Software Requirements Specification

for

Route Planner

Version 1.0 approved

Prepared by

Huang Yongjian U2320879D

Nadya Yuki Wangsajaya U2320059K

Nang Kal San Hom U2321661K

Serena Nathania Sihombing U2222257K

Shiu Lok Chun, Wesley U2321673A

Syed Ali Redha Alsagoff U2340588F

Nanyang Technological University, Team 52

10/11/2024

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Nang Kal San Hom	17/11/2024	Finalised SRS Documentation	1.0

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to specify the software requirements for the RoutePlanner application, Revision 1.0. RoutePlanner is an innovative mobile application designed to generate custom running routes based on user preferences, specifically tailored to the desired running distance and inclusion of specific landmarks. This SRS document outlines the core functionality of the app, including the generation of round-trip routes, real-time navigation assistance, and user tracking features.

This document describes all essential features of RoutePlanner, with a focus on generating safe and personalized running routes that accommodate weather conditions and user preferences. The SRS covers the integration of external datasets, such as the LTA Covered Linkway Dataset and the 2-Hour Weather Forecast API, to ensure user safety and comfort during unfavorable weather. It also explains the use of the GraphHopper routing engine for creating routes that match the user's desired distance and landmark preferences. This SRS includes all primary subsystems necessary for the app's route planning, real-time tracking, and run history functionalities.

The scope of this SRS covers the entire RoutePlanner application, ensuring that the specified software requirements lead to a fully functional product.

1.2 Document Conventions

This Software Requirement Specification (SRS) follows the following conventions.

1.2.1 Font
Text is presented in **Times New Roman**, size **11**

1.2.2 Formatting
Italic text is used for *citations* and *references*

1.2.3 Highlighting
Critical sections are bolded

1.2.4 Numbering
Sections are labeled in the format Section X.X

1.2.5 Terminology
Specific terminology is defined in Appendix A (Glossary)

1.2.6 References
External references are cited following APA 7th Edition style

1.2.7 Revision History
This document follows version 1.0 and was last revised on 10/11/24

1.2.8 Additional notes
Tables, diagrams, and charts are labeled and formatted according to IEEE standards.

1.3 Intended Audience and Reading Suggestions

This document is intended for all stakeholders involved. This includes the users of the app, the software developers of the app, project managers, marketing staff, and testers. The document contains details of the different functions of the app, its intended purpose and the reasoning for having such a feature. A detailed description of each feature is provided, and examples of the different app pages will be shown to allow better understanding of the app layout.

1.3.1 Developers
Developers are primarily concerned with the technical details of the system, including how it is supposed to work, how to implement features, and what constraints exist. They may find the System Features, Functional and Non-functional Requirements section most useful.

1.3.2 Project Managers
Project managers typically focus on understanding the scope, timeline, resource allocation, and risks associated with the project. They may find the Product Scope and Overall Description most useful.

1.3.3 Marketing staff
Marketing staff are interested in understanding the value the software provides to customers, and the features it offers. They may find the Functional and Non-functional Requirements and User Interface most useful.

1.3.4 Users
Users are concerned with the usability and functionality of the system, focusing on what the system can do for them. They may find the User Interface and Functional and Non-functional Requirements most useful.

1.3.5 Testers
Testers are responsible for validating that the system works according to the specifications. They may find the Functional and Non-functional Requirements most useful.

1.4 Product Scope

The primary objective of RoutePlanner is to enhance the running experience by addressing common challenges faced by runners, such as route planning, navigation, and weather concerns. The app aligns with the government's broader goals of building a smart nation, and promoting health and wellness by encouraging users to engage in regular physical activity. By integrating fitness and technology, the app aligns with corporate strategies to enhance user engagement and promote lifestyle improvements through digital solutions.

1.5 References

1.5.1 APIs

1.5.1.1 Routing
1.5.1.1.1 GraphHopper

1.5.1.2 Landmarks
1.5.1.2.1 OneMap

1.5.1.3 Weather
1.5.1.3.1 2-Hour Weather Forecast

1.5.1.4 Autosuggest locations
1.5.1.4.1 HERE

1.5.1.5 Database
1.5.1.5.1 Supabase
1.5.1.5.2 Prisma

1.5.1.6 Geocoding
1.5.1.6.1 GoogleMaps

1.5.2 Frameworks
1.5.2.1 Backend
1.5.2.1.1 NestJS
1.5.2.2 Frontend
1.5.2.2.1 React Native

1.5.3 Dataset
1.5.3.1 Covered Linkway
1.5.3.2 LTA

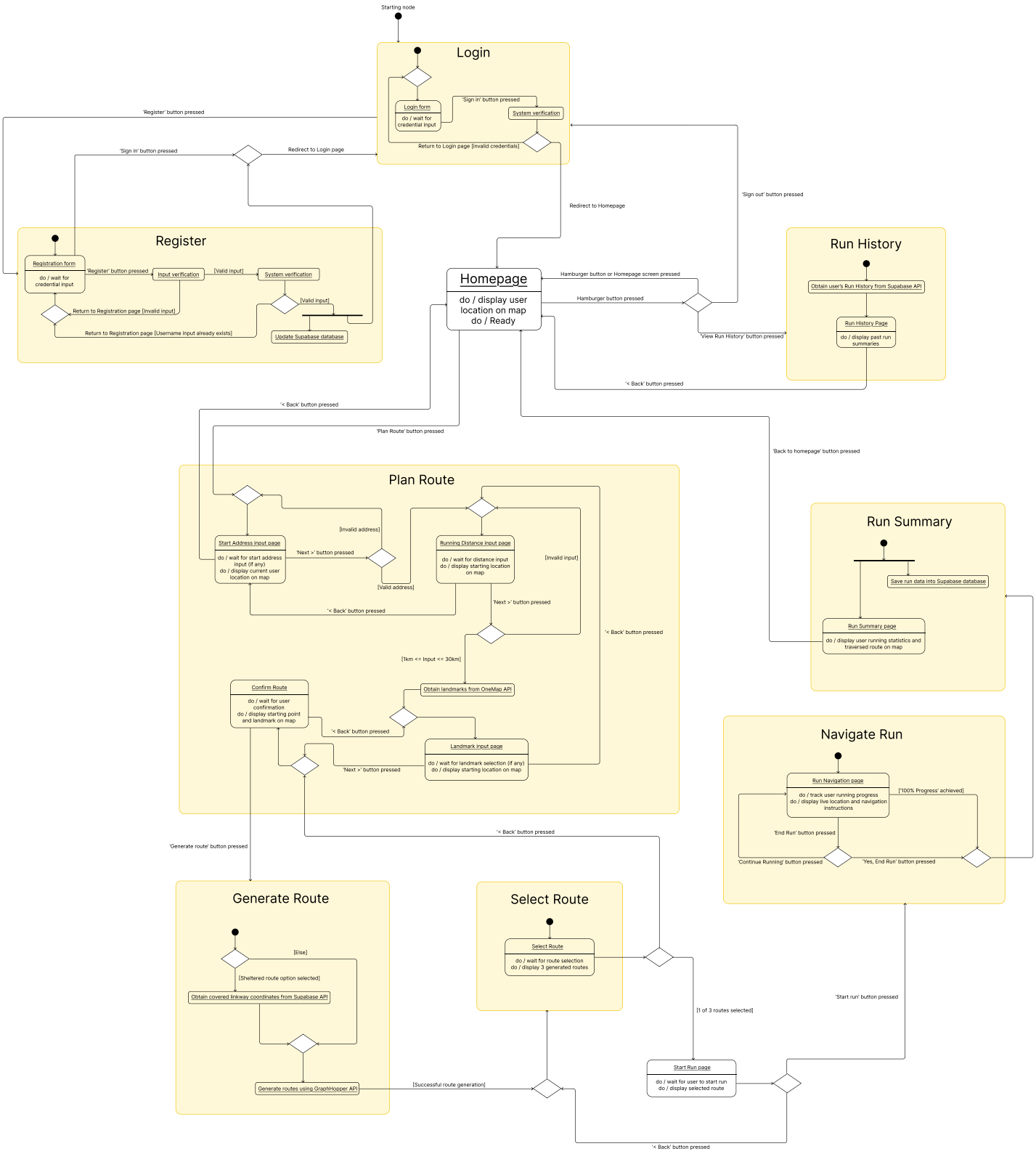
2. Overall Description

2.1 Product Perspective

RoutePlanner is designed to help users plan round-trip routes based on their desired distance, current weather conditions, and a specified landmark. The app integrates the LTA Covered Linkway Dataset to include sheltered pathways when weather conditions are unfavorable, adjusting the route based on real-time data from the 2-Hour Weather Forecast API. This ensures users can avoid exposure to rain or extreme weather by toggling the preference to generate a shelter-optimised route. GraphHopper is used to generate round-trip routes that align with the user's desired distance and any landmark that they wish to pass along the way, such as parks, monuments, or heritage trees. This allows users to customize their routes for specific fitness goals or sightseeing preferences while enjoying a more tailored outdoor experience.

In addition to route planning, RoutePlanner offers real-time tracking of users' runs, similar to GPS-based tracking, allowing them to follow the route they've selected step-by-step. As users run, the app provides navigation cues and updates, making it convenient for them to track their progress and stay on course. After completing a run, users can save their Run Summary into their Run History, inclusive of detailed data such as distance, time and the route traversed. This allows users to track their progress over time, set personal goals, and revisit their routes.

The image in the next page represents the dialog map for our project. This map visually outlines the flow and architecture of our application's user interface design.


Starting node
Login
Login form
do / wait for credential input
'Sign in' button pressed
System verification
Return to Login page [invalid credentials]
'Register' button pressed
'Sign in' button pressed
Redirect to Login page
Redirect to Homepage
'Sign out' button pressed
Register
Registration form
do / wait for credential input
'Register' button pressed
Input verification
[Valid input]
System verification
[Valid input]
Return to Registration page [Invalid input]
Return to Registration page [Username input already exists]
Update Supabase database
Homepage
do / display user location on map
do / Ready
Hamburger button or Homepage screen pressed
Hamburger button pressed
'View Run History' button pressed
Run History
Obtain user's Run History from Supabase API
Run History Page
do / display past run summaries
'< Back' button pressed
'< Back' button pressed
'Plan Route' button pressed
'Back to homepage' button pressed
Plan Route
Start Address input page
do / wait for start address input (if any)
do / display current user location on map
'Next >' button pressed
[Invalid address]
[Valid address]
Running Distance input page
do / wait for distance input
do / display starting location on map
[Invalid input]
'Next >' button pressed
'< Back' button pressed
'< Back' button pressed
[1km <= Input <= 30km]
Confirm Route
do / wait for user confirmation
do / display starting point and landmark on map
Obtain landmarks from OneMap API
'< Back' button pressed
Landmark input page
do / wait for landmark selection (if any)
do / display starting location on map
'Next >' button pressed
Run Summary
Save run data into Supabase database
Run Summary page
do / display user running statistics and traversed route on map
Navigate Run
Run Navigation page
do / track user running progress
do / display live location and navigation instructions
['100% Progress' achieved]
'End Run' button pressed
'Continue Running' button pressed
'Yes, End Run' button pressed
'< Back' button pressed
'Generate route' button pressed
Generate Route
[Else]
[Sheltered route option selected]
Obtain covered linkway coordinates from Supabase API
Generate routes using GraphHopper API
[Successful route generation]
Select Route
Select Route
do / wait for route selection
do / display 3 generated routes
[1 of 3 routes selected]
'Start run' button pressed
Start Run page
do / wait for user to start run
do / display selected route
'< Back' button pressed

2.2 Product Functions

RoutePlanner has the following main functions:

1. Provide login services before accessing the app's features.
2. Generate round-trip running routes based on user's desired distance.
3. Integrate sheltered pathways during unfavorable weather conditions.
4. Include user-specified landmark in running route if selected.
5. Provide GPS-based, real-time tracking of the user's location on the selected route
6. Save completed runs to a user's history, including details like distance, time, and route run.
7. Access real-time weather data through the 2-Hour Weather Forecast API.

2.3 User Classes and Characteristics

The RoutePlanner app is designed for the general public, welcoming runners of all experience levels, from beginners to seasoned athletes. Users are only required to have a basic understanding of smartphone apps and fitness tracking. However, it is primarily intended for users within Singapore given that the sheltered dataset is specific to Singapore's geographical boundaries.

2.4 Operating Environment

Users who use RoutePlanner, must have a working internet connection to access all the API services, and enable location services to access our features. The GraphHopper and OneMap API must be working well to provide the route generation features and fetching of landmarks respectively.

Product Environment:

- The RoutePlanner app requires a mobile operating system that supports React Native applications. Compatible platforms include iOS 12.0 or above and Android 9.0 (Pie) or above.

Development Environment:

- The frontend of the app requires a modern JavaScript runtime environment to execute React Native code, including support for JavaScript ES6 or later features.
- The back-end server is hosted and managed using NestJS. It requires a compatible cloud provider or server environment capable of running Node.js 16.0 or above.
- Prisma ORM requires support for PostgreSQL and a server environment that can handle SQL-based database operations.

2.5 Design and Implementation Constraints

2.5.1 API Usage Limits

RoutePlanner relies on multiple third-party APIs to provide core features, such as route generation (GraphHopper), location suggestions (HERE API), landmark data (OneMap API), and weather information (data.gov.sg's 2-Hour Weather Forecast API). Each of these APIs has usage caps that limit the number of queries per day. Exceeding these limits could result in service interruptions or degraded functionality, particularly if RoutePlanner has a high volume of active users.

2.5.2 Limited Token Validity

The OneMap API, which provides landmarks within the desired running distance, imposes a time limit on its generated access tokens. These tokens are valid only for a few days, requiring regular renewal to maintain continuous data access. This constraint adds an extra layer of complexity in ensuring that location data of landmarks is always accessible.

2.6 User Documentation

A short tutorial video on how to use the mobile application will be provided to guide the users:
Tutorial Video

For any technical concerns, the development team can be contacted via GitHub.

2.7 Assumptions and Dependencies

2.7.1 Assumptions

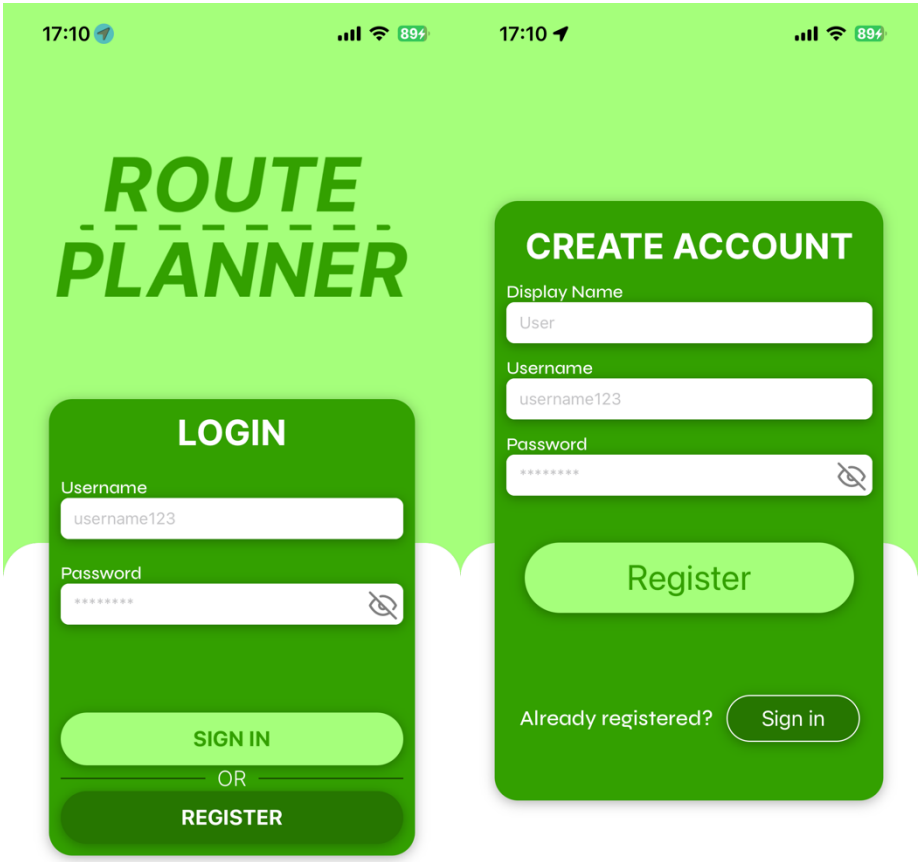
In order to use RoutePlanner, we assume that users have a stable Internet connection to make use of the APIs involved in our application. Additionally, the user's device must have a working Global Positioning System (GPS) to allow RoutePlanner to track their location during their run.

2.7.2 Dependencies

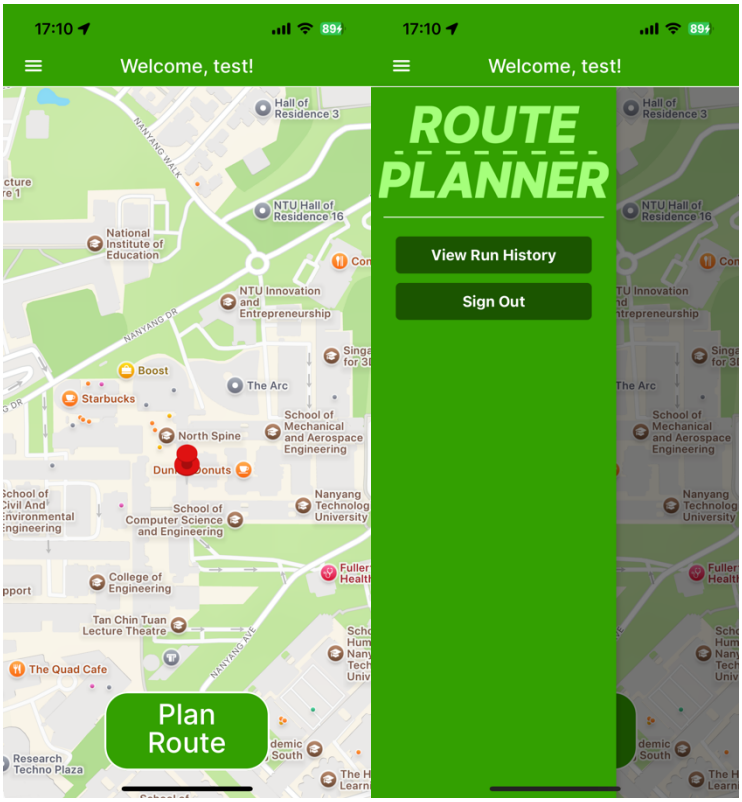
RoutePlanner uses the OneMap API to retrieve landmarks within the radius of the desired running distance specified by the user. GraphHopper API is used to generate the running routes while Google Maps API is used to fetch the geolocation of the user's starting address and the landmark selected. The Supabase API is used to retrieve the coordinates of the sheltered pathways stored in our database. Without these APIs, users are unable to make use of our app's core functionalities.

3. External Interface Requirements

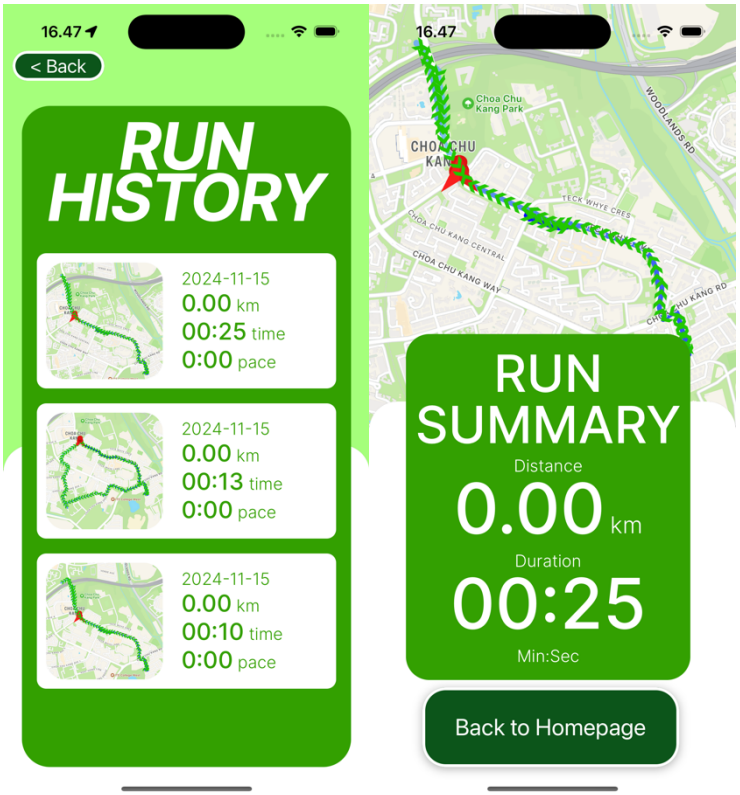
3.1 User Interfaces



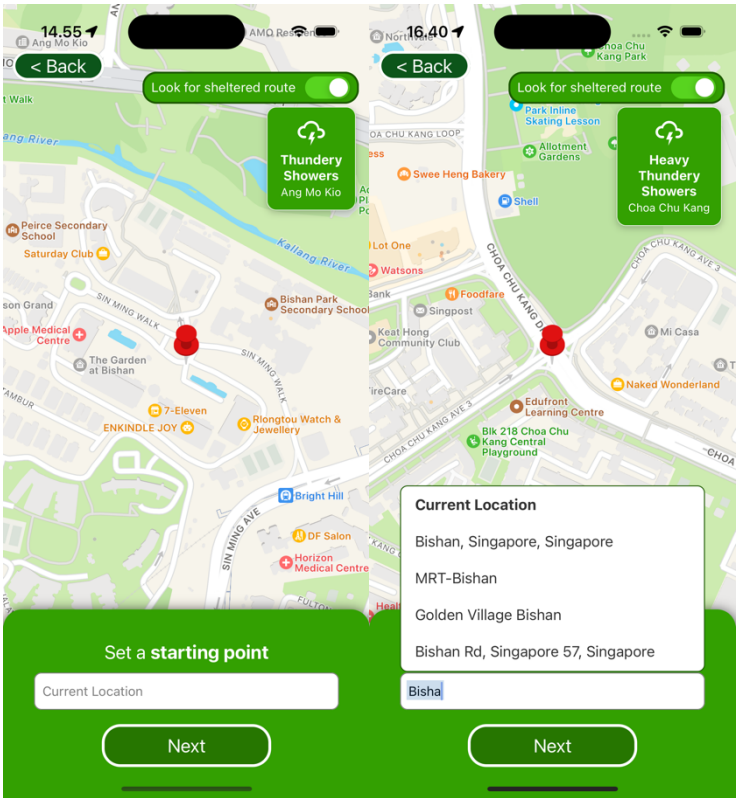
Login (Left) & Register (Right)



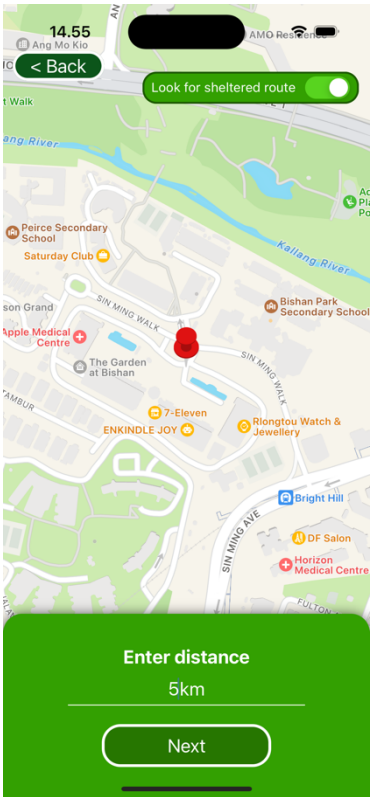
Homepage (Left) & Homepage Sidebar (Right)



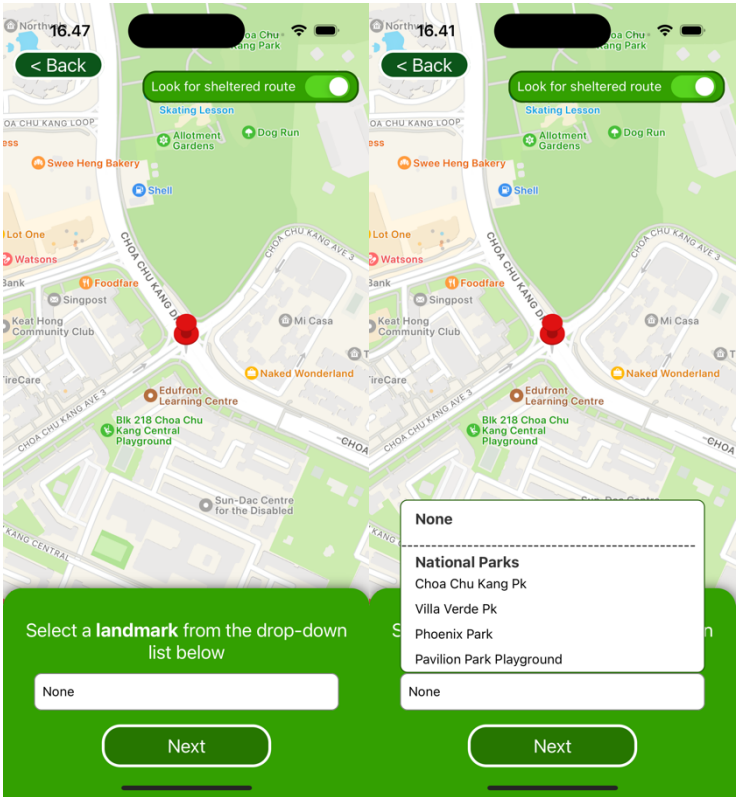
Run History (Left) & Specific Run Summary (Right)



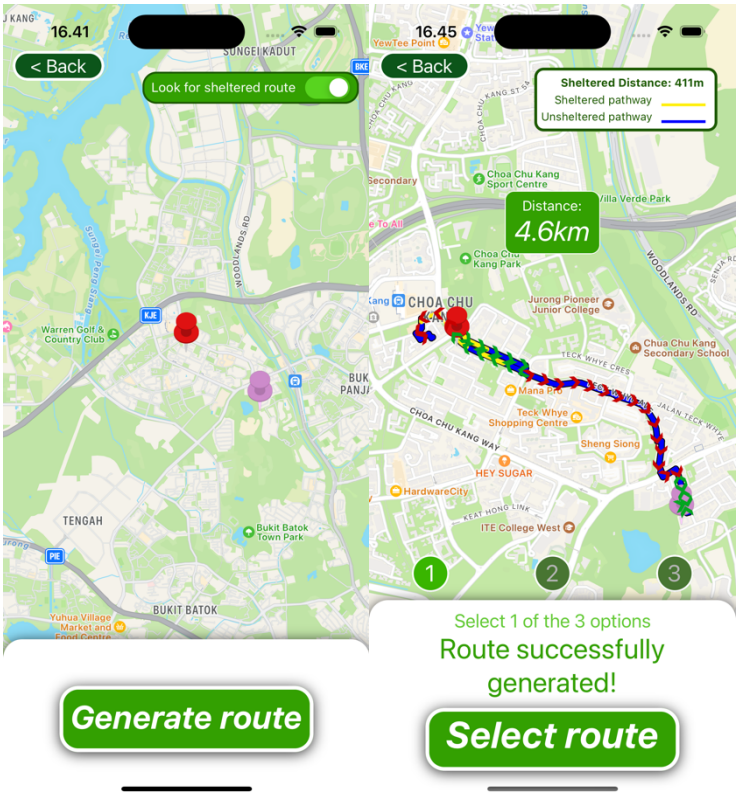
Select Start Point



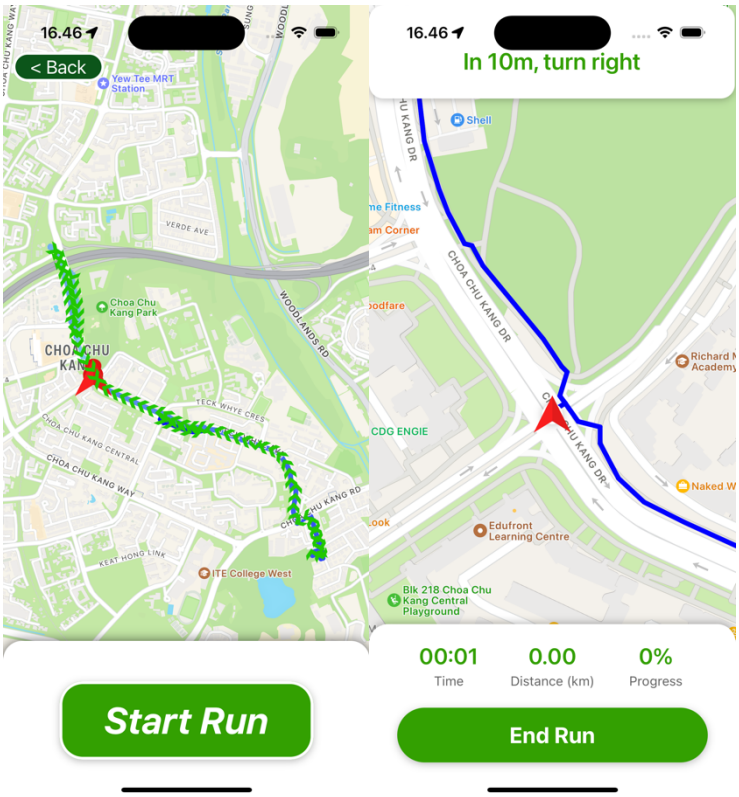
Select Distance



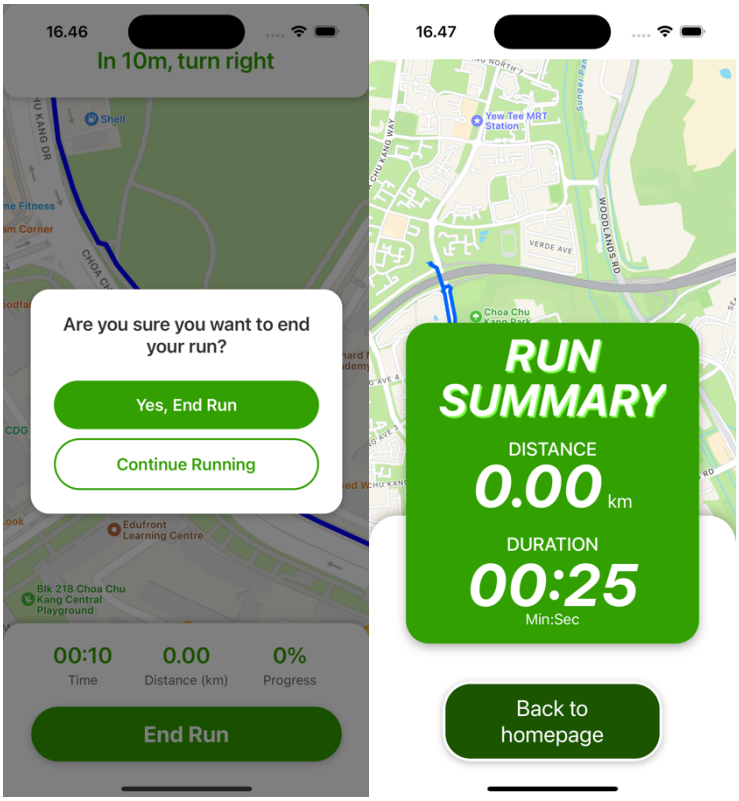
Select Landmark



Generate Route (Left) & Route Selection (Right)



Start Run (Left) & Route Navigation (Right)



End Run (Left) & Post-run Summary (Right)

3.2 Hardware Interfaces

The software product supports both iOS and Android smartphones with touch screens, GPS module and internet connectivity.

3.2.1. The user interacts with the app via touch gestures on the display screen
3.2.2. The software product periodically sends a request to the GPS module, and the GPS module sends data of the user's current location
3.2.3. The app controls the Wi-Fi module by sending instructions to interact with the backend or external APIs

3.3 Software Interfaces

3.3.1. Frontend
  3.3.1.1. React Native (Version 0.74.5)
  3.3.1.2. JavaScript (Version 22.10.0)
3.3.2. Backend
  3.3.2.1. NestJS (Version 10.4.7)
  3.3.2.2. TypeScript (Version 5.1.3)
  3.3.2.3. Supabase (Version 2.46.1)
  3.3.2.4. PostGreSQL (Version 15.6)
  3.3.2.5. Prisma (Version 5.22.0)
3.3.3. APIs
  3.3.3.1. GraphHopper API
  3.3.3.2. OneMap API
  3.3.3.3. Here API

The mobile application's user interface is built using React Native, which uses JavaScript to render components and connect with backend services.

The backend is built with NestJS, as the framework, using TypeScript. NestJS handles all the application logic ranging from user authentication via JWT Token to endpoint routing for HTTP requests. We use Supabase as the database since it provides easy access to S3 bucket which can store the running route images generate. PostgreSQL also serves as a database and enables us to store and process geo-spatial data from the sheltered paths dataset sourced from the LTA static dataset repository. Prisma is used as an Object-Relational Mapping tool that helps to simplify SQL queries and database interactions.

The system relies on APIs, including GraphHopper API for route optimization, OneMap API for retrieving nearby landmarks, and Here API for auto-suggestion of locations based on what the user inputs.

Incoming data includes user inputs (e.g., touch gestures, location queries), GPS coordinates, and API responses (e.g., optimized routes, landmarks), while outgoing messages involve location-based requests, user preferences, and route data. These interactions require a stable internet connection, enabling secure data exchange via HTTPS. Shared data, such as sheltered walkway information, flows between the frontend and backend using JSON, ensuring compatibility and simplicity. Implementation constraints include real-time responsiveness and secure communication, adhering to API protocols and data management standards specific to each API.

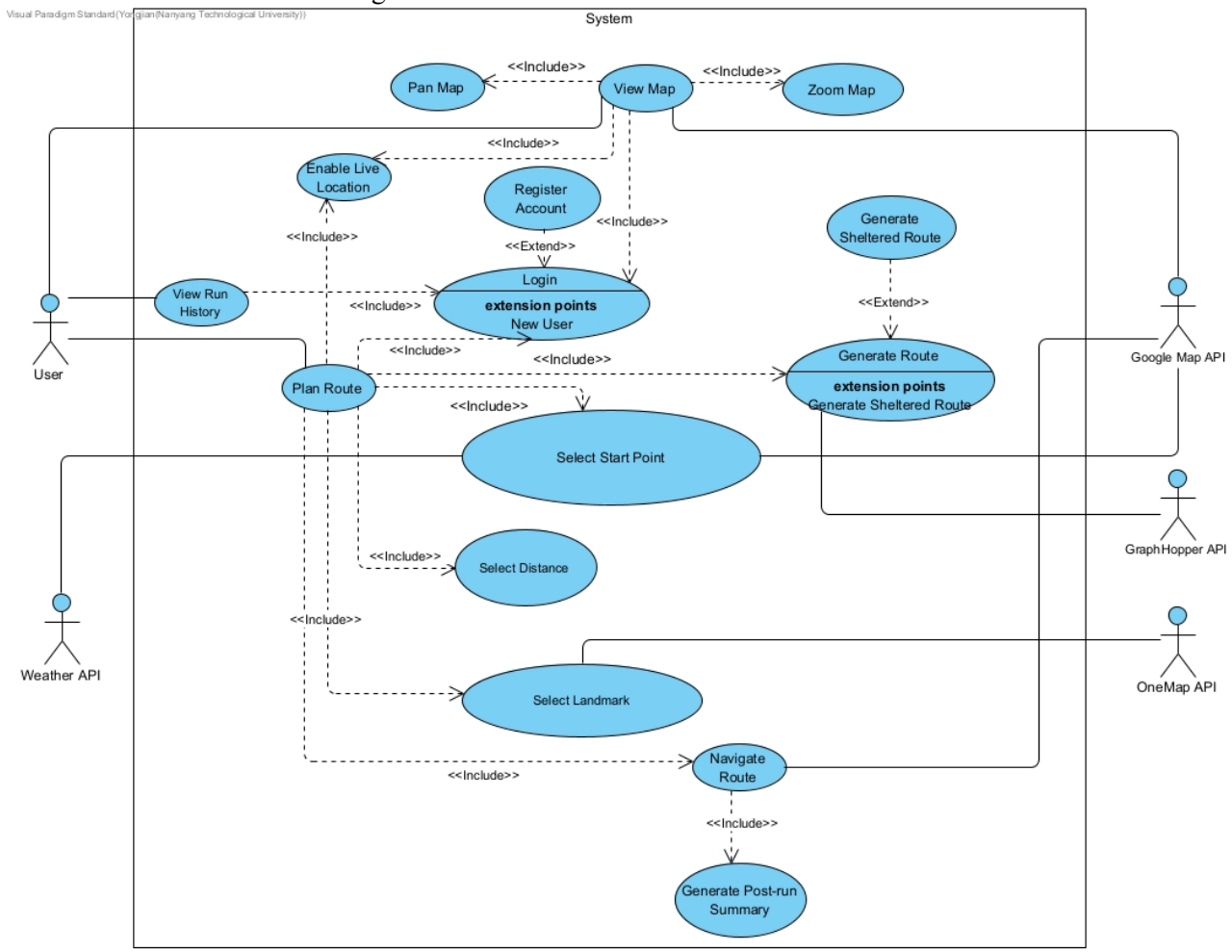
3.4 Communications Interfaces

The product requires seamless communication across multiple components to ensure functionality and user experience. Communication occurs over secure HTTP/HTTPS protocols for client-server interactions, leveraging REST API standards for message formatting, including JSON for data exchange. The frontend (React Native) interacts with the backend (NestJS) and external APIs like GraphHopper, OneMap, and Here API for route generation and location data, using internet connectivity. Communication security is maintained via TLS encryption, and sensitive data, such as user location or authentication tokens, is securely transmitted.

The backend integrates with PostgreSQL for database interactions using Prisma as the ORM, ensuring database queries are properly synchronized. Data transfer rates are optimized based on API and user interactions, ensuring minimal latency for map rendering and route calculations. Synchronization between data retrieval and frontend display is handled using asynchronous communication mechanisms (e.g., Promises in JavaScript), ensuring efficient and responsive user interactions.

4. System Features

Shown below is our use case diagram for Route Planner:

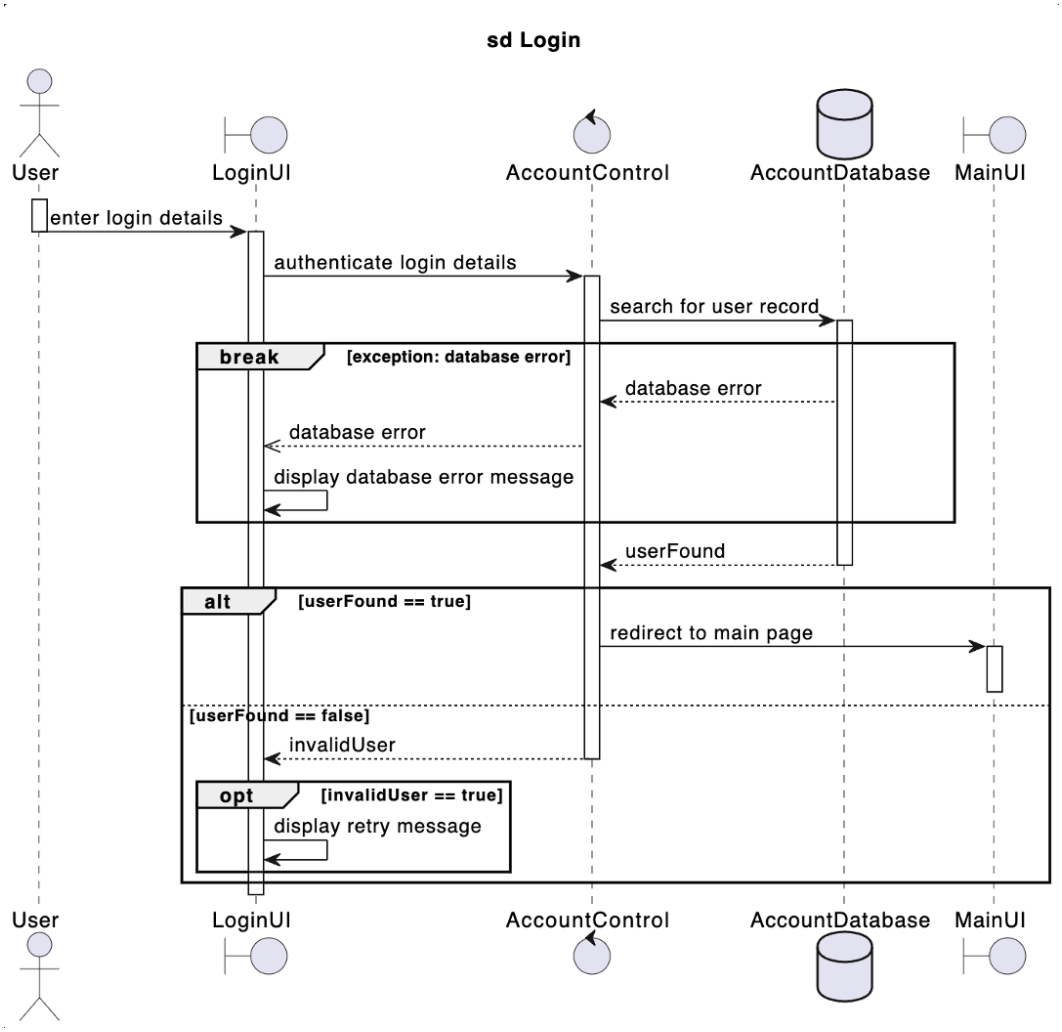


4.1 Login

4.1.1 Description and Priority

This use case allows the user to login into an existing account. This feature has high priority.

4.1.2 Stimulus/Response Sequences



4.1.3 Functional Requirements

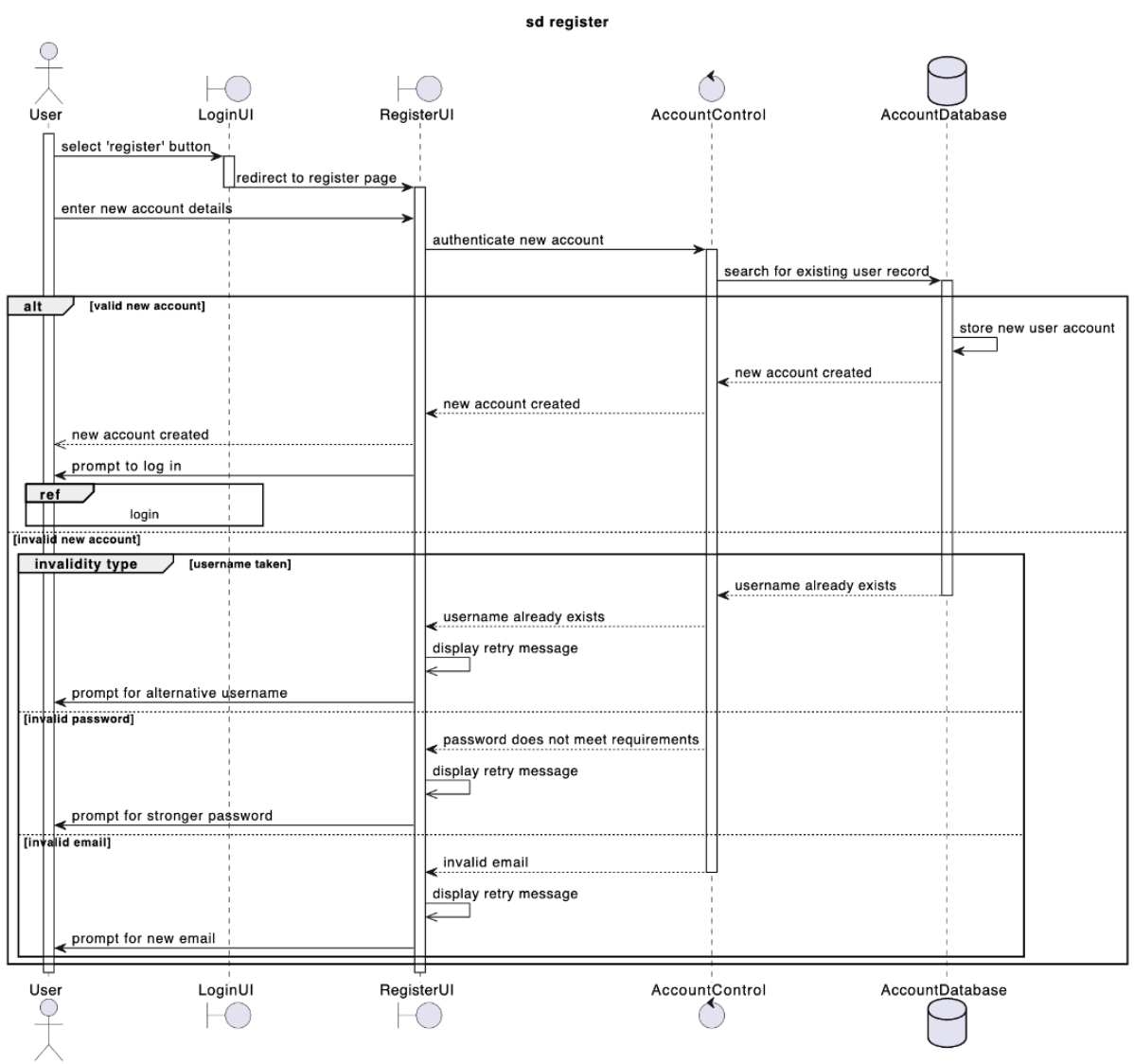
REQ-1: The app must allow an existing user to log in
REQ-2: The login must prompt the user for their login details: username and password
REQ-3: The login must be able to verify login credentials with a database
REQ-3.1: If the login credentials are valid, the app must log in the user
REQ-3.2: If the username or password are incorrect, the app must inform the user and reject the login request

4.2 Register

4.2.1 Description and Priority

This use case allows new users to register for a new account. This use case has high priority

4.2.2 Stimulus/Response Sequences



4.2.3 Functional Requirements

REQ-1: The app must allow new users to register an account
REQ-2: The registration page should take in the following information: display name, username, password

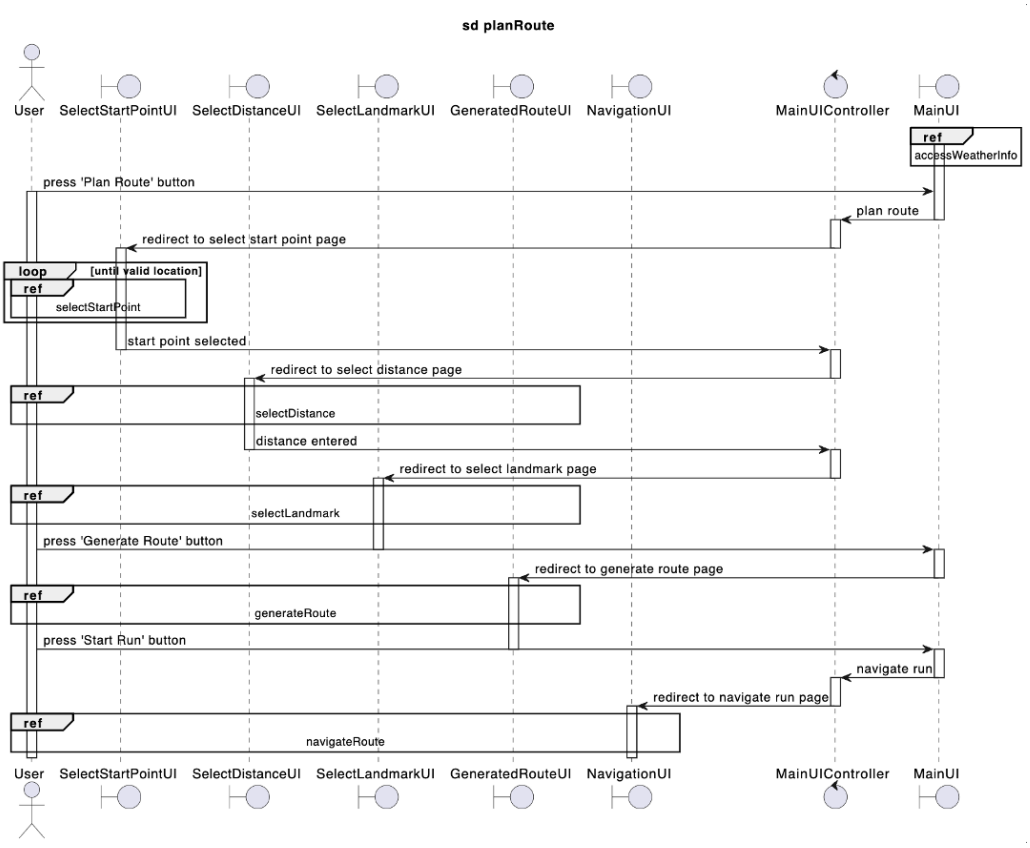
REQ-2.1: If the username is already in use, the app must prompt the user for a new username
REQ-2.2.: The password must be at least 8 characters long
REQ-2.3: If the password is invalid, the app must prompt the user for a new password
REQ-3: The app must add the new account into the database after valid registration

4.3 Plan Route

4.3.1 Description and Priority

This use case allows the user to plan a new running route. This use case has high priority.

4.3.2 Stimulus/Response Sequences



4.3.3 Functional Requirements

REQ-1: The app must prompt the user for the starting point of the run via use case "Select Start Point"
REQ-2: The app must prompt the user for the desired distance of the run via use case "Select Distance"

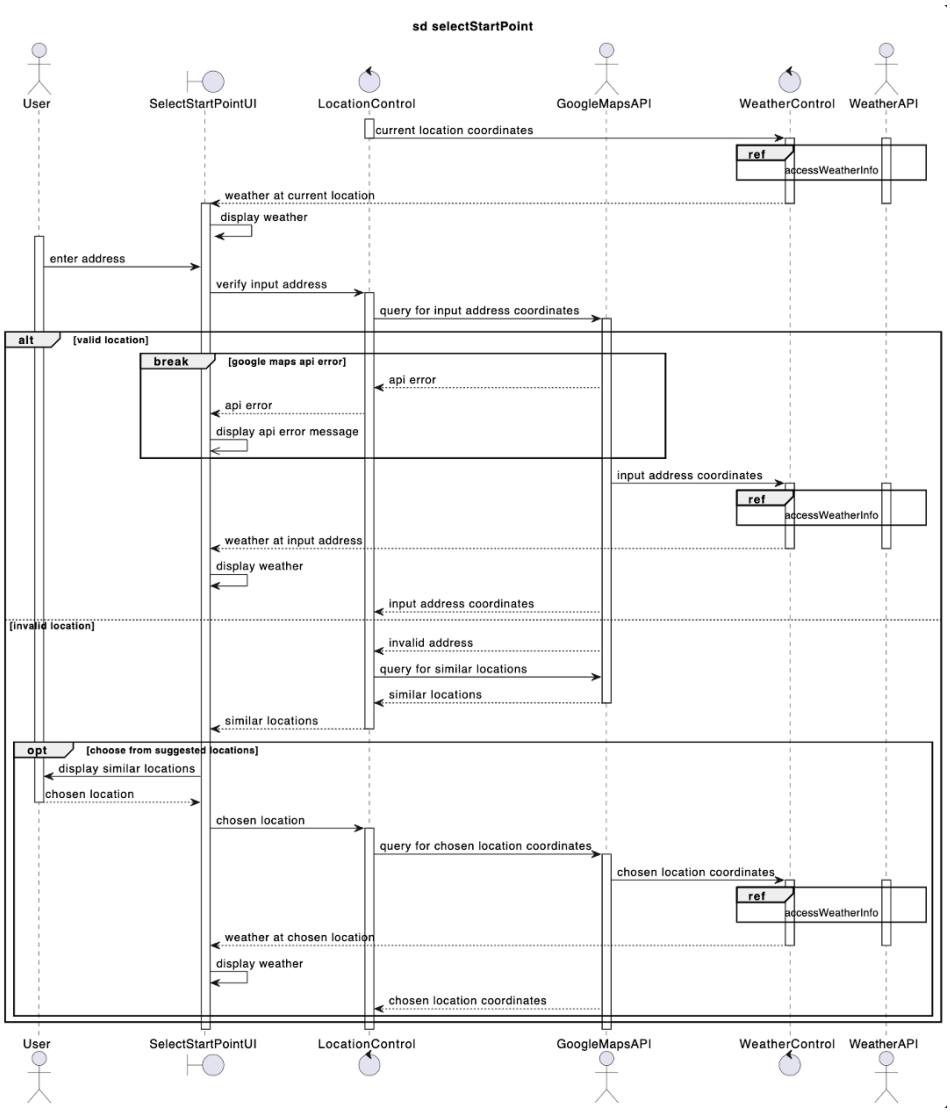
REQ-3: The app must allow the user to select any nearby landmarks that they might want to pass by in the running route via use case “Select Landmark”
REQ-4: The app must generate three routes for the user based on REQ-1, REQ-2 and REQ-3
REQ-5: The app must provide real-time navigation during the run via the use case “Navigate Route”

4.4 Select Start Point

4.4.1 Description and Priority

This use case allows the user to select a starting point for their running route. This use case has a high priority.

4.4.2 Stimulus/Response Sequences



4.4.3 Functional Requirements

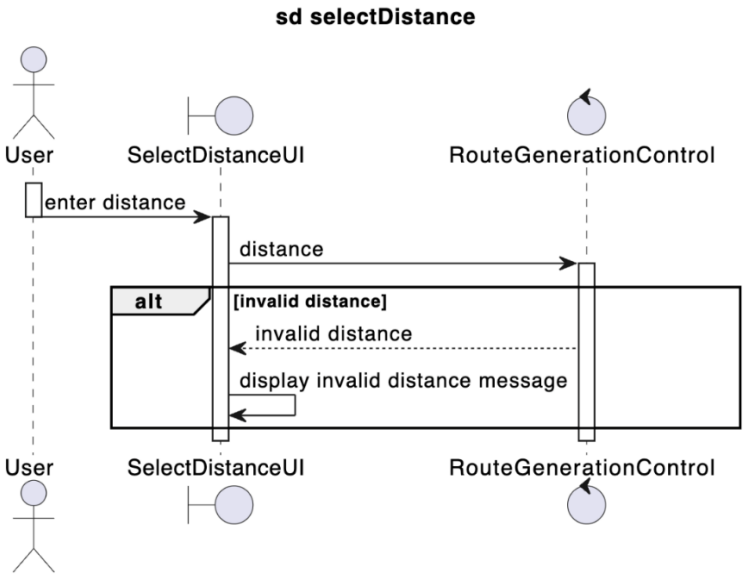
REQ-1: Users must be able to select their current location as the starting point
REQ-2: Users must be able to specify a starting location by inputting an address
REQ-2.1: The app must display auto-suggested locations in a drop-down list to the user while the user inputs an address
REQ-3: The app must display the weather at the selected location via use case "Access Weather Information"
REQ-4: If there is rainy weather at the selected location, the toggle to generate shelter-optimised routes will be turned on by default
REQ-5: Users must be able to toggle the option to generate shelter-optimised routes

4.5 Select Distance

4.5.1 Description and Priority

This use case allows the user to select the desired distance for the running route. This use case has a high priority.

4.5.2 Stimulus/Response Sequences



4.5.3 Functional Requirements

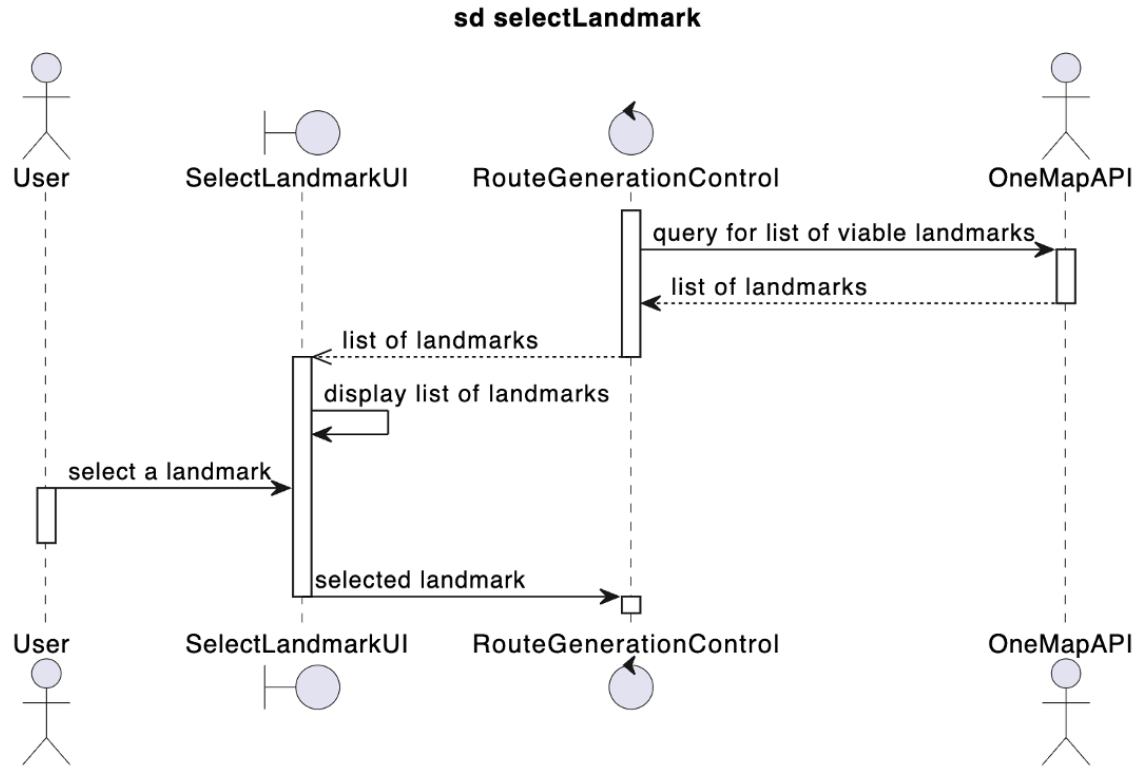
REQ-1: Users must be able to input the desired distance for their running route
REQ-2: The desired distance must be between 1km-30km
REQ-2.1: If the desired distance is out of this boundary, the app will prompt the user to enter a new valid desired distance
REQ-2.3: If a non-numeric value is input, the app will prompt the user to enter a new valid desired distance
REQ-3: Users must be able to toggle the option to generate shelter-optimised routes

4.6 Select Landmark

4.6.1 Description and Priority

This use case allows the user to select a landmark to pass by during their run. This use case has a high priority.

4.6.2 Stimulus/Response Sequences



4.6.3 Functional Requirements

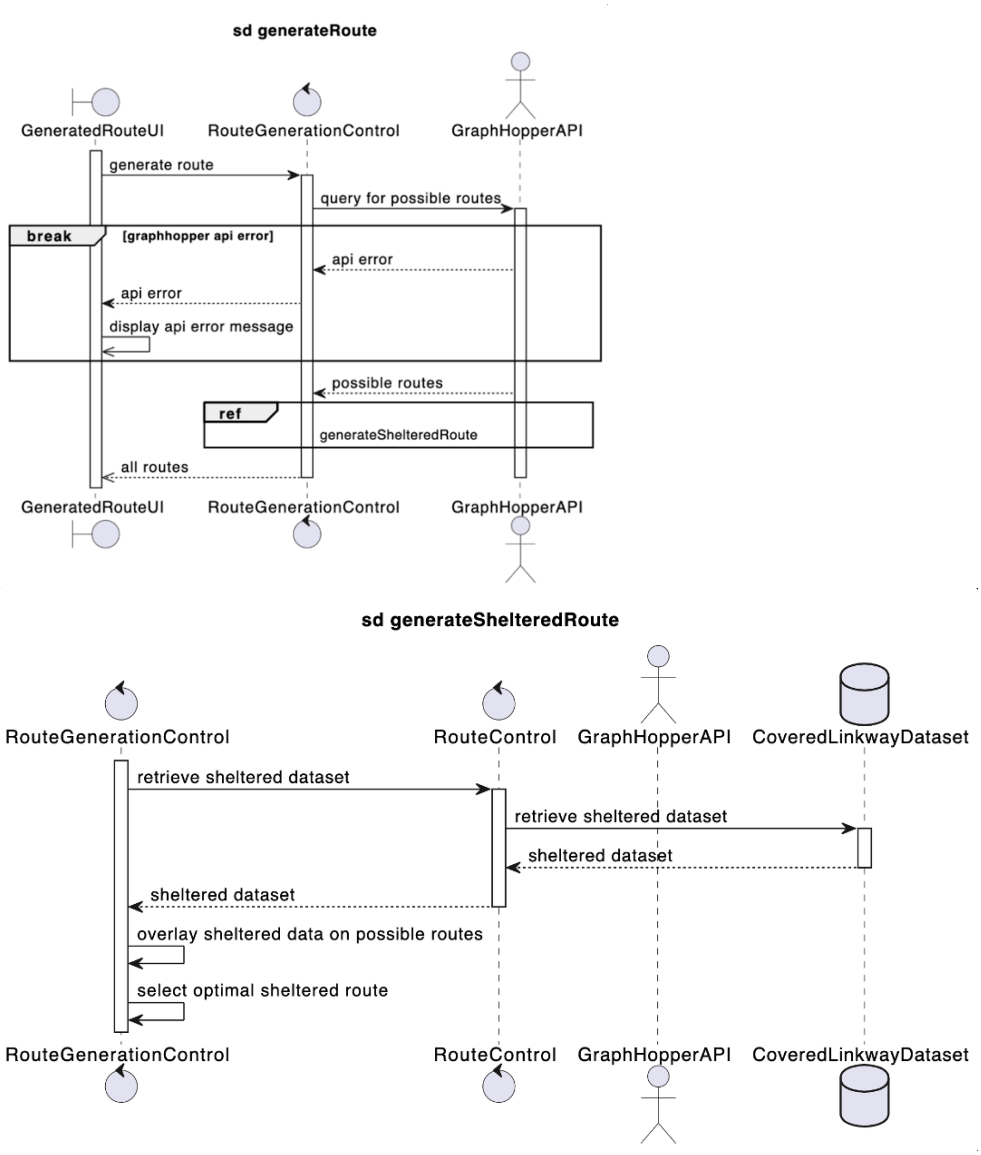
REQ-1: The app must generate a drop-down list of nearby landmarks according to the user's selected starting point and desired distance
REQ-2: User must be able to choose not to run by a landmark
REQ-3: If landmarks are available in the search radius from REQ-1, the user must be able to choose to pass by a landmark
REQ-4: Users must be able to toggle the option to generate shelter-optimised routes

4.7 Generate Route

4.7.1 Description and Priority

This use case allows the user to select a route from the routes generated by the application. This use case has high priority.

4.7.2 Stimulus/Response Sequences



4.7.3 Functional Requirements

REQ-1: The generated routes must start from the selected start point
REQ-2: The generated routes must end at the given starting point
REQ-3: The generated routes must be within 0.5km of the desired distance

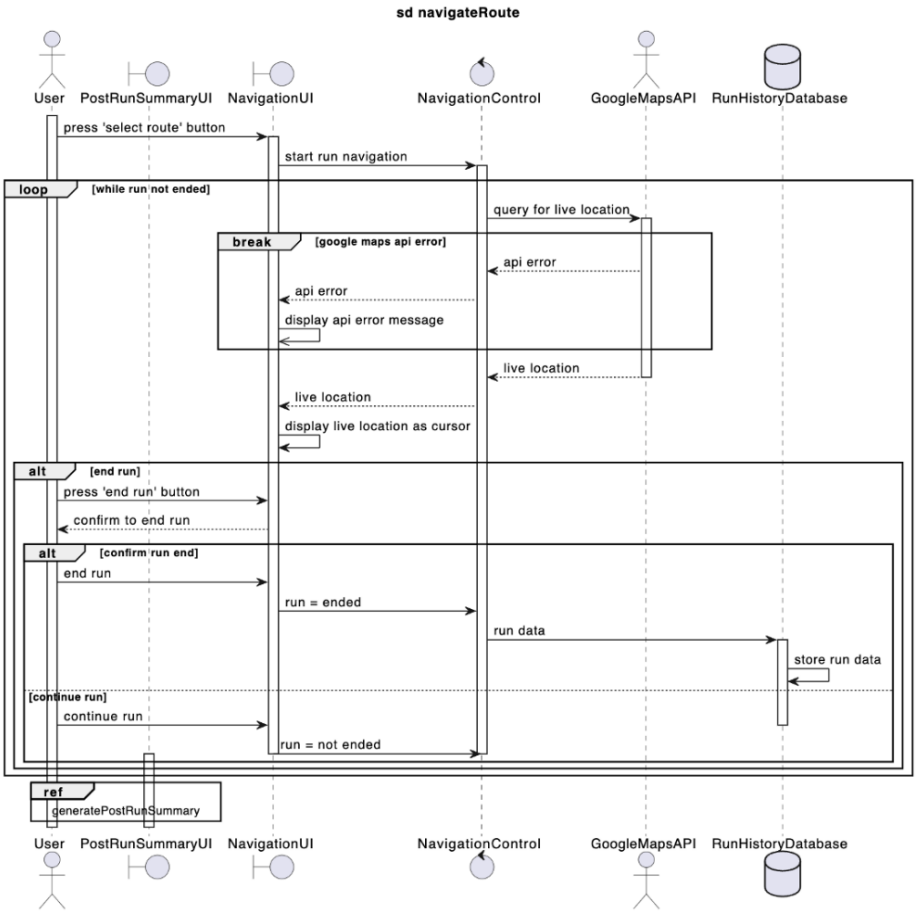
REQ-4: If the user specified a landmark, the generated route must pass by the landmark specified by the user
REQ-5: If the generate shelter-optimised route option was toggled on at any point during the use case “Plan Route”, the use case “Generate Sheltered Route” will be given a higher priority
REQ-6: Three routes fulfilling REQ-1, REQ-2, REQ-3, REQ-4 must be displayed to the user
REQ-7: The sheltered sections of the generated routes must be highlighted in yellow to the user.
REQ-8: The total sheltered distance in the generated routes must be shown to the user

4.8 Navigate Route

4.8.1 Description and Priority

This use case provides live navigation for the user throughout the duration of their run. This use case has high priority.

4.8.2 Stimulus/Response Sequences



4.8.3 Functional Requirements

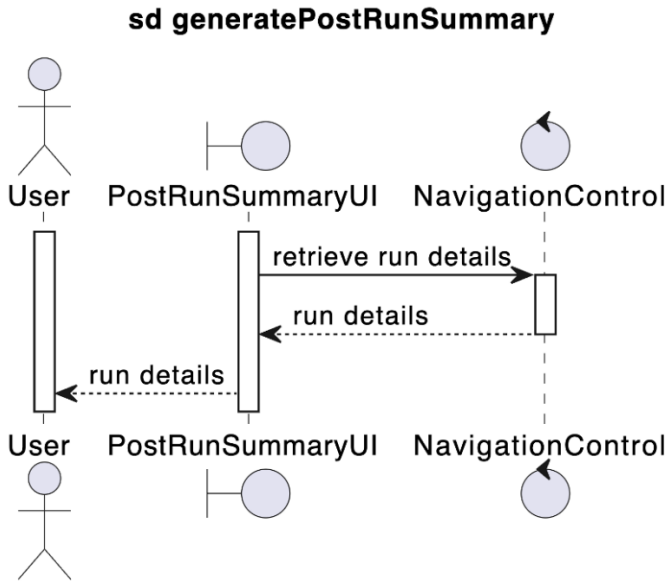
REQ-1: Users must be able to see their distance covered (in km) on the app while running
REQ-2: The app must continue to show the user's live location and updated path during the run
REQ-3: The app must display navigation instructions to the user in text
REQ-4: The app must automatically end the run once the user reaches the ending point
REQ-5: Users must be able to end the run prematurely
REQ-6: If the user strays too far from the generated route, the app must prompt the user to return to the path
REQ-7: Users must be able to see their run summary after their run is ended via the use case "Generate Post Run Summary"

4.9 Generate Post Run Summary

4.9.1 Description and Priority

This use case shows the user their run summary after completing a run. This use case has medium priority.

4.9.2 Stimulus/Response Sequences



4.9.3 Functional Requirements

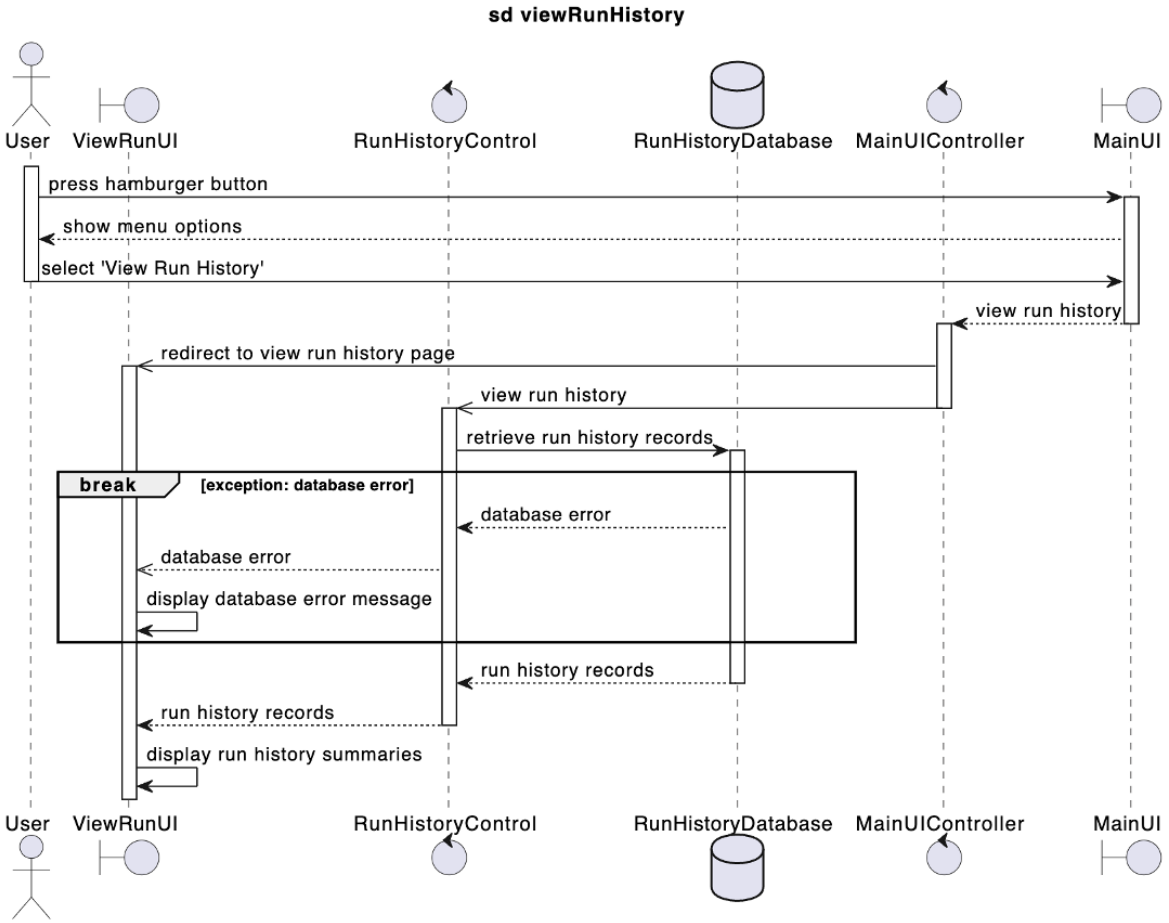
REQ-1: Users must be able to see the total time taken for the run in MM:SS format
REQ-2: Users must be able to see the total distance covered in km

4.10 View Run History

4.10.1 Description and Priority

This use case allows the user to view details for all their completed runs. This use case has medium priority.

4.10.2 Stimulus/Response Sequences



4.10.3 Functional Requirements

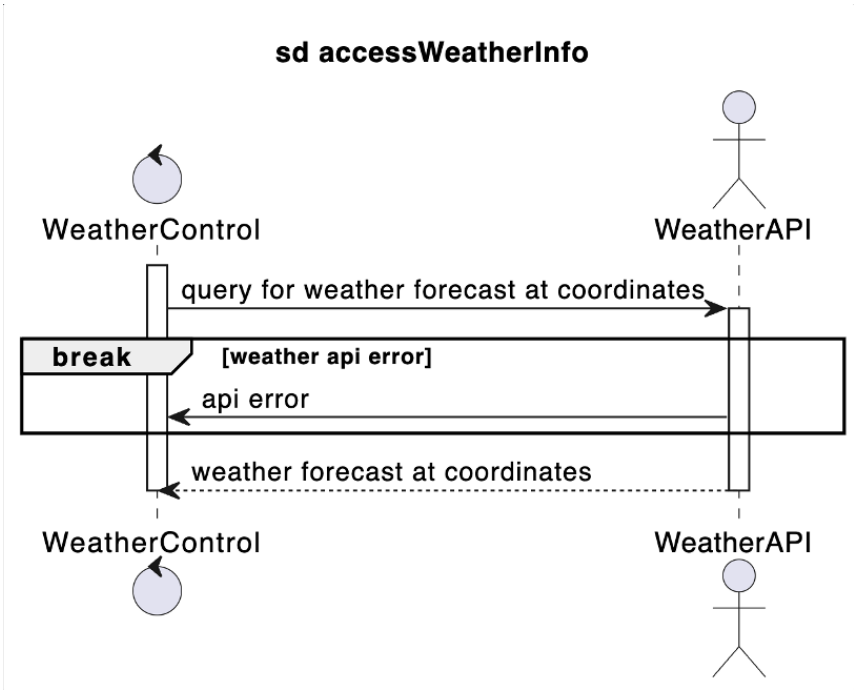
REQ-1: The app must display all the user's completed runs in a scrollable list
REQ-2: The displayed details for each run includes: image of the running route, total distance, total time taken, pace
REQ-3: Clicking on a specific run will expand the details in a separate page

4.11 Access Weather Information

4.11.1 Description and Priority

This use case retrieves weather information at a location from the Weather API to be displayed on the application. This use case has medium priority.

4.11.2 Stimulus/Response Sequences



4.11.3 Functional Requirements

REQ-1: The app must be able to access weather information based on location coordinates

5. Other Nonfunctional Requirements

5.1 Performance Requirements

5.1.1. The application must load each page within 3 seconds
5.1.2. The application must respond to user actions within 2 seconds
    5.1.2.1. The map responds to zoom in and out actions within 1 second
5.1.3. The application must generate possible routes within 20 seconds

5.2 Safety Requirements

5.2.1. The application must track the user's current location with an accuracy of at least 10m
5.2.2. The application must not generate routes that pass through unsafe or restricted zones
5.2.3. The application must inform the users to return to the route if they are 50m away from the planned route
5.2.4. The application must provide the users with running instructions to avoid users entering restricted or hazardous areas

5.3 Security Requirements

5.3.1. The application must clearly inform users of the reasons for collecting location data
5.3.2. The application must obtain user content to collect location data
5.3.3. Users must be allowed to sign out of their account at any time

5.4 Software Quality Attributes

5.4.1. Usability

5.4.1.1. More than 95% of users should be able to complete the route selection process within 2 minutes

5.4.2. Compatibility

5.4.2.1. Map view must be clearly viewed and compatible on a variety of mobile device models and screen sizes

5.4.3. Accessibility

5.4.3.1. The map interface must have appropriate labels
5.4.3.2. All font sizes used in the application must be of at least size 10

5.4.4. Accuracy

5.4.4.1. The generated routes must run through only accessible paths
5.4.4.2. The user's displayed live location must be within 10m of their actual location
5.4.4.3. For run navigation, upcoming changes in direction must be reported to the user when the user is 50m away from the change

5.4.5. Support

5.4.5.1. Proper documentation must be provided for future developers

Appendix A: Glossary

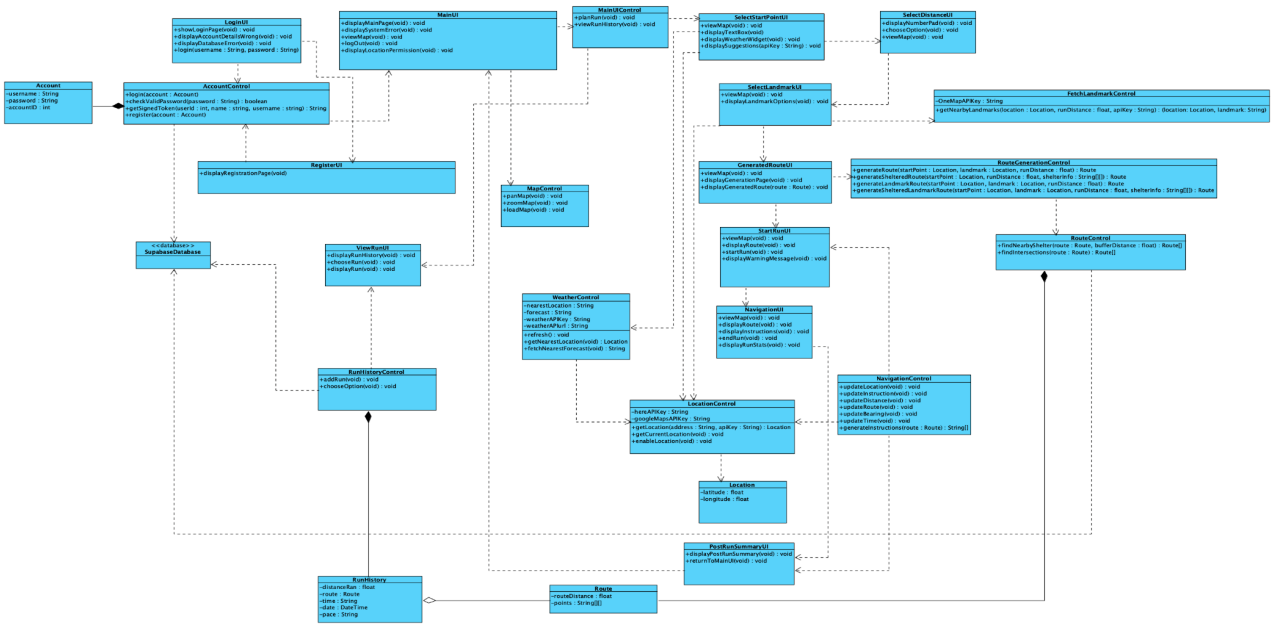
Data Dictionary

Term	Definition
User	The person using the application
Desired distance	The distance that the user wishes to run
Route	The running path the user can follow that is tailored to the user's desired distance, starting point, and optional landmark
Login services	The component of the application that allows users to create or sign in to their accounts in order to access the route generation features
Homepage	The page of the application displaying the user's current location on a map, the option to begin planning their route and a menu to direct them to Sign out or View Run History
Weather data	Real-time information regarding the weather at the current time and location the user is accessing the application, along with the weather forecast of the next 2 hours in the region the user is located in
Sheltered pathway	A path the user can run along that provides cover during unfavourable weather conditions
Starting point	The position at which the user begins their run
Landmark	A site or location the user can specify to pass by during their run
Tracker page	The page of the application displays to the user how far along the route they currently are on a map, the distance they have covered thus far and the time elapsed. This page would get updated periodically as the user runs along the route so as to reflect their running progress in real-time
Running progress	Real-time information indicating on the map how much of the route the user has covered up to that point in time
Running statistics	Data consisting of the user's running duration and distance
Running Duration	A unit of data indicating the amount of time the user has spent running in MM:SS

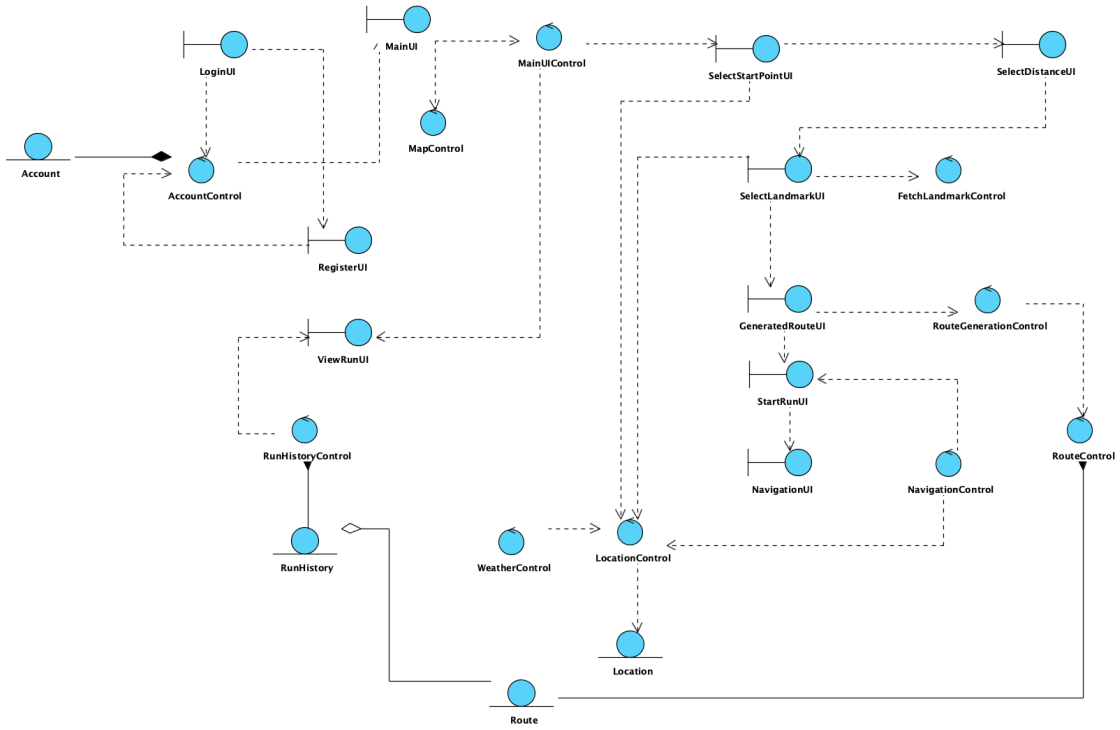
Post-run summary	Information consisting of the user's Running Duration, distance covered and a screenshot of the route the user has taken
Route Navigation	A set of directional instructions, displayed one by one, guiding the user from a starting point to an endpoint
Run History	A page of the app that displays the run summaries of all the user’s previous runs

Appendix B: Analysis Models

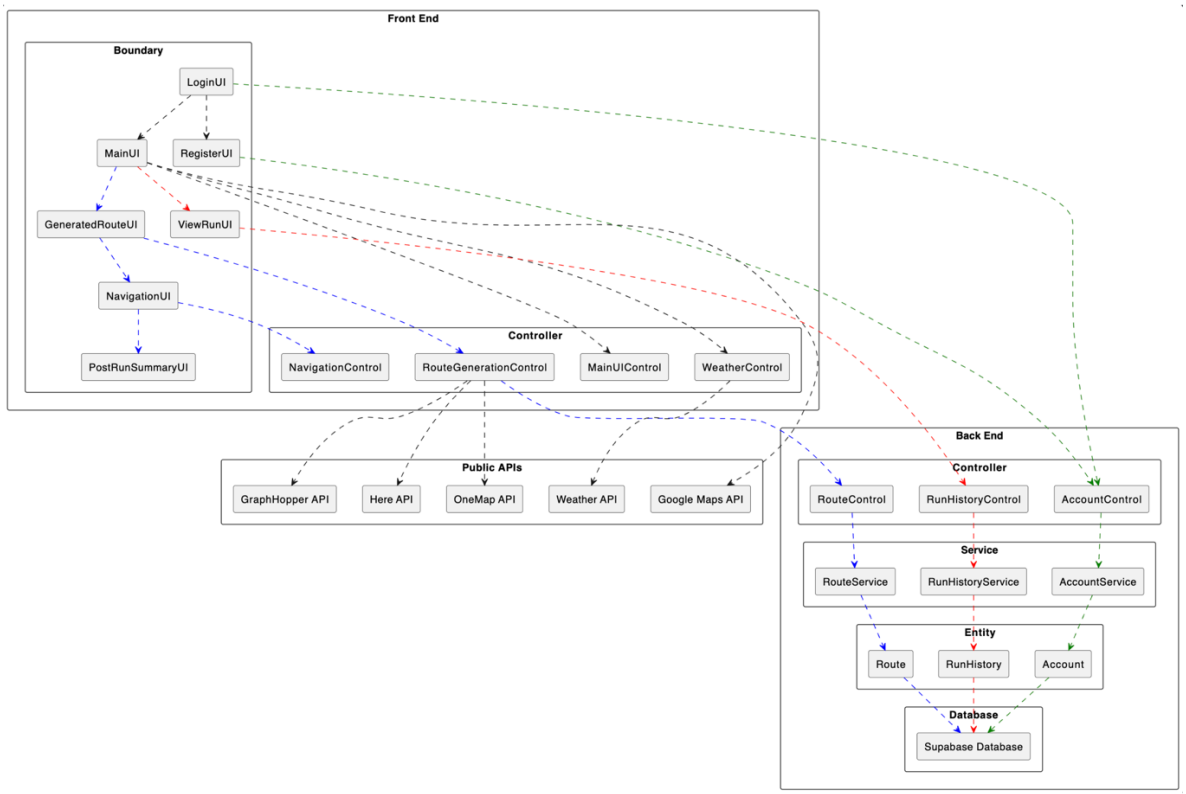
Class Diagram



Stereotype Diagram



System Architecture



Black Box Testing (Equivalence Class & Boundary Value Testing)

Use case 1: Login Account
Equivalence Class for Username:

Valid EC	Existing Username
Invalid EC	Empty field or non-existing Username

Equivalence Class for Password length:

Valid EC	Correct password
Invalid EC	Empty field or wrong password

Test Case 1	Invalid Credential		
Test Scenario	Login Account	**Test Case ID**	Login-1
Test Description	Login with invalid credential (Assume that correct username is "Test123" and password is "qwertyui")	**Test Priority**	Medium
Prerequisite	Stable Internet Connection	**Post-Requisite**	NA

No.	Username	Password	Expected Output	Actual Output	Test Result
1	Empty field	qwertyui	Displays error message: Username must not be empty	Displays error message: Username must not be empty	Pass
2.	Test12	qwertyui	Displays error message: Invalid Credential	Displays error message: Invalid Credential	Pass
3.	Test123	Empty field	Displays error message: Invalid Credential	Displays error message: Invalid Credential	Pass
4.	Test123	qwe	Displays error message: Invalid Credential	Displays error message: Invalid Credential	Pass

Test Case 2	Login successfully		
Test Scenario	Login Account	**Test Case ID**	Login-2
Test Description	Login with valid credential (Assume that correct username is "Test123" and password is "qwertyui")	**Test Priority**	Medium
Prerequisite	Stable Internet Connection	**Post-Requisite**	NA

No.	Username	Password	Expected Output	Actual Output	Test Result
1	Test123	qwertyui	Displays success message: Welcome back Test123	Displays success message: Welcome back Test123	Pass

Use case 2: Plan Route – Select Distance
Equivalence Class for Distance:

Valid EC	1 <= Distance <= 30
Invalid EC	Distance < 1 or Distance > 30

Boundary Values:

Valid EC	**Value on boundary values**	1, 30
	Lower boundary	0, 1, 2
	Upper boundary	29, 30, 31

Final input values for Distance:

- Valid: {1, 30}
- Invalid: {0, 31}

Test Case 1	Invalid Distance		
Test Scenario	Plan Route – Select Distance	**Test Case ID**	Select Distance -1
Test Description	User selects invalid distance	**Test Priority**	Medium
Prerequisite	Stable Internet Connection User is authenticated	**Post-Requisite**	NA

No.	Distance	Expected Output	Actual Output	Test Result
1	0	Displays error message: Run distance must be between 1 km and 30 km	Displays error message: Run distance must be between 1 km and 30 km	Pass
2	31	Displays error message: Run distance must be between 1 km and 30 km	Displays error message: Run distance must be between 1 km and 30 km	Pass

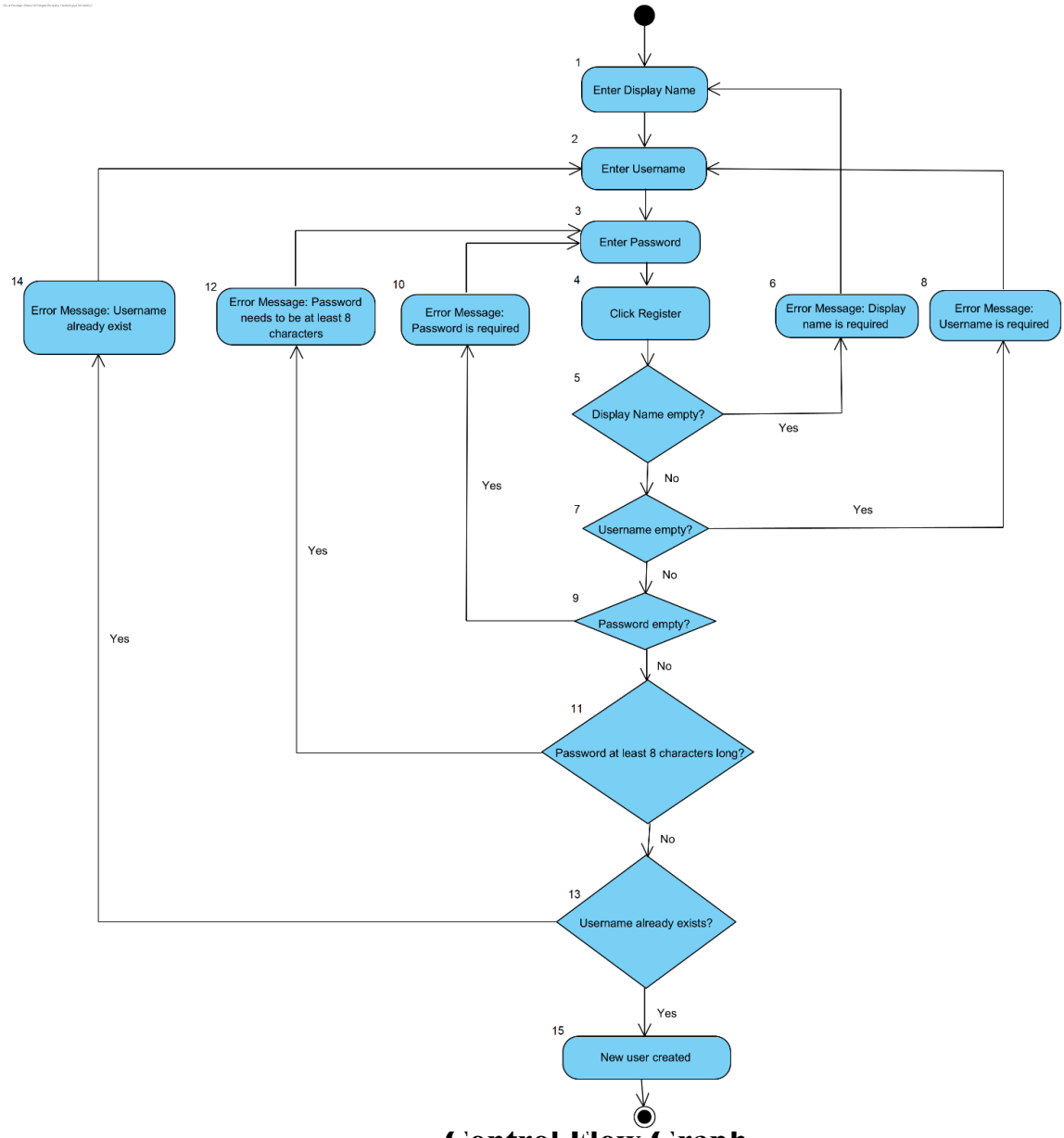
Test Case 2	Valid Distance		
Test Scenario	Plan Route – Select Distance	**Test Case ID**	Select Distance -2
Test Description	User selects valid distance	**Test Priority**	Medium
Prerequisite	Stable Internet Connection	**Post-Requisite**	NA

	User is authenticated		

No.	Distance	Expected Output	Actual Output	Test Result
1	1	User is able to select landmark	User is able to select landmark	Pass
2	30	User is able to select landmark	User is able to select landmark	Pass

White Box Testing

Use case 1: Register Account



Control Flow Graph

Table of Basis Paths

No.	Basis Path	Path Description
1	1 → 2 → 3 → 4 → 5 → 6	Empty Display Name
2	1 → 2 → 3 → 4 → 5 → 7 → 8	Empty Username
3	1 → 2 → 3 → 4 → 5 → 7 → 9 → 10	Empty Password
4	1 → 2 → 3 → 4 → 5 → 7 → 9 → 11 → 12	Password does not fulfil length requirement
5	1 → 2 → 3 → 4 → 5 → 7 → 9 → 11 → 13 → 14	Username already exists
6	1 → 2 → 3 → 4 → 5 → 7 → 9 → 11 → 13 → 15	User successfully registered

Test Case 1	Empty Display Name		
Test Scenario	Register Account	**Test Case ID**	Register-1
Test Path	1 → 2 → 3 → 4 → 5 → 6	**Test Priority**	Medium
Prerequisite	Stable Internet Connection	**Post-Requisite**	NA

No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	Click Register without entering Name	-	Displays error message: Name must not be empty	Displays error message: Name must not be empty	Pass

Test Case 2	Empty Username		
Test Scenario	Register Account	**Test Case ID**	Register-2
Test Path	1 → 2 → 3 → 4 → 5 → 7 → 8	**Test Priority**	Medium
Prerequisite	Stable Internet Connection	**Post-Requisite**	NA

No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	Click Register without entering Username	Name field is not empty	Displays error message: Username must not be empty	Displays error message: Username must not be empty	Pass

Test Case 3	Empty Password		
Test Scenario	Register Account	**Test Case ID**	Register-3
Test Path	1 → 2 → 3 → 4 → 5 → 7 → 9 → 10	**Test Priority**	Medium
Prerequisite	Stable Internet Connection	**Post-Requisite**	NA

No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	Click Register without entering Password	Name and Username field are not empty	Displays error message: Password must not be empty	Displays error message: Password must not be empty	Pass

Test Case 4	Password does not fulfil length requirement		
Test Scenario	Register Account	**Test Case ID**	Register-4
Test Path	1 → 2 → 3 → 4 → 5 → 7 → 9 → 11 → 12	**Test Priority**	Medium
Prerequisite	Stable Internet Connection	**Post-Requisite**	NA

No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	Click Register without entering Password of at least 8 characters long	Name and Username field are not empty, Password is less than 8 characters long	Displays error message: Password needs to be at least 8 characters long	Displays error message: Password needs to be at least 8 characters long	Pass

Test Case 5	Username already exists		
Test Scenario	Register Account	**Test Case ID**	Register-5
Test Path	1 → 2 → 3 → 4 → 5 → 7 → 9 → 11 → 13 → 14	**Test Priority**	Medium
Prerequisite	Stable Internet Connection	**Post-Requisite**	NA

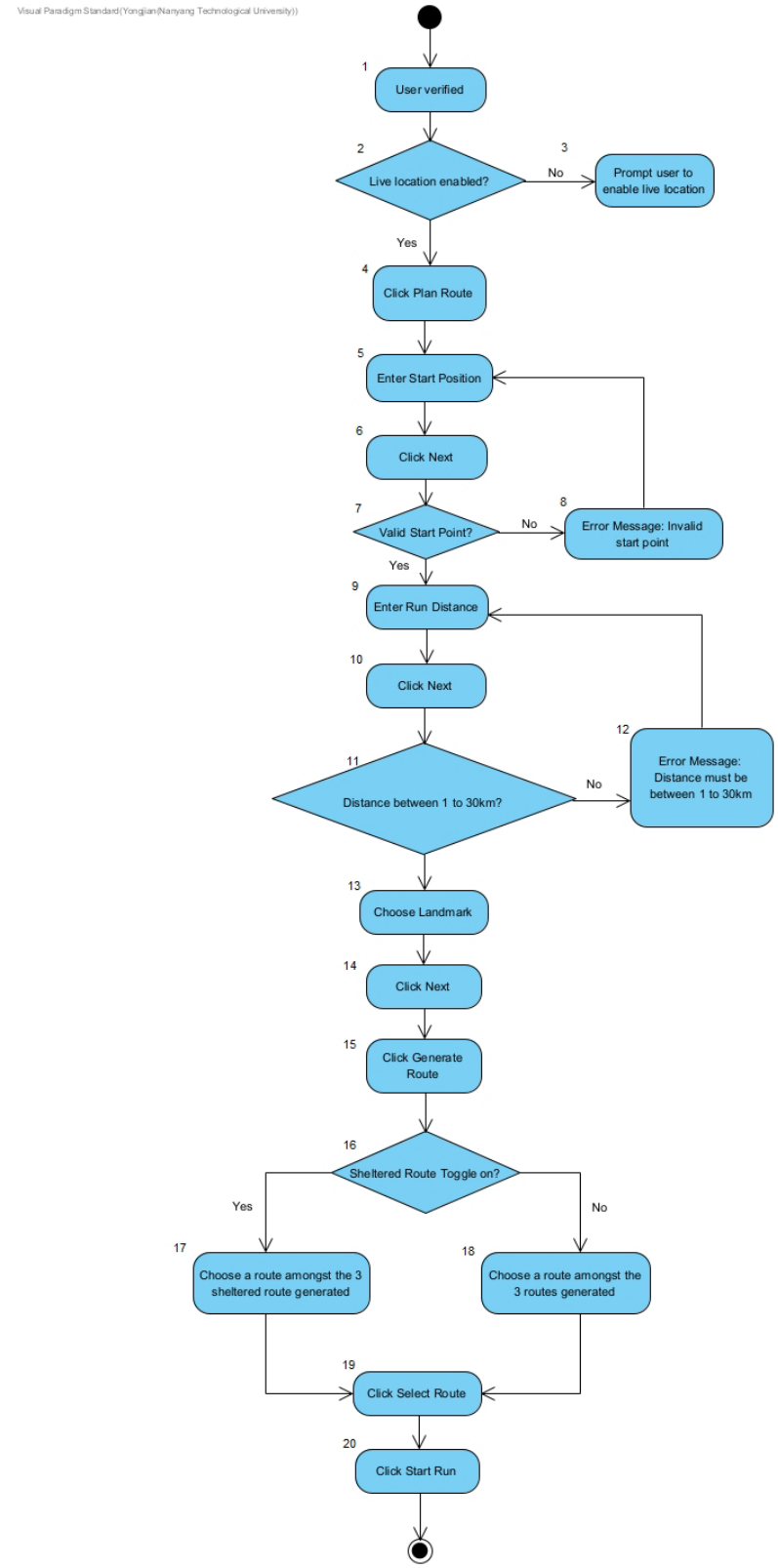
No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	Click Register using a Username that was previously used for account creation	Name field is not empty, Password is at least 8 characters long, Username field already exists	Displays error message: Username already exists	Displays error message: Username already exists	Pass

Test Case 6	User successfully registered		
Test Scenario	Register Account	**Test Case ID**	Register-5
Test Path	1 → 2 → 3 → 4 → 5 → 7 → 9 → 11 → 13 → 15	**Test Priority**	Medium
Prerequisite	Stable Internet Connection	**Post-Requisite**	NA

No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	Click	Name field is	Displays success	Displays success	Pass

	Register using complete and correct user details	not empty, Password is at least 8 characters long, Username field is unique	message: User successfully created	message: User successfully created	

Use case 2: Plan Route



Control Flow Graph

Table of Basis Paths

No.	Basis Path	Path Description
1	1 → 2 → 3	Live location not enabled
2	1 → 2 → 4 → 5 → 6 → 7 → 8	Invalid start point
3	1 → 2 → 4 → 5 → 6 → 7 → 9 → 10 → 11 → 12	Invalid run distance
4	1 → 2 → 4 → 5 → 6 → 7 → 9 → 10 → 11 → 13 → 14 → 15 → 16 → 17 → 19 → 20	Generate sheltered route
5	1 → 2 → 4 → 5 → 6 → 7 → 9 → 10 → 11 → 13 → 14 → 15 → 16 → 18 → 19 → 20	Generate non-sheltered route

Test Case 1	Live location not enabled		
Test Scenario	Plan Route	**Test Case ID**	Plan Route-1
Test Path	1 → 2 → 3	**Test Priority**	Medium
Prerequisite	Stable Internet Connection User is authenticated	**Post-Requisite**	NA

No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	User login to the app without enabling location tracking	-	Displays message: Please enable location permission in setting to use the app	Displays message: Please enable location permission in setting to use the app	Pass

Test Case 2	Invalid start point		
Test Scenario	Plan Route	**Test Case ID**	Plan Route-2
Test Path	1 → 2 → 4 → 5 → 6 → 7 → 8	**Test Priority**	Medium
Prerequisite	Stable Internet Connection User is authenticated	**Post-Requisite**	NA

No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	User enters a starting location that cannot be found	Invalid location	Displays message: Please enter a valid location	Displays message: Please enter a valid location	Pass

Test Case 3	Invalid run distance		
Test Scenario	Plan Route	**Test Case ID**	Plan Route-3

Test Path	1 → 2 → 4 → 5 → 6 → 7 → 9 → 10 → 11 → 12	**Test Priority**	Medium
Prerequisite	Stable Internet Connection User is authenticated	**Post-Requisite**	NA

No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	User enters a run distance shorter than 1km	Run distance: 0.1 km	Displays error message: Run distance must be between 1 km and 30 km	Displays error message: Run distance must be between 1 km and 30 km	Pass
2	User enters a run distance longer than 30km	Run distance: 50 km	Displays error message: Run distance must be between 1 km and 30 km	Displays error message: Run distance must be between 1 km and 30 km	Pass

Test Case 4	Generate sheltered route		
Test Scenario	Plan Route	**Test Case ID**	Plan Route-4
Test Path	1 → 2 → 4 → 5 → 6 → 7 → 9 → 10 → 11 → 13 → 14 → 15 → 16 → 17 → 19 → 20	**Test Priority**	High
Prerequisite	Stable Internet Connection User is authenticated	**Post-Requisite**	NA

No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	User chooses to generate sheltered route	User turns on “Sheltered route” toggle before generating route	Three sheltered routes are displayed for user selection	Three sheltered routes are displayed for user selection	Pass

Test Case 5	Generate non-sheltered route		
Test Scenario	Plan Route	**Test Case ID**	Plan Route-5
Test Path	1 → 2 → 4 → 5 → 6 → 7 → 9 → 10 → 11 → 13 → 14 → 15 → 16 → 18 → 19 → 20	**Test Priority**	High
Prerequisite	Stable Internet Connection	**Post-Requisite**	NA

	User is authenticated		

No.	Action	Inputs	Expected Output	Actual Output	Test Result
1	User chooses to generate non-sheltered route	User turns off "Sheltered route" toggle before generating route	Three non-sheltered routes are displayed for user selection	Three non-sheltered routes are displayed for user selection	Pass