**NANYANG TECHNOLOGICAL UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

****

**Cyke**

**CZ2006: Software Engineering**

**Lab Group: SS6**

**Group Name: 3riffic**

**Team Members:**

Amadeus Koh Ying Jie [U1922072A]

Cai Yiwei [U1920569G]

Cheah Mun Yan, Grace [U1920838B]

Guo Feiyan [U1922356D]

Quek Lin Hui [U1921918F]

Wu Jingyuan [U1920617K]

**2020/2021 Semester 2**

Table of Contents

[**1. INTRODUCTION**](#_gjdgxs) **5**

1[.1 Purpose 5](#_30j0zll)

1[.2](#_1fob9te) [Intended Audience and Reading Suggestion](#_3znysh7) [5](#_1fob9te)

1.3 [Project Scope](#_2et92p0) 6

1[.4 Assumptions and Constraints 6](#_tyjcwt)

1[.5 References](#_3dy6vkm) 6

**2.** [**FUNCTIONAL REQUIREMENT 7**](#_gjdgxs)

2[.1 Home Page](#_1t3h5sf) 7

2[.2 Registration](#_4d34og8) 7

2[.3 Login](#_2s8eyo1) 8

2[.4 Routing Planning](#_17dp8vu) 8

2[.5 View Weather Forecast](#_3rdcrjn) 9

2[.6 View Bicycle Rack Locations](#_26in1rg) 9

2[.7 View Saved Locations](#_lnxbz9) 9

2[.8 View History of Cycling Activity](#_35nkun2) 9

**3**[**. NON-FUNCTIONAL REQUIREMENTS**](#_1ksv4uv) **10**

3[.1 Performance](#_44sinio) 10

3[.2 Reliability](#_2jxsxqh) 10

3[.3 Security](#_z337ya) 10

3[.4 Supportability](#_3j2qqm3) 10

3[.5 Usability](#_1y810tw) 10

3[.6 Maintainability](#_4i7ojhp) 10

3[.7 Scalability](#_2xcytpi) 10

3[.8 Learnability](#_1ci93xb) 10

**4**[**. DATA DICTIONARY**](#_3whwml4) **11**

**5**[**. USE CASE MODEL 1**](#_2bn6wsx)**2**

5[.1 Use Case Diagram 1](#_qsh70q)2

5[.2 Use Case Description 1](#_3as4poj)3

5[.2.1 Query Route 1](#_1pxezwc)3

5[.2.2 Find Parking Spots 1](#_49x2ik5)4

5[.2.3 Navigation 1](#_2p2csry)6

5[.2.4 Check Weather 1](#_147n2zr)7

5[.2.5 User Registration 1](#_3o7alnk)8

5[.2.6 User Login 1](#_23ckvvd)9

**6**[**. UI MOCKUP 1**](#_ihv636)**9**

6[.1 Login 1](#_32hioqz)9

6[.2 Creating Account](#_1hmsyys) 20

6[.3 Home Page](#_41mghml) 21

6[.4 Routing](#_2grqrue) 21

6[.5 Navigation 2](#_vx1227)2

6[.6 System Errors 2](#_3fwokq0)3

6[.7 Parking Nearby 2](#_1v1yuxt)3

6[.8 Saved Locations 2](#_4f1mdlm)4

6[.9 History 2](#_2u6wntf)4

6[.10 Weather Forecast 2](#_19c6y18)5

**7**[**. UML MODELS**](#_3tbugp1) **26**

7[.1 Class Diagram](#_28h4qwu) 26

7[.1.1 Registration](#_nmf14n) 26

7[.1.2 Login](#_37m2jsg) 26

7[.1.3 Query Route](#_1mrcu09) 27

7[.1.4 Navigation](#_46r0co2) 27

7[.1.5 Check Weather](#_2lwamvv) 28

7[.1.6 Find Racks](#_111kx3o) 28

7[.1.7 Class Diagram of Entity Classes](#_3l18frh) 29

7[.2 Description of Boundary and Control Classes](#_206ipza) 30

7[.3 Sequence Diagrams](#_4k668n3) 31

7[.3.1 Registration](#_2zbgiuw) 31

7[.3.2 Login](#_1egqt2p) 32

7[.3.3 Query Route](#_3ygebqi) 33

7[.3.4 Navigation](#_vx1227) 33

7[.3.5 Check Weather](#_2dlolyb) 34

7[.3.6 Find Racks](#_sqyw64) 35

7[.4 Dialog Map](#_3cqmetx) 36

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| All members | 4/2/2021 | Added functional and non-functional requirements, data dictionary, initial use case model and the UI Mockups | 1.0 |
| All members | 18/2/2021 | Added project title, objective statement, class diagrams, sequence diagrams and dialog map | 1.1 |
|  |  |  |  |

# Introduction

## Purpose

Our application Cyke aims to aid cyclists in searching for an ideal cycling route to their destination, prioritising cycling paths and park connectors instead of roads. Cyke also helps navigate the cyclists while riding, providing them the directions. Overall, Cyke benefits cyclists by:

1. Providing them a cycling route and navigational information
2. Helping them search for nearby bicycle racks to safely park their bicycles
3. Allowing them to check the weather forecast to see if weather conditions are ideal for cycling

This document supplies an in-depth description of the mobile application Cyke. It explains the purpose of the development of the system and defines the scope of the system, including components such as the functional and non-functional requirements, UML diagrams, use cases, data dictionary, User Interface(UI) Prototype and System Testing of the application.

## Intended Audience and Reading Suggestions

The team intends this document for developers, investors, testers and general users of the proposed application. This document includes the scope of the project, an overall description of the project, interface requirements, functional requirements, and non-functional requirements.

**Developers** – For developers who want to view the application’s capabilities and have a better understanding of the application such that they will be able to add new features or improve the application. Developers can view the overall description, system features and other non-functional requirements for a more in-depth understanding of the application.

**Investors** – For investors who would like to have a general overview of the project such that they are able to get a better understanding to make potential investment decisions. Investors can view the overall description and system features for a more surface level understanding of the application.

**Testers** – For testers who want to use this document as a guideline for testing strategies that match the business’s needs. They would find this document essential to discover bugs in the application. This documentation would allow testing to become more structured and organized. Testers can view product perspective, design and implementation constraints, system features, safety and security requirements for base gathering of testing.

**Users** – For users who are interested to read the software requirement document which entails the main functionality of the application. Users can view product function and user interface to understand how to use the features of the application.

## Product Scope

Cyke is a mobile application that provides cyclists with useful information and navigation functions. The application encourages the use of park connectors to promote health and green transportation.

* + 1. Benefits for Cyclists
       1. Cyclists would be able to find nearby parking areas more easily.
       2. Cyclists would be able to check the weather conditions before commencing their cycling.
       3. Cyclists would be able to use the Park Connector Network (PCN) for their cycling.
       4. Cyclists will be able to use their real-time location to navigate towards their destinations.

## Assumptions and Constraints

* + 1. Users must always have enabled their Global Positioning System (GPS) when using this application.
    2. Users must verify their account details in order to access certain features.
    3. Users must always have their internet connectivity (3G/WiFi) when using the application.

## References

## Google Map API -<https://developers.google.com/maps/>

## Singapore Gov Data -<https://data.gov.sg/>

Flutter - <https://flutter.dev/>

# FUNCTIONAL REQUIREMENTS

## Home Page

* + 1. The home page must contain a main menu displaying all functions that a user can use.
    2. Functions include:
       - 1. Register/Login
         2. Route planning
         3. Weather forecast
         4. View bicycle rack locations
         5. View saved locations
         6. View history of cycling activity
       1. The system must allow the user to register for a new account.
    3. The system must allow the user to login to an existing account.
    4. The system must allow the user to search for a cycling route by providing:
       1. Starting location
       2. Ending location
    5. The system must allow the user to check the weather forecast.
    6. The system must allow the user to view the locations of all bicycle racks in Singapore.
    7. The system must allow the user to view his/her list of saved locations to select for quick access to route planning.
    8. The system must allow the user to view his/her cycling activity for the past year.

## Registration

* + 1. The user must be able to create a new account.
    2. Users must register using a valid Email Address that has not been registered before.
    3. Users must register with a password with at least 8 characters, consisting of at least 1 uppercase character and 1 special character.
    4. The system needs to send a verification number to the email address.
    5. The user must enter the verification number into the app to verify his/her new account.
    6. The system must validate all the fields.

## Login

* + 1. The user must be able to login to his/her existing account with an email and password
    2. The system needs to validate that the email and password are correct to allow successful login
    3. If invalid email or password is being entered, user will be prompted with an error message

## Routing planning

* + 1. The users’ query must include his/her starting location
       1. The starting location can be the users’ current location determined by his/her phone’s GPS
       2. The starting location can be a location selected by the user by inputting the name of the location
       3. The starting location can be a saved location from the user’s account (only when he/she is logged in)
    2. The users’ query must include his/her destination
       1. The user inputs the destination as the name of the location
       2. The destination can be a saved location from the user’s account (only when he/she is logged in)
    3. The result of the query is the cycling route from the starting location to the destination
       1. The system will utilise the OneMap API to search for the cycling route
       2. The cycling route will prioritise the usage of PCN over roads
       3. The cycling route will be displayed on a map with a path marked out in blue
    4. The system will discourage the user from cycling if the 2-hour forecast for any region that the cycling route covers is rainy
    5. The system will use the phone’s GPS to indicate the user’s location in real-time on the map view
       1. User’s location is indicated as a blue circle
    6. The system will use the phone’s gyroscope to indicate the direction that the user is facing on the map
       1. User’s direction is indicated as an arrow on the user’s current location
    7. The system will show a notification to remind the user to take a break if his/her cycling time exceeds 30 minutes

## View weather forecast

* + 1. The system will use the Weather Forecast API by data.gov.sg
    2. The user can check the weather forecast for all regions in Singapore within the next:
       - 1. 2-hour timeframe
         2. 24-hour timeframe
         3. The weather forecast is to be viewed on a map

## View Bicycle Rack Locations

* + 1. The system will use the LTA Bicycle Rack dataset from data.gov.sg
    2. Users must be able to view the locations of the 3 nearest bicycle racks from their current location
    3. The system must provide the navigational instructions to the user to the bicycle rack of his/her choice
    4. Users must be able to click on a bicycle rack location to view more information regarding it:
       1. Total number of slots
       2. Sheltered/Unsheltered

## View saved locations

* + 1. The user must be logged in to view his/her saved locations
    2. The system must display the user’s list of saved locations
    3. The user must be able to save a favourite location
    4. The user must be able to remove a saved location
    5. The user can click on a saved location
    6. System will redirect user to the route planning interface with the saved location inputted as the destination

## View history of cycling activity

* + 1. The user must be logged in to view his/her history of cycling activity
    2. The system must display the user’s cycling activity for the past 1 year
    3. The details of each cycling instance include the following:
       1. Date and time of event
       2. Start and end locations
       3. Distance travelled
       4. Duration of travel
       5. Cycling route displayed on map

# NON-FUNCTIONAL REQUIREMENT

## Performance

* + 1. After the query is made, results should be returned after 5 seconds.
    2. The GPS System must be accurate up to 5m.
    3. Response time between pages should be less than 2 seconds
    4. Upon start up, the system must be fully functional within 15 seconds.

## Reliability

* + 1. The system shall be available at all times.
    2. The system should not crash due to erroneous actions from the user.
    3. Upon system reboot, full functionality must be restored under 5 minutes.

## Security

* + 1. The application must be able to prevent denial-of-service (DoS) attacks
    2. The application will implement Secure Hash Algorithm (SHA) to perform salt-hashing on all the passwords before storing into the database.
    3. Users should not be able to access sensitive information to account and passwords

## Supportability

* + 1. Supported on both Android and iOS platforms

## Usability

* + 1. The system must reduce short-term memory load
       1. Users do not need to memorise the routes as the navigational instructions will be provided during the trip
       2. The system must provide informative feedback to the user
       3. The application must inform the user if the GPS is not available.
       4. The application will inform the user if he/she is out of the route during the trip.

## Maintainability

* + 1. An electronic notice will be issued and displayed on the application at least one week prior to maintenance, stating the time and date clearly.

## Scalability

* + 1. An object-oriented approach should be used when developing the application to allow integration of new functional requirements.

## Learnability

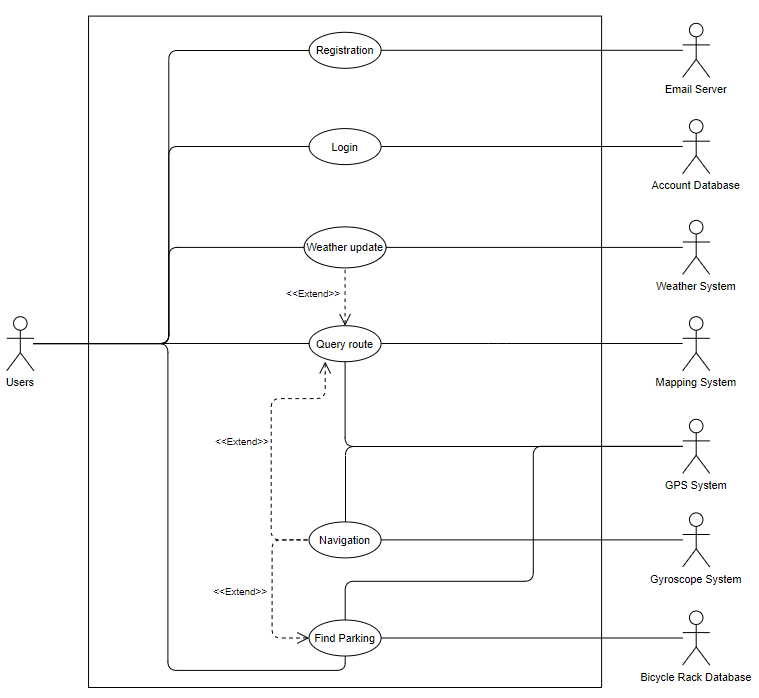
* + 1. The average user learning time should be less than a day
    2. 90% of the new users must be able to make a query within 5 minutes of using the application.

# DATA DICTIONARY

|  |  |
| --- | --- |
| **Term** | **Definition** |
| User | A user is a person who is using the application. |
| PCN | A network of cycling paths (Park Connector Network) that connects the various parks in Singapore. |
| GPS | Global Positioning System is responsible for tracking the user’s current location. This is used together with the Wi-Fi positioning system. |
| Query | Query is a feature that allows the user to check the most suitable path to be taken to the destination. |
| Route | An existing cycling path from the start to end location specified by the user. |
| Weather Forecast | Weather Forecast is a feature that is used to inform the user if the path is unsuitable to be taken due to poor weather. |
| DoS attack | A Denial-Of-Service attack is an attack that shuts down the network, making it inaccessible to its intended users. |
| IOS/Android platform | The various different platforms where the application is supposed to be operated on. |
| Gyroscope | A hardware used in the phone to sense the linear orientation of the mobile phone. |
| Salt | A salt is a randomly generated string that is used an additional input to a one-way function that hashes data. |
| SHA | Secure Hashing Algorithm is designed to keep data secured with the use of cryptographic functions. |
| API | Application Programming Interface is a software intermediary that allows two applications to communicate. |
| Object-Oriented approach | It is a computer programming model that organizes software design around data, or objects, rather than functions and logic. |
| Destination | A location that a user wants to reach via cycling. |

# USE CASE MODEL

## Use Case Diagram



## Use Case Descriptions

### Query route

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | 1.0 | | |
| Use Case Name: | Query route | | |
| Created By: | Wu Jingyuan | Last Updated By: | Amadeus Koh |
| Date Created: | 2/2/2021 | Date Last Updated: | 18/2/2021 |

|  |  |
| --- | --- |
| Actor: | User, Mapping System, Weather System, GPS System |
| Description: | User finds a cycling route to intended destination |
| Preconditions: | User has mobile data and location services enabled |
| Postconditions: | User successfully finds a route to the destination |
| Priority: | 1 |
| Frequency of Use: | Frequently |
| Flow of Events: | 1. User clicks on “Route Me” 2. QueryController queries GPS System for location of the user 3. GPSSystem returns the location of the user as start location 4. User inputs destination 5. QueryController queries MappingSystem for route 6. MappingSystem returns the route and system displays it 7. QueryController queries WeatherSystem for weather 8. WeatherSystem returns weather conditions 9. User successfully obtained route to destination |
| Alternative Flows: | 2a. User inputs custom start location   * custom start location is used instead of user’s current location   4a. Invalid user input   * system displays error message, return to 4   9a. Weather conditions along the route are bad   * system will warn of bad weather |
| Exceptions: | EX1. Systems are down   * System displays error message |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

### Find Racks

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | 2.0 | | |
| Use Case Name: | Find Racks | | |
| Created By: | Wu Jingyuan | Last Updated By: | Cheah Mun Yan, Grace |
| Date Created: | 2/2/2021 | Date Last Updated: | 18/2/2021 |

|  |  |
| --- | --- |
| Actor: | User, GPSSystem, BicycleRackDatabase, MappingSystem |
| Description: | User finds the location of the 3 closest bicycle racks to his/her current location |
| Preconditions: | User has mobile data and location services enabled |
| Postconditions: | User finds the route to one of the 3 closest bicycle racks to his/her current location |
| Priority: | 1 |
| Frequency of Use: | Occasionally |
| Flow of Events: | 1. User clicks on “Find Bicycle Racks” 2. RackController queries GPSSystem for the location of the user 3. GPSSystem returns the location 4. RackController queries BicycleRackDatabase for 3 closest bicycle racks 5. BicycleRackDatabase returns the locations and system displays it 6. User chooses one location and system displays details 7. User clicks on “Bring me there” 8. QueryController queries MappingSystem for route 9. MappingSystem returns route and system displays it |
| Alternative Flows: | 5a. Closest bicycle rack is more than 1km away   * System informs user |
| Exceptions: | EX1. Systems are down   * System displays error message |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

### Navigation

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | 3.0 | | |
| Use Case Name: | Navigation | | |
| Created By: | Wu Jingyuan | Last Updated By: | Amadeus Koh |
| Date Created: | 2/2/2021 | Date Last Updated: | 18/2/2021 |

|  |  |
| --- | --- |
| Actor: | User, GPS System, Gyroscope System |
| Description: | User is guided by the system to destination |
| Preconditions: | User has mobile data and location services enabled |
| Postconditions: | User arrives at the destination or navigation is cancelled by user |
| Priority: | 1 |
| Frequency of Use: | Frequently |
| Flow of Events: | 1. User begins navigation 2. NavigationController queries GPS System for the current location of user 3. GPS System returns current location 4. System queries Gyroscope System for the direction user is facing 5. Gyroscope System returns the direction 6. NavigationContoller updates user’s location and direction on the navigation map 7. Repeat 2-6 until destination is reached or user cancels the navigation |
| Alternative Flows: | 6a. User is off-route   * System displays an off-route alert |
| Exceptions: | EX1. Systems are down   * System displays error message |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

### Check Weather

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | 4.0 | | |
| Use Case Name: | Check Weather | | |
| Created By: | Amadeus Koh | Last Updated By: | Amadeus Koh |
| Date Created: | 17/2/2021 | Date Last Updated: | 18/2/2021 |

|  |  |
| --- | --- |
| Actor: | User, Weather System |
| Description: | User checks the weather forecast |
| Preconditions: | User has mobile data enabled |
| Postconditions: | User received the desired weather forecast for specific regions in Singapore |
| Priority: | 1 |
| Frequency of Use: | Frequently |
| Flow of Events: | 1. User clicks on “Weather Map” 2. WeatherController queries the WeatherSystem for the weather forecast 3. WeatherController returns weather information to the Weather Map 4. Weather forecast is displayed on Weather Map 5. User checks weather forecast for specific regions of Singapore |
| Alternative Flows: | - |
| Exceptions: | EX1. Systems are down   * System displays error message |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

### User registration

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | 5.0 | | |
| Use Case Name: | User registration | | |
| Created By: | Wu Jingyuan | Last Updated By: | Cheah Mun Yan, Grace |
| Date Created: | 3/2/2021 | Date Last Updated: | 19/2/2021 |

|  |  |
| --- | --- |
| Actor: | User, Account Database, Email server |
| Description: | User registers for an account |
| Preconditions: | Mobile data is enabled |
| Postconditions: | User successfully registers for an account |
| Priority: | 1 |
| Frequency of Use: | Rarely |
| Flow of Events: | 1. User presses “Create Account” 2. System prompts user to input their email and a password 3. User inputs email and password 4. System queries email address from the related email server 5. Email server sends OTP to email address 6. System prompts user for OTP 7. User successfully inputs OTP 8. System hashes the password and sends the hashed password and the email address to Account Database 9. System informs user an account has been created |
| Alternative Flows: | 4a. Invalid email   * System warns user and returns to 2   7a. Invalid OTP   * System prompts user to retry, returns to 6 |
| Exceptions: | EX1. Database is down   * System displays error message |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

### User login

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | 6.0 | | |
| Use Case Name: | User login | | |
| Created By: | Wu Jingyuan | Last Updated By: | Guo Feiyan |
| Date Created: | 3/2/2021 | Date Last Updated: | 18/2/2021 |

|  |  |
| --- | --- |
| Actor: | User, Account database |
| Description: | User logins to their account |
| Preconditions: | Mobile data is enabled |
| Postconditions: | User successfully logins to account |
| Priority: | 1 |
| Frequency of Use: | Frequently |
| Flow of Events: | 1. User presses “Login” 2. System prompts user for email and password 3. User inputs email and password 4. Password Hasher will hash the password 5. Account Controller queries account database to check validity of login details 6. Account database returns validity of login details 7. System informs user of successful login |
| Alternative Flows: | 6a. Invalid login details   * System warns user and returns to 2 |
| Exceptions: | EX1. Database is down   * System displays error message |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

# UI MOCKUP

## 

## Diagram 1: App Loading Page

## 

## Login

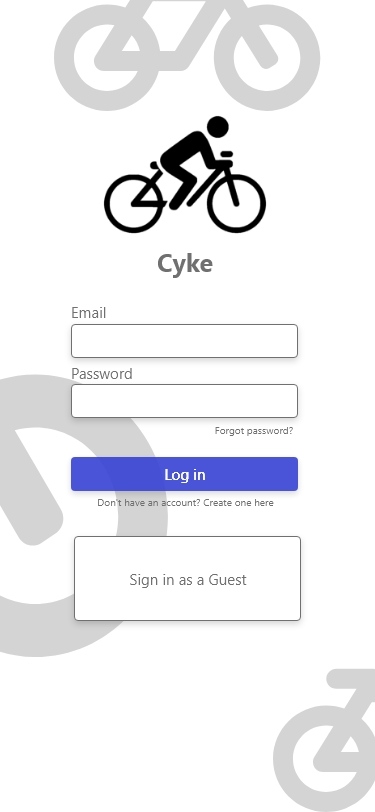
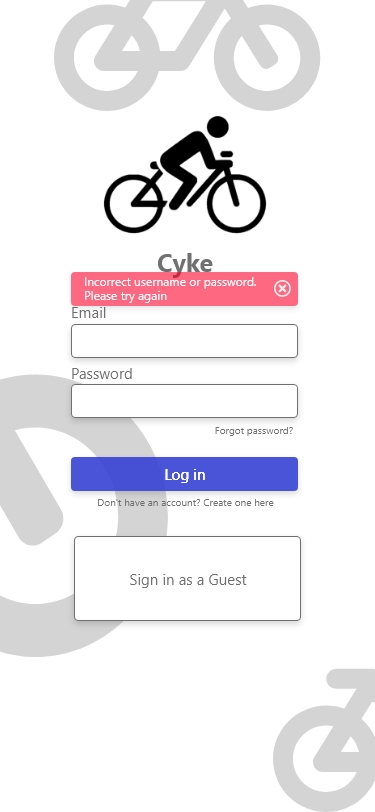
 

Diagram 2: Login Page Diagram 3: Incorrect Login field

## Creating Account

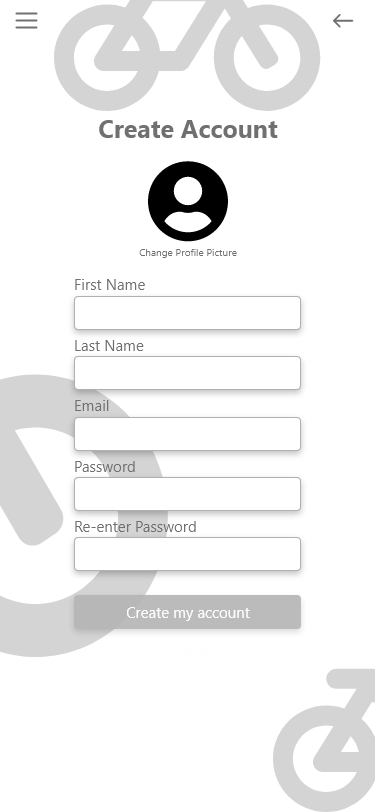
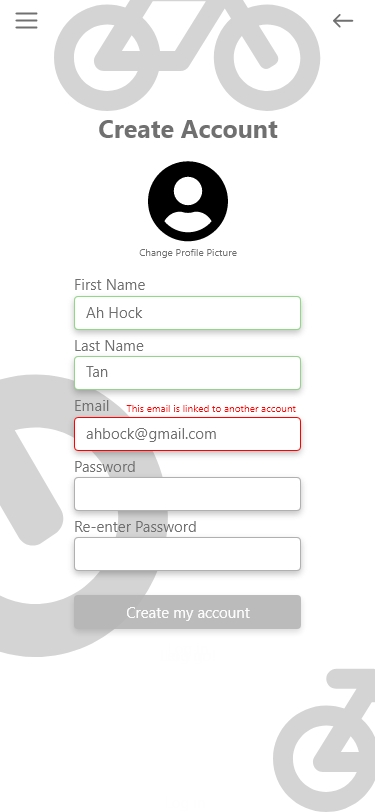
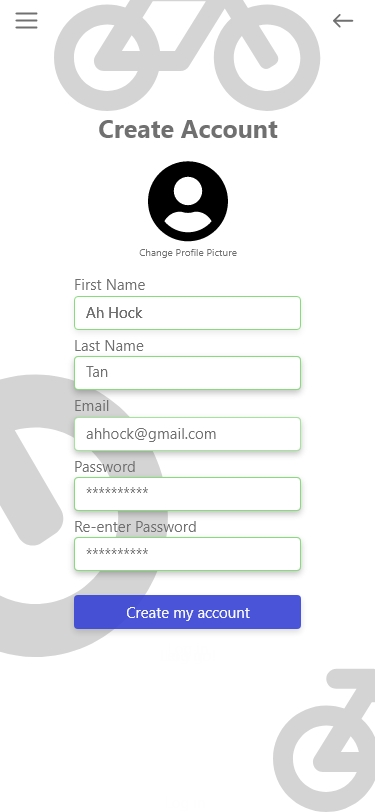
  

Diagram 4: Diagram 5: Diagram 6:

Account Creation Invalid email All fields validated

## Home Page

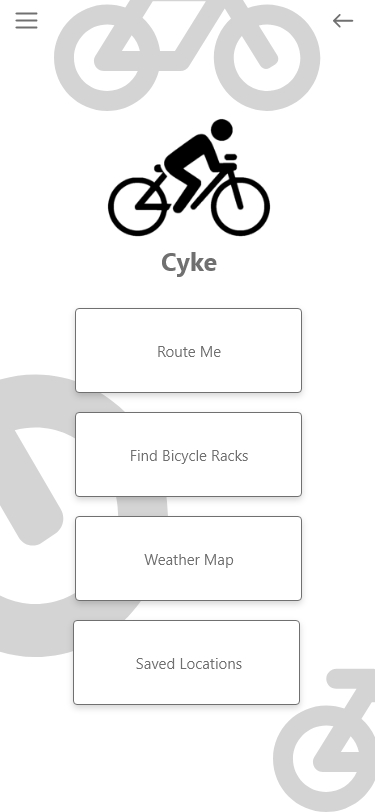
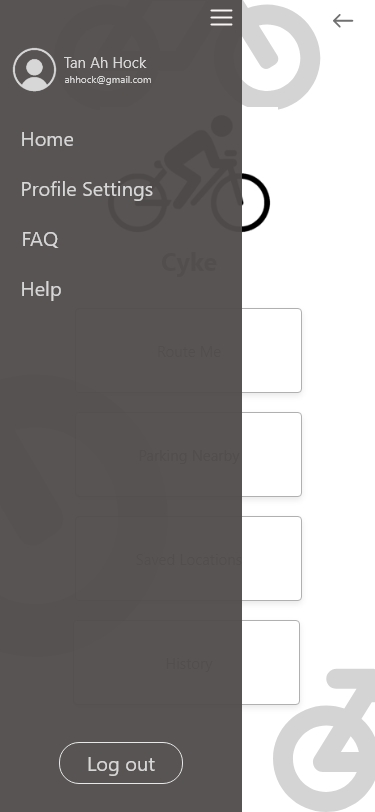
 

Diagram 7: Home Page Diagram 8: Side Bar

## Routing

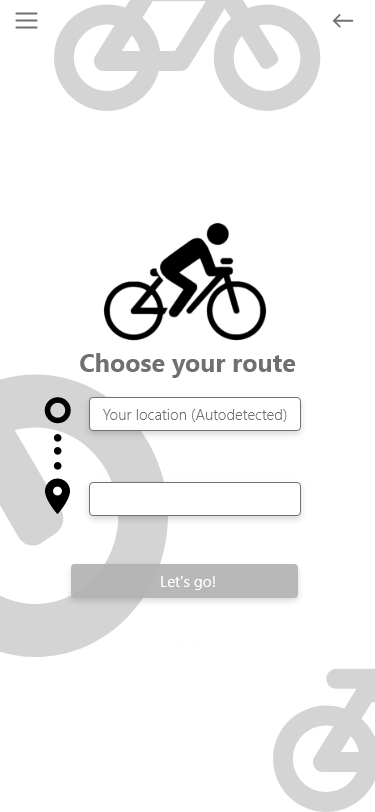
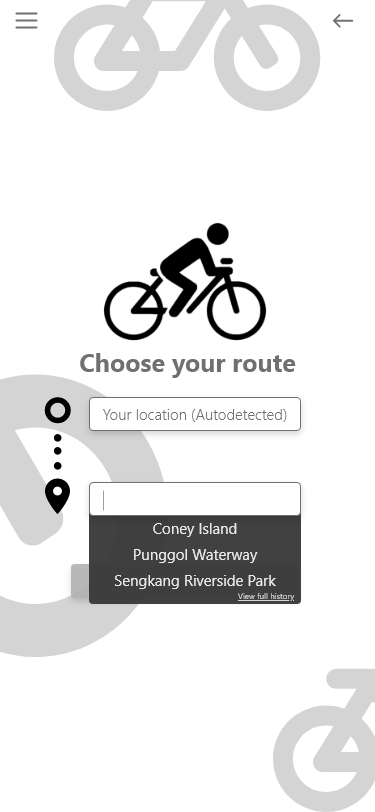
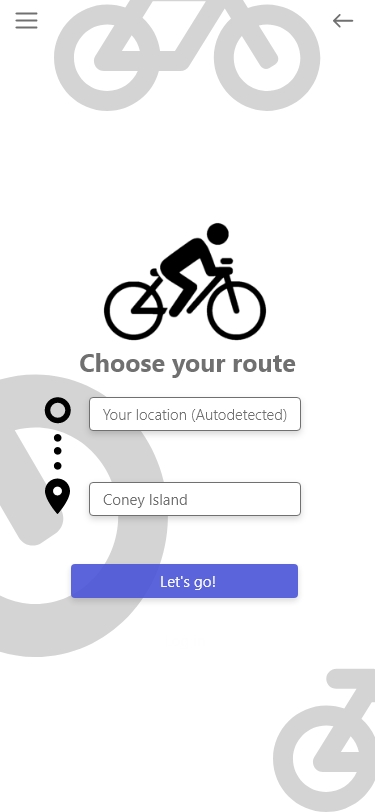
**  **

Diagram 9: Diagram 10: Diagram 11:

Route Input History suggestions Valid Route Input

## Navigation

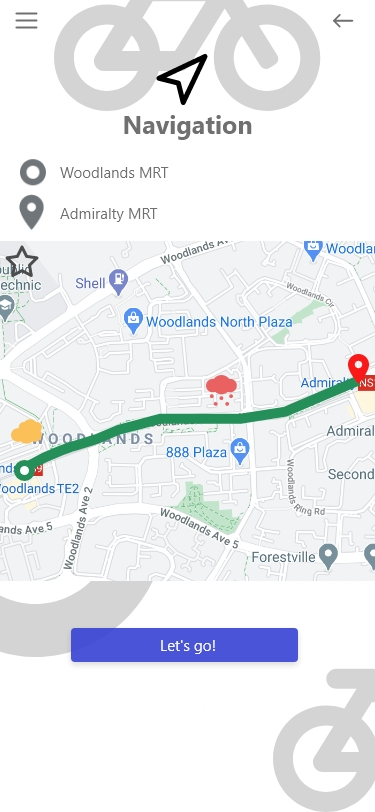
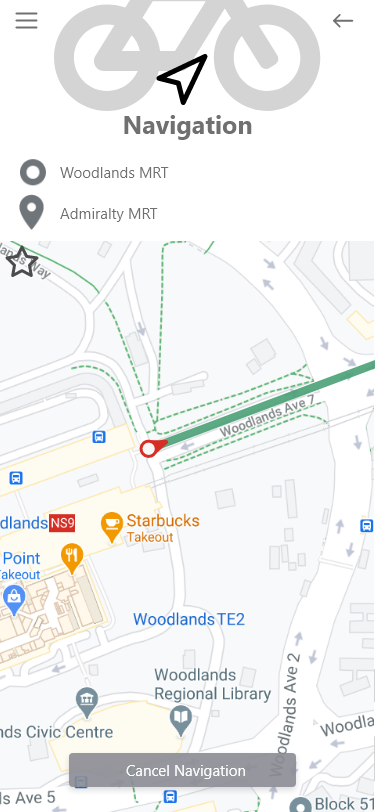
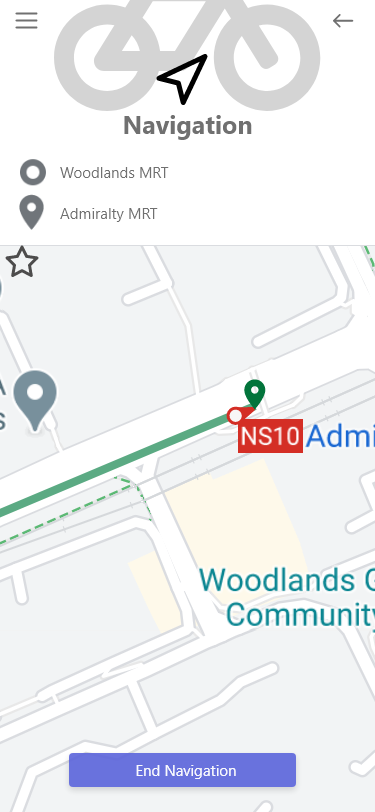
**  **

Diagram 12: Navigation Diagram 13: Diagram 14:

Navigation Process End of Navigation

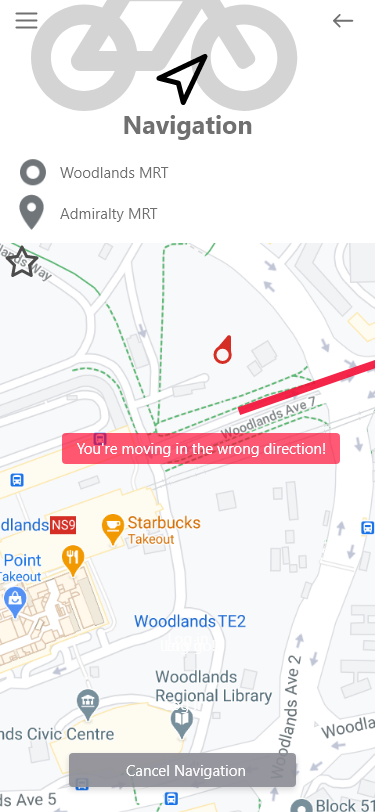
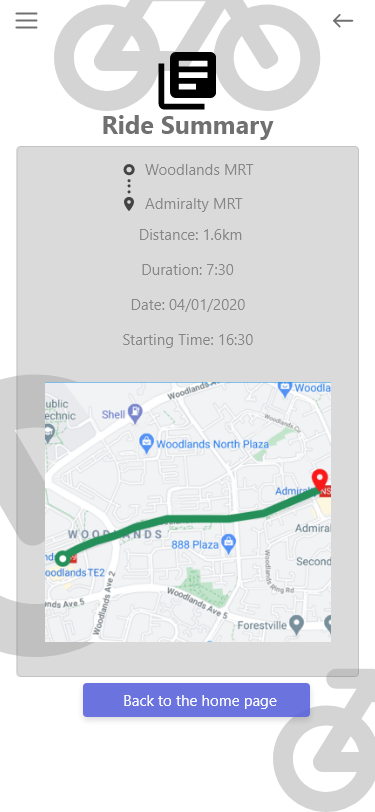
** **

Diagram 15: Diagram 16:

Going off-route Ride Summary

## System Errors

****

Diagram 17: No connection

## Parking Nearby

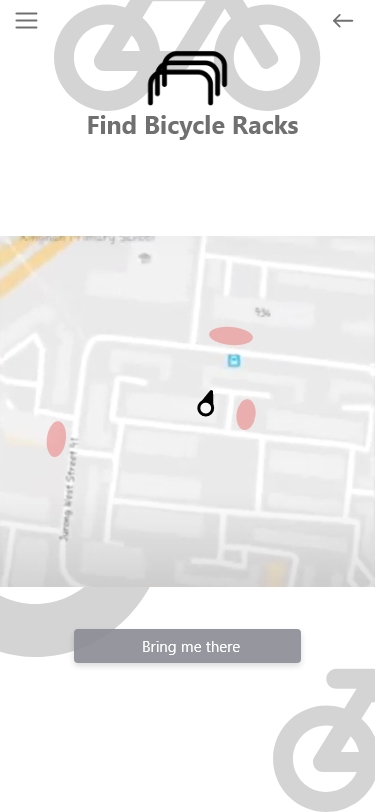
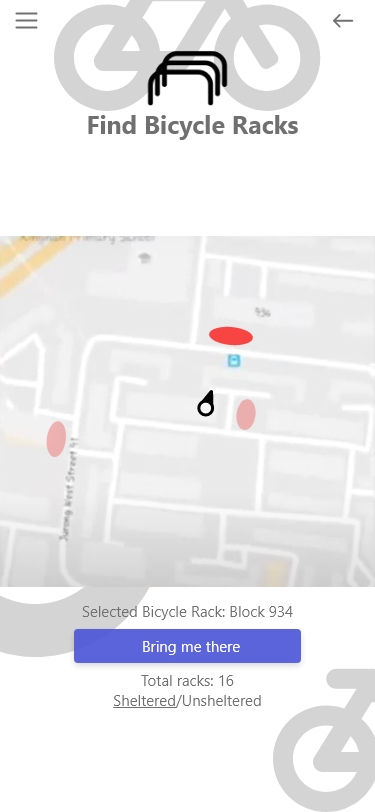
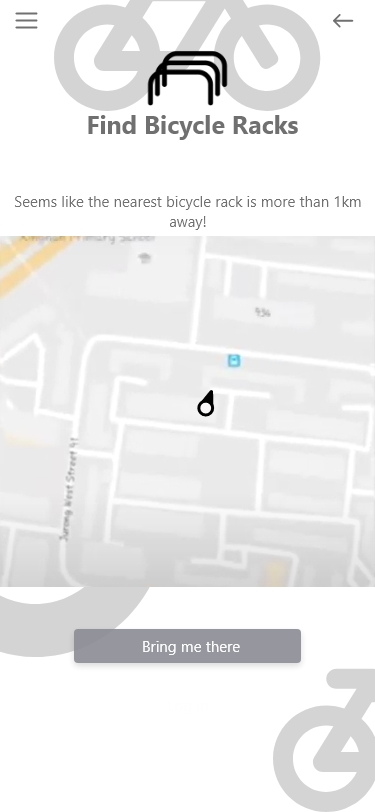
**  **

Diagram 18: Diagram 19: Diagram 20:

Display nearest racks Selecting Nearest Rack Nearest Rack is far away

## Saved Locations

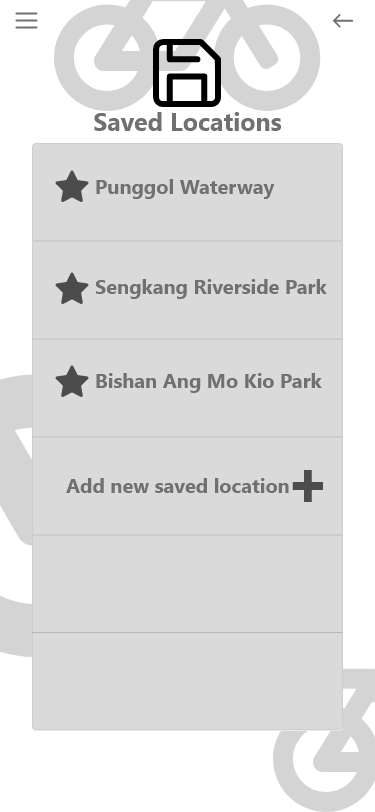
****

Diagram 21: Saved locations

## History

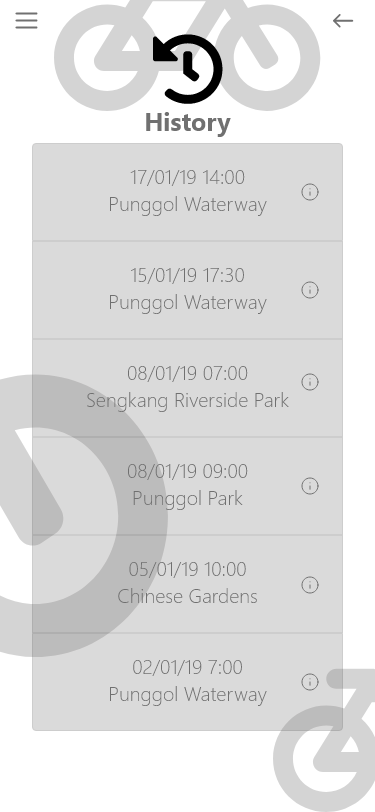
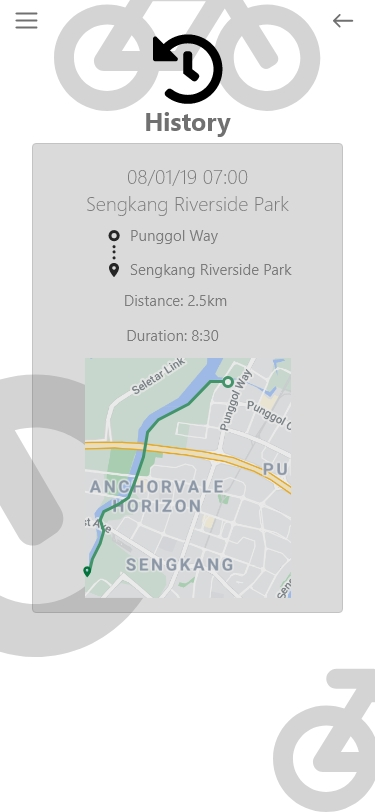
** **

Diagram 22: Diagram 23:

Log of History Detailed History

## Weather Forecast

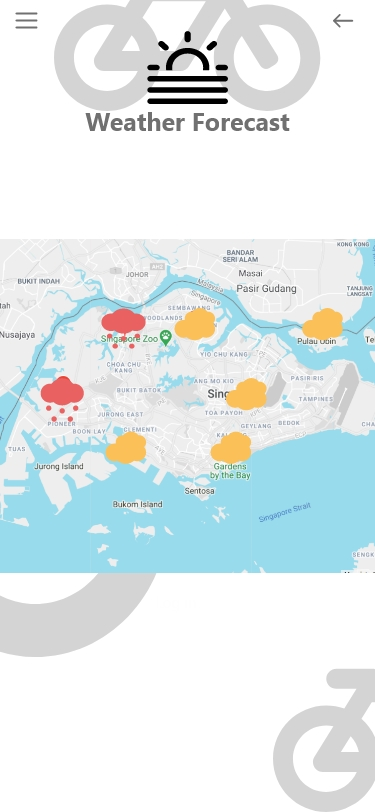
****

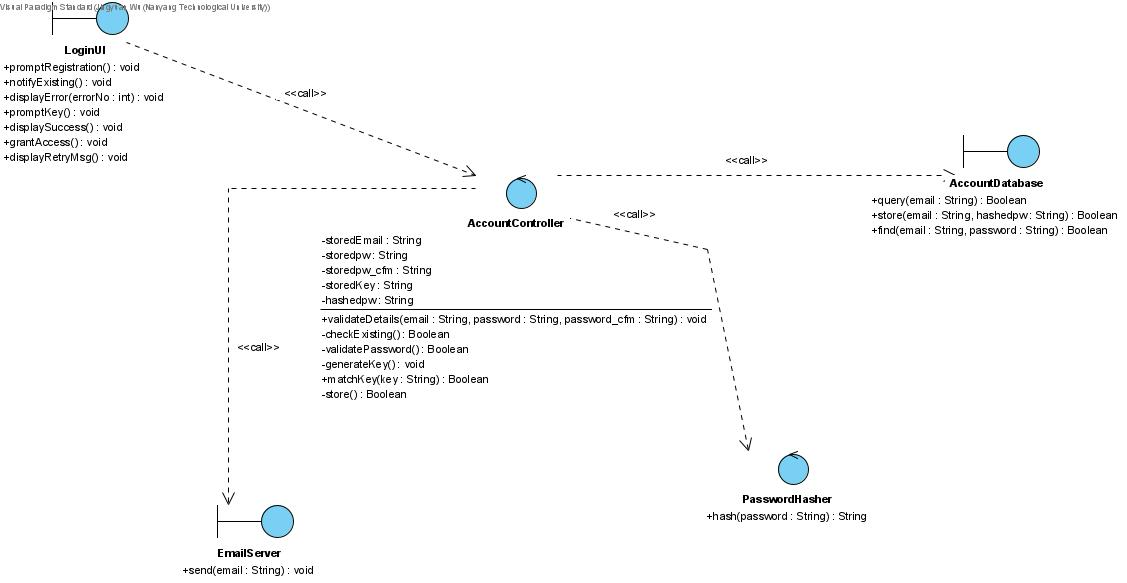
Diagram 24:

Weather Forecast

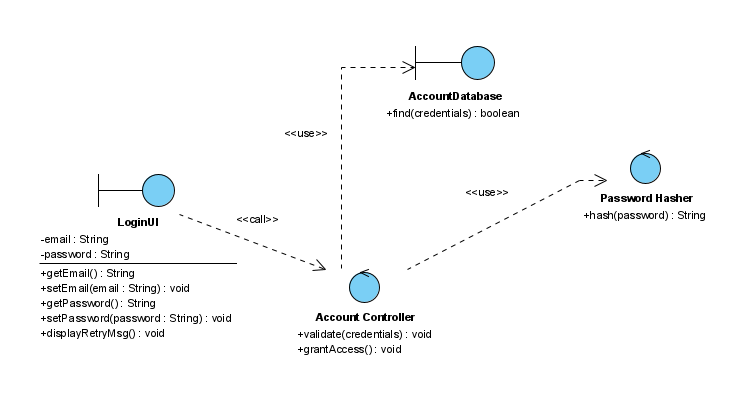
# UML MODELS

## Class Diagram

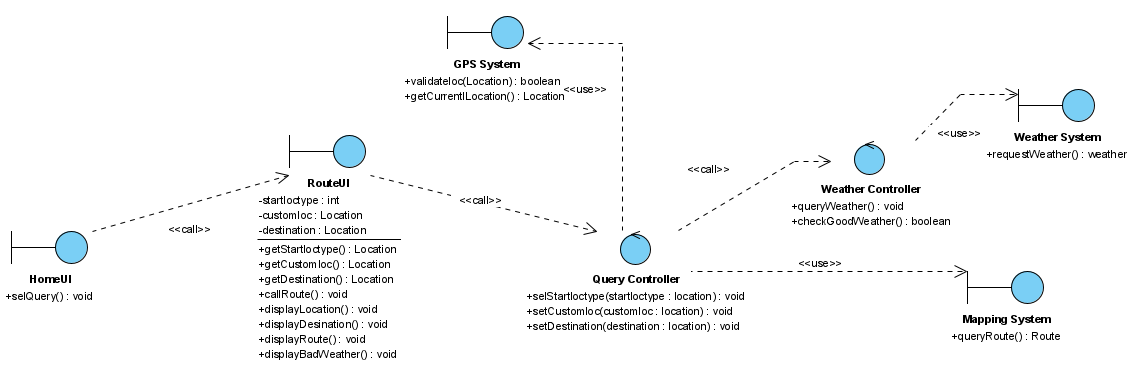
### Registration



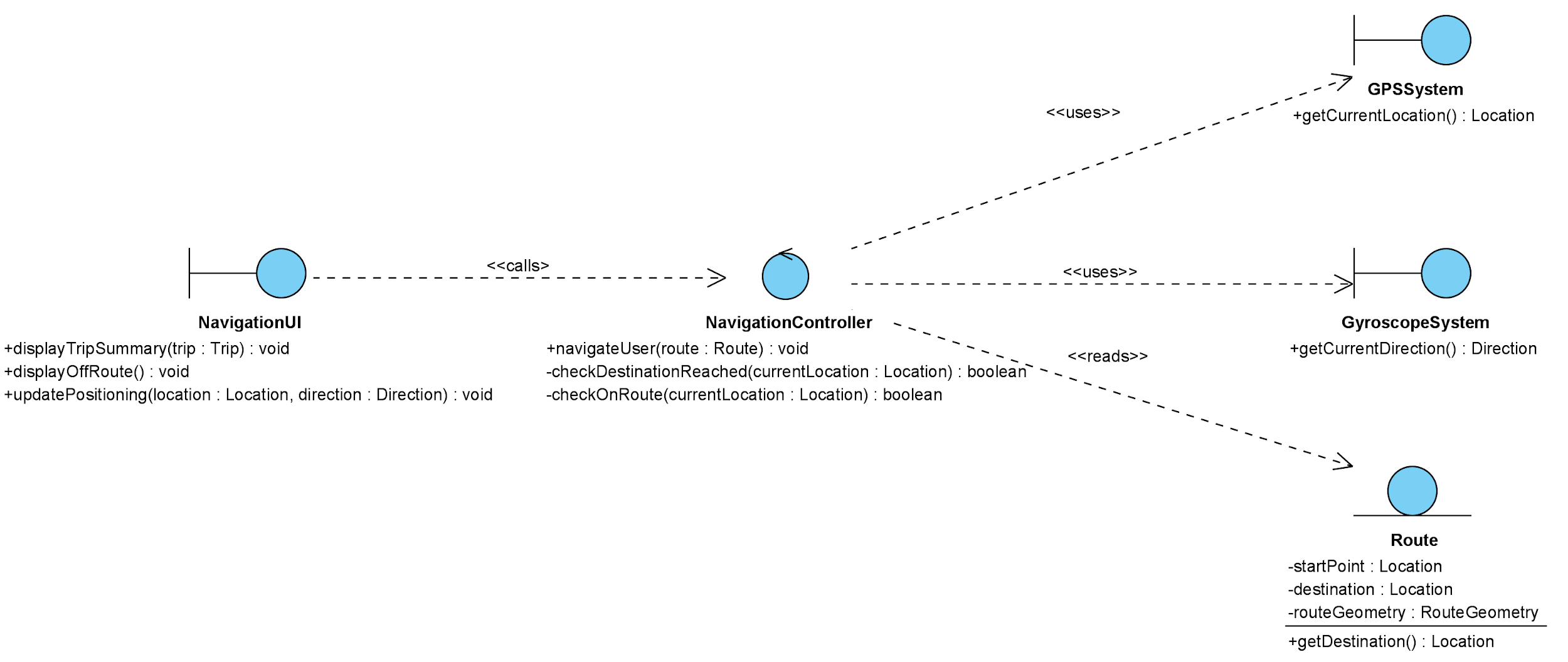
### Login



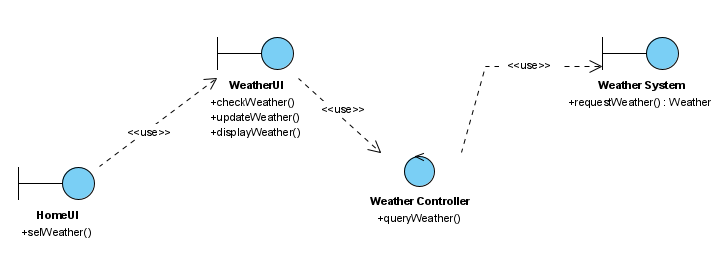
### Query Route



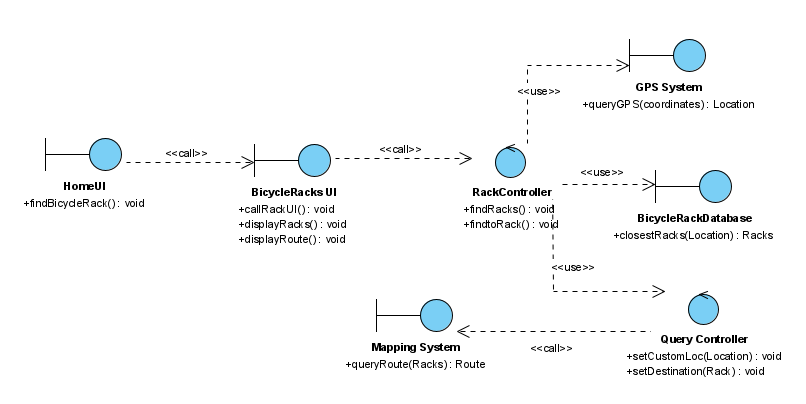
### Navigation



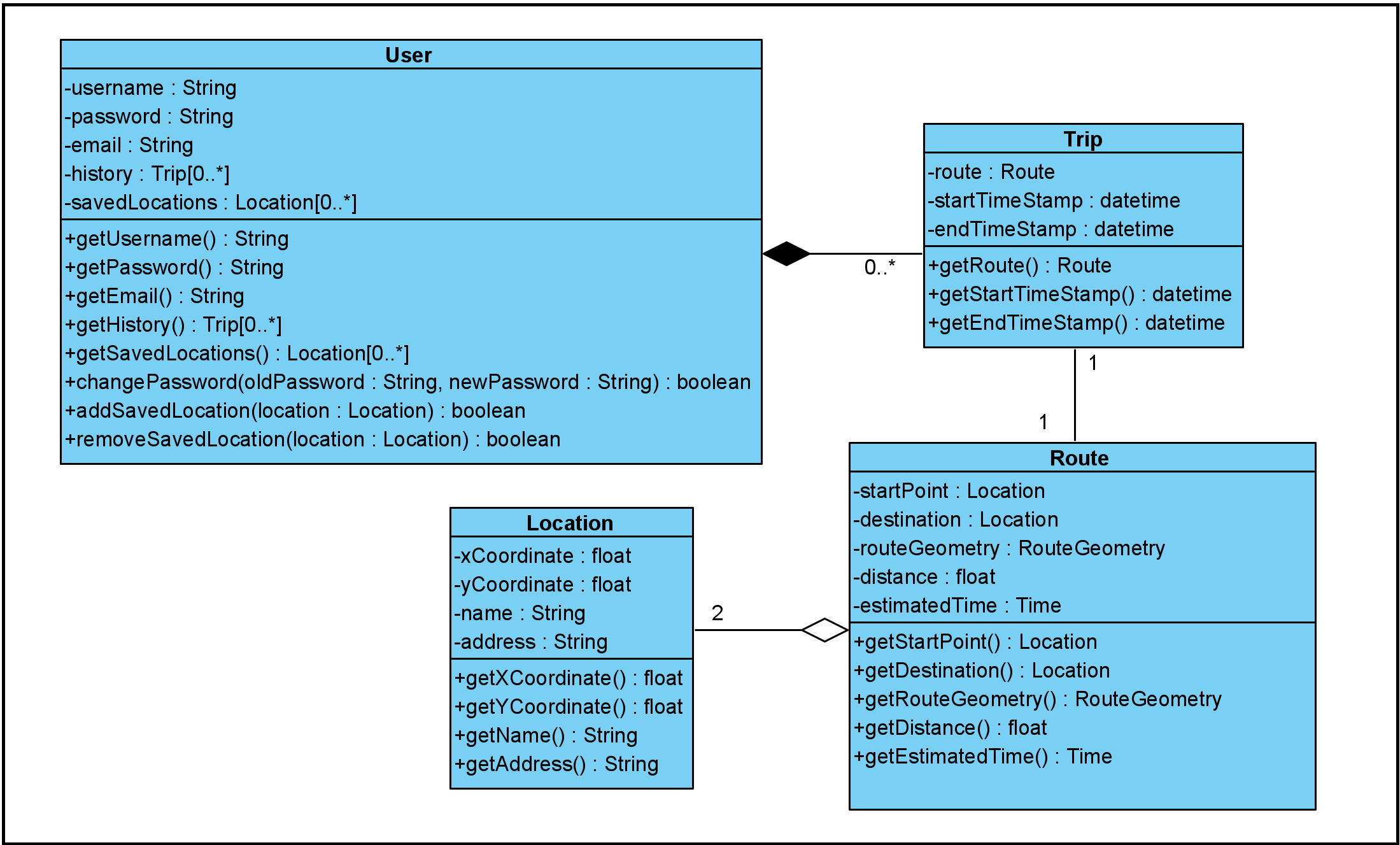
### Check Weather



### Find Racks



### Class Diagram of Entity Classes



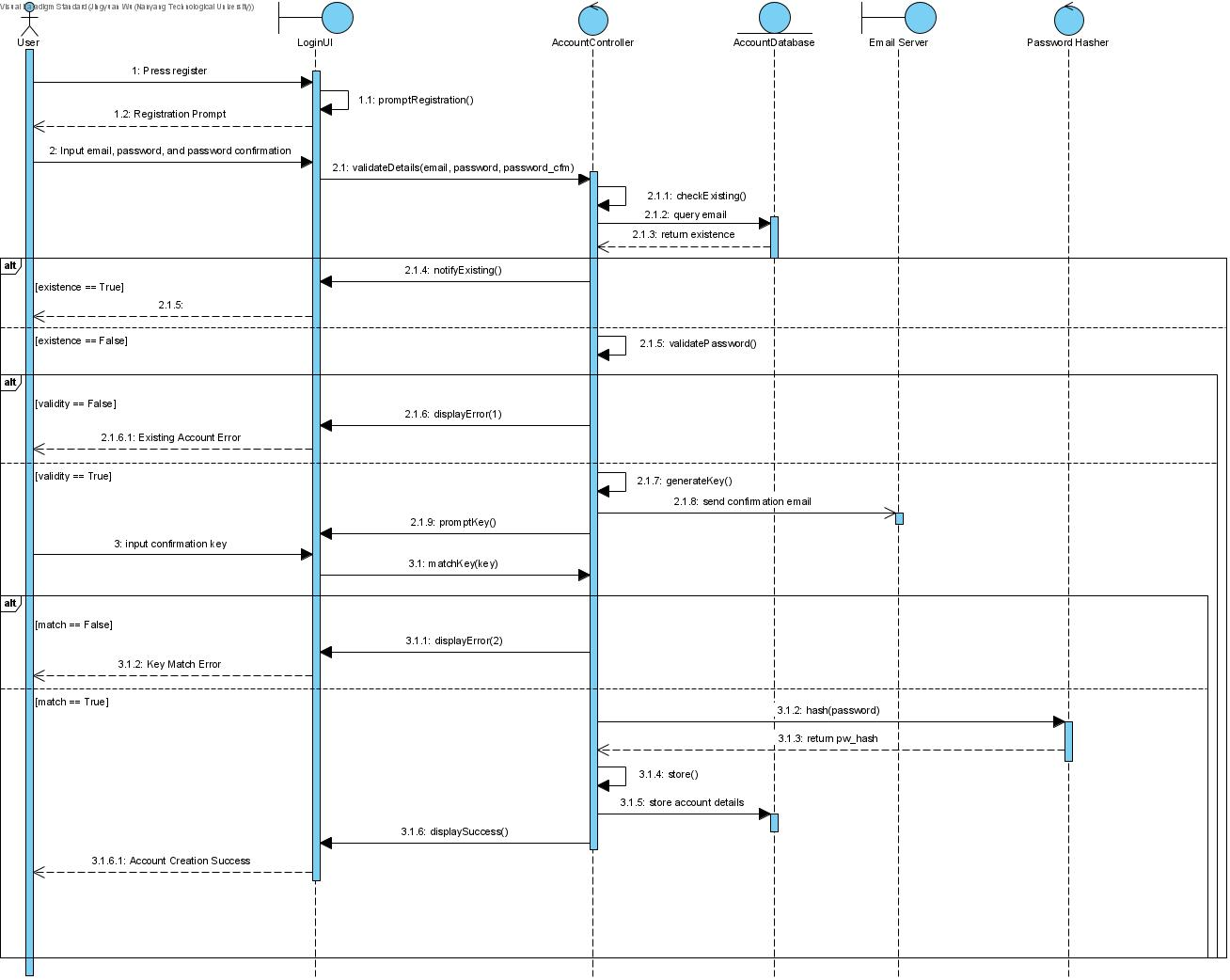
## Description of Boundary and Control Classes

|  |  |
| --- | --- |
| **Boundary Classes** | **Description** |
| RegistrationUI | Registration form for user |
| LoginUI | Allows user to login using registered account |
| HomeUI | Home Page with main menu options |
| WeatherUI | View weather conditions and forecast throughout Singapore through a Weather Map |
| BicycleRackUI | View closest racks to user’s current location |
| NavigationUI | Navigation of route to user’s destination |
| RouteUI | Form for user’s inputted start point and destination to search for a route |
| SavedLocationUI | Displays user’s list of saved locations |
| WeatherSystem | Getting weather conditions and forecasts of Singapore |
| GPSSystem | Getting user’s current coordinates |
| BicycleRackSystem | Getting bicycle rack locations in Singapore |
| MappingSystem | Finding route from user’s start location to destination |
| GyroscopeSystem | Getting user’s current direction |
| EmailSystem | Sending email to the user for verification when registering for a new account |

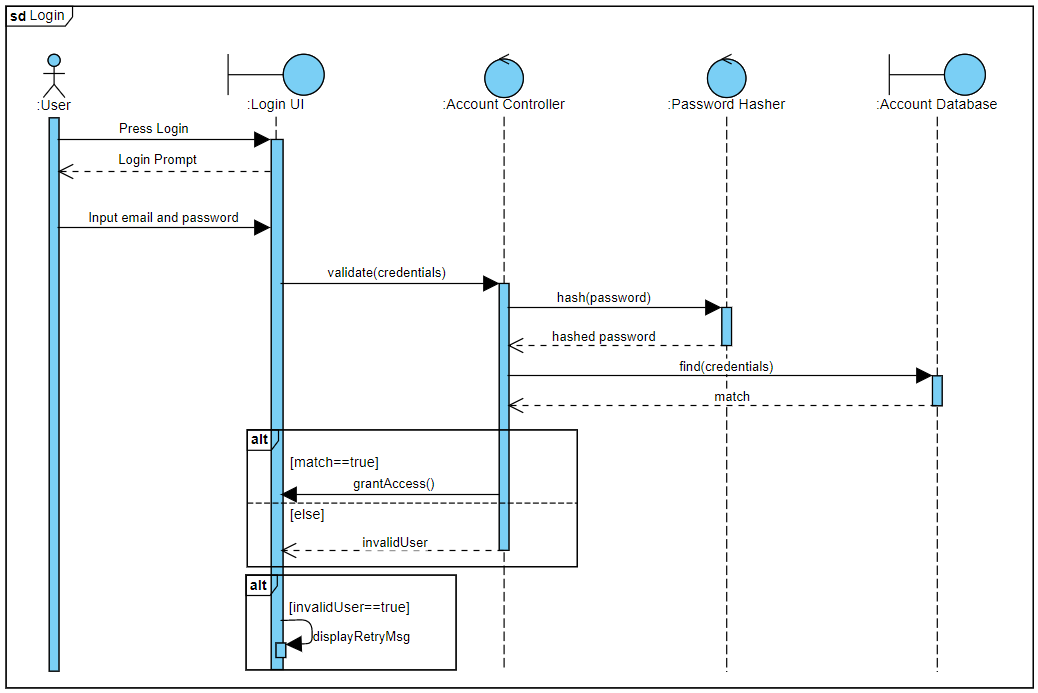
|  |  |
| --- | --- |
| **Control Classes** | **Description** |
| AccountController | Verifies account details for registration and login |
| WeatherController | Queries for the weather data and display the corresponding weather icon |
| RackController | Handles rack information and queries |
| NavigationController | Handles the tracking of user location and feeding of navigation instructions |
| QueryController | Formats the input destination data from user to find a route for the user |
| PasswordHasher | Hashes the password for the user account before storing into database |

## Sequence Diagrams

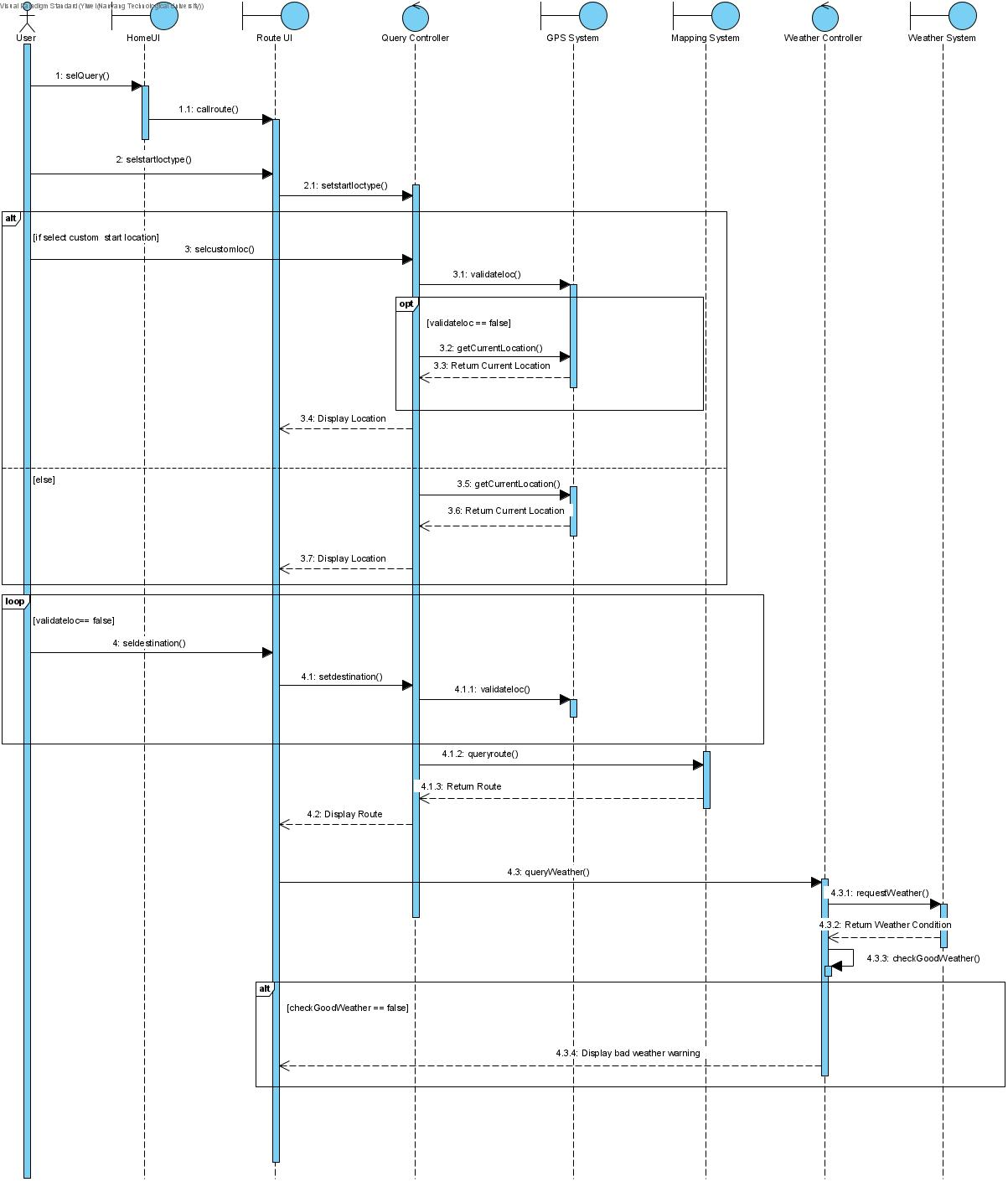
### Registration



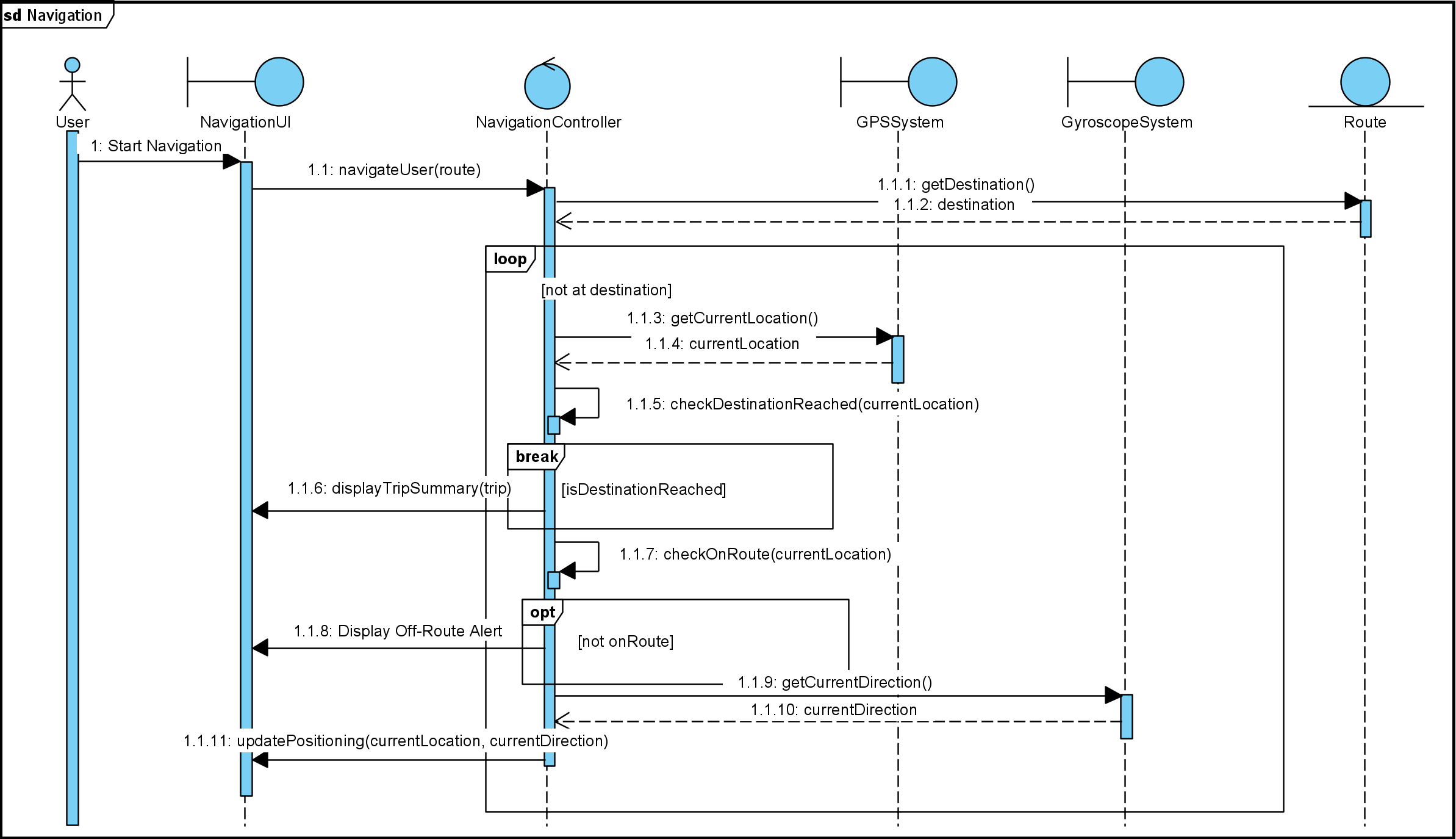
### Login



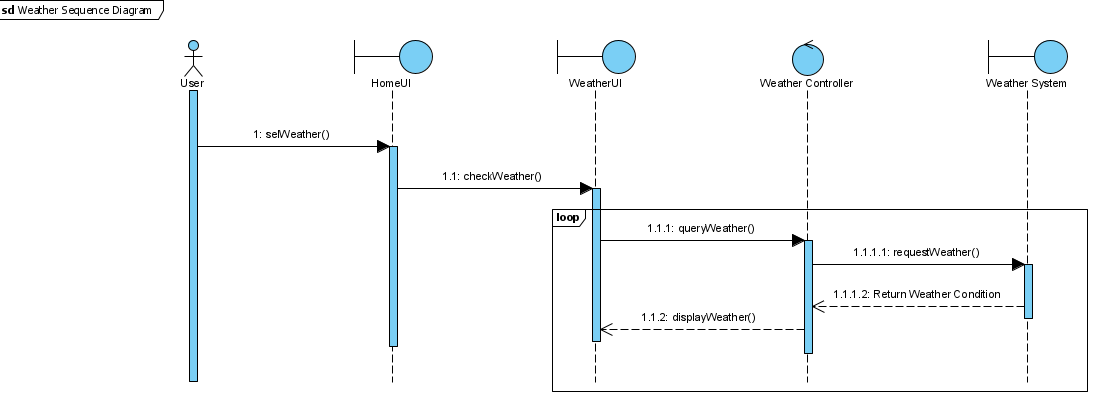
### Query Route

****

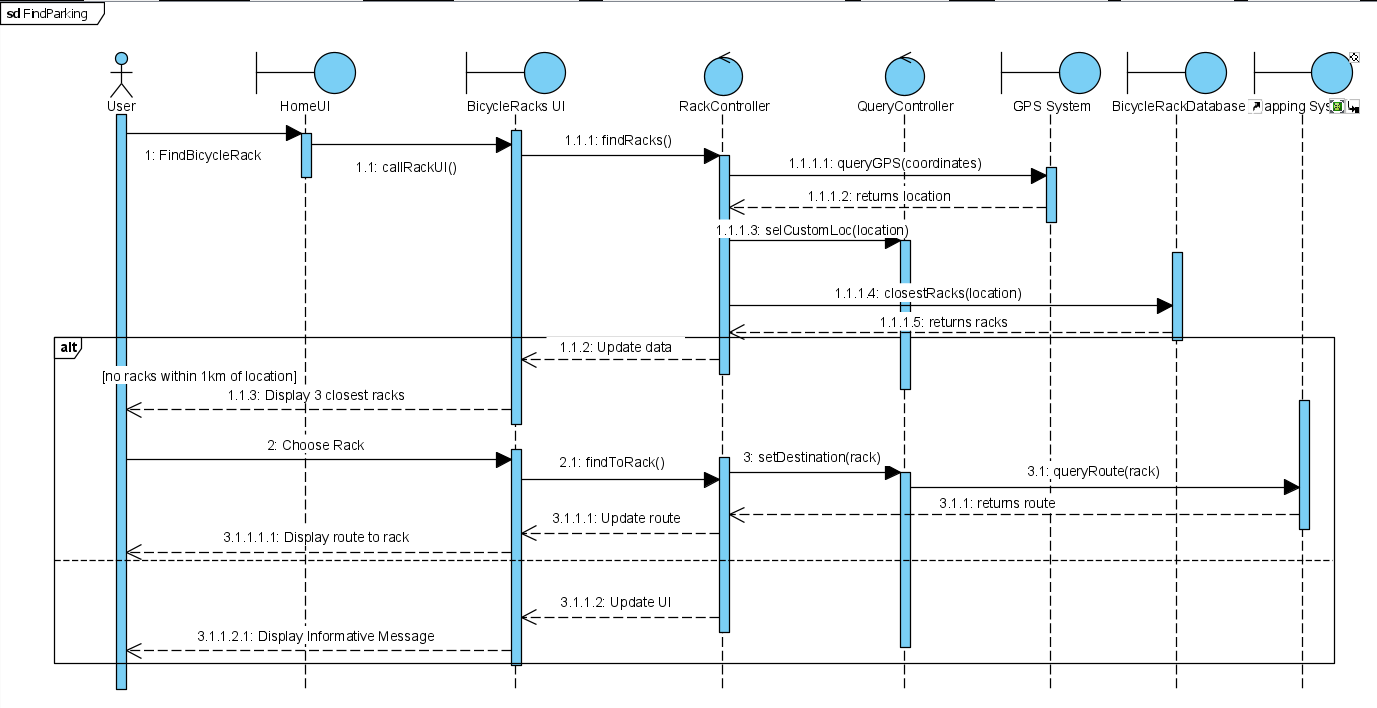
### Navigation

****

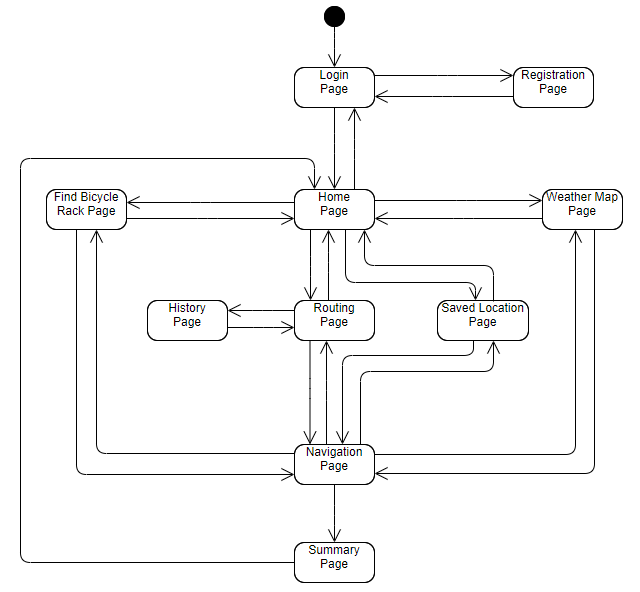
### Check Weather



### Find Racks



## Dialog Map

****