Software Requirements Specification

for

<Sportify>

Version 1.1 approved

Prepared by <Randall Chiang Tian Cong (U2222099A),

Tuan Ding Ren (U2223920F),

Shen Chihao (N2304724K),

Nanditha Kumar (U2221998B),

Ng Shi Wen (U2221975L)>

<SC2006>

<20 April 2024>

Table of Contents

1. Introduction 1
- **1.1 Purpose** 1
- **1.2 Document Conventions** 1
- **1.3 Intended Audience and Reading Suggestions** 1
- **1.4 Product Scope** 1
- **1.5 References** 2

2. Overall Description 2
- **2.1 Product Perspective** 2
- **2.2 Product Functions** 2
- **2.3 User Classes and Characteristics** 3
- **2.4 Operating Environment** 3
- **2.5 Design and Implementation Constraints** 3
- **2.6 User Documentation** 4
- **2.7 Assumptions and Dependencies** 4
  - **2.7.1 Assumptions** 4
  - **2.7.2 Dependencies** 4

3. External Interface Requirements 5
- **3.1 User Interfaces** 5
  - **3.1.1 Home Page** 5
  - **3.1.2 Result Page** 7
  - **3.1.3 Find Out More Page** 8
  - **3.1.4 Check-In/Pre-Check-Out Page** 9
  - **3.1.5 Playing/Check-Out Page** 9
  - **3.1.6 Check-Out Page** 10
- **3.2 Hardware Interfaces** 10
- **3.3 Software Interfaces** 10
- **3.4 Communications Interfaces** 11

4. System Features 12
- **4.1 Get Current Location** 12
  - **4.1.1 Description and Priority** 12
  - **4.1.2 Stimulus/Response Sequences** 12
  - **4.1.3 Functional Requirements** 12
- **4.2 Specify Search Criteria** 13
  - **4.2.1 Description and Priority** 13
  - **4.2.2 Stimulus/Response Sequences** 13
  - **4.2.3 Functional Requirements** 13
- **4.3 Display Recommendation** 14
  - **4.3.1 Description and Priority** 14
  - **4.3.2 Stimulus/Response Sequences** 14

4.3.3 Functional Requirements 14
4.4 Display Location Details 15
4.4.1 Description and Priority 15
4.4.2 Stimulus/Response Sequences 15
4.4.3 Functional Requirements 15
4.5 Get Location Details 16
4.5.1 Description and Priority 16
4.5.2 Stimulus/Response Sequences 16
4.5.3 Functional Requirements 16
4.6 Calculate Scores 17
4.6.1 Description and Priority 17
4.6.2 Stimulus/Response Sequences 17
4.6.3 Functional Requirements 17
4.7 Check-in/Check-out 18
4.7.1 Description and Priority 18
4.7.2 Stimulus/Response Sequences 18
4.7.3 Functional Requirements 18
4.8 Feedback 19
4.8.1 Description and Priority 19
4.8.2 Stimulus/Response Sequences 19
4.8.3 Functional Requirements 19
5. Other Nonfunctional Requirements 20
5.1 Performance Requirements 20
5.2 Safety Requirements 20
5.3 Security Requirements 20
5.4 Software Quality Attributes 20
5.4.1 Reliability: 20
5.4.2 Availability 21
5.4.3 Maintainability 21
5.4.4 Compatibility 21
5.4.5 Usability 21
5.4.6 Scalability 21
5.5 Business Rules 21
5.6 Design Patterns 22
5.6.1 Facade Pattern 22
5.6.2 Singleton Pattern 23
6. Use Case Diagram 24
7. Use Case Description 25
8. Data Dictionary 37
9. UI Mockups 41
9.1 Home Page 41

9.2 Home Page with Filter 41
9.3 Home Page with Address List 42
9.4 Help page 42
9.5 Result page 43
9.6 Find out more page 43
9.7 Check-in/Pre-Check-Out page 44
9.8 Playing/Check-Out page 44
9.9 Check-Out Page 45
10. Class Diagrams 46
10.1 Home 46
10.2 Search Results 47
10.3 Calculate Score 48
10.4 LocationPage 49
10.5 Feedback 50
10.6 Whole Thing 51
11. Class Entity Diagram 52
12. Key Public Methods 53
13. Sequence Diagram 58
13.1 EnterUserLocation 58
13.2 SpecifySearchCriteria 59
13.3 GetAPIValues 60
13.4 CalculateScore 60
13.5 DisplayRecommendations 61
13.6 Pre-Check-In 62
13.7 Check-In 63
13.8 Playing 64
13.9 Feedback 65
14. Dialog Map 66
15. System Architecture 67
16. Testing 68
16.1 Black Box Testing 68
16.1.1 Functionality: Inputting Search Criteria 68
16.1.2 Functionality: Search Results 69
16.1.3 Functionality: Check-In System 70
16.1.3 Functionality: “Help” Button Feedback 71
16.2 White Box Testing 72
17. Demo Script 82

Revision History

Name	Date	Reason For Changes	Version
Randall, Chihao, Ding Ren, Nanditha, Shi Wen	7/4/2024	The first draft	1.0
Randall, Chihao, Ding Ren, Nanditha, Shi Wen	20/4/2024	The final version	1.1

1. Introduction

1.1 Purpose

The main purpose of this Software Requirements and Specifications (SRS) document is to provide a detailed explanation and understanding of Sportify, a web-application where sports enthusiasts or anyone who wants to keep fit in Singapore can easily find a sport they like based on the weather condition and mode of transport.

The SRS will cover the functionalities, interface requirements and system features. Additionally, this document will provide diagrams and test results to provide an overview of Sportify and ensure the full working functionalities of the web-application.

1.2 Document Conventions

Heading1: Font: Arial, Size: 18, bold
Heading 2: Font: Arial, Size: 14, bold
Heading 3: Font: Arial, Size: 12, bold
Normal Text: Font: Arial size: 12

1.3 Intended Audience and Reading Suggestions

This document is intended for developers, project managers, users and testers. The recommended reading order is as follows:

- Developers and Project Managers: Follow the order of Table of Contents.
- Users: Read through the use case diagram and system features to understand how the web-application works
- Testers: Read through use case diagram, system features and testing sections to understand how the web-application works and the tests we have conducted

The recommended sequence of reading this SRS, would be to follow the sequence of sections as listed in the content page.

1.4 Product Scope

Sportify aims to help users to find a nearby sports location based on the weather condition and mode of transport. Through our application, users can view information about each location such as the type of activities the location offers and the address. Additionally, with the built in check-in and check-out features, users will get a fairly accurate idea of how

crowded the venue is at the current time. With Sportify, sports enthusiasts or those aiming for a healthy lifestyle will have an easier time deciding what type of sports to play or which sports venue to visit. This also reduces the possibility of making wasted trips due to unexpected weather conditions and overcrowding at the sports venue.

1.5 References

- 7 data integrity best practices you need to know. Atlan. (n.d.). https://atlan.com/data-integrity-best-practices/
- Author, G. (2022, July 14). What is react.js? introduction and definition. Code Institute Global. https://codeinstitute.net/global/blog/what-is-react-js/

2. Overall Description

2.1 Product Perspective

With Sportify, users will be able to find sports venues based on their choice of location or his/her current location, current weather condition and mode of transport. Additionally, users will also get to adjust the distance they are willing to travel to from their location using the radius function. The Sportify system will then return the list of locations based on the previously mentioned conditions. These functionalities were made possible with the use of data.gov.sg and GoogleMaps API.

Current methods of finding nearby sports locations are limited to google maps, which tend to contain unnecessary information and does not consider weather factors. Sportify's unique value proposition is to provide users a easy to use and intuitive application for users to easily discover suitable sports locations. This complements google maps as they can first use Sportify to find their desired sports location and then use Google Maps to navigate.

2.2 Product Functions

Major Functions:

1. Filter Options
2. Calculation of map distances
3. Collate weather details through data.gov.sg APIs
4. Search Results
5. Score Calculator
6. Location Details with path details using GoogleMap API
7. A live counter of number of users at different statuses

2.3 User Classes and Characteristics

Users who use Sportify are expected to have an interest in playing sports and be aiming to maintain a healthy lifestyle. In addition, users must be located in Singapore, have access to the internet any browser of their choice.

Users must have basic experience with using a web or mobile phone browser. They are also expected to be within the geographical boundaries of Singapore, given that the application only caters to locations in Singapore.

2.4 Operating Environment

Users who uses Sportify, must have a working internet connection and browser. The Google Maps API and all weather APIs from data.gov.sg must be working well to provide accurate data. Sportify's frontend is developed using React JS, version 18.2.0. The backend is developed using Firebase and the location data is stored locally using a CSV file.

React is a free open-source Javascript library. It is commonly used in building application's user interfaces by combining different components together to form the application's UI. Before running the application on the localhost, dependencies used in the application need to be downloaded using the command "npm install - - force". Once the dependencies have been downloaded, use the command "npm start" to run the application on the local host. Firebase is a tool that offers backend-as-a-service(Baas) for developing web and mobile applications, helping to handle backend infrastructure.

2.5 Design and Implementation Constraints

During the development stage, we faced difficulty in finding ways to extract all the data points for sport venues in Singapore as we could not find any existing APIs that could provide us with the necessary data. This posed a significant challenge because without the data points, we will not be able to generate the sport location near the users.

Sportify uses Google Maps API and five other APIs from data.gov.sg, if there are any technical issues with the APIs, the majority of our application's function will stop working, as it heavily relies on them to provide necessary information it needs to calculate the scores for each location near the user.

2.6 User Documentation

A short tutorial video on how to use the web-application as well as a user manual will be provided to guide the users:

1. Tutorial Video
2. User Manual

For any technical difficulties, contact the development team via Github.

2.7 Assumptions and Dependencies

2.7.1 Assumptions

For Sportify to be fully functional, it is assumed that users have a stable connection and the APIs used do not have any technical issues and are able to provide Sportify with the necessary accurate information it needs to function. Additionally, the user's device must have a working Global Positioning System(GPS) to allow Sportify to retrieve their current location if needed.

2.7.2 Dependencies

Sportify predominantly uses the Google Maps API to fetch the geolocation of sports locations, as well as any navigation routes. Without the API, users will not be able to access the map view to observe his/her current location, the location of the destination, as the route between these two locations.

Sportify also uses five real-time APIs provided by the Singapore government under data.gov.sg, namely:

1. Ultra-violet Index (UVI) API
2. Pollutants Standard Index (PSI) API
3. Air Temperature API
4. Rainfall API
5. 2-hour weather forecast API

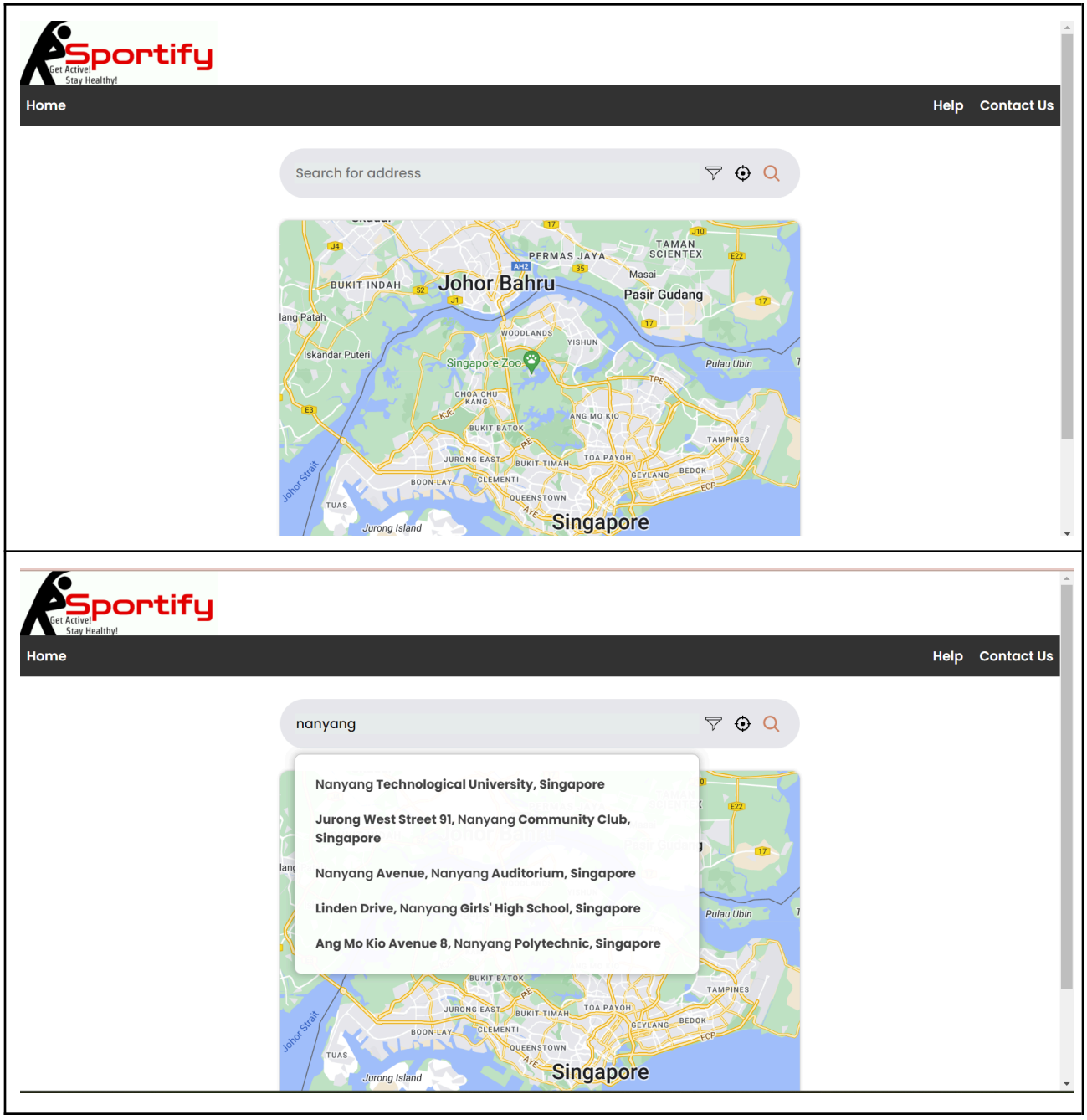
These APIs retrieve the most updated weather information with every search, and the score calculation and search results are dependent on the readings fetched by the APIs.
All of these APIs can be found on data.gov.sg and are licensed to be used under the Singapore Open Data License.

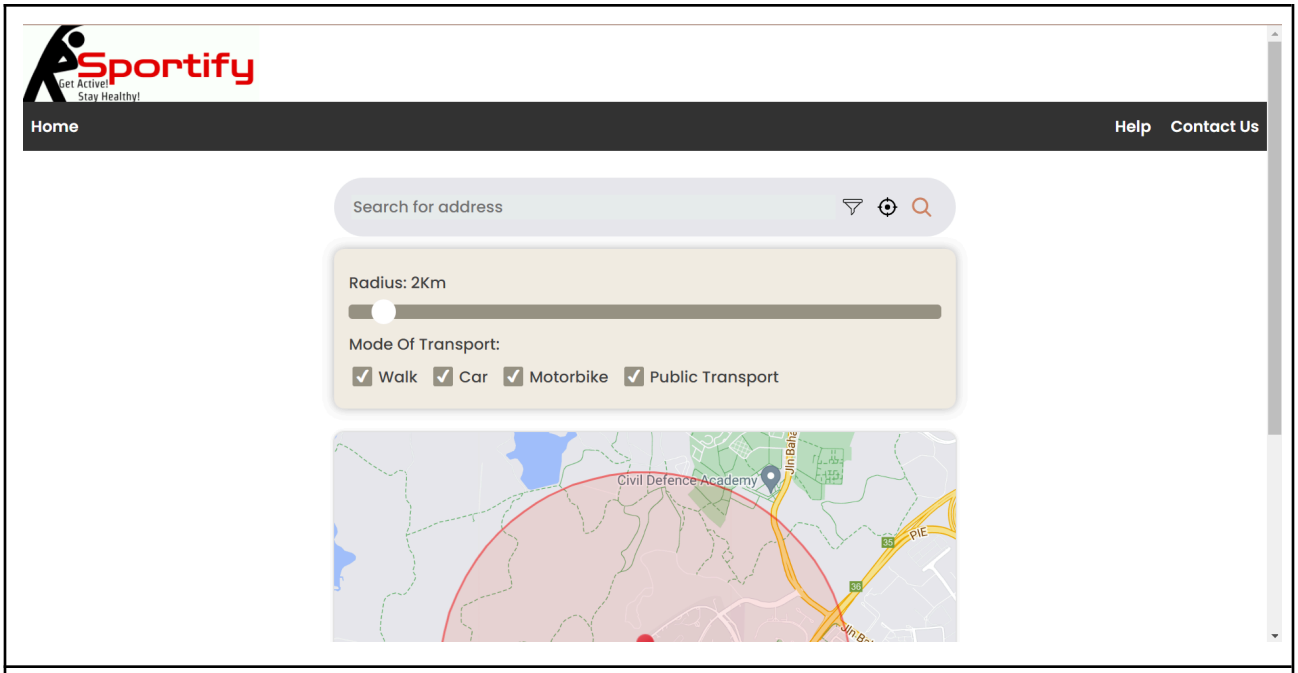
For easy code reusability and extension, our web application uses commonly used open-source frameworks. The frontend uses React as the primary frontend framework with JavaScript as the primary programming language.

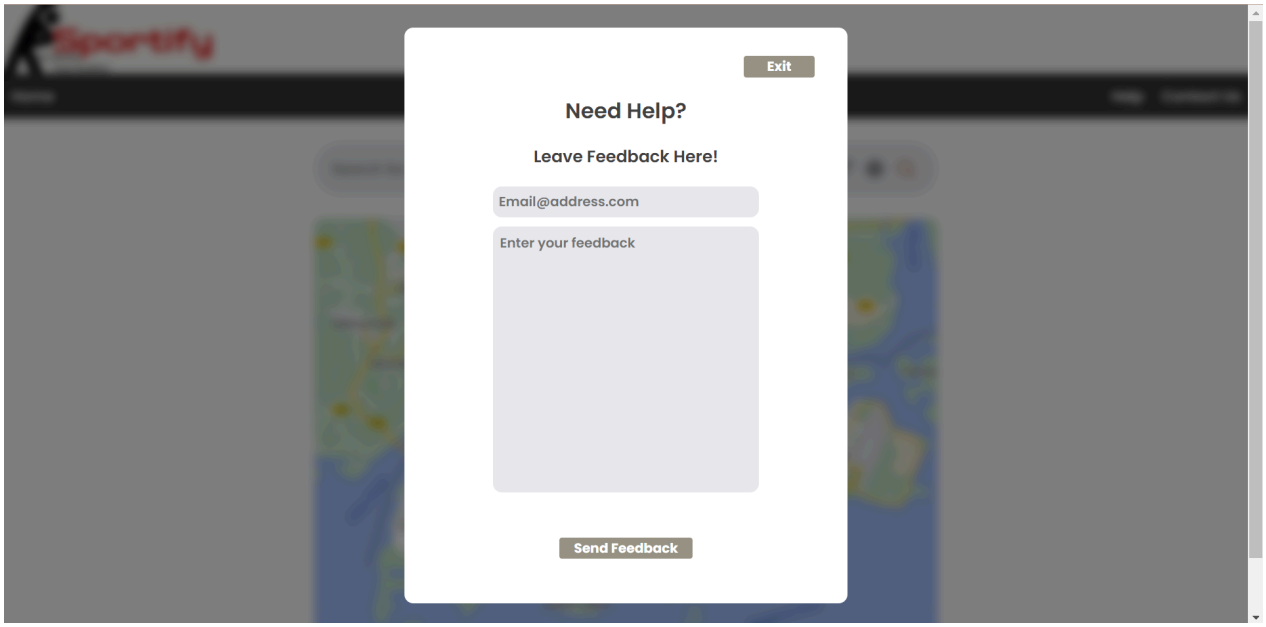
3. External Interface Requirements

3.1 User Interfaces

3.1.1 Home Page



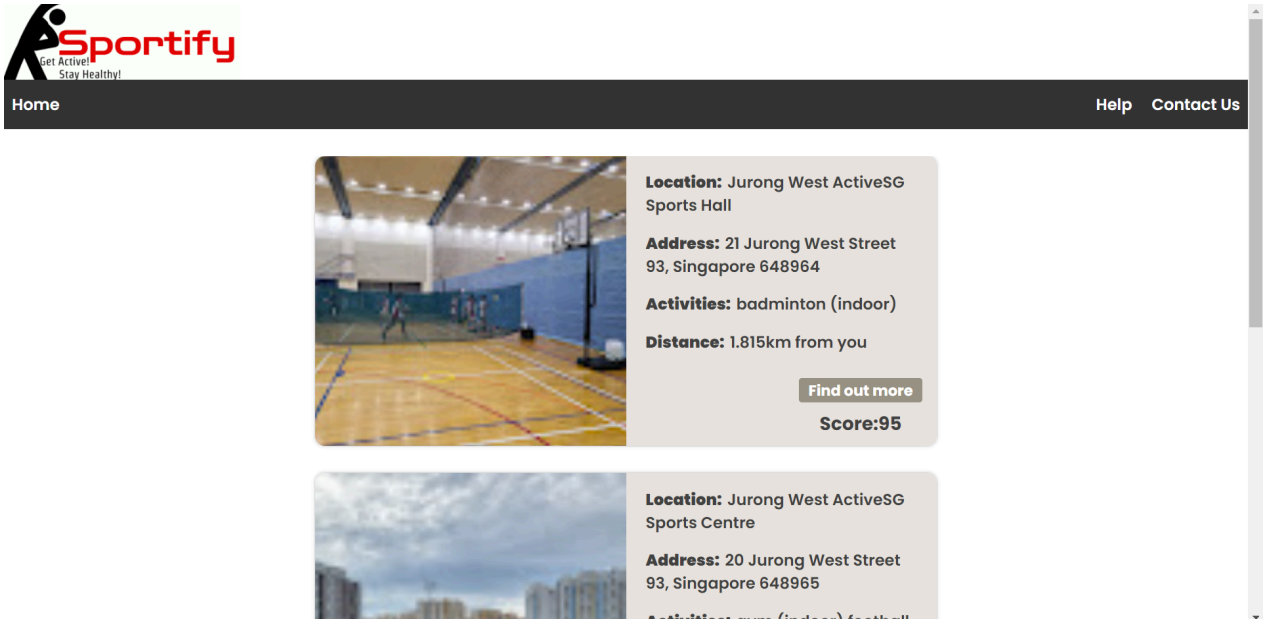




- Users can choose to enter their departure location in the search bar.
- Users can select 1 of the addresses from the suggested list of addresses to auto fill the search bar when entering their departure location.
- Users can also choose to automatically detect their departure location using GPS by clicking the GPS button instead.
- Users can filter and adjust radius distance and Mode of Transport by clicking the filter button.
- Users can contact the development team via GitHub by clicking the “Contact Us” button.

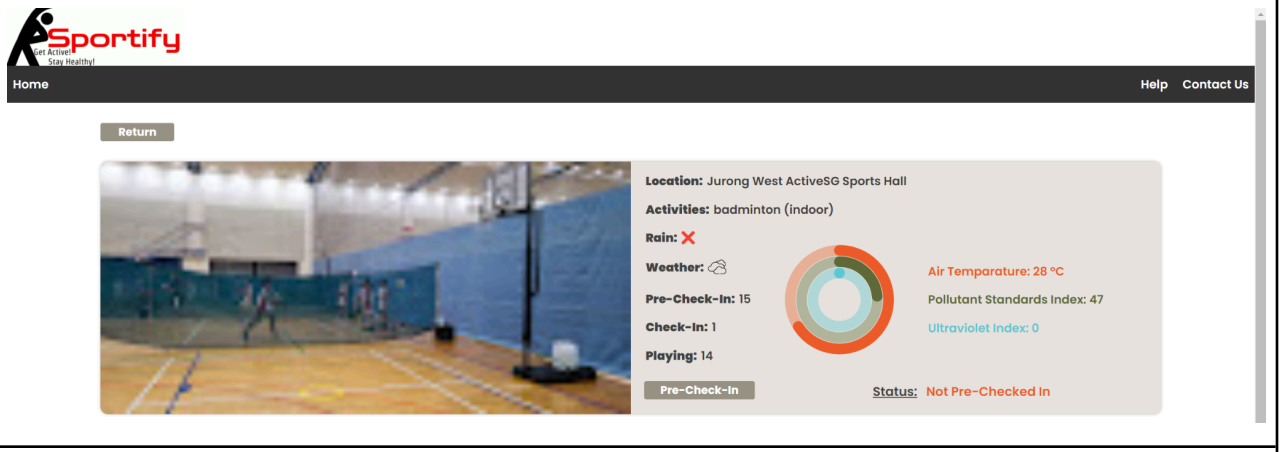
- User can request for help by clicking the “Help” button and sending a feedback.
- Users can search for the recommended sports location by clicking the search button after setting their departure location and filter option.

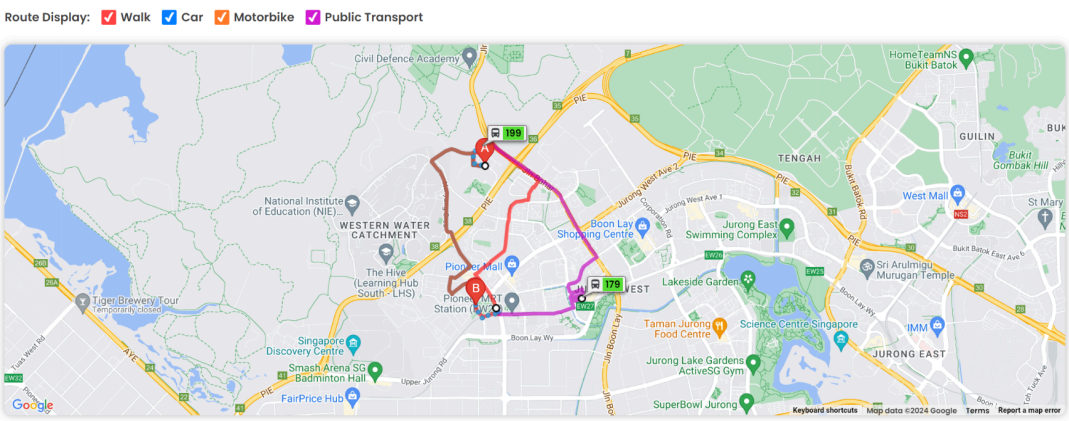
3.1.2 Result Page



- Users can view the sports locations with their respective scores, with the locations being ranked in a descending order according to their scores.
- Users can view the location name, address, type of activities and distance of t from their departure location.
- Users can click the “Find out More” button to view additional details about their desired sports location.
- Users can return back to the Home Page by clicking the “Home” button.

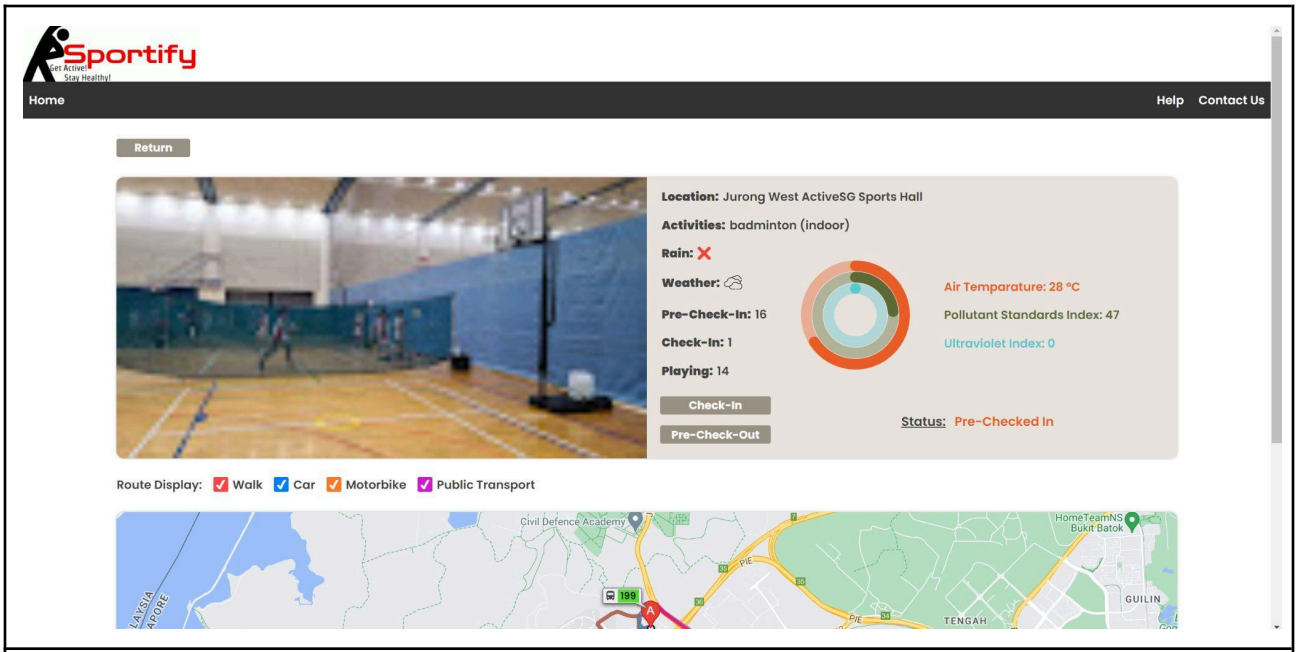
3.1.3 Find Out More Page





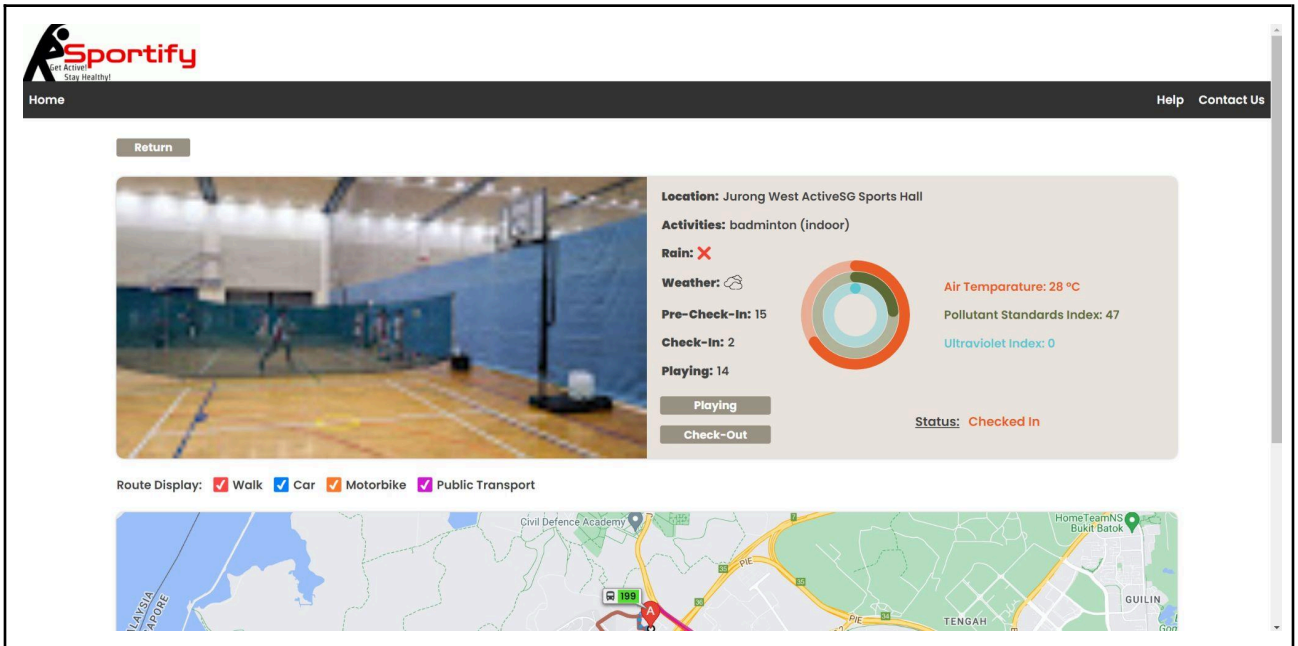
- Users can view the location name, environment conditions, playing, pre-check-in value and check-in value.
- Users can pre-check-in by clicking the “Pre-Check-In” button.
- Users can contact the development team via GitHub by clicking the “Contact Us” button.
- Users can go back to the Result Page by clicking the “Return” button.
- Users can view the route based on their preferred Mode of Transport.
- User can check their current status

3.1.4 Check-In/Pre-Check-Out Page



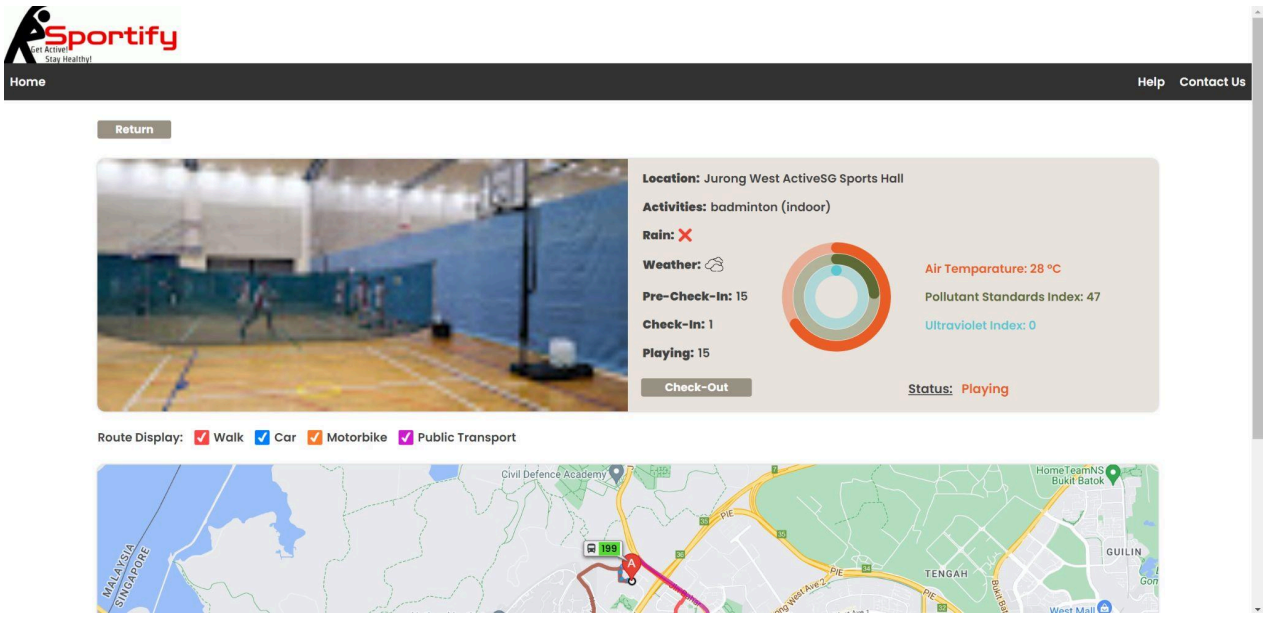
- Users can Check-in by clicking the “Check-In” button.
- Users can Pre-check-out by clicking the “Pre-check-out” button.

3.1.5 Playing/Check-Out Page



- Users can indicate they are playing by clicking the “Playing” button.
- Users can Check-out by clicking the “Check-out” button.

3.1.6 Check-Out Page



- Users can Check-out by clicking the “Check-out” button.

3.2 Hardware Interfaces

Sportify works on various device types, including mobile phones, laptop and desktop with working browsers with working GPS. Additionally, the device is also able to receive inputs from I/O peripherals such as touch screen and mouse.

3.3 Software Interfaces

Installation of a web browser is needed to use the web-application. React JS version 18.2.0 is used to create individual components within the web-application. The Google Maps API is used to provide the longitude and latitude coordinates of a user's location, assisting us in calculating the distance between the sports venue location and user’s location.

The APIs from data.gov.sg we are using include:

1. Air Temperature across Singapore: To retrieve real-time air temperature readings in degreeC across Singapore.
2. 2-hour Weather Forecast: To retrieve the next 2 hours weather forecast across Singapore
3. Ultraviolet Index (UVI): To retrieve hourly UVI results in Singapore

4. Pollutant Standards Index (PSI): To retrieve real-time PSI information across Singapore
5. Rainfall across Singapore: To retrieve the total rainfall measure in mm over 5 minutes in real-time across Singapore.

3.4 Communications Interfaces

Sportify require HTTP communication protocol to retrieve data from data.gov.sg APIs. To retrieve data from all the APIs, users must be connected to the internet. Additionally, it also requires the user's geolocation, this can be obtained via GPS or user inputting it to the system manually.

4. System Features

4.1 Get Current Location

4.1.1 Description and Priority

The “Get current location” feature will get the user's current location either through their device’s GPS or the user will enter their current location into the search bar. This feature is of high priority since the user’s location is one of the essential information the system needs for generating the list of suggested sports venues.

4.1.2 Stimulus/Response Sequences

SEQ-1: User selects the search bar
SEQ-2: User enters name of location
SEQ-3: User selects location from a dropdown list of recommendations obtained from Google Map API.

OR

SEQ-1: User press GPS button
SEQ-2: User allow use of the device's GPS

4.1.3 Functional Requirements

REQ-1: User must query the Sportify system by device’s GPS or by typing in their current location to confirm the departure location.

1.1 Results for a query containing a specific address must report all locations relevant to that address in a drop-down list for the user to choose.

1.1.1 User shall choose a location as the departure location.

1.1.2 User shall report the issue or start another search if no wanted result.

1.2 User must query the Sportify system by the use of their device’s GPS

1.3 Sportify system must verify the current location provided by the user’s GPS or by the user themselves

1.4 Results for a query about current location must indicate the user’s location as the departure location.

REQ-2: The Sportify system shall display some useful information for the departure location.

2.1 Google Map shall set the departure location and display it as the center of the map

4.2 Specify Search Criteria

4.2.1 Description and Priority

The “Specify Search Criteria” feature will prompt the user to specify their search radius and preferred mode of transport. This feature is of high priority since the search radius plays an important role in locating the list of recommended sports venue.

4.2.2 Stimulus/Response Sequences

SEQ-1: User selects filter button
SEQ-2: User adjust the search radius slider bar to their desired search radius
SEQ-3: User select their preferred mode(s) of transport

4.2.3 Functional Requirements

REQ-1: User must specify a search radius to determine the user reachable scope.

1.1 System must check that there are sport venues located within the specified search radius

1.1.1 System will prompt user for a new address if there are no sport venues located within the specified search radius

REQ-2: User must choose at least one of their preferred mode of transport.

REQ-3: The Sportify system shall display some useful information for all the sports locations within the search radius.

3.1 Google Map shall return a specific map with markers to indicate the location for all sports locations within the search radius.

3.1.1 Each markers in the map shall include the name and address of the sports location

4.3 Display Recommendation

4.3.1 Description and Priority

The “Display Recommendation” feature will display the list of locations within the search radius in descending order based on the score calculated for each location. This feature is of high priority since this feature is responsible for providing the user the recommended sports location.

4.3.2 Stimulus/Response Sequences

SEQ-1: User clicks the search button after providing the system with its current locations and adjusting the filter options.

4.3.3 Functional Requirements

REQ-1: The Sportify system shall display all recommendation results within search radius.

1.1 System must check that there are sport venues located within the specified search radius

1.2 The Sportify system shall derive the result based on considering factors.

1.2.1 Factors shall consider weather conditions in the sports venue.

1.2.1.1 Weather conditions shall consider rainfall level.

1.2.1.2 Weather conditions shall consider Ultraviolet index (UVI).

1.2.1.3 Weather conditions shall consider the Pollutant Standards Index (PSI).

1.2.1.4 Weather conditions shall consider the air temperature

1.2.2 Factors shall consider the distance from the user’s current location to the sports location.

1.3 Each result must display the following recommended information.

1.3.1 Recommended information must contain the name of the sport facility location.

1.3.2 Recommended information must contain the address of the sport facility location.

1.3.3 Recommended information must contain the type of the sport activity.

1.3.4 Recommended information must contain the distance from the user’s current location to the sports location.

1.3.5 Recommended information must contain the final score of the sports location

1.4 Each result must include a “find out more” button.

4.4 Display Location Details

4.4.1 Description and Priority

The “Display Location Details” feature will display additional information about the location.This include environmental condition, routing details, user status, etc. This feature is of high priority since this feature is responsible for providing the user additional information about the recommended sports location.

4.4.2 Stimulus/Response Sequences

SEQ-1: User clicks “Find out More” button in the Search Result page

4.4.3 Functional Requirements

REQ-1: User shall click the “Find out More” button to see extra information of the recommended result.

1.1 The extra information must contain the temperature of the sport facility location.

1.2 The extra information must contain whether there would be rain at the sports facility location.

1.3 The extra information must contain the PSI of the sport facility location.

1.4 The extra information must contain the UVI of the sport facility location.

1.5 The extra information must contain the weather forecast of the sport facility location.

1.6 The extra information shall contain buttons for the check in process.

- **1.6.1** Buttons shall include a “Pre-check-in” button.
- **1.6.2** Buttons shall include a “Pre-check-out” button.
- **1.6.3** Buttons shall include a “Check-in” button.
- **1.6.4** Buttons shall include a “Check-out” button.
- **1.6.5** Buttons shall include a “Playing” button.

1.7 The extra information shall contain routing details of user’s preferred mode of transport

- **1.7.1** routing details shall include buttons for user to toggle in between their desired mode of transport

1.8 The extra information must contain the status of the user

1.9 The extra information must contain vacancy information in the sports location

- **1.9.1** Vacancy information shall include check-in count
- **1.9.2** Vacancy information shall include playing count

4.5 Get Location Details

4.5.1 Description and Priority

The “Get Location Detail” feature will get the API data and the Travel Mode that will be used in the Location Detail page to provide users with the additional information about the sports location. This feature is of high priority since this feature is responsible for getting the additional information on the sports location.

4.5.2 Stimulus/Response Sequences

SEQ-1: Sportify queries the PSI API, Air Temperature API, Weather API, UVI API and Rainfall API to get their respective values from areas across Singapore.
SEQ-2: Sportify retrieves the environmental data from the area which the location is closest to.
SEQ-3: Sportify returns the values from the identified area to take reference from for location.
SEQ-4: Sportify queries the Google Map API to get the most optimized routes to location based on the user's selected mode of travel.
SEQ-5: The Google Map API returns the routes of location in search radius from user’s specified location.

4.5.3 Functional Requirements

REQ-1: The Sportify system shall query the weather APIs to collect the weather information.

> **1.1** The Sportify system shall query the 2-hour Weather Forecast API to collect the weather forecast.
> **1.2** The Sportify system shall query the Ultraviolet Index (UVI) API to collect the UVI information.
> **1.3** The Sportify system shall query the 2-hour Pollutant Standards Index (PSI) API to collect the PSI information.
> **1.4** The Sportify system shall query the Air Temperature API to collect the air temperatures (Degree Celsius).
> **1.5** The Sportify system shall query the Rainfall API to collect the real-time rainfall level in mm.

REQ-2: The Sportify system shall query the Google Maps API to collect the routing information on the preferred mode of transport.
REQ-3: The Sportify system shall display the data collected from the API and routing information on the location detail page

4.6 Calculate Scores

4.6.1 Description and Priority

The “Calculate Scores” feature will calculate the final score for all locations within the search radius based on weather conditions and the distance from the user’s location to the sports facilities. This feature is of high priority since this feature is responsible for scoring each location and it will be used to rank the location in descending order.

4.6.2 Stimulus/Response Sequences

SEQ-1: Sportify uses the included use case “Get API Values' to get the PSI, rainfall, UVI, air temperature and distance by specified travel mode of a location within the search radius.

SEQ-2: Sportify uses the retrieved values to generate an overall score out of 100 for each sport location within the search radius.

4.6.3 Functional Requirements

REQ-1: Sportify system shall calculate the linear distance from the user’s current location to the sports location

1.1 Sportify system shall assign a score to the linear distance calculated

REQ-2: Sportify system shall query weather APIs to retrieve environmental data of the sports location

2.1 Sportify system shall categorized the data retrieved from the API and assign it a score

REQ-3: Sportify system shall calculate the final score by tallying up the score from the distance and weather APIs

3.1 Sportify system shall assign 20% each to both Air Temperature and PSI score and 60% to linear distance score if the sports location is indoor

3.1 Sportify system shall assign 10% each to Air Temperature, UVI, rainfall and PSI score and 60% to linear distance score if the sports location is outdoor

4.7 Check-in/Check-out

4.7.1 Description and Priority

The “Check-in/Check-out” feature will allow users to pre-check-in, pre-check-out, check-in, check-out and playing to the sports location. This feature is of high priority since this feature provides user information on how crowded the sports location will be.

4.7.2 Stimulus/Response Sequences

SEQ-1: User will click “Pre-Check-In” button to pre-check-in at the sports location

SEQ-2: User will click “Pre-Check-Out” button to pre-check-out from the sports location

SEQ-3: User will click “Check-In” button to Check-In at the sports location

SEQ-4: User will click “Playing” button to indicate they have found players at the sports location

SEQ-5: User will click “Check-Out” button to check-out from the sports location

To Pre-Check-Out:

SEQ-1 -> SEQ-2

To Check-In:

SEQ-1 -> SEQ-3

To Playing:

SEQ-1 -> SEQ-3-> SEQ-4

To Check-Out:

SEQ-1 -> SEQ-3->SEQ-5

OR

SEQ-1 -> SEQ-3-> SEQ-4->SEQ-5

4.7.3 Functional Requirements

REQ-1: User shall click the relevant buttons to fulfill the check in/check-out process.

1.1 The Sportify shall display the current status of the user.

1.1.1 Default current status shall display “Not Pre-Checked-In”

1.2 The Sportify shall display the relevant buttons of each status.

1.2.1 Default status shall display the “Pre-check-in” button.

1.2.2 “Pre-checked-in” status shall display the “Check-in” button and the “Pre-check-out” button.

1.2.3 “Checked-in” status shall display the “Playing” button and “Check-out” button.

1.2.4 “Playing” status shall display the “Check-out” button.

1.3 User must click the “Pre-check-in” button to indicate their interest in going to a sports facility.

1.3.1 Sportify shall update current status of the user to Pre-Checked-in

1.3.2 User must click the “Pre-check-out” button if they are no longer interested.

1.3.3 Sportify shall automatically cancel the “pre-check-in” status after 120 minutes if the check-in process is not done.

1.4 User shall click the “check in” button to do check in. This indicates that the user is already at the sports location.

1.4.1 Sportify shall update current status of the user to Checked-In

1.4.2 User must click the “check out” button once they are leaving the sports facility.

1.4.3 User must click the “playing” button once found people to play with

1.4.3 Sportify will automatically perform check-out after 2 hours if the checkout is not done.

1.5 User shall click the “Playing” button to indicate that the user is currently playing that sport.

1.5.1 Sportify shall update current status of the user to Playing

1.6 User shall click the “Check-out” button to indicate they are leaving the sports facility.

1.7 User shall pre-check-in for only one location at a time.

1.7.1 Sportify shall indicate error if user attempts to pre-check-in while in the following statuses: “Pre-checked-in”, “Checked-in”, “Playing”

4.8 Feedback

4.8.1 Description and Priority

The “Feedback” feature allows the user to report issues or send feedback to improve Sportify. Users will include their email address and the content of the feedback. This feature is of low priority as it does not affect the main functionality of Sportify.

4.8.2 Stimulus/Response Sequences

SEQ-1: User will click “Help” button at the Top Navigation Bar
SEQ-2: User will enter their email address and the content of their feedback
SEQ-3: User will click “Submit” button to submit their feedback

4.8.3 Functional Requirements

REQ-1: User will click “Help” button to submit a feedback

1.1 User will enter a valid email address

1.2 User will include the content of their feedback

REQ-2: Sportify will display “Feedback Sent!” and store the feedback details in the database after the user presses the “Submit” button.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

1. Landing page must be fully loaded within 3 seconds on a desktop and 9 seconds on a mobile phone.
2. The system must be able to support 100,000 number of users concurrently while maintaining optimal performance
3. The system must have a response time between 200ms and 1s
4. The APIs must have a response time less than 0.1 second
5. Search results must be displayed within 3 seconds
6. Find out more page must be displayed within 1 second

5.2 Safety Requirements

To maintain data integrity in the system, we can conduct the following practices:

1. Data Encryption: Encrypt sensitive information retrieved from users by implementing SSL/TLS during transmission and disk encryption at rest. Additionally we can encrypt our databases as well.
2. Regular data backups: To be prepared for data restoration in case of data loss or corruption.
3. Error Handling: Implement exception handling to catch and rectify data errors.

5.3 Security Requirements

1. Location details retrieved from users must be encrypted and protected to prevent outsiders from using it for other purposes.
2. Regular penetration test must be conducted to ensure security of user information is up to date with current cyber attacks

5.4 Software Quality Attributes

5.4.1 Reliability:

1. System shall have an uptime of 99.9%
2. Accurately represent environmental condition across major regions in Singapore
3. Information provided shall be kept current and up-to-date

4. Accurately detect user's location

5.4.2 Availability

Sportify shall be available 24/7

5.4.3 Maintainability

Maintenance that requires downtime of Sportify would ideally be done during non-peak hours to a maximum number of 9 hours per year, inline with our targeted uptime of 99%

5.4.4 Compatibility

1. Sportify shall work on different operating systems. E.g Windows, Apple, Linux, Android, etc.
2. Sportify shall be compatible with commonly used devices with GPS and browser. E.g. Laptop, Mobile Phone, tablet etc.

5.4.5 Usability

Sportify can include filter option to filter out different Mode of Transport.

5.4.6 Scalability

1. Sportify shall be built on framework that allows easy updates to include more niche sports and environmental conditions to cater to as many users as possible
2. Sportify shall be able to handle large amount of users during peak hours without compromising performance and reliability

5.5 Business Rules

Users:

1. Users can enter their current location or automatically let the system detect their current location using GPS.
2. Users can filter their preferred radius distance and mode of transport
3. Users can do pre-check-in
4. Users can do pre-check-out after pre-check-in
5. User can do check-in after pre-check-in
6. User can do playing after check-in
7. User can check-out after check-in

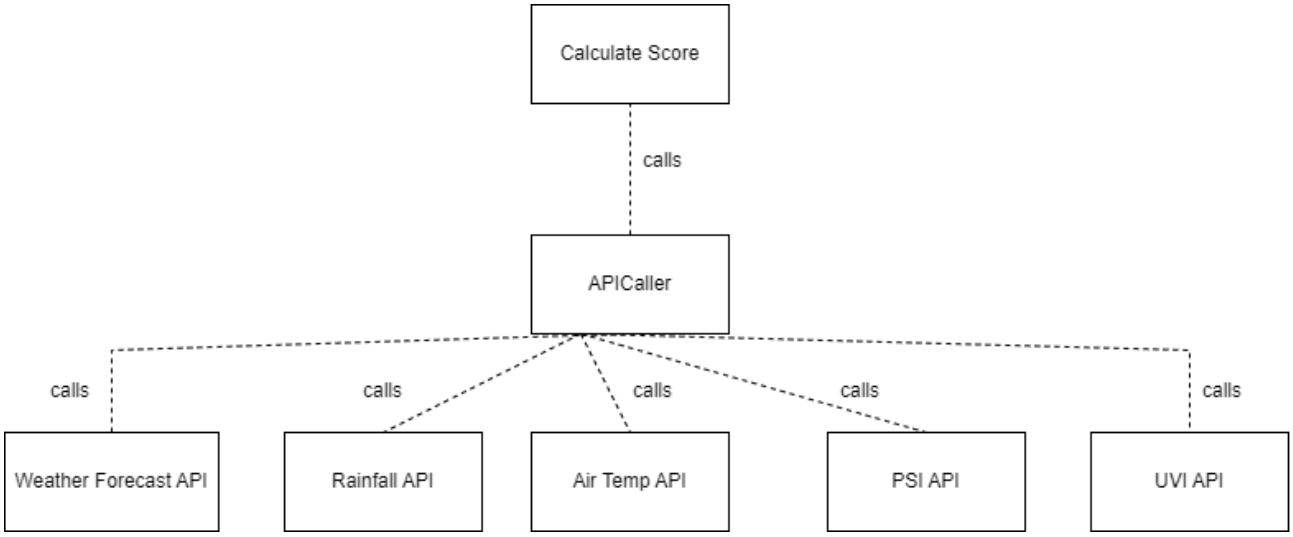
5.6 Design Patterns

5.6.1 Facade Pattern

Sportify relies on facade pattern to simplify our complex backend system. Instead of having a mess of different functions scattered throughout our code base, we centralized all our API interactions into one unified interface known as APICaller. This allows our components to extract the necessary values required for a location through this backend interface without having to individually locate and call each API.

Furthermore, our calculators are able to provide a score by simply passing in a location as a parameter, without the need of inputting each of the location's weather values as the extraction of these values are done internally, shielding the backend from further complexities. All in all the heavy use of facade pattern in our code has allowed us to decouple the information relay between our controllers in the backend.

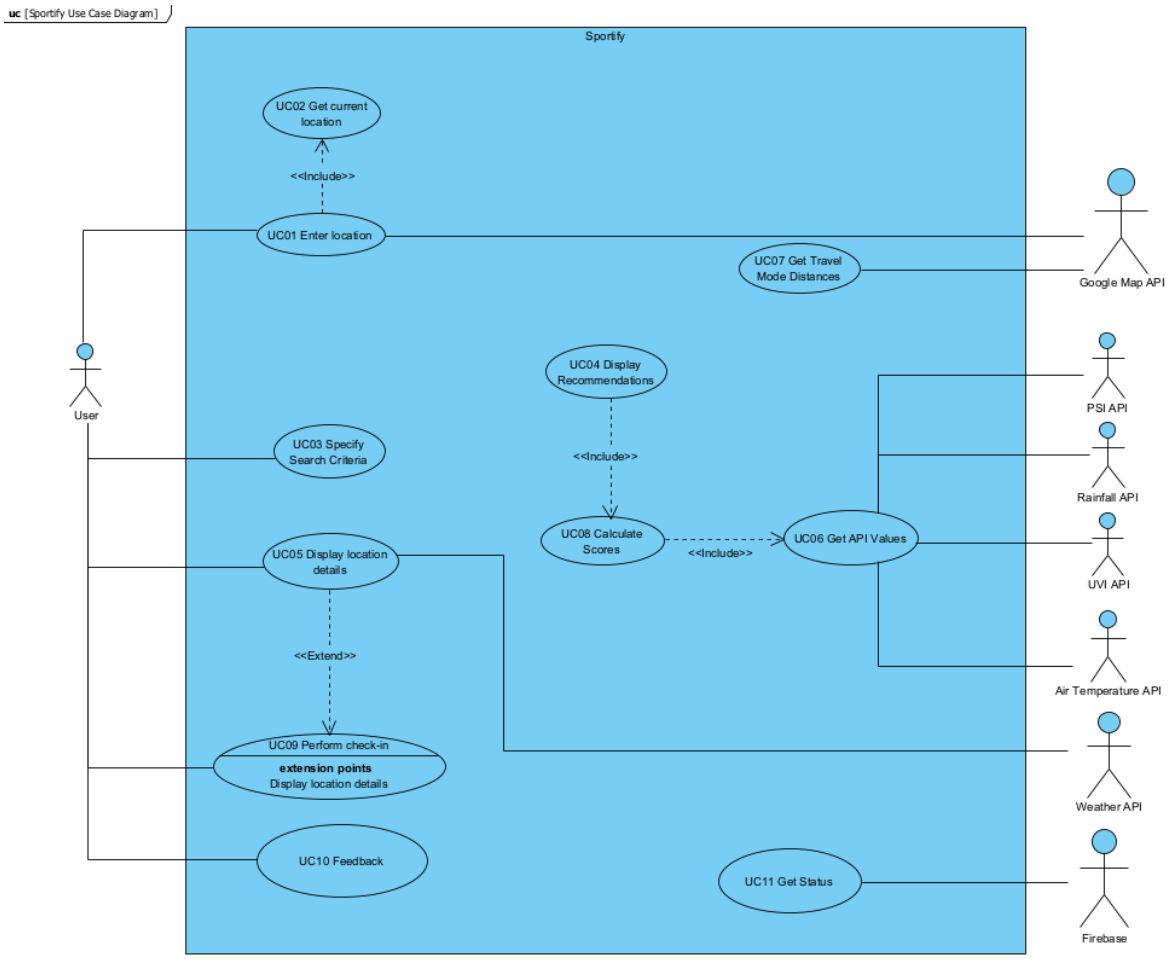
```
const CalculateScores = async (displayData,ori)=>{
    const overallScores = {};

    const WEATHER_WEIGHTAGE = 0.4;
    const DISTANCE_WEIGHTAGE = 1 - WEATHER_WEIGHTAGE;
    const [distances, minDistance] = await CalculateDistance(displayData,ori);
    const distanceScores = CalCulateDistanceScore(distances,minDistance);
    const distanceScoresIndex = Object.keys(distanceScores);

    const apiCaller = new APICaller()
    const airData = await apiCaller.fetchAirReadings()
    const psiData = await apiCaller.fetchPSIReadings()
    const rainfallData = await apiCaller.fetchRainfallReadings()
    const UVIData = await apiCaller.fetchUVIReadings()
    for(let element of displayData) {
        const weatherScore = await CalculateWeatherScore(element, airData, psiData, rainfallData, UVIData);
        if (element && distanceScoresIndex.length !== 0 && distanceScoresIndex.includes(element.index)){
            overallScores[element.index] = (distanceScores[element.index] * DISTANCE_WEIGHTAGE) + (weatherScore* WEATHER_WEIGHTAGE)
        }
        else{
            console.log("cannot calculate overall scores!")
        }
    }

return overallScores;
}
```



5.6.2 Singleton Pattern

Sportify utilizes Firebase, which is an external Google web app development platform, to store our user's statuses and feedbacks. This ensures that during the runtime of our application, our components would only be referencing one existing copy of the user's statuses. Using this one centralized server storage system, Sportify components are able to access the required data whenever necessary and also allows changes to the data to persist across page reloads and browser sessions.

6. Use Case Diagram



7. Use Case Description

Use Case ID:	UC01		
Use Case Name:	Enter location		
Created By:	Ding Ren	Last Updated By:	Ding Ren
Date Created:	8/2/24	Date Last Updated:	8/2/24

Actor:	User & Google Map API
Description:	User can set specific location used as departure location and basis of search radius
Preconditions:	1. User has not specified a location
Postconditions:	1. Map reflects marker on specified location
Priority:	
Frequency of Use:	1. Everytime a new search for a new location is made.
Flow of Events:	1. User selects search bar 2. User enters name of location 3. User can select location from a dropdown list of recommendations obtained from Google Map API.
Alternative Flows:	AF-S3: 1. User can select the current location button to use the included use case “Get current location” to use the user's current location by GPS as specified user’s location.
Exceptions:	EX1: Location specified not within Singapore 1. Sportify return empty location dropdown list
Includes:	UC02
Special Requirements:	API response time must be 0.1s which can be counted as immediate response where users would not feel any interruption.
Assumptions:	
Notes and Issues:	

Use Case ID:	UC02		
Use Case Name:	Get current location		
Created By:	Ding Ren	Last Updated By:	Ding Ren
Date Created:	8/2/24	Date Last Updated:	8/2/24

Actor:	
Description:	Obtains the current location of the user to use as the user's specified location.
Preconditions:	
Postconditions:	1. Sportify now selects user’s current location as departure location. 2. Sportify marks the current location of the user on the map.
Priority:	
Frequency of Use:	1. Everytime user wishes to use their current location as departure location.
Flow of Events:	3. Sportify prompts users to allow use of the device's GPS to know current location. 4. Sportify returns the current location of user.
Alternative Flows:	AF-S1: User rejects use of GPS 1. “Current Location” button no longer works. 2. User can only select address through the search bar.
Exceptions:	
Includes:	
Special Requirements:	User is in a location where GPS signal is available.
Assumptions:	
Notes and Issues:	

Use Case ID:	UC03		
Use Case Name:	Specify Search Criteria		
Created By:	Randall	Last Updated By:	Randall
Date Created:	8/2/24	Date Last Updated:	8/2/24

Actor:	User
Description:	User specifies the search radius and preferred modes of transport.
Preconditions:	1. User has specified a location.
Postconditions:	
Priority:	
Frequency of Use:	1. Everytime user wishes to change the search radius and modes of transport.
Flow of Events:	2. Upon entering location or getting current location, user can select the radius button to adjust the search radius. 3. Users can shift the slider to desired radius. 4. Upon selecting the search radius button, the user can select a preferred mode of travel. 5. Users must select at least one mode of travel before proceeding to submit a query.
Alternative Flows:	
Exceptions:	EX1: User didn't specify search criteria 1. Sportify prompt user to set search criteria. EX2: User didn't specify mode of transport 1. Sportify prompts user to select mode of transport. EX3: User didn't specify location before specifying search criteria 1. Sportify prompts user to specify a departure location.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	UC04		
Use Case Name:	Display Recommendations		
Created By:	Steven	Last Updated By:	Steven
Date Created:	8/2/24	Date Last Updated:	8/2/24

Actor:	
Description:	Displays recommended sports locations ranked according to their overall scores.
Preconditions:	
Postconditions:	
Priority:	
Frequency of Use:	1. Everytime new query is made.
Flow of Events:	1. Sportify uses the included use case “Calculate Scores” to get overall score for all sports facilities within the search radius. 2. Sportify ranks them in descending order. 3. Sportify displays them according to their ranks.Spotify shows all available locations in the search radius including: a. Picture of location b. Name of sport location c. Address of sport location d. Type of activity available in sport location e. Distance of sport location from user specified location f. Location fees/charges
Alternative Flows:	
Exceptions:	EX1: No locations found in search radius 1. Sportify prompts user with message “No location!” at home page.
Includes:	UC08
Special Requirements:	Requires data from other use cases.
Assumptions:	All previous use cases are correct and provide accurate information.
Notes and Issues:	

Use Case ID:	UC05		
Use Case Name:	Display location details		
Created By:	Randall	Last Updated By:	Randall
Date Created:	8/2/24	Date Last Updated:	8/2/24

Actor:	User
Description:	User specifies the search radius.
Preconditions:	1. User has submitted a search 2. Sportify displays recommendations. 3. User has selected the “Find Out More” button for a sports location from the result page.
Postconditions:	
Priority:	
Frequency of Use:	1. Everytime user selects “Find out more” button of a sports location
Flow of Events:	1. Sportify gets weather condition of sport location from Weather API. 2. User presses find out more button of sport location to show: a. Playing count b. Check-In count c. Pre-check-in count d. Air temperature e. Weather condition f. Rain g. PSI h. UVI
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	API response time must be 0.1s which can be counted as immediate response where users would not feel any interruption.
Assumptions:	
Notes and Issues:	

Use Case ID:	UC06		
Use Case Name:	Get API values		
Created By:	Ding Ren	Last Updated By:	Ding Ren
Date Created:	8/2/24	Date Last Updated:	8/2/24

Actor:	PSI API
Description:	Obtains and collates PSI values for the different areas of Singapore.
Preconditions:	1. User has specified location. 2. User has specified search-radius. 3. User has specified modes of travel. 4. User has selected search button to show recommendations result.
Postconditions:	
Priority:	
Frequency of Use:	1. Everytime a new search is made.
Flow of Events:	1. Sportify queries the PSI API, Air Temperature API, Weather API, UVI API and Rainfall API to get their respective values from areas across Singapore. 2. Sportify finds out the area where values are taken from at which the location is closest to. 3. Sportify returns the values from the identified area to take reference from for location.
Alternative Flows:	
Exceptions:	EX1: If any of the APIs does not return a result 1. Replace any of the weather quantities with NaN if that API does not return a result.
Includes:	
Special Requirements:	API response time must be 0.1s which can be counted as immediate response where users would not feel any interruption.
Assumptions:	
Notes and Issues:	

Use Case ID:	UC07		
Use Case Name:	Get Travel Mode		
Created By:	Ding Ren	Last Updated By:	Ding Ren
Date Created:	8/2/24	Date Last Updated:	8/2/24

Actor:	Google Map API
Description:	Obtain and collate the travel distance to locations within specified search radius from specified location based on the user selected mode of travel.
Preconditions:	1. User has specified location. 2. User has specified search-radius. 3. User has specified mode of travel. 4. User has selected search button to show recommendations result. 5. User has selected Find Out More button from one of the sports location in the recommendations.
Postconditions:	
Priority:	
Frequency of Use:	1. Everytime a new search is made.
Flow of Events:	1. Sportify queries the Google Map API to get the most optimized routes to location based on the user's selected mode of travel. 2. The Google Map API returns the routes of location in search radius from user’s specified location.
Alternative Flows:	
Exceptions:	EX1: If any of the APIs does not return a result. 1. Replace any distance not returned with NaN.
Includes:	
Special Requirements:	API response time must be 0.1s which can be counted as immediate response where users would not feel any interruption.
Assumptions:	
Notes and Issues:	

Use Case ID:	UC08		
Use Case Name:	Calculate Scores		
Created By:	Ding Ren	Last Updated By:	Ding Ren
Date Created:	8/2/24	Date Last Updated:	8/2/24

Actor:	
Description:	Calculates score for every sport location located within the specified search radius for Sportify to rank.
Preconditions:	1. Sport locations within search radius have been filtered. 2. User has specified a departure location, search radius and mode of transport. 3. Use has selected the search button to show recommendation results.
Postconditions:	
Priority:	
Frequency of Use:	1. Everytime a new search is made.
Flow of Events:	1. Sportify uses the included use case “Get API Values” to get the PSI, rainfall, UVI and air temperatures. 2. Sportify uses the retrieved values to generate an overall score out of 100 for each sport location within the search radius.
Alternative Flows:	
Exceptions:	
Includes:	UC06
Special Requirements:	API response time must be 0.1s which can be counted as immediate response where users would not feel any interruption.
Assumptions:	
Notes and Issues:	

Use Case ID:	UC09		
Use Case Name:	Perform check-in		
Created By:	Ding Ren	Last Updated By:	Ding Ren
Date Created:	19/2/24	Date Last Updated:	19/2/24

Actor:	User
Description:	Allows user to inform others that they have arrived and is currently at the sport facility. This adds on to a live-count of the number of people present at the sport facility but has yet started playing or has yet to find a playing group or has yet to find a playing partner.
Preconditions:	1. Status of user must be “Pre-checked-in” 2. User not in any of the following statuses: “Default”, “Checked-in”, “Playing”
Postconditions:	1. Changes user status to “Checked-in” 2. Increases the counter for Check-in at that sport facility 3. Decreases the counter for Pre-check-in at that sport facility
Priority:	
Frequency of Use:	Everytime the “Check-in” button on the subpage of a sport facility is pressed
Flow of Events:	1. User presses “Pre Check-in” button 2. Sportify checks if the status of user is “Pre-checked-in” 3. If no, Sportify changes user status to “Pre checked-in” 4. Sportify increments the counter for pre-check-in at that sport facility by 1 5. User presses “Check-in” button 6. Sportify checks if the status of user is “Pre Checked-in” 1. If yes, Sportify changes user status to “Checked-in” 2. Sportify decrements counter for pre-check-in at sport location by 1 and increment counter for check-in at sport location by 1 3. User presses “Playing” button 4. Spotify changes user status to “Playing”. 5. Sportify starts timer of 180 minutes for user. 6. User can press “Check-out” button manually while status is checked-in to change user status to checked-out

	7. Sportify decrements counter for check-in by 1
Alternative Flows:	AF-S2: User status is not “pre-checked-in” 1. Sportify displays error message “ Please perform pre-check-in first!” 1. Sportify prompts user to restart app. AF-S6: User presses “Check-out” button 1. Sportify changes user status to “Default” 2. Sportify decrements the counter for “Checked-in” at that sport facility by 1 AF-S6: Timer of 180 minutes is up 1. Sporify displays error message “ Check-in has expired! Playing time is up!” 2. Sportify changes user status to “Default” 3. Sportify decrements the counter for “Checked-in” at that sport facility by 1
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	All previous use cases are correct and provide accurate information.
Notes and Issues:	

Use Case ID:	UC10		
Use Case Name:	Feedback		
Created By:	Randall	Last Updated By:	Randall
Date Created:	8/2/24	Date Last Updated:	8/2/24

Actor:	User
Description:	User is able to provide feedback on Sportify.
Preconditions:	
Postconditions:	
Priority:	
Frequency of Use:	1. Everytime user wishes to leave a feedback.
Flow of Events:	1. User clicks on “Help” button on top navigation bar. 2. User keys in email. 3. User keys in content of feedback. 4. User clicks on “Send Feedback” 5. User is prompted that feedback has been successfully sent. 6. Feedback is sent to Sportify’s external Firebase server where it can be read.
Alternative Flows:	
Exceptions:	EX1: User leaves either email or feedback content blank upon sending feedback. 1. Sportify prevents sending of feedback. 2. Sportify prompts user to fill up both contents.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	UC11		
Use Case Name:	Get Status		
Created By:	Randall	Last Updated By:	Randall
Date Created:	8/2/24	Date Last Updated:	8/2/24

Actor:	Firebase
Description:	Sportify fetches feedback and user's status from Firebase server.
Preconditions:	1. User has submitted a search. 2. Sportify displays recommendations. 3. User has selected the "Find Out More" button for a sports location from the result page. 4. User has clicked on "Pre Check-in", "Pre Check-out", "Check-in", "Check-out" and "Playing" buttons and trigger a change in user status.
Postconditions:	
Priority:	
Frequency of Use:	1. Everytime user has triggered a change in user status.
Flow of Events:	1. Sportify fetches user's status from external Firebase server. 2. Sportify changes user status and sends new status back to external Firebase server.
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

8. Data Dictionary

Term	Definition
API	An application programming interface (API) is a way for two or more computer programs or components to communicate with each other. In this project, we will be using google maps, weather, ultraviolet(UV), air temperature and pollutant standard index (PSI) APIs, which is a set of protocols and tools that allow for the retrieval of the required data from various sources. It acts as an intermediary, enabling software applications to access real-time, forecasted, and historical weather information.
Sport	An athletic activity requiring skill or physical prowess and often of a competitive nature. In this project, when we refer to sports, we will be referring to Basketball, Football, Swimming, Running, Tennis, Weightlifting and Badminton.
Sports facilities	Sports Facilities means areas of sports pavilions, stadiums, gymnasiums, health spas, boxing arenas, swimming pools, roller and ice rinks, billiard halls, bowling alleys, and other similar places where members of the general public assemble to engage in physical exercise, participate in athletic competition, or witness sporting events. In this project, we will refer to sport facilities which are mainly used to engage in physical exercise.
Sports court	An outdoor or indoor asphalt court (not including parking lots) designed for athletic purposes (i.e. basketball court, tennis court, etc.) surrounded by fencing or on a standalone pad.
Sportify system	System that ranks all the sports that the user can do within the specified search radius, ranked in a descending order based on the data derived from the weather, rainfall, UVI, Google maps and PSI APIs
Default location	The current location of the user when they use Sportify. They can use the GPS on their device. This is the default option.
Departure Location	The location the user wishes to set as their location when they use Sportify. They can type down their location instead of using their GPS.

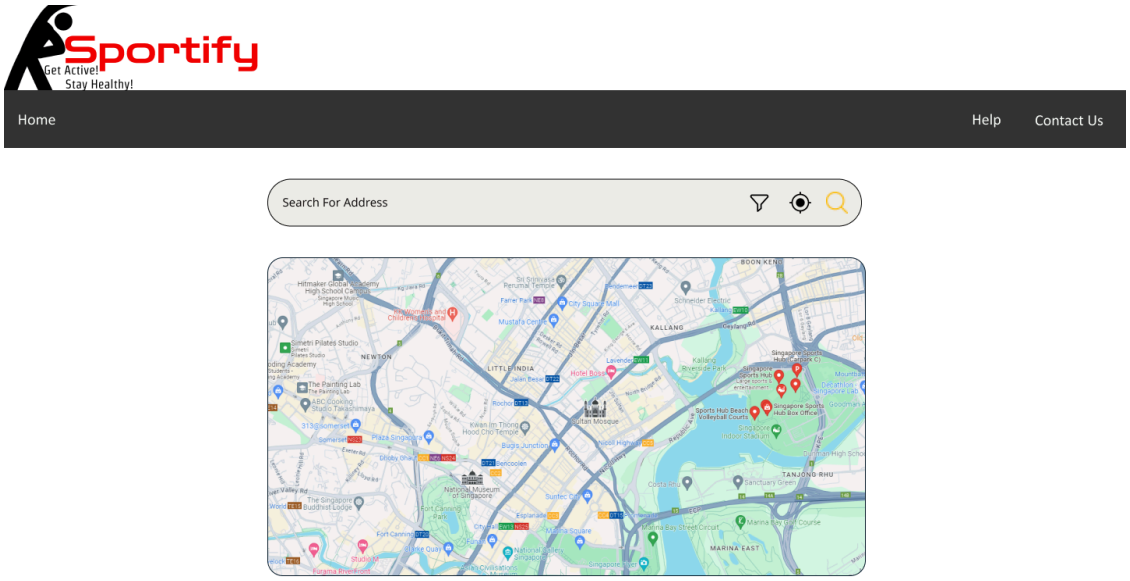
Weather forecast	The state of the atmosphere with respect to wind, temperature, cloudiness, moisture, pressure, etc. In this project, the weather forecast at the sports location will be classified into: 1. Clear 2. Partly Cloudy 3. Cloudy 4. Light Showers 5. Thundery Showers 6. Heavy Thundery Showers 7. Heavy 8. Showers
Rainfall	Accumulated rainfall within 5 minutes in the sports facilities or sports court area in mm. In this project, it will be classified into 2 category: 1. Yes : Raining, accumulated rainfall > 0 2. No : Not raining, accumulated rainfall = 0
UVI	Ultraviolet Index (UVI) is a measure of the level of UV radiation. The values of the index range from zero upward. The higher the UVI, the greater the potential for damage to the skin and eye, and the less time it takes for harm to occur. In this project, the current UVI value in Singapore will be used to determine what type of sport and location is the most suitable. It will be classified into: 1. 0 to 2: Safe to be outside 2. 3 to 7: Seek shade during midday hours, but the person can be outside if they take extra protection against the UV radiation 3. 8 and above: Going outside is not advised
PSI	The Pollutant Standards Index (PSI) is a type of air quality index used in Singapore, which is a number used to indicate the level of pollutants in air. In this project, the current PSI at the sports location will be used to determine what type of sport and location is the most suitable. It will be classified into: 1. 0 to 50: Good 2. 51 to 100: Moderate 3. 101 to 200: Unhealthy 4. 201 to 300: Very unhealthy 5. 300 and above: Hazardous
Thunderstorm	A transient storm of lightning and thunder, usually with rain and gusty winds, sometimes with hail or snow, produced by cumulonimbus

	clouds. In this project, we will be using boolean to indicate if there is a thunderstorm, with 0 being that there is a thunderstorm and 1 being that there is no thunderstorm.
Pre-check-in	The user must pre-check-in to indicate their interest in going to the sports facility. If the sport that the user wishes to play requires two or more people, such as badminton and basketball, the user will be able to use the pre check-in feature to see the number of people who plan on going to the same sports location,are also looking for another person to play the sport with and are on their way to the sports facility.
Pre-check-out	The user must use the Pre-check-out button to indicate if they are no longer interested in going to the sports facility after they have used the pre-check-in. If the check-in button is not asserted 90 minutes after the user asserted pre-check-in, the pre-check-in will be canceled automatically.
Check-in	If the sport that the user wishes to play requires two or more people, such as badminton and basketball, the user will be able to use the check-in feature to see the number of people who plan on playing the same sport,are also looking for another person to play the sport with and are currently in the sports facility. This will also enable the users to see the total number of people inside the court and decide whether the sports facility is full.
Playing	The user will assert this feature to indicate that they have found people to play their sports with. This feature will indicate the number of people who have found a person to play with as a ratio of the people who have checked in.
Check-out	Once the user leaves the sports facility, they can use the check-out feature to indicate that they are no longer within the sports facility. If the check-out feature is not used and three hours has passed since the check-in of the user, the check-out feature will be asserted automatically.
Status	The status of the user shows what they are doing currently. There are four statuses:Not Pre-Checked In Default, Pre-Checked In, Checked-In and Playing

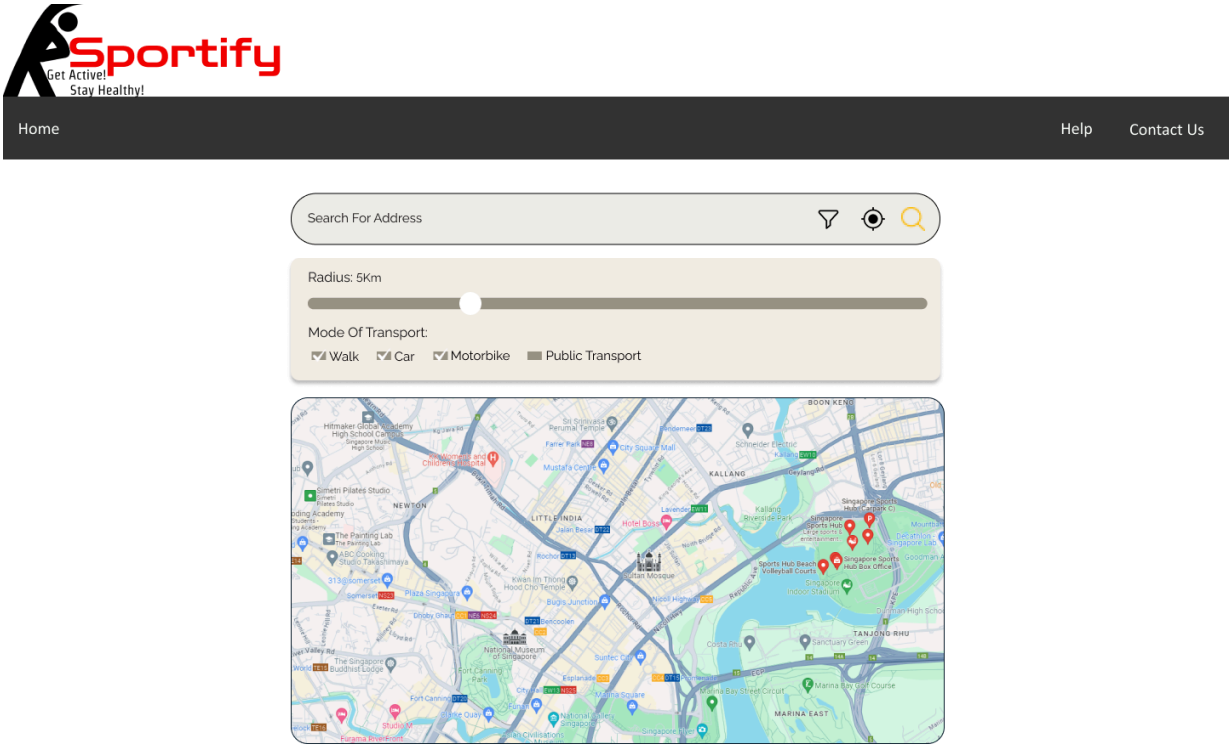
	'Not Pre-Checked In' is the status for when the user has just entered the webpage and not used the system. 'Pre-Checked In' status is achieved when the user asserts the pre-check-in button. 'Checked-In' status is achieved when the user asserts the check-in button 'Playing' status is achieved when the playing button is asserted.
Search radius	A straight line extending from the center of a circle or sphere to the circumference or surface. In this project, we will be using this radius, with the user's departure radius as the center, within which we will identify all the sports facilities we wish to consider.
Recommended sports list	The sports facilities within the radius specified by the user will have their scores calculated and ranked, based on weather conditions and user preferences.The sports facilities will be displayed in a descending order in terms of score, with the facility with the highest score being displayed first.
Final Score	Score derived from evaluating the ultraviolet index, pollutant standard index, rainfall amount, distance to the location of the sports facility based on the chosen mode of transport and air temperature from a scale of 1 to 100.
Mode of Transport	The user will have the option of choosing their mode of transport, from these following choices: ● Walk ● Car ● Motorbike ● Public Transport
Distance	Direct map distance from departure location to the sports location
Sports Location	Location of the sports facility

9. UI Mockups

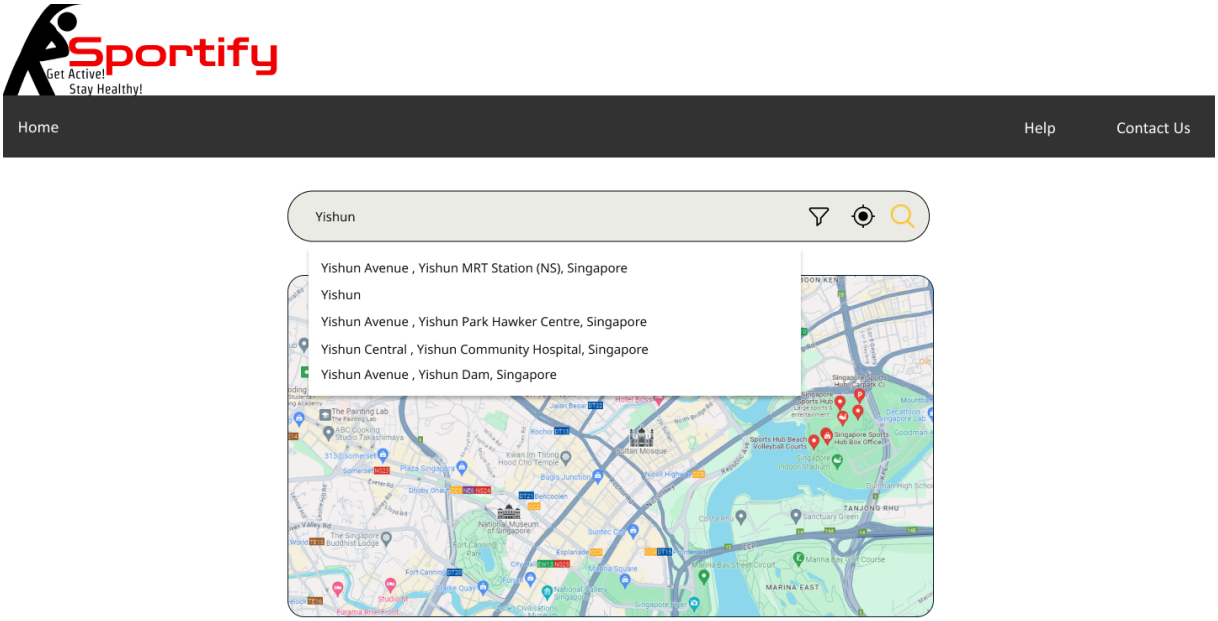
9.1 Home Page



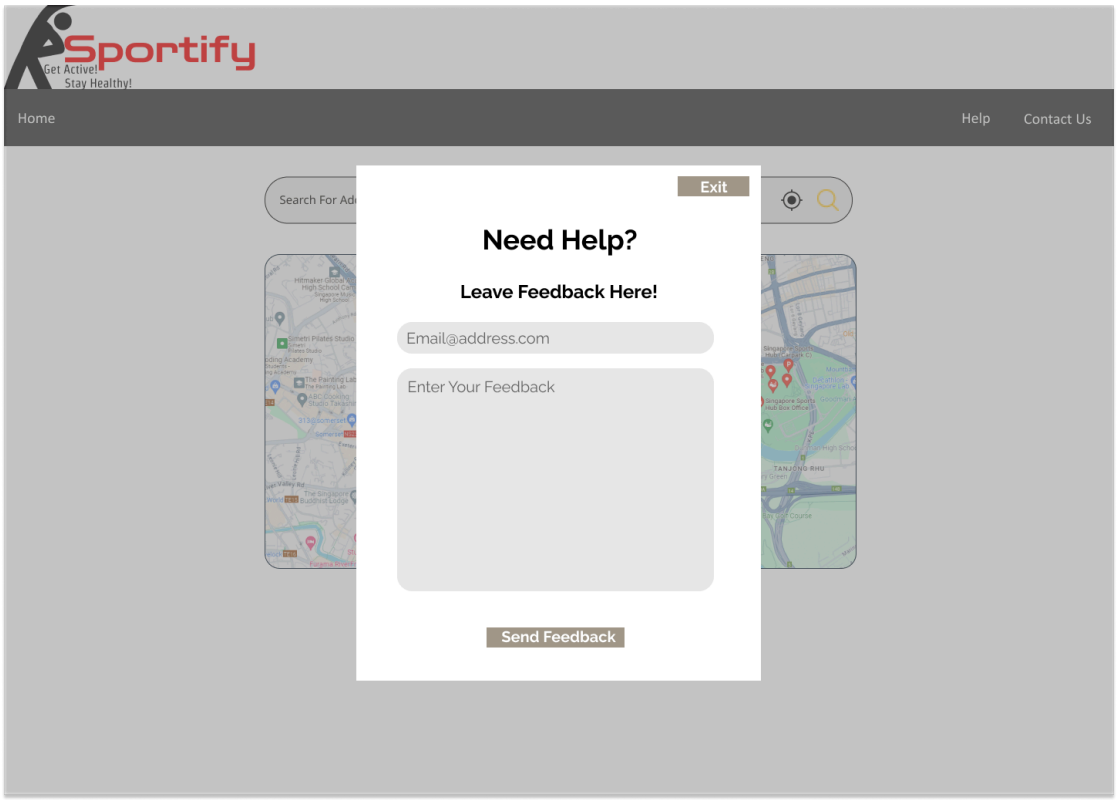
9.2 Home Page with Filter



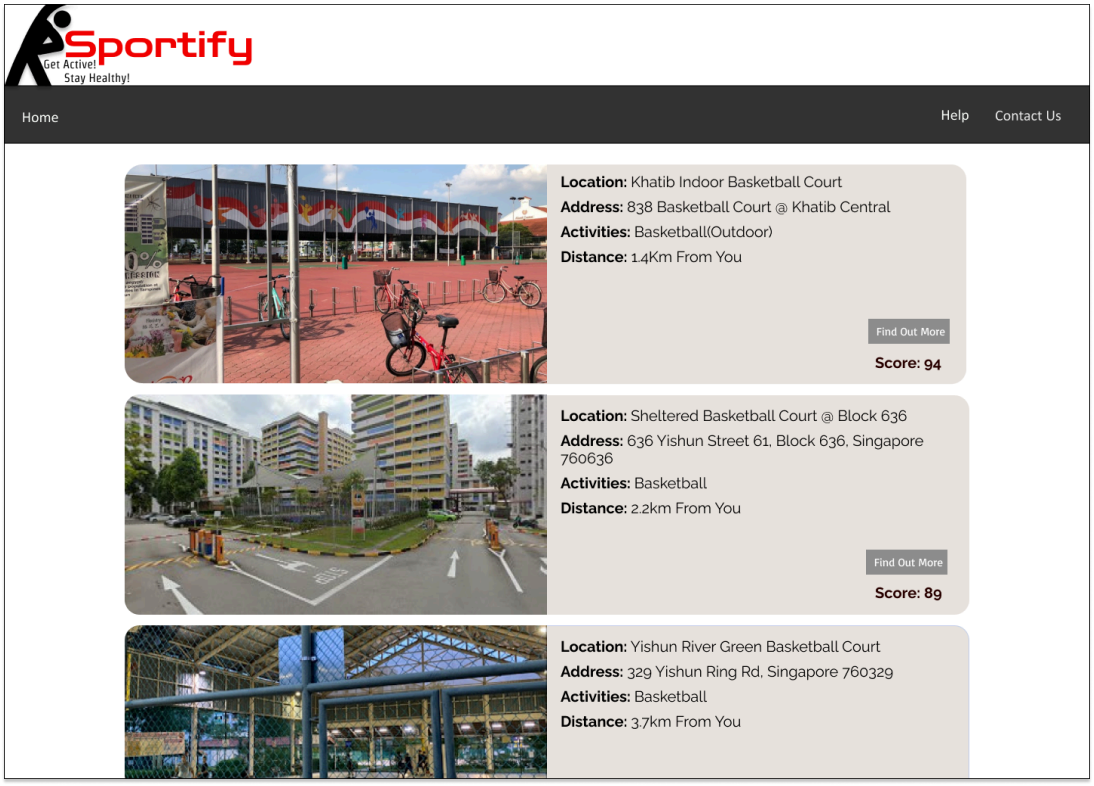
9.3 Home Page with Address List



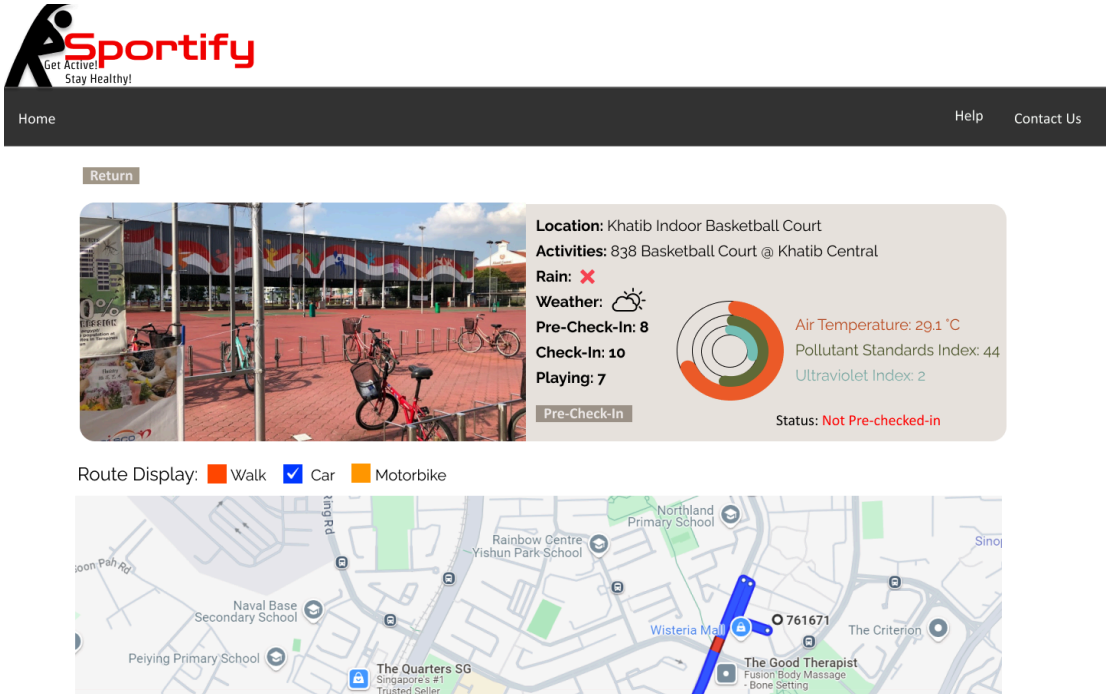
9.4 Help page



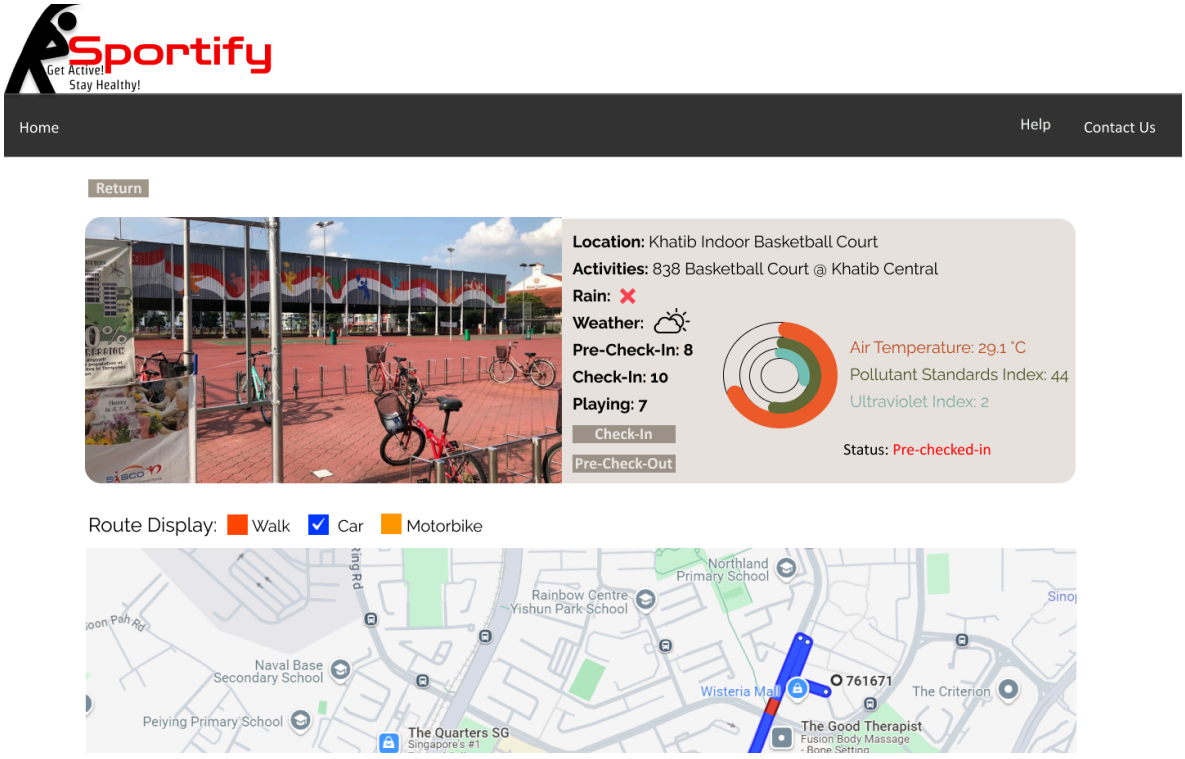
9.5 Result page



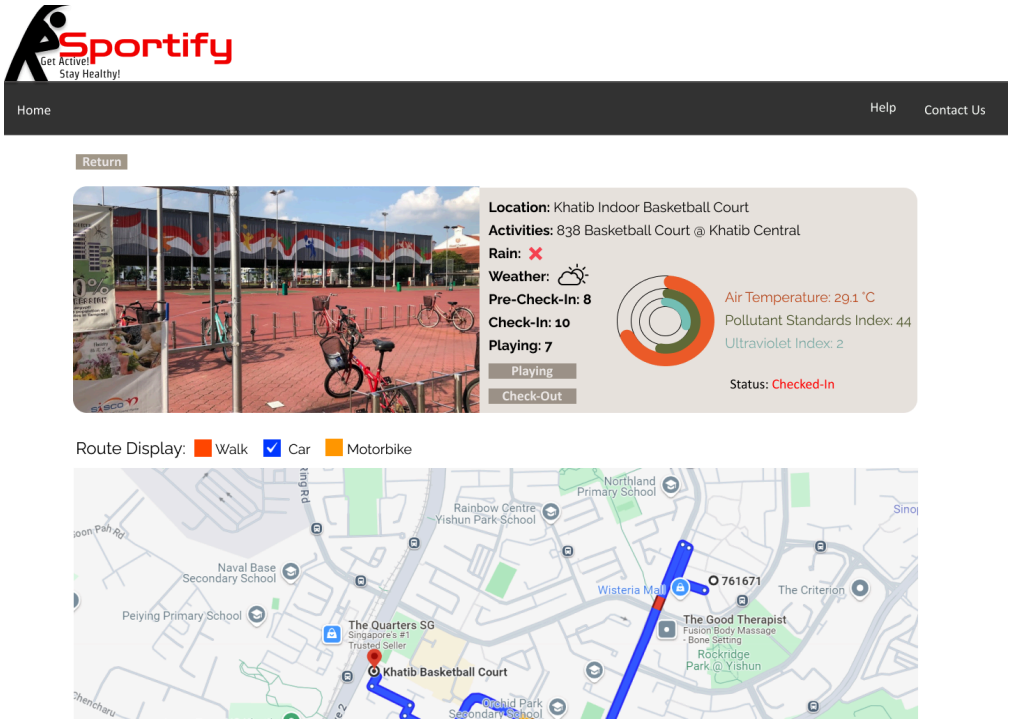
9.6 Find out more page



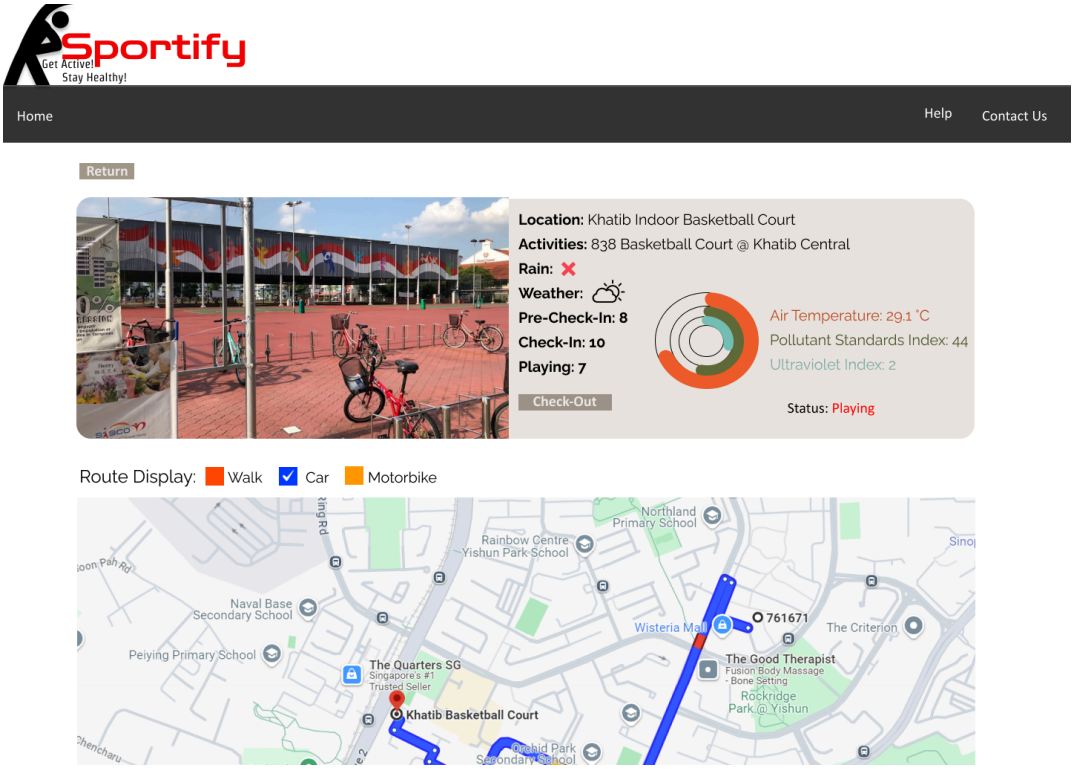
9.7 Check-in/Pre-Check-Out page



9.8 Playing/Check-Out page

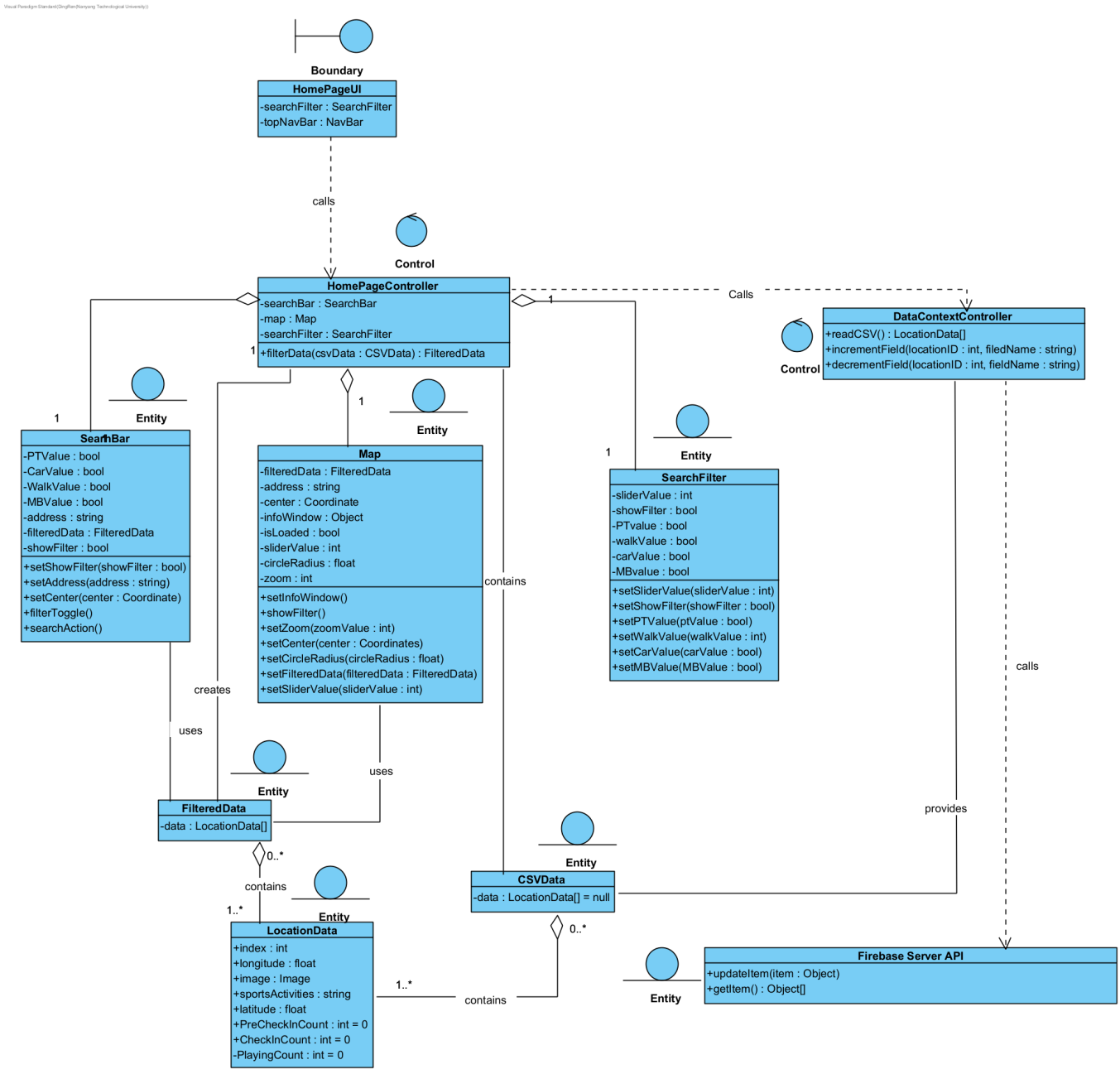


9.9 Check-Out Page

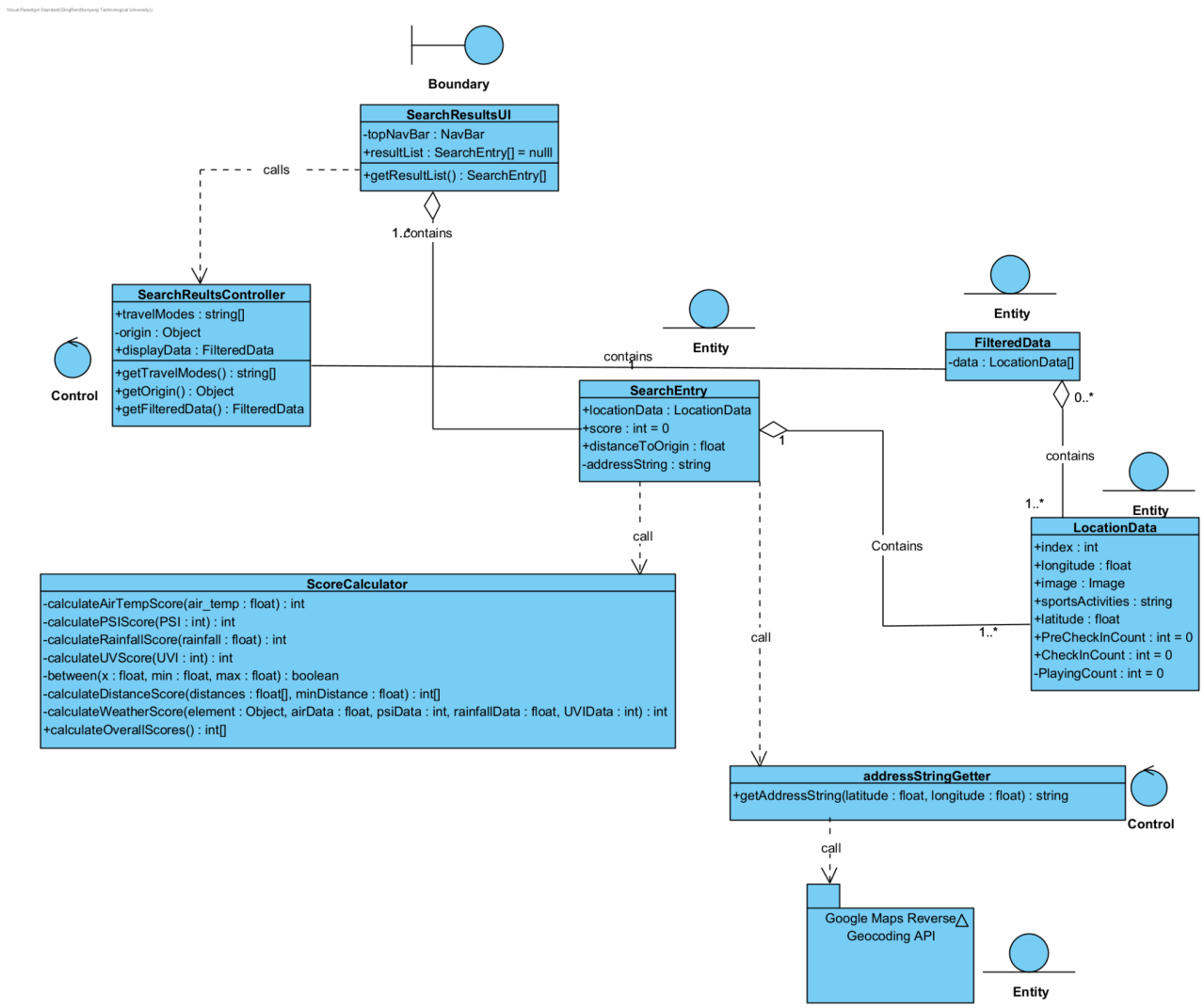


10. Class Diagrams

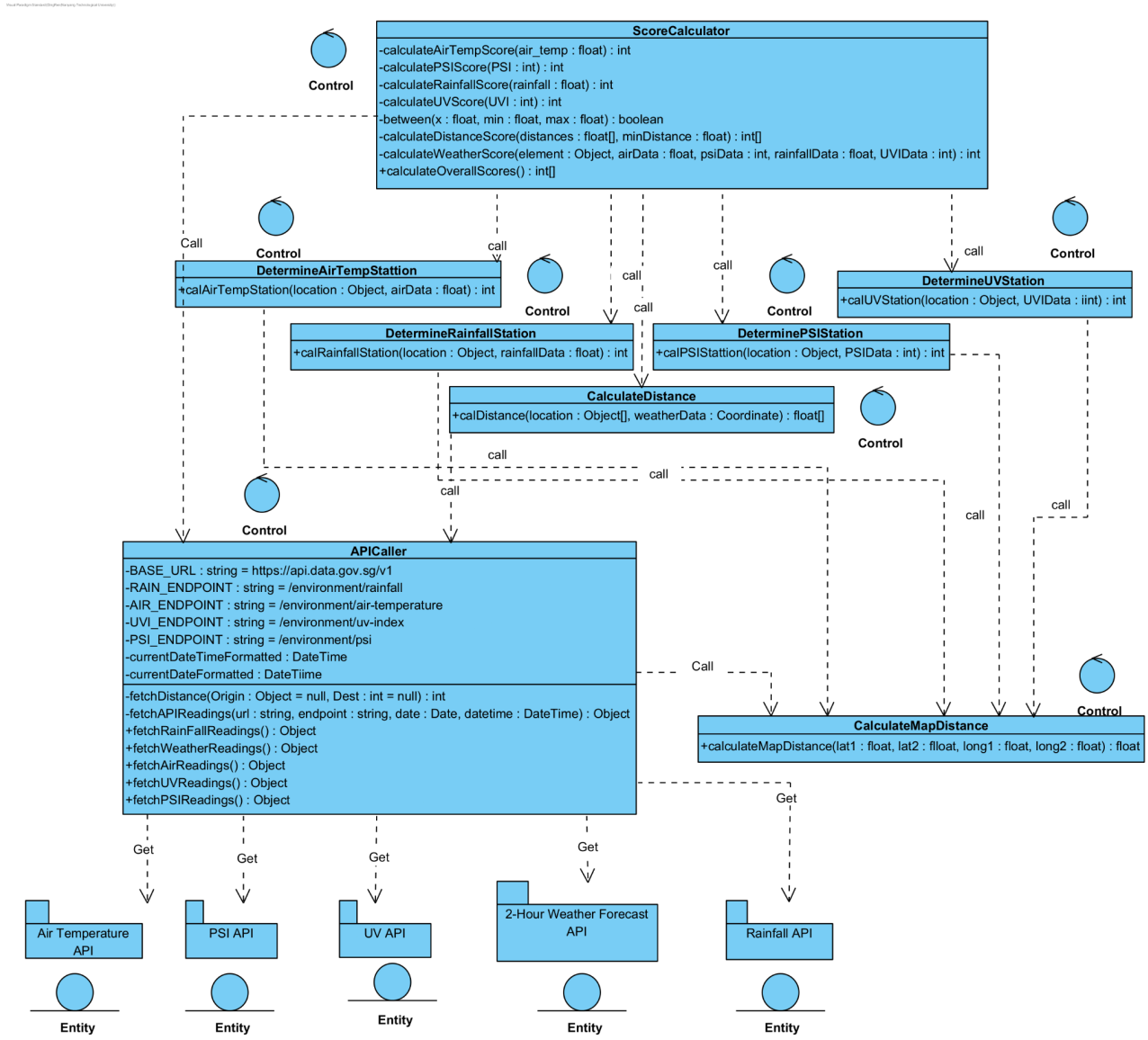
10.1 Home



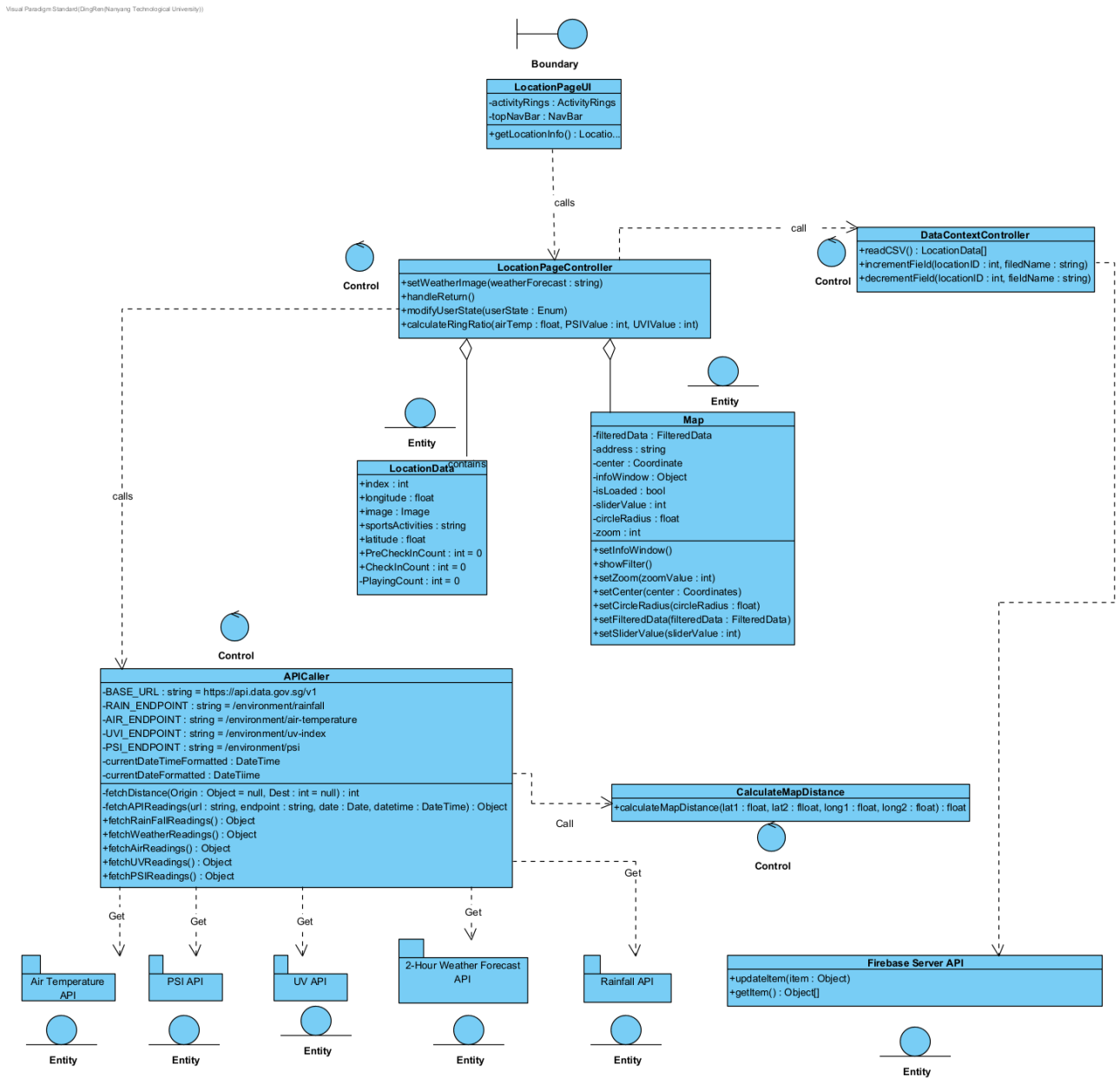
10.2 Search Results



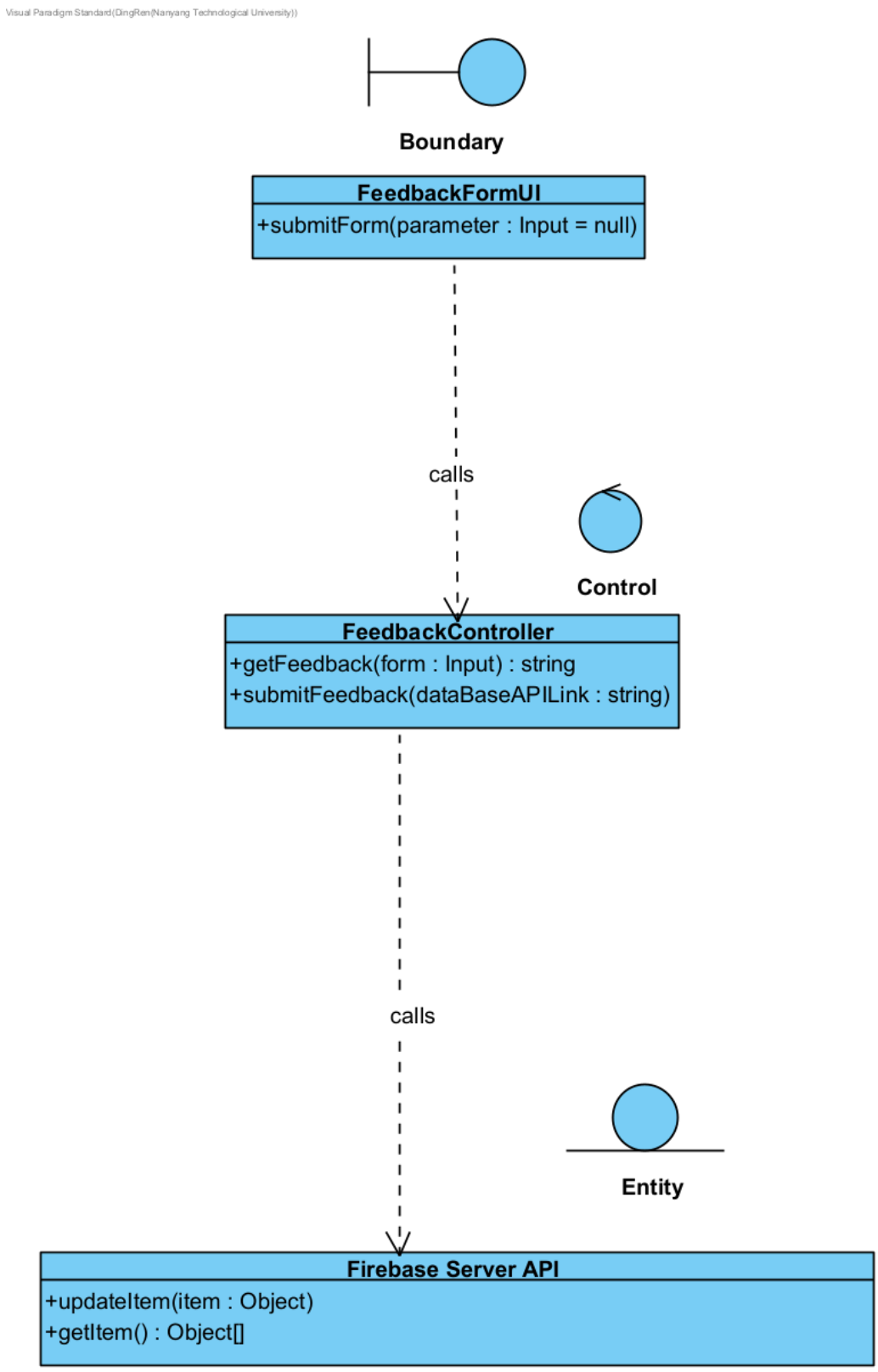
10.3 Calculate Score



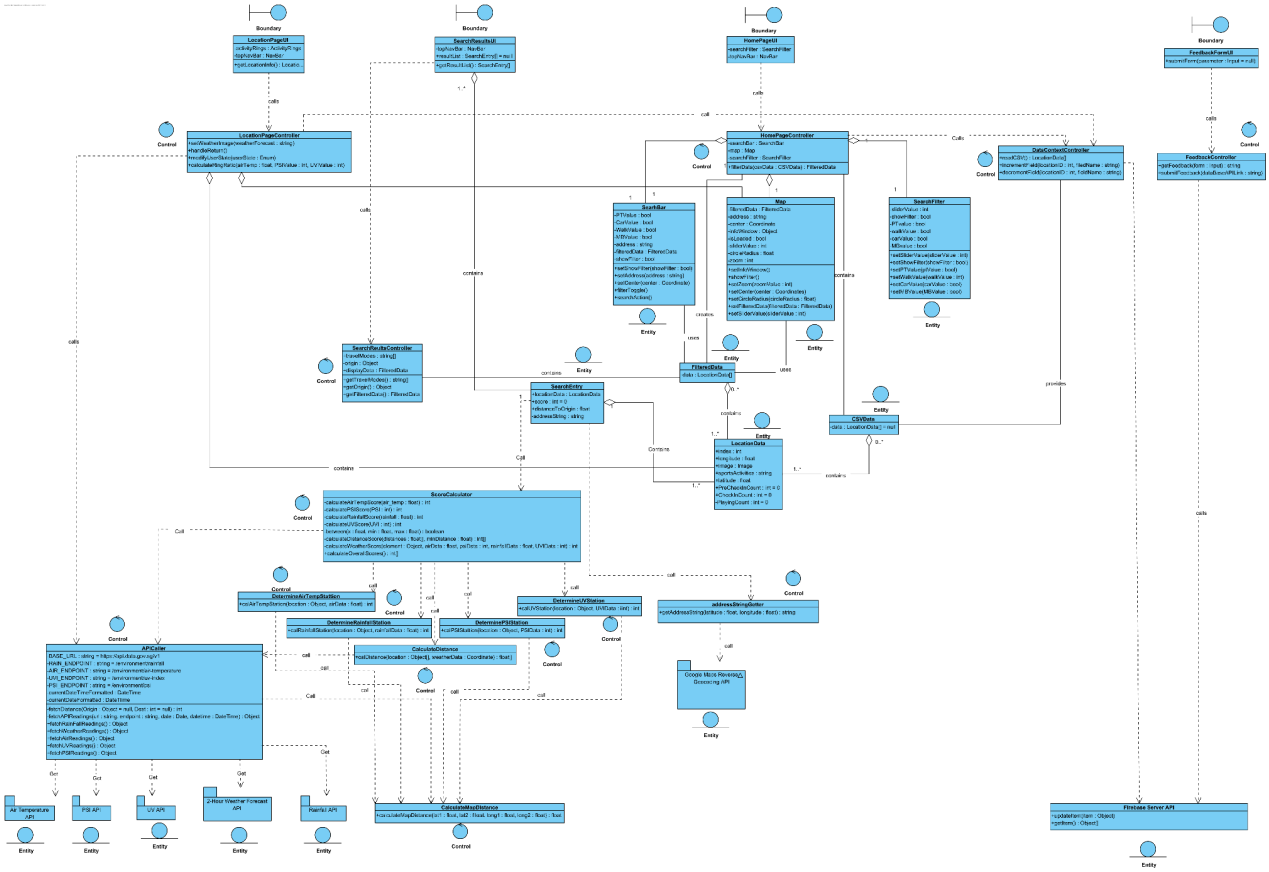
10.4 LocationPage



10.5 Feedback

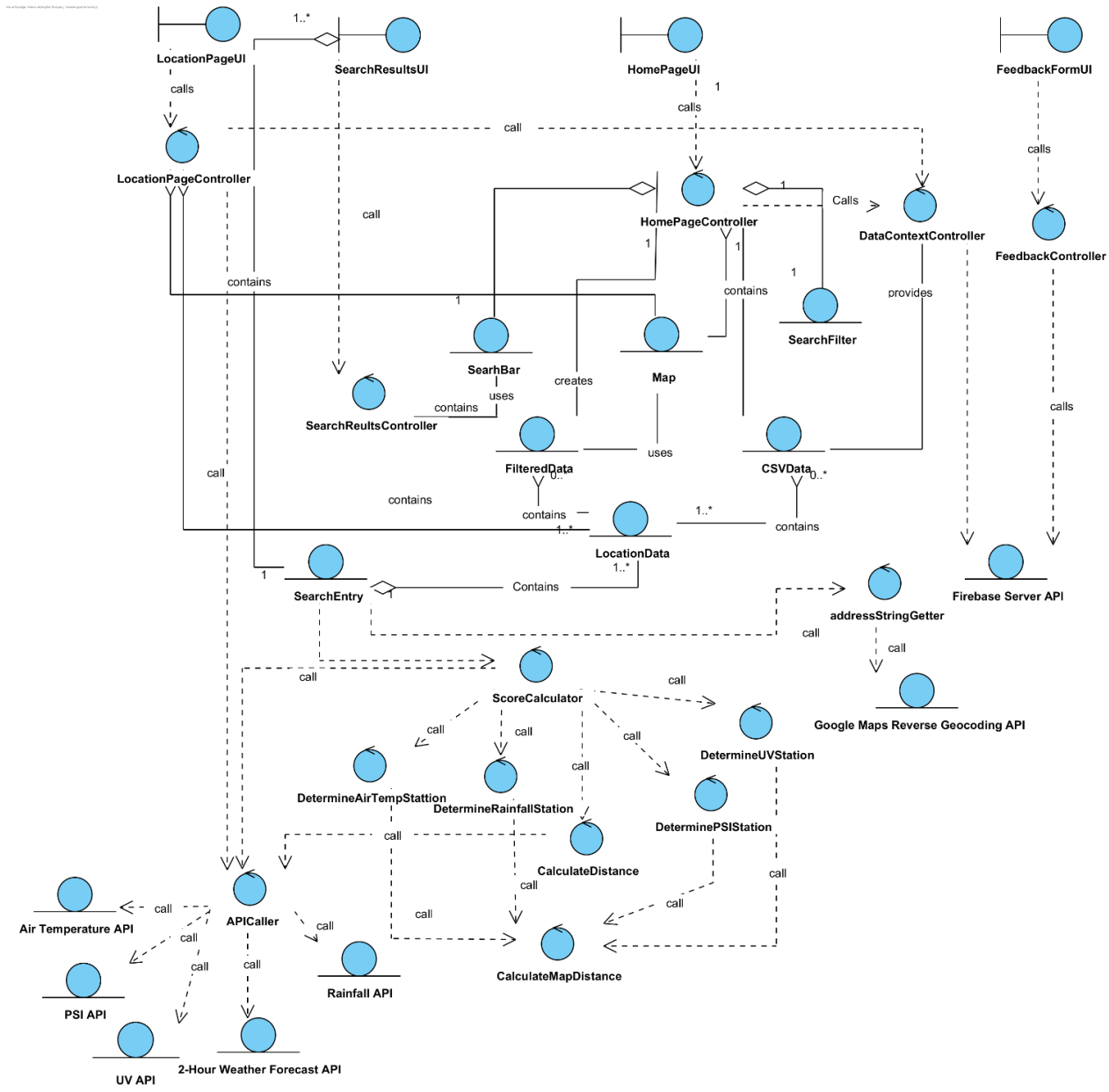


10.6 Whole Thing



Full pdf can be found here: Sportify Full Class Diagram.pdf

11. Class Entity Diagram



12. Key Public Methods

Class	Descriptions of public methods
HomePageController	+filterData(csvData: CSVData) :LocationData[] - Filters the complete list of sports locations in Singapore by whether they are within the user-specified search radius from the user-selected departure location and adds these locations to an array
SearchBar	+setShowFilter(showFilter: bool) - Setter method for showFilter variable, which determines whether the search filter will be displayed +setAddress(address: string) - Setter method for the starting address of departure location specified by the user +setCenter(center: Coordinate) - Setter method for the center coordinate for the current location marker to appear on the map +filterToggle() - Toggles each of user-selected modes of transport +searchAction() - Anonymous javascript callback function which gets executed when the search button is pressed
SearchFilter	+setShowFilter(showFilter: bool) - Setter method for showFilter variable, which determines whether the search filter will be displayed +setSliderValue(sliderValue: int) - Setter method for sliderValue, which is the search radius specified by the user +setPTValue(ptValue: bool) - Setter method for whether the public transport is selected as a mode of transport by the user +setWalkValue(walkValue: bool) - Setter method for whether the walking is selected as a mode of transport by the user

	+setCarValue(carValue:bool) - Setter method for whether the car is selected as a mode of transport by the user +setMBValue(MBvalue:bool) - Setter method for whether the motorbike is selected as a mode of transport by the user
Map	+setInfoWindow(contents: string) - Setter method for setting the contents inside the information window +setZoom(zoomValue: int) - Setter method for setting the zoom value of the map to keep the size of the circular overlay the same, depends on the searchRadius specified by the user +setCenter(center: Coordinate) - Setter method for the center coordinate for the current location marker to appear on the map +setCircleRadius(circleRadius: float) - Setter method for the search radius specified by the user +setFilteredData(filteredData: LocationData[]) - Setter method for the array of sports location which are inside the search radius +setSliderValue(sliderValue: int) - Setter method for sliderValue, which is the search radius specified by the user
DataContextController	+readCSV(): LocationData[] - Reads the full list of sports locations and their information and returns an array of LocationData objects +incrementField(locationID: int, fieldName: string) - Increments the value of a field of any sports location based on their ID +decrementField(locationID: int, fieldName: string) - Decrements the value of a field of any sports location based on their ID

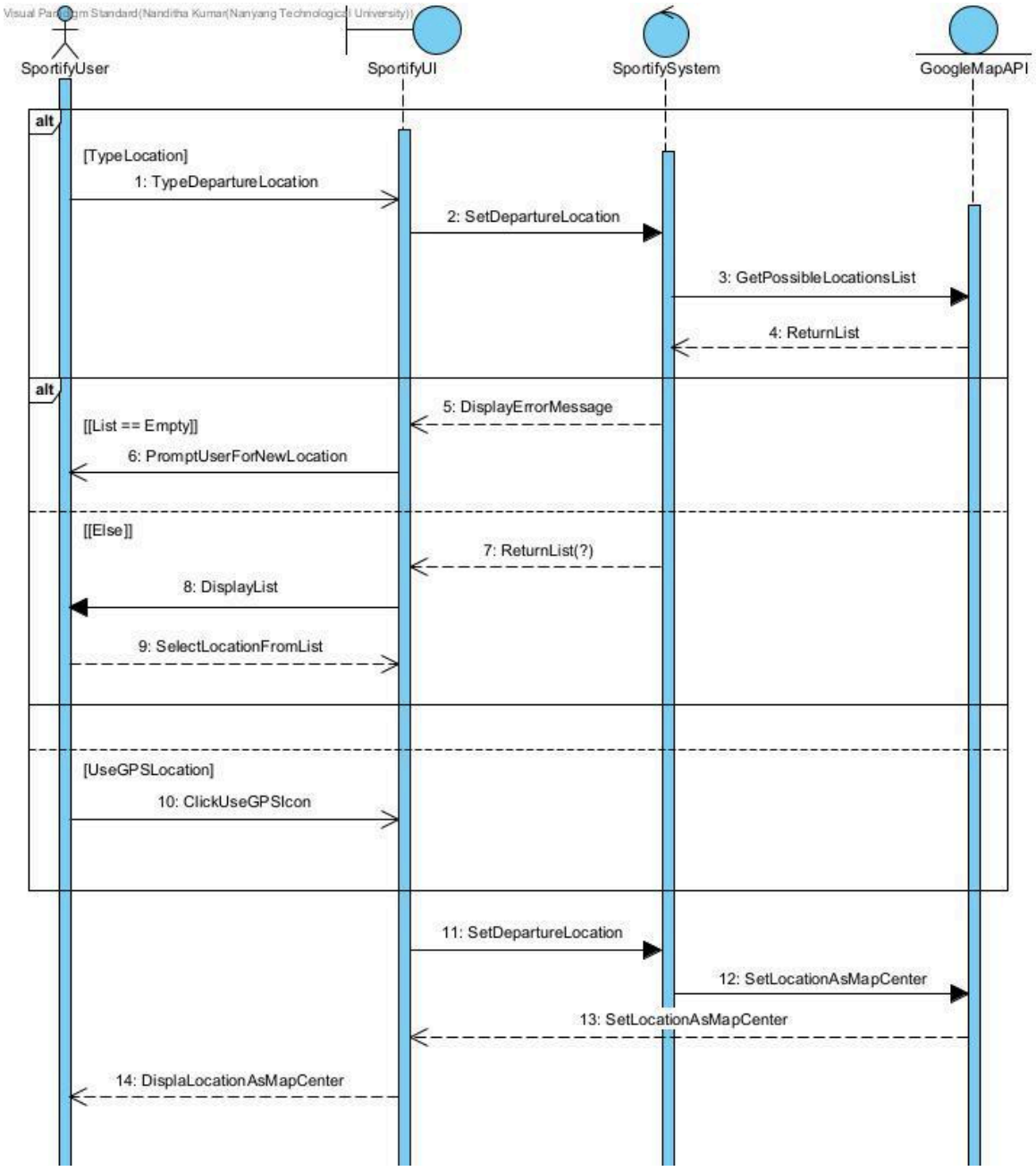
FeedbackFormUI	+submitForm(parameter: string = null): string - parses text in user text box into string
FeedbackController	+getFeedback(form: input) : string - gets the user feedback from the text box in the form +submitFeedback(databaseAPILink: string) - sends user feedback to database to be stored
SearchResultsUI	+setResultList(): SearchEntry[] - Sets the array of SearchEntry objects to displayed as search results
SearchResultsController	+getTravelModes(): string[] - Getter method for the types of travel modes selected by the user +getOrigin(): Coordinate - Getter method for the center coordinate of the departure location specified by the user +getFilteredData(): LocationData[] - Getter method for the list of sports locations in the search radius and their information
LocationPageUI	+setLocationInfo(): LocationData - Setter method for setting LocationData to be displayed in the UI
LocationPageController	+setWeatherImage(weatherForecast: string) - Setter method for setting the image that represents 2-hour weather forecast +handleReturn() - Callback function which executes upon the click of the return UI button +modifyUserState(userState: enum) - Changes the user state from a list of predefined possible states, namely a) Default b) Pre-checked in c) Checked-in d) Playing
ScoreCalculator	+calculateOverallScores(): int[] - calculates overall scores of each sports location in

	filtered list making use of both distance scores and weather scores from respective internal methods. Returns array of integers which are the scores for each sports location. From 0-100.
DetermineAirTempStation	+calAirTempStation(location: Object, airData: float): int - calculates the map distances between departure location and each air temperature station and returns the ID of the air temperature station which is closest
DetermineRainfallStation	+calRainfallStation(location: Object, rainfallData: float): int - calculates the map distances between departure location and each rainfall station and returns the ID of the rainfall station which is closest
DeterminePSIStation	+calPSIStation(location: Object, rainfallData: float): int - calculates the map distances between departure location and each PSI station and returns the ID of the PSI station which is closest
DetermineUVIStation	+calUVIStation(location: Object, rainfallData: float): int - calculates the map distances between departure location and each UVI station and returns the ID of the UVI station which is closest
DistanceCalculator	+calDistance(locationA: coordinate,locationB: coordinate): float - calculates the map distance between 2 points on the map and returns the distance is kilometers
AddressGetter	+getAddressString(latitude: float, longitude: float): string - calls Google's reverse geo-coding API to obtain the address string from coordinates of a location
APICaller	+fetchRanfallReadings(): Object - calls rainfall API from https://beta.data.gov.sg/collections/1459/datasets/d_72ef872af64a49bbf85c037e714267c0/view to obtain rainfall readings in mm from all rainfall monitoring stations in singapore and returns the location of each rainfall monitoring station together with its reading in an Object +fetchWeatherReadiings(): Object - calls 2-hour weather forecast API from https://beta.data.gov.sg/collections/1456/datasets/d_91ffc58263cff535910c16a4166ccbc3/view to obtain forecast as a string from all forecast stations in

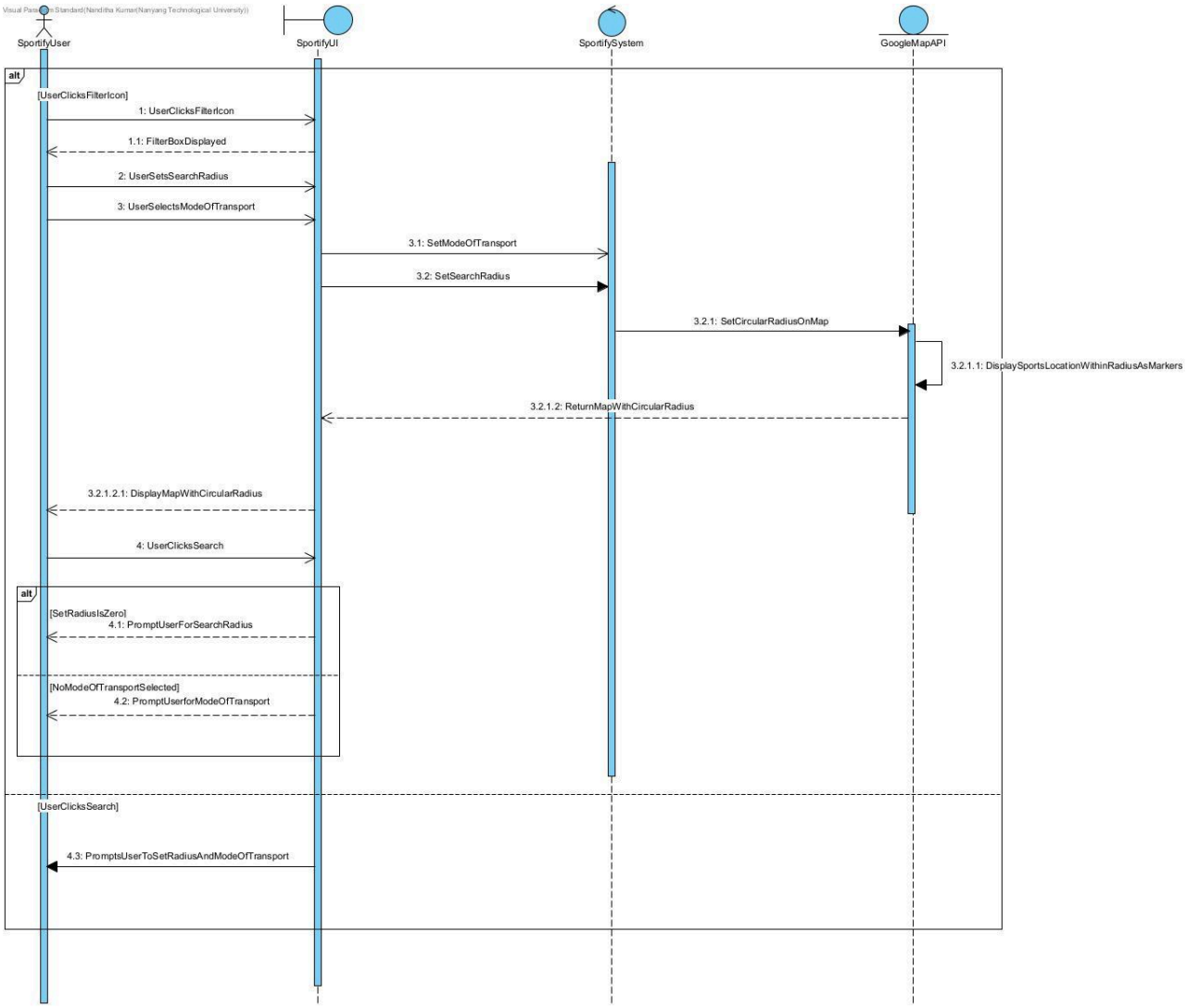
	singapore and returns the location of each forecast station together with its forecast in an Object +fetchAirTempReadings(): Object - calls air temperature API from https://beta.data.gov.sg/collections/1459/datasets/d_5b1a6d3688427dd41e2c234fe42fb863/view to obtain air temperature readings in degrees celsius from all air temperature monitoring stations in singapore and returns the location of each air temperature monitoring station together with its reading in an Object +fetchPSIReadings(): Object - calls PSI API from https://beta.data.gov.sg/collections/1396/datasets/d_8a7850dc3993dc45f1620b9972c58d4d/view to obtain PSI readings from all PSI stations in singapore and returns the location of each PSI station together with its reading in an Object +fetchUVIReadings(): Object - calls UVI API from https://beta.data.gov.sg/collections/1420/datasets/d_076774d6843cc3369731f5abaef83d30/view to obtain UVI readings from all UVI stations in singapore and returns the location of each UVI station together with its reading in an Object

13. Sequence Diagram

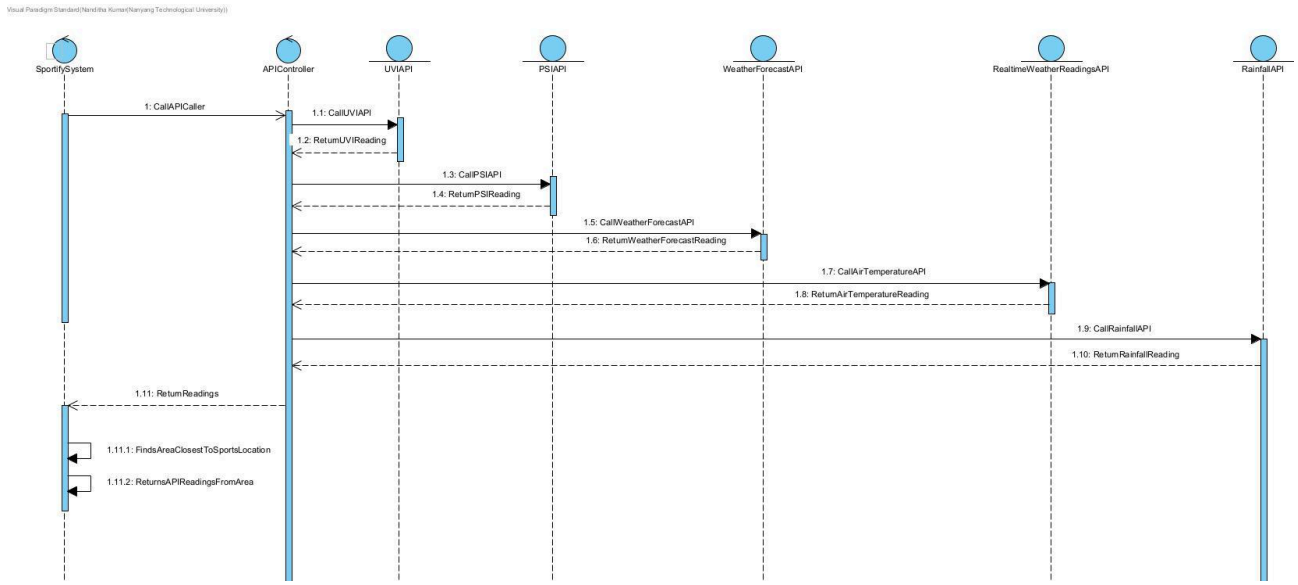
13.1 EnterUserLocation



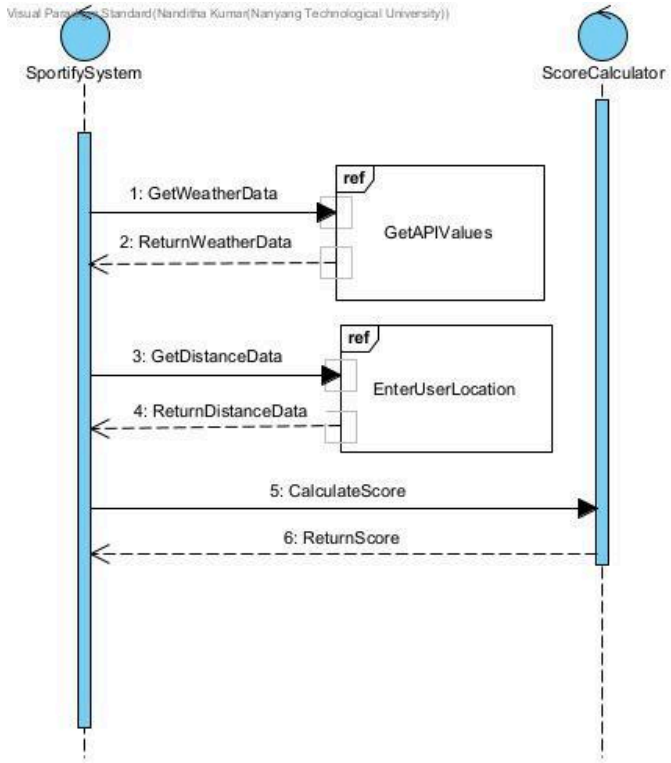
13.2 SpecifySearchCriteria



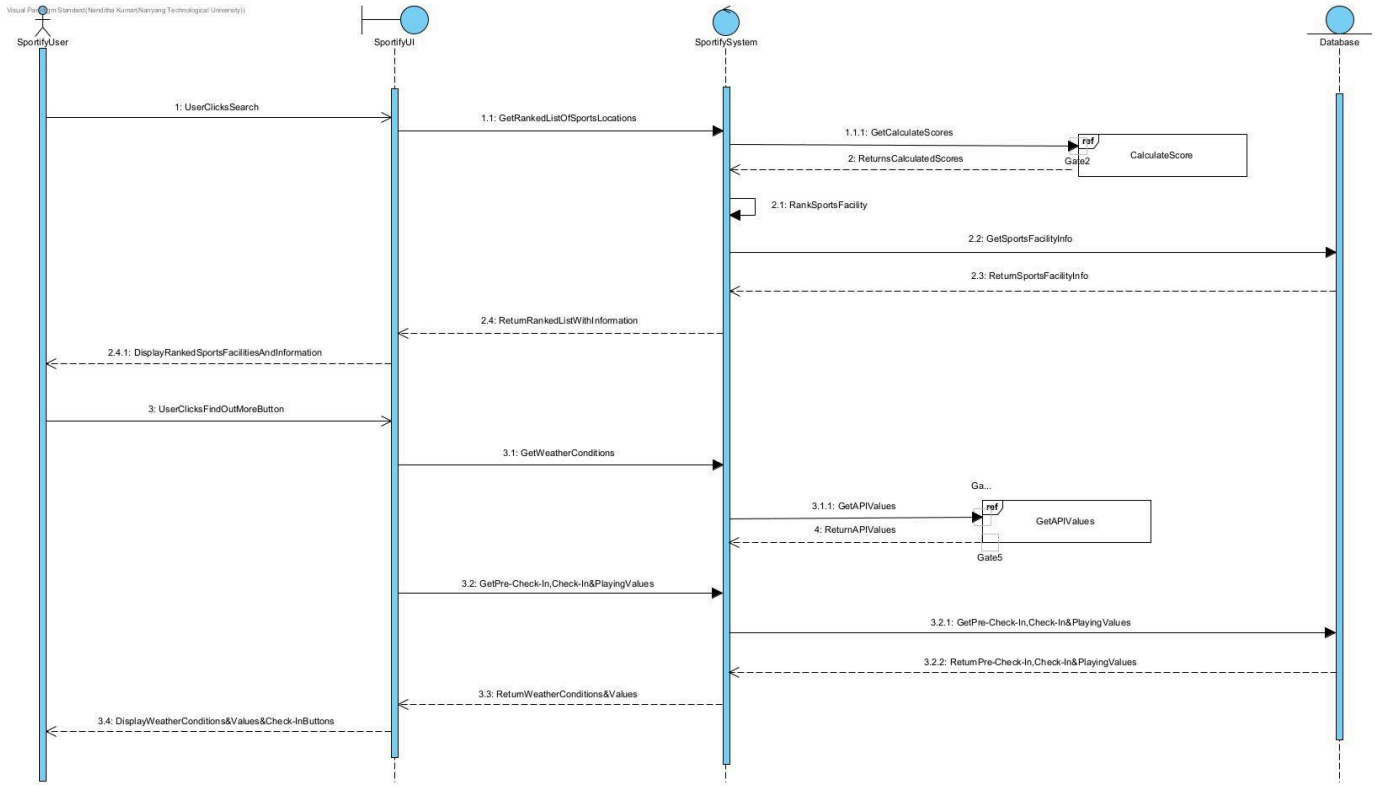
13.3 GetAPIValues



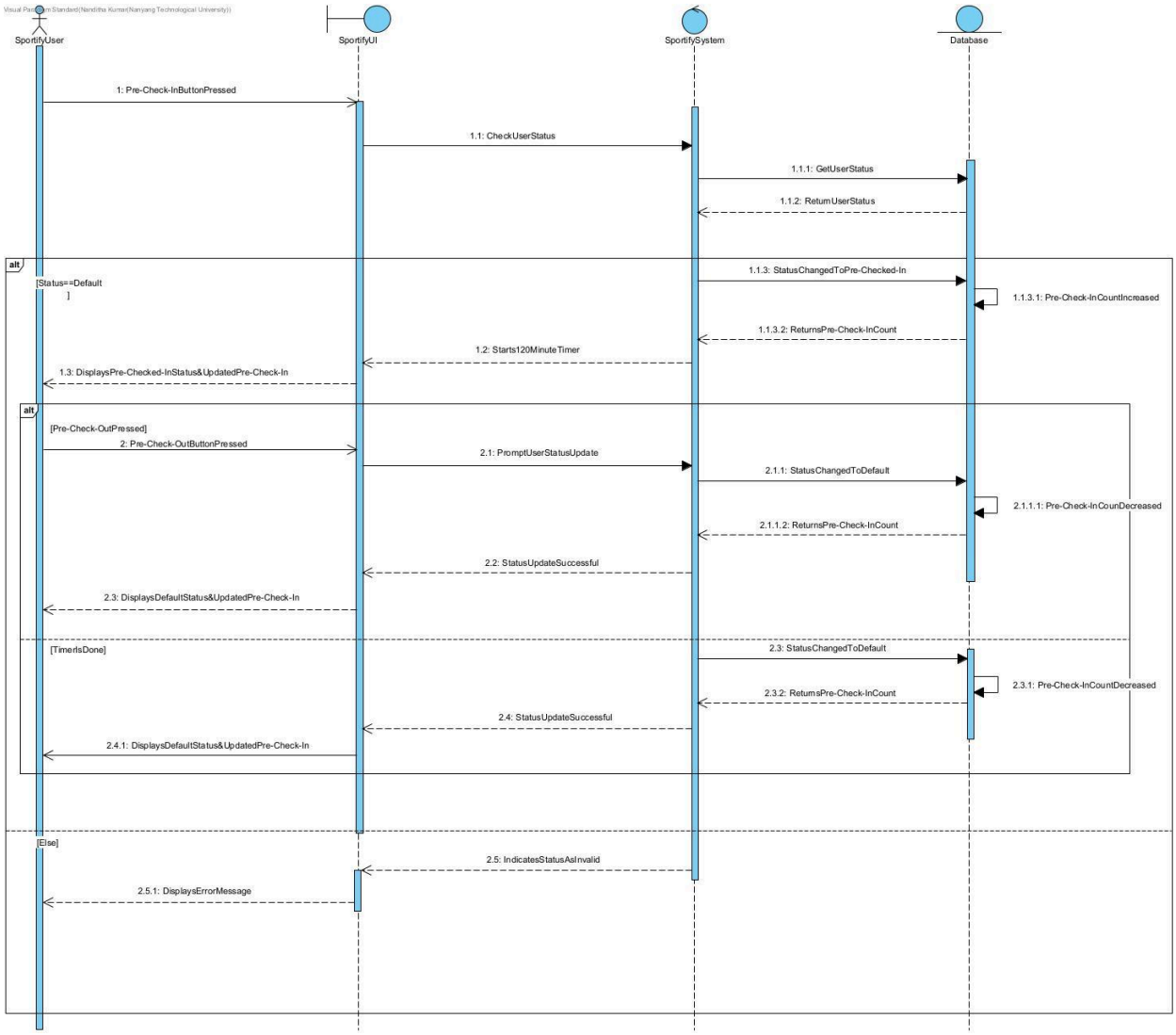
13.4 CalculateScore



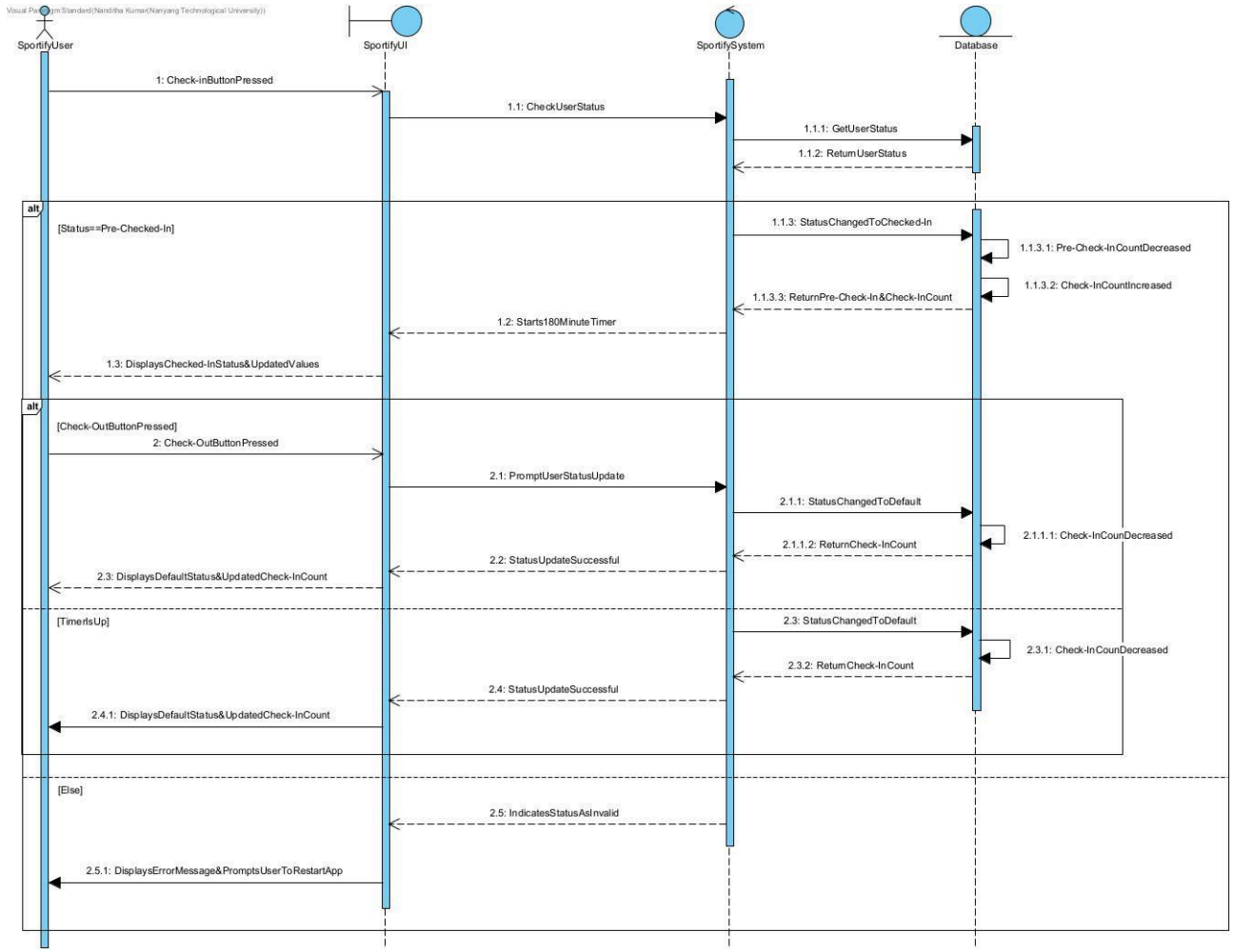
13.5 DisplayRecommendations



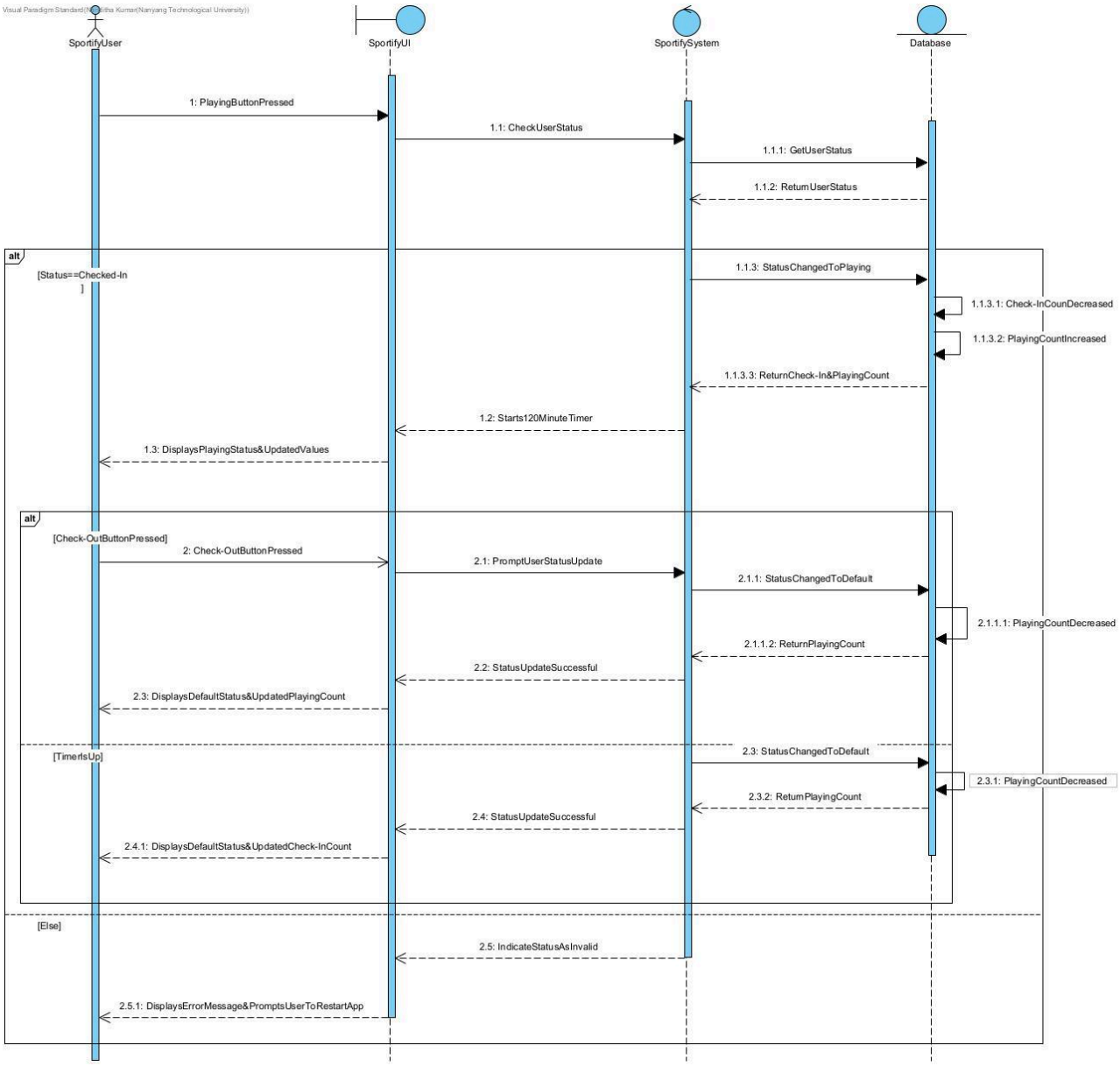
13.6 Pre-Check-In



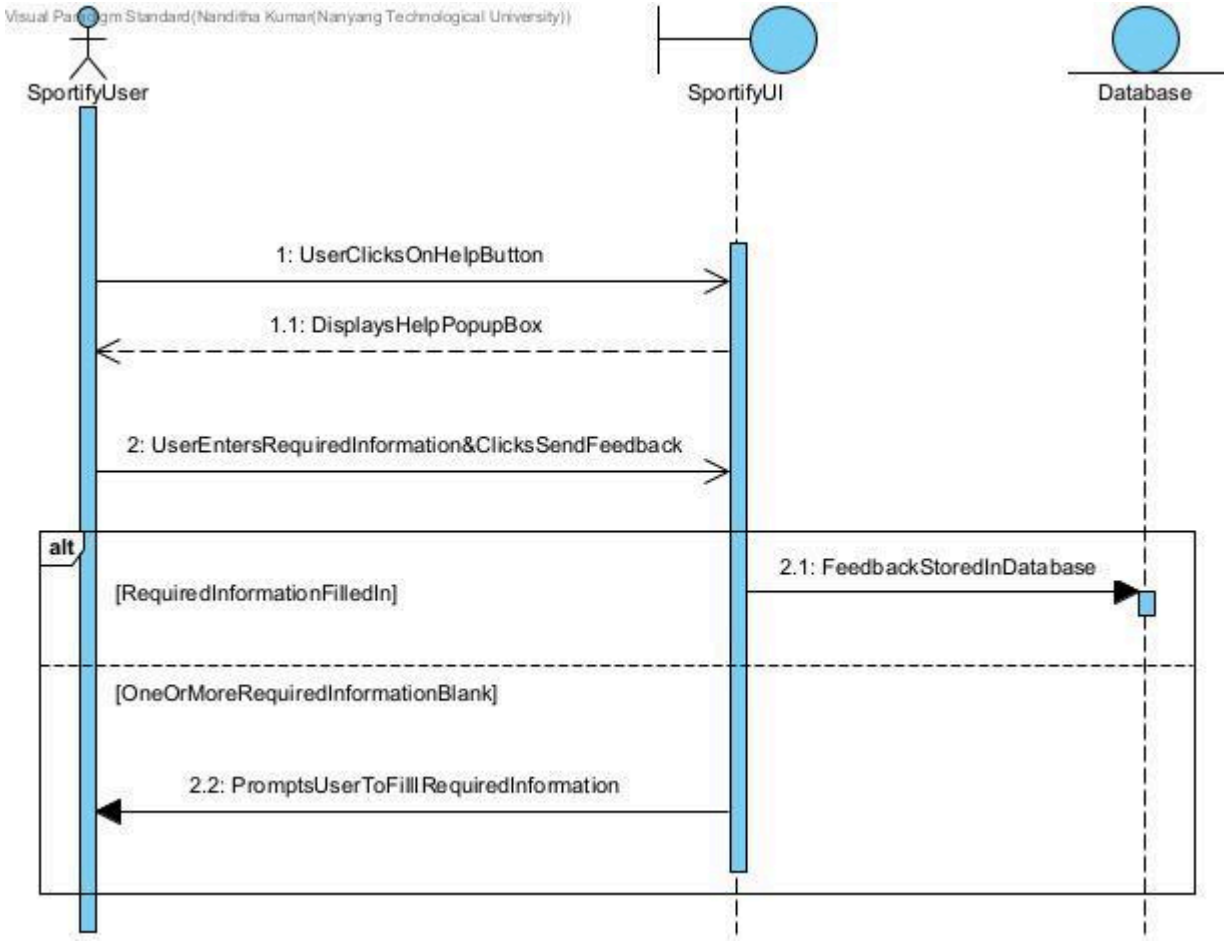
13.7 Check-In


SportifyUser
SportifyUI
SportifySystem
Database
1: Check-inButtonPressed
1.1: CheckUserStatus
1.1.1: GetUserStatus
1.1.2: ReturnUserStatus
alt
[Status==Pre-Checked-In]
1.1.3: StatusChangedToChecked-In
1.1.3.1: Pre-Check-InCountDecreased
1.1.3.2: Check-InCountIncreased
1.1.3.3: ReturnPre-Check-In&Check-InCount
1.2: Starts180MinuteTimer
1.3: DisplaysChecked-InStatus&UpdatedValues
alt
[Check-OutButtonPressed]
2: Check-OutButtonPressed
2.1: PromptUserStatusUpdate
2.1.1: StatusChangedToDefault
2.1.1.1: Check-InCounDecreased
2.1.1.2: ReturnCheck-InCount
2.2: StatusUpdateSuccessful
2.3: DisplaysDefaultStatus&UpdatedCheck-InCount
[TimerIsUp]
2.3: StatusChangedToDefault
2.3.1: Check-InCounDecreased
2.3.2: ReturnCheck-InCount
2.4: StatusUpdateSuccessful
2.4.1: DisplaysDefaultStatus&UpdatedCheck-InCount
[Else]
2.5: IndicatesStatusAsInvalid
2.5.1: DisplaysErrorMessage&PromptsUserToRestartApp

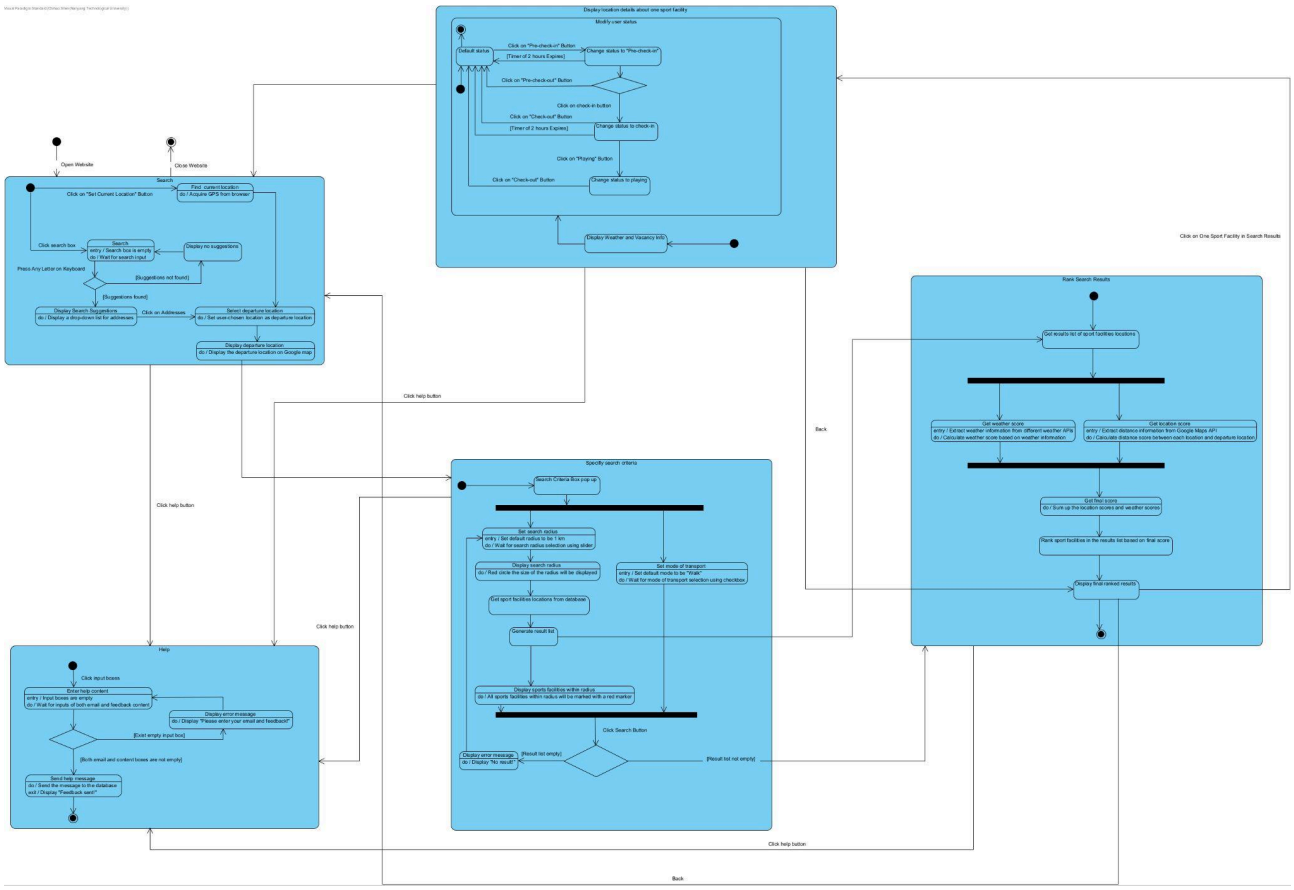
13.8 Playing


SportifyUser
SportifyUI
SportifySystem
Database
1: PlayingButtonPressed
1.1: CheckUserStatus
1.1.1: GetUserStatus
1.1.2: ReturnUserStatus
alt
[Status==Checked-In]
1.1.3: StatusChangedToPlaying
1.1.3.1: Check-InCounDecreased
1.1.3.2: PlayingCountIncreased
1.1.3.3: ReturnCheck-In&PlayingCount
1.2: Starts120MinuteTimer
1.3: DisplaysPlayingStatus&UpdatedValues
alt
[Check-OutButtonPressed]
2: Check-OutButtonPressed
2.1: PromptUserStatusUpdate
2.1.1: StatusChangedToDefault
2.1.1.1: PlayingCountDecreased
2.1.1.2: ReturnPlayingCount
2.2: StatusUpdateSuccessful
2.3: DisplaysDefaultStatus&UpdatedPlayingCount
[TimerIsUp]
2.3: StatusChangedToDefault
2.3.1: PlayingCountDecreased
2.3.2: ReturnPlayingCount
2.4: StatusUpdateSuccessful
2.4.1: DisplaysDefaultStatus&UpdatedCheck-InCount
[Else]
2.5: IndicateStatusAsInvalid
2.5.1: DisplaysErrorMessage&PromptsUserToRestartApp

13.9 Feedback

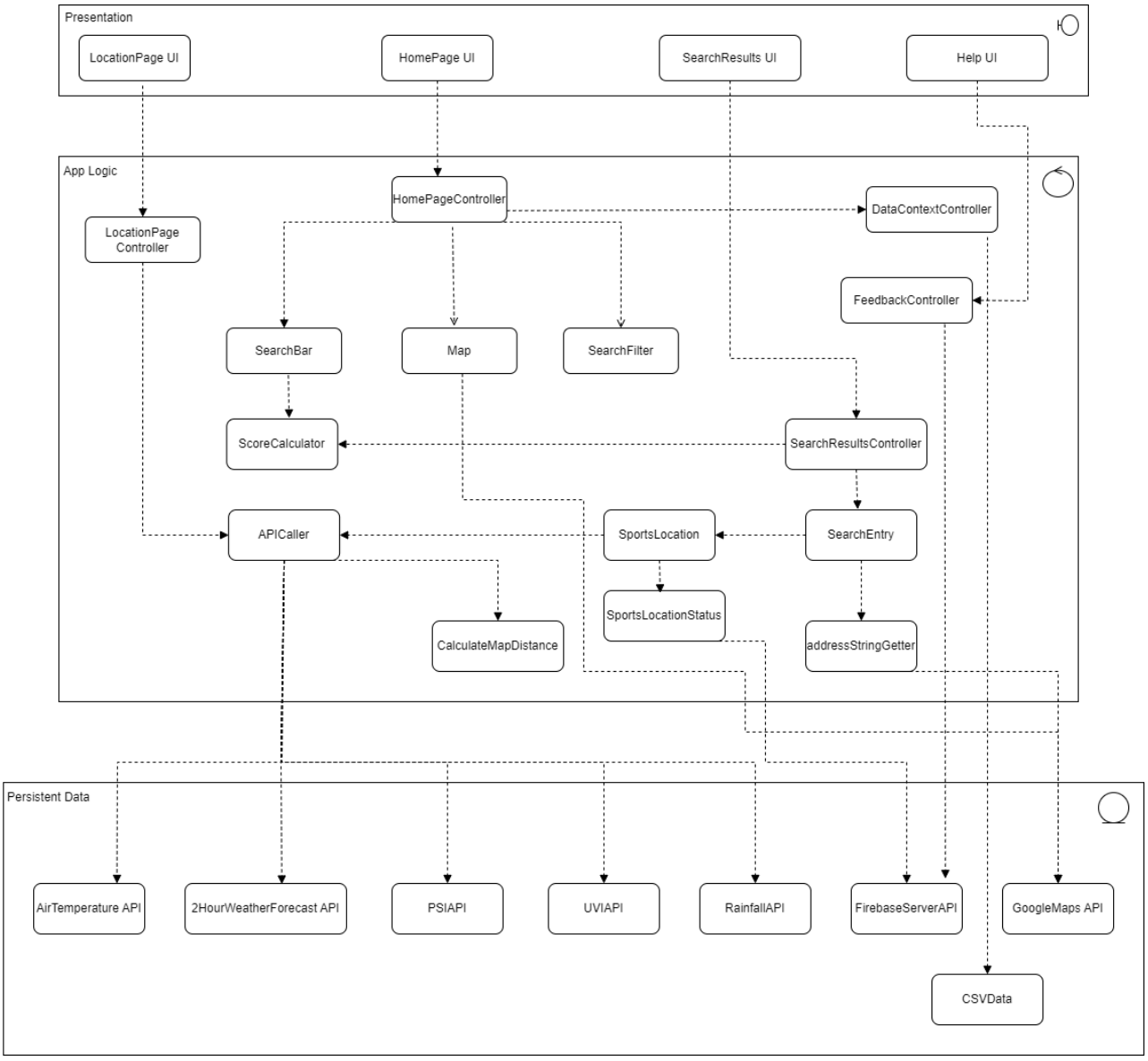


14. Dialog Map



The complete dialog map can be viewed here: Sportify dialog map v2.jpg

15. System Architecture



16. Testing

16.1 Black Box Testing

16.1.1 Functionality: Inputting Search Criteria

Input	Oracle	Output	Pass/Fail
No input for address. Search button is pressed.	Search shall not be allowed.	Search blocked. User prompted to enter address: “Please enter the location first!”	Pass
Address input lies outside of Singapore. Input: 20 W 34th St., New York, NY 10001, USA	No location suggestions shall appear.	No location suggestions appeared.	Pass
Filter not specified. Address input: Punggol Field, Punggol Plaza, Singapore Search button is pressed.	Search shall not be allowed.	Search blocked. User prompted to set filters: “Please set the filter first!”	Pass
Preferred modes of transport not specified. Address input: Punggol Field, Punggol Plaza, Singapore Search button is pressed. Radius: 2 km	Search shall not be allowed.	Search blocked. User prompted to set modes of transport: “Please choose at least one mode of transport!”	Pass
No locations found in search radius. Address input: Punggol Field, Punggol Plaza, Singapore Radius: 0.5 km	No search results.	Search blocked. User prompted: “No locations!”	Pass

Current location is outside of Singapore. Current location button is used. Location used via VPN: Japan	Must not be valid address input.	Location in Japan used as valid address input and marked on map. Able to set filters but no result when searched as no valid locations found in viable search radius.	Fail

16.1.2 Functionality: Search Results

Input	Oracle	Output	Pass/Fail
Input address itself is a sports location and is the only location in radius. Address input: Toh Yi Drive, Badminton courts @ Toh Yi Drive, Singapore Radius: 0.5 km	Must include input address as result.	Badminton courts @ Toh Yi Drive marked in map with marker. Only Badminton courts @ Toh Yi Drive shows up in result page.	Pass
No valid route for preferred modes of transport. Address input: Pulau Ubin Radius: 8 km	Results with no possible routes must not show up.	Still shows results in search radius that are not reachable with modes of transport. Route display does not show any possible routes. Toggles for modes of transport still present.	Fail

16.1.3 Functionality: Check-In System

Input	Oracle	Output	Pass/Fail
Status shall remain when returning and looking at other locations.	Status remains when returning back to location.	Pre-checked in status still remains when returning to look at other result locations. Pre-checked in status remains when returning to home page and searching with other address inputs and search criteria before returning.	Pass
Status shall remain when refreshing page.	Status remains after refreshing page.	Status remains after refreshing page.	Pass
Status shall remain after closing page and returning later.	Status remains after reopening page after closing.	Status remains after reopening page after closing.	Pass
User shall not be checked-in to multiple locations at once.	User prevented from checking-in to other locations while still checked-in in another location.	User able to be checked-in to multiple locations simultaneously.	Fail
User shall not be able to check-in in other locations while “Playing” in another location.	User prevented from checking-in in other locations while still “Playing” in another location.	User able to check-in to other locations while “Playing” in another location.	Fail
User shall not be “Playing” in multiple locations at once	User prevented from “Playing” in other locations while still “Playing” in another location.	User able to be “Playing” while “Playing” in another location.	Fail

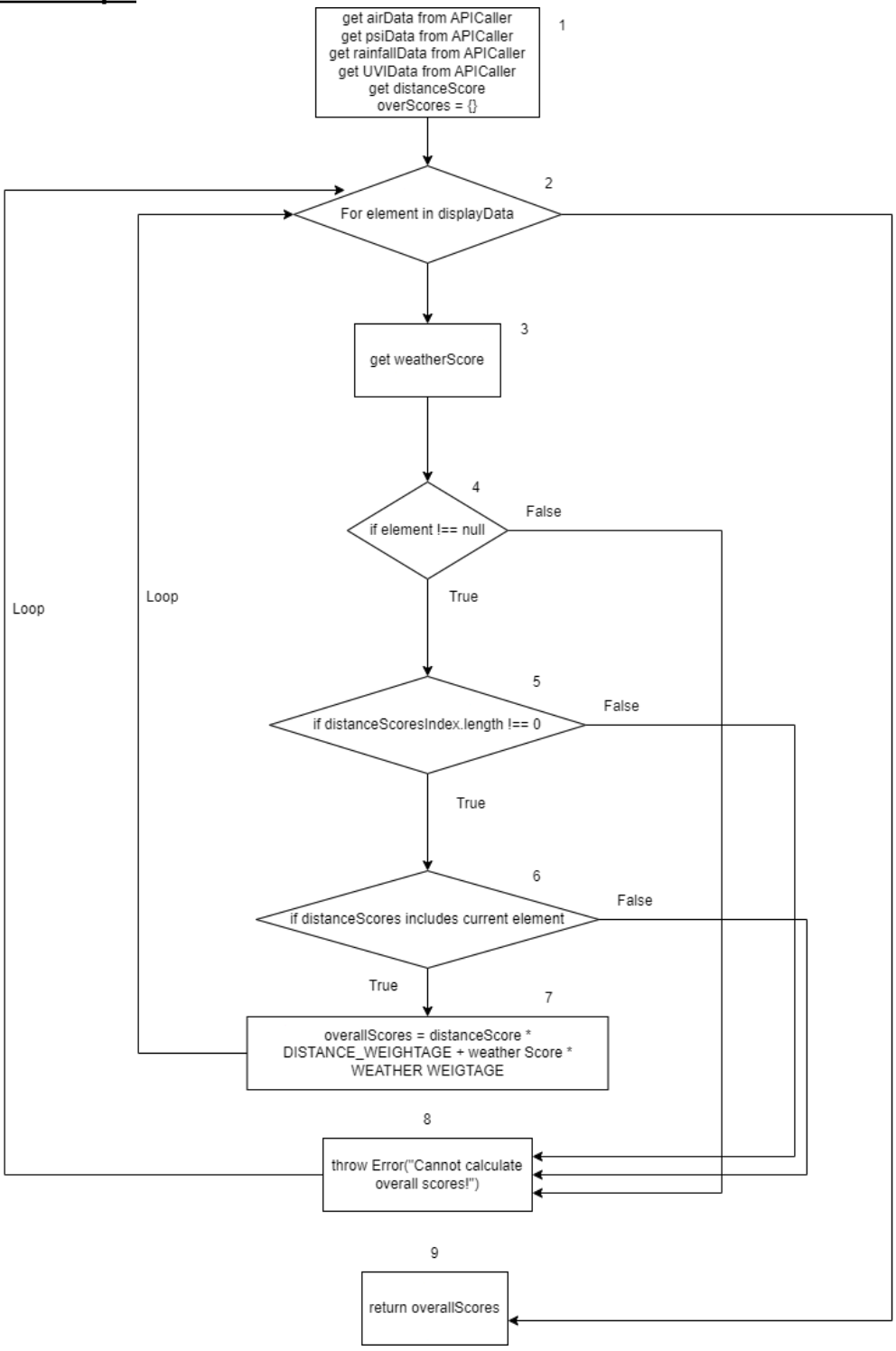
16.1.3 Functionality: “Help” Button Feedback

Input	Oracle	Output	Pass/Fail
No input for email or feedback.	Feedback shall not be sent. User shall be prompted to fill up relevant information.	Feedback not sent. User prompted: “Please enter your email and feedback!”	Pass
No email input. Feedback is filled. Feedback: “Great App!”	Feedback shall not be sent. User shall be prompted to fill up relevant information.	Feedback not sent. User prompted: “Please enter your email and feedback!”	Pass
No feedback input. Email is filled. Email: randall@gmail.com	Feedback shall not be sent. User shall be prompted to fill up relevant information.	Feedback not sent. User prompted: “Please enter your email and feedback!”	Pass
Invalid email input. Feedback is filled. Email: notarealemail@notarealemail.com Feedback: “Great App!”	Feedback shall not be sent. User shall be prompted to fill up relevant information properly.	Feedback sent successfully. User prompted: “Feedback sent!” Feedback found in Firebase with invalid email.	Fail

16.2 White Box Testing

Functionality: calculateScore

Control Flow Graph



Cyclometric complexity

$CC\ calculation = |edges| - |nodes| + 2$
= 12 - 9 + 2
= 5

Basis paths:

I. 1, 2, 3, 4, 5, 6, 7, 2, 9 (baseline)
II. 1, 2, 9
III. 1, 2, 3, 4, 8, 2, 9
IV. 1, 2, 3, 4, 5, 8, 2, 9
V. 1, 2, 3, 4, 5, 6, 8, 2, 9

Path	Path Flow	Description of scenario and Basis Path choice
I	1→ 2→ 3→ 4→ 5→ 6→ 7→ 2 (loop until all elements in displayData have been processed) → 9 **(baseline)**	This is the typical execution path that the user will experience when calculateScores is called. Hence this will be the baseline path. Upon deciding the departure location from an address, as well as a search radius as a maximum distance the user would like to travel, sports locations within that selected radius will be added to displayData and appear as map markers as an overlay inside of the polyline radius. calculateScores will then call separate functions to calculate weatherScore and distanceScore for each location in the displayData, before combining them based on weightages to generate a final score. overallScores is an array of final scores for the sports locations inside displayData.
II	1→ 2→ 9	This basis path represents a scenario for a false condition at the first decision point which is the for loop. This is when displayData is empty, i.e. there are no elements inside displayData which means it will not enter the for loop and will return early.
III	1→ 2→ 3→ 4 → 8→ 2→ 9	This basis path represents a scenario for a false condition at the second decision point, for element=== null. This is when sports locations generated and added into displayData are not valid objects and are null. It will then throw an error message and continue the loop with the next sports location.

IV	1→ 2→ 3→ 4→ 5→ 8→ 2→9	This basis path represents a scenario for a false condition at the third decision point, for distanceScoresIndex.length=== 0. This is when sports locations generated and added into displayData and passed into calculateDistanceScores encounters an error resulting in distance scores not being generated and resulting in the distance scores array to be empty. It will then throw an error message and continue the loop with the next sports location.
V	1→ 2→ 3→ 4→ 5→ 6→8→ 2→ 9	This basis path represents a scenario for a false condition at the fourth decision point, for when distanceScores array do not contain the ID of the current sports location. This is when sports locations generated and added into displayData and passed into calculateDistanceScores encounters an error resulting in distance scores being calculated for other sports location which are different and mismatch those in displayData. It will then throw an error message and continue the loop with the next sports location.

Test Cases

Path	Test Case	Values of Test Case	Output
I	Baseline path	displayData = [{Facility : "Park" Images :"https://lh5.googleusercontent.com/p/AF1QipMQ7vAbm1IccWAw38paO05ofLIQ5HefKZl2nuPg=w122-h92-k-no" Name : "Hoover Park" Ratings : "4.3" Sports : "running (outdoor)" X : "103.7730586" Y : "1.3438729" distanceFromCenter : 1.9991648496827612	[93, 88, 86]

		index : "111" },{Facility : "Gym and Fitness Centre" Images : "https://lh5.googleusercontent.com/p/AF1QipNgpEyvSbsBrrWBPVbWJyZ29aXo35ijDgnC3MFs=w122-h92-k-no" Name : "Badminton courts @ Toh Yi Drive" Ratings : "3.5" Sports : "badminton (indoor)" X : "103.773527" Y : "1.3386533" distanceFromCenter : 1.591955602754787 index : "157" },....] --- element = {Facility : "Park" Images :"https://lh5.googleusercontent.com/p/AF1QipMQ7vAbm1IccWAw38paO05ofLIQ5HefKZl2nuPg=w122-h92-k-no" Name : "Hoover Park" Ratings : "4.3" Sports : "running (outdoor)"	

		X : "103.7730586" Y : "1.3438729" distanceFromCenter : 1.9991648496827612 index : "111" } --- distanceScoresIndex.length = 3 distanceScores includes element = True	
II	There are no elements inside displayData	displayData = [] element = null distanceScoresIndex.length = 0 distanceScores includes element = False	(3) “Cannot calculate overall scores!”
III	Sports locations generated and added into displayData are null.	displayData = [{Facility : "Park" Images :"https://lh5.googleusercontent.com/p/AF1QipMQ7vAbm1IccWAw38paO05ofLIQ5HefKZl2nuPg=w122-h92-k-no" Name : "Hoover Park" Ratings : "4.3" Sports : "running (outdoor)" X : "103.7730586" Y : "1.3438729" distanceFromCenter : 1.9991648496827612 index : "111" },{Facility : "Gym and Fitness Centre" Images	(3) “Cannot calculate overall scores!”

		: "https://lh5.googleusercontent.com/p/AF1QipNgpEyvSbsBrrWBPVbWJyZ29aXo35ijDgnC3MFs=w122-h92-k-no" Name : "Badminton courts @ Toh Yi Drive" Ratings : "3.5" Sports : "badminton (indoor)" X : "103.773527" Y : "1.3386533" distanceFromCenter : 1.591955602754787 index : "157" },....] --- element = null distanceScoresIndex.length = 3 distanceScores includes element = false	
IV	Sports locations passed into calculateDistanceScores encounters an error resulting in distance scores not being generated.	displayData = [{Facility : "Park" Images :"https://lh5.googleusercontent.com/p/AF1QipMQ7vAbm1IccWAw38paO05ofLIQ5HefKZl2nuPg=w122-h92-k-no" Name : "Hoover Park" Ratings : "4.3" Sports : "running (outdoor)" X : "103.7730586"	(3) “Cannot calculate overall scores!”

		Y : "1.3438729" distanceFromCenter : 1.9991648496827612 index : "111" },{Facility : "Gym and Fitness Centre" Images : "https://lh5.googleusercontent.com/p/AF1QipNgpEyvSbsBrrWBPVbWJyZ29aXo35ijDgnC3MFs=w122-h92-k-no" Name : "Badminton courts @ Toh Yi Drive" Ratings : "3.5" Sports : "badminton (indoor)" X : "103.773527" Y : "1.3386533" distanceFromCenter : 1.591955602754787 index : "157" },....] --- element = {Facility : "Park" Images :"https://lh5.googleusercontent.com/p/AF1QipMQ7vAbm1IccWAw38paO05ofLIQ5HefKZl2nuPg=w122-h92-k-no" Name : "Hoover Park"	

		Ratings : "4.3" Sports : "running (outdoor)" X : "103.7730586" Y : "1.3438729" distanceFromCenter : 1.9991648496827612 index : "111" } distanceScoresIndex.length = 0 distanceScores includes element = false	
V	Mismatching sports location IDs between displayData and distanceScores	displayData = [{Facility : "Park" Images :"https://lh5.googleusercontent.com/p/AF1QipMQ7vAbm1IccWAw38paO05ofLIQ5HefKZl2nuPg=w122-h92-k-no" Name : "Hoover Park" Ratings : "4.3" Sports : "running (outdoor)" X : "103.7730586" Y : "1.3438729" distanceFromCenter : 1.9991648496827612 index : "111" },{Facility : "Gym and Fitness Centre" Images : "https://lh5.googleusercontent.com/p/AF1QipNgpEyvSbsBrrWBPVbWJyZ29aXo35ijDgnC3MFs=w122-h92-k-no"	(3) “Cannot calculate overall scores!”

		Name : "Badminton courts @ Toh Yi Drive" Ratings : "3.5" Sports : "badminton (indoor)" X : "103.773527" Y : "1.3386533" distanceFromCenter : 1.591955602754787 index : "157" },....] --- element = {Facility : "Park" Images :"https://lh5.googleusercontent.com/p/AF1QipMQ7vAbm1IccWAw38paO05ofLIQ5HefKZl2nuPg=w122-h92-k-no" Name : "Hoover Park" Ratings : "4.3" Sports : "running (outdoor)" X : "103.7730586" Y : "1.3438729" distanceFromCenter : 1.9991648496827612 index : "111" }	

		distanceScoresIndex.length = 3 distanceScores includes element = True	

17. Demo Script

Steps:

Stage 1:

1) Executive summary
2) Use current location button
3) Press search button
4) Show no filter error, then set filter
5) Slide to show marker appearances
6) Select Walk and Car as mode of transport
7) Press search button
8) Describe result card address, activities, distance, type meaning
9) Click on find out more
10) Explain details of find out more (Leave out check in check out), explain apple watch circle (How to read circles and legends)
11) Click return and go to another location find out more (Of lower score) and explain how scoring differs from previous location
12) Click on "Home" button
13) Click on "Contact Us" to show popup
14) Click on "Help" to show feedback functionality
15) Submit demo feedback

Stage 2:

1) Show that filter doesn't work without address
2) Type in address E.g. Clementi Loop
3) Explain drop down bar and select location E.g. Toh Yi badminton
4) Adjust radius to 4km and search, select all modes of transport
5) Show results and click on one location
6) Explain route display and deselect public transport and motorbike to show extra features
7) Select public transport and show how it displays which public transport to take
8) Explain Check-in system and idea behind this system
9) Demonstrate Check-in system
10) Go back to home

Script:
Hello, so now we will be starting the live demo of Sportify, an app that suggests relevant nearby sports locations.

We begin at the home page, which consists of a navigation bar, a map as well as a search bar where we can start by typing in a departure address or simply use the “Current Location” button to select the user’s current location as the departure address. The user will be prompted to allow the use of their device GPS and automatically selects the current location of the user as can be seen marked by the pin on the map, which in this case would be NTU. From here, the user would need to set their search radius and modes of transport. Here we can see that as we modify the search radius, the number of nearby sports locations will change accordingly. With each location having one marker on the maps interface. Each marker is also interactive, and users can quickly click on the markers to see the name and address of the location. For this search, we will be using 6km as search radius and select “walk” and “car” as our preferred modes of transport.

Here we can see the results of all the sports locations within the search radius ranked according to their score in descending order. Each result displays the name, address, and distance of the location, as well as provides what type of activities are available at the location. The location’s surrounding PSI, UVI, rainfall, air temperature and distance values are obtained through calling APIs, and a score for each location is calculated using these values. We can also see more detailed information about each location by clicking on this find out more button.. In addition to its name and relevant activities, we are able to see if the location is affected by rain, and the current weather condition of the surrounding area. The colorful rings provide a clearer visual in showcasing the weather conditions of a location, with the red ring showing air temperature, the green ring showing PSI value and the blue ring showing UVI value. It is preferable for the rings to be empty, as it signifies more favorable weather conditions. The user can also click “return” to look at other locations at the previous result page. *Scroll down select lower score location* Here we can see that this location is more poorly scored as it is located further away from the departure location as compared to the previous location.

If the user is unsatisfied with the results and wishes to enter another search, they can simply click the “Home” button on the top navigation bar to return to the original home page. They can also click the “Contact Us” button to find our contact details should they face any problems, they can contact us directly. For demonstration purposes we only placed our github usernames there.

Sportify works fully both as a website and as an application on our mobile devices, such as ipad, and iphone. For the rest of this demonstration, we will be doing so from the perspective of an iPhone 14 Max Pro.

Now, we will try to enter an address for our next search. As you can see, by keying in the relevant keywords, Sportify would suggest a drop bar of search results and the user can simply select one of them as the departure location. Here we can see that locations existing outside of Singapore would not be featured as an available departure location.If the user chooses a very remote location, there would also be no markers that would show up and Sportify would not allow users to search if there are no locations nearby. Now we will continue with entering a valid location like clementi. Should the user forget to set the search radius and modes of transport before searching, Sportify will also pop up the filter to prompt them to select the search radius and modes of transport.

Here we select 4km as our search radius and all the modes of transport. Similar to before, Sportify will display the locations within the search radius of the address selected, ranked according to their scores As Singapore is a small country, it is common to see locations share the same weather conditions, hence why we decided that distance shall play a larger role in the scoring system. . *Click find out more* Here we can see the different modes of transport mapped out for a selected location. Each line shows the recommended route to take based on a mode of transport selected. If the lines are cluttered and are hard to read, there are toggles for each of the modes of transport above to clear up the map.. In addition, the public transport line also indicates which bus to take in order to reach the destination.

Another additional feature of Sportify is our Check- in system. We implemented this system for our users who are interested in participating in team sports. Users can also see how crowded a location is based on how many people are playing currently, so if its too crowded, they might choose another location. Here, the user is able to “Pre-check in” to indicate that the user is heading towards the location. Should the user change their mind, they can “Pre-Check out” to remove their indication of heading to the location. Upon reaching, the user can select “Check-in” to indicate that they have reached the location and are ready to play.. click check in* After finding other users at the location and starting a game, the user can select the “Playing” status to show that they have already found their teammates and opponents. When they are ready to head home, the user can select “Check out” to indicate that they are leaving the location. Our system also has an in-built timer that automatically checks out the user after 2 hours in case they forgot to do so when leaving.For the purpose of this demonstration we have shortened the timer. *Automatically check out* as can be seen here.

With that, we have covered what Spotify provides. Thanks for taking the time to watch this demonstration and we hope to see you soon on Sportify!