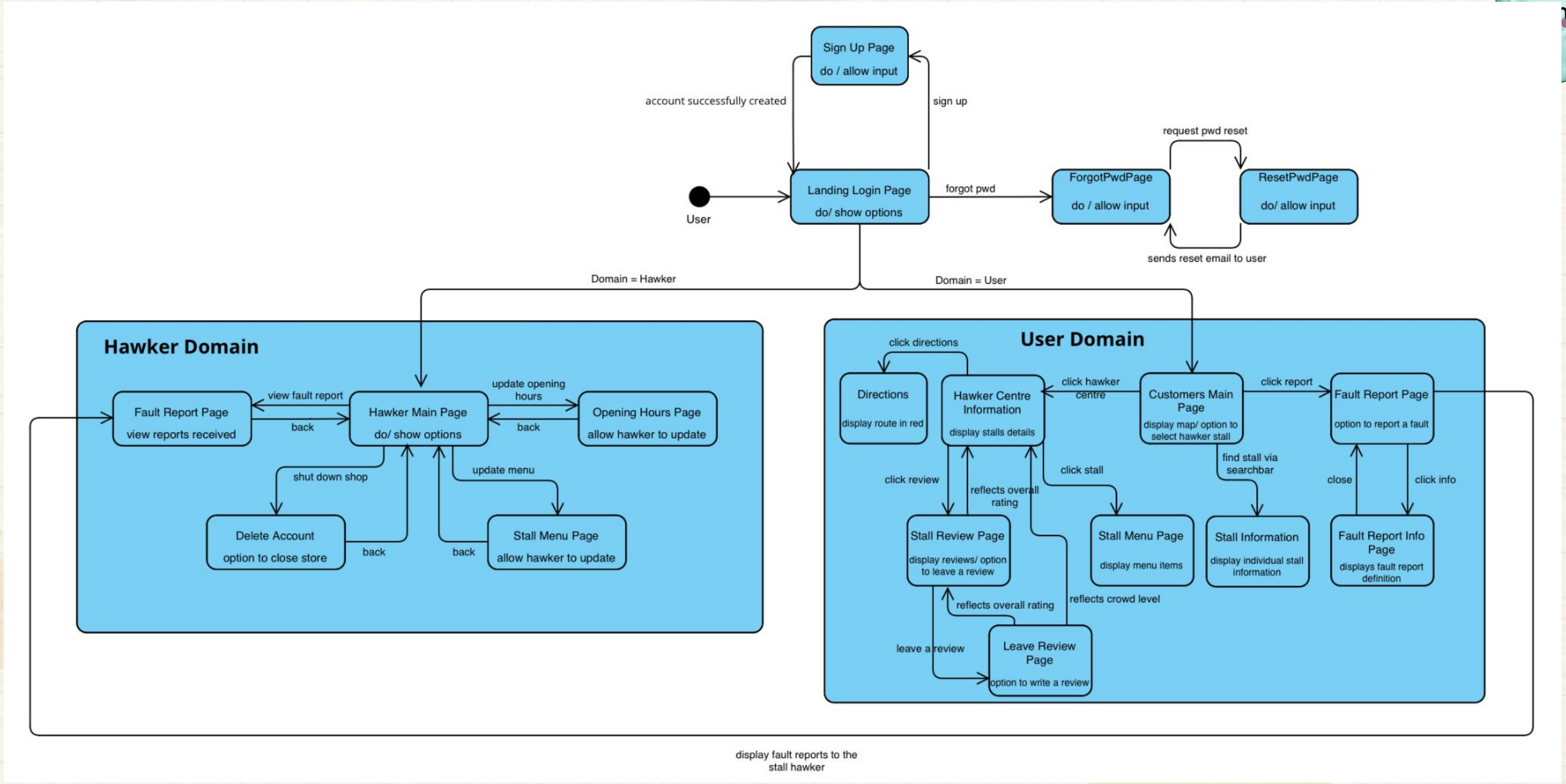
Architecture Diagram



Class Diagram


Made with Visual Paradigm For non-commercial use
<<control>> directionValidator
+redirectGoogleMap(): void
+redirectUI(): void
<<boundary>> UI
+hawkerSignUpPage(): void
+userSignUpPage(): void
+signUpMessage(messageCode: int): void
+submit(email: String, password: String, operatingLicense: ImageType): void
+submit(email: String, password: String): void
+loginPage(): void
+loginMessage(messageCode: int): void
+forgotPasswordPage(): void
+submitResetEmail(email: String): void
+removeAccountConfirmationPage(): void
+removeAccountPage(): void
<<boundary>> GoogleMapUI
+detectLocation(): float[]
+returnLocationData(): float[]
<<control>> mapController
+requestLocatoin(): void
+returnLocation(): float[]
<<boundary>> ReviewerUI
-status: String
+returnAccountCreationStatus(status: String): bool
+displayApprovalPage(): void
Review
-rating: int
-description: String
+Review()
+getRating(): int
+getDescription(): String
Hawker
-email: String
-password: String
-license: ImageType
+Hawker()
+getEmail(): String
+getPassword(): String
+license():ImageType
User
-email: String
-password: String
+User()
+getEmail(): String
+getPassword(): String
HawkerCenter
-stall: Stall[]
-locatoin: float[]
+HawkerCenter()
+getLocation(): float[]
+getStall(): Stall[]
Stall
-stallName: String
-Menu: MenuItem[]
-review: Review[]
+Stall()
+getStallName(): String
+getMenu(): MenuItem[]
+getReview(): Review[]
MenuItem
-itemName: String
-price: double
-picture: ImageType
-description: String
+MenuItem()
+getItemName(): String
+getPrice(): double
+getPicture(): ImageType
+getDescription(): String
<<user>> MapSystem
-distance: String
-location: String
+retrieveData(distance): String[]
+requestRealTimeLocation(location): String[]
+showArriveDestination(destination: String): void
+displayButton(buttonCode: int): void
<<control>> HawkerSignupValidator
-email: String
-password: String
-license: ImageType
+displayMessage(messageCode: int): void
+findEmail(email: String): bool
+validateEmail(email: String): bool
+validatePassword(password: String): bool
+store(email: String, password: String): void
+requestConfirmationEmail(email: String): void
+reuestApproval(operatingLicense: ImageType): bool
<<control>> updateValidator
+displayPage(pageIndex: int): void
+displayMessage(messageIndex: int): void
+request(requestIndex: int): void
+display(searchResults: String[]): void
+filterData(): String []
+displayRating(chosenStar: int): void
+askFaultReportNoti(stall: String): String[]
+deleteFaultNoti(faultReportNotiIndex: int, stall_Name: String): void
+askReviewNoti(stall: String): String[]
+deleteReviewNoti(reiewNoti: int, stall_Name: String): void
+askMenu(stall: String): MenuItem[]
+askOpeningHours(stall: String): int[]
<<control>> searchValidator
+display(searhcResults: String[]): void
+filterData(): String[]
<<control>> UserSignupValidator
-email: String
-password: String
+displayMessage(messageCode: int): void
+findEmail(email: String): bool
+validateEmail(email: String): bool
+validatePassword(password: String): bool
+store(email: String, password: String): void
+requestConfirmationEmail(email: String): void
<<control>> LoginValidator
-email: String
-password: String
+findEmail(email: String): bool
+findPassword(password: String): bool
+submit(email, password: String): void
<<control>> ResetPasswordValidator
-email: String
+findEmail(email: String): bool
+requestSendEmail(email: String): void
+displayMessage(messageCode: int): void
<<control>> AccountValidator
-email: String
-password: String
+displayMessage(messageCode: int): void
+findPassword(password: String): bool
+requestPassword(): String
+submit(email, password: String): void
+displayButton(buttonCode: int): void
<<control>> ResetPwdWebValidator
+requestPassword(): void
+validatePassword(password: String): void
+displayMessage(messageCode: int): void
+updatePassword(email: String, password: String): void
+requestSendEmail(email: String): void
<<boundary>> ResetPwdWebUI
+resetPasswordPage(): void
+resetMessage(messageCode: int): void
<<entity>> Database
-hawker_data: Hawker[]
-user_data: User[]
-hawkerCenter_data: HawkerCenter[]
+Database()
+store(email, password: String, user_type: bool): void
+retrieveData(search: String): String[]
+updatePassword(email, password: String): String
+updateMenu(menuItem, stall: String, update_type: int): void
-updateMenuPrice(menuItem, stall: String, newPrice: float):void
-updateMenuDescription(menuItem, stall, description: String): void
-updateMenuPicture(menuItem, stall: String, image: ImageType): void
+addMenuItem(menuItem, stall, description: String, price: float, image: ImageType): void
+findPassword(email, password: String): bool
+deleteAccount(email: String): void
+updateCrowdSituation(crowdLevel: int, stall: String): void
+getFaultReportNoti(stall: String): String[]
+removeFaultReport(stall: String, noti_fault_index: int): void
+getReviewNoti(stall: String): String[]
+removeReviewNoti(stall: String, noti_review_index: int): void
+findEmail(email: String): bool
+getMenu(stall: String): MenuItem[]
+getOpeningHours(stall: String): int[][]
+updateOpeningHours(stall: String, openingTime, closingTime: int, daysToOpen: int[]): void
+updateOpeningStatus(stall: String, status: bool): void
+requestNearbyHawkerCenter(location: float[]): int
+reqeustHawkerStall(HawkerCenterIndex: int): String
+requestStallData(Stall_index: int): Stall
+requestReviewData(Stall_index: int): String[]
+requestFaultReport(Stall_index: int): String[]
+updateReview(chosen_star: int, description, stall: String): void
<<use>>
<<call>>
<<read>>

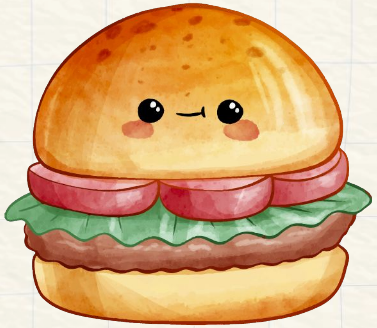
Dialog Map



Sign Up Page
do / allow input
account successfully created
sign up
request pwd reset
User
Landing Login Page
do/ show options
forgot pwd
ForgotPwdPage
do / allow input
ResetPwdPage
do/ allow input
sends reset email to user
Domain = Hawker
Domain = User
Hawker Domain
Fault Report Page
view reports received
view fault report
back
Hawker Main Page
do/ show options
update opening hours
back
Opening Hours Page
allow hawker to update
shut down shop
update menu
Delete Account
option to close store
back
back
Stall Menu Page
allow hawker to update
User Domain
click directions
Directions
display route in red
Hawker Centre Information
display stalls details
click hawker centre
Customers Main Page
display map/ option to select hawker stall
click report
Fault Report Page
option to report a fault
click review
reflects overall rating
click stall
find stall via searchbar
close
click info
Stall Review Page
display reviews/ option to leave a review
Stall Menu Page
display menu items
Stall Information
display individual stall information
Fault Report Info Page
displays fault report definition
reflects overall rating
reflects crowd level
leave a review
Leave Review Page
option to write a review
display fault reports to the stall hawker

4.

Design Pattern



MVC Pattern - Justification

Cater to 3 Distinct User Groups

- Hawker, User, Admin
- These groups interact with the app in different ways

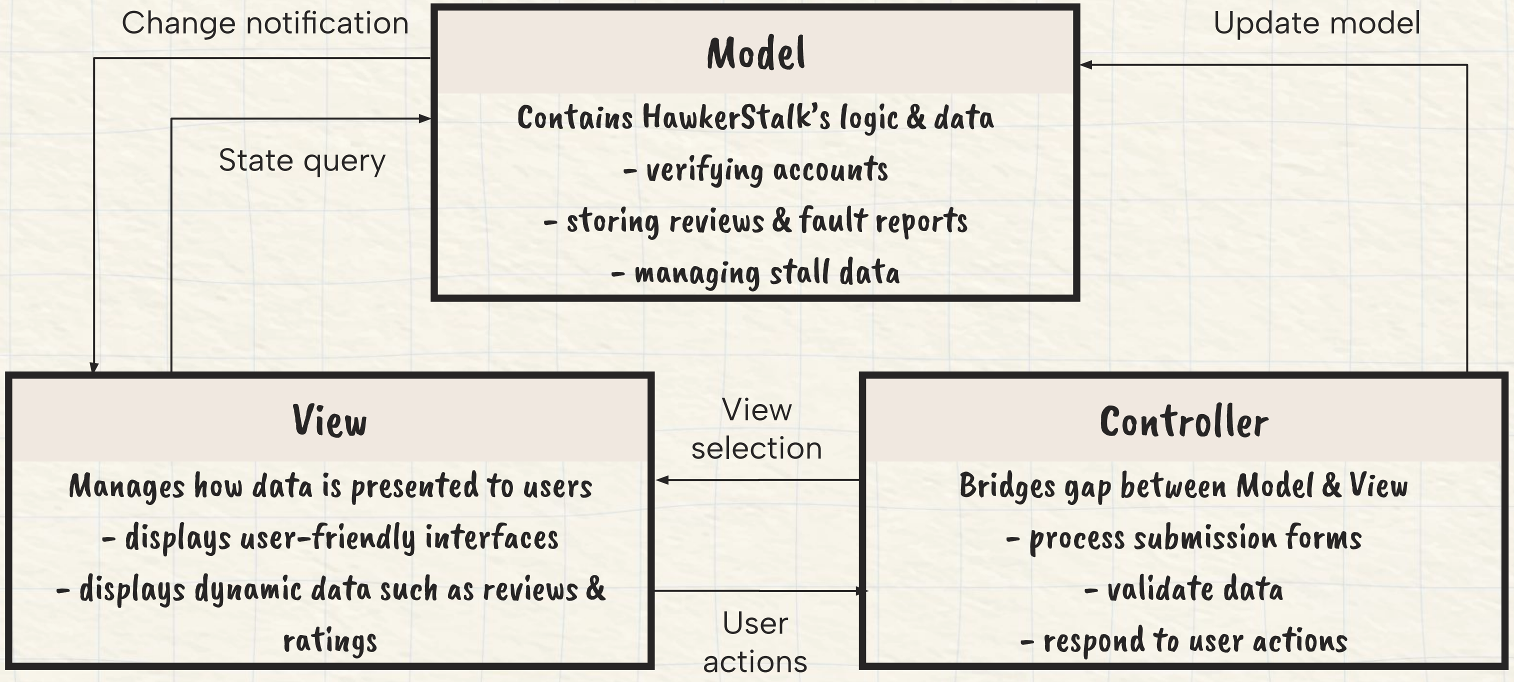
Handle Real-Time Data Efficiently

- Dynamic and responsive
- Real-time updates on stall operating status

Good Scalability & Maintainability

- Can modify each M-V-C part independently
- Can add more features with minimal bugs

MVC Pattern



Model

Contains HawkerStalk's logic & data
- verifying accounts
- storing reviews & fault reports
- managing stall data

View

Manages how data is presented to users
- displays user-friendly interfaces
- displays dynamic data such as reviews & ratings

Controller

Bridges gap between Model & View
- process submission forms
- validate data
- respond to user actions

Observer & Strategy Pattern in MVC

Observer Pattern

The Observer Pattern is applied between the Model and the View layers to automatically update the view when changes occur in the model.

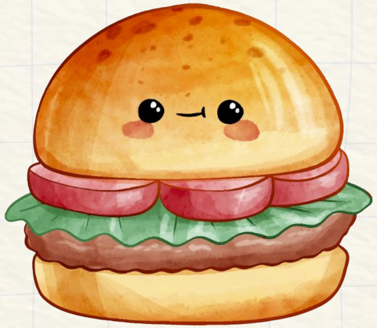
- Real-time operating status update
- Real-time review & ratings
- Menu & operating-hour updates

Strategy Pattern

The Strategy Pattern is useful in the controller layer to manage different business logic and behavior without hardcoding any one solution.

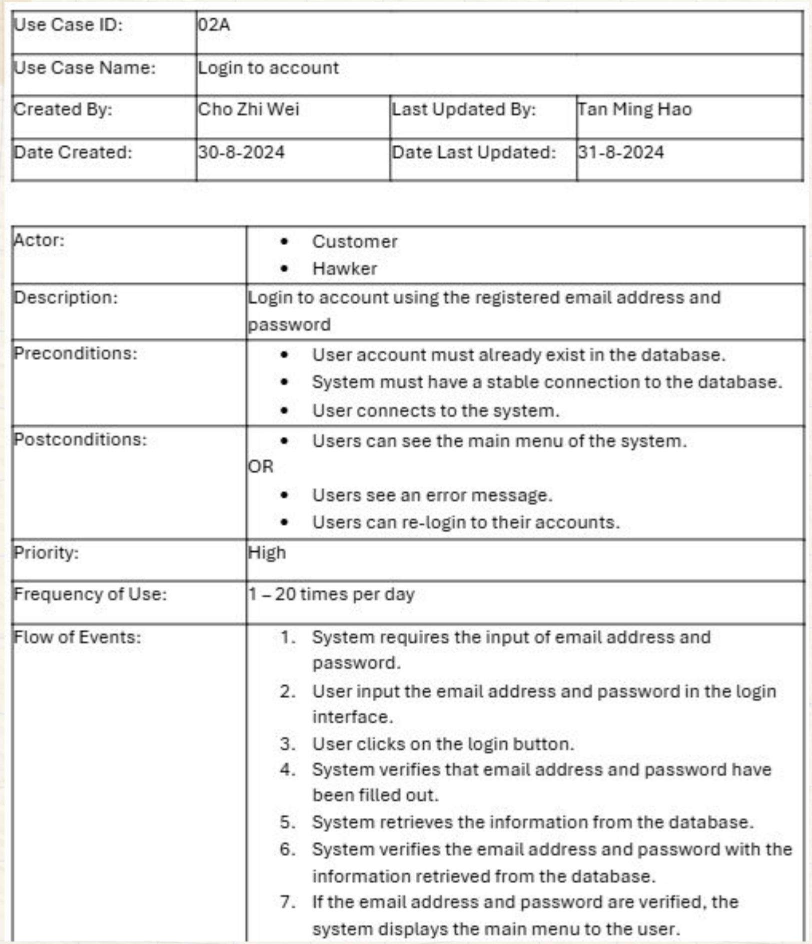
- Search & filter options
- Sorting & ranking of results
- Rating calculation

5. Traceability in Project Deliverables

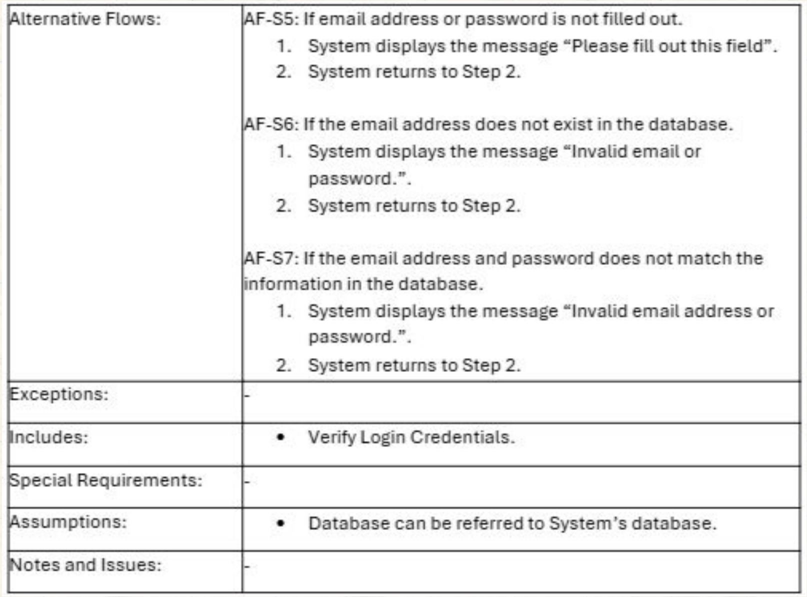


Use Case Description

Login

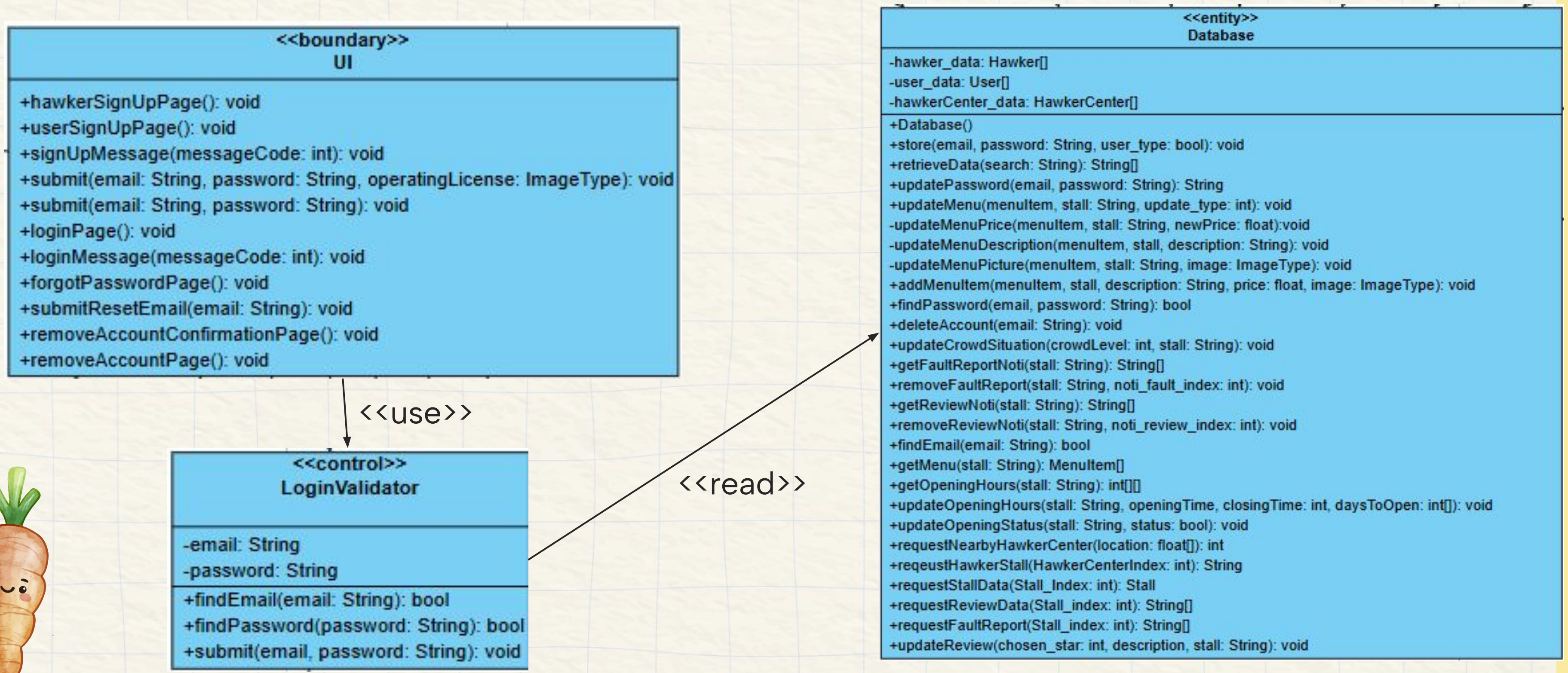


Use Case ID:	02A		
Use Case Name:	Login to account		
Created By:	Cho Zhi Wei	Last Updated By:	Tan Ming Hao
Date Created:	30-8-2024	Date Last Updated:	31-8-2024



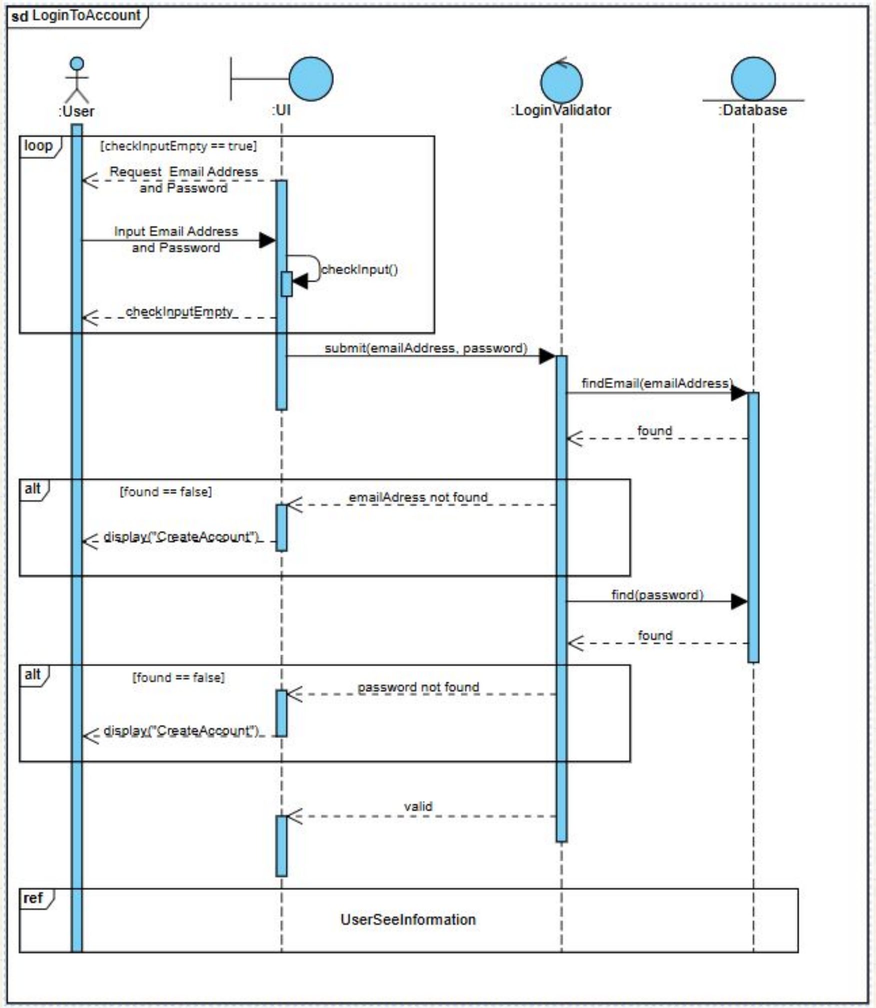
Actor:	• Customer • Hawker
Description:	Login to account using the registered email address and password
Preconditions:	• User account must already exist in the database. • System must have a stable connection to the database. • User connects to the system.
Postconditions:	• Users can see the main menu 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:	-

Relevant Class Diagram (Login)

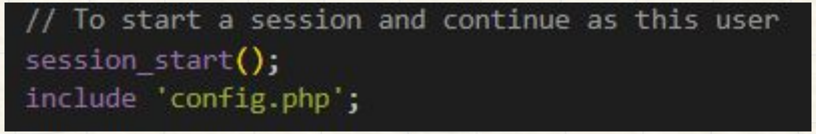

<<boundary>>
UI
+hawkerSignUpPage(): void
+userSignUpPage(): void
+signUpMessage(messageCode: int): void
+submit(email: String, password: String, operatingLicense: ImageType): void
+submit(email: String, password: String): void
+loginPage(): void
+loginMessage(messageCode: int): void
+forgotPasswordPage(): void
+submitResetEmail(email: String): void
+removeAccountConfirmationPage(): void
+removeAccountPage(): void
<<use>>
<<control>>
LoginValidator
-email: String
-password: String
+findEmail(email: String): bool
+findPassword(password: String): bool
+submit(email, password: String): void
<<read>>
<<entity>>
Database
-hawker_data: Hawker[]
-user_data: User[]
-hawkerCenter_data: HawkerCenter[]
+Database()
+store(email, password: String, user_type: bool): void
+retrieveData(search: String): String[]
+updatePassword(email, password: String): String
+updateMenu(menuItem, stall: String, update_type: int): void
-updateMenuPrice(menuItem, stall: String, newPrice: float):void
-updateMenuDescription(menuItem, stall, description: String): void
-updateMenuPicture(menuItem, stall: String, image: ImageType): void
+addMenuItem(menuItem, stall, description: String, price: float, image: ImageType): void
+findPassword(email, password: String): bool
+deleteAccount(email: String): void
+updateCrowdSituation(crowdLevel: int, stall: String): void
+getFaultReportNoti(stall: String): String[]
+removeFaultReport(stall: String, noti_fault_index: int): void
+getReviewNoti(stall: String): String[]
+removeReviewNoti(stall: String, noti_review_index: int): void
+findEmail(email: String): bool
+getMenu(stall: String): MenuItem[]
+getOpeningHours(stall: String): int[][]
+updateOpeningHours(stall: String, openingTime, closingTime: int, daysToOpen: int[]): void
+updateOpeningStatus(stall: String, status: bool): void
+requestNearbyHawkerCenter(location: float[]): int
+reqeustHawkerStall(HawkerCenterIndex: int): String
+requestStallData(Stall_Index: int): Stall
+requestReviewData(Stall_index: int): String[]
+requestFaultReport(Stall_index: int): String[]
+updateReview(chosen_star: int, description, stall: String): void

Relevant Sequence Diagram

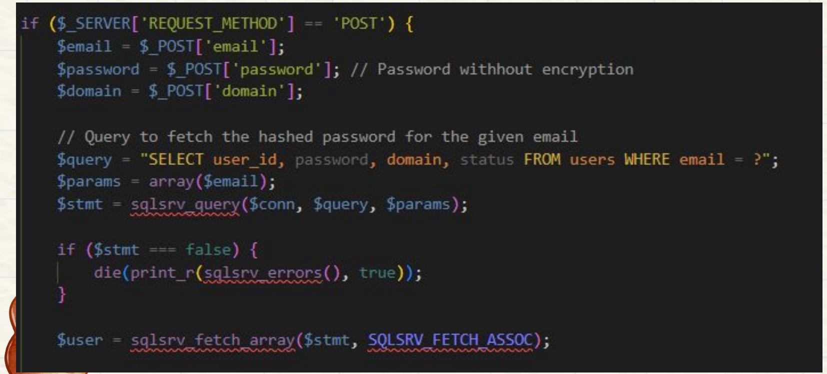
Login



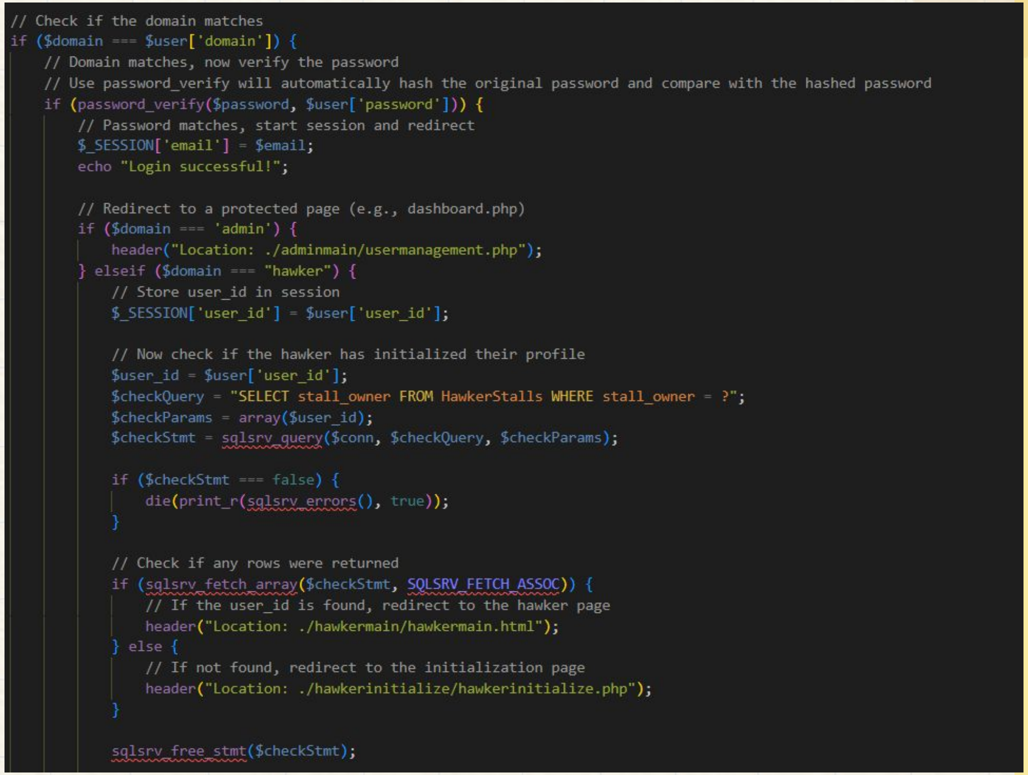
Good Designs Applied



```php
// To start a session and continue as this user
session_start();
include 'config.php';
```



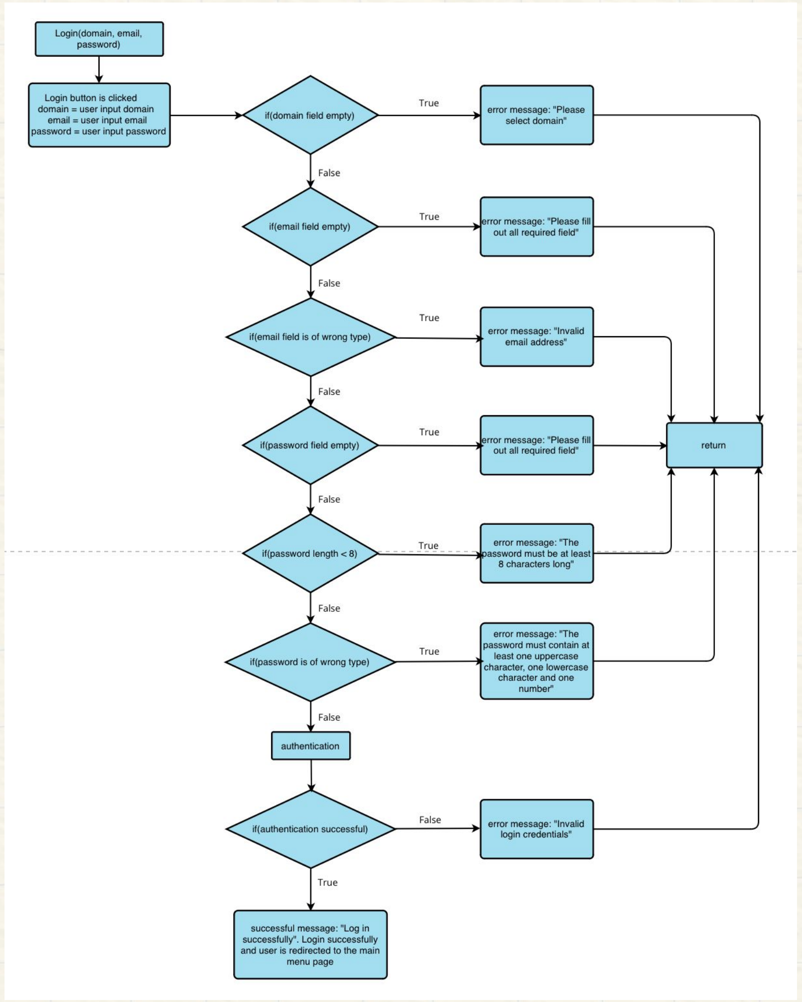
```php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $email = $_POST['email'];
    $password = $_POST['password']; // Password withhout encryption
    $domain = $_POST['domain'];

    // Query to fetch the hashed password for the given email
    $query = "SELECT user_id, password, domain, status FROM users WHERE email = ?";
    $params = array($email);
    $stmt = sqlsrv_query($conn, $query, $params);

    if ($stmt === false) {
        die(print_r(sqlsrv_errors(), true));
    }

    $user = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_ASSOC);
```



```php
// Check if the domain matches
if ($domain === $user['domain']) {
    // Domain matches, now verify the password
    // Use password_verify will automatically hash the original password and compare with the hashed password
    if (password_verify($password, $user['password'])) {
        // Password matches, start session and redirect
        $_SESSION['email'] = $email;
        echo "Login successful!";

        // Redirect to a protected page (e.g., dashboard.php)
        if ($domain === 'admin') {
            header("Location: ./adminmain/usermanagement.php");
        } elseif ($domain === "hawker") {
            // Store user_id in session
            $_SESSION['user_id'] = $user['user_id'];

            // Now check if the hawker has initialized their profile
            $user_id = $user['user_id'];
            $checkQuery = "SELECT stall_owner FROM HawkerStalls WHERE stall_owner = ?";
            $checkParams = array($user_id);
            $checkStmt = sqlsrv_query($conn, $checkQuery, $checkParams);

            if ($checkStmt === false) {
                die(print_r(sqlsrv_errors(), true));
            }

            // Check if any rows were returned
            if (sqlsrv_fetch_array($checkStmt, SQLSRV_FETCH_ASSOC)) {
                // If the user_id is found, redirect to the hawker page
                header("Location: ./hawkermain/hawkermain.html");
            } else {
                // If not found, redirect to the initialization page
                header("Location: ./hawkerinitialize/hawkerinitialize.php");
            }

            sqlsrv_free_stmt($checkStmt);
```

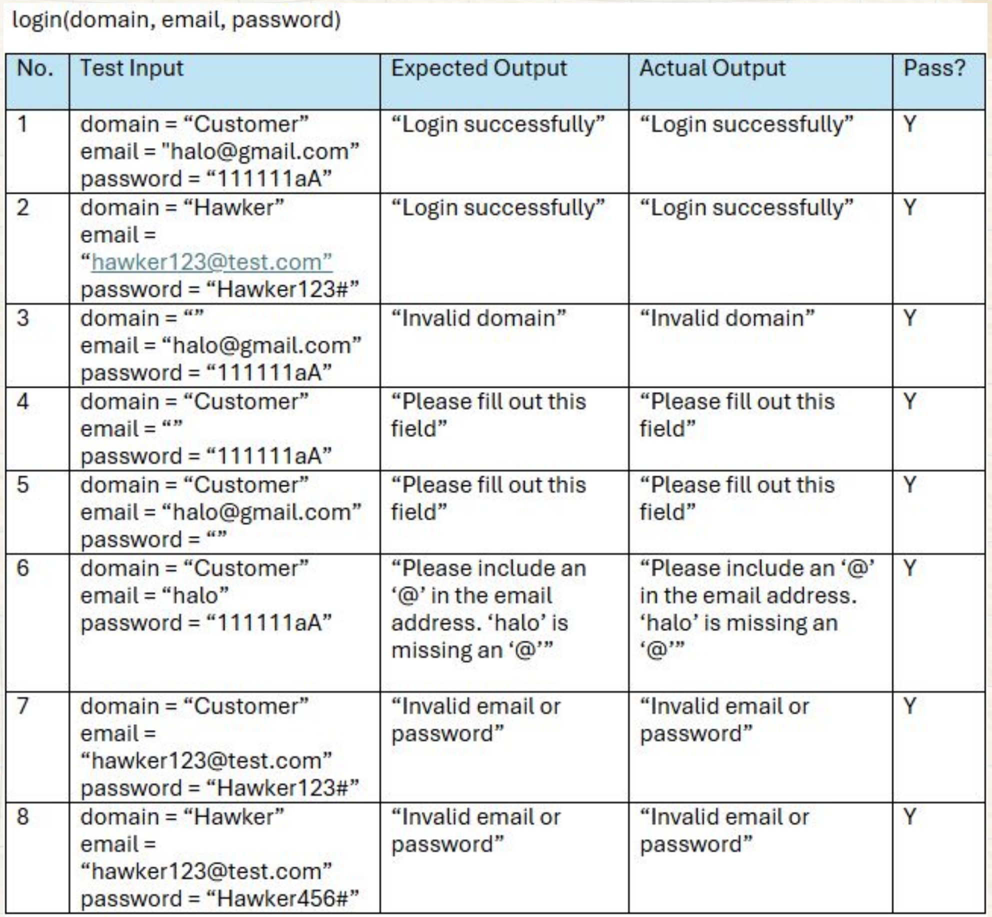
Black and White Box Testing

Control Flow Graph (Login)


Login(domain, email, password)
Login button is clicked domain = user input domain email = user input email password = user input password
if(domain field empty)
True
error message: "Please select domain"
False
if(email field empty)
True
error message: "Please fill out all required field"
False
if(email field is of wrong type)
True
error message: "Invalid email address"
False
if(password field empty)
True
error message: "Please fill out all required field"
False
if(password length < 8)
True
error message: "The password must be at least 8 characters long"
False
if(password is of wrong type)
True
error message: "The password must contain at least one uppercase character, one lowercase character and one number"
False
authentication
if(authentication successful)
False
error message: "Invalid login credentials"
True
successful message: "Log in successfully". Login successfully and user is redirected to the main menu page
return

Black and White Box Testing

Test Cases and Results (Login)


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

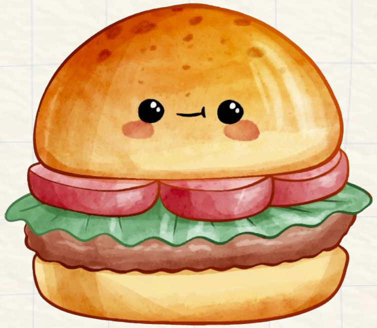
6	domain = “Customer”
	email = “halo”
	password = “111111aA”
		“Please include an ‘@’ in the email address. ‘halo’ is missing an ‘@’”	“Please include an ‘@’ in the email address. ‘halo’ is missing an ‘@’”	Y

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

6.

Future Prospects



Future Prospects

Live Crowd Monitoring

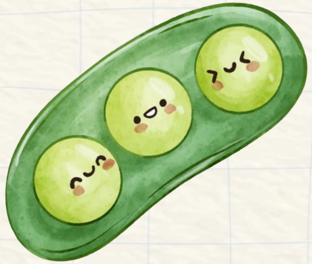
Use data from mobile networks or sensors to inform users of **peak and off-peak hours**.

Personalised Recommendations

Use **AI to recommend stalls or dishes** based on user search history, stall ratings and user's past preferences.

Mobile Payment Integration

Allow users to **pay directly** through the app via digital wallets, credit cards or QR codes.



Thank You!

Join Hawker Stalk to preserve Singapore's Hawker Culture with us!