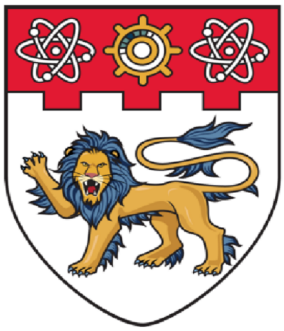


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

School of Computer Science & Engineering
CZ2006 Software Engineering

Project Name - QQueue
GenericSGTeam (DSS1)

Team Members:
Tan Liang Wei (Leader) (U2020479F)
Puah Yi Hao (U2021722G)
Ng Tze Minh (U2020036J)
Wong Kar Onn Nicholas (U2021172D)
Foo Junwei Lucas (U2022707B)

Table Of Contents

1. Introduction

1.1 Context

The popularity of the virtual queuing system has been growing especially during the COVID-19 pandemic. Many companies are turning to this cost-effective method that reduces the manpower needed for Service-Providers as well as a time-saving method for customers.

1.2 Purpose

The aim of QQueue app is to streamline the queuing process for all types of stores in Singapore, by providing an island-wide queue management system. We seek to design the first mobile application in Singapore providing live updates on the queue status of various stores in Singapore and remote queuing all via a single avenue.

QQueue serves three main purposes:

1) Providing users with up-to-date queue information on various stores across Singapore
2) Allowing users to join and leave queues of locations remotely
3) Allowing Service-Providers to gain publicity and better management of their customer queue

1.3 Document Overview

This report is a compilation of all lab deliverables in the CZ2006 course. It includes the following documentation:

1. Functional, Non-Functional Requirements
2. System Architecture
3. System Features (Use Cases, Conceptual Model and Sequence Diagrams)
4. Class Diagrams
5. Dialog Map

6. Testing (Whitebox and Blackbox)
7. Data Dictionary

All diagrams in this document are meant to give an overall preview. The diagrams are also inserted with a hyperlink to the actual image in the google drive for a full-sized view. Diagrams used are also included in the SVN folders.

1.4 Document Conventions

	Font	Size	Style
Title	Times New Roman	16	**Bold**
Subtitle	Times New Roman	14	**Bold**
Text	Times New Roman	12	Normal
Figure	Times New Roman	10	*Italicised*

1.5 Users and stakeholders

The stakeholders of QQueue include various F&B establishments and retail stores throughout Singapore, android mobile users, and the QQueue development team. Information on the location will be provided by the Service-Providers when they register their store in the application. Live time crowding details is obtained through the BestTimeAPI. Android mobile users will be able to use our application to view the wait times of locations and to join their queue via the app (Up to one queue concurrently). QQueue provides an easy platform for smaller stores to connect with a wider customer base with ease of managing their customer's information.

1.6 Intended Audience and Reading Suggestions

- Users and stakeholders mentioned above in *Section 1.5* who want to use QQueue to either find stores to queue for, or to have a platform to set up an online queue for their store
- Programming Developers who may be interested to further develop the project

The document is organised to be read sequentially to better understand how the features came about

1.7 Scope

QQueue app allows users to remotely view the queue times of locations listed on the app simply by selecting the location they are interested in. Users will then also be provided the option to join the queue of any location by tapping on the “Queue” button. Users may view their position in the queue at any time and a ping notification will be given to them when it is nearing their turn. An in-app QR code scanner will also be used to indicate a user has reached the location, thus checking out of the queue. The user's history is recorded and a recommendation engine is used to suggest potential stores that a user likes based on his previously visited stores.

1.8 Assumptions

- Users should have Internet access in order to use the application as retrieval of stores information is connected to the cloud service
- Users should be on an android device
- Users should have a GPS-enabled Android device capable of running the application
- BestTimeAPI does not provide exact numbers on the number of people at the location

2. Overall Description

2.1 User Classes and Characteristics

2.1.1 QQueue End-User

QQueue end-users are limited to the scope of Singapore, thus any active user will have to reside within the country to utilise the functions.

2.1.2 QQueue Service-Provider

QQueue Service-Providers have to be locally registered within Singapore and will be responsible for maintaining and updating the flow of their own queue. They can also maintain a better user ecosystem by looking at reviews from various users.

2.2 Operating Environment

- Android Oreo
- Android (Q)
- Android (R)

2.3 Design and Implementation Constraints

- We retrieve data from BestTimeAPI and this data is added to MongoDB. The stored data is then displayed on the phone. If the phone has full memory, it may not be able to render the information and store it properly.
- Due to hardware limitations, connection to the web takes time and the stores and recommendations may be slower as a result. We tried to solve this by using asynchronous connections when retrieving the data.
- BestTimeAPI has a limit to its free usage. As part of our development process, we pulled 300 stores via the API and then developed using only that data of 300 entries. This compromises the accuracy of our estimated wait time of stores which needs to be pulled consistently. We tried to solve this by generating our own algorithm to estimate the wait time based on the number of users.
- Comments are written alongside the code, so that it can be easily maintained and read by other clients.

2.4 User Stories

Story Description	Acceptance Criteria
Tom is a 24-year-old young adult who just graduated and started working. Eager to climb the corporate ladder and find his place in the company, he spends a great deal of time working. As such, the opportunities that he finds to spend time with his friends and family are precious to him. For Tom, having a nice meal with them after a long day of work is all he really wants sometimes. However, owing to his tight and busy work schedule, he finds it difficult to commit to reserving seats at an establishment, lest he disappoint his friends and family. Instead, he typically queues with them for up to 30 mins, taking the opportunity to catch up with them while waiting.	● View a store near his location with a queue time under 30 minutes ● Join the queue successfully
Tze Minh is an impatient 21-year-old student who wants to visit fancy stores with her friends. She enters the mall and is unsure what to do but she immediately would like to see something popular yet has a reasonable queue under 15 minutes	● Filter stores by category ● View the queue time of those stores ● Join a queue of a store under 15 minutes

3. External Interface Requirements

3.1 User Interfaces

The following are the initial UI mockups done by our group and reflect most (not all) the user interfaces later implemented in our application:

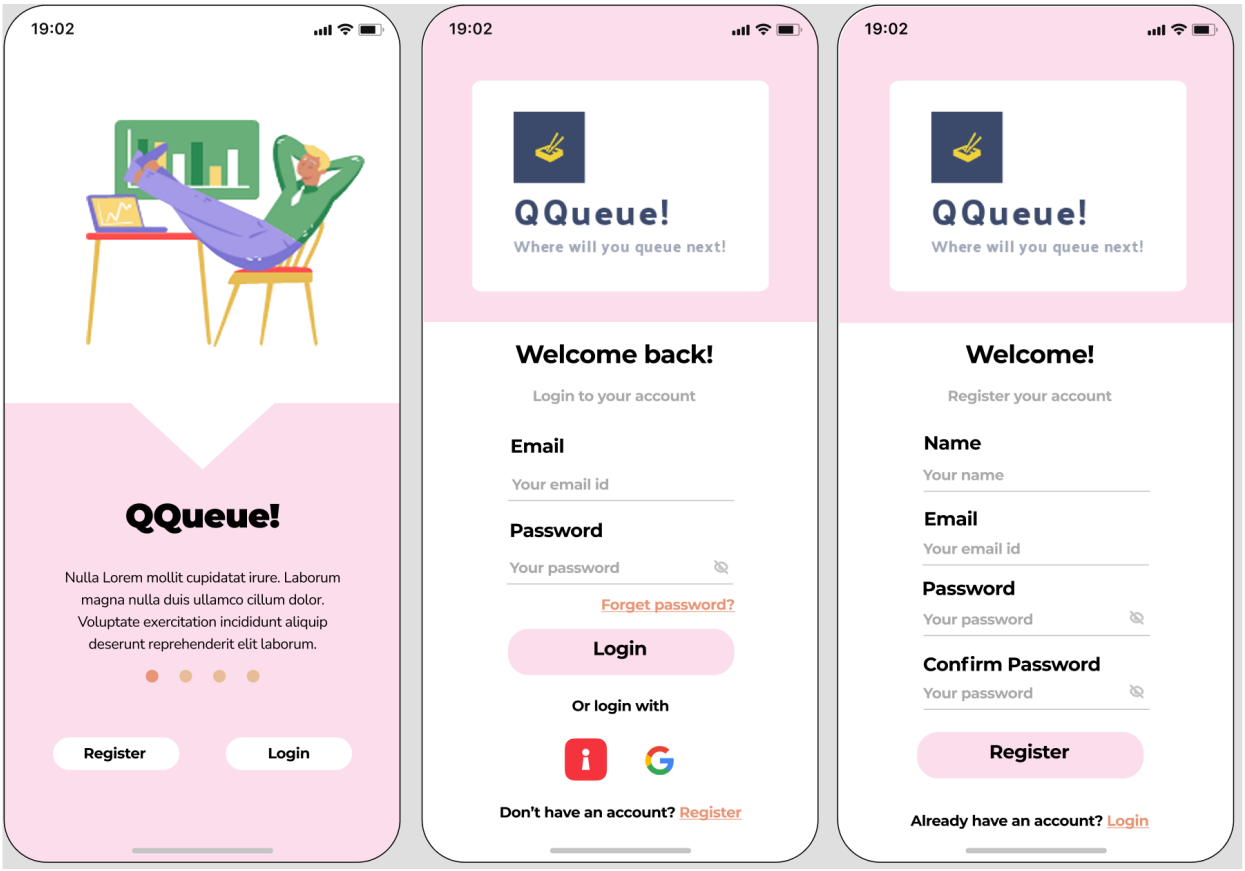


Figure 1: Login & Registration Page

Figure 1 shows the user interface for the onboarding screen, login and registration. The user interface is kept simple and designed to be similar to the login and registration screens of other commonly used applications. This will enforce a popular HCI design principle "Strive for consistency".

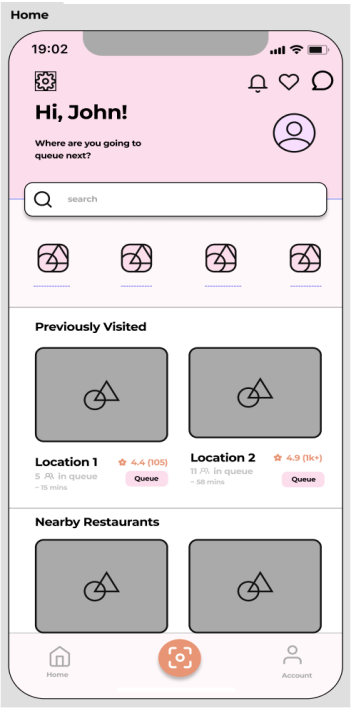


Figure 2: Home Screen

Figure 2 shows the user interface for the home page. The stores of nearby restaurants and previously visited/recommendations are immediately displayed to users. The various stores can be viewed by either searching or sliding left/right.

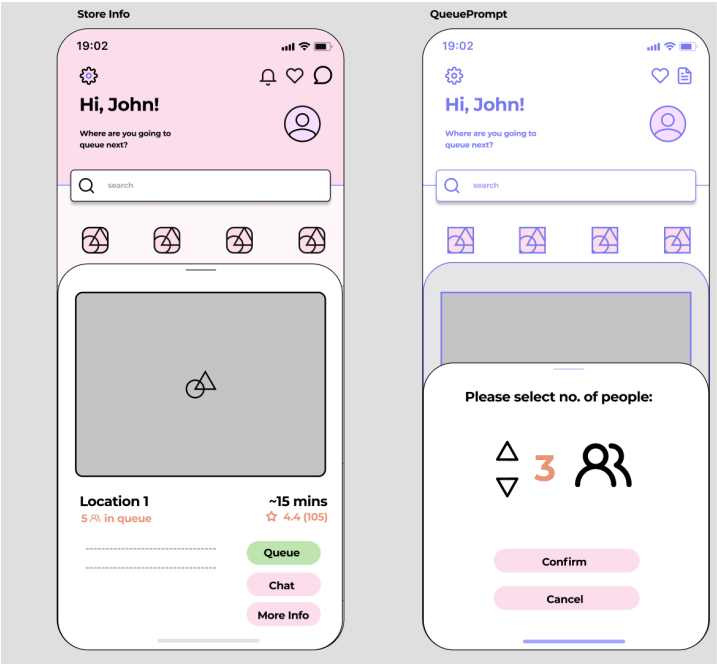


Figure 3: Selecting a store & Joining the Queue

Figure 3 shows the user interface for selecting a store to view the information and then joining a queue. Only crucial information about the store is displayed such as Name, Location, Wait time, Queue and Chat. Users have the option to view a store's detailed information if required.

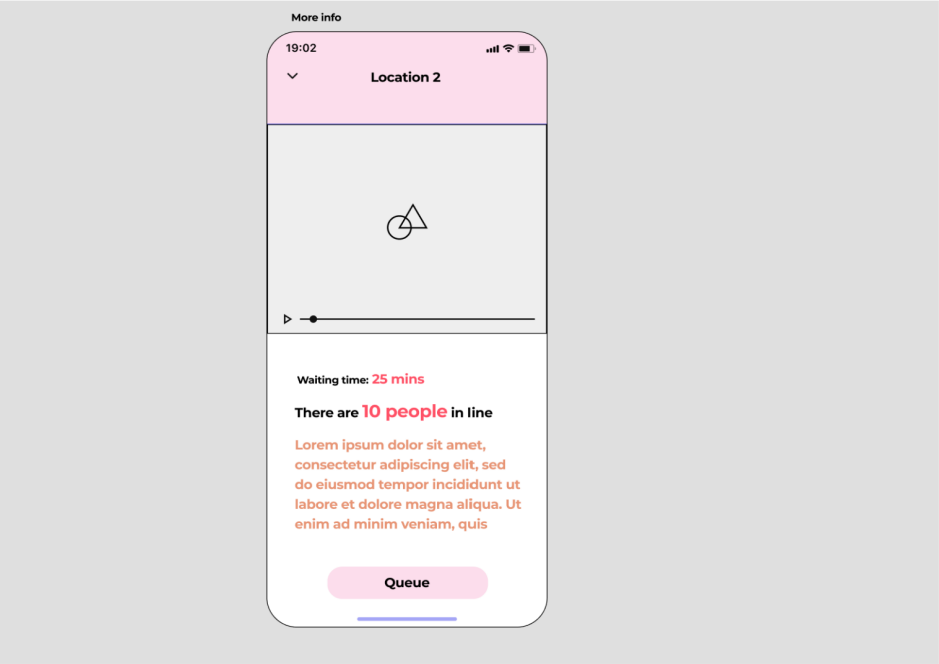


Figure 4: More Information

Figure 4 shows the information about that particular store in detail, including the wait time and the exact number of people queuing via our application.

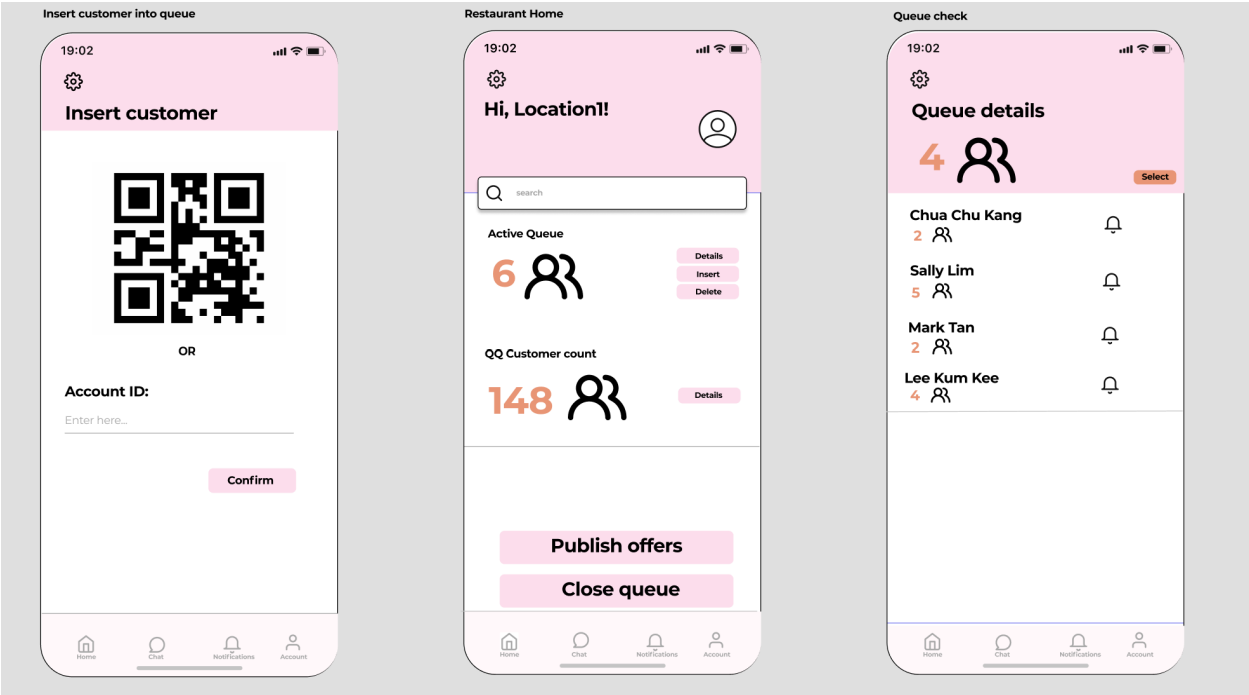


Figure 5: Service-Provider Screens

Figure 5 showcases the Service-Provider user interface. The screen shows the Service-Provider is able to manually insert users into their queue, view the number of people in their queue, and signal the next person in the queue that their turn is soon.

3.2 Hardware Interfaces

To run on this system

- An android version is needed with a minimum of Android 8.0 Oreo installed

To connect to the network

- Wireless Network Interface Card or a modern chip with cellular modem is needed

For Storage:

- The phone must have enough storage space for the system to download the application

3.3 Software Interfaces

The standard development kit version used is 33. The minimum version needed is 27 (Oreo).

The software does not require much space for storage as it is hosted on the cloud and data is stored on MongoDB.

The software requires google services to be enabled as it provides the option for Google sign-in/registration.

The software also requires the Axios library package for HTTP connections to gather data from the internet. This includes pulling information from the BestTimeAPI.

3.4 Communication Interfaces

QQueue requires an internet connection to pull information from BestTimeAPI, MongoDB, SocketIO chat and to retrieve the recommendation system stores. The HTTP connection is used to retrieve data from the applications mentioned above.

4. Functional Requirements

4.1 Registration

4.1.1. The end-user and Service-Provider must both have their own account creation process.

4.1.1.1. The end-user and Service-Provider must be able to sign up via Google.

4.1.2. The system must be able to validate all fields that have been filled up.

4.1.2.1. The system must validate that the password and confirm password fields are filled up.

4.1.2.2. The system must validate that the user's username fields are filled up.

4.1.2.3. The system must validate that the user's email fields are filled up.

4.1.2.4. If validation fails, the corresponding error must be displayed for the missing field.

4.1.3. The system must be able to check the validity of the fields.

4.1.3.1. The system must validate that the password and confirm password are the same.

4.1.3.2. The system must validate that the user's username fields do not exist.

4.1.3.3. The system must validate that the user's email fields do not exist.

4.1.3.4. The system must validate the verification code matches.

4.1.3.5. If validation fails, the corresponding error must be displayed for the missing field.

4.1.4. The users should be able to access the registration page from the login page.

4.2 Login

4.2.1. The system must provide a login entry for all users.

4.2.1.1. The system must provide an email address field for the user.

4.2.1.2. The system must provide a password field for the user.

4.2.2. The system must be able to validate the credentials entered by the users.

4.2.2.1. The system must validate that the email address field is filled.

4.2.2.2. The system must validate that the password field is filled.

4.2.2.3. If any of the fields are not filled, a corresponding error message will be displayed for the empty field.

4.2.5. A successful login shall redirect the different users to their respective homepages.

4.3 Home/Main Page (End-User)

4.3.1. An End-User must be able to view the wait/queue times for shops, eateries, and attractions.

4.3.1.1. The system shall show the wait times of establishments listed.

4.3.1.2. An End-User should be able to filter/view the wait times by category.

4.3.1.3. An End-User should be able to do a quick search for a particular store.

4.3.1.4. An End-User should be able to chat with various users in that store

4.3.2. An End-User must be able to queue for an establishment using the app.

4.3.2.1. An End-User must link their SingPass to join the queue (Theoretical).

4.3.2.2. An End-User must be able to obtain a queue number via the app.

4.3.2.3. Prompts/notifications must be shown when approaching one's turn/missed queue number (notification system).

4.3.2.4. The system must provide an option for the user to rejoin the queue in the event of a missed queue number.

4.3.2.5. The End-User should be able to checkout of the store.

4.3.2.6. The End-User should be able to leave the queue in the system at any point of time.

4.3.3. The system should be able to store the End-Users favourites section.

4.3.3.1. The End-User should be able to add stores to their favourites list.

4.3.3.2 The End User should be able to view their favourites list

4.3.4. The system should provide a recommended list of stores to the users.

4.4 Home/Main Page (Service-Provider)

4.4.1. A Service-Provider must be able to view and manage its own service queue.

4.4.1.1. A service queue should be set up for each provider.

4.4.1.2. The service queue does not require any inputs from the provider.

4.4.1.3. The Service-Provider must be able to manually insert and remove users in the queue.

4.4.1.4. The Service-Provider must be able to notify End-Users in the queue.

4.4.1.5 Service-Provider home screen should display summary information on the store's queue

4.4.1.6 Service-Provider should be able to get detailed information on the queue

4.5 Feedback

4.5.1. An End-User must be able to report issues and submit feedback.

4.5.2. End-User should be able to provide feedback on their experience regarding app usage

4.6 Rewards

4.6.1. End-User can earn points only if they check out of the store.

4.6.2. The system must allow users to view the points they have.

4.6.3. The system must allow users to view claimable rewards.

4.6.3.1. The system should allow users to exchange points for rewards only if they have enough points.

5. System Features

This section is organised by the **priority of use cases** (From High to Low)

Each system feature is mapped to the respective functional requirement in ***Section 4***

5.0.1 Initial Use Case Diagram

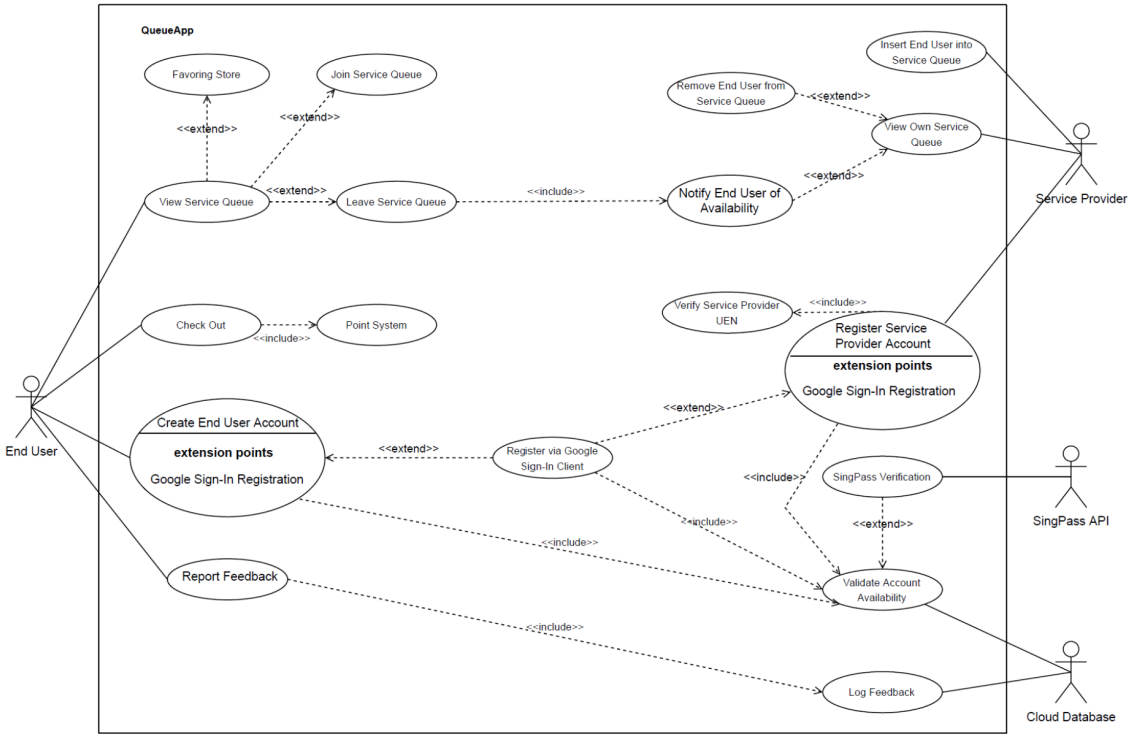


Figure 6: Initial Use Case Diagram

5.1 Register End-User account

5.1.1 Description and Priority

When the user first runs the application, the system will enter the onboarding screen where the user can register/login to an account. The end-user may then register for a new account assuming they do not already have an existing account in the system.

Use Case ID: 10 (Appendix 13.2.10)

Priority: High
This information is required to enable the user to enter the application to utilise its features

5.1.2 Stimulus/Response Sequences

User Action	System Response
Enter their personal information (Name, Password, Confirm Password, and Email)	Verify the input fields are valid and create the Schema JSON object to be saved in MongoDB
Click on register button	Display the next screen a validation screen with the verification code being emailed to the user
Enter their verification code	Authenticate the generated ID for the user is correct
Click of verify button	Display the next screen, Home page with the new account information
Alternative Flow: Google Sign-In Client Registration	
Select Google Sign-in	System will prompt the user to register via their google account and then display the next screen, Home page with the new account information

5.1.3 Functional Requirements

4.1.1. The end-user must be able to create an account

4.1.1.1 The end-user must be able to sign up for his account via Google

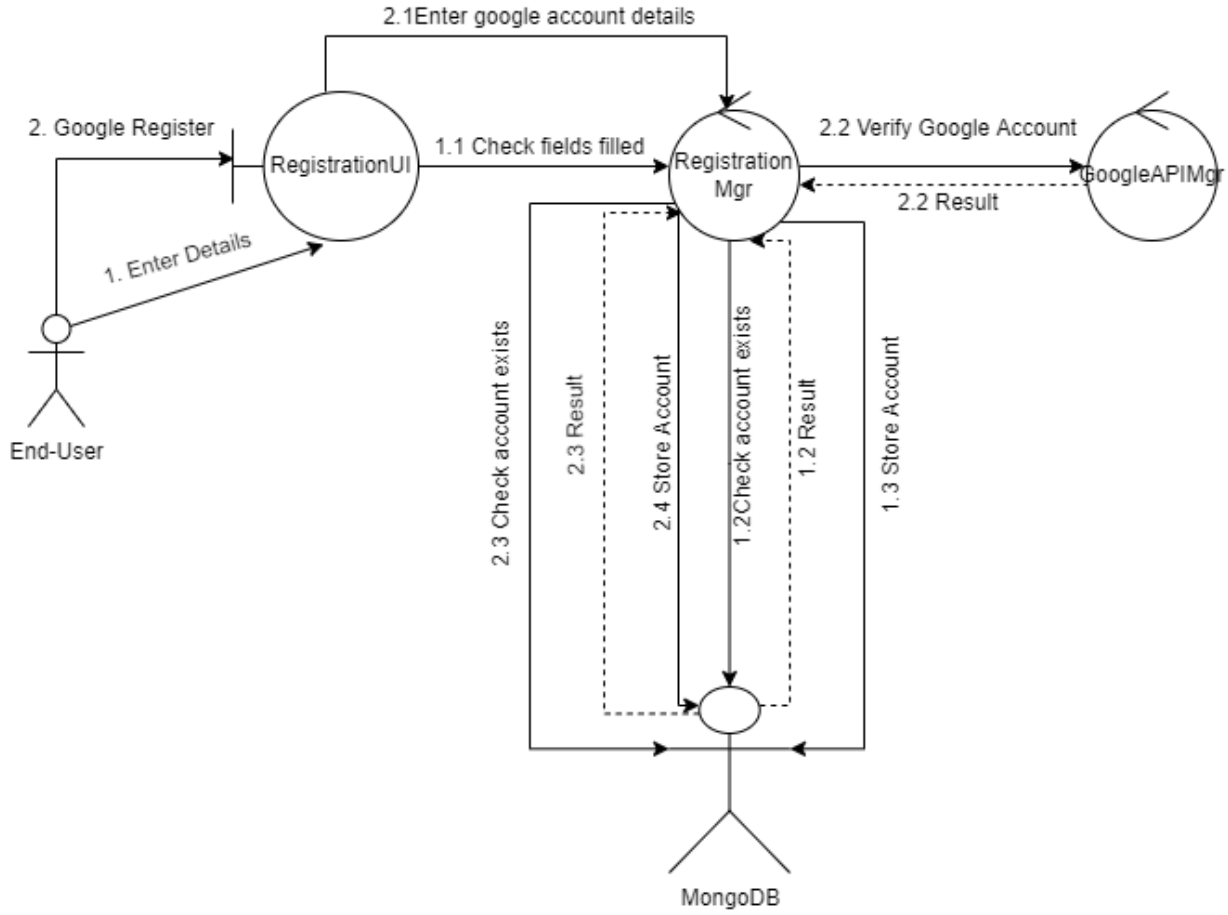
4.1.3.

4.1.3.2. The system must indicate and reject the registration if the account already exists

4.1.3.3. The system must reject the registration if the password is different from the confirmation password

4.1.3.4. The system must prompt the user for the verification code and reject the code if it is entered wrongly

5.1.4 Conceptual Model



5.1.5 Sequence Diagram

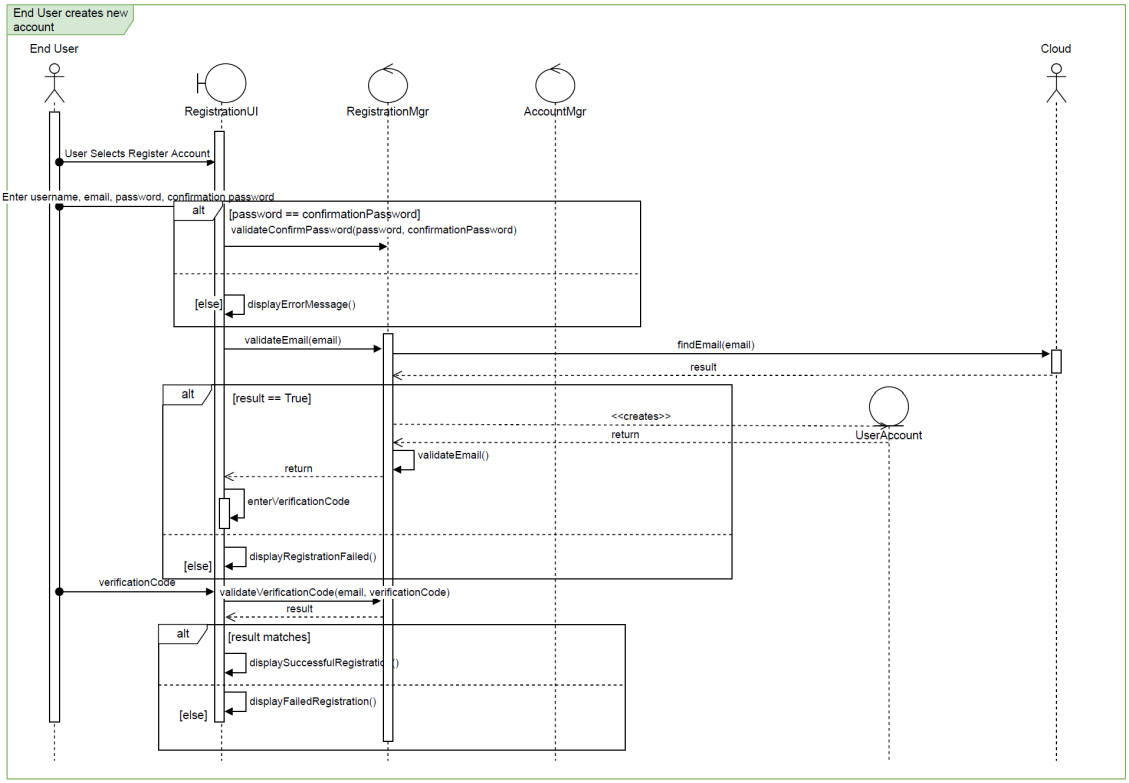


Diagram 5.1.1: Use Case 10 Sequence Diagram

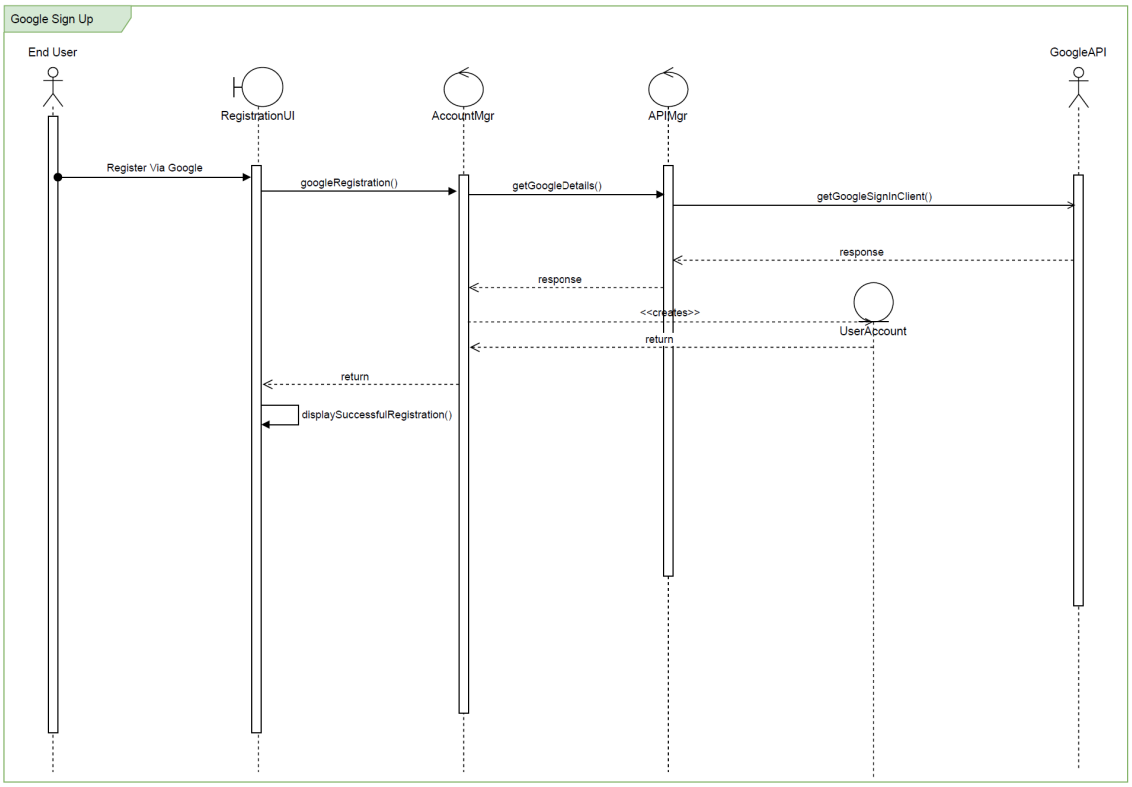


Diagram 5.1.2: Use Case 10 (Alternative Flow) Sequence Diagram

5.2 End-User View Service Queue

5.2.1 Description and Priority

The end-user is allowed to view a list of various stores and their information/queue time. stores nearer to the user's location should be displayed to them and they should be able to search the stores by name

Use Case ID: 3 (Appendix 13.2.3)

Priority: High
End-User is required to click on a particular store icon to see the information of the stores. They can then subsequently click to view more information about the store if required.

5.2.2 Stimulus/Response Sequences

User Action	System Response
User can swipe left or right on the home screen	Returns the list of stores that are currently available to be viewed by the user
User clicks on a store	Returns the information about the store including a description, queue time as well as the availability to join the chat
Alternative Flow: Using chat to get information	
User can swipe left or right on the home screen	Returns the list of stores that are currently available to be viewed by the user
User clicks on a store	Returns the information about the store including a description, queue time as well as the availability to join the chat
User clicks on chat	End-User account connects to the chat room of that particular store. Service-Providers and other End-Users can provide information about the store and queue

5.2.3 Functional Requirements

4.3.1. An End-User must be able to view the wait/queue times for shops, eateries and attractions
- 4.3.1.1. The system shall show the wait times of the establishments listed
- 4.3.1.2. An End-User should be able to filter/view the wait times by category
- 4.3.1.3. The End-User should be able to do a quick search for stores

5.2.4 Conceptual Model

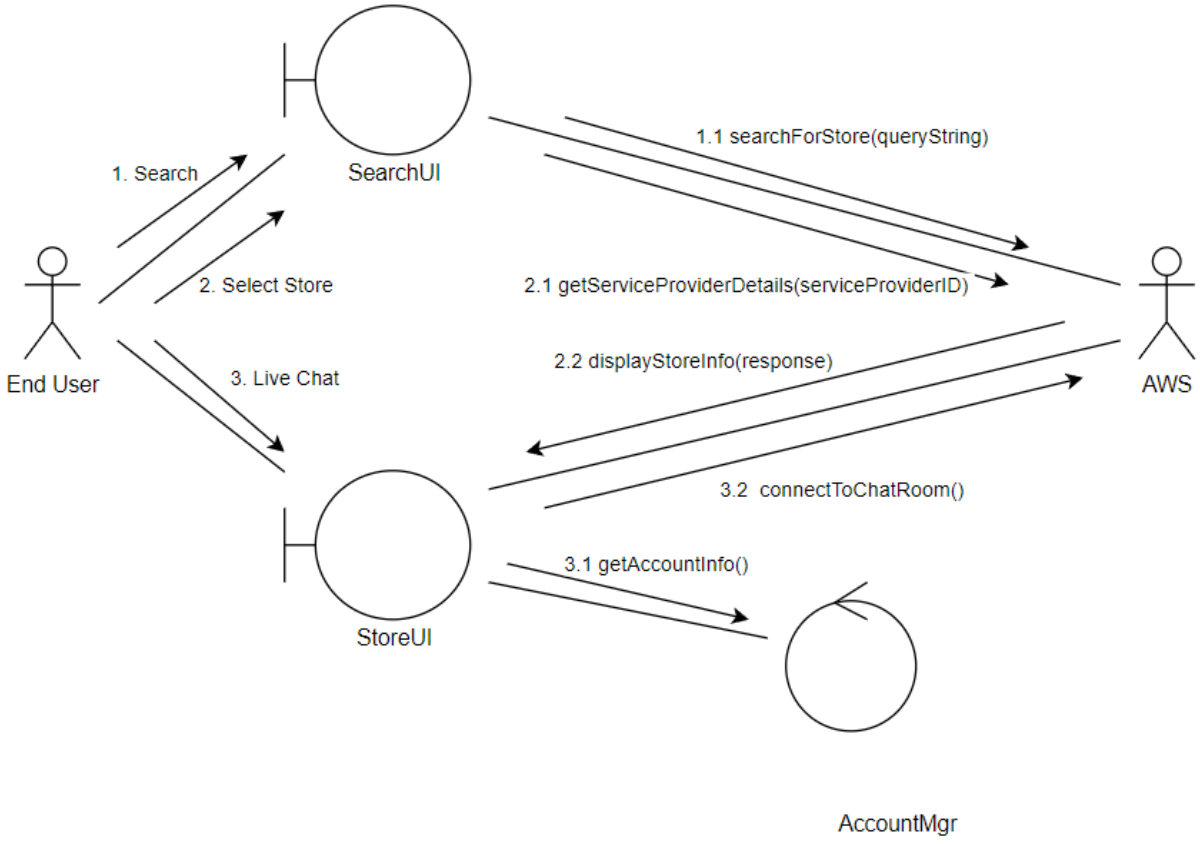


Diagram 5.2.1: Use Case 3 Conceptual Mode

5.2.5 Sequence Diagram

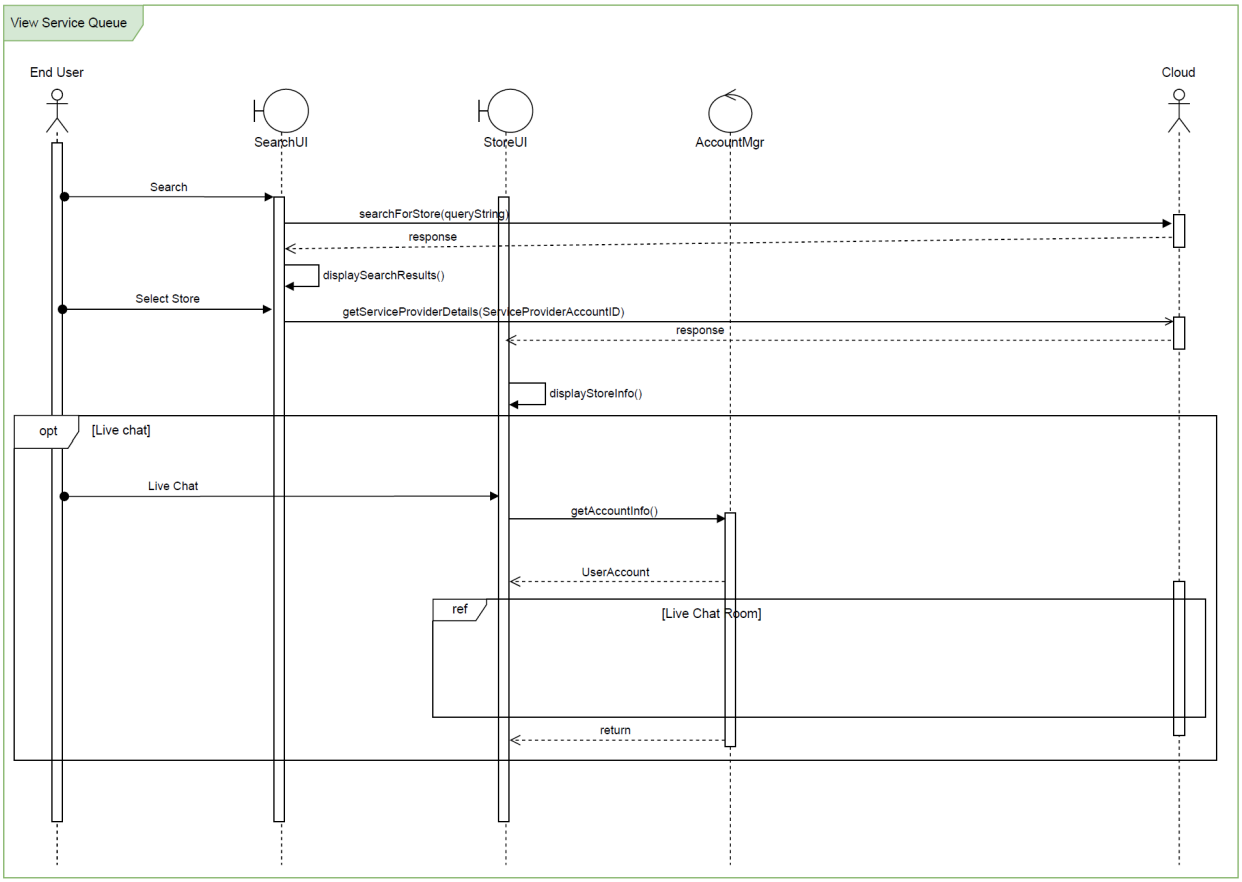


Diagram 5.2.2: Use Case 3 Sequence Diagram

5.3 End-User Joining/Leaving Service Queue

5.3.1 Description and Priority

The end-user should be able to join a service queue of one store at any point of time and to leave it as well.

Use Case ID: 1, 2 (Appendix 13.2.1, 13.2.2)

Priority: Medium
End-User is required to click the Queue button or Leave button depending on the action they want to take

5.3.2 Stimulus/Response Sequences

User Action	System Response

Select Queue Button in a store	Returns a pop-up screen with the pax number that the user is queuing with
Entering Pax number and press the confirm button	Register the user into the database with the pax number in the queue and to update the number of people in the queue
Select Leave Queue button	Remove the user from the queue and in their queue id in MongoDB
Alternative Flow: Service-Provider notifies End-User of their turn	
Service-Provider taps on the details button	System will change screen to the Queue Details screen
Service-Provider taps on the bell icon	Searches for the account name in the Service Queue and send a notification to the user followed by removing user from the mongoDB queue database

5.3.3 Functional Requirements

4.3.1 An End-User must be able to queue for an establishment using the app

4.3.2.1 A End-User must have their account signed up with the app

4.3.2.2 A End-User must get their queue position via the app

4.3.2.3 Prompts/Notifications must be shown when approaching one's turn/missed queue number via the notification system

4.3.2.4 The system must provide an option for the user to rejoin the queue in the event of a missed number

4.3.2.6 A End-User can leave the queue in the system at any point of time

5.3.4 Conceptual Model

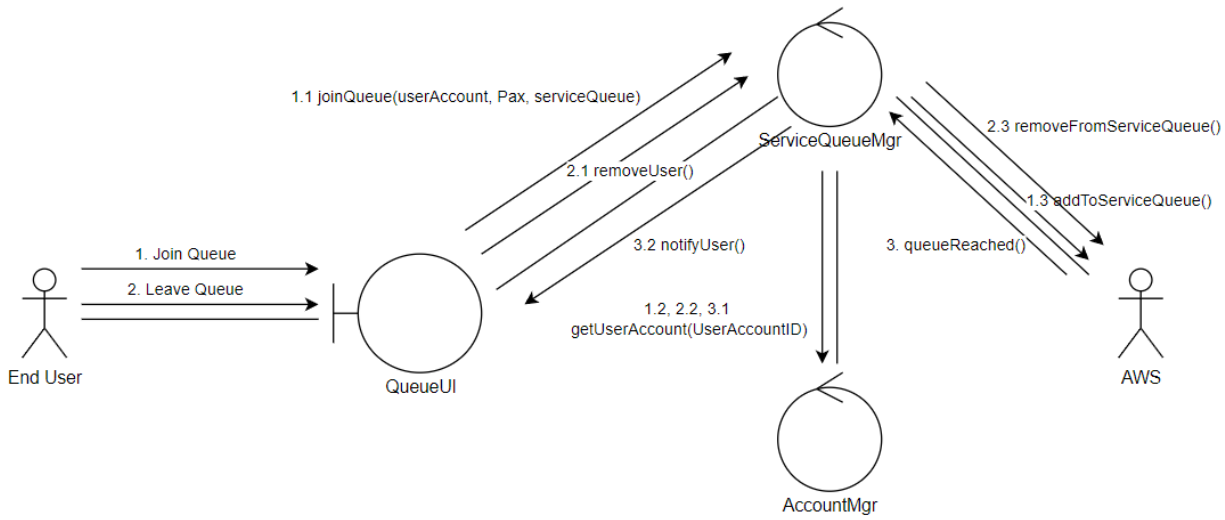


Diagram 5.3.1: Use Case 1 & 2 Conceptual Model

5.3.5 Sequence Diagram

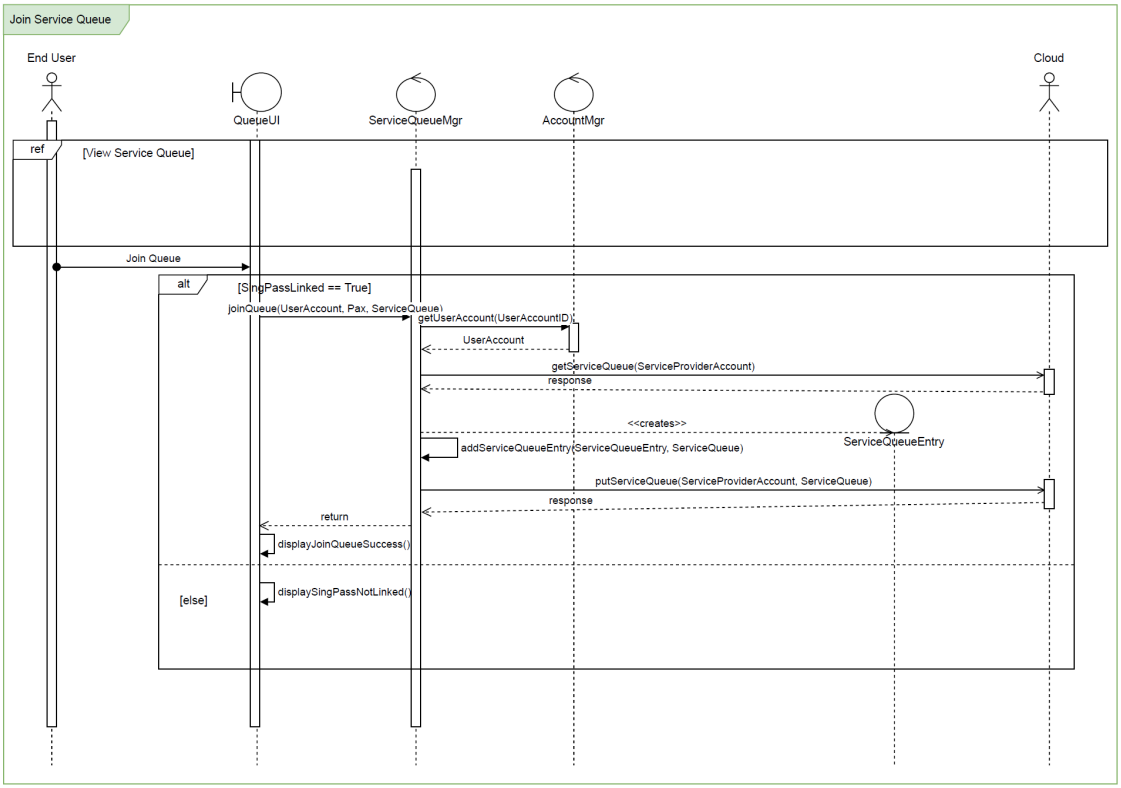


Diagram 5.3.2: Use Case 1 Sequence Diagram

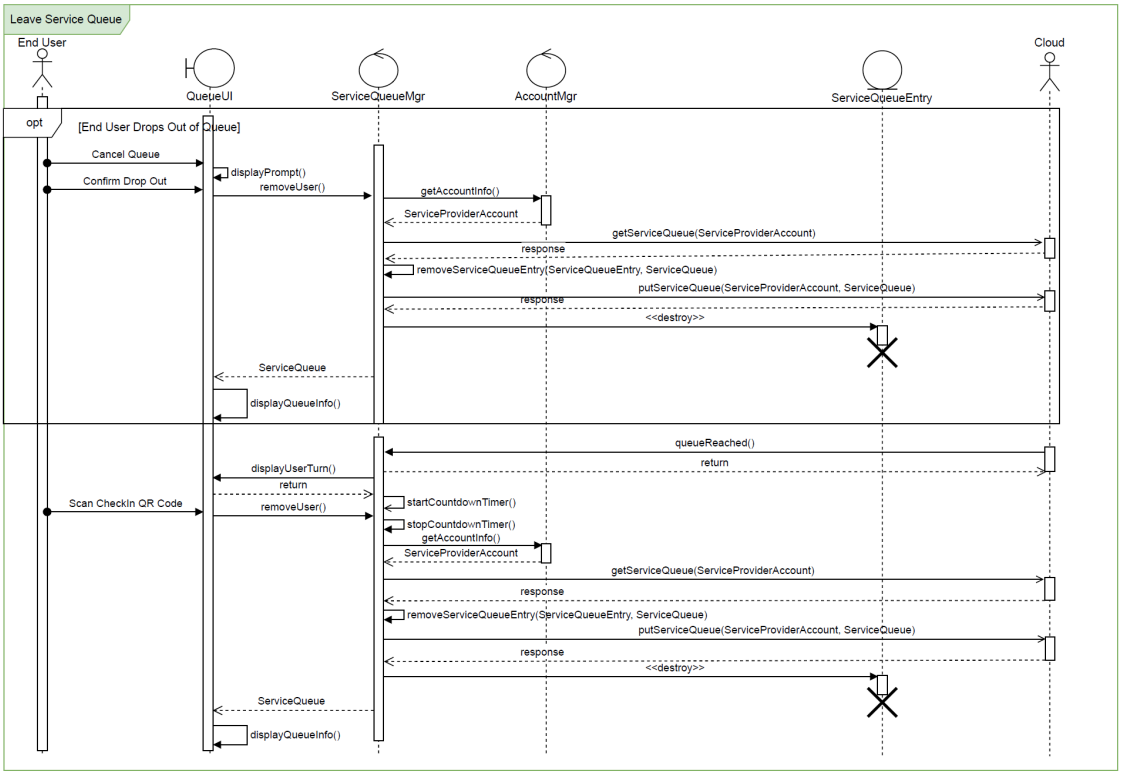


Diagram 5.3.3: Use Case 2 Sequence Diagram

5.4 View Service Queue

5.4.1 Description and Priority

Allow the Service-Provider to view their own Service Queue

Use Case ID: 5 (Appendix 13.2.5)
Priority: Medium

Service-Provider Home Screen should display a summary of his queue. He can get an in-depth view of the queue by tapping on details

5.4.2 Stimulus/Response Sequences

User Action	System Response
Service-Provider first enters home screen	System will pull current queue from MongoDB and make use of the length of the table as the length of the queue

Service-Provider taps on “Details”	System displays the individual accounts that are part of the queue, including the number of pax for each account

5.4.3 Functional Requirements

4.4.1 Service-Provider must be able to see details of his store’s queue

4.4.1.5 Service-Provider home screen should display summary information on the store’s queue

4.4.1.6 Service-Provider should be able to get detailed information on the queue

5.4.4 Conceptual Model

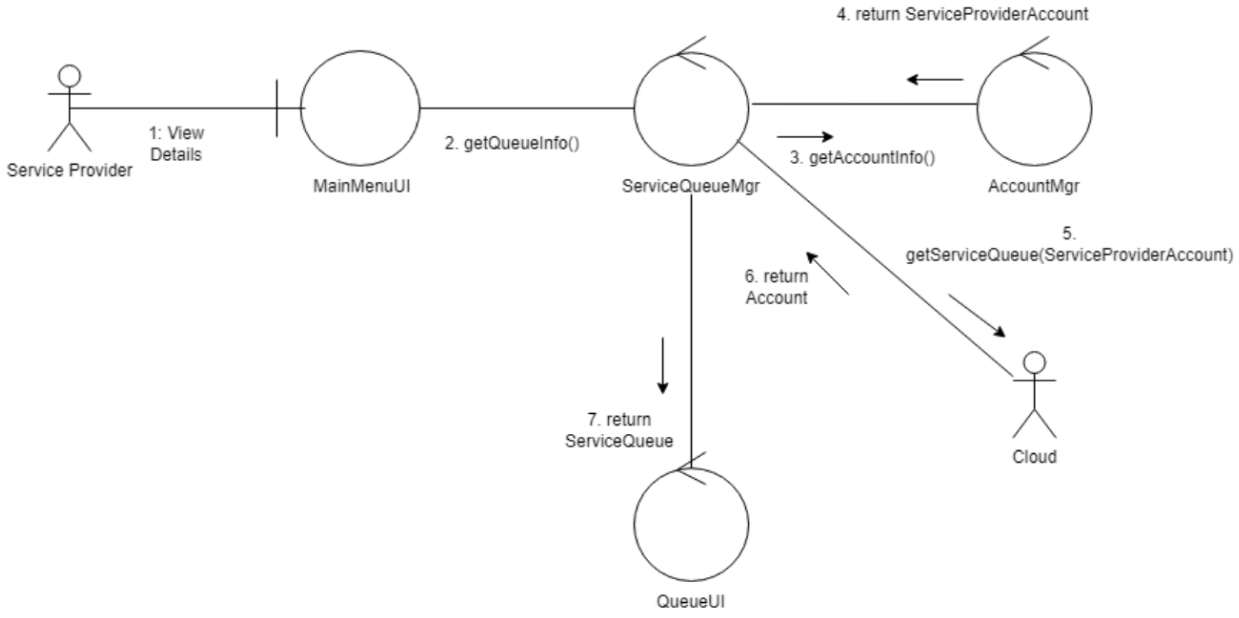


Diagram 5.4.1: Use Case 5 Conceptual Diagram

5.4.5 Sequence Diagram

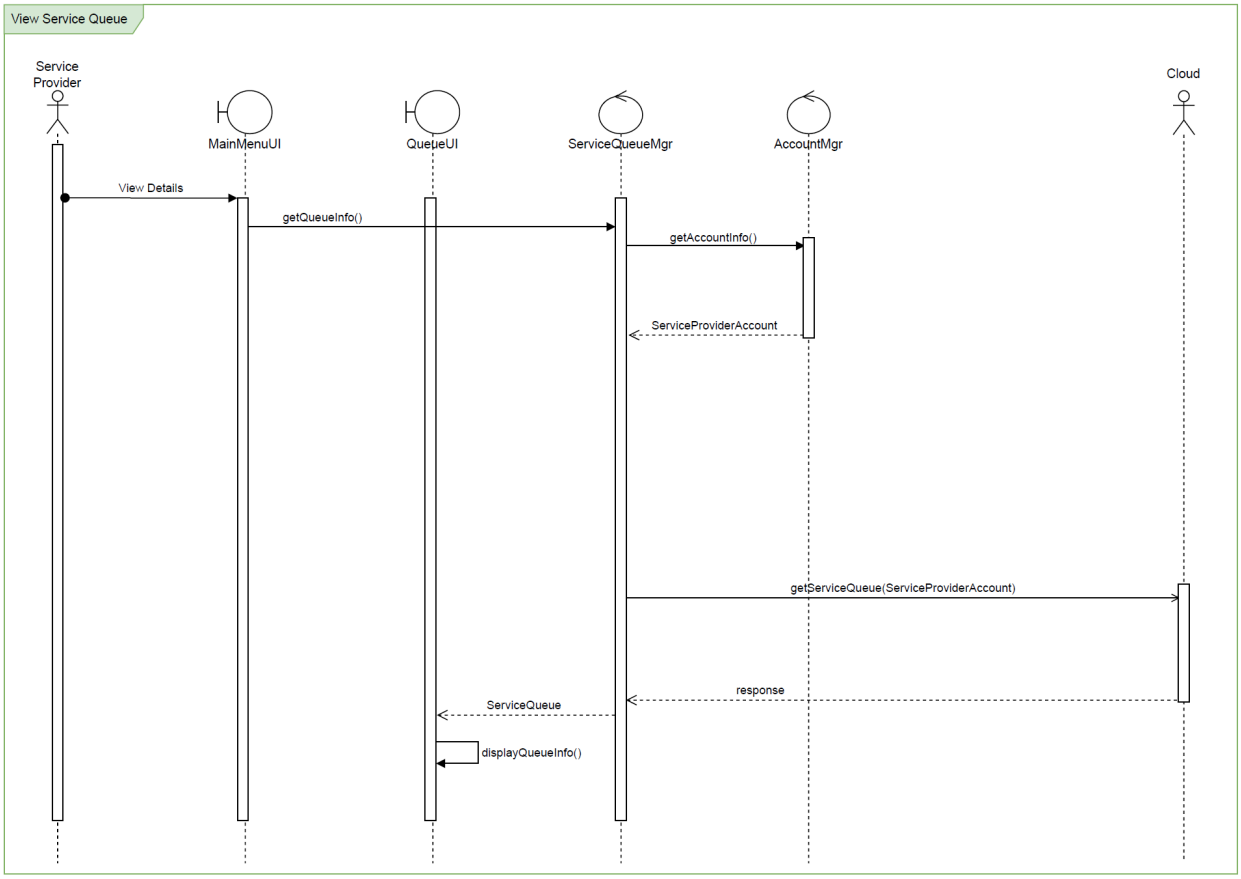


Diagram 5.4.2: Use Case 5 Sequence Diagram

5.5 Register Service-Provider Account

5.5.1 Description and Priority

When the Service-Provider first runs the application, the system will enter the onboarding screen where the Service-Provider can select the Business option. After that the Service-Provider will be prompted either to login or register. By selecting the register option, the Service-Provider will be directed to the registration page

Use Case ID: 4 (Appendix 13.2.4)

Priority: High
This information is required to enable the Service-Provider to enter the application to utilise its features

5.5.2 Stimulus/Response Sequences

User Action	System Response
Enter their personal information (Name, Password, Confirm Password, and Email)	Verify the input fields are valid and create the Schema JSON object to be saved in MongoDB
Click on register button	Display the next screen a validation screen with the verification code being emailed to the user
Enter their verification code	Authenticate the generated venueID for the user is correct
Click verify button	Display the next screen, Home page with the new account information
Alternative Flow: Google Sign-In Client Registration	
Select Google Sign-in	System will prompt the user to register via their google account and then display the next screen, Home page with the new account information

5.5.3 Functional Requirements

4.1.1. The Service-Provider must be able to create an account

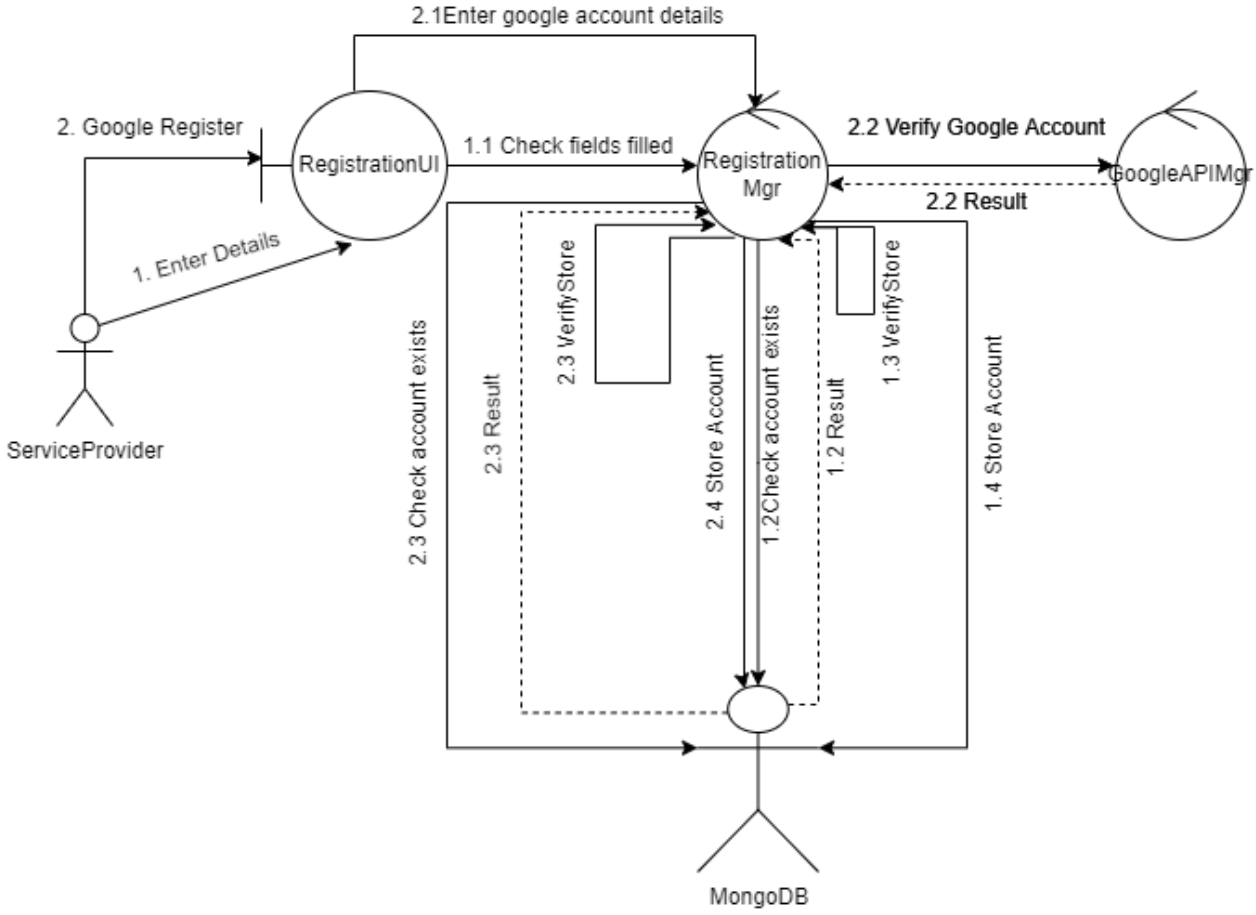
4.1.1.1 The Service-Provider must be able to sign up for his account via Google

4.1.3.2 The system must indicate and reject the registration if the account already exists

4.1.3.3 The system must reject the registration if the password is different from the confirmation password

4.1.3.4 The system must prompt the Service-Provider for the verification code and reject the code if it is entered wrongly

5.5.4 Conceptual Model



5.5.5 Sequence Diagram

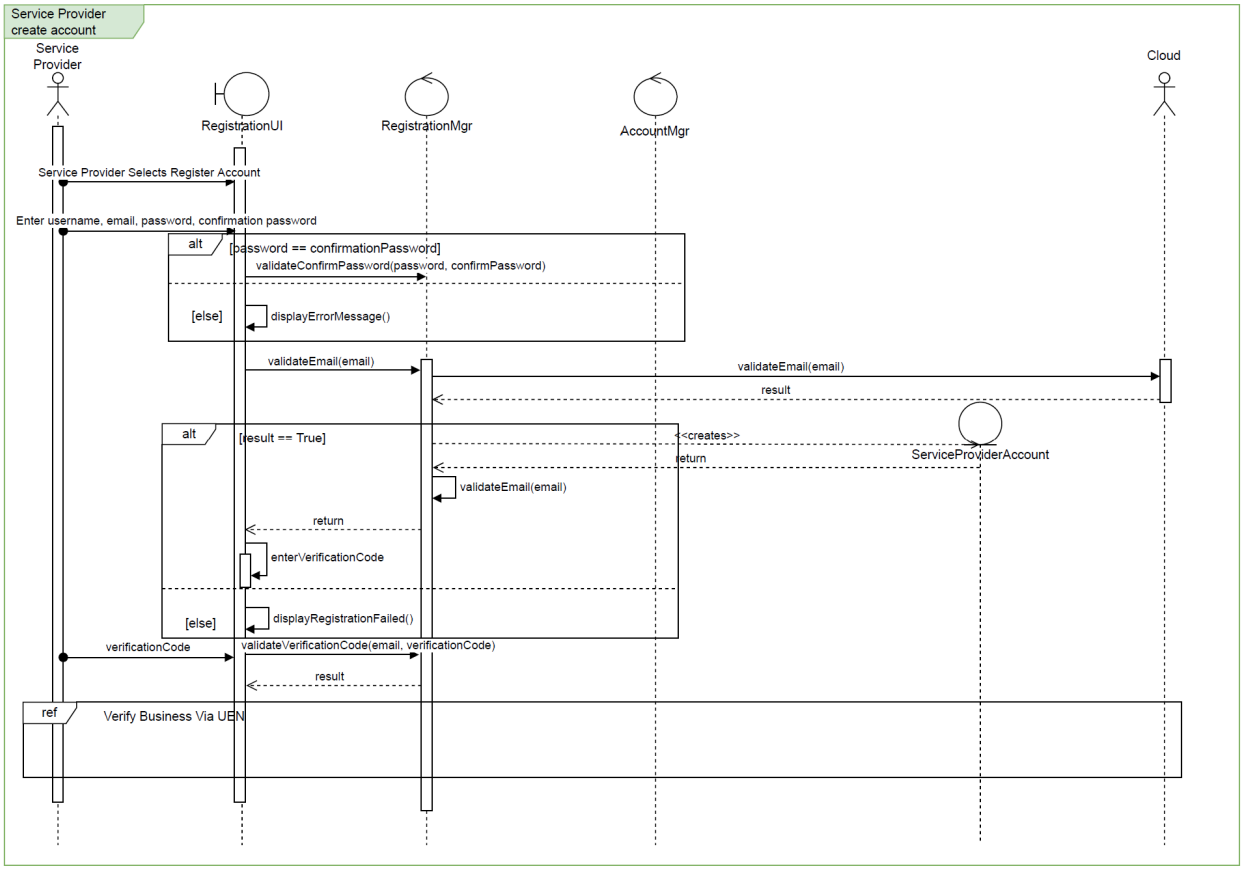


Diagram 5.5.2: Use Case 4 Sequence Diagram

5.6 Remove End-User from Service Queue

5.6.1 Description and Priority

Service-Provider is able to provide a notification to the End-User and remove him from the Queue. Alternatively, the Service-Provider is able to manually delete the End-User from the queue using the delete button.

Use Case ID: 7 (Appendix 13.2.7)
Priority: Medium

The Service-Provider will tap on the bell icon to notify the End-User that it is their turn. End-User will receive a notification on their device to inform them it is their tur

5.6.2 Stimulus/Response Sequences

User Action	System Response

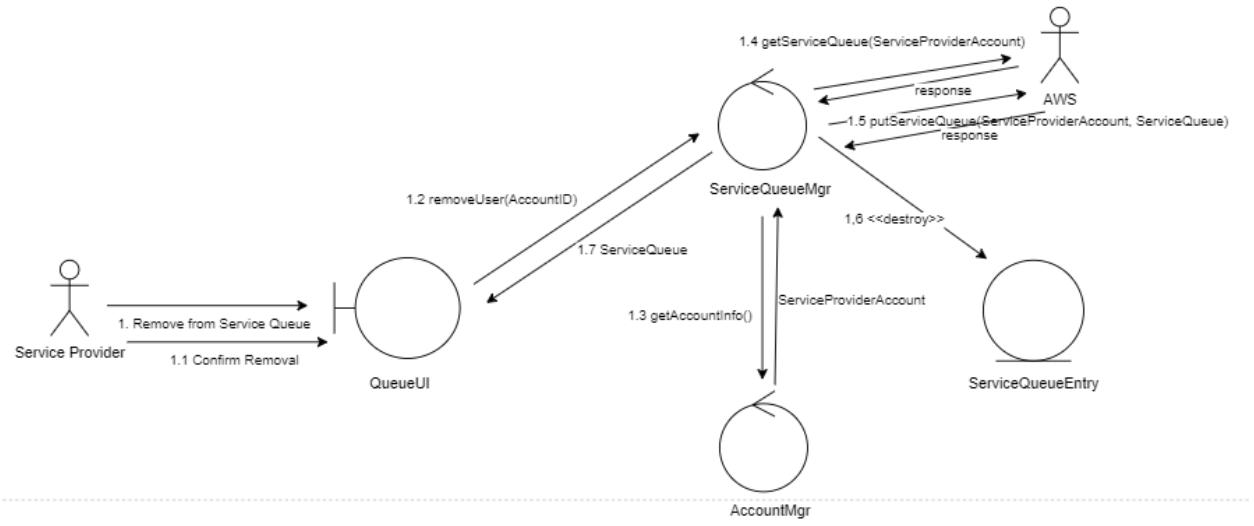
Service-Provider taps on the “Delete” button	System will change screen to the Queue Details screen
Service-Provider taps on the “Trash Can” icon beside the user they want to delete	Searches for the account name in the Service Queue and removes it from the mongoDB queue database. S
Service-Provider selects confirm to the delete request	

5.6.3 Functional Requirements

4.4.1.

4.4.1.3. Service-Provider must be able to remove users from their Queue

5.6.4 Conceptual Model



5.6.5 Sequence Diagram

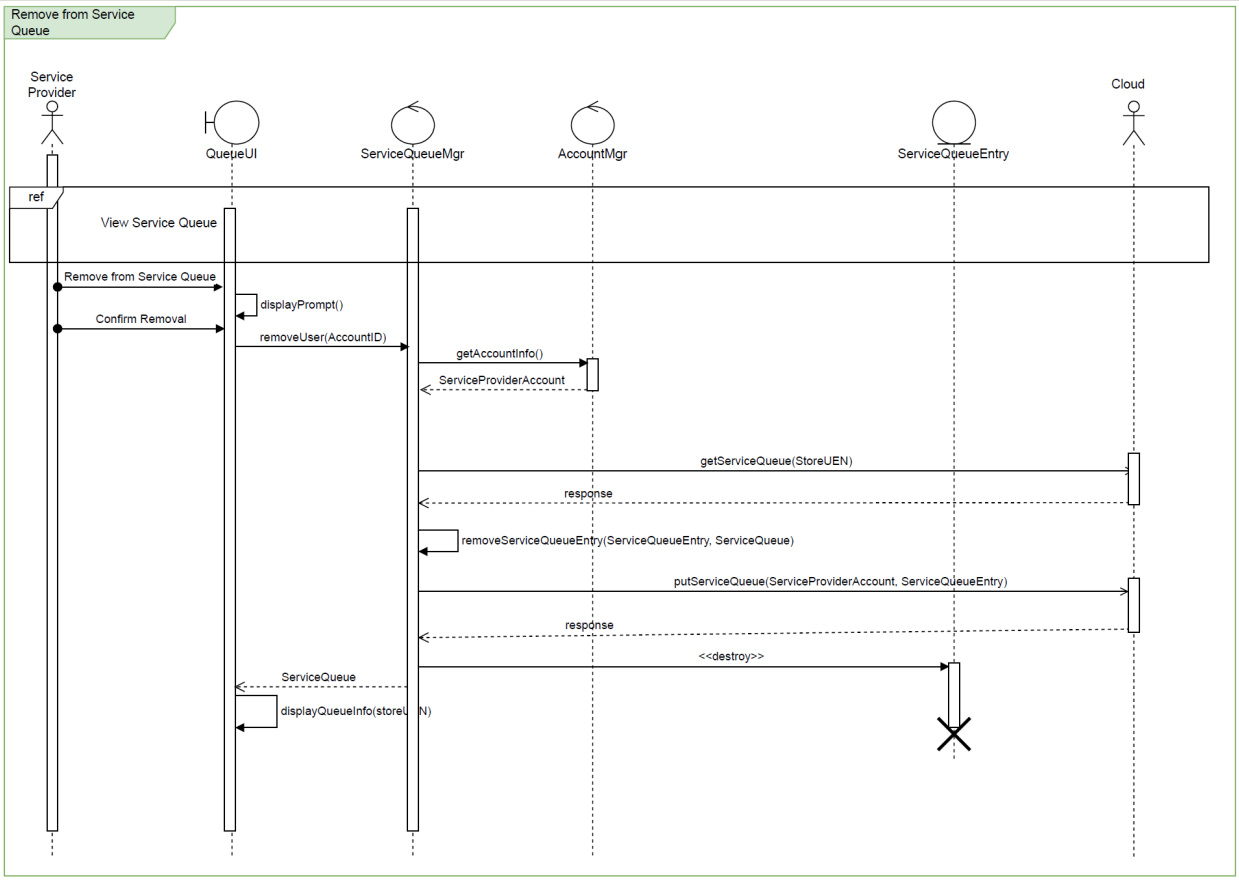


Diagram 5.6.2: Use Case 7 Sequence Diagram

5.7 Insert into ServiceQueue by Service-Provider

5.7.1 Description and Priority

Service-Provider manually inserts an End-User into Service Queue

Use Case ID: 6 (Appendix 13.2.6)
Priority: Medium

Service-Provider is able to tap on the insert button under “Active Queue” and enter user’s username to insert them into their ServiceQueue

5.7.2 Stimulus/Response Sequences

User Action	System Response
Service-Provider taps on insert button	System will change screen to a page for Service-Provider to input End-User username

Service-Provider enters End-User username and press confirm	System will add End-User' accounts to the end of the Service-Provider's ServiceQueue

5.7.3 Functional Requirements

4.4.1.3 Service-Provider should be able to manually insert user into the ServiceQueue given the End-User's username

5.7.4 Conceptual Model

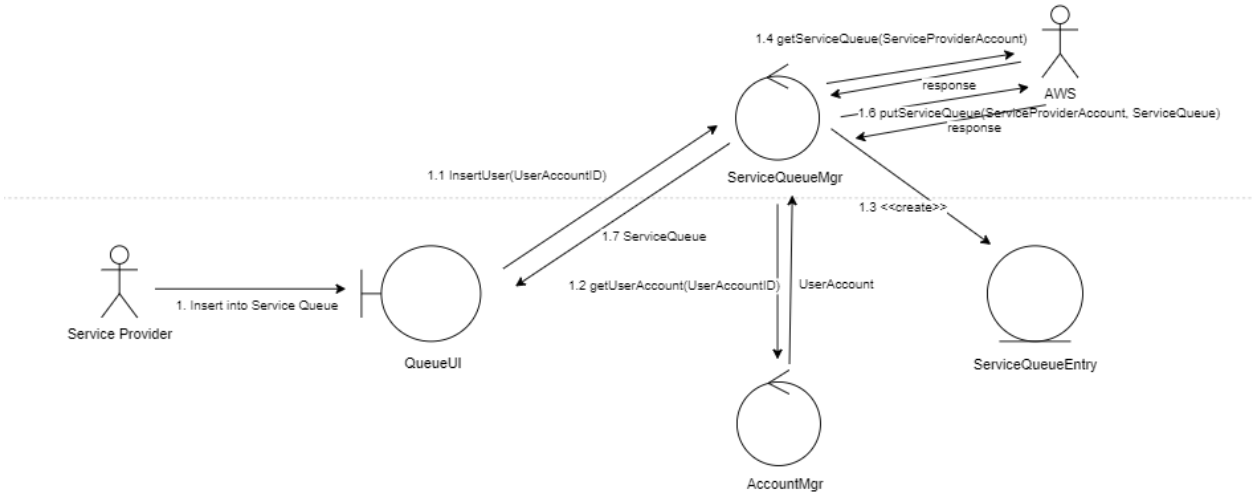


Diagram 5.7.1: Use Case 8 Conceptual Diagram

5.7.5 Sequence Diagram

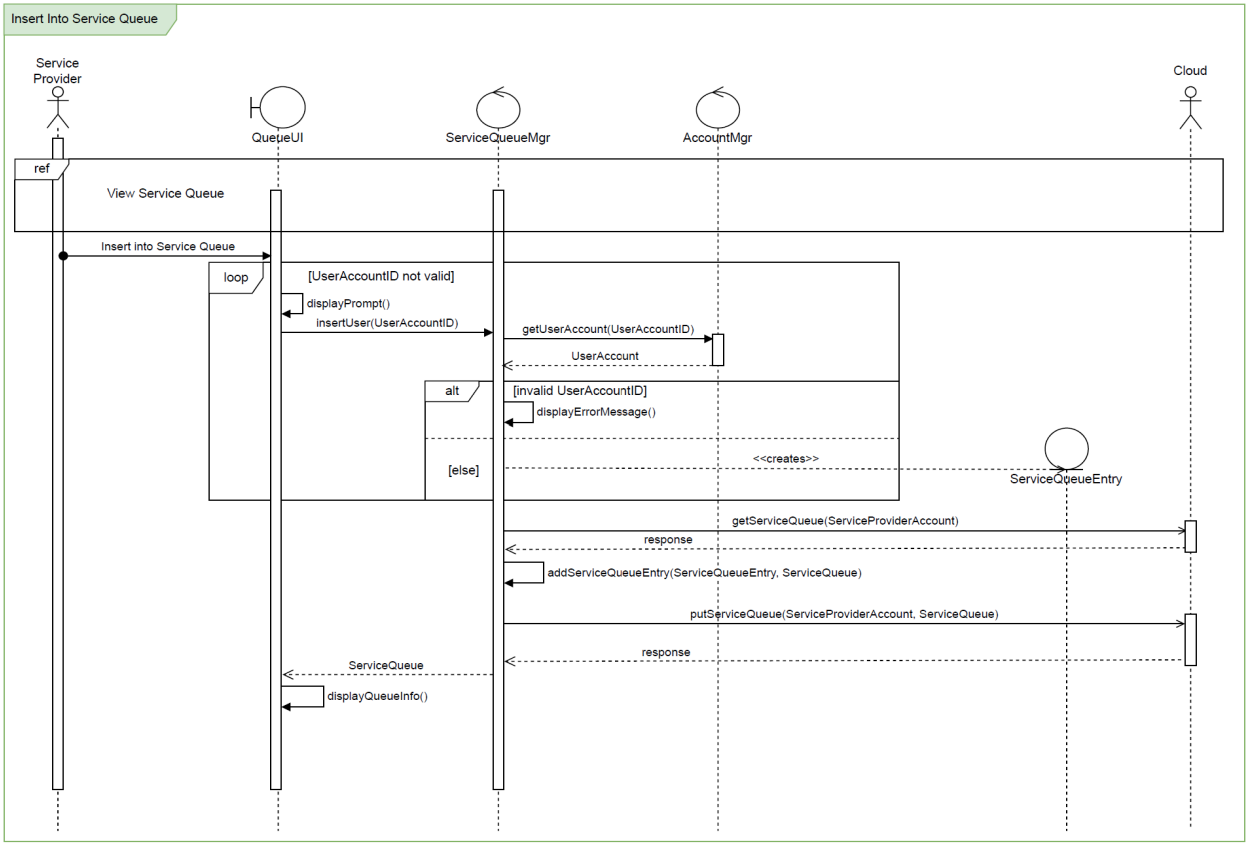


Diagram 5.7.2: Use Case 6 Sequence Diagram

5.8 End-User Favouring Store & Viewing Favourites

5.8.1 Description and Priority

End-User saving stores on his favourites to view

Use Case ID: 9 (Appendix 13.2.9)
Priority: Low

End-User can tap on the “heart” icon to view their favourites list. End-User can also tap on the “star” icon on stores to add that store to their favourite list

5.8.2 Stimulus/Response Sequences

User Action	System Response
End User taps on the “heart” icon	System will pull the user’s favourite list from mongoDB and display each store

Service-Provider taps on the “star” icon	System will add that store to the user’s favourite list in mongoDB

5.8.3 Functional Requirements

4.3.3.

4.3.3.1 End User should be able to add stores to their favourites list

4.3.3.2 End User should be able to view their favourites list

5.8.4 Conceptual Model

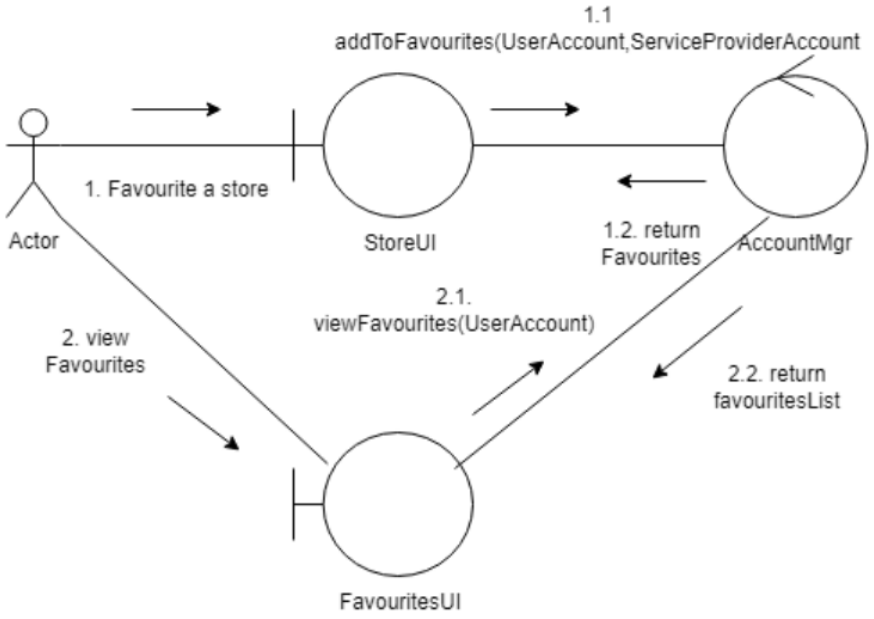


Diagram 5.8.1: Use Case 9 Conceptual Diagram

5.8.5 Sequence Diagram

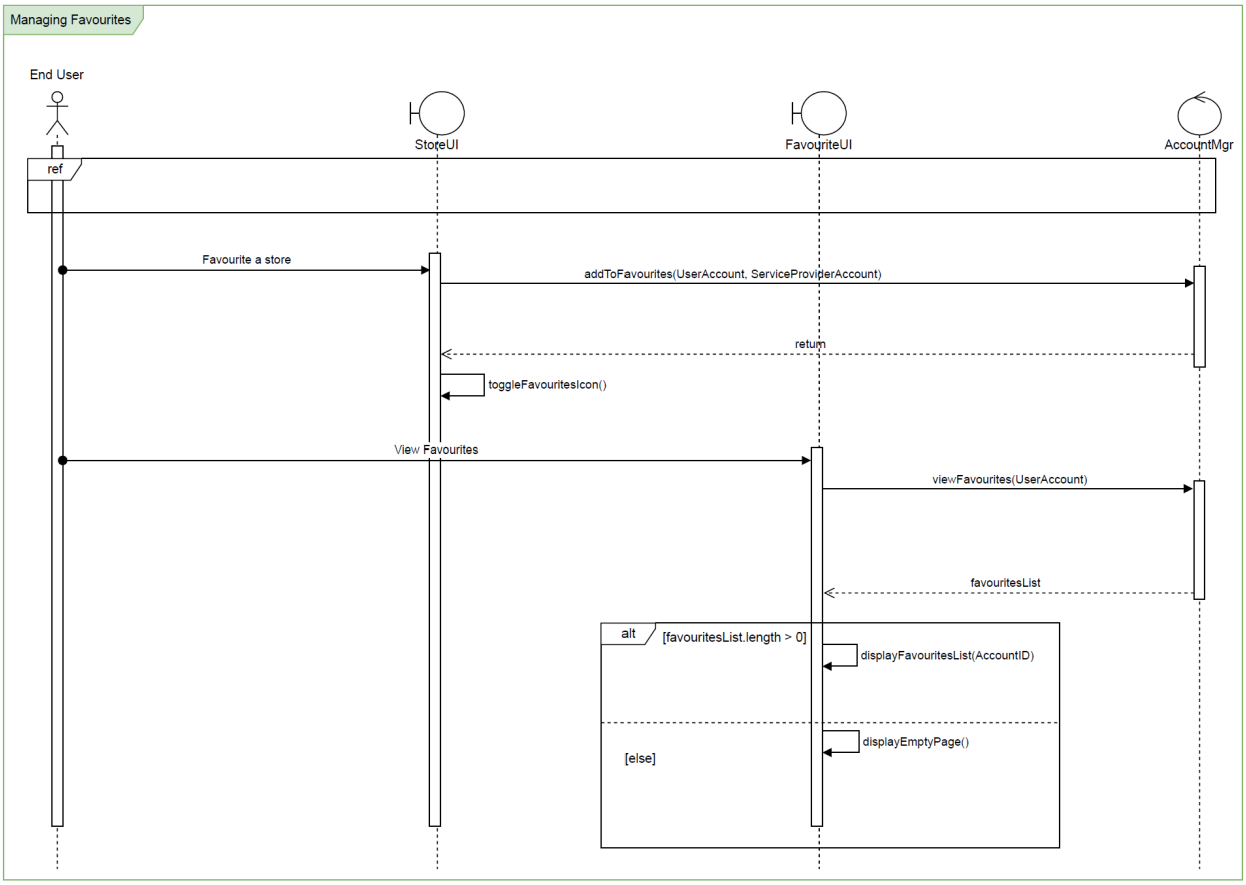


Diagram 5.8.2: Use Case 9 Sequence Diagram

5.9 End-User Providing Feedback

5.9.1 Description and Priority

End-User submit feedback or reports an issue using the application

Use Case ID: 8 (Appendix 13.2.8)
Priority: Low

End-User can tap on the gear icon to go to the settings pages. Next user can tap on the report button to reach the feedback page.

5.9.2 Stimulus/Response Sequences

User Action	System Response
User taps on “gear” icon to go to the settings page	System displays the setting page

User taps on Feedback option	System displays the Feedback page
User provides Feedback in feedback box and press submit	System will store the feedback to the MongoDB database

5.9.3 Functional Requirements

4.5.2. End-User should be able to provide feedback on their experience regarding app usage

5.9.4 Conceptual Model

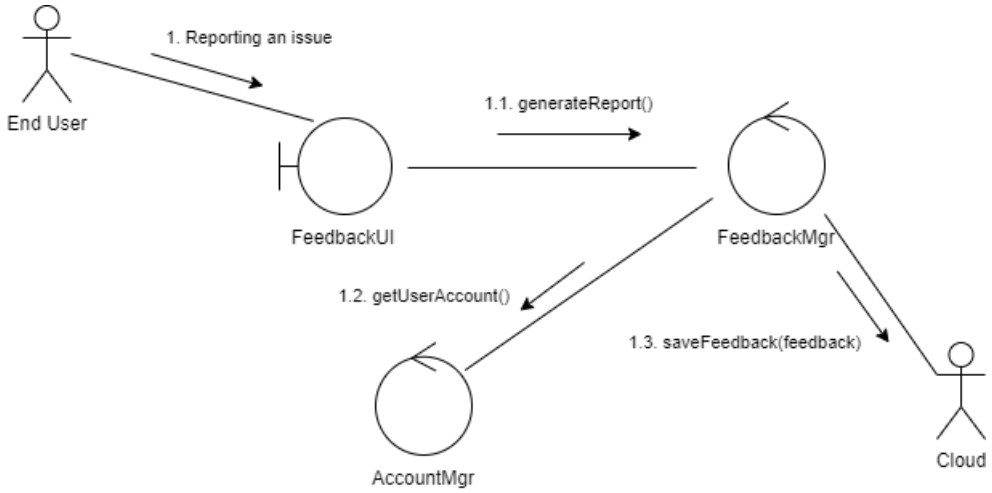


Diagram 5.9.1 : Use Case 8 Conceptual Diagram

5.9.5 Sequence Diagram

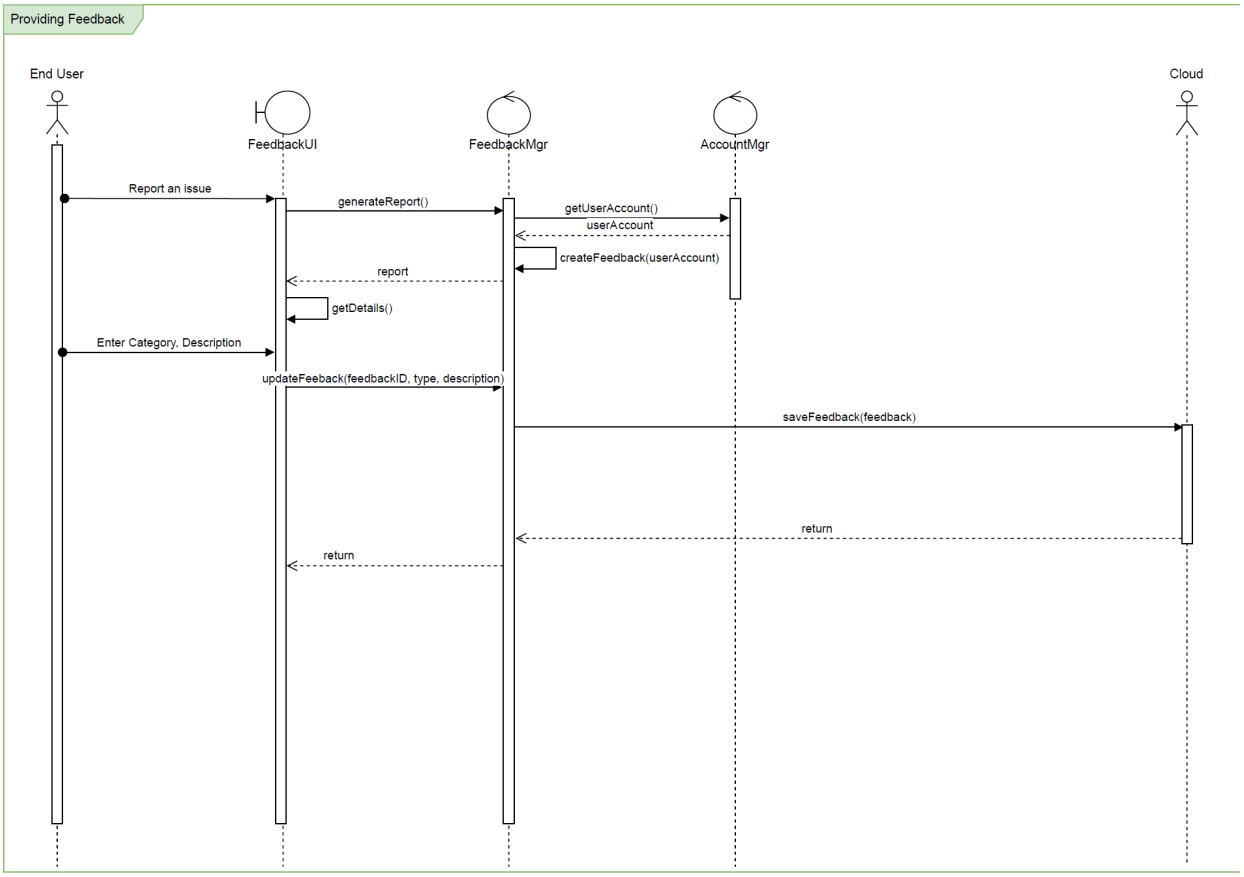


Diagram 5.9.2: Use Case 8 Sequence Diagram

5.10 End-User Check out of the store

5.10.1 Description and Priority

Process through which the End-User checks out of the F&B Store to earn points

Use Case ID: 11 (Appendix 13.2.11)

Priority: Low
End-User should be able to scan a QR code unique to a store to indicate they have left the store. Once the End-User checks out, he/she will get points that could be used for rewards within the app.

5.10.2 Stimulus/Response Sequences

User Action	System Response
User scans QR code of specific store	System will update the user's account points, reducing the store's current capacity by the number of pax left.

5.10.3 Functional Requirements

4.3.2.

4.3.2.5. The End-User must be able to check out of the store

4.6.1. The End-User account should be updated with the new amount of points

4.6.2. The Store should reflect the new total pax within the establishment

5.10.4 Conceptual Model

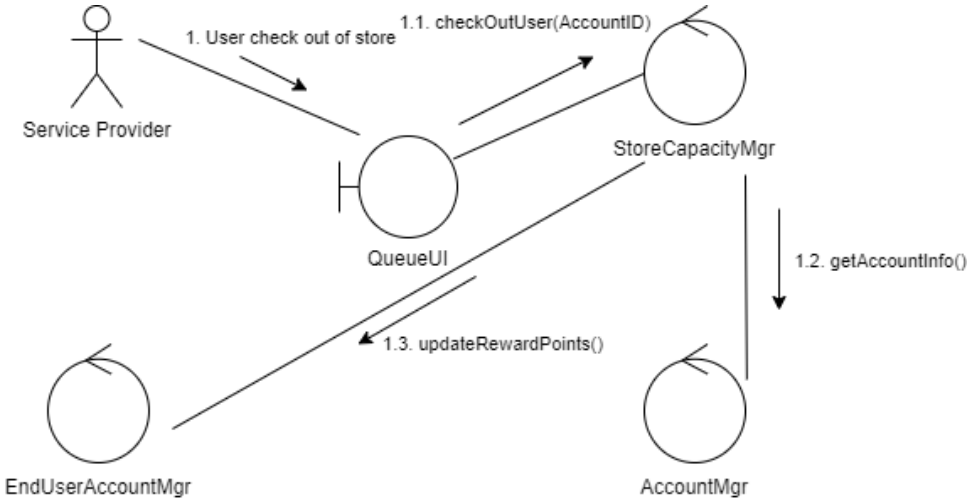


Diagram 5.10.1: Use Case 11 Sequence Diagram

5.10.5 Sequence Diagram

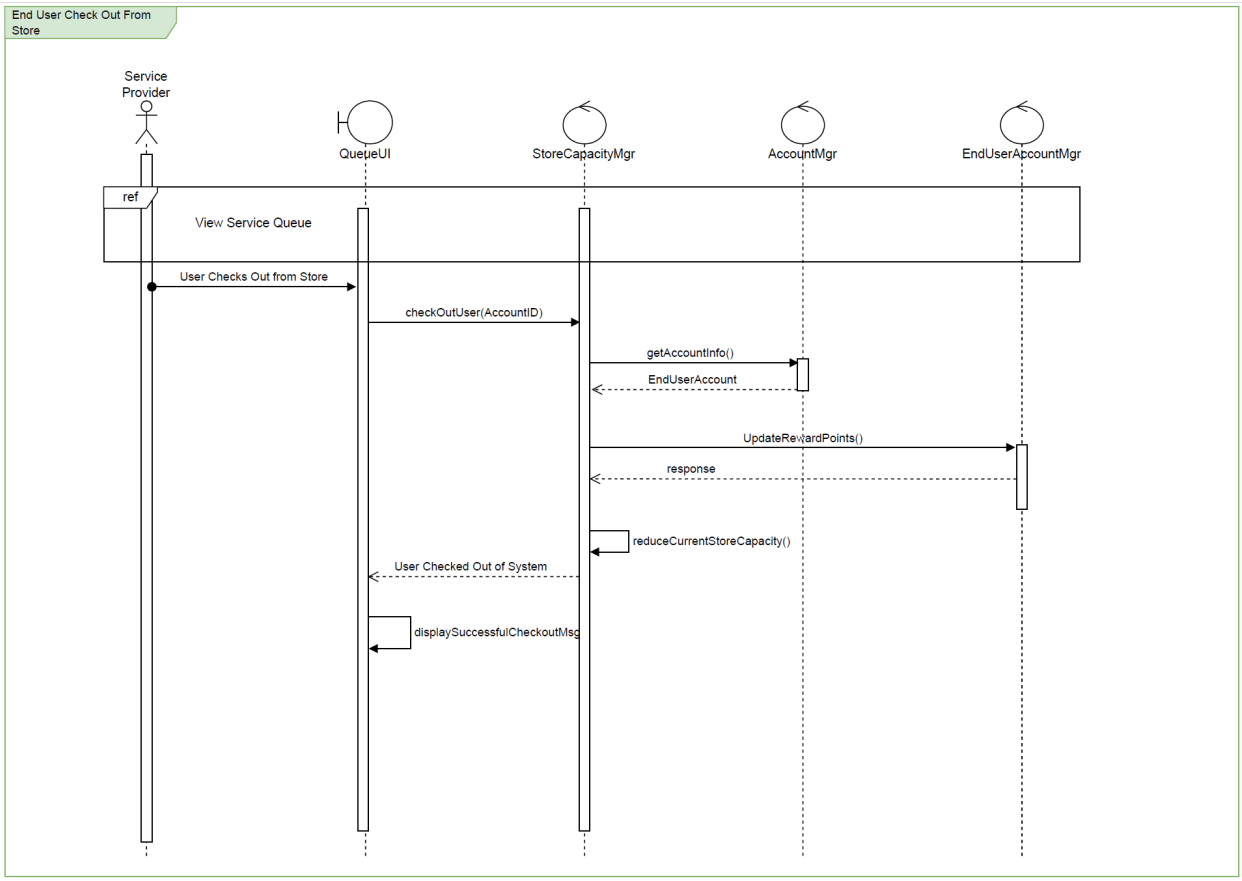


Diagram 5.10.2: Use Case 11 Sequence Diagram

6. Non-Functional Requirements

6.1 Performance Requirements

1. QQueue requires an android device with at least 97.13 MB of memory, 1 GB RAM, and a stable network connection. The performance depends highly on the network speed and the number of stores partnered with the application as it is being pulled from the database.

6.2 Safety Requirements

1. QQueue does not have any safety requirements as all data is held on the database. Therefore, there is no risk of data loss.

6.3 Security Requirements

1. User authentication is needed to verify end-users and Service-Providers.
2. Additionally, Service-Providers have to provide their Unique Entity Number (UEN) to verify the authenticity of their stores.
3. End-Users should be uniquely identified by syncing their Singpass account and all data should be cleared upon exiting the session to prevent storing of personal information. (Unable to implement this as our groups request for the Singpass API was rejected)

6.4 Software Quality Attributes

Speed

1. The system must update the crowds every 10 min
2. Display crowd/waiting time information to the user within 10s

Ease of Use

1. An end-user shall be able to use all system functions within 10 minutes of first use.
2. An end-user shall be able to join a service queue within 2 screen transitions from the home screen
3. Reaching an endpoint in the system must require a minimised number of clicks (3 - 4)
4. All features should be reachable within 5 actions

5. Once an end-user has joined a service queue, the current queue status of that service queue shall be displayed on the home screen
6. Queue number must be shown on the homepage for existing users checking their queue
7. Scalability - show places previously visited/liked

Reliability

1. There shall be 0 incidents of synchronicity of end-user status across different devices
2. There shall be 0 incidents of synchronicity between the service queue status displayed to end-users and Service-Providers
3. Penalty system for false queue
4. Allow users to only queue when they are sufficiently near the area by tracking their location from the store they are trying to join

Robustness

1. In the event of an unexpected system failure, the system shall be restarted within 1 minute
2. There shall be at most 1% of unique scenarios that result in a system failure in a month
3. There shall be at most 1% of system failures that result in data corruption in a month

Portability

1. An end-user shall be able to join the end of the queue using only the system

Usability

1. The application should have Multiple Language support
2. An end-user shall have the option of enabling push notifications to be notified when they are nearing the front of the service queue

6.5 Business Rules

---

1. Required fields in the login/register process are used to prevent erroneous inputs during the process
2. End-User may only be present in one queue at any moment
3. Automatic update of queue time regardless of users refreshing the page

7. System Architecture

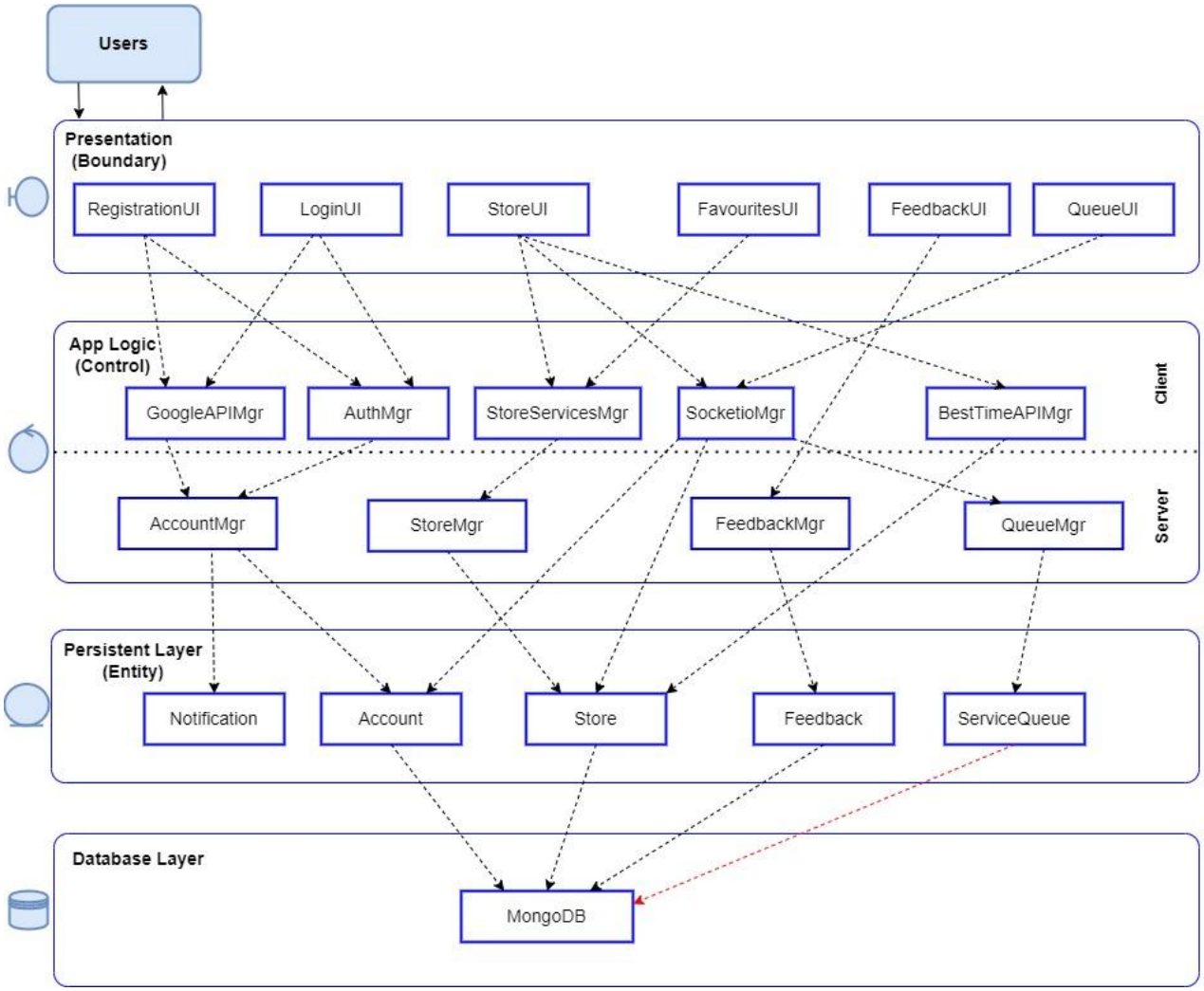


Figure 6: System Architecture

8. Class Diagram

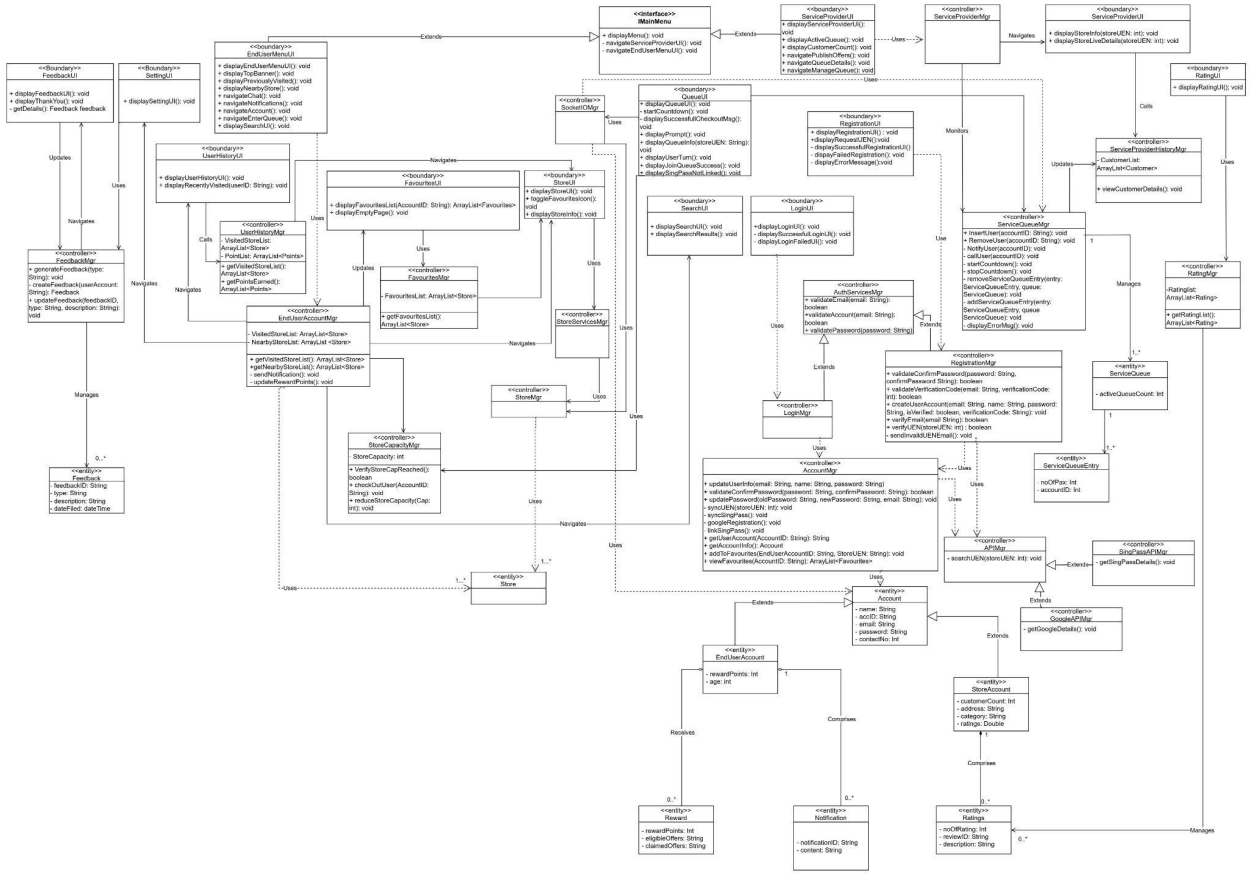


Figure 7: Class Diagram

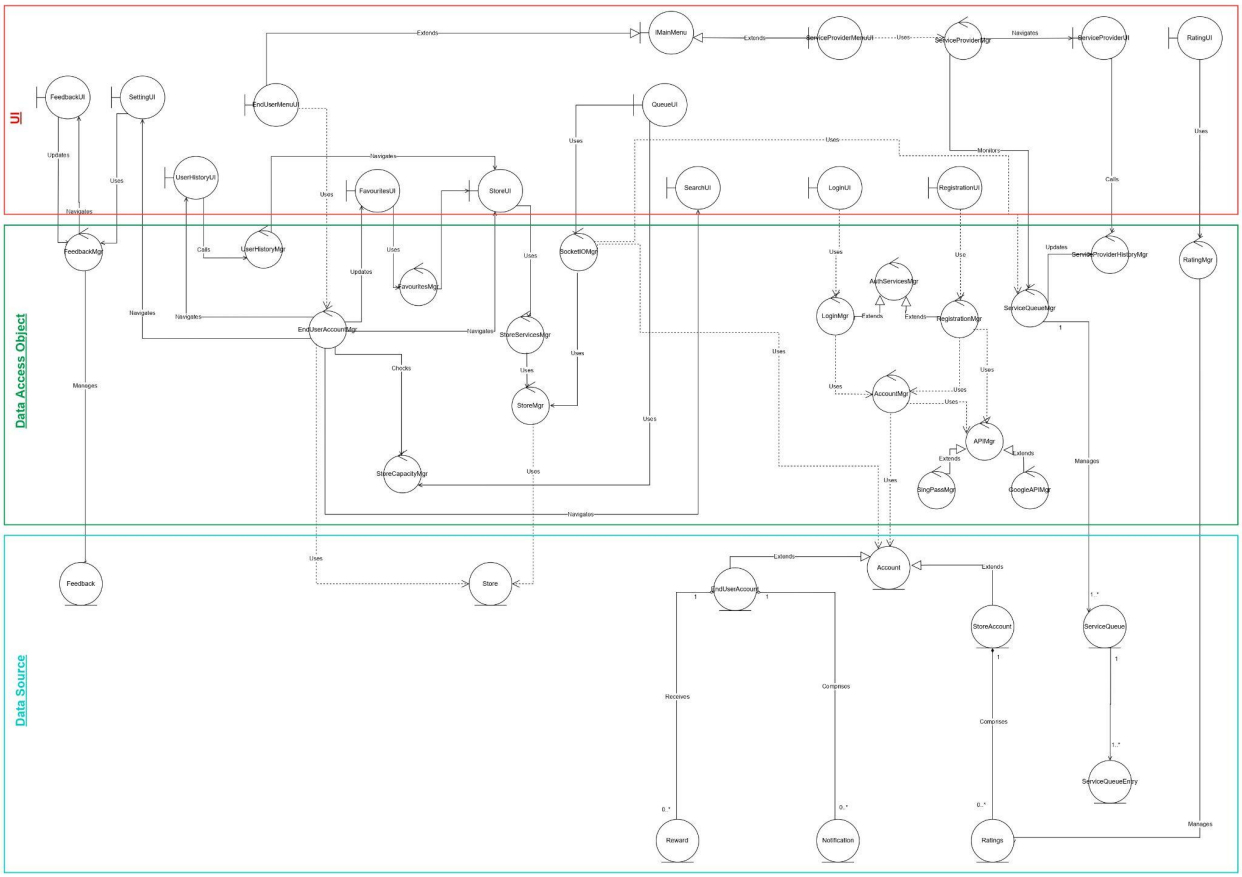


Figure 8: UML Stereotype of Class Diagram

9. Dialog Map

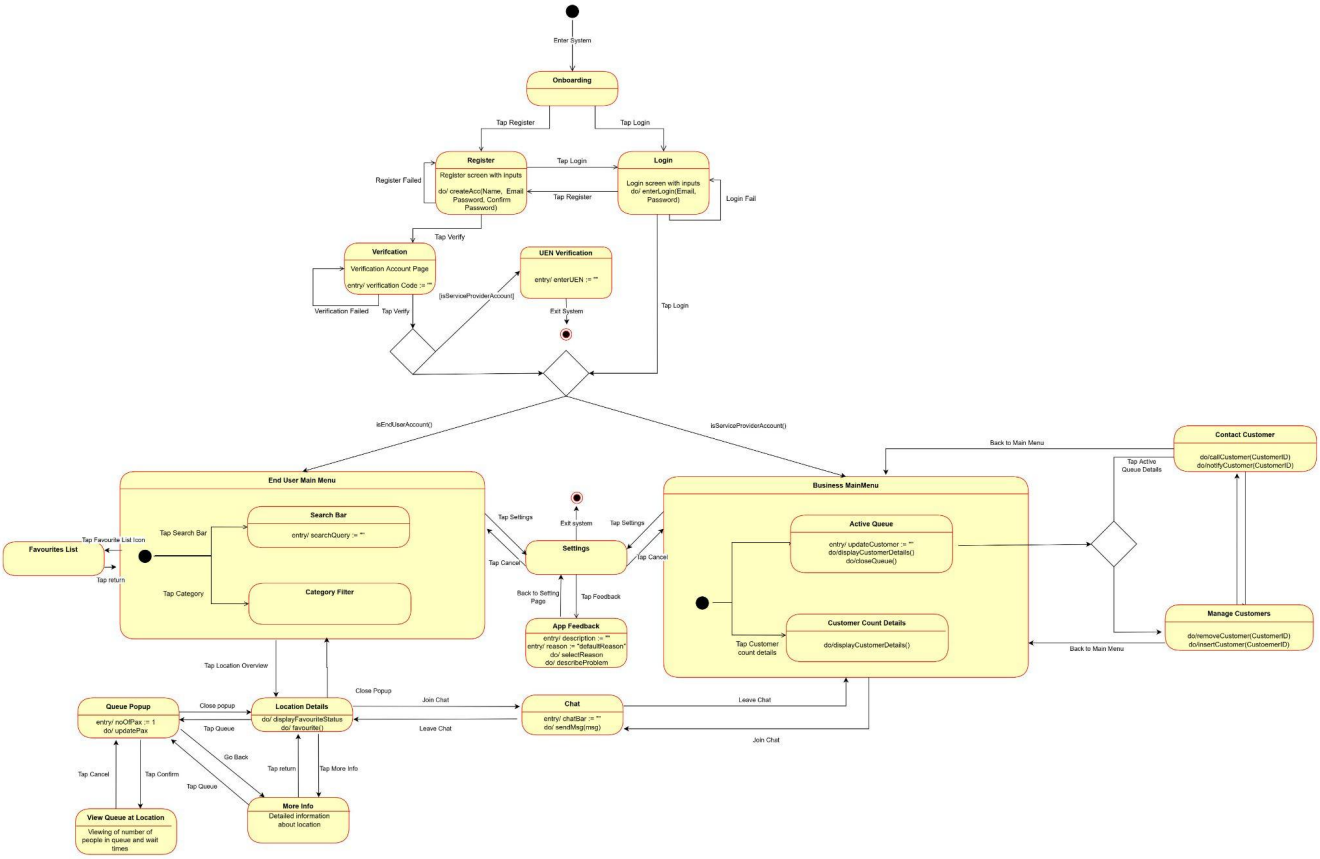


Figure 9: Dialog Map

10. Software Engineering Practices

Listed below are some of the **recommended** software engineering practices our group enforced throughout the software development life cycle:

10.1 Incorporating Design Patterns

We implemented different design patterns throughout our application. One such example is the use of ‘Observer Pattern’ to display the stores based on the automatic update to the user's device location. This pattern supports the principle of loose coupling between objects that interact with each other improving the code structure.

10.2 Pair Programming

We used pair programming on certain features in order to improve the quality of our code and to reduce the number of errors in the code. While this practice increased the time taken to program, it allowed members in the team with more experience to suggest better design methods to those who were less experienced. To ensure the productivity of our application development, pair programming was only used in the development of certain components that were more complex

10.3 Version Control

Our group utilised Github to push and pull changes within our project. We maintained a branch for each member for the separate development of our respective parts and also had a common branch to which we pushed the latest code. Github allows us to resolve merge conflicts easily which speeds up the process. We were also able to refer back to our older version control whenever a serious issue is encountered in a release.

10.4 Design Principles

As our application was done in react-native using functional programming, we tried to create our code structure according to the official documentation for design principles in React [1]. This includes principles such as composition and common abstraction. Utilising good design principles ensures the above.

11. Testing

11.1 Black Box Testing

11.1.1 Login

a. Generic cases

Test ID	Scenario	Expected Result	Actual Result
1	Login with a valid account username and password	The system displays the home screen menu for the user to continue the operation. (401)	The system displays the home screen menu for the user to continue the operation. (401)
2	Login using Google	The system displays the home screen menu for the user to continue the operation.	The system displays the home screen menu for the user to continue the operation.
3	Login without valid credentials	The system prompts the user to enter the credentials again. (now is error 401)	The system prompts the user to enter the credentials again. (now is error 401)
4	Login without filling up required fields	The system prompts the user to fill up the required fields for logging in. (now is error 401)	The system prompts the user to fill up the required fields for logging in. (now is error 401)

b. Specific cases (Combination)

Email	Password	Expected Result	Actual Result
testemail	testpass	Successful login (401)	Successful login (401)
invalidemail	testpass	Invalid email/password (401)	Invalid email/password (401)
Empty(“”)	testpass	Please fill in all required fields (401)	Please fill in all required fields (401)
testemail	Empty(“”)	Please fill in all required fields (401)	Please fill in all required fields (401)
testemail	wrongpass	Invalid email/password (401)	Invalid email/password (401)

11.1.2 Registration

a. Generic cases

Test ID	Scenario	Expected Result	Actual Result

1	Register with valid account username, password and verification code	The system displays the home screen menu for the user to continue the operation.	The system displays the home screen menu for the user to continue the operation.
2	Register with Google	The system displays the home screen menu for the user to continue the operation.	The system displays the home screen menu for the user to continue the operation.
3	Register with incomplete fields	The system prompts the user to fill up the required fields for logging in. (now is error 504)	The system prompts the user to fill up the required fields for logging in. (now is error 504)
4	Register with existing email	The system prompts the user to use a different email for registering. (now is error 401)	The system prompts the user to use a different email for registering. (now is error 401)
5	Register with valid account username and password but wrong verification code	The system prompts the user to fill in the correct verification codes. (now is error 401)	The system prompts the user to fill in the correct verification codes. (now is error 401)
6.	Register with incorrect matching password	The system prompts the user re-enter the password	The system prompts the user re-enter the password
7.	Register with invalid password	The system prompts the user to use a valid password for creation.	The system prompts the user to use a valid password for creation.

b. Specific cases (Combination)

Username	**Email**	**Password**	**Confirm Password**	**VerificationCode**	**Expected Result**	**Actual Result**
testname	testemail	testpass	testpass	correctcode	Successful register	Successful register
Empty(“”)	testemail	testpass	testpass	-	Please fill in all required fields	Please fill in all required fields
testname	Empty(“”)	testpass	testpass	-	Please fill in all required fields	Please fill in all required fields
testname	testemail	Empty(“”)	testpass	-	Please fill in all required fields	Please fill in all required fields
testname	testemail	testpass	Empty(“”)	-	Please fill in all required fields	Please fill in all required fields
testname	existed	testpass	testpass	-	Email has been	Email has been used

	email				used	
testname	testemail	testpass	testpass	wrongcode	Invalid verification code	Invalid verification code
existedname	testemail	testpass	testpass	-	Username has been used	Username has been used
testname	testemail	testpass1	testpass2	-	Password mismatched	Password mismatched
testname	testemail	testpass2	testpass1	-	Password mismatched	Password mismatched
testname	testemail	invalidpass	invalidpass	-	Password less than 8 char	Password less than 8 char

11.1.3 Queue Counter

a. Valid cases

Test ID	Scenario	Expected Result	Actual Result
1	Enter >0 number of pax into the queue.	The system will add a group of that many pax into the queue.	The system will add a group of that many pax into the queue.

11.1.4 User's location

a. Invalid cases

Test Input	Expected Result	Actual Result
Do not allow permission for location service	Authorisation denied.	Authorisation denied.

b. Valid cases

Test Input		Expected Result	Actual Result
1.3483	103.6831	Nanyang Technological University, Block N3.1, Singapore	Nanyang Technological University, Block N3.1, Singapore
1.3477	103.7034	12 Jurong West Street 76, Singapore 648349	12 Jurong West Street 76, Singapore 648349
1.3460	103.6882	152 Nanyang Cres, Crescent Hall, Crescent Hall - Block 17C, Singapore 637123	152 Nanyang Cres, Crescent Hall, Crescent Hall - Block 17C, Singapore 637123

11.2 White Box Testing

11.2.1.1

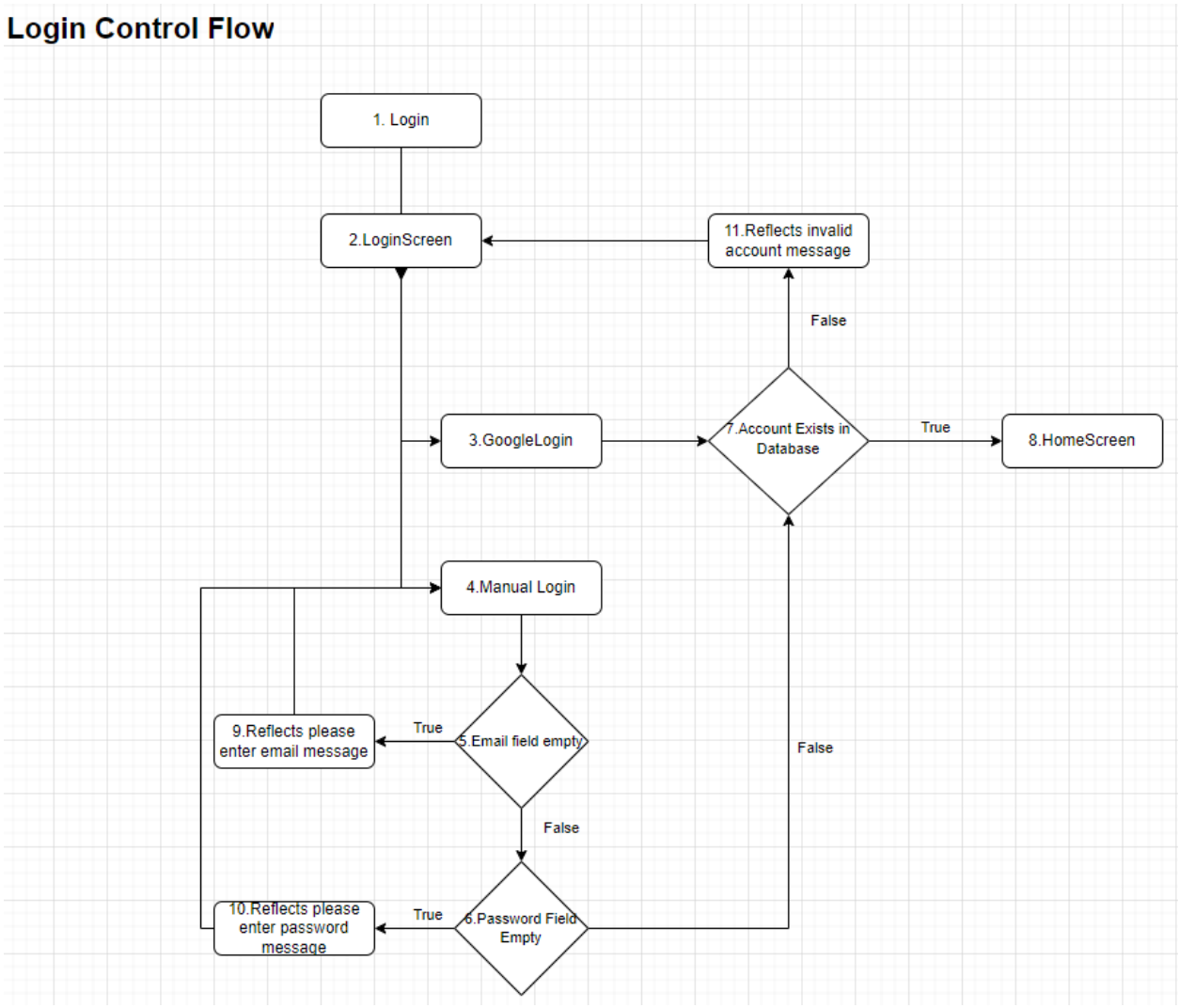


Figure 10: Control Flow Graph for Login

11.2.1.2 Basis Paths

a. {1,2,3,7,8}
b. {1,2,4,5,6,7,8}
c. {1,2,3,11,2}
d. {1,2,4,5,9,4}
e. {1,2,4,5,6,10,4}
f. {1,2,4,5,6,7,11,2}

11.2.1.3 Test Cases

Basis path (a):
The successful login process for a google account. This happens when the user tries to login with a google account that has been registered before and is in our database

Basis path (b):
The successful login process for manual login. This happens when the user ensures that all fields are filed and the account has been registered before and is in our database. The user must also ensure that he enters the correct email and password

Basis path (c):
The failed login process for a google account. This happens when the user tries to login with a google account that has not been registered before and is not in our database

Basis path (d):
The failed login process for manual login. This happens when the user does not fill up the email field and is prompted with an alert to fill up the email field and try again

Basis path (e):
The failed login process for manual login. This happens when the user does not fill up the password field and is prompted with an alert to fill up the password field and try again.

Basis path (f):
The failed login process for manual login. This happens when the user either tries to login with an account that is not registered before and hence not in our database. Another possible failure is that the user has mistyped the username and password pair and is thus not able to log in.

11.2.2.1

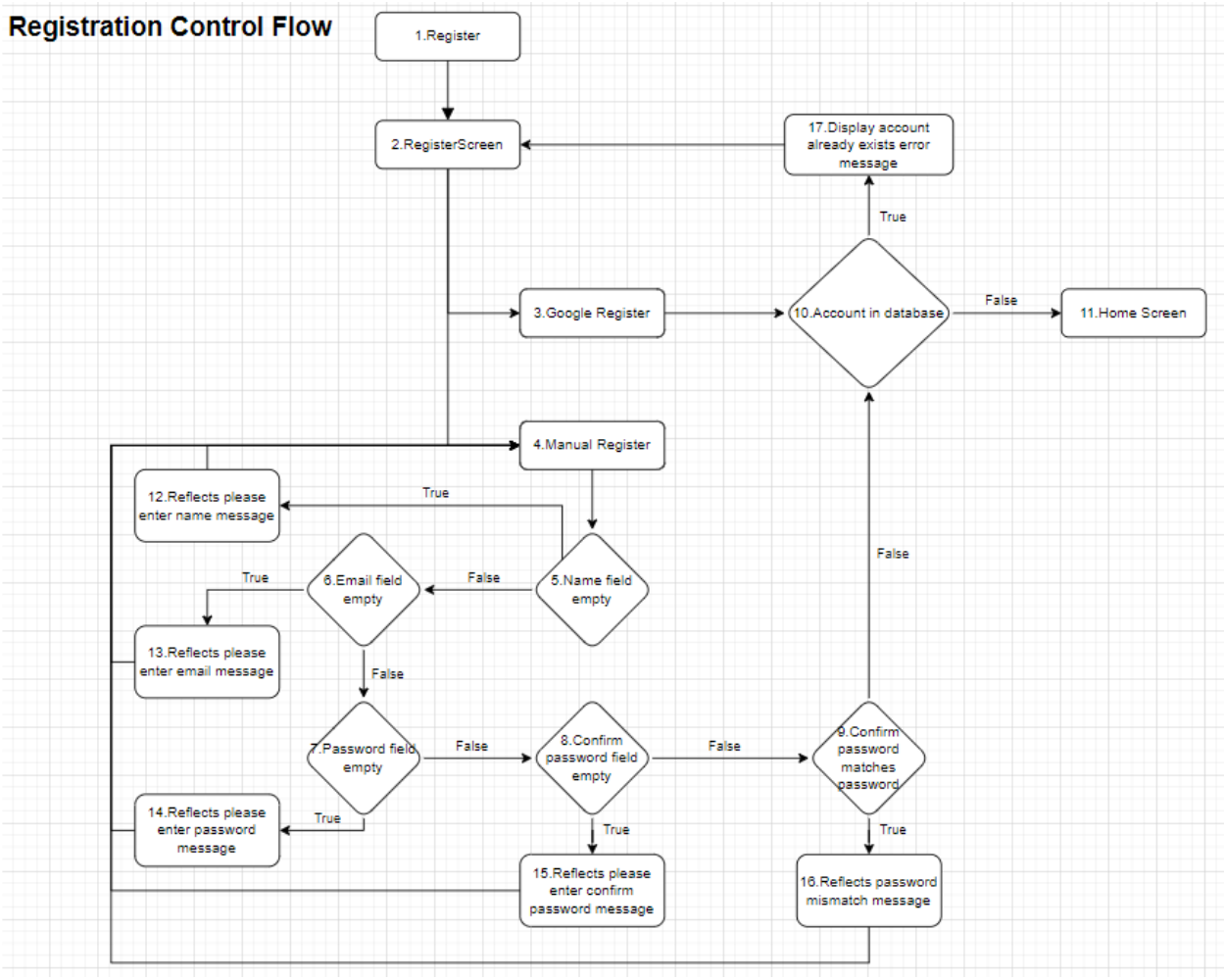


Figure 11: Control Flow Graph for Registration

11.2.2.2 Basis Path

a. {1,2,3,10,11}
b. {1,2,4,5,6,7,8,9,10,11}
c. {1,2,3,10,17,2}
d. {1,2,4,5,12,4}
e. {1,2,4,5,6,13,4}
f. {1,2,4,5,6,7,14,4}
g. {1,2,4,5,6,7,8,15,4}
h. {1,2,4,5,6,7,8,9,16,4}
i. {1,2,4,5,6,7,8,9,10,17,2}

11.2.2.3 Test Cases

Test Case (a):
The successful registration attempt with a google account. This occurs when the user tries to register a google account that is not in our database

Test Case (b):
The successful registration attempt with manual registration. This occurs when all fields are filled up, password and confirm password matches and account has not been created before and is not in our database

Test Case (c):
The failed registration attempt with google account. This occurs when the user tries to register a google account that has been used before and is in our database.

Test Case (d):
The failed registration attempt with manual registration. This occurs when the username field is left empty

Test Case (e):
The failed registration attempt with manual registration. This occurs when the email field is left empty

Test Case (f):
The failed registration attempt with manual registration. This occurs when the password field is left empty

Test Case (g):
The failed registration attempt with manual registration. This occurs when the confirm password field is left empty

Test Case (h):
The failed registration attempt with manual registration. This occurs when the confirm password and password field entries mismatch

Test Case (i):
The failed registration attempt with manual registration. This occurs when the account details used for registration have been used and created before.

11.2.3.1

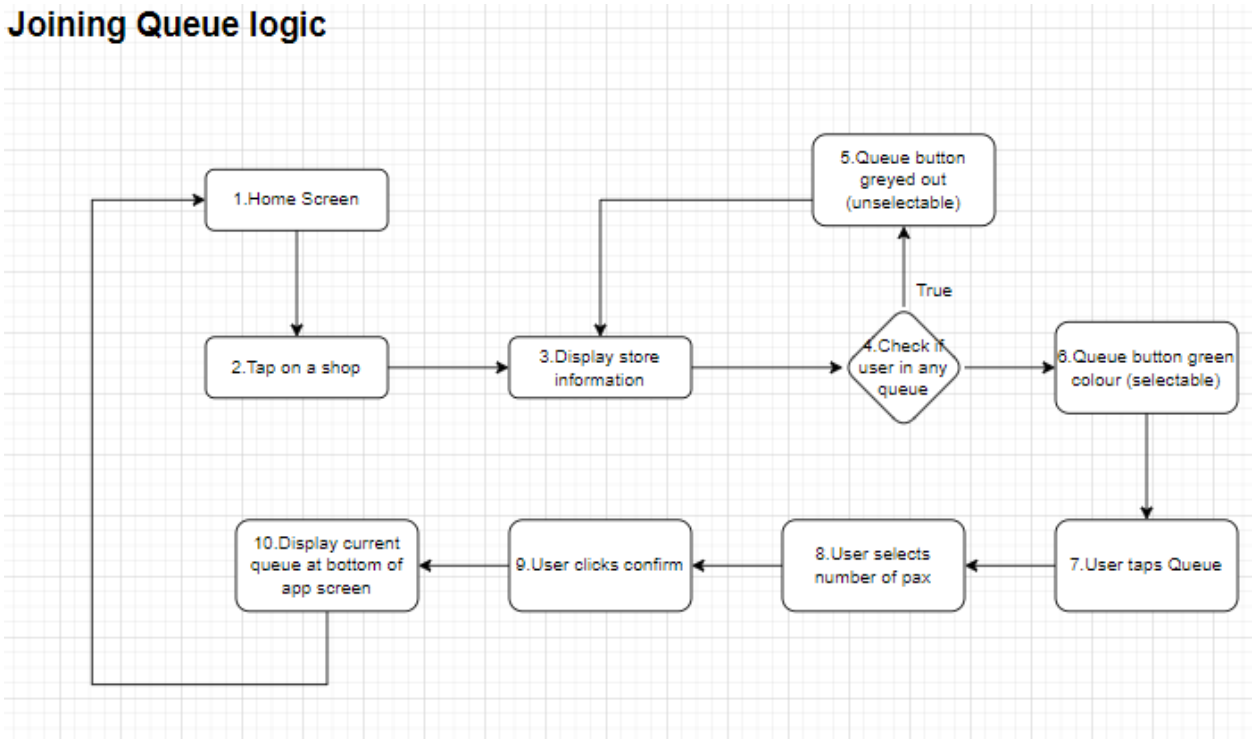


Figure 12: Control Flow for joining queue

11.2.3.2 Basis Path

a. {1,2,3,4,6,7,8,9,10}
b. {1,2,3,4,5,3}

11.2.3.3 Test Cases

Test Case (a):
The successful attempt to join a ServiceQueue. This occurs when the user is logged in and is not part of **ANY** ServiceQueue.

Test Case (b):
The failed attempt at joining a ServiceQueue. This occurs when the user is already part of a ServiceQueue either of the same ServiceProvider or another ServiceProvider.

11.2.4.1

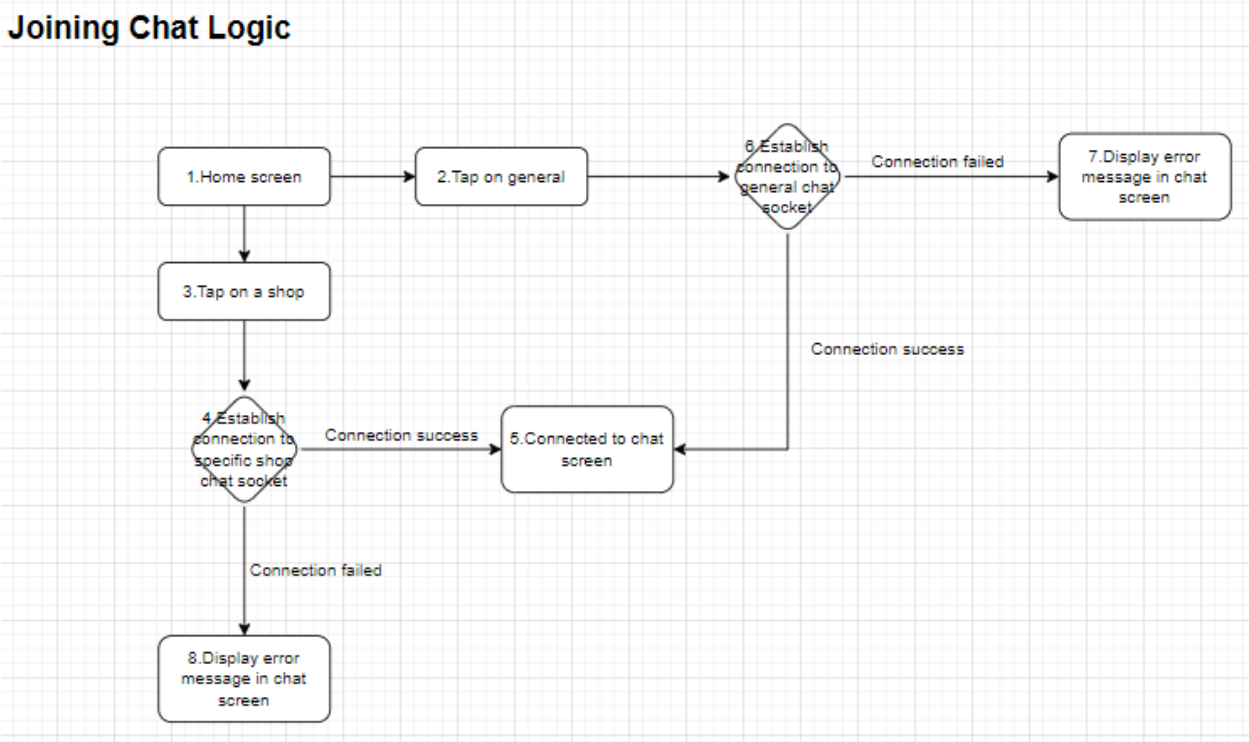


Figure 13: Control Flow for joining chat

11.2.4.2 Basis Path

a. {1,2,6,5}
b. {1,3,4,5}
c. {1,3,4,8}
d. {1,2,6,7}

11.2.4.3 Testing Case

Test Case (a):
The successful connection to the general chat.

Test Case (b):
The successful connection to a ServiceProvider chat.

Test Case (c):
The unsuccessful connection to a ServiceProvider chat. This occurs if the user is unable to establish the websocket connection

Test Case (d):
The unsuccessful connection to a general chat. This occurs if the user is unable to establish the websocket connection.

11.2.5.1

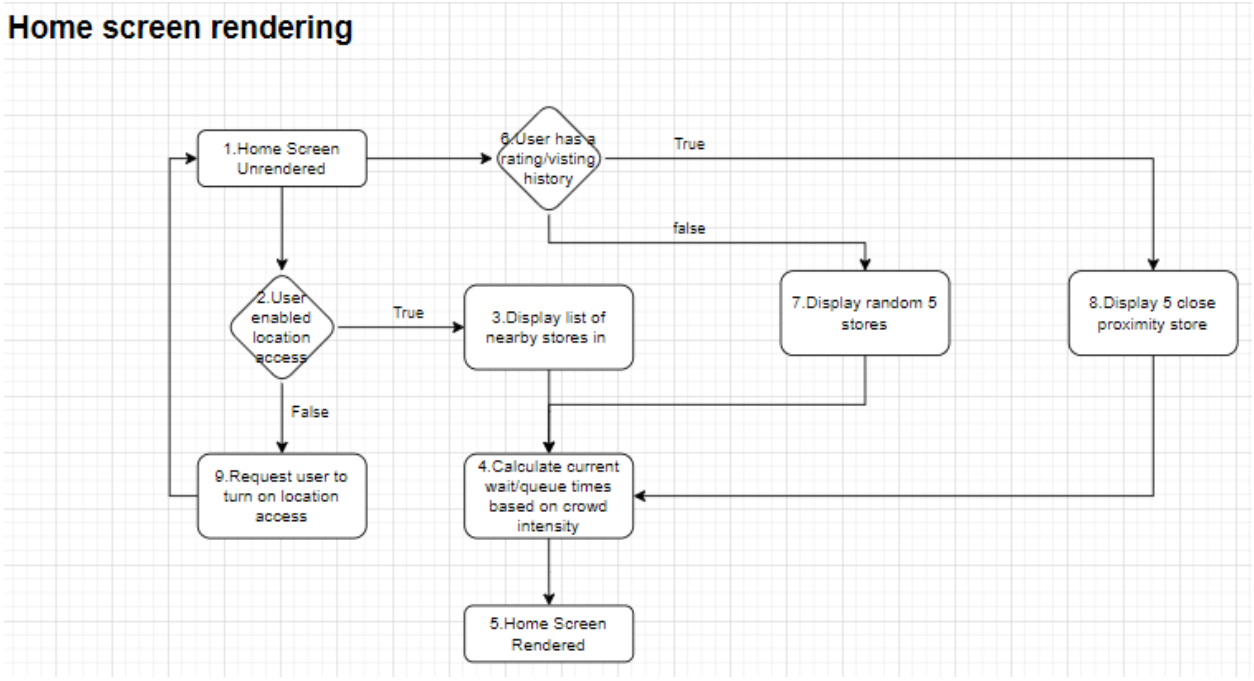


Figure 14:Control Flow for screen rendering

11.2.5.2 Basis Path

a. {1,2,3,4,5} + {1,6,8,4,5}
b. {1,2,3,4,5} + {1,6,7,4,5}
c. {1,2,9,1}

11.2.5.3 Testing Cases

Test Case (a):
The successful rendering of the home screen. This occurs when both paths are successful and stores are pulled from the MongoDB database to be displayed on the user screen. In the second half of the basis path is the recommendation feature of the application recommending the top 5 stores based on proximity to the user.

Test Case (b):
The successful rendering of the home screen. This occurs when both paths are successful and stores are pulled from the MongoDB database to be displayed on the user screen. In the second half of the basis path is the recommendation feature of the application just recommending 5 random stores due to the absence of a user history.

Test Case (c):
The unsuccessful rendering of the home screen. This occurs when the user has not enabled user location to be enabled.

11.2.6.1

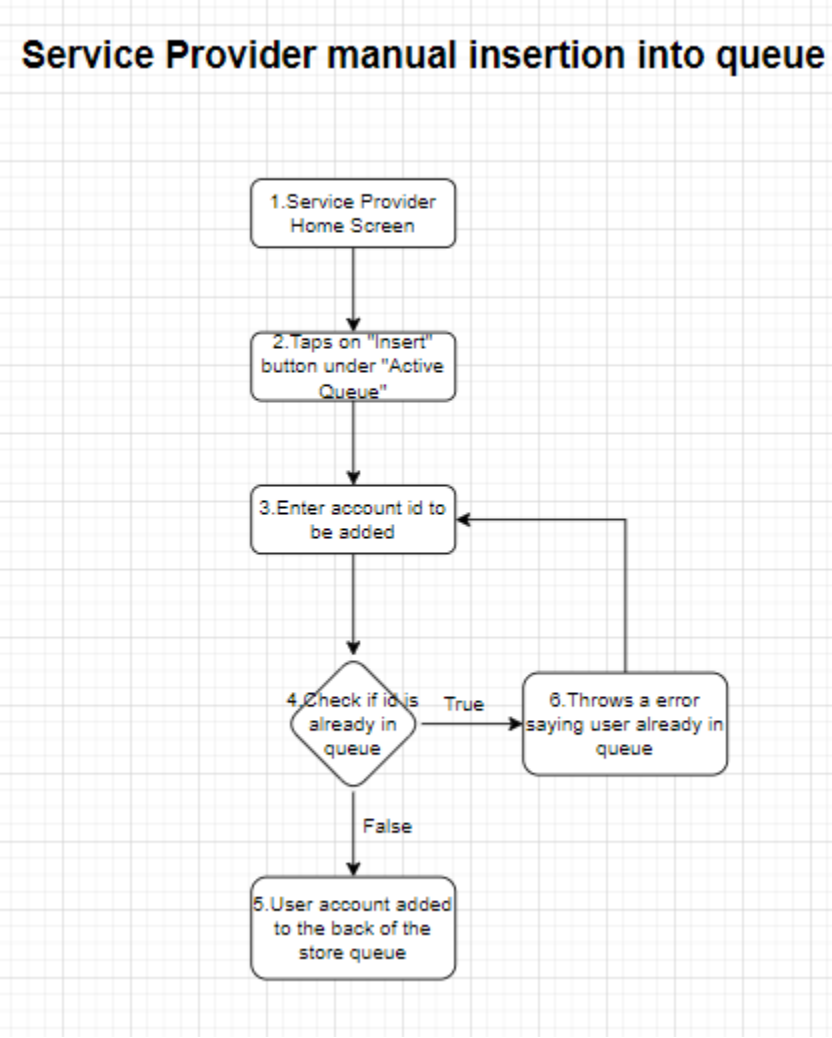


Figure 15: Control Flow for manual insertion of user to queue

11.2.6.2 Basis Path

a. {1,2,3,4,5}
b. {1,2,3,4,6,3}

11.2.6.3 Testing Cases

Test Case (a):
The successful addition of an End User to a ServiceQueue manually.

Test Case (b):
The failed addition of an End User to a ServiceQueue manually. This occurs when a End User is already in a ServiceQueue of another ServiceProvider.

12. Other Requirements

12.1. Legal Requirements

Copyright laws and licence agreements must be respected for any third-party software used in the creation of this system.

12.2. Database Requirements

The software requires an internet connection to access MongoDB M0 Sandbox (General) for cloud data storage.

13. Appendix

13.1 Data Dictionary

13.1.1. Data Dictionary: User

Element Name		**Description**
User		This is a model schema of the users in the application (End-Users, Service-Providers)
Attributes	accountType: Object	The account type of a user End-User: A customer or potential customer at a specific establishment using the system. Service-Provider: An establishment utilizing the system to complement or replace their physical queuing system
	googleRegistered: bool	Referencing whether the account was registered via google
	email: String	Text box for users to input email
	userName: String	Text box for users to input username
	Password: String	Text box for users to input password

13.1.2. Data Dictionary: RegistrationForm

Element Name		**Description**
RegistrationForm		Functional component that handles registering of users into MongoDB
Attributes	userName: String	Text box for users to input username
	email: String	Text box for users to input email
	password: String	Text box for users to input password
	confirmPassword: String	Text box for users to input confirmed password
	_id: ID	An ID allocated by default by MongoDB to uniquely identify this user
Operation	registerOnPress(props): void	This function is called to register a new account with the input attributes

13.1.3. Data Dictionary: LoginForm

Element Name		**Description**
LoginForm		Functional component to authenticate users in MongoDB
Attributes	email: String	Text box for users to input email
	password: String	Text box for users to input password
Operation	submitForm(props): void	This function is called to check if the user is a valid account

13.1.3. Data Dictionary: ServiceQueue

Element Name		**Description**
ServiceQueue		This is a model schema of the queue
Attributes	venueID	The unique store ID allocated to a Service-Provider
	userID	The unique ID allocated to an end-user
	pax: int	The number of people in the queue
	queueNumber: int	The queue number allocated when a user joins the queue
Operation	joinQueue(props): void	Adds the record to the queue
	pushQueue(): Object	Returns the JSON serialized object for the next person in the queue
	leaveQueue(props): void	Remove the record (if exists) from the queue

13.1.3. Data Dictionary: General

API	Application Programming Interface.
Authentication	The process of verifying if a user is registered in the system.
Check Out	The action of checking out is done by the user and is the act of the user leaving the store/attraction/location
Feedback	Feedback is what users can provide regarding the APPLICATION via the application feedback page. They may also report any issues with the app through the reports page

Google Services	An authentication API
Join Queue	The action of getting added/entering a ServiceQueue of a specific Service-Provider
Leave Queue	The action of getting removed/exiting a ServiceQueue of a specific Service-Provider
MongoDB	MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas
Queue Status	Contains the following information about the service queue: Number of end-users in ServiceQueue, Estimated wait time
Rating	What users can provide regarding their experience with a particular store/attraction/location. It is given via the app after the user has successfully checked out of the store
Store/Attraction/Location	A store, location and attraction refer to the same thing in this context, that is the physical setting of a particular store/location/attraction
System	The application as a whole, including the user interface and all backend services
Terminology	Description
User Interface	The boundary used to collect input from and display output to a user
venueID	venueID uniquely identifies each serviceProvider in the application

13.2 Use Case Descriptions

13.2.1. UC1 End-User Joining Service Queue

Use Case ID:	1		
Use Case Name:	End-User Joining Service Queue		
Created By:	Nicholas	Last Updated By:	Nicholas
Date Created:	02/2/22	Date Last Updated:	16/04/2022

Actor:	End-User
Description:	Process for End-User to join
Preconditions:	1. End-User Logged in 2. End-User not in queue 3. End-User account is synced with SingPass
Postconditions:	1. End-User joined a queue
Priority:	Medium
Frequency of Use:	0 - 3 per day
Flow of Events:	1. End-User taps on a location in the Home Screen 2. End-User taps on “Queue” button 3. End-User indicates how many pax he wants to queue for 4. End-User taps on confirm

Alternative Flows:	-
Exceptions:	-
Includes:	-
Special Requirements:	-

Assumptions:	-
Notes and Issues:	-

13.2.2. UC2 Process for End-User to leave queue

Use Case ID:	2		
Use Case Name:	Process for End-User to leave queue		
Created By:	Nicholas	Last Updated By:	Nicholas
Date Created:	2/2/22	Date Last Updated:	11/02/22

Actor:	End-user
Description:	How the End-User can go about leaving the queue

Preconditions:	End-User logged in End-User is in a queue
Postconditions:	End-User has information on queue time for locations
Priority:	Medium
Frequency of Use:	0 - 3 per day
Flow of Events:	1. End-User's turn is reached, Service-Provider will notify End-User of turn 2. End-User arrives at the location, and scans QR code at Location to indicate they left the queue.
Alternative Flows:	1-1-AC-1: User chooses to drop out of queue 1. End-User taps on the tab for the location they are queuing for. 2. End-User taps on "Leave" 3. The popup appears confirming if the End-User wants to drop out 4. End-User taps "Drop out from queue"
Exceptions:	-
Includes	1. Viewing Service Queue
Special Requirements:	-

Assumptions:	-
Notes and Issues:	-

13.2.3. UC3 End-User View Service Queue

Use Case ID:	3		
Use Case Name:	End-User View Service Queue		
Created By:	Nicholas	Last Updated By:	Nicholas
Date Created:	2/2/22	Date Last Updated:	16/04/2022

Actor:	End-User
Description:	End-User Viewing Queue
Preconditions:	End-User Logged in
Postconditions:	User Logged in
Priority:	High
Frequency of Use:	0 - 10 per day

Flow of Events:	1. End-User is at the Home Screen with recommended/nearby locations

	2. User taps on a Location in the Home Screen 3. User is able to view the number of people in Service Queue and the projected wait times
Alternative Flows	1-3-AC-1: User wants live update 1. End-User taps on “Chat” button 2. A chat screen opens and End-User is able to view live chat from that particular Location/Store.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

13.2.4. UC4 Register Service-Provider Account

Use Case ID:	4		
Use Case Name:	Register Service-Provider Account		
Created By:	Lucas	Last Updated By:	Lucas
Date Created:	01/02/2022	Date Last Updated:	16/04/2022

Actor:	Service-Provider
Description:	Account Creation Process for Service-Providers using Google Sign-In Client.
Preconditions:	1. The Service-Provider is not logged in.
Postconditions:	1. A new Service-Provider account is added to the database. 2. Service-Provider is able to login to view and manage their own Service Queue
Priority:	Medium
Frequency of Use:	1 - 3 times per lifetime
Flow of Events:	1. Service-Provider is prompted to enter a username, phone number, email address, password, and confirm password. 2. Service-Provider selects the Sign-Up button. 3. The System will ensure that both password fields are identical and the phone number and email are of valid formats. 4. The System will validate account availability. 5. If there is no existing account, then the Service-Provider is prompted to enter their business venueID 6. The System will verify venueID

	7. Once verified, the System creates a ServiceQueue for the Service-Provider. 8. The System displays the log-in screen.
Alternative Flows:	4-1-AC-1: Google Sign-In Client Registration 1. Service-Provider is prompted to Register Via Google Sign-In Client. 2. Once the Service-Provider is successfully signed into Google, return to Step 4.

Exceptions:	4-4-EX-1: The Google Account selected is already linked to an existing Service-Provider account. 1. The System displays an error message “This account already exists. Would you like to log in instead?” 4-5-EX-2: The venueID provided fails the verification. 1. The System displays an error message “The venueID provided could not be verified. Please enter a valid venueID.” 2. Return to Step 5. 4-5-EX-3: The venueID provided is already linked to an existing account. 1. The System displays an error message “The provided venueID is already linked to an existing account. Please log in using that account instead.”
Includes:	1. Verify Service-Provider venueID

	2. Validate Account Availability
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

13.2.5. UC5 View Service Queue

Use Case ID:	5		
Use Case Name:	View Service Queue		
Created By:	Lucas	Last Updated By:	Nicholas
Date Created:	01/02/2022	Date Last Updated:	11/02/2022

Actor:	Service-Provider
Description:	Allows the Service-Provider to view their own Service Queue
Preconditions:	1. Service-Provider is logged into their account.
Postconditions:	1. Service Queue details will be displayed on the screen.
Priority:	Medium

Frequency of Use:	10 - 20 times per day
Flow of Events:	1. Service-Provider see an overview of his current store's queue 2. Service-Provider taps on "Details" under "Active Queue" to view a detailed list of people queuing 3. The System displays a detailed view of Service Queue.
Alternative Flows:	-
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

13.2.6. UC6 Insert End-User into Service Queue

Use Case ID:	6		
Use Case Name:	Insert End-User into Service Queue		
Created By:	Lucas	Last Updated By:	Yi Hao
Date Created:	01/02/2022	Date Last Updated:	06/02/2022

Actor:	Service-Provider
Description:	Service-Provider manually inserts an End-User into the Service Queue.
Preconditions:	1. Service-Provider is logged into their account.
Postconditions:	1. The inserted End-User is found at the right position in the Service Queue. 2. The updated Service Queue is displayed.
Priority:	Medium
Frequency of Use:	0 - 10 times per day
Flow of Events:	1. Service-Provider taps on “Insert” button of Service Queue. 2. The System prompts the Service-Provider to enter the username of End-User. 3. Service-Provider enters the username of the End-User to be inserted. 4. The System adds the End-User to the end of the Service Queue.
Alternative Flows:	6-4-AC-1: The username entered is invalid. 1. The System displays an error message “There is no account associated with this username.” 2. Return to Step 2 6-3-AC-2: Insert physical customers that do not have the app into the queue.

	1. Physical End-User provides their Name, Phone Number, Group Size 2. Service-Provider enters the details into the system 3. Queue number is generated for Service-Provider to drag and insert accordingly
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

13.2.7. UC7 Remove End-User from Service Queue

Use Case ID:	7		
Use Case Name:	Remove End-User from Service Queue		
Created By:	Lucas	Last Updated By:	Nicholas
Date Created:	01/02/2022	Date Last Updated:	16/04/2022

Actor:	Service-Provider
Description:	Service-Provider manually removes an End-User from the Service Queue.

Preconditions:	1. Service-Provider is logged into their account.
Postconditions:	1. The removed End-User is no longer in the Service Queue. 2. The updated Service Queue is displayed.
Priority:	Medium
Frequency of Use:	0 - 5 times per day.

Flow of Events:	1. Service-Provider taps on the “Delete” button of Service Queue. 2. The System displays a detailed view of the Service Queue. 3. Service-Provider selects the “Trash Can” icon next to the End-User to be removed. 4. The System prompts the Service-Provider for confirmation to remove the selected End-User from the Service Queue. 5. The Service-Provider confirms the decision to remove the selected End-User. 6. The System removes the selected End-User from the Service Queue.
Alternative Flows:	7-5-AC-1: The Service-Provider cancels the decision to remove the selected End-User. 1. The System displays a detailed view of the Service Queue.

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

13.2.8. UC8 End-User providing Feedback

Use Case ID:	8		
Use Case Name:	End-User providing Feedback		
Created By:	Yi Hao	Last Updated By:	Nicholas
Date Created:	04/02/2022	Date Last Updated:	16/04/2022

Actor:	End-User
Description:	End-User submits a Feedback or reports an issue using the application
Preconditions:	1. The End-User must be logged in to the system with a valid account
Postconditions:	1. The End-User will receive a notification informing them the Feedback was received

Priority:	Low
Frequency of Use:	0-2 times per day.
Flow of Events:	1. End-User selects “Report” button 2. End-User selects from a list of categories of either Feedback or issues to report 3. End-Users presses the “Submit” button 4. System will generate a notification to inform the End-User of the submission and log the result
Alternative Flows:	8-3-AC-1: End-User provides Feedback description 1. End-User fills up an optional description box. 2. Return to Step 3.
Exceptions:	-
Includes:	1. Log Feedback
Special Requirements:	-
Assumptions:	1. System can draw on its most popular feedback to list them out
Notes and Issues:	

13.2.9. UC9 End-User Favouring Store & Viewing Favourites

Use Case ID:	9		
Use Case Name:	End-User Favouring Store & Viewing Favourites		
Created By:	Yi Hao	Last Updated By:	Nicholas
Date Created:	04/02/2022	Date Last Updated:	16/04/2022

Actor:	End-User
Description:	End-User searching and saving stores on his favourites to view
Preconditions:	1. User logged in
Postconditions:	1. End-User favourited a location
Priority:	Low
Frequency of Use:	0 - 5 times per day
Flow of Events:	1. End-User searches for any stores either using the search function or through the home screen 2. End-User taps on the location he wants to favourite 3. End-User selects the “ ” button found at the corner of all Stores' information section 4. System reflects the change by updating the display of the “ ” button

Alternative Flows:	9-1-AC-1: End-User viewing his list of favourite Stores 1. End-User selects the “ ” button at the Home Screen 2. System returns an alphabetically sorted list of the users favourites stores or displays “No stores have been added” if End-User has not saved any Locations before
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

13.2.10. UC10 End-User Registers Account

Use Case ID:	10		
Use Case Name:	End-User Registers Account		
Created By:	Yi Hao	Last Updated By:	Yi Hao
Date Created:	04/02/2022	Date Last Updated:	16/04/2022

Actor:	End-User

Description:	End-User registering a new account
Preconditions:	1. The End-User cannot have an existing account in the system
Postconditions:	1. The End-User receives a confirmation message
Priority:	High
Frequency of Use:	1 - 2 times per lifetime

Flow of Events:	1. End-User enters their Name, Password, Confirm Password, and Email 2. End-User selects the "Register" button 3. The System will validate account availability. 4. System sends a verification code to the End-User's email 5. End-User enters successful verification code 6. System prompts End-User to sync their SingPass 7. End-User connects their SingPass account 8. System acknowledges the successful creation of the account

Alternative Flows:	10-1-AC-1: Google Sign-In Client Registration 1. End-User may instead choose to register with their google account 2. System opens the users saved google accounts and request permission to obtain details 3. End-User grants permission to the system and the verification code will be sent to the registered email address 4. Return to step 5 10-6-AC-2: End-User does not want to sync their SingPass account 1. End-User selects “No” 2. Return to step 8 10-4-AC-3: End-User Google Account is already used 1. End-User receives a notification that their Google Account is already in use 2. End-User is returned to the Registration Screen
Exceptions:	-
Includes	1. Validate Account Availability
Special Requirements:	-
Assumptions:	Google authentication is enabled and integrated into the System

Notes and Issues:	-

13.2.11. UC11 End-User Checking out of store

Use Case ID:	11		
Use Case Name:	End-User Checking Out Of Store		
Created By:	Lucas	Last Updated By:	Lucas
Date Created:	06/02/2022	Date Last Updated:	16/04/2022

Actor:	End-User
Description:	Process through which the End-User checks out of the Store to earn points
Preconditions:	1. The End-User must be logged in
Postconditions:	1. The End-User receives points
Priority:	Low
Frequency of Use:	0 - 10 times per day
Flow of Events:	1. End-User chooses to “Check Out” from the Location. 2. The System moves the next End-User that has the same group size or smaller to the front of the Service Queue.

	3. The system awards points to the End-User using the Point System. 4. End-User receives a notification that they have been awarded points.

Alternative Flows:	-
Exceptions:	-
Includes:	1. Point System
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

13.2.12. UC12 Service-Provider Notifies End-User Of Availability

Use Case ID:	12		
Use Case Name:	Service-Provider Notifies End-User Of Availability		
Created By:	Lucas	Last Updated By:	Nicholas
Date Created:	06/02/2022	Date Last Updated:	16/04/2022

Actor:	Service-Provider

Description:	Process through which Service-Provider manually notifies End-User of space availability at the Location
Preconditions:	1. The Service-Provider must be logged in 2. There is space in the location to accommodate an End-User not at the front of the Service Queue
Postconditions:	1. End-User no longer in Service Queue
Priority:	Medium
Frequency of Use:	0 - 10 times per day
Flow of Events:	1. Service-Provider View Service Queue. 2. Service-Provider taps on the "Bell" button for the End-User nearest the front of the Service Queue, whose group size can fit the available space. 3. The System sends a notification to the End-User, indicating that "A Space is Available". 4. End-User receives the notification. 5. End-User removed from Service Queue.
Alternative Flows:	-
Exceptions:	12-4-EX-1: End-User rejects or fails to acknowledge notification after 5 minutes 1. The System notifies the Service-Provider that the End-User has not responded, and prompts the

	Service-Provider to notify the next suitable End-User. 2. If the Service-Provider accepts the suggestion, return to Step 3. Otherwise, end flow of events.
Includes:	1. Leave Service Queue
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

14. References

[1] Virtual Queueing,
https://www.imakan.com.sg/why-should-you-move-your-customers-to-a-virtual-queue
[2] Design Principles in React, https://reactjs.org/docs/design-principles.html
[3] BestTime API, https://besttime.app/#api