

**NANYANG  
TECHNOLOGICAL  
UNIVERSITY**  

---

**SINGAPORE**

School of Computer Science & Engineering  
CZ2006 Software Engineering  
Project Name – SG Discoverer

**Lab Group: SS3**  
**Team Name: Team KOK**

**Team Members:**

Ang Jun Liang (Leader) (U1722336E)

Yee Wei Min (U1720243E)

Nigel Ang Wei Jun (U1721087C)

Ang Yong Loong (U1720458G)

Kok Zi Ming (U1721517B)

Chua Jia Ren (U1720720K)

<b>Software Requirements Specification (SRS)</b>	<b>4</b>
<b>1. Product Description</b>	<b>4</b>
1.1 Purpose	4
1.2 Scope of the System	4
1.3 Users and Stakeholders	4
1.4 Assumptions and Constraints	4
1.5 Future scope	4
<b>2. Functional Requirements</b>	<b>5</b>
2.1 Use Case Diagram	8
2.2 Use Case Descriptions	9
2.2.1 Create Account	9
2.2.2 Login Account	10
2.2.3 Log Out Account	11
2.2.4 Favourite location	12
2.2.5 Set User's location	14
2.2.6 Discover Point of Interests	15
2.2.7 Sort by Distance	16
2.2.8 Sort by Ratings	18
2.2.9 Provide POI's Details	19
2.2.10 Provide Directions from User's Location to POI	20
2.2.11 Weather forecast	22
2.2.12 Discounts feature	23
2.3 System Architecture diagram	25
2.4 Design Patterns	26
2.5 Class Diagrams	29
2.6 Sequence Diagrams	30
2.6.1 Login Account	30
2.6.2 Discover POIs	31
2.6.3 Sort by Distance	31
2.6.4 Favourite Location	32
2.6.5 Display Weather Forecast	32

2.6.6 Display Discount Available	33
2.7 State transition diagram	34
2.7.1 SG Discoverer	34
2.7.2 Create account	34
2.7.3 Set User Location	35
2.7.4 SearchforPOIs	35
2.7.5 Sort POIs	36
2.7.6 GetDirectionsToPOILocation	36
2.7.7 ViewDiscounts	36
2.8 Website Screenshot	37
2.8.1 Register page	37
2.8.2 Login page	38
2.8.3 Home page	38
2.8.4 Map view	39
2.8.5 Satellite view	39
2.8.6 Map view after searching (Marina Square)	40
2.8.7 POI List after sort by rating (Marina Square)	40
2.8.8 Weather forecast	41
2.8.9 Favourites	41
2.8.10 Get directions	42
2.8.11 Deals page	42
<b>3. Non-Functional Requirements</b>	<b>43</b>
<b>4. Interface Requirements</b>	<b>44</b>
4.1 User interface	44
4.2 Hardware	45
4.3 Software	45
<b>5. Data Dictionary</b>	<b>45</b>
<b>6. Testing</b>	<b>48</b>
6.1 Black Box Testing	48
6.2 White Box Testing	52

# Software Requirements Specification (SRS)

## 1. Product Description

### 1.1 Purpose

SG Discoverer website aims to help users find amenities that they might be interested through the use of categories and search radius. It will help to solve problems of decision making and allow us to discover more places that we never knew of. 20 seconds is all it takes to discover your next destination.

### 1.2 Scope of the System

What makes our website special is the way in which users can discover places specific to a certain district and also finding Place of Interest (POI) near his current location. In addition, users can sort the places by rating and distance from an input starting reference location. Also, there are additional features to make your experience more meaningful such as providing weather information of the various areas of Singapore to provide users with more information to make a better decision and having the ability to favourite POI to allow convenience and future planning. There is also a page which displays attractive food deals that are ongoing around the island given with their respective details and outlet that may possibly be your next destination.

### 1.3 Users and Stakeholders

The stakeholders for this project are the users and SG Discoverer. Users can be anyone with a device that can be connected to the internet. These people can be looking for new possible destinations to explore or simply just food choices and amenities of an area to better spice up their lives or assist in their decision making.

Information of amenities/facilities and weather are collected using the Data.gov.sg weather API and Google Maps API. Deals are scraped from SG Kiasu Foodies [Discounts/Promos]. Images of POI are scraped from streetdirectory.com and Yahoo Images.

### 1.4 Assumptions and Constraints

Users have Global Positioning System (GPS) installed in their device for our website to collect their current location and their device must be able to run the website.

Users should have internet access to be able to use SG Discoverer's features.

### 1.5 Future scope

1. Host the website online or as an app online.
2. Test the screen display on other platforms such as tablets and mobile phones.
3. Use hashing for storing users' password in database to increase security.
4. Have forget username/password feature by connecting to user's their email account.
5. Have security question to enhance account security.
6. Link to facebook/google login.
7. Allow users to input their preference into their account such as food choices.
8. Recommend food or events on startup based on user preference.
9. Recommend food or events based on user past searches using cookies.
10. Have operational hours, telephone number and reservation link to the place.
11. Have further sub-category for the food places such as ice cream, coffee instead of just cafe.
12. Improve accuracy of data.

13. Allow shop owners to have their own account to manage their places such as add location or photos and details.
14. Allow input of exact POI onto search bar.
15. Have autocomplete or suggestions when using search bar.
16. Deals and events extends towards other categories such as sports instead of only food.
17. MongoDB files can be branched out to 3rd normal forms to increase data integrity.
18. Have more photos for each POI such as photo slideshow to have a better representation of the place.
19. Sorting to allow more layer of sorting such as sort by rating, then by distance.
20. Have more sorting categories.
21. Current location as an option when picking starting location for sort by distance.
22. Choosing mode of transport can be option picking instead of typing in choice.
23. Allow users to save route for one POI to another POI on top of saving just POI.
24. Include support for other languages.
25. Possibly find ways to speed up the search process time.

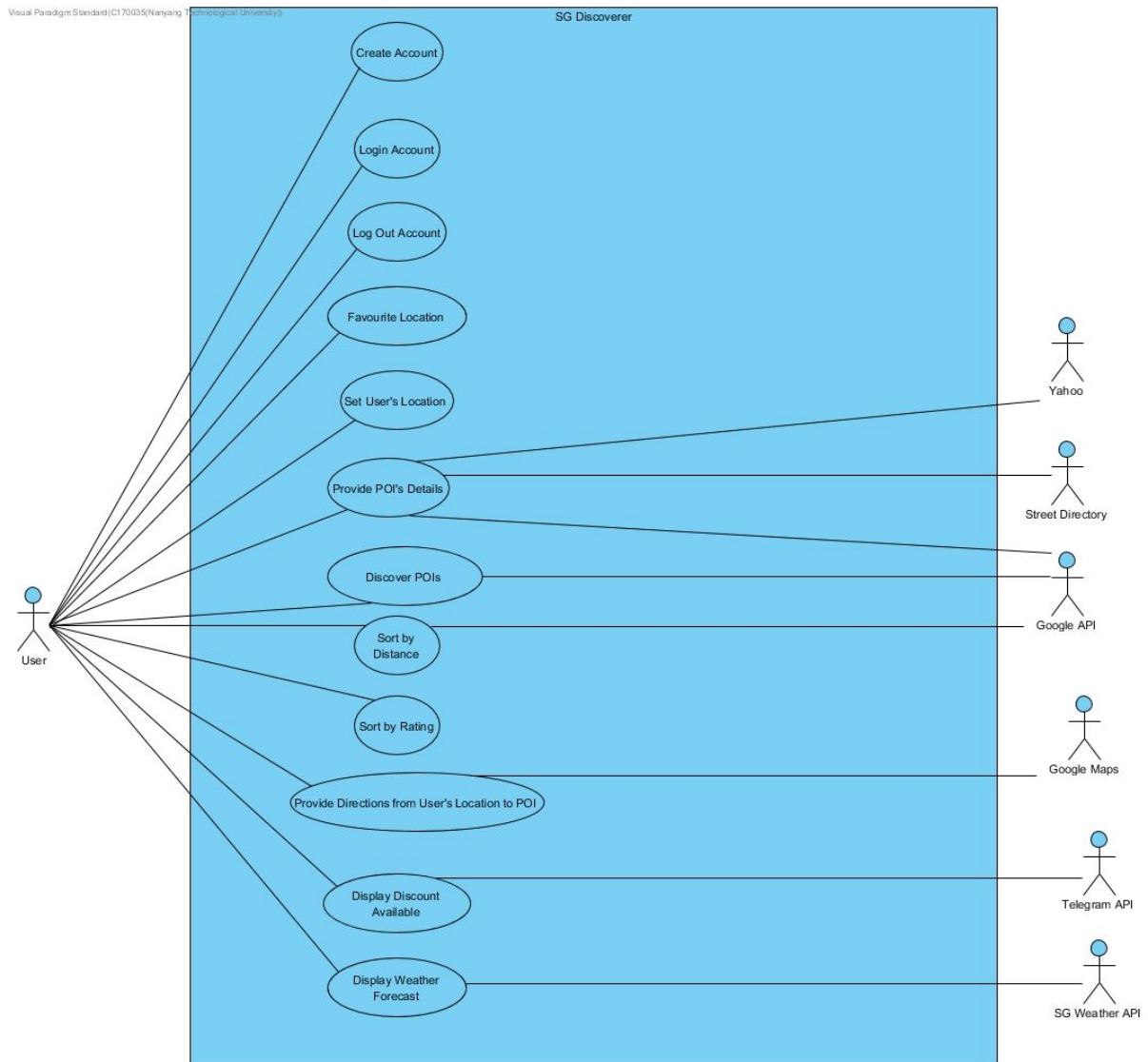
## 2. Functional Requirements

1. The system must have functionality for account management.
  - 1.1. The system must allow the user to register account.
    - 1.1.1. The system must display a form for new user to key in his personal data.
      - 1.1.1.1. The user must provide username in the form.
      - 1.1.1.2. The user must provide password in the form.
      - 1.1.1.3. The user must provide retyped password in the form.
    - 1.1.2. The system must ensure user accounts are unique.
      - 1.1.2.1. The system must ensure there are no duplicates of username in the database.
    - 1.1.3. The system must ensure that password matches retyped password.
    - 1.1.4. The system must be able to connect to database software to save all the personal data of the user.
  - 1.2. The system must confirm the user's identity when there is an attempt to log in.
    - 1.2.1. The system must prompt for username.
    - 1.2.2. The system must prompt for password.
    - 1.2.3. The system must verify username and password is correct.
  - 1.3. The system must allow the user to log out of his account.
  - 1.4. The system must allow the user to favourite locations
    - 1.4.1. The user must click on "♥" icon at the POI to favourite the location.
    - 1.4.2. The system must record down the favourite location and store it in the database.
    - 1.4.3. The user must be able to visit their favourite POIs by clicking on the 'Favourites' drop down bar location on the navigation bar.
    - 1.4.4. The system must allow the user to unfavourite their favourite location.
      - 1.4.4.1. The user must click on "♥" icon of a POI that they have already favorited before.
      - 1.4.4.2. The system removes favourite location from database.

2. The system must be able to determine the user's location whenever the system requires it
  - 2.1. The system must prompt the user to let the system know their location.
    - 2.1.1. The system must show the user's location on the map if he allows.
3. The system must allow the user to search for Point of Interests (POIs) based on their input choices.
  - 3.1. The system must allow the user to select the area of interest.
    - 3.1.1. The system must allow the user to select the district from a drop-down menu. (e.g. South West, Orchard)
      - 3.1.1.1. The system must produce a drop-down menu that contains all the districts in Singapore.
    - 3.1.2. The system must allow the user to set the area of interest through text input.
      - 3.1.2.1. The system must provide a text field for the user to key in a location.
    - 3.1.3. The system must allow the user to choose the search radius around the location selected.
      - 3.1.3.1. The system must determine Nearby POIs based on proximity the location, within a specified radius chosen by user (e.g. locations within 350m).
  - 3.2. The system must allow categorical search for POIs.
    - 3.2.1. The user must click on a specific category to search for POIs under the specified category.
    - 3.2.2. The system must categorise the POIs into different categories.(e.g. Restaurants, Shopping malls, etc.)
  - 3.3. The system must allow to user to search based on rating.
    - 3.3.1. The system must display POI with rating not less than the rating chose by the user. (eg. if the user chose rating of 4, only POI with rating 4 and above is displayed etc.)
  - 3.4. The user can choose not to select category, radius and rating for a generic search. The default would be 'food' category, '<400m' radius and '1-5 star' rating.
  - 3.5. The system must generate POIs based on the user input.
    - 3.5.1. The user must click "search" button to search for POIs.
    - 3.5.2. The system must display the POI pins on the map.
    - 3.5.3. The system must display the list of POIs.
4. The system must allow the user to select sorting method for the POIs.
  - 4.1. The system must allow the user to sort POIs via distance from a starting point chosen.
  - 4.2. The system must allow the user to sort POIs based on ratings.
    - 4.2.1. The system will sort the ratings in descending order.
5. The system must be able to display information about POI generated from the search.
  - 5.1. The system must provide information regarding Point of Interest (POI).
    - 5.1.1. The system must provide the name of the POI.
    - 5.1.2. The system must provide the description of the POI.
    - 5.1.3. The system must provide the rating of the POI.

- 5.1.4. The system must provide the distance from user's current location to the POI.
- 5.1.5. The system must provide the address of the POI.
- 5.1.6. The system must provide the category of the POI.
- 5.1.7. The system must provide the image of the POI.
  - 5.1.7.1. The system must be able to scrape relevant image from Yahoo.
  - 5.1.7.2. The system must be able to scrape relevant image from Street Directory.
- 5.2. The system must provide directions from user's location to POI .
  - 5.2.1. The system must contain information on user's current location.
    - 5.2.1.1. The system must prompt the user for his current location if there is no information on his location.
  - 5.2.2. The system must direct the user to google map for the directions.
    - 5.2.2.1. The system must pass the data on user's current location and POI to google map.
- 6. The system must display weather forecast.
  - 6.1. The system must get the weather forecast from the government api.
  - 6.2. The system must display the weather forecast.
  - 6.3. The system must allow user to toggle weather forecast detail on the map.
    - 6.3.1. The user must click on 'Show weather conditions' checkbox to toggle weather conditions on or off.
- 7. The system must have a discount feature.
  - 7.1. The system must get the promotions available from the telegram api.
  - 7.2. The system must display necessary information of promotions on discount card.
    - 7.2.1. The system must display address location of the place where the promotion is happening.
    - 7.2.2. The system must display the end date of the promotion.
    - 7.2.3. The system must show the discount details of the promotion.
  - 7.3. The system must show the particular discount location on the map after the user click on the card.
  - 7.4. The system must allow user to toggle discounts details on the map.
    - 7.4.1. The user must click on 'Show discounts' checkbox to toggle discount deals on or off.
    - 7.4.2. The user must click on popup icons to see details about the discount or weather forecast.

## 2.1 Use Case Diagram





## 2.2 Use Case Descriptions

### 2.2.1 Create Account

<b>Use Case ID:</b>	<b>UC 1</b>		
<b>Use Case Name:</b>	<b>Create Account</b>		
<b>Created By:</b>	<b>Chua Jia Ren</b>	<b>Last Updated By:</b>	<b>Chua Jia Ren</b>
<b>Date Created:</b>	<b>12<sup>th</sup> January 2019</b>	<b>Date Last Updated:</b>	<b>12<sup>th</sup> January 2019</b>

<b>Actor:</b>	<b>User (Initiating Actor)</b>
<b>Description:</b>	<b>User creates an account.</b>
<b>Pre-conditions:</b>	<b>1. User clicks on register account button.</b>
<b>Post-conditions:</b>	<b>1. System creates a new account for User.</b>
<b>Priority:</b>	<b>High</b>
<b>Frequency of Use:</b>	<b>Once</b>
<b>Flow of Events:</b>	<ol style="list-style-type: none"> <li><b>1. User selects the register account button.</b></li> <li><b>2. System displays a form asking new User to key in his personal data.</b></li> <li><b>3. User provides username, password, retyped password.</b></li> <li><b>4. System ensures no duplicates of username.</b></li> <li><b>5. System will connect to a database software to save credentials of User.</b></li> </ol>

<b>Alternative Flows:</b>	<p><b>AF-S4.1: If the username is taken</b></p> <ol style="list-style-type: none"> <li>1. User inputs an unavailable username.</li> <li>2. System displays message saying, "Username is taken".</li> <li>3. System returns to Step 3.</li> </ol> <p><b>AF-S5.1: If the username is empty</b></p> <ol style="list-style-type: none"> <li>1. User inputs an invalid Username.</li> <li>2. System displays message saying, "Please enter a username" and remind the user about the required format for username.</li> <li>3. System returns to Step 3.</li> </ol> <p><b>AF-S5.2: If the password is empty</b></p> <ol style="list-style-type: none"> <li>1. User inputs an invalid password.</li> <li>2. System displays message saying, "Please enter a password" and remind the user about the required format for password.</li> <li>3. System returns to Step 3.</li> </ol>
<b>Exceptions:</b>	-
<b>Includes:</b>	-
<b>Special Requirements:</b>	<p><b>NFR 1.2.5: The system must not leak any of the user's data.</b></p> <p><b>NFR 1.3.1: The system must respond to user's inputs within 5 seconds.</b></p> <p><b>NFR 1.3.2: The system is able to respond to at least 10000 requests at any given time.</b></p>
<b>Assumptions:</b>	-
<b>Notes and Issues:</b>	-

### 2.2.2 Login Account

<b>Use Case ID:</b>	UC 2		
<b>Use Case Name:</b>	Login Account		
<b>Created By:</b>	Chua Jia Ren	<b>Last Updated By:</b>	Chua Jia Ren
<b>Date Created:</b>	12 <sup>th</sup> January 2019	<b>Date Last Updated:</b>	12 <sup>th</sup> January 2019

<b>Actor:</b>	<b>User (Initiating Actor)</b>
<b>Description:</b>	<b>User logs in to account.</b>
<b>Pre-conditions:</b>	<b>1. User has an account.</b>
<b>Post-conditions:</b>	<b>1. System directs User to the home page.</b>
<b>Priority:</b>	<b>High</b>
<b>Frequency of Use:</b>	<b>0-20 times/day</b>
<b>Flow of Events:</b>	<b>1. System prompts User to provide username and password.</b> <b>2. User provides username and password.</b> <b>3. System verifies whether username and password are correct.</b> <b>4. System logs user into account.</b>
<b>Alternative Flows:</b>	<b>AF-S4: Incorrect username and password combination</b> <b>1. User provides an incorrect username/password.</b> <b>2. The System displays message saying, "Incorrect username/password".</b> <b>3. The System returns to Step 2.</b>
<b>Exceptions:</b>	-
<b>Includes:</b>	-
<b>Special Requirements:</b>	<b>NFR 1.3.1: The system must respond to user's inputs within 5 seconds.</b> <b>NFR 1.3.2: The system is able to respond to at least 10000 requests at any given time.</b>
<b>Assumptions:</b>	-
<b>Notes and Issues:</b>	-

### 2.2.3 Log Out Account

<b>Use Case ID:</b>	<b>UC 3</b>
<b>Use Case Name:</b>	<b>Log Out Account</b>

<b>Created By:</b>	<b>Chua Jia Ren</b>	<b>Last Updated By:</b>	<b>Chua Jia Ren</b>
<b>Date Created:</b>	<b>12<sup>th</sup> January 2019</b>	<b>Date Last Updated:</b>	<b>12<sup>th</sup> January 2019</b>

<b>Actor:</b>	<b>User (Initiating Actor)</b>
<b>Description:</b>	<b>User logs out of his account.</b>
<b>Pre-conditions:</b>	<b>1. User is logged in.</b>
<b>Post-conditions:</b>	<b>1. User is logged out of his account.</b>
<b>Priority:</b>	<b>Medium</b>
<b>Frequency of Use:</b>	<b>0-20 times/day</b>
<b>Flow of Events:</b>	<b>1. User clicks on the log out button.</b> <b>2. System disconnects User from the server.</b> <b>3. System displays dialog confirming User's log out.</b>
<b>Alternative Flows:</b>	-
<b>Exceptions:</b>	-
<b>Includes:</b>	-
<b>Special Requirements:</b>	<b>NFR 1.3.1: The system must respond to user's inputs within 5 seconds.</b> <b>NFR 1.3.2: The system is able to respond to at least 10000 requests at any given time.</b>
<b>Assumptions:</b>	-
<b>Notes and Issues:</b>	-

#### 2.2.4 Favourite location

<b>Use Case ID:</b>	<b>UC 4</b>		
<b>Use Case Name:</b>	<b>Favourite location</b>		
<b>Created By:</b>	<b>Chua Jia Ren</b>	<b>Last Updated By:</b>	<b>Chua Jia Ren</b>

<b>Date Created:</b>	<b>12<sup>th</sup> January 2019</b>	<b>Date Last Updated:</b>	<b>12<sup>th</sup> January 2019</b>
----------------------	-------------------------------------	---------------------------	-------------------------------------

<b>Actor:</b>	<b>User (Initiating Actor)</b>
<b>Description:</b>	<b>User logs out of his account.</b>
<b>Pre-conditions:</b>	<ol style="list-style-type: none"> <li><b>1. User is logged in.</b></li> <li><b>2. Locations must be favorited before if user wants to unfavourite it.</b></li> </ol>
<b>Post-conditions:</b>	<ol style="list-style-type: none"> <li><b>1. User has favourited location if wants to favourite it</b></li> <li><b>2. The system has removed the favourite location from database when the user wants to unfavourite it</b></li> </ol>
<b>Priority:</b>	<b>Medium</b>
<b>Frequency of Use:</b>	<b>0-20 times/day</b>
<b>Flow of Events:</b>	<ol style="list-style-type: none"> <li><b>1. The user must click on “♥” icon at the POI list if he wants to favourite a location.</b></li> <li><b>2. The system records down favourite location and stores it in the database.</b></li> <li><b>3. The user must be able to visit favourite POI through ‘Favourites’ drop down bar.</b></li> <li><b>4. The user must click on a highlighted “♥” icon at the POI list if he wants to unfavourite a location.</b></li> <li><b>5. The system removed the favourite location from database.</b></li> </ol>
<b>Alternative Flows:</b>	-
<b>Exceptions:</b>	-
<b>Includes:</b>	-
<b>Special Requirements:</b>	<b>NFR 1.3.1: The system must respond to user's inputs within 5 seconds.</b> <b>NFR 1.3.2: The system is able to respond to at least 10000 requests at any given time.</b>
<b>Assumptions:</b>	-
<b>Notes and Issues:</b>	-

### 2.2.5 Set User's location

<b>Use Case ID:</b>	UC 5		
<b>Use Case Name:</b>	Set User's Location		
<b>Created By:</b>	Kok Zi Ming	<b>Last Updated By:</b>	Ang Jun Liang
<b>Date Created:</b>	16 <sup>th</sup> January 2019	<b>Date Last Updated:</b>	17 <sup>th</sup> January 2019

<b>Actor:</b>	User (Participating actor)
<b>Description:</b>	System determines User's location.
<b>Pre-conditions:</b>	<ol style="list-style-type: none"> <li>1. Invocation as an included use case by use case UC3.2 (Text Input Area of Interest), or use case UC3.4 (Sort by Distance)</li> <li>2. User's location is not set.</li> </ol>
<b>Post-conditions:</b>	<ol style="list-style-type: none"> <li>1. System obtains information about User's current location.</li> <li>2. User's location is displayed on the map.</li> </ol>
<b>Priority:</b>	Medium
<b>Frequency of Use:</b>	0-5 times/day
<b>Flow of Events:</b>	<ol style="list-style-type: none"> <li>1. System prompts the User to allow the website to detect his current location.</li> <li>2. The system calls the script api to locate User location and returns his location coordinates.</li> <li>3. System displays User location on the map with marker, using his location coordinates.</li> </ol>
<b>Alternative Flows:</b>	<p>AF-S4: The address typed in by User does not yield any possible coordinate</p> <ol style="list-style-type: none"> <li>1. System displays the message below the search bar "Please enter a valid Singapore address".</li> <li>2. System returns to Step 2.</li> </ol>
<b>Exceptions:</b>	-
<b>Includes:</b>	-

<b>Special Requirements:</b>	<b>NFR 1.3.1: The system must respond to user's inputs within 5 seconds.</b> <b>NFR 1.3.2: The system is able to respond to at least 10000 requests at any given time.</b> <b>NFR 1.5.1: The system reflects user's location with accuracy of up to 50m.</b>
<b>Assumptions:</b>	-
<b>Notes and Issues:</b>	-

#### 2.2.6 Discover Point of Interests

<b>Use Case ID:</b>	<b>UC 6</b>		
<b>Use Case Name:</b>	<b>Discover Points of Interest</b>		
<b>Created By:</b>	<b>Yee Wei Min</b>	<b>Last Updated By:</b>	<b>Yee Wei Min</b>
<b>Date Created:</b>	<b>17<sup>th</sup> January 2019</b>	<b>Date Last Updated:</b>	<b>17<sup>th</sup> January 2019</b>

<b>Actor:</b>	<b>User (Initiating actor), Google API (Participating actor)</b>
<b>Description:</b>	<b>The user selects the area of interest and category that he wants to discover.</b>
<b>Pre-conditions:</b>	<b>1. The system requires the area of interest for other functions, or</b> <b>2. The user wants to change the area of interest.</b>
<b>Post-conditions:</b>	<b>1. The user has managed to successfully found and set his point of interest.</b>
<b>Priority:</b>	<b>High</b>
<b>Frequency of Use:</b>	<b>0-20 times/day</b>

<b>Flow of Events:</b>	<ol style="list-style-type: none"> <li>1. The user selects a district he wants to discover or text input the location he wants to discover.</li> <li>2. The user selects the search radius around the district.</li> <li>3. The user selects the category of the POI that he want to discover.</li> <li>4. The user selects the rating of the POI that he want to discover.</li> <li>5. The user clicks the search button.</li> <li>6. The system calls the google API to generate a list of POIs based on the user input.</li> <li>7. The system displays all the relevant information on the POIs generated.</li> </ol>
<b>Alternative Flows:</b>	
<b>Exceptions:</b>	-
<b>Includes:</b>	-
<b>Special Requirements:</b>	<p>NFR 1.3.1: The system must respond to user's inputs within 5 seconds.</p> <p>NFR 1.3.2: The system is able to respond to at least 10000 requests at any given time.</p>
<b>Assumptions:</b>	-
<b>Notes and Issues:</b>	-

### 2.2.7 Sort by Distance

<b>Use Case ID:</b>	UC 7		
<b>Use Case Name:</b>	Sort by Distance		
<b>Created By:</b>	Yee Wei Min	<b>Last Updated By:</b>	Yee Wei Min
<b>Date Created:</b>	17 <sup>th</sup> January 2019	<b>Date Last Updated:</b>	17 <sup>th</sup> January 2019

<b>Actor:</b>	User (Initiating actor), Google API (Participating actor)
<b>Description:</b>	The system sorts the POIs based on their distances from a starting point.



<b>Pre-conditions:</b>	1. The user clicks the sort by distance button.
<b>Post-conditions:</b>	1. The system displays the POIs based on distances from a starting point in ascending order.
<b>Priority:</b>	Medium
<b>Frequency of Use:</b>	0-20 times/day
<b>Flow of Events:</b>	<ol style="list-style-type: none"> <li>1. The user initiates this use case by clicking the sort by distance.</li> <li>2. The system sets the starting point using the included use case UC2.0 (Set User's location).</li> <li>3. The system calculates the distance from the starting point to each POI.</li> <li>4. The system sorts the POIs based on the distances calculated in ascending order.</li> <li>5. The system displays the POIs based on the distances calculated in ascending order.</li> </ol>
<b>Alternative Flows:</b>	-
<b>Exceptions:</b>	-
<b>Includes:</b>	UC 2.0 (Set User's location)
<b>Special Requirements:</b>	<p>NFR 1.3.1: The system must respond to user's inputs within 5 seconds.</p> <p>NFR 1.3.2: The system is able to respond to at least 10000 requests at any given time.</p> <p>NFR 1.5.2: The system reflects POI's location with accuracy of up to 50m.</p>
<b>Assumptions:</b>	-
<b>Notes and Issues:</b>	-

### 2.2.8 Sort by Ratings

<b>Use Case ID:</b>	UC 8		
<b>Use Case Name:</b>	Sort by Ratings		
<b>Created By:</b>	Yee Wei Min	<b>Last Updated By:</b>	Yee Wei Min
<b>Date Created:</b>	17 <sup>th</sup> January 2019	<b>Date Last Updated:</b>	17 <sup>th</sup> January 2019

<b>Actor:</b>	User (Initiating actor)
<b>Description:</b>	The system sorts the POIs based on their ratings in descending order.
<b>Pre-conditions:</b>	1. The user clicks the sort by rating button.
<b>Post-conditions:</b>	1. The system displays the POIs based on their ratings in descending order.
<b>Priority:</b>	Medium
<b>Frequency of Use:</b>	0-20 times/day
<b>Flow of Events:</b>	<ol style="list-style-type: none"> <li>1. The user clicks the sort by ratings button.</li> <li>2. The system provides the rating for each POI.</li> <li>3. The system sorts the POIs based on their ratings in descending order.</li> <li>4. The system displays the POIs based on their ratings in descending order.</li> </ol>
<b>Alternative Flows:</b>	-
<b>Exceptions:</b>	-
<b>Includes:</b>	-
<b>Special Requirements:</b>	<p>NFR 1.3.1: The system must respond to user's inputs within 5 seconds.</p> <p>NFR 1.3.2: The system is able to respond to at least 10000 requests at any given time.</p> <p>NFR 1.5.2: The system reflects POI's location with accuracy of up to 50m.</p>

<b>Assumptions:</b>	-
<b>Notes and Issues:</b>	-

### 2.2.9 Provide POI's Details

<b>Use Case ID:</b>	UC 9		
<b>Use Case Name:</b>	Provide POI's Details		
<b>Created By:</b>	Nigel Ang Wei Jun	<b>Last Updated By:</b>	Nigel Ang Wei Jun
<b>Date Created:</b>	16 <sup>th</sup> February 2019	<b>Date Last Updated:</b>	18 <sup>th</sup> February 2019

<b>Actor:</b>	User (Initiating actor), Google API (Participating actor), Yahoo (Participating actor), Street Directory (Participating actor)
<b>Description:</b>	System provides information regarding POI.
<b>Pre-conditions:</b>	<ol style="list-style-type: none"> <li>1. User must select a valid POI.</li> <li>2. User must be connected to the internet.</li> </ol>
<b>Post-conditions:</b>	<ol style="list-style-type: none"> <li>1. System displays information regarding POI, or</li> <li>2. User receives dialog explaining reason why the information could not be displayed (Connection error, no data available, etc.).</li> </ol>
<b>Priority:</b>	High
<b>Frequency of Use:</b>	0-20 times/day
<b>Flow of Events:</b>	<ol style="list-style-type: none"> <li>1. User clicks on any POI.</li> <li>2. User requests information regarding POI.</li> <li>3. System will retrieve information available regarding POI from Google, via Google API.</li> <li>4. System will retrieve relevant image by scraping from Yahoo.</li> <li>5. System will retrieve relevant image by scraping from Street Directory.</li> <li>6. System displays information retrieved on screen.</li> </ol>
<b>Alternative Flows:</b>	-

<b>Exceptions:</b>	<b>EX1: Data cannot be displayed</b> <ol style="list-style-type: none"> <li>1. System displays message “Data cannot be displayed. &lt;&lt;Reason for error&gt;&gt;”.</li> <li>2. System returns to before POI is selected.</li> </ol>
<b>Includes:</b>	-
<b>Special Requirements:</b>	<b>NFR 1.3.1: The system must respond to user's inputs within 5 seconds.</b> <b>NFR 1.3.2: The system is able to respond to at least 10000 requests at any given time.</b>
<b>Assumptions:</b>	-
<b>Notes and Issues:</b>	-

#### 2.2.10 Provide Directions from User’s Location to POI

<b>Use Case ID:</b>	<b>UC 10</b>		
<b>Use Case Name:</b>	<b>Provide Directions from User’s Location to POI</b>		
<b>Created By:</b>	<b>Nigel Ang Wei Jun</b>	<b>Last Updated By:</b>	<b>Nigel Ang Wei Jun</b>
<b>Date Created:</b>	<b>16<sup>th</sup> February 2019</b>	<b>Date Last Updated:</b>	<b>18<sup>th</sup> February 2019</b>

<b>Actor:</b>	<b>User (Initiating actor), Google Maps (Participating actor)</b>
<b>Description:</b>	<b>System provide directions from User’s current location to the POI.</b>
<b>Pre-conditions:</b>	<ol style="list-style-type: none"> <li>1. User must have entered a valid POI.</li> <li>2. User must be connected to the internet.</li> <li>3. User must have allowed application to know current location.</li> </ol>
<b>Post-conditions:</b>	<ol style="list-style-type: none"> <li>1. System is directed to Google Maps, or</li> <li>2. User receives dialog explaining reason why User could not be directed to Google Maps (Connection error, etc.).</li> </ol>
<b>Priority:</b>	<b>High</b>

<b>Frequency of Use:</b>	<b>0-20 times/day</b>
<b>Flow of Events:</b>	<ol style="list-style-type: none"> <li>1. User clicks on any POI.</li> <li>2. User clicks on 'get direction' POI marker.</li> <li>3. System will direct User to Google Maps at a new tab.</li> <li>4. System accesses Google Maps and sets up a new route in Google Maps, from User's location to the destination POI.</li> </ol>
<b>Alternative Flows:</b>	<b>AF-S3: User cannot be directed to Google Maps</b> <ol style="list-style-type: none"> <li>1. System displays message "Unable to access Google Maps. &lt;&lt;Reason for error&gt;&gt;"</li> <li>2. System returns to Step 3.</li> </ol>
<b>Exceptions:</b>	-
<b>Includes:</b>	-
<b>Special Requirements:</b>	<p><b>NFR 1.3.1:</b> The system must respond to user's inputs within 5 seconds.</p> <p><b>NFR 1.3.2:</b> The system is able to respond to at least 10000 requests at any given time.</p> <p><b>NFR 1.5.1:</b> The system reflects user's location with accuracy of up to 50m.</p> <p><b>NFR 1.5.2:</b> The system reflects POI's location with accuracy of up to 50m.</p> <p><b>NFR 1.5.4:</b> The system generates a path that is within the user's transport preference.</p> <p><b>NFR 1.6.1:</b> The system interacts with online mapping services.</p> <p><b>NFR 1.6.3:</b> The system exports data on user location to the interface without errors.</p> <p><b>NFR 1.6.4:</b> The system exports data on POI to the interface without errors.</p>
<b>Assumptions:</b>	-
<b>Notes and Issues:</b>	-

### 2.2.11 Weather forecast

<b>Use Case ID:</b>	UC 11		
<b>Use Case Name:</b>	Display Weather forecast		
<b>Created By:</b>	Chua Jia Ren	<b>Last Updated By:</b>	Chua Jia Ren
<b>Date Created:</b>	14 <sup>th</sup> April 2019	<b>Date Last Updated:</b>	14 <sup>th</sup> April 2019

<b>Actor:</b>	User (Initiating actor), SG Weather API (Participating actor)
<b>Description:</b>	System must display forecast of Singapore.
<b>Pre-conditions:</b>	<ol style="list-style-type: none"> <li>1. User must be connected to the internet.</li> </ol>
<b>Post-conditions:</b>	<p>Case 1 Unticked weather forecast checkbox:</p> <ol style="list-style-type: none"> <li>1. System displays weather forecast markers on map.</li> </ol> <p>Case 2 Ticked weather forecast checkbox:</p> <ol style="list-style-type: none"> <li>2. System removes weather forecast markers on map.</li> </ol>
<b>Priority:</b>	High
<b>Frequency of Use:</b>	0-20 times/day
<b>Flow of Events:</b>	<p>Case 1: Unticked weather condition checkbox.</p> <ol style="list-style-type: none"> <li>1. The system must get weather forecast from the government API.</li> <li>2. The user clicks on 'Show weather condition' checkbox.</li> <li>3. The system displays weather forecast markers on map.</li> <li>4. The system must display weather condition of the marker's area when user clicks on the it.</li> </ol> <p>Case 2: Ticked weather condition checkbox.</p> <ol style="list-style-type: none"> <li>1. The system must get weather forecast from the government API.</li> <li>2. The user clicks on 'Show weather condition' checkbox.</li> <li>3. The system hides the weather forecast markers on map.</li> </ol>

<b>Alternative Flows:</b>	
<b>Exceptions:</b>	-
<b>Includes:</b>	-
<b>Special Requirements:</b>	<b>NFR 1.3.1: The system must respond to user's inputs within 5 seconds.</b> <b>NFR 1.3.2: The system is able to respond to at least 10000 requests at any given time.</b> <b>NFR 1.5.1: The system reflects user's location with accuracy of up to 50m.</b> <b>NFR 1.6.1: The system interacts with online mapping services.</b> <b>NFR 1.6.3: The system exports data on user location to the interface without errors.</b>
<b>Assumptions:</b>	-
<b>Notes and Issues:</b>	-

### 2.2.12 Discounts feature

<b>Use Case ID:</b>	UC 12		
<b>Use Case Name:</b>	Display Discounts Available		
<b>Created By:</b>	Kok Zi Ming	<b>Last Updated By:</b>	Kok Zi Ming
<b>Date Created:</b>	14 <sup>th</sup> April 2019	<b>Date Last Updated:</b>	14 <sup>th</sup> April 2019

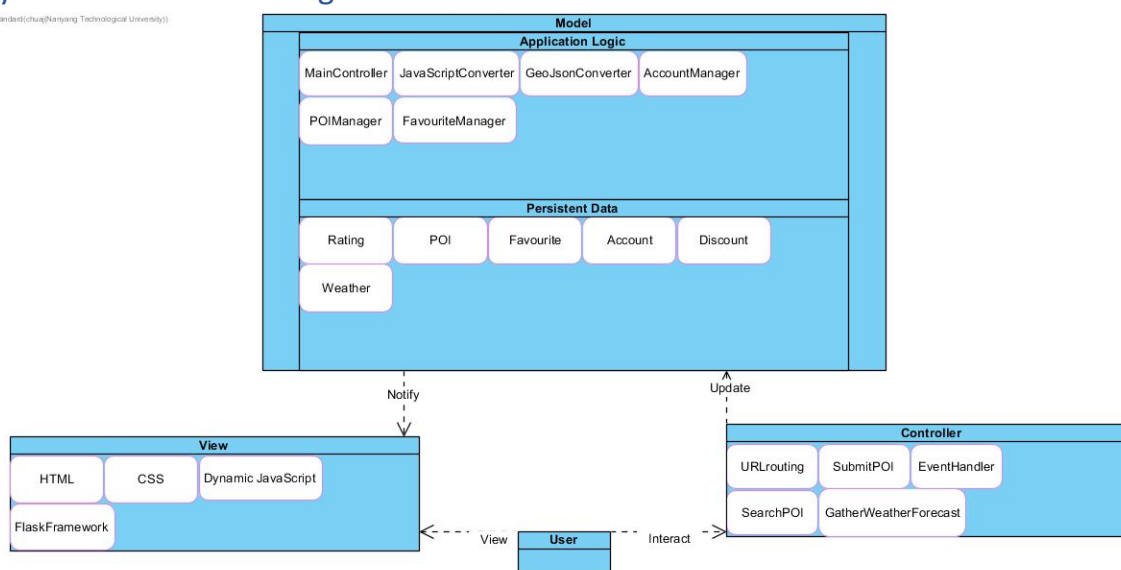
<b>Actor:</b>	User (Initiating actor), Telegram API (Participating actor)
<b>Description:</b>	User uses Discounts feature to generate list of promotions
<b>Pre-condition:</b>	1. User click on deals at the navigation bar on the website
<b>Post-conditions:</b>	1. The system will display a list of necessary information of promotion
<b>Priority:</b>	Medium

<b>Frequency of Use:</b>	<b>0-5 times/day</b>
<b>Flow of Events:</b>	<ol style="list-style-type: none"> <li>1. User clicks on deals on the navigation bar</li> <li>2. The system must display information of promotions on discount card</li> <li>3. The system must display the address location of the place where the promotion is happening</li> <li>4. The system must display the end date of the promotion</li> <li>5. The system must display the discount details of the promotion</li> <li>6. The user must click on the discount card</li> <li>7. The system must show the particular discount location on the map</li> <li>8. The system must allow user to toggle discount deals by clicking on 'Show discounts' checkbox</li> <li>9. The user must click on popup icons to see details about the discount</li> </ol>
<b>Alternative Flows:</b>	-
<b>Exceptions:</b>	-
<b>Includes:</b>	-
<b>Special Requirements:</b>	<b>NFR 1.3.1: The system must respond to user's inputs within 5 seconds.</b>
<b>Assumptions:</b>	-
<b>Notes and Issues:</b>	



## 2.3 System Architecture diagram

Visual Paradigm Standard (chug(Nanyang Technological University))

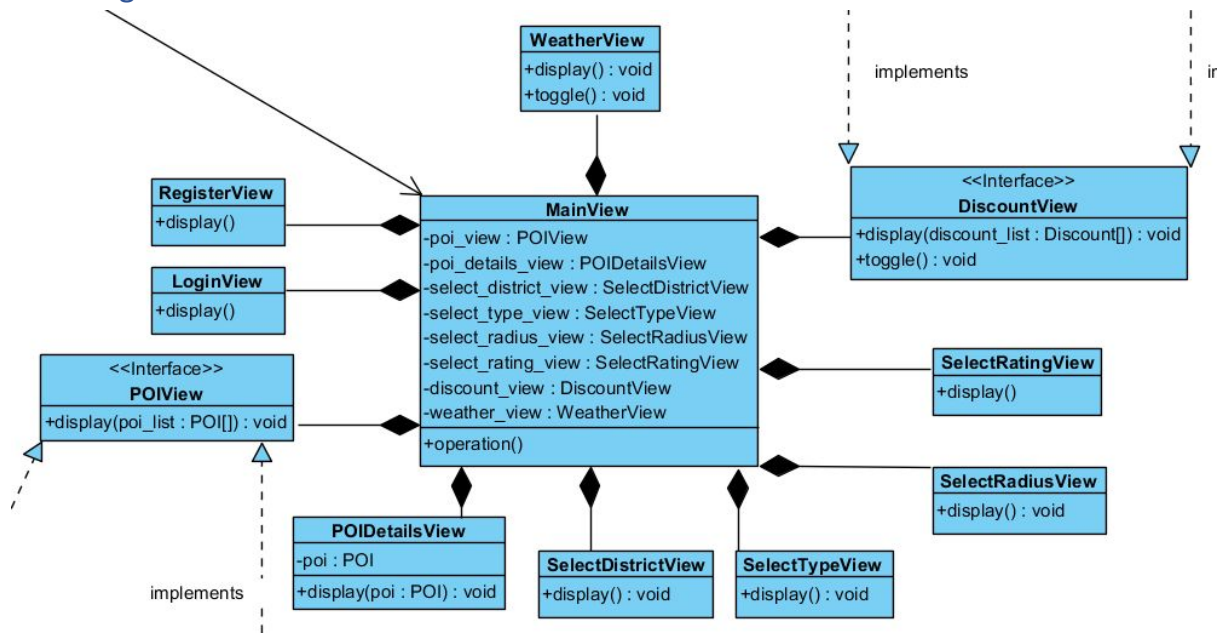


The System Architecture used is Model-View-Controller architecture (MVC). MVC architecture is most suited for applications where the same data can be presented differently. The same presentation can have different look and feel. This is applicable to our application because we have 2 ways of viewing searched locations - Map View of markers and List View of cards. They use the same data, but can react to user input differently. For example, we can favourite a location using List View but not Map View. Both Views must be able to reflect data changes immediately.

A layered architecture does not allow for upcalls from the control layers to presentation layers. However, our application requires the controller classes to regularly update the Views, hence we favor MVC architecture over layered architecture. MVC uses the observer pattern which allows Views to be notified of changes, without being tightly coupled to the control classes.

Being a single page application with intensive dynamic user interaction, the MVC architecture will be able to facilitate and optimize the implementation of the system. The separation of the Model from View and Controller components allows for multiple Views of the same Model - It separates presentation and interaction from the data. If the user changes the Model via the Controller of one View, all other Views dependent on this data should reflect the changes. For example, when user favourite a location by clicking on the “♥” icon on a card, the navigation drop-down bar for favourites will immediately show the favourite, even without refreshing the page.

## 2.4 Design Patterns



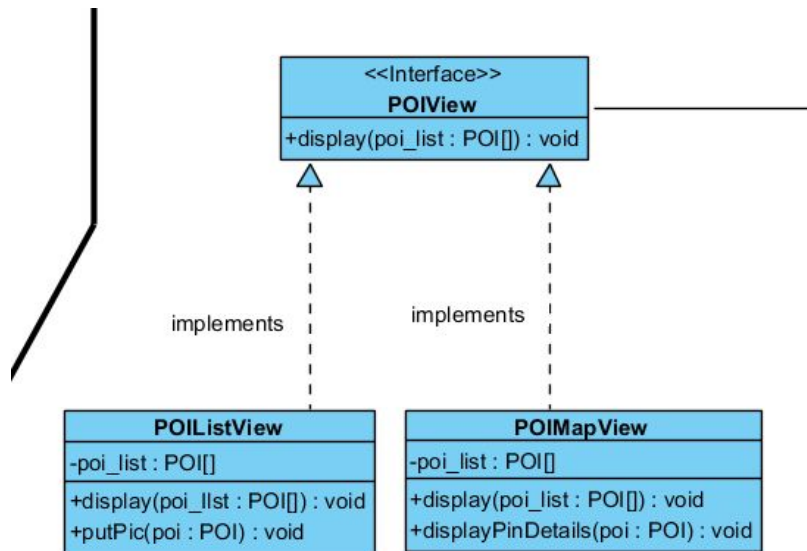
### Problem

There are many views, resulting in tight coupling between DiscovererManager and the views. There exist many links between DiscovererManager and the views.

### Solution: Facade Pattern

DiscovererManager needs to update several Views in the application. The Facade Pattern is a software design pattern that provides a simplified interface to a larger body of code. The Facade design pattern is often used when a system is very complex or difficult to understand because the system has a large number of interdependent classes which is true in our case (we have many views).

The pattern hides the complexities of the larger system (Views) and provides a simpler interface to the client. It typically involves a single wrapper class which contains a set of members (Views) required by client (DiscovererManager). These members access the system on behalf of the facade client and hide the implementation details.



### Problem

A set of algorithms or objects should be interchangeable.

Different methods of viewing the POIs should be interchangeable at runtime and transparent from other parts of the SGDiscoverer system.

Future methods of viewing the POIs should be able to be added to the system with minimal impacts to other parts of the system.

### Solution: Strategy Pattern

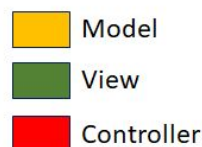
Context/Client is DiscovererManager. Strategy is POIView. Concrete strategies are POIListView and POIMapView.

Strategy Pattern defines a family of algorithms (methods of viewing POIs), encapsulates each one, and makes them interchangeable.

Strategy lets the algorithm vary independently from clients that use it, making it easier to switch, understand, and extend (e.g. adding new algorithms in the future).

Strategy Pattern simplifies Client classes by relieving them of any responsibility for selecting behaviour or implementing alternate behaviours.

The Context is not aware of the strategy implementation.



## 2.5 Class Diagrams

### Conceptual Class Diagram

```

graph TD
    MainUI((MainUI)) -- "<<use>>" --> DiscovererManager((DiscovererManager))
    DiscovererManager -- "<<use>>" --> AccountManager((AccountManager))
    DiscovererManager -- "<<use>>" --> FavouriteManager((FavouriteManager))
    DiscovererManager -- "<<use>>" --> SearchManager((SearchManager))
    DiscovererManager -- "<<use>>" --> CurrentLocationManager((CurrentLocationManager))
    DiscovererManager -- "<<use>>" --> POIManager((POIManager))
    DiscovererManager -- "<<use>>" --> WeatherManager((WeatherManager))
    DiscovererManager -- "<<use>>" --> DiscountManager((DiscountManager))
    AccountManager -- "<<read>>" --> AccountDatabase[AccountDatabase]
    FavouriteManager -- "<<read>>" --> FavouriteDatabase[FavouriteDatabase]
    DiscountManager -- "<<read>>" --> DiscountDatabase[DiscountDatabase]

```

The image displays three UML diagrams for a mobile application, categorized by a legend: Model (yellow), View (green), and Controller (red).

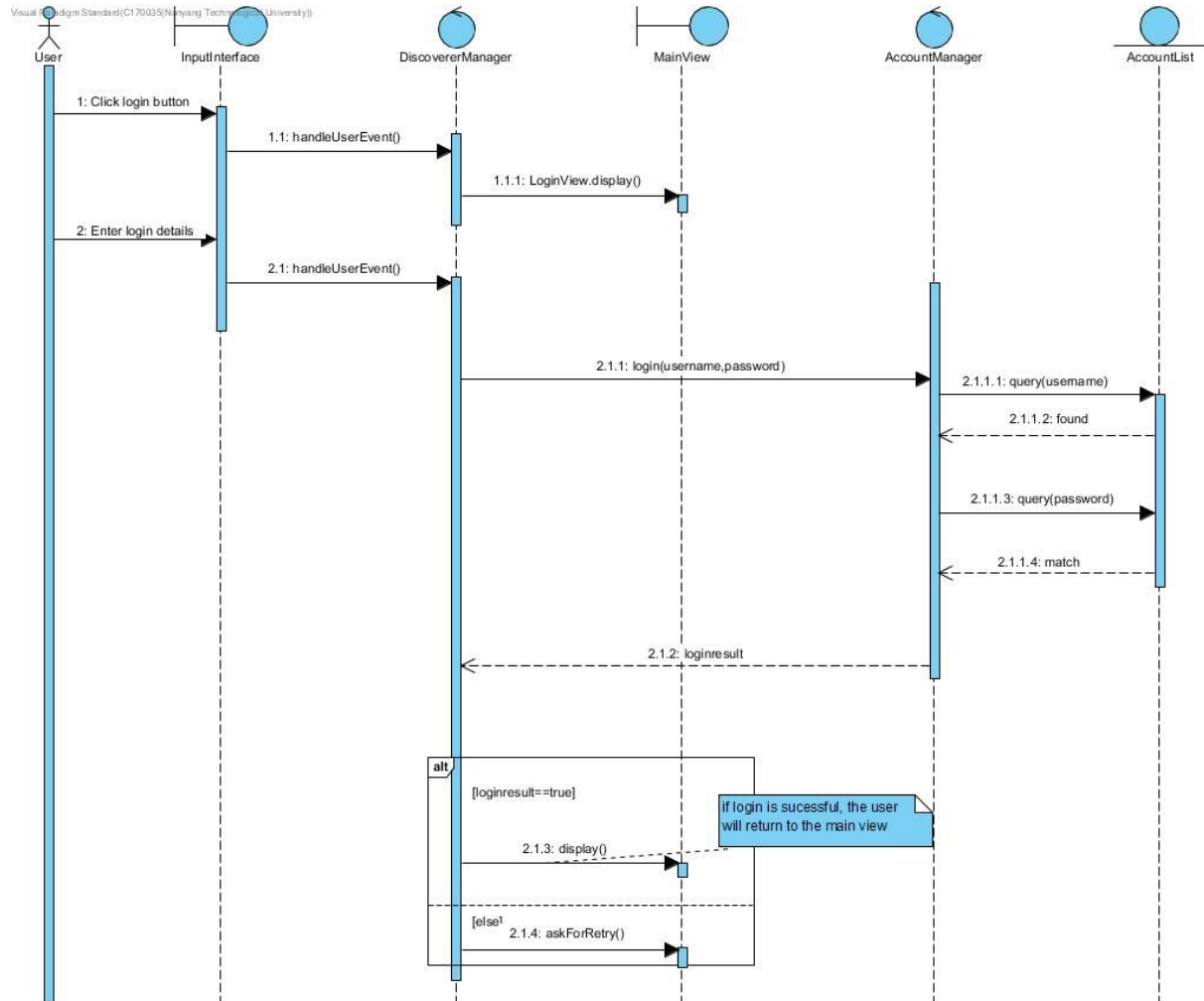
- Controller Layer (Red):**
  - InputInterface**: An interface with a method `getInput() void`.
  - DiscovererManager**: Implements `InputInterface`. It has methods `mainView: MainView`, `map() void`, and `handleUserEvent() void`. It is associated with `WeatherManager`, `DiscountManager`, `SearchManager`, `CoordinateLocationManager`, `AccountManager`, and `DataManagerInterface`.
- Model Layer (Yellow):**
  - WeatherManager**: `generateWeatherData()`
  - DiscountManager**: `discount_list: Discount[]`, `generateDiscounts() Discount[]`
  - SearchManager**: `poi_list: POI[]`, `generatePOIAndSelect(String type, String distance, String rating, String) POI[]`, `generateReferencePoint() Coordinates`, `generateMarkers(poi_list: POI[], Marker[])`, `generateCard(poi_list: POI[], Card[])`
  - CoordinateLocationManager**: `current_coordinates: Coordinates`, `nextCurrentCoordinates() Coordinates`
  - AccountManager**: `account_list: Account[]`, `createAccount() Account` void, `deleteAccount() Account` void, `loginAccount(userName: String, password: String) void`, `loginAccount()` void
  - DataManagerInterface**: `insert() void`, `delete() void`, `query() void`
  - MySQL**: `insert() void`, `delete() void`, `query() void`
  - SQLite**: `insert() void`, `delete() void`, `query() void`
  - FavouriteList**: `favourites: List<Favourite>`
  - AccountList**: `accounts: List<Account>`
  - DiscountList**: `discount: List<Discount>`
  - Favourite**: `place_id: String`, `account_name: String`, `address: String`, `place_name: String`, `point_code: String`
  - Account**: `id: String`, `account_name: String`, `address: String`, `date_of_birth: boolean`, `date_of_birth: String`, `email: String`, `password: String`, `real_name: String`, `security_answer: String`, `security_question: String`
  - Discount**: `discount_name: String`, `place_name: String`, `discount_description: String`, `end_date: String`, `address: String`
- View Layer (Green):**
  - DiscountMapView**: `discount_list: Discount[]`, `display(discount_list: Discount[]) void`, `toggle() void`
  - DiscountCardView**: `discount_list: Discount[]`, `display(discount_list: Discount[]) void`, `toggle() void`
  - RegisterView**: `display() void`
  - LoginView**: `display() void`
  - WeatherView**: `display() void`, `toggle() void`
  - POIDetailsView**: `poi: POI`, `display(poi: POI) void`
  - SelectDistrictView**: `display() void`
  - SelectTypeView**: `display() void`
  - SelectRatingView**: `display() void`
  - SelectRadiusView**: `display() void`
  - POIMapView**: `poi_list: POI[]`, `display(poi_list: POI[]) void`, `displayDetails(poi: POI) void`
  - POIMapView**: `poi_list: POI[]`, `display(poi_list: POI[]) void`, `displayDetails(poi: POI) void`
  - MainView**: `poi_view: POIView`, `poi_details_view: POIDetailsView`, `select_district_view: SelectDistrictView`, `select_type_view: SelectTypeView`, `select_rating_view: SelectRatingView`, `select_radius_view: SelectRadiusView`, `discount_view: DiscountView`, `weather_view: WeatherView`, `register() void`

**Legend:**

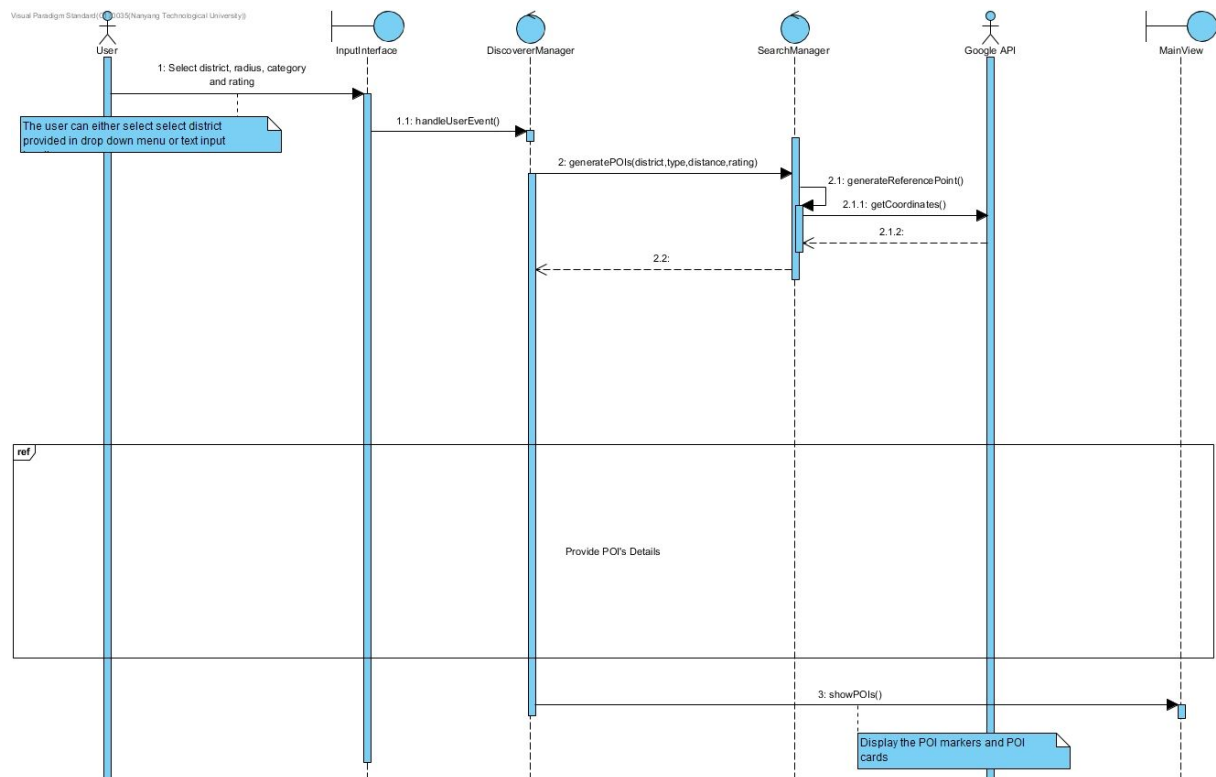
- Model
- View
- Controller

## 2.6 Sequence Diagrams

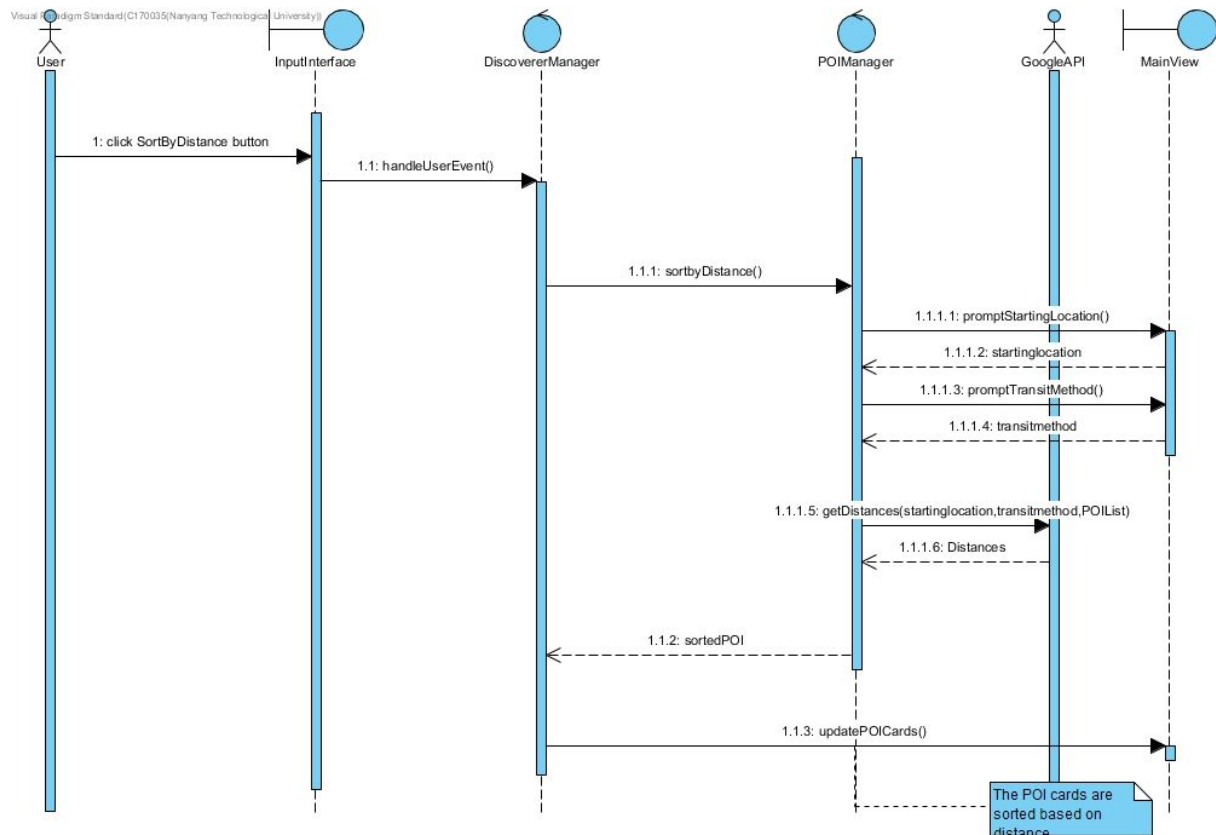
### 2.6.1 Login Account



## 2.6.2 Discover POIs

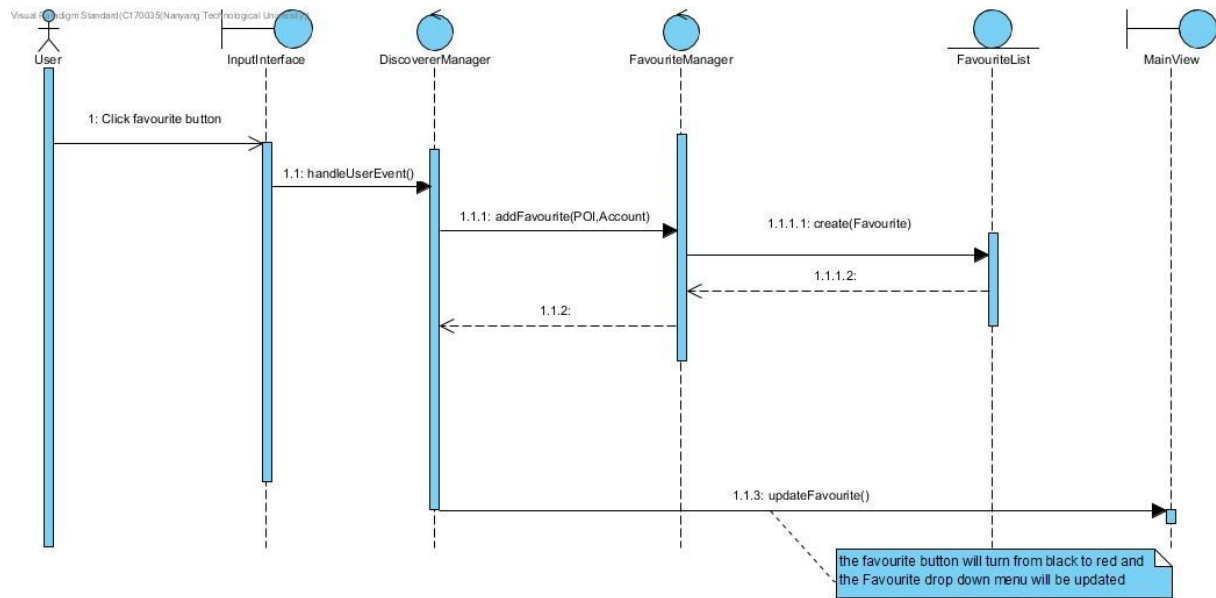


## 2.6.3 Sort by Distance

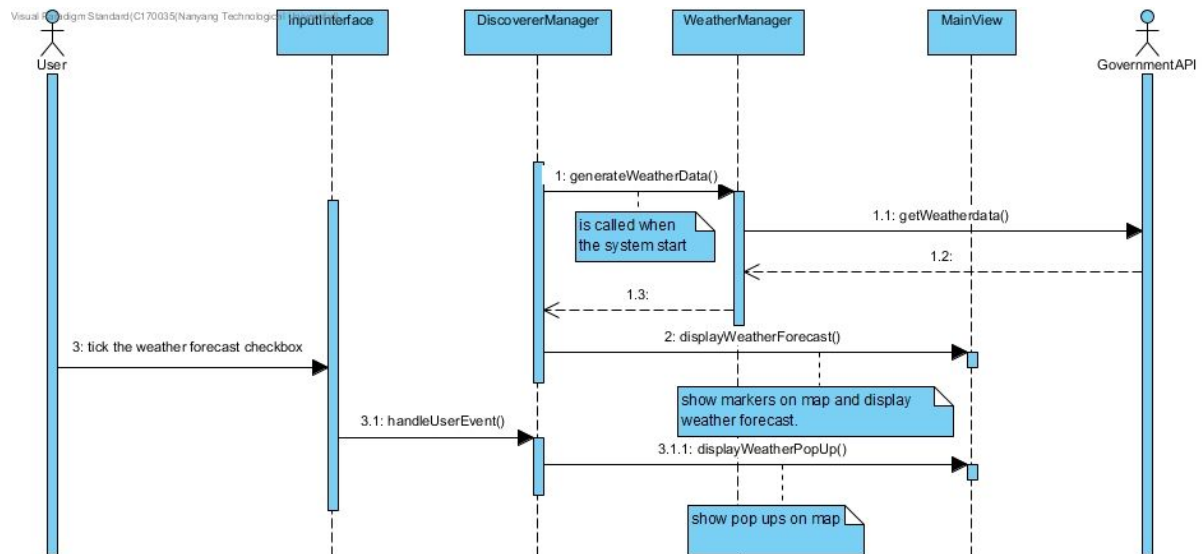




## 2.6.4 Favourite Location

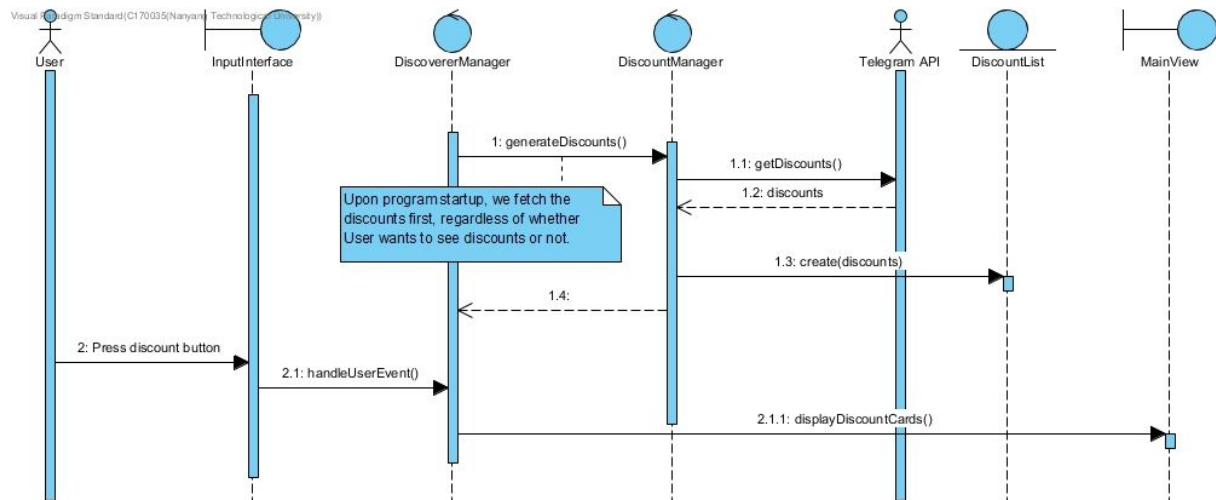


## 2.6.5 Display Weather Forecast



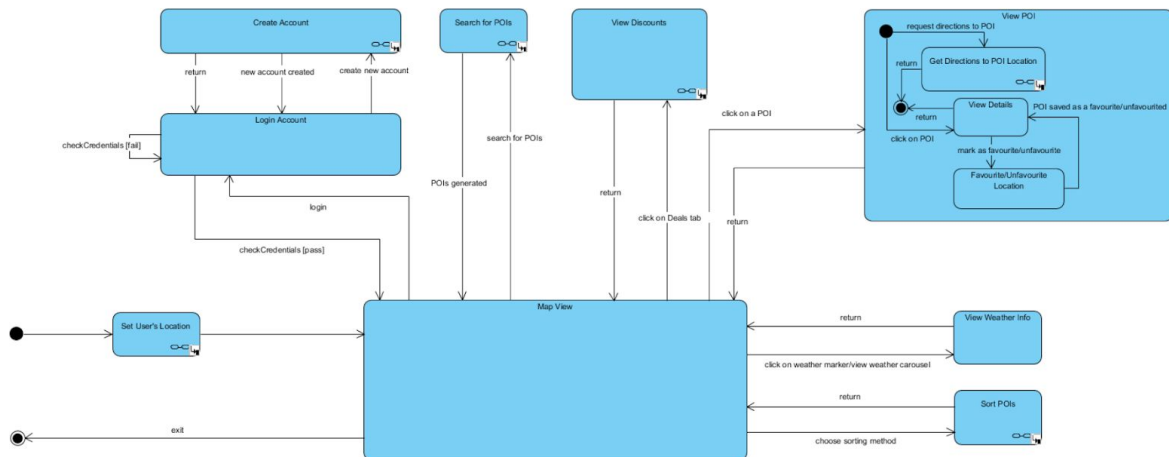


### 2.6.6 Display Discount Available

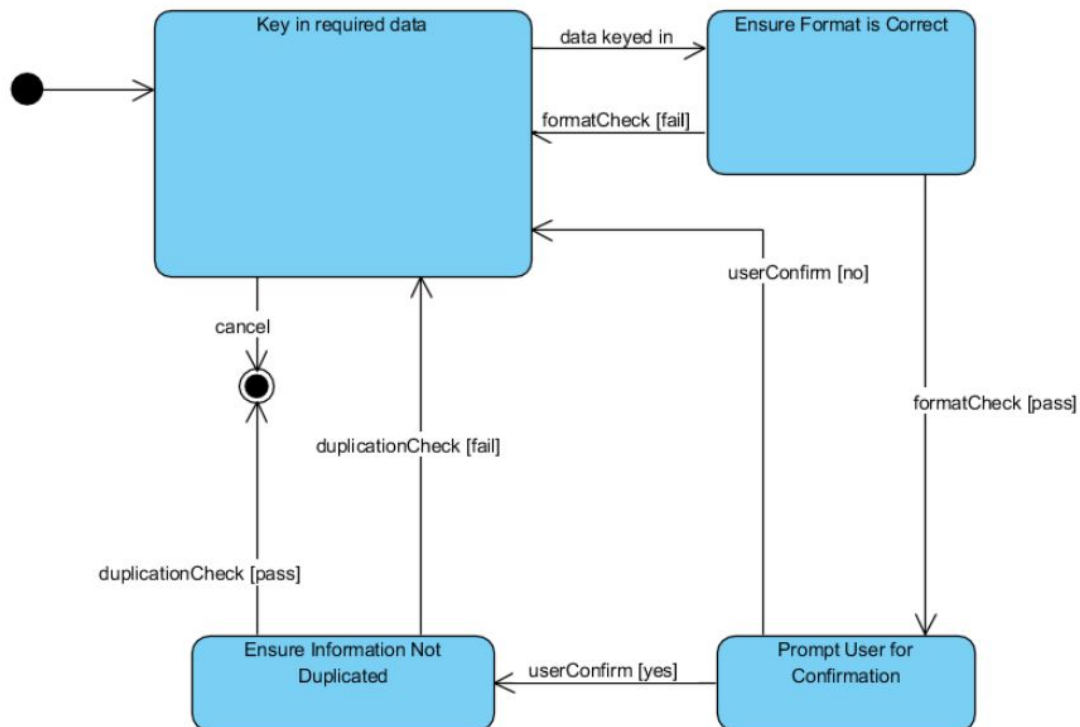


## 2.7 State transition diagram

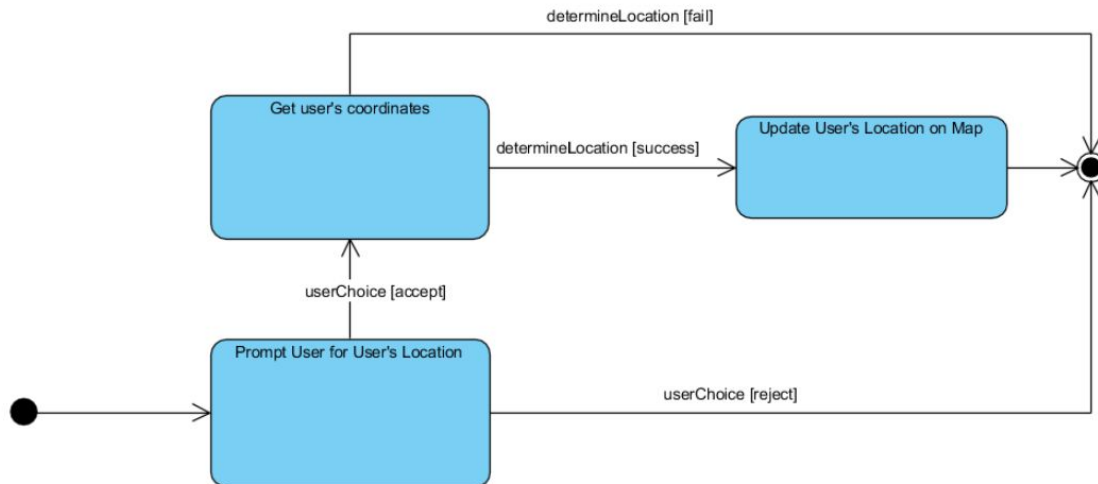
### 2.7.1 SG Discoverer



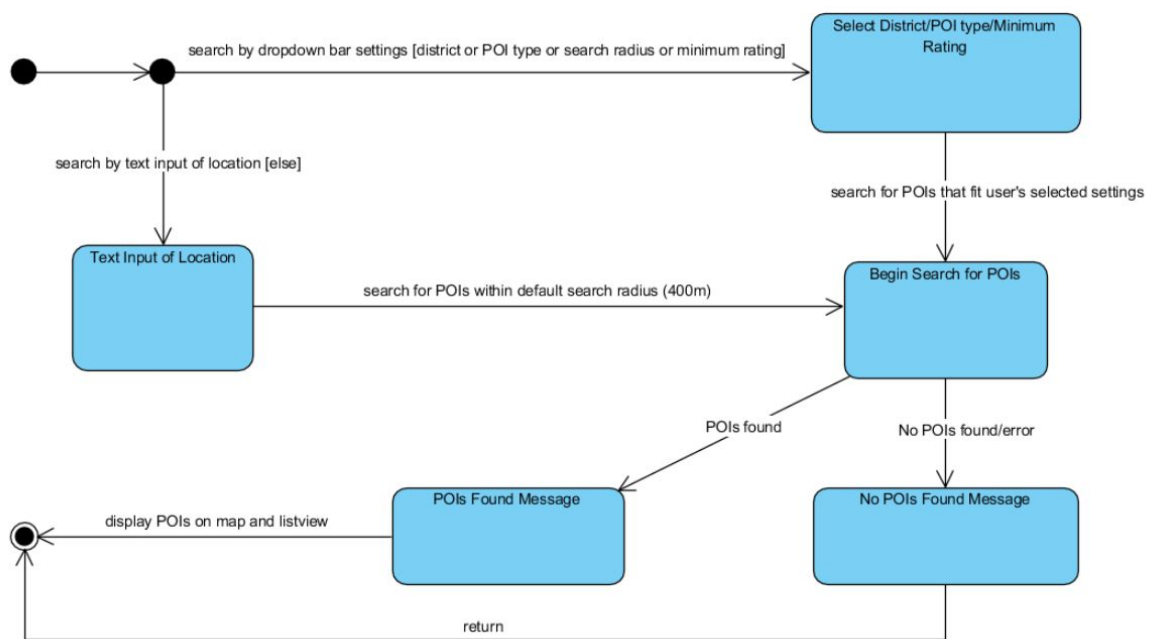
### 2.7.2 Create account



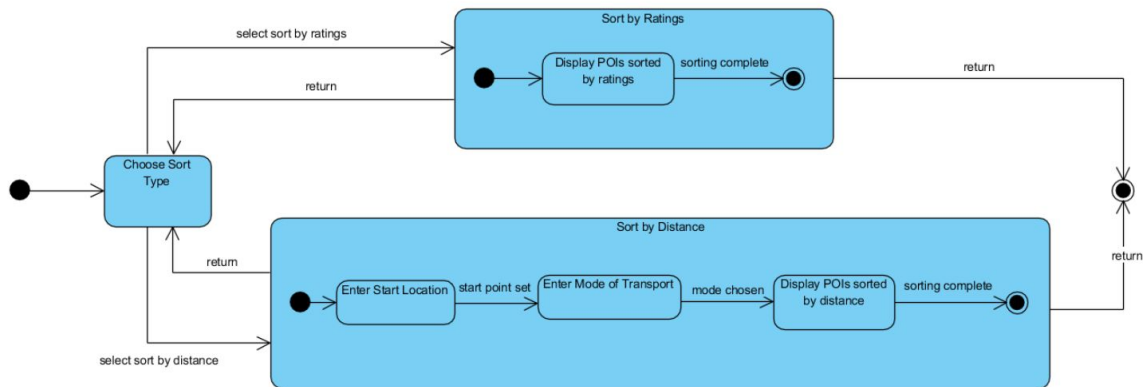
### 2.7.3 Set User Location



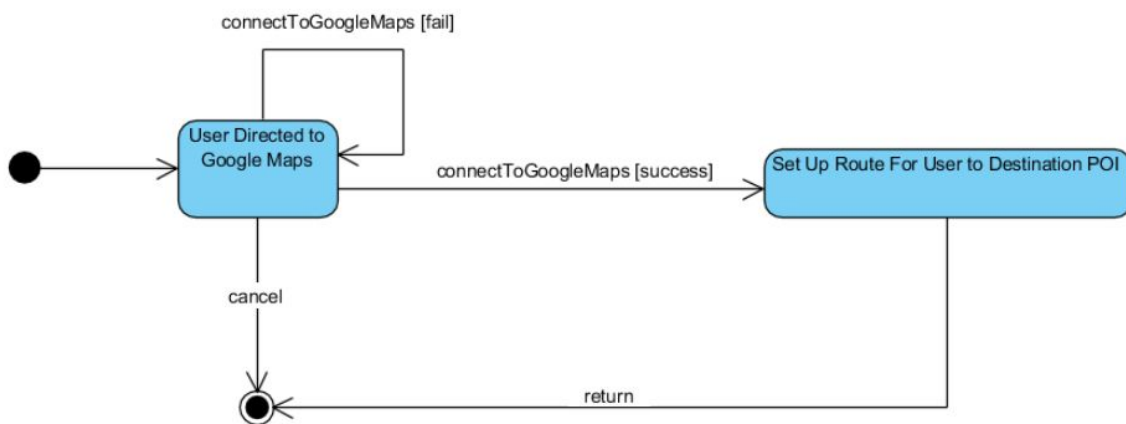
### 2.7.4 SearchforPOIs



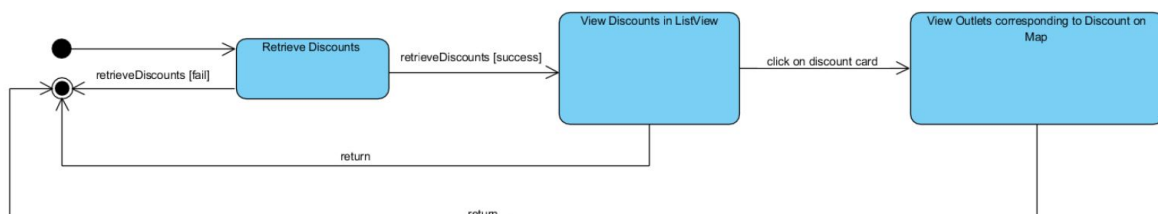
### 2.7.5 Sort POIs



### 2.7.6 GetDirectionsToPOILocation



### 2.7.7 ViewDiscounts



## 2.8 Website Screenshot

### 2.8.1 Register page



**Username :**

**Password :**

**Retype Password :**

### 2.8.2 Login page

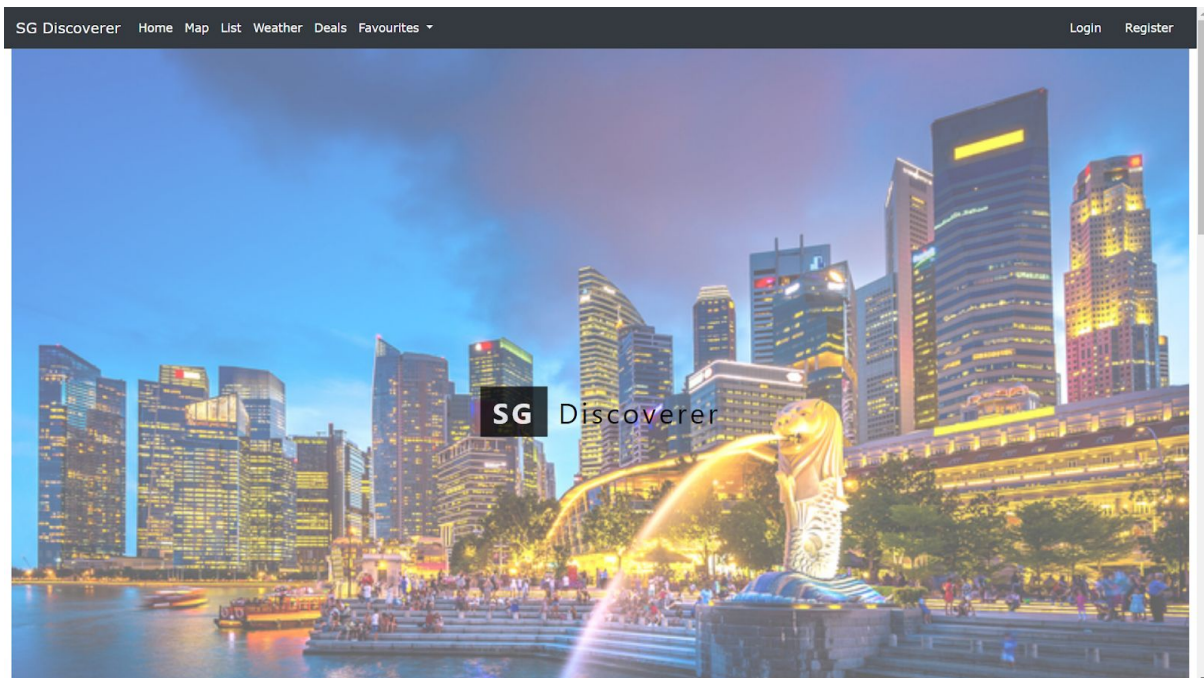


**Username :**

**Password :**

Login

### 2.8.3 Home page



## 2.8.4 Map view

SG Discoverer
Home
Map
List
Weather
Deals
Favourites
Login
Register

Discover the places you can go in Singapore

Don't know where to go? Enter your desired location and hit search to discover new places to explore.

Enter a place to discover or choose a district

Select District

Select Type

Search Radius/m

1-5

Search

Current position loaded.

The map shows Singapore with numerous red location pins and blue weather icons. Labels include BUKIT INDAH, DANGA BAY, JOHNSON, MASAL, PASIR GUDANG, PULAU BIN, TEKONG, PENERANG, MASANGI, SINGAPORE, JURONG ISLAND, BUKOM ISLAND, and SINGAPORE STRAIT. The Google logo is in the bottom left.

☒ Show weather conditions
☒ Show discounts

## 2.8.5 Satellite view

SG Discoverer
Home
Map
List
Weather
Deals
Favourites
Logout

Discover the places you can go in Singapore

Don't know where to go? Enter your desired location and hit search to discover new places to explore.

Enter a place to discover or choose a district

Select District

Select Type

Search Radius/m

1-5

Search

Current position loaded.

The satellite map shows detailed terrain of Singapore. Labels include JURONG EAST, Ngee Ann Polytechnic, BUKIT TIMAH, Jacob Ballas Children's Garden, TOA PAYOH, NOVENA, City Square Mall, KALLANG, Golden Mile Complex, Suntec City, Singapore Flyer, Gardens by the Bay, Marina Barrage, DOWNTOWN CORE, BUKIT MERAH, HORT PARK, KENT RIDGE PARK, HAW PAR VILLA, QUEENSTOWN, National University of Singapore, Singapore Polytechnic, CLEMENTI, PANDAN GARDENS, and WEST COAST PIER. The Google logo is in the bottom left.

☒ Show weather conditions
☒ Show discounts



## 2.8.6 Map view after searching (Marina Square)

SG Discoverer
Home
Map
List
Weather
Deals
Favourites
Login
Register

Discover the places you can go in Singapore

Don't know where to go? Enter your desired location and hit search to discover new places to explore.

Marina Square
Select Type
Search Radius/m
1-5

Search

Locations found!

## 2.8.7 POI List after sort by rating (Marina Square)

SG Discoverer
Home
Map
List
Weather
Deals
Favourites
Login
Register

List of POIs

Find the list of places and their details here after applying the search


Sort By Distance
Sort By Rating

<b>KOI Café</b> 4.8 1 Raffles Link, #B1-66, CityLink, Singapore 039393 Type: Cafe	<b>Gajalee</b> 4.7 8 Raffles Ave, Esplanade Mall, Singapore 039802 Type: Restaurant	<b>Bay@5</b> 4.6 5 Raffles Ave, Singapore 039797 Type: Bar	<b>Dolce Vita</b> 4.5 5 Raffles Ave, Singapore 039797 Type: Restaurant
<b>Axis Bar and Lounge</b> 4.4 5 Raffles Ave, Singapore	<b>Pacific Marketplace</b> 4.4 Level 1, Pan Pacific, 7 Raffles	<b>Cherry Garden</b> 4.4 5 Raffles Ave, Singapore	<b>Quaich Bar</b> 4.4 28 Beach Rd, Singapore



## 2.8.8 Weather forecast

SG Discoverer
Home
Map
List
Weather
Deals
Favourites
Login
Register



### Hourly Weather Forecast

Don't forget to check the weather before you go.

Choa Chu Kang: Showers	Clementi: Showers	City: Fair & Warm	Geylang: Fair & Warm	Hougang: Fair & Warm
Jalan Bahar: Thundery Showers	Jurong East: Showers	Jurong Island: Thundery Showers	Jurong West: Showers	Kallang: Fair & Warm

Show on map

© 2019 SG Discoverer

## 2.8.9 Favourites

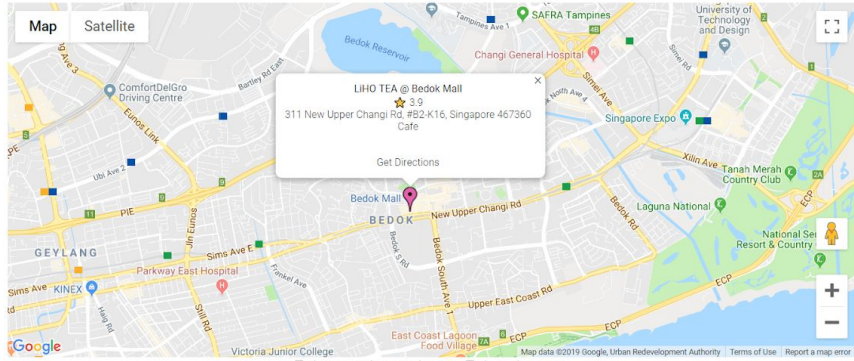
SG Discoverer
Home
Map
List
Weather
Deals
Favourites
Logout

LIHO TEA @ Bedok Mall  
Koi Café  
Block 718a HDB Woodlands (MSCP)

### Discover the place you love

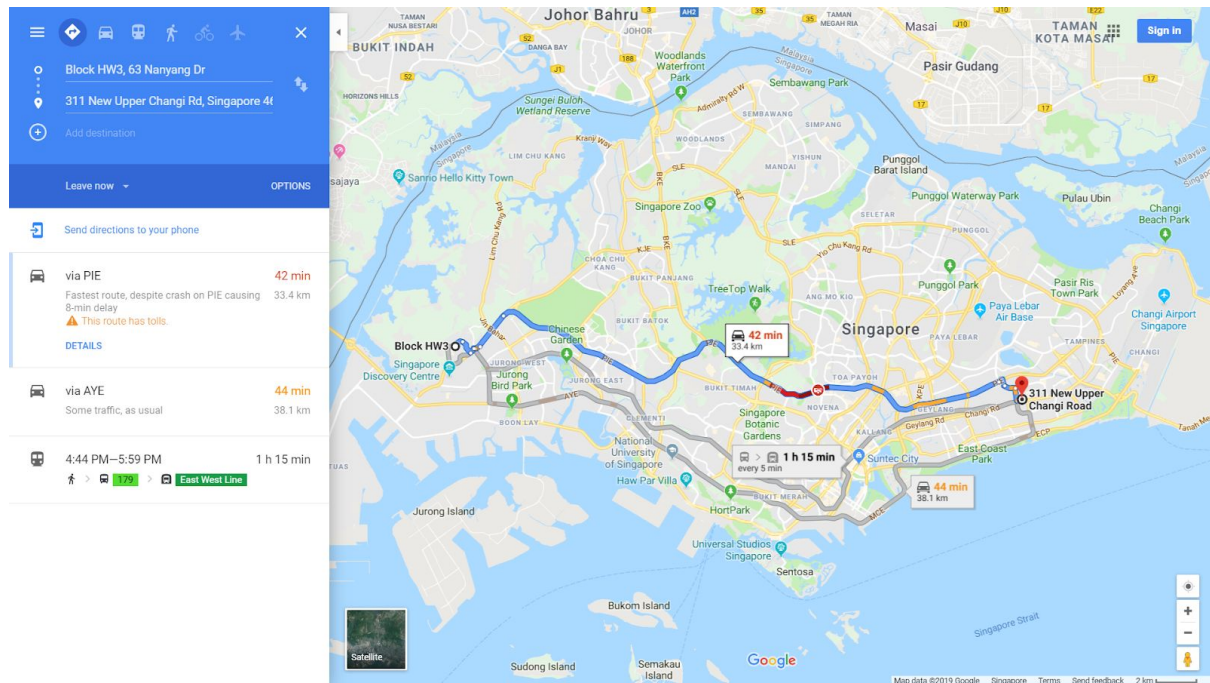
Don't know where to go? Enter your desired location and hit search to discover new places to explore.

Current position loaded.



☐ Show weather conditions
☐ Show discounts

## 2.8.10 Get directions



## 2.8.11 Deals page

SG Discoverer
Home
Map
List
Weather
Deals
Favourites
Login
Register

### List of Discounts

Find the list of coupons and discounts here

<b>Gloria Jean's 1-For-1 Regular Drinks</b> Marina Square #03-132 Ends Tue, 31 Dec 2019	<b>Wheat 1-For-1 Salmon Poke Bowl</b> 313 Somerset #B3-46 Ends Tue, 30 Apr 2019	<b>Dunkin Donuts x Fave \$1 for 2 Donuts + 1 Iced Coffee</b> All Outlets Ends Tue, 30 Apr 2019	<b>Kakuiida's 1-For-1 Soft Serve</b> 12 Gopeng St, #01-23 Icon Village, 5078877 Ends Sat, 06 Apr 2019
<b>Umi Sushi \$1 Sushi Box</b> All outlets except Kallang Bahru, ONE Raffles Place and Tan Tock Seng Every Thursday Ends Tue, 31 Dec 2019	<b>LiHo \$2 Strawberry Cold Foam Black Tea (M)</b> All outlets Ends Tue, 31 Dec 2019	<b>Bing Bing 1-For-1 Ice Cream</b> Bedok, Eastwood Centre, #01-13, S486442 Ends Sun, 07 Apr 2019	<b>KFC 1-For-1 Chicken, S'gara and Milo</b> Tampines Mall #01-47 Ends Sun, 07 Apr 2019
<b>Boufe Boutique Cafe 1-For-1 Coffee</b> Income at Raffles, 16 Collyer Quay, #02-12, S049318 Ends Fri, 05 Apr 2019	<b>Old Town White Coffee \$1 Selected Items</b> City Square Mall #B1-31 Ends Mon, 15 Apr 2019	<b>Cold Stone Creamery 1-For-1 Thai Milk Tea Ice Cream</b> Orchard Central #01-17 Ends Sun, 07 Apr 2019	<b>Spinelli 1-For-1 Sakura Beverage Series</b> All Outlets Ends Fri, 26 Apr 2019
<b>KFC \$1.95 Cheese Fries</b> All outlets except KidZania, Sentosa, Zoo	<b>LiHo \$0.90 Milk Tea</b> Suntec City, Far East Square, Centrepoint, Capitol Piazza,	<b>One Sushi 1-For-1 Mental Dishes</b> 3 Northpoint Drive, Yishun Townsquare, #01-04	<b>Soi 55 1-For-1 Thai Milk Tea</b> The Cathay #05-03

### 3. Non-Functional Requirements

1. Non-Functional Requirements describe the properties the system must have, that is not directly related to the functional behaviour of the system.
  - 1.1. Usability
    - 1.1.1. The system must display help messages in the local language according to the user's locale.
    - 1.1.2. The system must contain all the valid addresses in Singapore.
    - 1.1.3. The user interface must be intuitive and simple to accommodate users of all ages.
    - 1.1.4. The system must have at least one documentation or help function to guide users.
  - 1.2. Reliability
    - 1.2.1. In the event of a system reboot, system functionality must be restored in under 5 minutes.
    - 1.2.2. The system must be running 99% of the time.
    - 1.2.3. The system must not be able to lose more than 5% of any data.
    - 1.2.4. The map data must be at least 90% accurate.
    - 1.2.5. The user's data must not be leaked at all.
  - 1.3. Performance
    - 1.3.1. The system must respond to user input within 5 seconds.
    - 1.3.2. The system must be able to accommodate at least 10000 requests at any given point in time.
  - 1.4. Supportability
    - 1.4.1. The system must exhibit software design traits that show low coupling for maintainability and extensibility.
    - 1.4.2. The system must exhibit software design traits that show high modularity for maintainability and extensibility.
    - 1.4.3. There are plans to port the system over to mobile application.
    - 1.4.4. The system must be maintained with someone with approval from SG Discoverer developer.
    - 1.4.5. The system must be well documented.
  - 1.5. Correctness
    - 1.5.1. The system must reflect user's location accurate up to 50m.
    - 1.5.2. The system must reflect POI accurate up to 50m.
    - 1.5.3. The system must ensure data is real-time by updating the data within every 1 minute.
    - 1.5.4. The path generated by the system for guiding users to POI must be within the user's transport preference.
  - 1.6. Interface
    - 1.6.1. The system must interact with online mapping services.
    - 1.6.2. The system must interact with at least 1 application programming interface to import data.
    - 1.6.3. The system must export data on POIs to online mapping services without any errors.
  - 1.7. Legal
    - 1.7.1. The terms and condition for SG Discoverer must abide by Singapore's law.

## 4. Interface Requirements

### 4.1 User interface

1. Schneiderman's eight golden rule

- 1.1. Strive for consistency

Headers and buttons used in the website are consistent with one another with close to no changes in the design when navigating from page to page.

- 1.2. Cater to universal usability

The website is easy to navigate as most buttons are self-explanatory due to the placeholders of the buttons. Certain buttons will also make use of icons which are intuitive for the user. Furthermore, the number of buttons on screen are enough for user to navigate the necessary functions provided while not being overwhelming.

- 1.3. Offer informative feedback

User will be aware of the actions they do. There will be a alert pop up telling user when they have successfully registered for their account. Message boxes that gives details about the Point Of Interest will appear when user clicks the marker on the map. The change in login and logout button at the navigation bar to inform user on the login and logout status. When loading the map, the loading status of the map is displayed to inform that the map is loading instead of showing a blank page.

- 1.4. Design dialog to yield closure

There is no fixed sequence for the user and the functions in the website are all intuitive. Therefore, this golden rule is not applicable.

- 1.5. Permit easy reversal of actions

User can simply do another search to override their error. There is also backspace for when user types wrongly in the search bar.

- 1.6. Support internal locus of control

The actions done by the website is a direct consequence from what the user specifies it to do. Therefore, the user is in control when navigating the website.

- 1.7. Reduce short-term memory

The design of the interface is simple and intuitive to reduce the memory load on the user. When navigating to new pages like list view, the POI is arranged in rows and columns to help the user group them up and alleviate the amount of information being displayed.

- 1.8. Prevent error

There is minimal amount of input that is required by the user. The website will also search the next best alternative or rectify the error on its own. As an example, if the user misspells and types 'Ang No Kio'. The website will display results for 'Ang Mo Kio'.

## 4.2 Hardware

SG Discoverer requires hardware devices with location services to find user's current location. The hardware also requires internet connection to be able to search for POIs

## 4.3 Software

SG Discoverer is designed to work on website browsers including those on computer or browsers such as safari/opera which is typically used in mobile devices.

## 5. Data Dictionary

Term	Definition
User	A person using the application
Database	Stores information in tables (Account information, discounts, etc.)
Interface	The application platform which displays text and media to the user
Volume	Adjusts decibel (dB) level of application sound effects
Language	Method of communication through the interface (e.g. English, Mandarin, Japanese, etc.)
Menu	A list displayed, showcasing various elements
Drop-down Menu	A Menu that expands upon clicking to list more elements
Map	A diagrammatic representation of the surrounding area featuring roads, structures, etc.
Location	Indication of position on a map
POI (Point of Interest)	Refers to a location/structure on the map
Favourites	List of POIs saved by the user
Account	Belongs to the user. Saves user details, settings as well as favorited POIs. Can be used interchangeably with User
Username	A unique name tagged to an account, used to identify accounts
Account Profile	Displays details pertaining to the user, such as username or e-mail address, excluding confidential details such as passwords and phone numbers

Account Creation	A new account is created for a new user, with a username and password as input by the user
Login	User enters account credentials to be logged in, after application verifies authenticity of user
Password	A unique alphanumeric code set by the user, used to access one's own account
Marker	A pin to indicate location of a POI. POIs can be of varying types (Weather, Discount, Favourites, etc.)
Marker Details	A marker popup to display details of a marker, such as the POI type, name, address and Google Ratings
Discount	A type of marker showcasing discounts and outlets offering the discount/deal
Route	A path containing a series of instructions to travel from a user's current location to another stop
Destination	A final stop in a route, to which a user intends to arrive
Search Box	A search box to key in desired destination
API (Application Programming Interface)	Set of procedures for interacting with pre-existing pieces of software
GPS (Global Positioning Satellite)	Tracks the user's location, to allow application to provide accurate instructions
Navigate	The act of making one's way towards a desired destination
Directions	Step-by-step instructions on how to navigate towards destination
Distance	How far apart two locations are on the map
District	Referring to a distinct area in Singapore, that has no clear border - Singapore is split into 23 distinct areas (districts)
Search	A function that allows users to filter through locations on the map via keywords, to return locations more relevant to the keyword inputs



Search Radius	A radius of search defined such that only POIs that fall within the circle defined, centered on a reference point, are returned (e.g. 400m - only POIs within 400m from reference point are displayed)
Sort	A function to arrange a list according to user's specified ordering - e.g. (Sort by ratings, sort by proximity)
Keywords	Words that give details or additional information on a location to narrow search results to better cater to user's desires
Input	Text - Terms or words keyed in by the user into text fields, to be processed by the application  Voice - Inputs made via microphone
Filter	A function performed by the application to remove items deemed irrelevant by the application, via algorithms, to narrow down or give more specific results, particularly used in "Searching"
Category	Method of sorting POIs by their type - e.g. Restaurants, Bars, Parking, Health, etc.
Rating	A Google Rating for each POI marker

## 6. Testing

### 6.1 Black Box Testing

#### 1. Login

##### a. Generic Cases

Test Id	Scenario	Expected Result	Actual Result
1	Login with valid account username and password	User successfully logs in and is brought to homepage	User successfully logs in and is brought to homepage
2	Login without valid credentials	The system prompts the user to enter credentials again	The system prompts the user to enter credentials again
3	Login without filling up the required fields	The system prompts the user to fill up the required fields for logging in	The system prompts the user to fill up the required fields for logging in

##### b. Specific Cases

Username	Password	Expected Result	Actual Result
username	password	Successful login	Successful login
wronguser	password	Invalid username/password	Invalid username/password
Empty ("")	password	Please fill in all required fields	Please fill in all required fields
username	wrongpass	Invalid username/password	Invalid username/password
username	Empty ("")	Please fill in all required fields	Please fill in all required fields

#### 2. Registration

##### a. Generic Cases

Test Id	Scenario	Expected Result	Actual Result
1	Register with valid account username and password	The system displays the main menu for user to continue the operation	The system displays the main menu for user to continue the operation
2	Register with incomplete fields	The system prompts the user to fill up the required fields for registration	The system prompts the user to fill up the required fields for registration
3	Register with password mismatch	The system prompts the user re-enter the password	The system prompts the user re-enter the password



4	Register with used username	The system prompts the user to choose another username	The system prompts the user to choose another username
---	-----------------------------	--	--

#### b. Specific Cases

Username	Password	Retype Password	Expected Result	Actual Result
username	password	password	Created new user	Created new user
Empty ("")	password	password	Please fill in all required fields	Please fill in all required fields
username	Empty ("")	password	Please fill in all required fields	Please fill in all required fields
username	password	Empty ("")	Please fill in all required fields	Please fill in all required fields
username	password	password1	Passwords do not match	Passwords do not match
username2	password	password	Username is taken	Username is taken

### 3. Favourites

#### a. Generic Cases

Test Id	Scenario	Expected Result	Actual Result
1	Favourite a location	The system saves the location into favourite list. Favourite appears in navbar favourites dropdown.	The system saves the location into favourite list. Favourite appears in navbar favourites dropdown.
2	Unfavourite a location	The system removes the location from favourite list. Favourite disappears from navbar favourites dropdown.	The system removes the location from favourite list. Favourite disappears from navbar favourites dropdown.

#### b. Specific Cases

Scenario	Expected Result	Actual Result
Favourite 'Northspine KFC'	'Northspine KFC' appears in navbar favourites dropdown	'Northspine KFC' appears in navbar favourites dropdown
Unfavourite 'Northspine KFC'	'Northspine KFC' disappears from navbar favourites dropdown	'Northspine KFC' disappears from navbar favourites dropdown

### 4. Get User's current location

Scenario	Expected Result	Actual Result
User allows browser to get	User's location appears as	User's location appears as

current location	yellow pin on the map.	yellow pin on the map.
User does not allow browser to get current location	User's location does not appear on the map.	User's location does not appear on the map.

**5. Search for locations (Choose at least 2 representative values from each parameter)**

Text entry	Select District	Select Type	Search Radius/m	Rating (1-5)	Expected Result	Actual Result
Empty ("")	Not Selected (Default: User location)	Not Selected (Default: Food)	Not Selected (Default: 400)	Not Selected (Default: 0)	Food locations within 400m of user's location with any rating	Food locations within 400m of user's location with any rating
Empty ("")	Bedok	Food	200	1	Food locations within 200m of Bedok with rating of at least 1	Food locations within 200m of Bedok with rating of at least 1
Empty ("")	Anson	Parking	800	3	Parking locations within 800m of Anson with rating of at least 3	Parking locations within 800m of Anson with rating of at least 3
Tuas	Not Selected (Cannot enter text entry unless no district selected)	School	600	2	School locations within 600m of Tuas with rating of at least 2	School locations within 600m of Tuas with rating of at least 2

**6. Sort by rating**

Scenario	Expected Result	Actual Result
Click sort by rating after searching for locations	Cards are sorted in order, from highest to lowest rating.	Cards are sorted in order, from highest to lowest rating.

**Note: Verify ratings using google.**

### 7. Sort by distance (Test cases start after clicking on sort by distance)

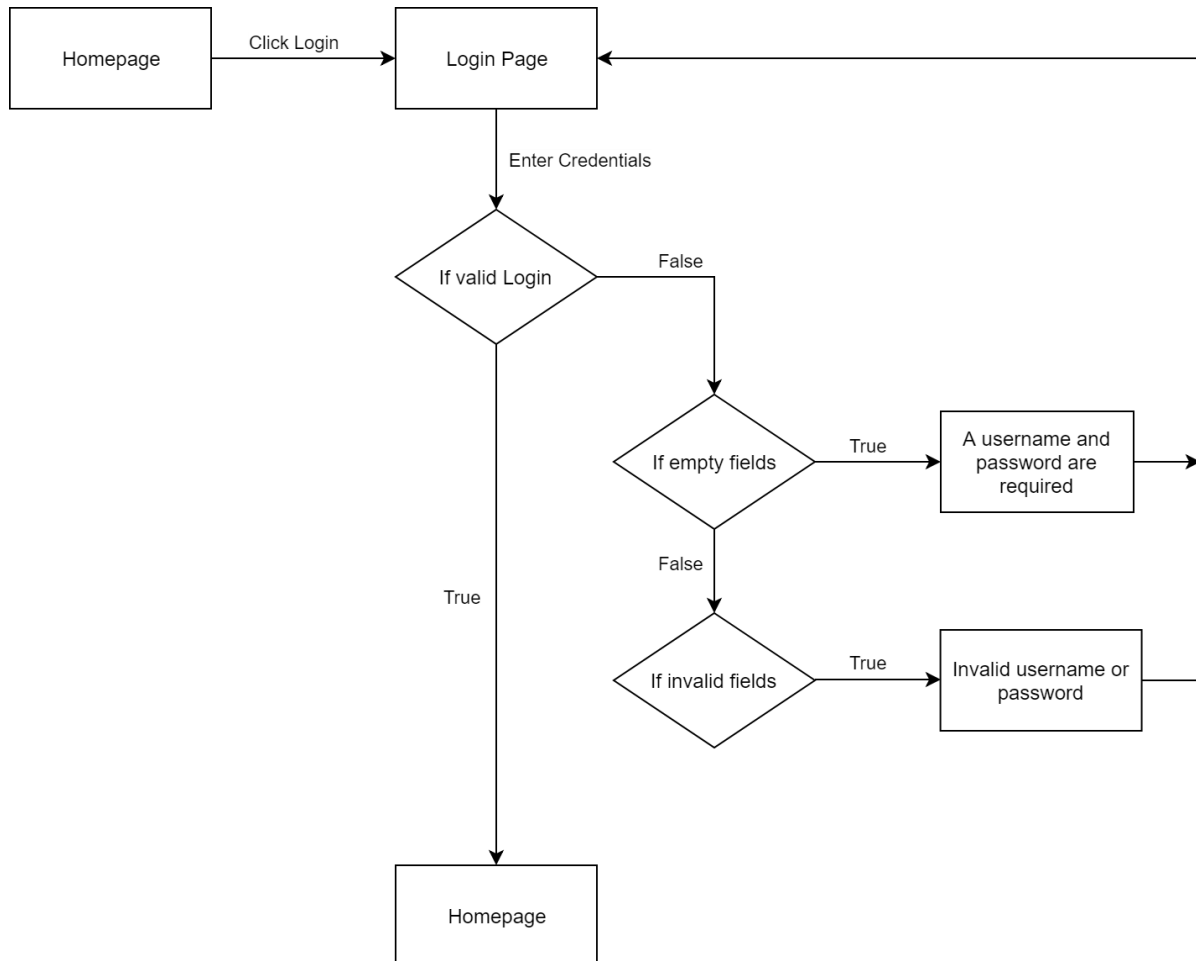
Enter current location	Mode of Transit	Expected Result	Actual Result
Empty ("")	N.A	User is prompted to enter current location	User is prompted to enter current location
"Anchorvale Road"	"Flying"	User is prompted to enter either "Walking, Driving or Transit"	User is prompted to enter either "Walking, Driving or Transit"
"Anchorvale Road"	Empty ("")	User is prompted to enter either "Walking, Driving or Transit"	User is prompted to enter either "Walking, Driving or Transit"
"aosfoaf"	"Walking"	Current location is not valid, user is prompted to enter current location	Current location is not valid, user is prompted to enter current location
"Anchorvale Road"	"Walking"	Cards are sorted in order, from closest place to Anchorvale Road to furthest place to Anchorvale Road, based on walking distances from Google API.	Cards are sorted in order, from closest place to Anchorvale Road to furthest place to Anchorvale Road, based on walking distances from Google API.
"Rivervale Plaza"	"Driving"	Cards are sorted in order, from closest place to Rivervale Plaza to furthest place to Rivervale Plaza, based on driving distances from Google API.	Cards are sorted in order, from closest place to Rivervale Plaza to furthest place to Rivervale Plaza, based on driving distances from Google API.
"NTU"	"Transit"	Cards are sorted in order, from closest place to NTU to furthest place to NTU, based on public transport distances from Google API.	Cards are sorted in order, from closest place to NTU to furthest place to NTU, based on public transport distances from Google API.

**Note: Verify distances using google map.**

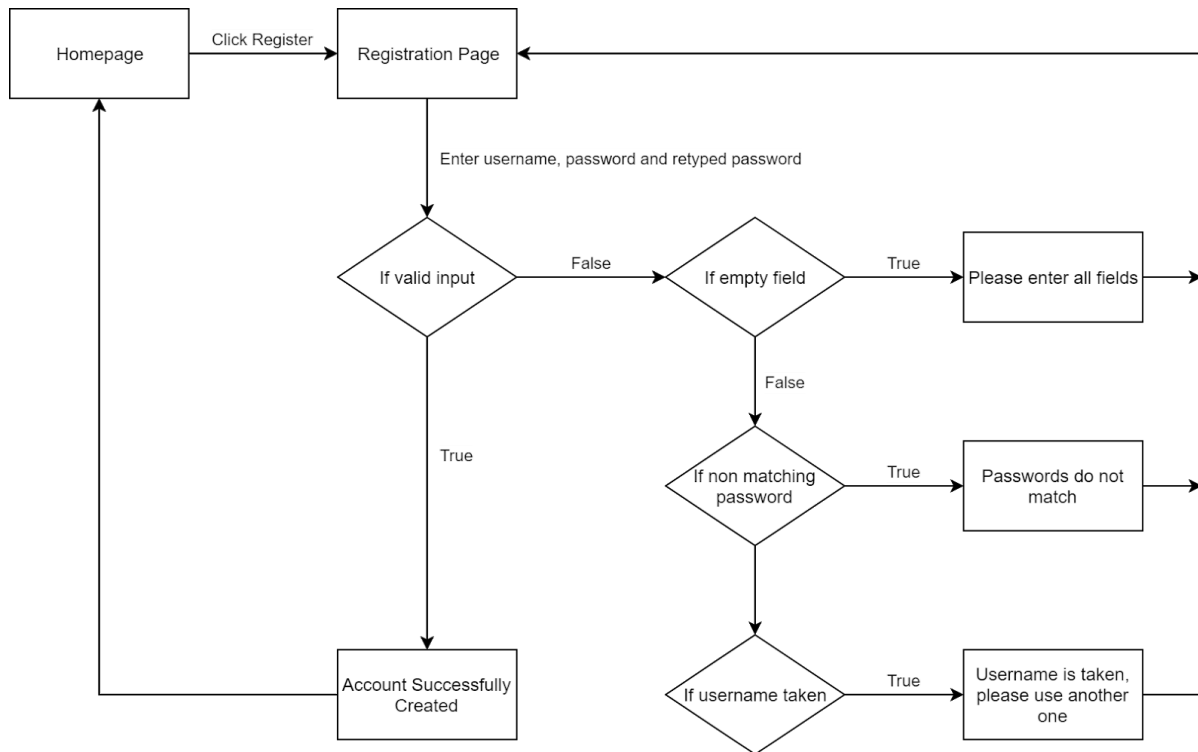
## 6.2 White Box Testing

White box testing is performed for more complication functions.

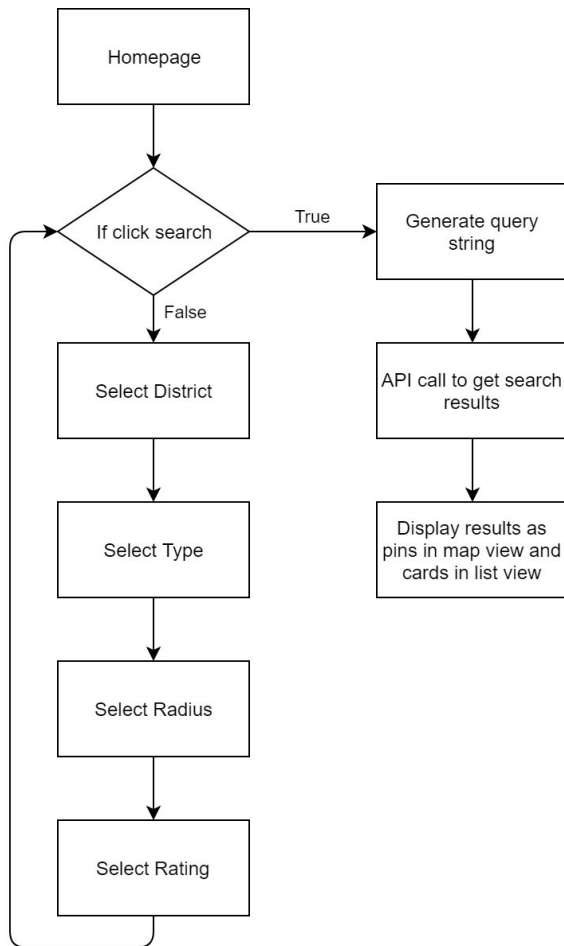
### 1. Login



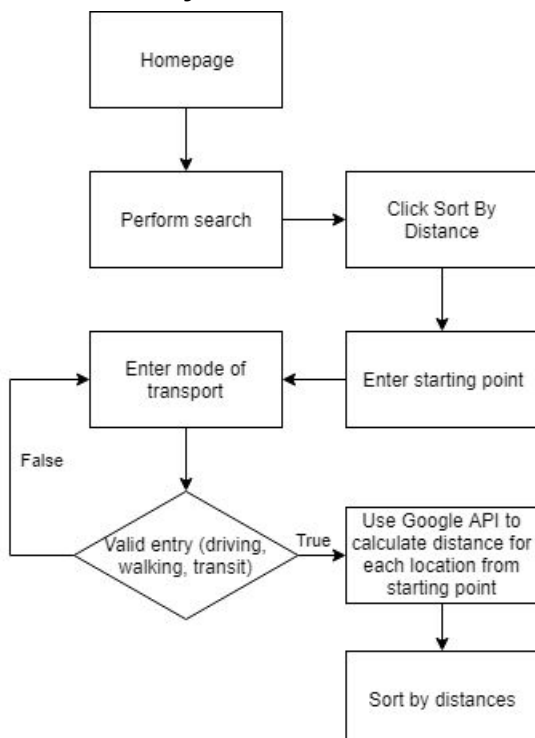
## 2. Registration



### 3. Search



### 4. Sort By Distance



### 5. Favourite(Add, Remove, View)

