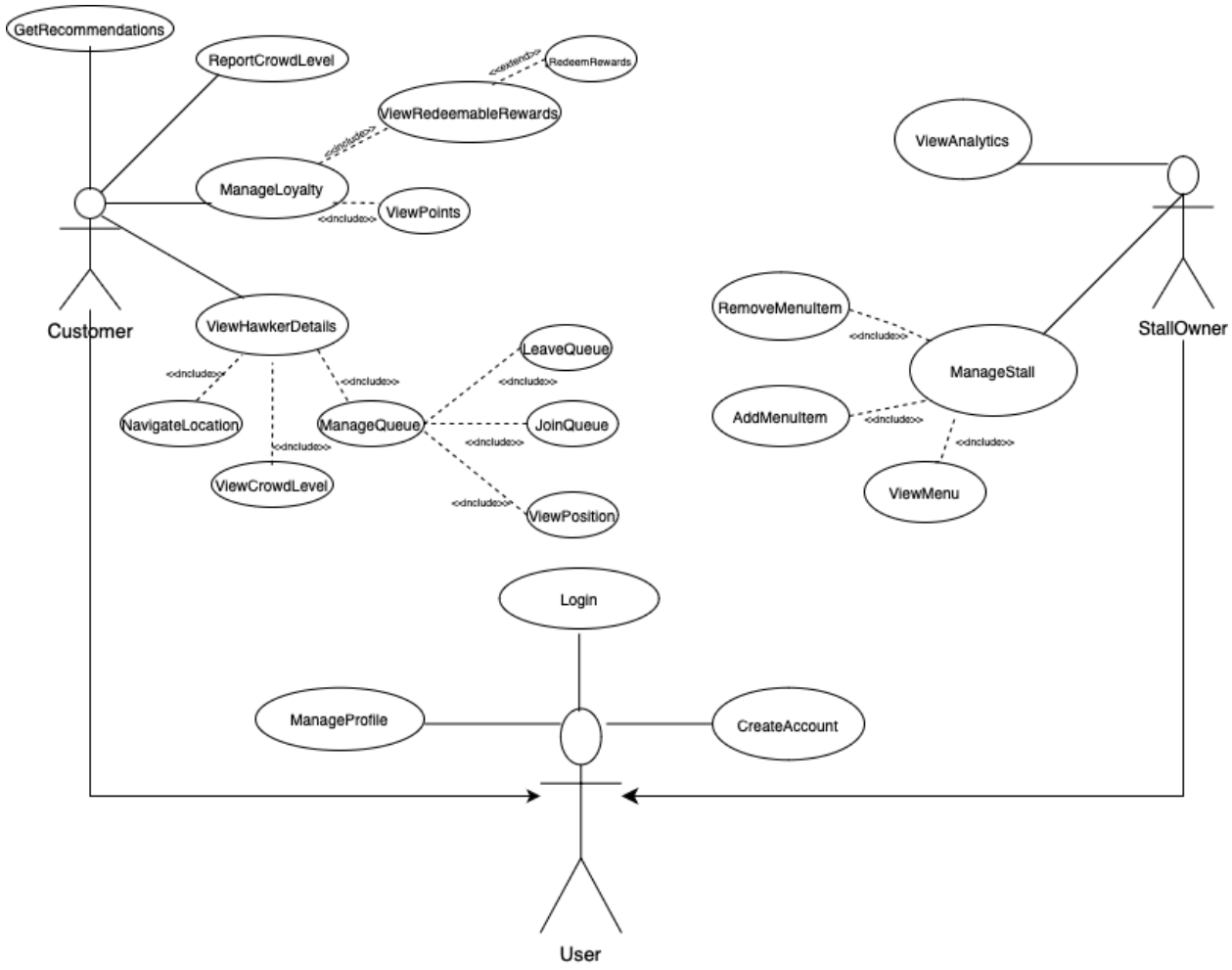
Use Case Diagram:



Functional Requirement #1

1.0 Authorization

1.0.1 Login

Actor	User
Description	Allows the User to login to his/her HawkerGo account with their email and password
Preconditions	None
Postconditions	The User is logged into the HawkerGo application and is navigated to the home screen of the application.
Priority	High
Frequency of Use	High
Flow of events	1. The system allows the User to login with email and password 2. The User chooses to log in with email and password 3. The User enters his/her email and password. The password is hidden as dots, but the User can choose to see it by selecting the eye icon 4. The User selects the "Login" button
Alternative flows	None
Exceptions	Information missing or invalid: 1. If any required information is missing or invalid, an error message is displayed 2. If email and password do not match when the User tries to login in Step 4, System shall display “Email and password do not match” to the User
Includes	None
Special requirements	The system needs to validate User input data
Assumptions	The User has an existing HawkerGo Account
Notes and Issues	None

1.0.2 CreateAccount

Actor	User
Description	Allows a person to create an Account in HawkerGo to become a User
Preconditions	None
Postconditions	An account is created
Priority	High
Frequency	High
Flow of Events	1. The system prompts the User to enter their name, address, password, email, contact number, profile picture, and role. 2. The User enters the required information and selects the "Create Account" button to confirm his/her inputs. 3. An Account is created
Alternative Flow	<u>Returning to the Login page:</u> 1. If the User selects the back navigation icon, the system will navigate the User to the login page.
Exceptions	<u>Missing or invalid information:</u> 1. If any required information is missing or invalid, an error message is displayed.
Include	None
Special requirements	The system needs to validate User input data
Assumptions	None
Notes and Issues	None

1.0.3 ManageProfile

Actor	User
Description	This use case describes how Users (both hawkers and Customers) can manage their profiles within the HawkerGo app. This includes updating personal details, changing passwords, setting preferences, and managing contact information
Preconditions	1. The User must be registered in the HawkerGo app 2. The User must be logged into their account
Postconditions	1. Users profile is updated successfully 2. The updated information is stored successfully on system 3. User receives confirmation message
Priority	High
Frequency	Moderate
Flow of Events	1. The User navigates to the profile management section in the HawkerGo app 2. The system displays the User's current profile information 3. The User selects the field(s) they want to update (e.g., name, email, phone number, password, preferences) 4. The User enters the new information 5. The User submits the changes 6. The system validates the new information and updates the profile 7. The system confirms the successful update to the User
Alternative Flow	None
Exceptions	1. The system displays an error message if the User enters invalid information (e.g., incorrect email format or weak password) 2. If there is a system error, the User is notified, and the update is not saved 3. If the User tries to update with an email or phone number already in use, the system rejects the change
Include	None
Special requirements	None
Assumptions	Users provide accurate information
Notes and Issues	None

Functional Requirement #2

1.1 ReportCrowdLevel

Actor	Customer
Description	Allows the Customer to report crowd level according to 3 different tiers; low, medium or high. The Customer can toggle between the 3 levels while waiting in line
Preconditions	1. The Customer must have an active account i.e (logged into the app) 2. Customer must have their location services enabled to submit reports of nearby Hawker Centres 3. Should have access to an internet
Postconditions	Crowd level is stored in the system
Priority	High
Frequency of Use	High
Flow of Events	1. Customer reports crowd level by selecting the appropriate crowd level from predefined levels: low, medium or high 2. Customer submits report 3. The System saves the report to the database 4. The System updates the hawker centres' real-time crowd estimate 5. The System displays a confirmation message to the User
Alternative Flows	1. The Customer manually selects the location of their desired hawker centre 2. The System checks Customers' current location 3. If the Customer isn't at the location of hawker centre, system prevents input of crowd level 4. The system informs the Customer that reporting is restricted to Customers physically present at the location
Exceptions	<u>Repeated Reporting:</u> 1. If the Customer repeatedly reports crowd level more than 5 times consecutively, crowd reporting action will be suspended for 10 minutes
Includes	None
Special Requirements	None
Assumptions	The Customer reports accurate crowd-level estimates

Notes and Issues	None

1.2 GetRecommendations

Actor	Customer
Description	Customers get food suggestions based on their dietary preferences, meal type, and other criteria
Preconditions	1. The Customer is logged into the System 2. The System has access to all the food options available in real-time
Postconditions	1. The Customer can view food options and receives a list of food recommendations based on their inputs
Priority	Moderate
Frequency	Moderate
Flow of events	1. Customer selects GetRecommendations 2. The System prompts Customer to input a. Cuisine type b. Dietary restrictions c. Price range 3. System processes the inputs and filters out the hawker options also taking into consideration the location and reviews that match the Customers inputs 4. The System displays a list of food recommendations from the hawker stalls, based on Customer inputs
Alternative Flows	<u>The Customer wants more information:</u> 1. The Customer requests details on the meal, eg calories, ingredients, etc <u>No suitable Options available:</u> 1. The System displays messages and suggests broader alternatives
Exceptions	<u>Invalid Customer input:</u> 1. The System prompts Customer to change input or enter input again
Includes	None
Special Requirements	None
Assumption	None
Notes and Issue	None

1.3 ManageLoyalty

Actor	Customer
Description	Helps Customers track loyalty points and rewards
Preconditions	The Customer should have an account and must be logged in
Postconditions	Loyalty points are updated and stored in the system.
Priority	High
Frequency	Moderate
Flow of events	1. The Customer logs into the HawkerGo app 2. The Customer selects the ManageLoyalty option 3. Customers can choose to either ViewPoints, ViewRedeemableRewards, or RedeemRewards 4. If the Customer selects Viewpoints, the use case ViewPoints displays Customers' points which have been currently accumulated 5. If the Customer selects ViewRedeemableRewards, the use case ViewRedeemableRewards displays rewards which can be redeemed, given the Customer has sufficient points to redeem them 6. If the Customer selects RedeemRewards, the use case RedeemRewards will update the Customer’s points balance if a reward is redeemed 7. The System confirms transactions
Alternative Flows	None
Exceptions	None
Includes	1. Viewpoints 2. ViewRedeemableRewards 3. RedeemRewards
Special Requirements	The HawkerGo app supports push notifications for rewards
Assumption	Users get points based on interactions
Notes and Issue	None

1.3.1 ViewPoints

Actor	Customer
Description	Customers can view accumulated points
Preconditions	1. The Customer is logged in 2. The System has accumulated the points for the Customer currently
Postconditions	1. The Customer can view the current point balance
Priority	Moderate
Frequency	Moderate
Flow of events	1. The Customer selects ViewPoints in the app 2. The System retrieves and displays the Customer's current point balance 3. Customers can see additional details such as how many points were earned/expired
Alternative Flows	<u>The Customer has no points:</u> 1. The System displays a message informing the Customer they have no points <u>Points display error</u> 1. Issue with retrieving points, the system displays an error message
Exceptions	None
Includes	None
Special Requirements	None
Assumption	None
Notes and Issue	None

1.3.2 ViewRedeemableRewards

Actor	Customer
Description	Customer can view the list of redeemable rewards
Preconditions	1. The Customer is logged into the system 2. There are rewards available to be redeemed
Postconditions	1. Customers view a list of rewards they can redeem using points
Priority	Moderate
Frequency	Moderate
Flow of events	1. The Customer selects ViewRedeemableRewards on the app 2. Based on the Customer's current point balance, the system finds all the rewards the Customer can redeem 3. The System displays a list of redeemable rewards 4. If the Customer selects RedeemRewards, the use case RedeemRewards allows the Customer to redeem rewards successfully 5. The Customer can view additional details on these rewards i.e terms/description
Alternative Flows	<u>The Customer has no redeemable rewards:</u> 1. The System displays a message saying there are no redeemable rewards
Exceptions	None
Includes	RedeemRewards
Special Requirements	None
Assumption	None
Notes and Issue	None

1.3.3 RedeemRewards

Actor	Customer
Description	Customers can use points to redeem any rewards from redeemable reward listing
Preconditions	The Customer is logged on to the System Rewards are available on the System
Postconditions	Customers can earn rewards
Priority	Moderate
Frequency	Moderate
Flow of events	1. The Customer selects RedeemRewards on the app 2. The System displays a list of rewards to be redeemed. 3. The Customer selects a reward to redeem 4. The System verified Customer has sufficient points in balance to redeem the reward 5. The System confirms redemption and deducts points from the balance, respectively. 6. The Customer receives a reward and is notified.
Alternative Flows	<u>The Customer wants to cancel the redemption:</u> 1. Customers may cancel the redemption process before completing it, and the System returns them to the rewards catalogue without changing their points balance.
Exceptions	<u>Insufficient points:</u> 1. The System displays insufficient points message and advises the Customer to earn more points <u>Reward is unavailable:</u> 1. If the reward is no longer available or has expired, the System informs the Customer and prompts them to choose a different reward.
Includes	None
Special Requirements	None
Assumption	None
Notes and Issue	None

1.4 ViewHawkerDetails

Actor	Customer
Description	Permits the Customer to view the hawker details, including location, crowd level and menu
Preconditions	1. The Customer must be logged into the app 2. Hawker center data must be available to the System
Postconditions	1. The Customer obtains relevant stall information
Priority	High
Frequency	High
Flow of events	1. The Customer logs into the app 2. Customer selects ViewHawkerDetails 3. The System prompts the Customer to choose a desired activity: NavigateLocation, ViewCrowdLevel, ManageQueue 4. If the Customer selects NavigateLocation, the Customer uses the included use case NavigateLocation to search for nearby hawkers 5. If the Customer selects ViewCrowdLevel, the Customer uses the included use case ViewCrowdLevel to gauge the crowd level of hawker centres 6. If the Customer selects ManageQueue, the Customer uses the included use case to view the queue levels
Alternative Flows	None
Exceptions	None
Includes	1.NavigateLocation 2.ViewCrowdLevel 3.ManageQueue
Special Requirements	None
Assumption	None
Notes and Issue	None

1.4.1 NavigateLocation

Actor	User
Description	Provides Users with directions to their selected hawker centre or stall
Preconditions	1. The User has selected a hawker stall 2. GPS and maps services are enabled
Postconditions	The User receives real-time directions to a hawker stall
Priority	High
Frequency	High
Flow of events	1. The User selects NavigateLocation after selecting the desired hawker stall 2. GPS detects Users' current location 3. GPS provides a route from the User's current location to the hawker stall 4. The User follows the route and the system concurrently tracks the User's location
Alternative Flows	<u>User goes the wrong way:</u> 1. If the User follows the wrong direction, mapping services will automatically, recalculate the distance, and provide an adjusted route accordingly
Exceptions	<u>GPS is disabled:</u> 1. If GPS is disabled, the System prompts the User to enable location services
Includes	None
Special Requirements	None
Assumption	None
Notes and Issue	None

1.4.2 ViewCrowdLevel

Actor	Customer
Description	Allows Customers to view crowd levels of a selected location
Preconditions	1. Customer should be logged into the System 2. System should have access to real time crowd data
Postconditions	1. Customer successfully views crowd-level data information for the selected location
Priority	High
Frequency	Moderate
Flow of events	1. Customer navigates to ViewCrowdLevel feature 2. System prompts Customer to select desired location 3. The System uses real-time data, historical trends and Customer reports to calculate and display crowd level based on predefined terms ie: low, medium, high 4. Customer makes decision based on crowd level data
Alternative Flows	<u>System can't retrieve data:</u> 1. Customer is informed that System can't retrieve real-time data 2. System displays last known crowd level data along with timestamp
Exceptions	<u>The system fails to fetch data:</u> 1. The System fails to fetch data due to connectivity issues, and an error message is displayed 2. The selected location has no available crowd data, prompting the Customer to check another place.
Includes	None
Special Requirements	None
Assumption	None
Notes and Issue	None

1.4.3 ManageQueue (User)

Actor	Customer
Description	Allows Customer to view queue for desired hawker stall
Preconditions	1. The Customer has access to the queue management System 2. The System has an active queue for a specific location/service
Postconditions	1. Customer successfully joins, view, or manages their position in the queue
Priority	Moderate
Frequency	Moderate
Flow of events	1. Customer selects desired location 2. Customer selects manage queue 3. The system displays current queue status 4. The Customer can choose to a. Join the queue b. View the position of the queue c. Leave the queue 5. System updates queue and notifies Customer for any changes 6. System alerts Customer when their turn is nearing 7. System removes Customer from the queue once their order is fulfilled
Alternative Flows	<u>Customer missed their number:</u> 1. System reassigns Customer to the back of the line <u>Customer has to reschedule their order for a later time:</u> 1. Instead of waiting, the Customer selects a later time slot (if available) 2. The System updates their queue position accordingly
Exceptions	<u>Queue is Full:</u> 1. System will notify Customer if queue is full prompting them to join another queue <u>Customer tries to join multiple queues:</u> 1. System prevents Customer from joining multiple queues concurrently, and will display error message
Includes	1. JoinQueue 2. ViewPosition 3. LeaveQueue

Special Requirements	None
Assumption	None
Notes and Issue	None

1.4.3.1 JoinQueue

Actor	Customer
Description	Customers can join a virtual queue for a hawker stall through our HawkerGo app
Preconditions	1. Customer has logged into the System 2. Hawker stall supports virtual queueing 3. Queue is open for other Customers to join
Postconditions	1. Customer joins the virtual queue and is assigned a queue number 2. Queue position updates in real-time
Priority	High
Frequency	High
Flow of Events	1. The Customer selects a hawker stall in the app 2. The Customer selects the "Join Queue" button 3. The System validates the stall's queue status 4. The system assigns a queue number to the Customer 5. The system updates the Customer's position in the queue in real-time 6. The System notifies the Customer when their turn is approaching via a notification
Alternative Flow	<u>Queue if full:</u> 1. If queue is full, System notifies the Customer and prevents more Customers from joining
Exceptions	<u>The stall closes before the Customer joins:</u> 1. The System prevents Customer from joining and notifies the Customer
Include	None
Special requirements	System should update queue status dynamically
Assumptions	Customer will check app for queue status updates
Notes and Issues	None

1.4.3.2 ViewPosition

Actor	Customer
Description	Customers can check their position in the queue using the HawkerGo app
Preconditions	1. The Customer is logged in 2. The Customer has joined the queue
Postconditions	1. The Customer can see their current position in the queue 2. The system should update the queue position dynamically
Priority	High
Frequency	High
Flow of Events	1. The Customer navigates to their active queue 2. The System retrieves and displays the current queue position 3. The System updates the queue position in real-time
Alternative Flow	<u>Queue ends:</u> 1. If the queue has ended, the System informs the User
Exceptions	<u>The System fails to retrieve the queue position:</u> 1. Error message is displayed
Include	None
Special requirements	Queue status must refresh in real-time
Assumptions	Customer regularly checks app for queue updates
Notes and Issues	None

1.4.3.3 LeaveQueue

Actor	Customer
Description	Customer can voluntarily exit the virtual queue before their turn
Preconditions	1. The Customer is logged into System 2. The Customer has joined a queue
Postconditions	1. The Customer is removed from the queue 2. The queue is updated for other Customers
Priority	High
Frequency	High
Flow of Events	1. The Customer navigates to their active queue 2. The Customer clicks the "Leave Queue" button 3. The system confirms the action with a prompt 4. The Customer confirms leaving the queue 5. The system removes the Customer from the queue 6. The queue updates for remaining Customers
Alternative Flow	<u>Doesn't want to leave queue:</u> 1. If the Customer changes their mind before confirming to leave the queue, they can click cancel and remain in the queue
Exceptions	None
Include	None
Special requirements	The queue must update dynamically after the Customer leaves
Assumptions	Customer leaves queue for personal reasons
Notes and Issues	None

Functional requirement #3

1.5 ManageStall

Actor	Stall Owner
Description	Allow Stall Owners to manage their stall's menu
Preconditions	Stall Owner is logged into the System
Postconditions	Menu is updated with the latest items that are available
Priority	Moderate
Frequency	Moderate
Flow of events	1. Stall Owner will be presented with 3 use cases a. RemoveMenuItem b. AddMenuItem c. ViewMenu 2. If Stall Owner selects RemoveMenuItem, then the Stall Owner uses the included use case RemoveMenuItem to remove items from the menu or mark them as unavailable 3. If Stall Owner selects AddMenuItem, then Stall Owner uses the included AddMenuItem use case to add more items into their menus 4. If Stall Owner selects ViewMenu, then Stall Owner uses the included ViewMenu use case to view the current items on their menu
Alternative Flows	None
Exceptions	None
Includes	1. RemoveMenuItem 2. AddMenuItem 3. ViewMenu
Special Requirements	None
Assumption	None
Notes and Issue	None

1.5.1 RemoveMenuItem

Actor	Stall Owner
Description	Allows Store Owner to remove unwanted items or mark items as unavailable
Precondition	1. Store Owner is logged in and authenticated 2. There exists at least 1 item on the Stall Owner's menu
Postconditions	1. The menu is updated with the latest list of available items
Priority	Moderate
Frequency	Moderate
Flow of events	1. If Stall Owner selects Remove_Item 2. Stall Owner selects which specific item to remove from the menu 3. The System deletes that item form the menu 4. The System will display the updated menu 5. If Stall Owner selects to mark an item as unavailable 6. Stall Owner specifies which item to mark 7. That item will be marked temporarily unavailable 8. The System will display the updated menu
Alternative Flows	None
Exceptions	None
Includes	None
Special Requirements	None
Assumption	None
Notes and Issue	None

1.5.2 AddMenuItem

Actor	Stall Owner
Description	Allows Store Owner to add an item to their menu
Preconditions	1. Store Owner is logged in and authenticated
Postconditions	2. The menu is updated with the latest list of available items
Priority	Moderate
Frequency	Moderate
Flow of events	1. Store Owner selects AddMenuItem 2. Store Owner enters the name, price and includes a picture of that item 3. The System will display the updated menu
Alternative Flows	<u>Doesn't provide sufficient information:</u> 1. If Store Owner does not provide either the name, price or picture, they will be prompted to provide the missing information
Exceptions	None
Includes	None
Special Requirements	None
Assumption	Store Owner has the name, price and picture of the menu item
Notes and Issue	None

1.5.3 ViewMenu

Actor	Stall Owner
Description	Allows Stall Owner to view their menu
Preconditions	1. Store Owner is logged in and authenticated
Postconditions	1. Store Owner sees menu items
Priority	High
Frequency	High
Flow of events	1. Store Owner selects View_Menu 2. System will display the name, price and picture of the items on the menu
Alternative Flows	<u>Menu is empty:</u> 1. If the menu does not contain any items, System will display “Menu is Empty”
Exceptions	None
Includes	None
Special Requirements	None
Assumption	None
Notes and Issue	None

1.6 ViewAnalytics

Actor	Stall Owner
Description	Allows Stall Owner to view their sales volume over a specific time period
Preconditions	1. Stall Owner is logged in and authenticated 2. Stall has past sales record
Postconditions	1. Stall Owner can view analytics of their sales volume
Priority	High
Frequency	High
Flow of events	1. Stall Owner selects ViewAnalytics 2. Stall Owner chooses a specific time range for the sales data 3. The System displays the corresponding sales volume
Alternative Flows	<u>No time specified:</u> 1. If no time period was specified, the time range will default to the last 30 days
Exceptions	None
Includes	None
Special Requirements	None
Assumption	Sales volume is updated accurately
Notes and Issue	None