Software Requirements Specification

for

TwoTyred Web Application

Version 1.0 approved

Prepared by Ang Kai Jun, Chang Dao Zheng, Chay Hui Xiang, Ivan Loke Zhi Hao, Ng Li Lin Evonne, Liu Liwen

SC2006 Software Engineering, REP Lab Group, Team Fantastic 4

6 November 2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
All members	6/11/2022	Populate all content	1.0

1. Introduction

1.1 Purpose

This document outlines the software requirements of the web application - TwoTyred. It provides descriptions on the features, interfaces and guidelines of the application. The use cases and functional requirements will also be detailed in this document. Additionally, the SRS document will provide information on the operating environment, implementation and design constraints of TwoTyred. It serves as a framework for all developers and allows the client to review whether the requirements are aligned with their specifications.

1.2 Document Conventions

- Boldface font and indentation are used for headings and sub-headings
- Sections headers are labelled with decimal numbered lists e.g. 3.1.1.8.1 Mobile View: Manual Route Selection
- All main headers (No decimal places) are in bold Arial font, font size 20
- All sub headers (1 decimal place) are in bold Arial font, font size 16
- All other headers (2 or more decimal places) are in bold Arial, font size 14
- All body text are in Arial font, font size 11, justified and 1.5 line spacing
- Diagrams and figures are labelled with captions in italicised Arial font, font size 11 e.g. Figure 1: Overall Product Perspective

1.3 Intended Audience and Reading Suggestions

The intended audience for this SRS will be the grading professor and teaching assistant for the course SC2006 Software Engineering.

The individuals reading this report should have a background in software engineering and have a thorough understanding on the process of developing an application as well as how to interpret the various diagrams that illustrate how the app functions and behaves. Readers may begin from Section 1. Introduction and carry on in numerical order to Section 2. Overall Description etc. For readers who are interested in determining the product's requirement elicitation, they may choose to skip to Section 3 where interactions between software components are discussed in further detail or to Section 4 where specific use cases will be mentioned complete with their diagrams.

1.4 Product Scope

The product implementation is TwoTyred. TwoTyred is a progressive web app that can be viewed through a desktop browser or on a user's mobile web browser. The purpose of TwoTyred is to create a social media platform for cyclists to create and keep a record of their previous cycling routes. Users who use this app can also view other user's created routes and have the option to "like " their route to bring it to the front of our dashboard or "favourite" the route for it to show up in their profile page under a separate column. This app will take advantage of OneMap's API to generate a cycling route based on either the cyclists' preferred distance or intermediate locations. The benefits of this app are its versatility on multiple platforms and its social media component where users can view and interact with other users' created routes. This builds an ecosystem where cycling enthusiasts are able to mimic popular routes by other users and creates a better cycling experience for our users. Our vision is to create an avenue for cyclists of all experience groups to connect and discover exciting cycling opportunities which are just a click away.

1.5 References

[1] IEEE Std 830 IEEE Recommended Practice for Software Requirements Specifications https://ieeexplore.ieee.org/document/720574

[2] Bernd Breugge, Allen H Dutoit, Object-Oriented Software Engineering: Using UML, Patterns and Java, 3rd Edition, ISBN 10, 0-13-815221-7, Pearson, 2010.

[3] Martin Fowler, UML Distilled: A brief guide to the standard object modelling language, Third Edition, ISBN 0-321-19368-7, Pearson, 2004

2. Overall Description

2.1 Product Perspective

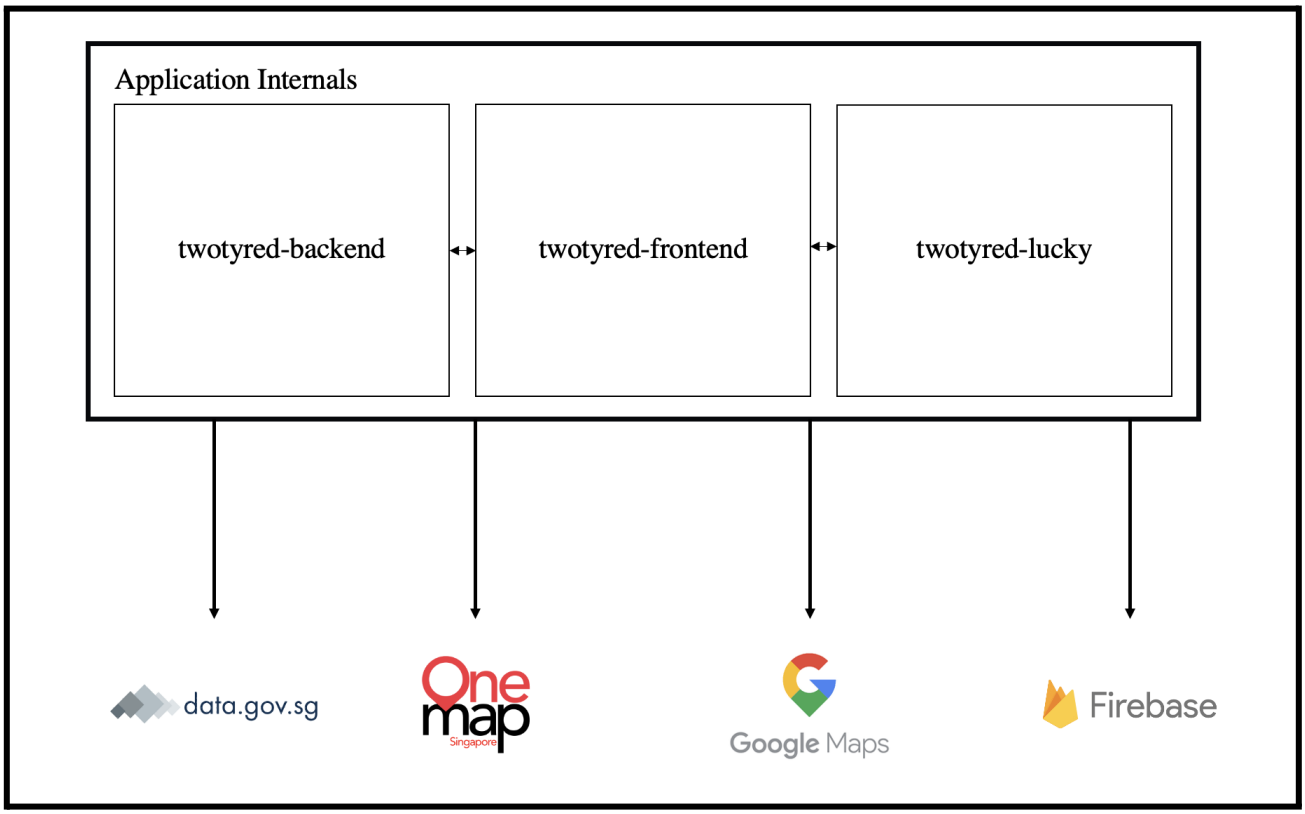


Figure 1: Overall Product Perspective of TwoTyred

TwoTyred is a progressive web application that facilitates the planning and sharing of cycling routes between fellow Singaporean cycling enthusiasts. The entire application is a novel, new and self-contained product that leverages various 3rd party services and application programming interfaces (APIs).

Internally, the TwoTyred application is split into 3 different codebases:

1. twotyred-frontend
2. twotyred-backend
3. twotyred-lucky

twotyred-frontend is responsible for the frontend of the TwoTyred application.

twotyred-backend is the backend for the TwoTyred application, responsible for database and frontend interactions
twotyred-lucky is a separate backend, developed specially for the purposes of the automatic route planning mode. More information on automatic route planning will be detailed in Section 3.1.1.5.

Externally, as aforementioned, the application interacts with multiple different 3rd-party services and APIs, namely OneMap, Data.gov, Google Maps and Firebase. All internal and external interactions and data fetching are fulfilled via the Representational State Transfer (REST)ful API protocol.

2.2 Product Functions

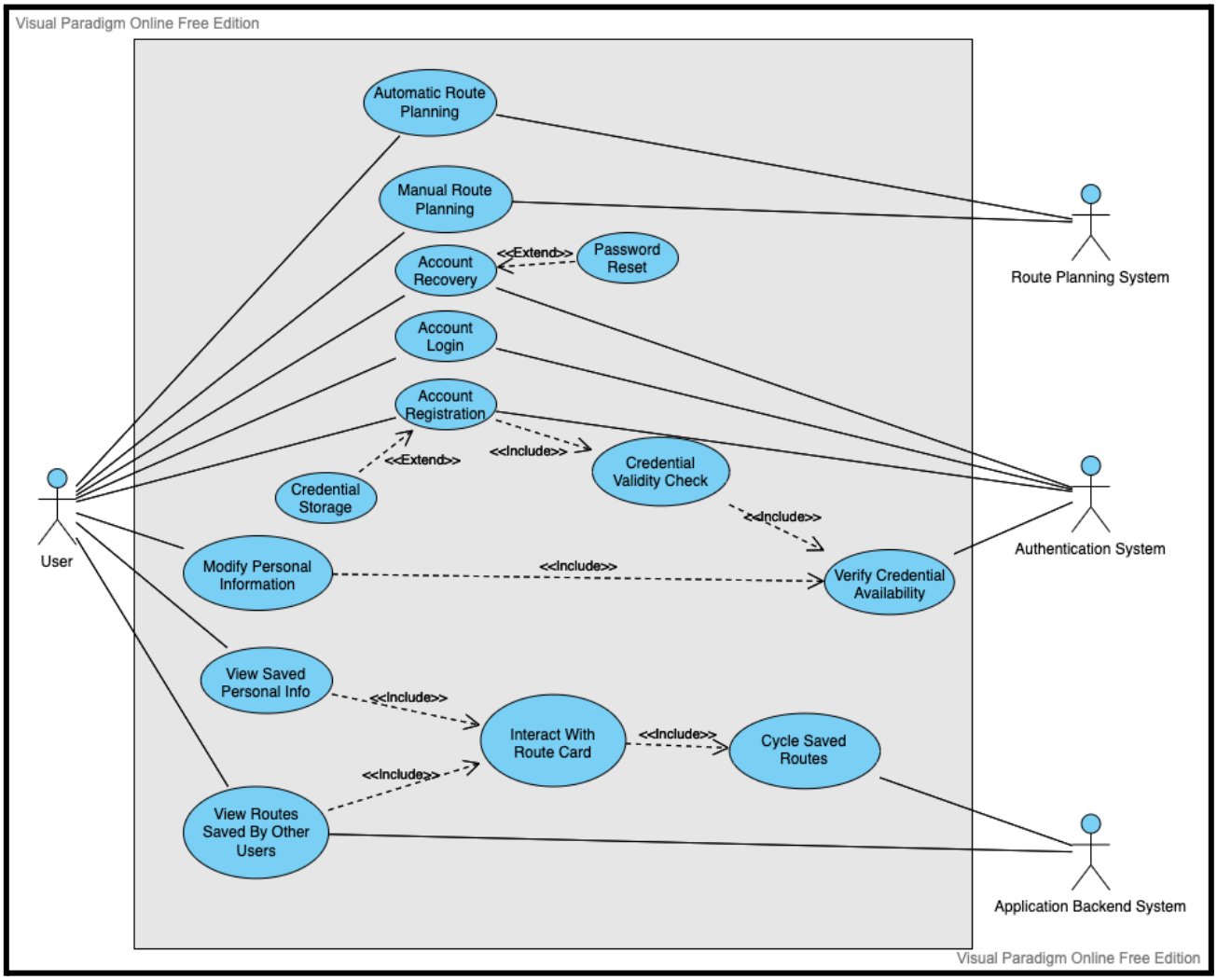


Figure 2: Use case diagram

Figure 2 details the various functions of the TwoTyred application, as well as the interrelation between these functions. They will be handled by the application backend, authentication and route planning systems behind the scenes.

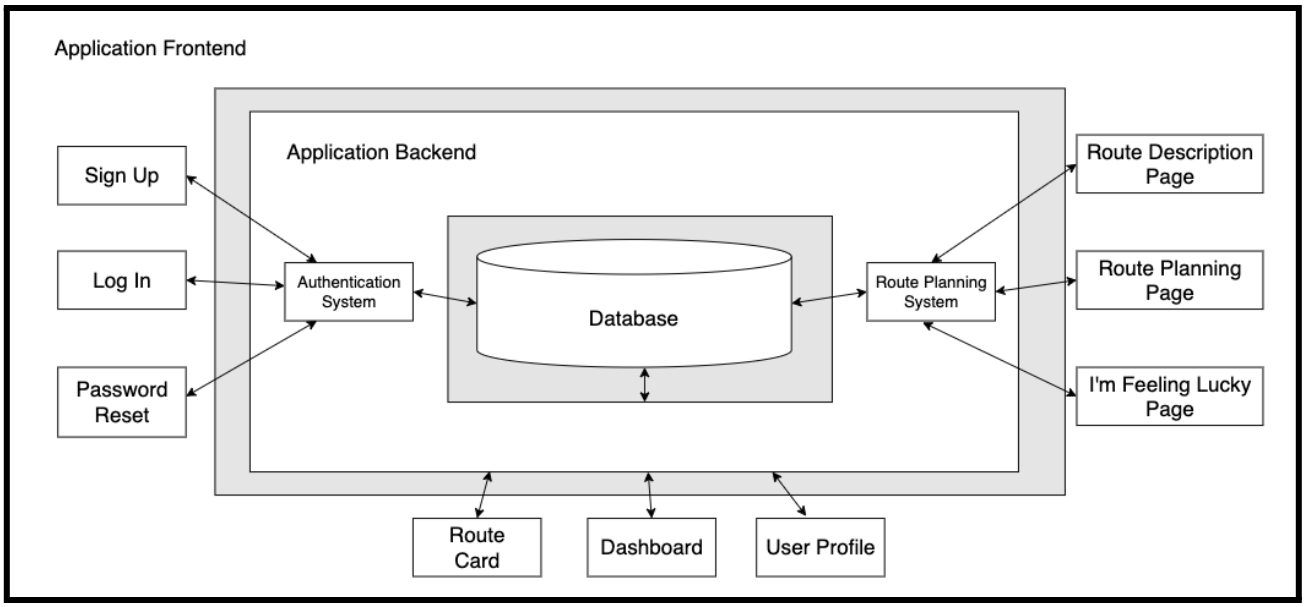


Figure 3: Application Architecture Diagram

Figure 3 gives a high-level overview of the entire system architecture, as well as their encompassing classes. More information regarding the various application classes can be found in Appendix B.

The TwoTyred Application must provide the following functionality at minimum, together with other requirements specified within this SRS document.

1. Account Registration: Users must be able to register an account on the TwoTyred platform. Saved routes, together with other miscellaneous cycling statistics, will be logged to their respective accounts.

2. Account Login: Users must be able to login to their registered accounts, to be able to access the various route planning features, and view routes created by other users.

3. Modify Personal Information: Users must be able to modify their personal information on the TwoTyred application platform, namely their username and profile picture.

4. Route Planning: Users must be able to plan cycling routes using one of the following two route planning features

a. Manual Route Planning: Users will manually indicate on the application the points that they wish to visit on their cycling trip, and the route planning service will connect these route points to provide users with a finalised route.
b. Automatic Route Planning: Users will indicate on the application the start point of their journey, as well as the distance that they wish to cycle, and the route planning service will automatically generate a route for them.

5. Interaction with Route Card: Routes that are planned and cycled by other users will be displayed on the dashboard page in the form of a route card. Users must hence be able to interact with this route card via liking, favouriting, and viewing more information about the route.

6. View Saved Personal Info: Users must be able to view their own personal statistics and routes tagged to their own accounts. All previously cycled routes, as well as cycling statistics, like total duration must be visible to the user

7. View Routes Saved by Other Users: Users must be able to view routes saved by other users, on the dashboard page in the form of a route card.

2.3 User Classes and Characteristics

2.3.1 User (Singaporean Cyclists)

The development of TwoTyred was targeted at Singaporean Cyclists. As we aim for our target users to utilise TwoTyred regardless of educational levels, proficiencies or technical expertise, the user interface was designed with a seamless user experience (UX) in mind, to minimise friction between current manual planning methods and the TwoTyred application.

2.3.1.1 Characteristics of User (Singaporean Cyclists)

1. The user is expected to possess an email account.
2. The user is expected to be Internet literate.
3. The user is expected to be able to navigate around the TwoTyred application, and interface with the various planning and route APIs.

2.4 Operating Environment

2.4.1 Client Side

Environments	Specifications
Operating System	Cross-Platform (MacOS, Linux, Windows, Android & iOS)
Browser	Apple Safari / Google Chrome / Mozilla Firefox / Microsoft Edge
Supported Devices	Desktop / Laptop / Tablet / Mobile

Figure 3: Client Side Operating Environment

2.4.2 Server Side

Environments	Specifications
Operating System	Cross-Platform (MacOS, Linux, Windows)
Web Server	Python3 with Flask / Docker & NodeJS with Express
Database	Cloud Firestore

Figure 4: Server Side Operating Environment

2.5 Design and Implementation Constraints

1. TwoTyred Frontend: Recommended to be developed on *NodeJS versions 16.17.0* and above. Packages recommended to be installed with *NPM versions 8.15.0* and above. Google Cloud Platform must be set up and Maps Javascript API must be enabled. API Key for Maps Javascript API must be stored under *twotyred-frontend/.env.local* with the VITE_MAPS_APIKEY variable (ie. VITE_MAPS_APIKEY=REDACTED).

2. TwoTyred Backend: Recommended to be developed on *NodeJS versions 16.17.0* and above. Packages recommended to be installed with *NPM versions 8.15.0* and above. Backend services must be instantiated over REST API calls. Personal OneMap API account is needed to interface with OneMap API, and credentials must be stored under *twotyred-backend/onemap-config/onemapCred.json*. Personal Firebase Project needs to be created to interface with Firebase Authentication and Cloud Firestore, with email authentication enabled. Firebase credentials must be stored under *twotyred-backend/firebase-config/service-account.json*.

3. TwoTyred Lucky: Recommended to be developed on *Python versions 3.11.0* and above. If building on docker, at least 2.1GB of disk space is required. Route Planning service must be instantiated over REST API calls. Personal OneMap API account is needed to interface with OneMap API, and credentials must be stored under *twotyred-backend/onemap-config/onemapCred.json*.

2.6 User Documentation

2.6.1 Product Demonstration

To ensure that the entire user experience is as seamless as possible, the various features of TwoTyred, along with accompanying visuals have been encapsulated inside a product demonstration video, which can be found at https://youtu.be/-XNhBp3FfF4.

2.6.2 Technical Documentation

The entirety of the source code can be found at: https://github.com/orgs/SWE-Fantastic-Four/repositories.

Repository	Description
TwoTyred	Overall Repository for all source code, including frontend, backend and I'mFeelingLucky backend
twotyred-frontend	TwoTyred Frontend Progressive Web Application
twotyred-backend	TwoTyred Backend to interface with cloud firestore, and various Singapore governmental APIs
twotyred-lucky	Separate TwoTyred I'mFeelingLucky Backend

lab1	• Functional & Non-functional Requirements
	• Data Dictionary
	• Initial Use Case Model
	  ○ Initial Use Case Diagram
	  ○ Initial Use Case Descriptions
	• UI Mockups

lab2	• Updated Use Case Diagram & Descriptions • Class Diagram (Key Boundary & Control Classes) • Sequence Diagrams • Dialog Map

lab3	• Complete Use Case Model • System Architecture • Completed Sequence Diagrams • Completed Dialog Map • Completed Class Diagram • Initial Project Skeleton
lab4	• Test Cases and Results • Demo Script • Source Code
lab5	• Presentation Slides • Software Requirements Specification

Figure 5: Github Documentation Specification

In-depth information about our technical specifications can be found at the README here: https://github.com/SWE-Fantastic-Four/TwoTyred

2.7 Assumptions and Dependencies

2.7.1 Assumptions

The list of assumptions are as follows:

1. This application will be used in Singapore only.
2. The user has a working network connection.
3. Uptime of third-party deployment services (netlify, replit, pythonanywhere) is guaranteed.
4. Uptime of Google's Javascript Maps APIs and Cloud Firestore is guaranteed.

2.7.2 Dependencies

2.7.2.1 Frontend Software Dependencies

All frontend software dependencies can be found under *package.json* inside the *twotyred-frontend* source code repository at:
https://github.com/SWE-Fantastic-Four/twotyred-frontend/blob/master/package.json
The frontend source code will be developed on *NodeJS Version 16.17.0*.

Dependency / Package	**Version**
@headlessUI/react	1.7.3
@heroicons/react	2.0.11
@react-google-maps/api	2.13.1
@reduxjs/toolkit	1.8.5
awesome-debounce-promise	2.1.0
axios	1.1.2
firebase	9.10.0
flowbite	1.5.3

flowbite-react	0.2.0
framer-motion	7.5.3
polyline-encoded	0.0.9
react	18.2.0
react-dom	18.2.0
react-redux	8.0.4
react-router-dom	6.4.0
vite-plugin-pwa	0.13.1
workbox-window	6.5.4

Figure 6: Frontend Software Dependencies (package.json)

2.7.2.2 Software Backend Dependencies

The software dependencies for the backend can be found under *package.json* inside the *twotyred-backend* source code repository at: https://github.com/SWE-Fantastic-Four/twotyred-backend/blob/master/package.json.
The backend source code was developed on *NodeJS Version 16.17.0*.

Dependency / Package	**Version**
cors	2.8.5
express	4.18.1
firebase-admin	11.0.1
node-fetch	3.2.10
polyline-encoded	0.0.9

Figure 7: Backend Software Dependencies (package.json)

The software dependencies for the I'mFeelingLucky backend can be found under *requirements.txt* inside the *twotyred-lucky* source code repository at: https://github.com/SWE-Fantastic-Four/twotyred-lucky/blob/main/requirements.txt
The I'mFeelingLucky backend source code was developed on *Python Version 3.11.0*.

Dependency / Package	**Version**
Flask	2.2.2
Flask-Cors	3.0.10
polyline	1.4.0
requests	2.28.1
waitress	2.1.2
osmx	1.2.2
networkx	2.8.7

Figure 8: I'mFeelingLucky Software Backend Dependencies (requirements.txt)

2.7.2.3 3rd-Party Hosting / Service Dependencies

Hosting Dependencies	**Up and Running as of (dd/MM/yyyy)**
Netlify	06/11/2022
Replit	06/11/2022
PythonAnywhere	06/11/2022
Service Dependencies	**Up and Running as of (dd/MM/yyyy)**
Firebase Firestore / Authentication	06/11/2022

Data.gov API services	06/11/2022
OneMap API Services	06/11/2022
Google Map API Services	06/11/2022

Figure 9: 3rd Party Hosting / Service Dependencies

3. External Interface Requirements

3.1 User Interfaces

3.1.1 UI Interfaces

3.1.1.1 Login Page

The login page has three buttons, “Sign In”, “forgot password”, and “create account”. If the user is using the application for the first time and does not have an account, the user should click on the “create account” first.

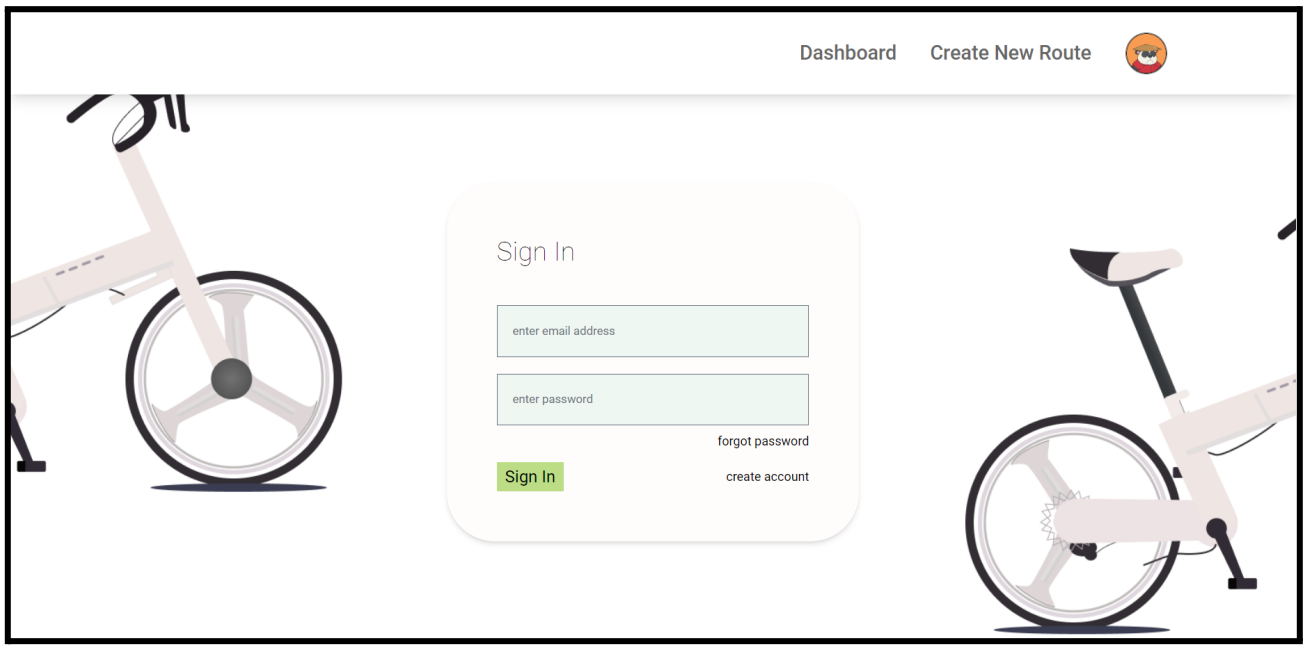


Figure 10: Login Page

3.1.1.2 Sign Up Page

On the Signup page, users should enter their email address, username, password and confirmed password. If the password doesn't meet the requirement, the user will get an error message such as "Password length too short." Clicking the "Back" button will bring the user back to the login page. Clicking the "Register" button will proceed to register. Upon successful registration, the user will be directed to the main dashboard page.

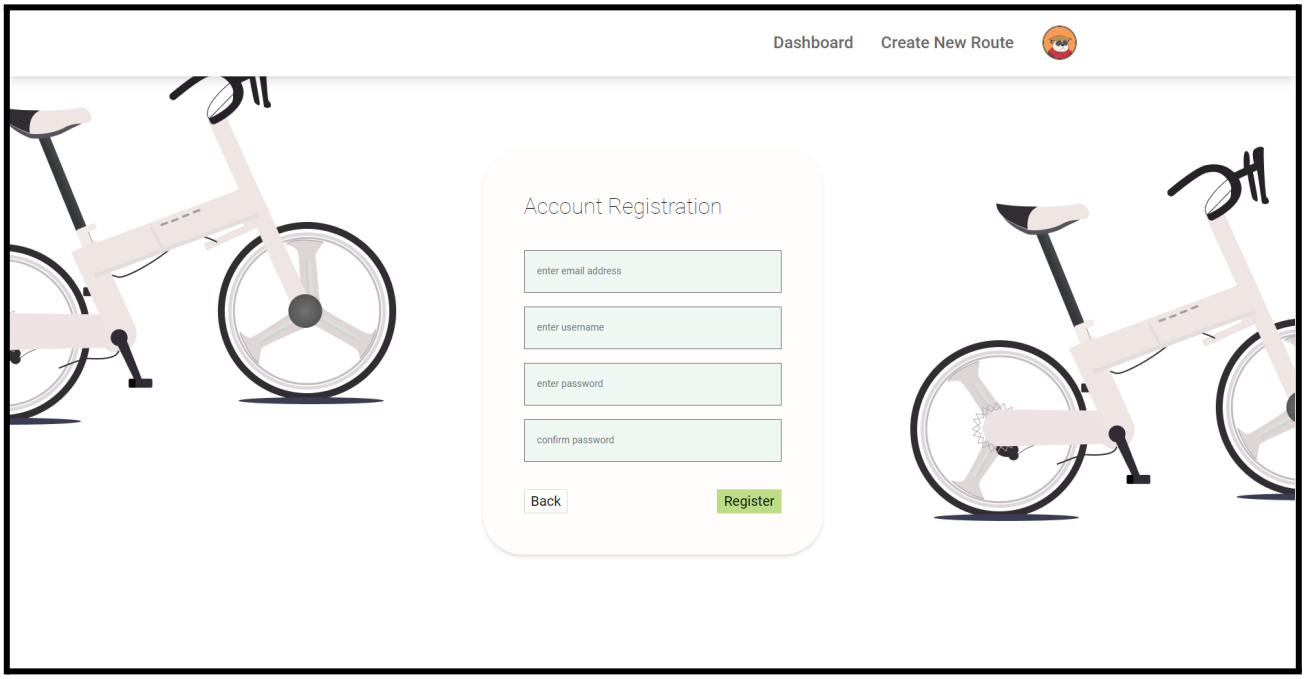


Figure 11: Sign up page

3.1.1.3 Dashboard Page

On the dashboard page users will be able to see other users' routes. Users are able to view routes based on recency or number of likes by selecting "Most Recent" or "Most Liked". Users can scroll down to view more routes. Users may click on the star button on each route card to add the route to their favourites, or click on the like button to like the route. If the user clicks the star button or like button again, they will be able to unfavourite or unlike the route.

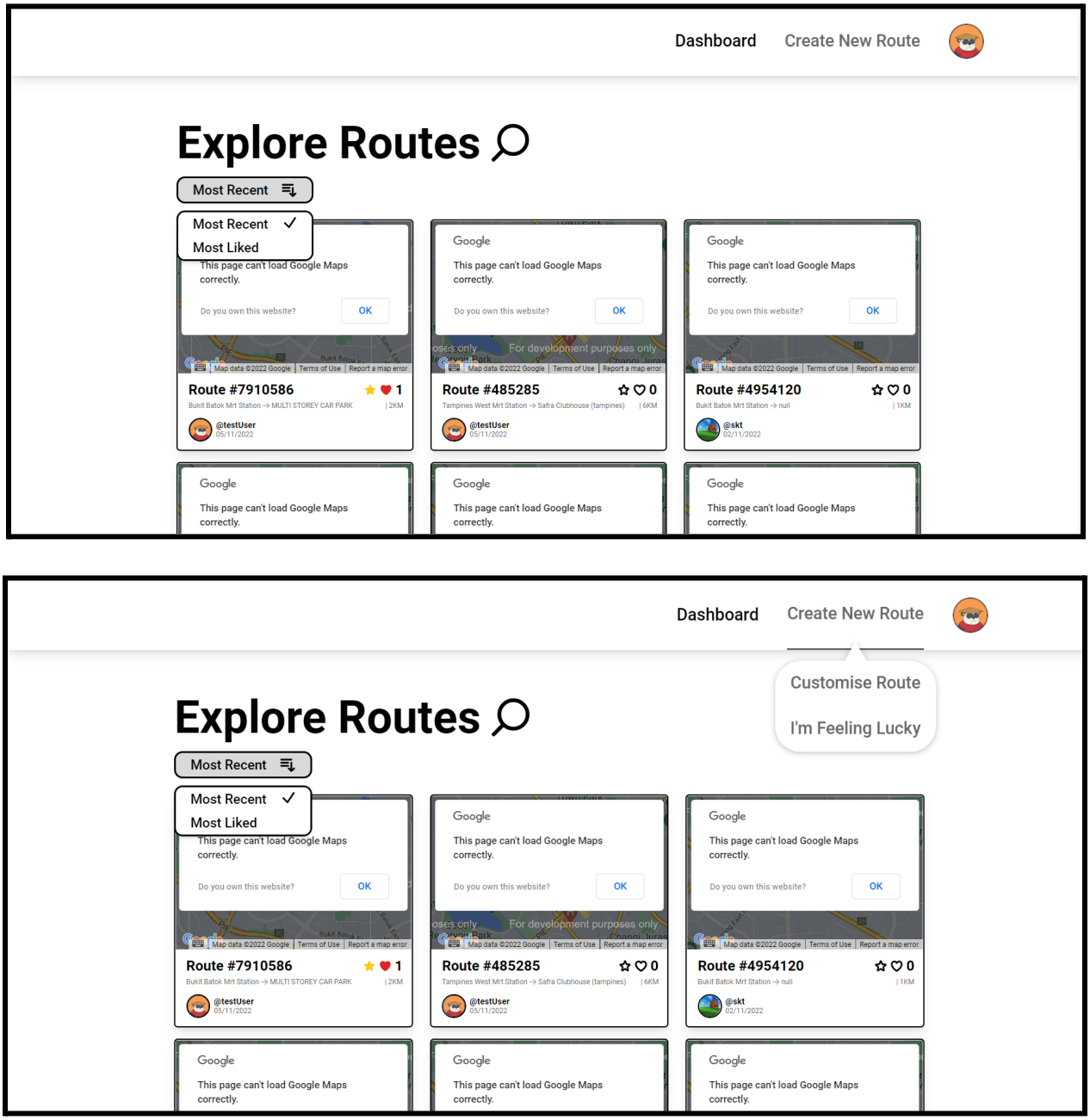
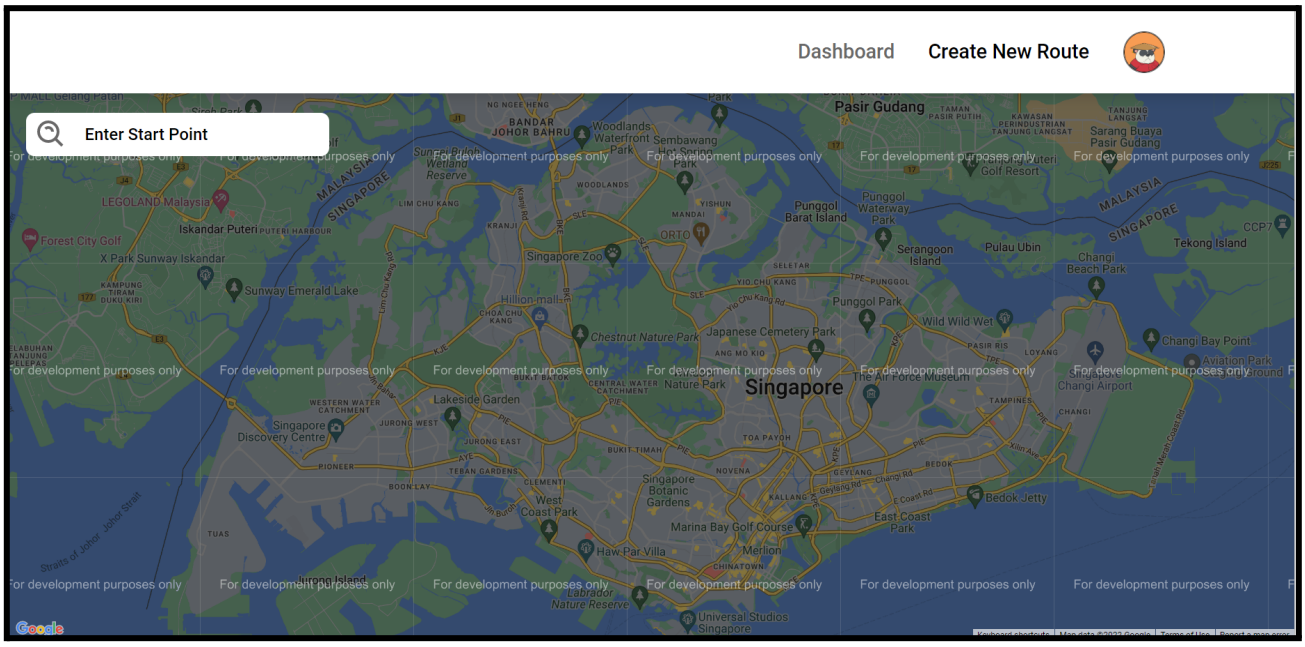
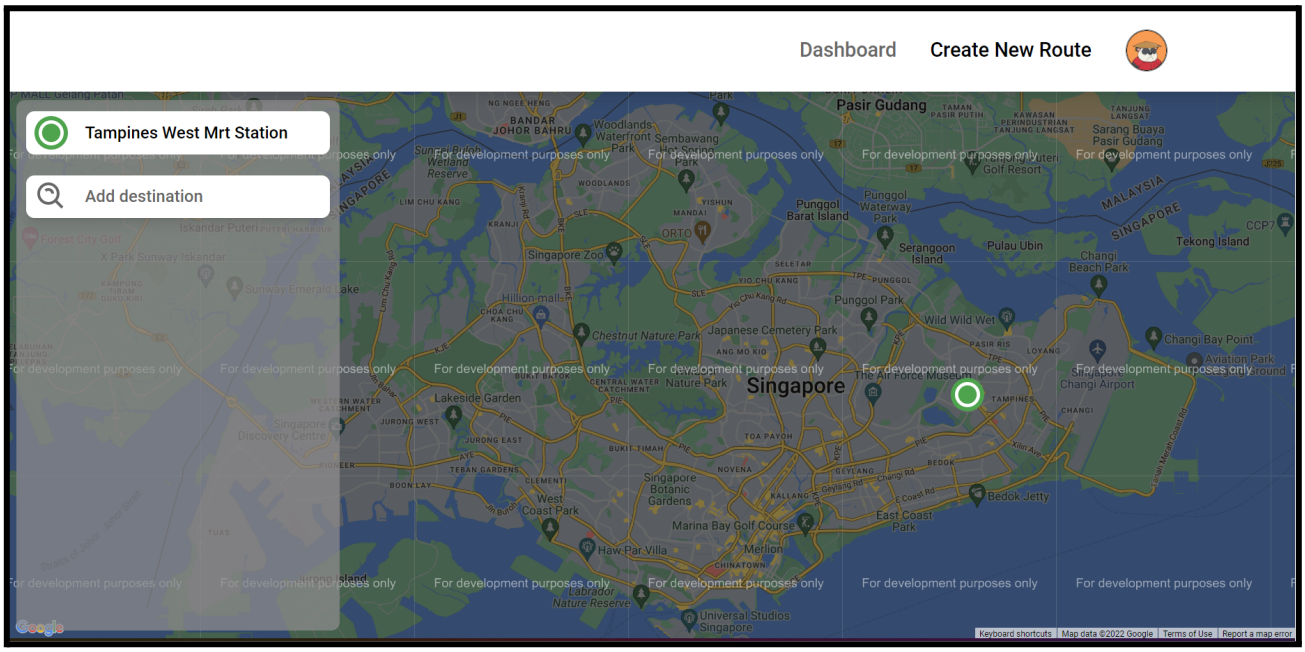


Figure 12: Dashboard

3.1.1.4 Customise Route Page

On the Customise Route Page, the "enter start point" button will allow the user to select the start point, and the user can add as many destination points as they need by selecting "Add destination". Clicking the "Generate Route" button will generate the route for the user.





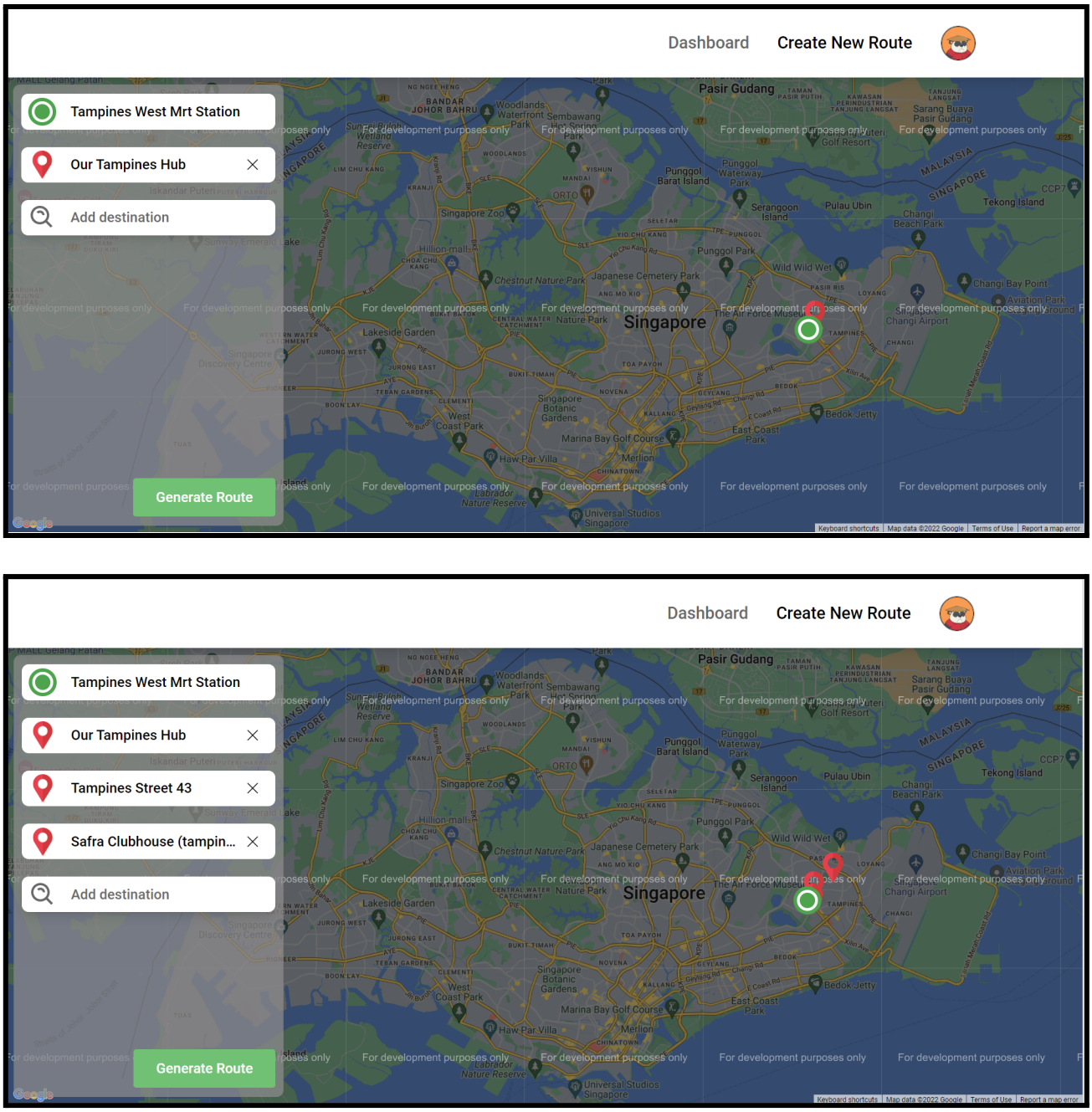
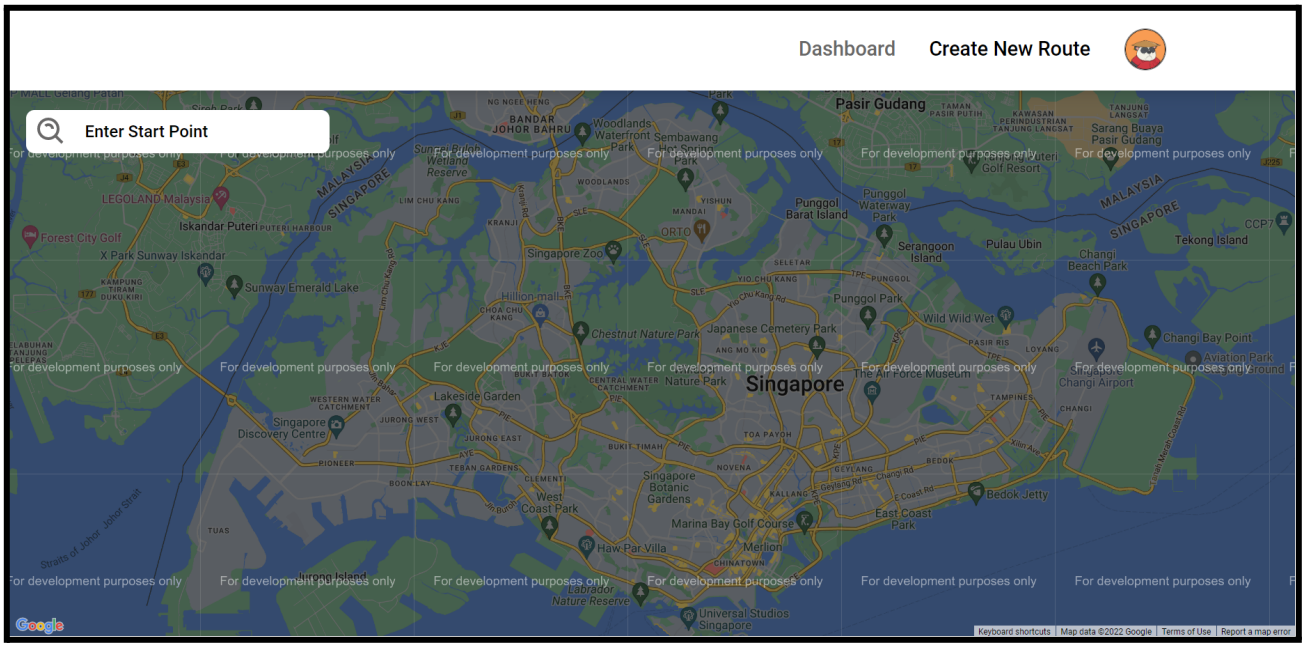
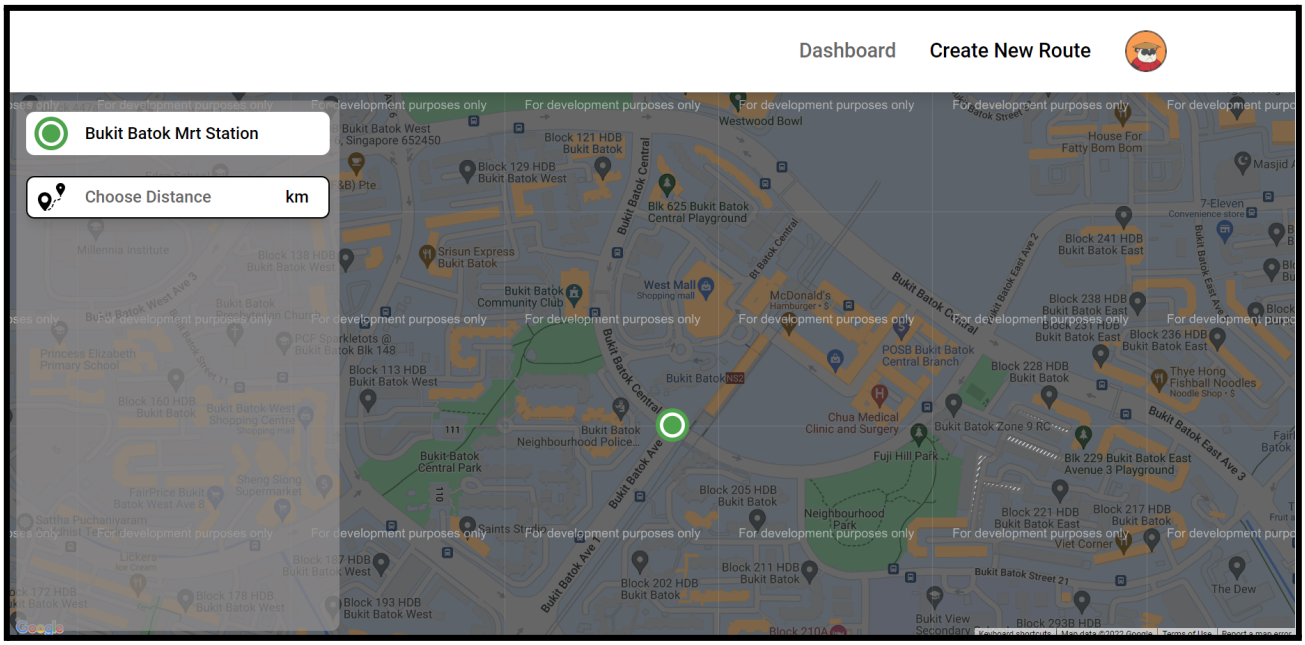


Figure 13: Customise Route page

3.1.1.5 I'm Feeling Lucky Page

On the I'm Feeling Lucky Page, the "enter start point" button will allow the user to select the start point. The user can enter the desired distance for the trip and after entering the distance a "Generate Route" button will show up on the page. Clicking the "Generate Route" button will generate the route for the user.





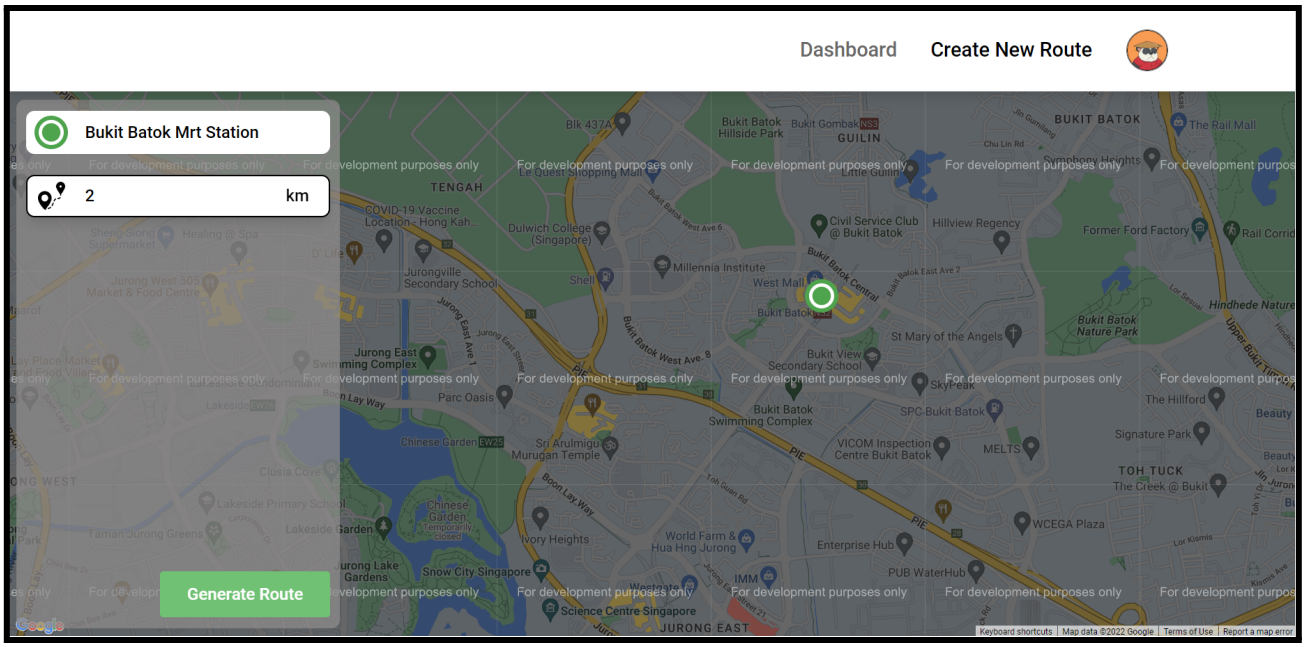


Figure 14: I'm feeling lucky page

3.1.1.6 Route Description Page

Under “Route Details”, the first box displays the date, time, weather condition and temperature. The second box displays the PM 2.5 index, and the third box displays the Route distance.

“Edit Route” button will bring the user to the Customise Route Page or I’m Feeling Lucky Page, where the user can modify the start point, the destination points or the route distance. “Cycle route” button will automatically save the route and a “Route successfully saved!” alert will pop up on the top right of the screen.

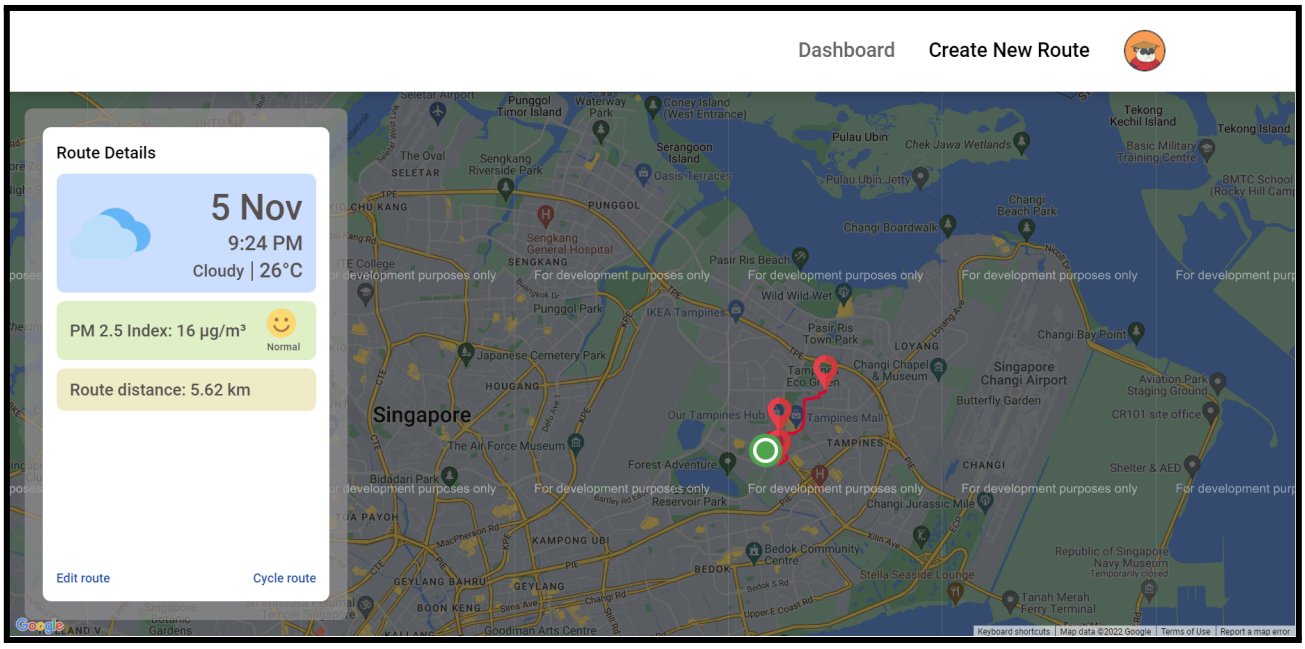


Figure 15: Route Description page

3.1.1.7 User Profile Page

The distance the user travelled and duration cycled will show on the user profile page. All the routes that the user has cycled will display under “Past Routes” and the routes that the user favorited will display under “Favourites”. If the user clicks on the setting icon beside the username, the user will be able to edit the username and profile picture.

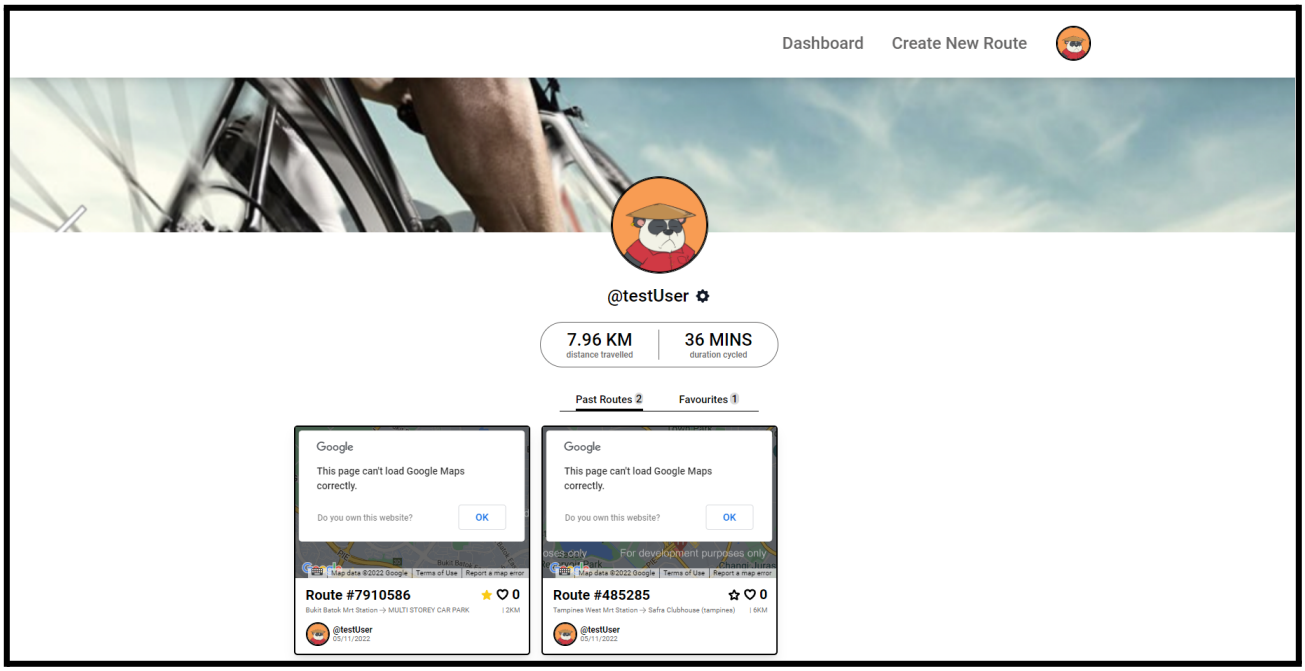


Figure 16: User Profile

3.1.1.7.1 Edit Profile

In the edit profile page, the user can enter his new username. The user will be able to edit his profile photo by clicking on the profile photo. The “Remove Current Photo” button will remove the user’s current photo and change to the default profile photo. The “Upload Photo” button will allow the user to upload a new profile photo.

The “Make Changes” button will update the changes.

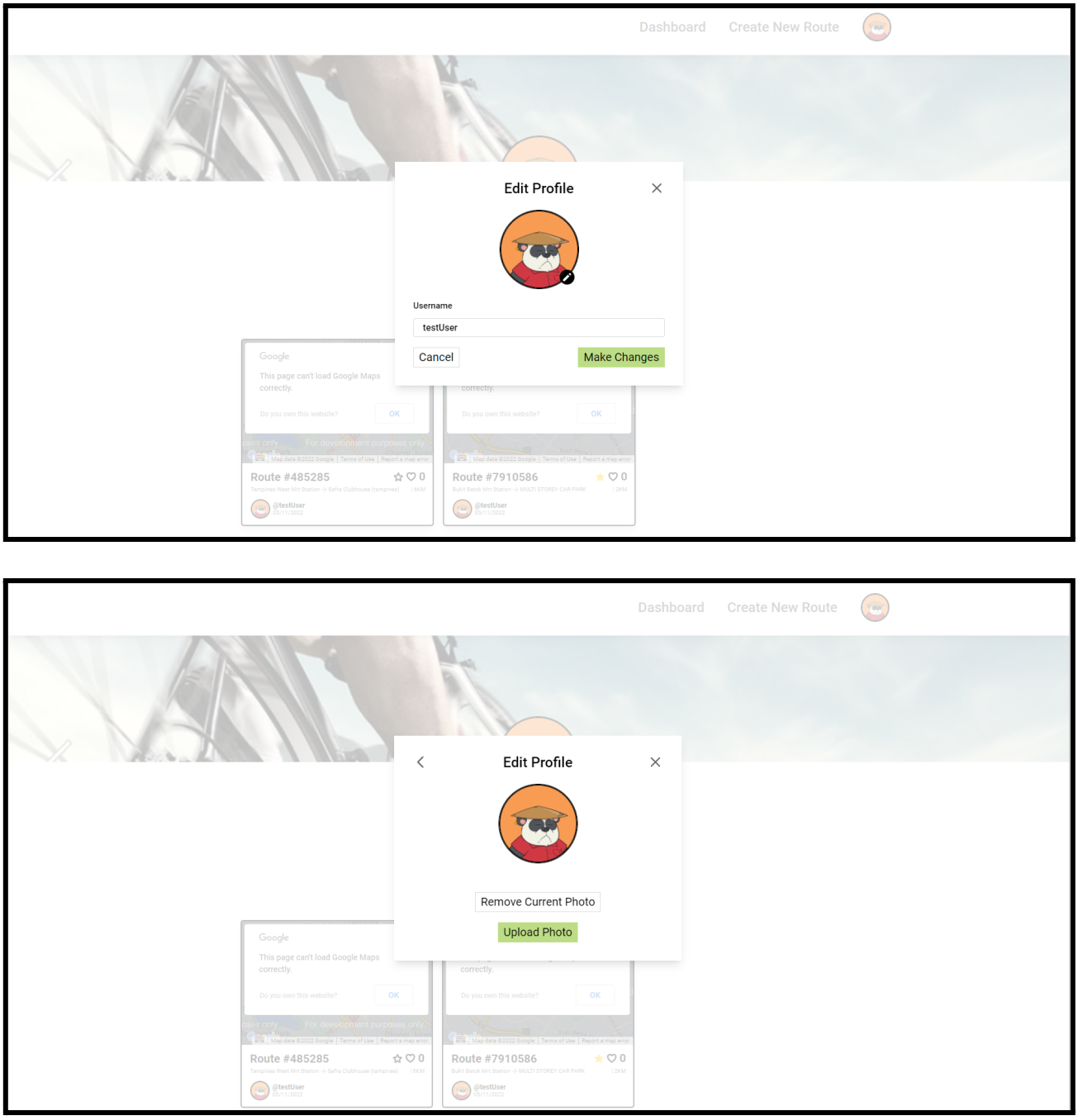


Figure 17: Edit Profile function

3.1.1.8 Mobile View

The TwoTyred application also has a mobile view, which allows the user to use the application on his mobile phone while cycling.

3.1.1.8.1 Mobile View: Manual Route Selection

Figure 18: Mobile View - Customise Route page

3.1.1.8.2 Mobile View: Automatic Route Selection

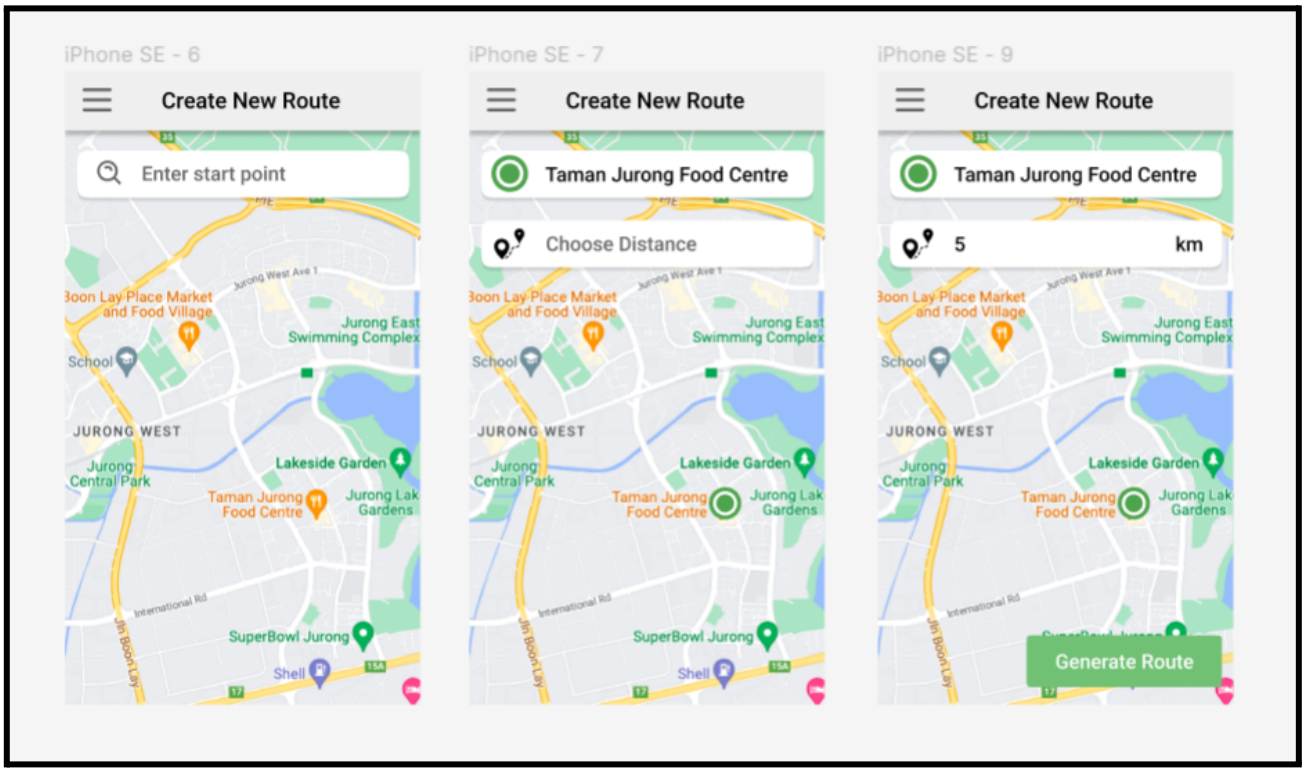


Figure 19: Mobile View - I'm feeling lucky page

3.1.1.8.3 Mobile View: User Authentication

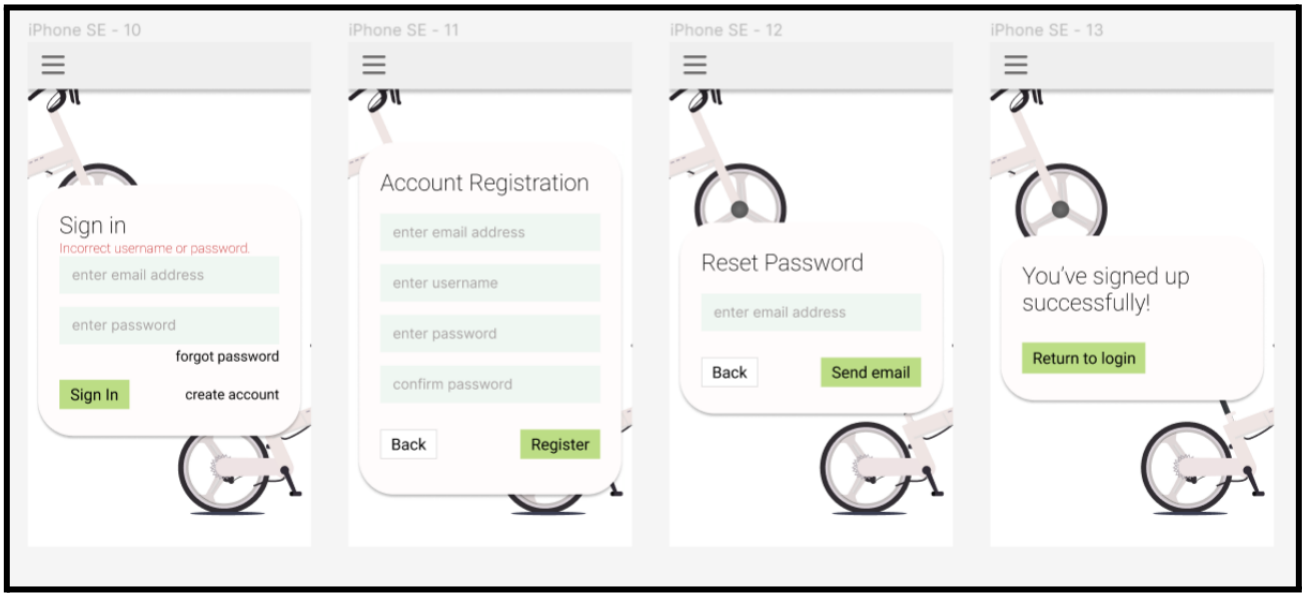


Figure 20: Mobile View - Sign In and Sign Up pages

3.1.1.8.4 Mobile View: User Profile

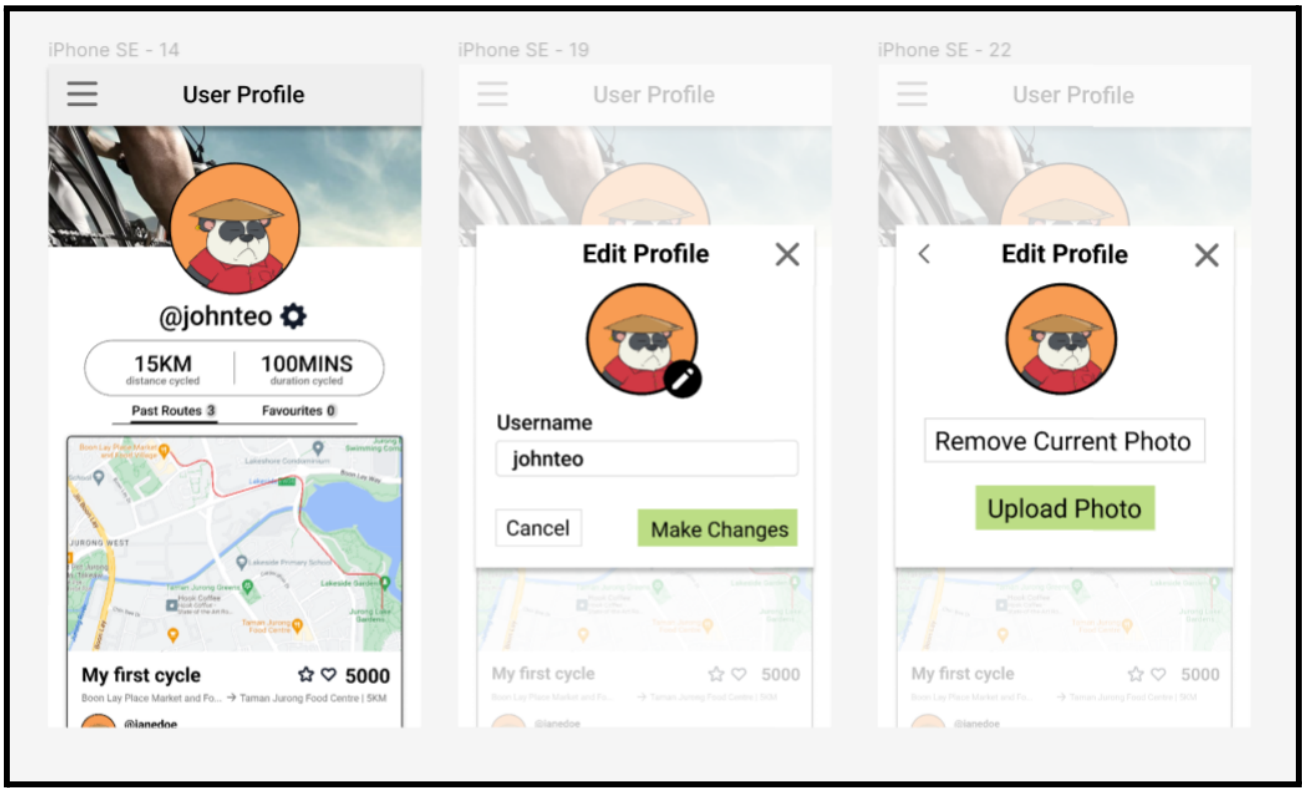


Figure 21: Mobile View - User Profile

3.1.1.8.5 Mobile View: Dashboard

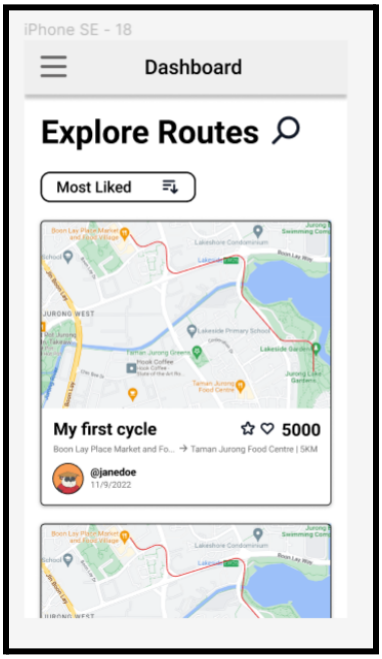


Figure 22: Mobile View - Dashboard

3.1.2 Standard Buttons and Functions

The Navbar will appear on every screen of this application.

3.1.2.1 Navbar - inactive before login

The Navbar is disabled on the login page and sign up page. After login, the Navbar will be active.

Figure 23: Navbar - inactive state

3.1.2.2 Navbar - Dashboard

Clicking on “Dashboard” will lead the user to the dashboard page.

Figure 24: Navbar - Dashboard Active

3.1.2.3 Navbar - New Route

On the Navbar, users can select “Customise Route” or “I’m Feeling Lucky” by hovering on “Create New Route”.

Figure 25: Navbar - Create New Route Active

3.1.2.4 Navbar - Profile

Hover on the top right will allow the user to go to the user profile page or log out.

Figure 26: Navbar - Profile Active

3.1.2.5 Mobile View: Side Navbar

In mobile view, the Navbar will appear as a side Navbar.

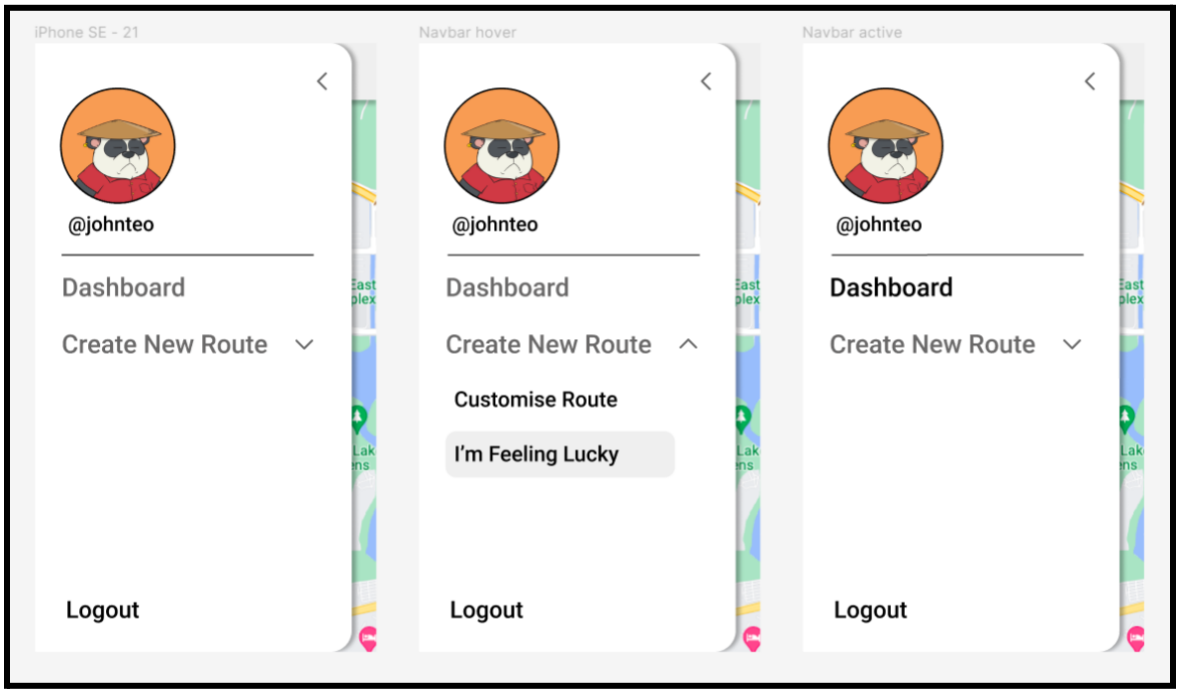


Figure 27: Mobile View - Navbar

3.1.3 Error Message Display

If the page the user is looking for cannot be found, an error page will show up. The “Return Home” button will bring the user back to the dashboard page.

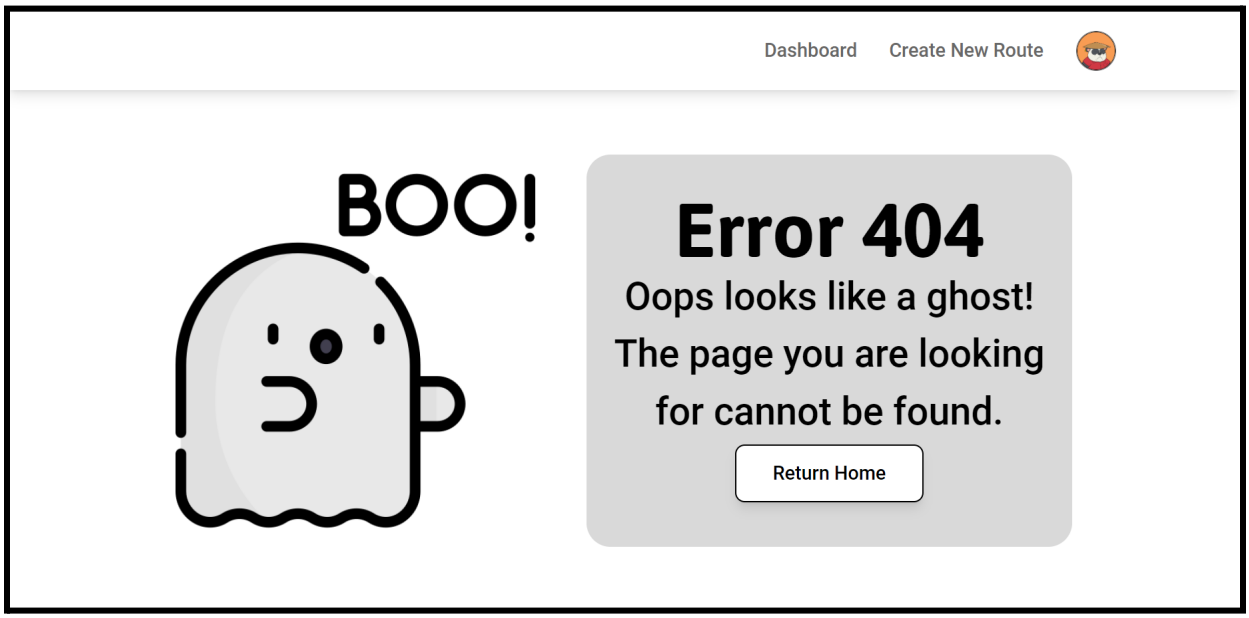


Figure 28: Error Message Page

Mobile View:

Figure 29: Mobile View - Error message

3.1.4 Screen Layout Constraint

Device	Screen Width
Mobile	320px - 480px
Tablet	480px - 640px
Laptop, Desktop	> 640px

Figure 30: Application width breakpoints

Figure 30 details the various screen width breakpoints for the TwoTyred application. The application operates on a minimum screen width of 320px and supports any larger screen widths. Keeping in mind different screen widths, the application was designed with a responsive layout in mind to accommodate for all devices. TwoTyred responsively renders and switches its UI at the application breakpoints of 480px and 640px.

3.2 Hardware Interfaces

The TwoTyred application must support desktops, laptops, Android and IOS mobiles and tablets. It must support MacOS, Linux and Windows operating systems.

All server-side components must execute on server-class computers. All client-side components must execute on workstation-class and personal-class computers.

The various hardware interface components are as follows:

1. Client Device (Mobile & Tablets: iOS / Android + Laptops & Desktops: macOS / Linux / Windows)
2. Server Hosting Computational Resources (Application Frontend: Netlify + Application Backend: Replit / PythonAnywhere)
3. Database Computational Resources (Google Cloud Firestore)

3.3 Software Interfaces

The software interface should follow the application frontend, application backend and database model. The interface must be able to connect to the Firestore database via application backend to store User and Route information.

3.3.1 Description of Services

Firestore Database

- Input: Create, Retrieve, Update, and Delete (CRUD) information
- Output: CRUD data
- Purpose: Provide application frontend with information about the route and user profile

Firebase Authentication

- Input: User email address and password
- Output: Validation of email address and/or password
- Purpose: To ensure that the user logs in to our application with the correct credentials

Data.gov.sg Weather Forecast API

- Input: Timestamp
- Output: Weather forecast for the next 2 hours, across all weather stations in Singapore
- Purpose: To retrieve weather forecast for Route Description Page
- GET request link: https://api.data.gov.sg/v1/environment/2-hour-weather-forecast

Data.gov.sg PM 2.5 API

- Input: Timestamp
- Output: Hourly PM 2.5 readings, across the major regions in Singapore
- Purpose: To retrieve PM 2.5 reading for Route Description Page
- GET request link: https://api.data.gov.sg/v1/environment/2-hour-weather-forecast

Data.gov.sg temperature API

- Input: timestamp
- Output: Air Temperature readings across 4 stations in Singapore
- Purpose: To retrieve temperature readings for Route Description Page
- GET request link: https://api.data.gov.sg/v1/environment/air-temperature

OneMap Routing API

- Input: 2 coordinates in WGS84 format
- Output: Cycling route connecting the start and end point
- Purpose: To obtain cycling route for Route Description Page
- GET request link: https://developers.onemap.sg/privateapi/routingsvc/route?start={start}&end={end}&routeType=cycle&token={token}

OpenStreetMap Dataset

- Input: Singapore's Relation ID
- Output: Graph that represents all cycling paths in Singapore
- Purpose: To create route for I'm Feeling Lucky function

Client-side UI applications must run within a web browser and not require any additional software interfaces on the client.

3.4 Communications Interfaces

The communication architecture must follow the client-server model. Communication between the client and server should utilise a REST-compliant web service and must be served over HTTP Secure (HTTPS). The client-server communication must be stateless. A uniform backend interface must separate the client roles from the server roles. Data flow between application frontend and application backend as well as between application backend and database must be asynchronous.

Behind the scenes, firebase authentication uses an internally modified version of scrypt to hash account passwords. This drastically improves the security of our stored passwords, ensuring that even admin members who are able to access the firebase project console are unable to view the password information.

4. System Features

4.1 Account Registration

4.1.1 Description and Priority

All users are required to register and create an account with the application before they are allowed to use other features in the application. Account registration requires the user to provide an email, a username and a password.

Priority: Medium

This feature is crucial as it allows users to access all other information. Proper optimisation of the data collected also decreases the costs of maintaining the database as the application increases in scale and the adverse impacts due to a data leak.

4.1.2 Stimulus/Response Sequences

Pre-condition	The user must have a valid email account.
Stimulus	The user selects the “create account” option in the login page
Flow of events	1. The frontend prompts the user for their account username, email address and password. 2. The user inputs their desired account username, email address, password and confirm password in another input field 3. The user clicks on the submit button to submit account credentials. 4. After the user has submitted their desired account credentials, the system checks for the validity of input credentials (Email must be valid, username must not be taken, passwords must match and must be alphanumeric, contain uppercase letters and symbols) 5. The system stores all the credentials in the database.
Alternative flows	AF-S1: Passwords do not match or username already taken 1. An error message will show on the application frontend.

	2. The user can edit input fields to rectify errors and resubmit a new set of credentials. AF-S2: Email account already registered 1. An error will show on the application frontend
Post-conditions	The user will be logged into the newly created account with the submitted credentials.
Exceptions	No network connection

4.1.3 Functional Requirements

No.	Requirement	Description	Error handling
1	Input credentials	Front-end must have input fields for the new user to enter their email, username, password and confirm password.	-
2	Submit credentials	A submit button must be available for the users to use entered credentials to create and account.	Error message will be shown on the frontend if the credentials are invalid.
3	Store credentials	If credentials are valid and an account was created, the credentials must be stored in a database. Users should then be able to use these credentials to log in to their account in the future.	Error message will be shown on the frontend and the account will not be created if there is a network error that prevents successful creation of the account.

4.2 Account Login

4.2.1 Description and Priority

All users should be able to login to their accounts by providing their account email address and password.

Priority: Medium
An account login feature is key as it is required for the user to access all of the route services. In addition, an account login feature is key to ensure customization of route information for the user. Nevertheless, the crux of the project lies in the route planning features, hence the priority of this use case is Medium.

4.2.2 Stimulus/Response Sequences

Pre-condition	The user must have a valid account
Stimulus	The user access the login page
Flow of events	1. The system prompts the user for their account username and password. 2. The user enters his/her account username and password into the respective fields on the Login page. 3. The system cross-checks the input credentials with those from their database. 4. The frontend redirects the user back to the dashboard page.
Alternative flows	AF-S1: If the input credentials are wrong 1. The frontend displays the message “Wrong username or password.” 2. The frontend returns to step 1. AF-S2: If the input credentials are wrong for 5 times 3. The frontend displays the message “You have been locked out of your account.”

	4. The authentication system locks the user out of their account for 10 minutes. 5. The authentication system sends an email to the user's email to inform them that their account has been temporarily locked for 10 minutes, and recommend them to change their password if they were not the ones trying to login to the account. 6. The frontend returns to step 1
Post-conditions	The user must be logged in to a registered account
Exceptions	No Network Connection

4.2.3 Functional Requirements

No.	Requirement	Description	Error handling
1	Input credentials	Front-end must have input fields for the user to enter their email and password.	-
2	Submit credentials	A submit button must be available for the users to submit their credentials.	Error message will be shown on the frontend if the credentials are invalid.
3	Validate credentials	If the email and password are valid, the user will be able to log in to his/her account and will be redirected to the dashboard page.	Error message will be shown on the front-end if the email and/ or password entered are invalid or if there is a network error.

4.3 Modify Personal Information

4.3.1 Description and Priority

All users should be able to modify their username and profile picture

Priority: Low
This feature is of low priority as it does not value-add to the route planning functions of our project and only serves to increase the list of “account settings” functionality that is available.

4.3.2 Stimulus/Response Sequences

Pre-condition	1. The user must be logged in to modify username and profile picture. 2. The application must have access to the user’s local storage or local camera device (if any).
Stimulus	–
Flow of events	1. The user will request to edit their personal profile. 2. The user can choose to upload a new photo from their local storage or to take a new picture with the local camera device. 3. The Application Backend System will store the new profile picture in the database. 4. The user can also choose to give a new username. 5. When the user confirms the modification of username, the system will verify if the new username input has already been used. 6. If there are no duplicates, the Authentication System will update the user’s username in the database.
Alternative flows	AF-1: If the username is already in use 1. The Authentication System will reject the updating of the new username. 2. The Authentication System will send an error message to the application front-end. 3. The application returns to step 2.

Post-conditions	4. The new profile picture must be saved in the Application Backend System. 5. The new username must be saved in the Authentication System. 6. The new username and profile picture must be displayed in the user profile page.
Exceptions	No Network Connection

4.3.3 Functional Requirements

No.	Requirement	Description	Error handling
1	Upload new profile picture	Front-end must have buttons to allow for the users to upload their photos from local storage storage or local camera device (if any)	Error message will be shown if the user is unable to upload the photo
2	Input new password	Front-end must have a field for the user to input the new password	–
3	Validate new username	System will verify if the new username is already used by another account	Error message will be shown if the username is already used by another account
4	Store new username	System will store new username in the database	Error message will be shown if the username is already used by another account
5	Display updated personal information	Front-end must be able to reflect the updated username and/ or profile picture	–

4.4 Manual Route Planning

4.4.1 Description and Priority

Users can use the manual route planning function to plan their cycling route. They are required to select the start and end point of their journey and are allowed to select any other mid-points. A cycling route, which relies on OneMap API routing service, will be displayed on the map in the application.

Priority: High
As our application is a cycling route planning and sharing application, the ability to plan a cycling route is of high priority. Good design principles should be applied to the feature to increase its responsiveness, allowing users to view the planned route quickly

4.4.2 Stimulus / Response Sequences

Pre-condition	1. The user must be logged in to a registered account. 2. Database has sufficient storage available for route storage.
Stimulus	User selects the “Customise Route” option under the “Create New Route” tab.
Flow of events	1. The frontend prompts the user for their start point, end point and intermediate points along the route. 2. The user selects their route points via map markers. 3. The route planning system will determine the optimal route, PM2.5 index and weather data by querying the OneMap API and weather API. 4. The frontend will display the proposed route, along with the route distance, PM2.5 index and weather data. 5. The user will choose “Cycle Route” or “Edit Route”. 6. In the case that the user chooses “Cycle Route”, the route will be saved under the user’s profile. 7. In the case that the user chooses “Edit route”, the frontend will redirect the user back to step 2.

Alternative flows	AF-S3: Point not within Singapore 1. Display Error Message - Failed to generate route
Post-conditions	A route recommended based on the user's start, end and intermediate points input.
Exceptions	1. No Network Connection 2. Point chosen is inaccessible by cycling. a. A route will be returned by factoring in the nearest point to the chosen point.

4.4.3 Functional Requirements

No.	Requirement	Description	Error handling
1	Select points on map	The user must be able to indicate a start and end point on the map, as well as other intermediate points selected	-
2	Generating Route	The user must be able to get a planned route after submitting all selected points	An error message will be displayed on the screen when the system is unable to generate a cycling route.
3	Display Route	The route returned by the route planning system must be displayed on the map.	An error message will be displayed on the screen when the system is unable to display a cycling route.
4	Display Route Details	The system must display the conditions of the route, including the distance, weather forecast, temperature and PM2.5 index at the time of route generation.	No value will be given to these fields if the API request fails and a "-" will be shown at each of the fields.

4.5 Automatic Route Planning

4.5.1 Description and Priority

Users can use the route planning function to plan their cycling route. They are required to select the start and the desired cycling distance. A cycling route, relying on the I'mFeelingLucky backend, will be displayed on the map in the application.

Priority: High
As our application is a cycling route planning and sharing application, the ability to plan a cycling route is of high priority. Good design principles should be applied to the feature to increase its responsiveness, allowing users to view the planned route quickly

4.5.2 Stimulus / Response Sequences

Pre-condition	1. The user must be logged in to a registered account. 2. Database has sufficient storage available for route storage.
Stimulus	User selects the "Customise Route" option under the "Create New Route" tab.
Flow of events	1. The frontend prompts the user for their start point and distance 2. The user selects their start point via map markers or address input. 3. The user inputs the distance into a field on the frontend 4. The route planning system will determine the optimal route, PM2.5 index and weather data by querying the I'mFeelingLucky backend and Data.gov Weather API. 5. The frontend will display the proposed route, along with the route distance, and the latest PM2.5 index and weather data. 6. The user will choose to "Cycle Route" or "Generate another route". 7. In the case that the user chooses to "Cycle Route", the route will be saved under the user's profile. 8. In the case that the user chooses to "Generate another route", the frontend will redirect the user back to step 1.

Alternative flows	–
Post-conditions	A route must be recommended based on the user's start point and cycling distance inputs.
Exceptions	No Network Connection

4.5.3 Functional Requirements

No.	Requirement	Description	Error handling
1	Select points on map	The user must be able to indicate a start on the map and the desired cycling route distance in a field	-
2	Generating Route	The user must be able to get a planned route after submitting all selected points	An error message will be displayed on the screen when the system is unable to generate a cycling route.
3	Display Route	The route returned by the route planning system must be displayed on the map.	An error message will be displayed on the screen when the system is unable to display a cycling route.
4	Display Route Details	The system must display the conditions of the route, including the distance, weather forecast, temperature and PM2.5 index at the time of route generation.	No value will be given to these fields if the API request fails and a "-" will be shown at each of the fields.

4.6 Interact with Route Card

4.6.1 Description and Priority

All users should be able to carry out all interactions designed for the route card. The designed interactions are mentioned below:

1) Like and Unlike routes
2) Favourite and Unfavourite routes
3) View more route information

Priority: Medium

Even though Interactions 1 (Like and unlike routes) and 2 (Favourite and Unfavourite routes) mentioned above are not key to the route planning features which form the crux of this project, Interaction 3 (View more route information) plays a crucial part in the flow for a user to cycle a route shown on the dashboard. Hence, overall, the priority for this use case is Medium.

4.6.2 Stimulus/Response Sequences

Pre-condition	1. The user must be logged in. 2. The user must be on a page where he can view a route card on the user interface (Dashboard / User Profile page)
Stimulus	–
Flow of events	1. The user should be able to see a route card. Each of these route cards should have a like button (heart), a like counter, a favourites button (star). 2. If the user clicks on the like button, the Application Backend System should record a like for the route in the database and increase the like counter accordingly on the application frontend. 3. If the user clicks on the favourite button, the Application Backend System should store the route into the user's database and subsequently display the route in the user's "Favourites" tab in his/her User Profile page.

	4. If the users were to click on any other parts of the route card, the application will redirect them to the route description page, where they can view more details about the route. 5. The users will be able to select a “Cycle Route” button, after which the Backend Application System will record the ride in the database. 6. This newly saved route will be displayed in the user’s “Past Routes” tab in his/her User Profile page.
Alternative flows	–
Post-conditions	The user successfully interacts with a route card.
Exceptions	No Network Connection

4.6.3 Functional Requirements

No.	Requirement	Description	Error handling
1	Like Route	● The route card should consist of a “Like” button (heart) and the like counter. ● Upon clicking the unfilled “Like” button, the “Like” button will be filled ● The like count displayed on the route card will increase by 1	Error message will be shown if the route cannot be liked
2	Unlike Route	● The route card should consist of a “Like” button (heart) and the like counter. ● Upon clicking the filled “Like” button, the “Like” button will be unfilled ● The like count displayed on the route card will decrease by 1	Error message will be shown if the route cannot be unliked
3	Favourite Route	● The route card should consist of a Favourite button ● Upon clicking the unfilled “Favourite”	Error message will be shown if the route cannot be added to

		button, the “Favourite” button will be filled ● The route will be displayed in the user’s “Favourites” tab in his/her User Profile page.	“Favourites”
4	Unfavourite Route	● The route card should consist of a Favourite button ● Upon clicking the filled “Favourite” button, the “Favourite” button will be unfilled ● The route will be removed from the user’s “Favourites” tab in his/her User Profile page.	Error message will be shown if the route cannot be removed from “Favourites”
5	View More Route Information	● If the users were to click on any other parts of the route card, the application will redirect them to the route description page, where they can view more details about the route.	Error message will be shown if additional route information cannot be displayed

4.7 View Saved Personal Info

4.7.1 Description and Priority

Users will be able to access a user profile page where they can view all of their information within the application. This includes user profile picture, username, distance they've cycled, duration they've cycled, routes that they've cycled and routes they've favourited.

Priority: Medium

The user page helps the users to track cycling statistics and routes that they've cycled, increasing utility for the users who actively track their exercise activities. Furthermore, the ability to view username and profile picture, as well as the ability to view routes that they've saved, is crucial to the social media aspect of the application. However, this feature is an added feature and will not affect the application's route planning abilities if it was absent.

4.7.2 Stimulus/Response Sequences

Pre-condition	1. The user must be logged in.
Stimulus	The user must click on the "profile" option under the "User" tab.
Flow of events	1. The user should be able to see their latest username and profile picture when he/she clicks on the application's personal profile tab. 2. The user should also be on the "Past Routes" tab and should be able to see a list of past rides in the form of multiple route cards. This should come with details such as start and end points, as well as the number of likes for each route. 3. If the user clicks on any part of the route card, he/she should be able to interact with the route card. 4. If the user clicks on the "Favourites" tab, he/she should be able to see all route cards that they've favourited
Alternative flows	AF-S1: No routes cycled 1. The application will show a "No Previous Rides" message AF-S2: No favourite routes

	1. The application will show a “No Favourite Routes” message
Post-conditions	The user should be able to view all personal information mentioned above
Exceptions	No Network Connection

4.7.3 Functional Requirements

No.	Requirement	Description	Error handling
1	View profile picture and username	Application must be able to display account’s latest username and profile picture	-
2	View route cards of past routes cycled	Application must fetch all past routes cycled from the database and display the routes as route cards	-
3	View route cards of favourite routes	Application must fetch all favourite routes from the database and displaye the routes as route cards	-

4.8 View routes saved by other users

4.8.1 Description and Priority

Users will be able to access a dashboard where they are able to see all routes that have been created by all other users. These routes can be sorted by recency of creation or by the number of likes that they have received.

Priority: Medium
The ability to view and interact with routes saved by other users, is crucial to the social media aspect of the application. However, this feature is an added feature and will not affect the application's route planning abilities if it was absent.

4.8.2 Stimulus/Response Sequences

Pre-condition	1. The user must be logged in.
Stimulus	The user must click on the "Dashboard" tab.
Flow of events	1. The user should be able to see a list of routes saved by other users, in the form of a route card, when they select the application's dashboard tab. The route cards will be sorted based on the greatest number of likes by default. 2. If the user clicks on any part of the route card, he/she will be able to interact with the route card. 3. If the user chooses to click on the dropdown menu and sorts by date, the Application Backend System will return a list of route cards, sorted by their recency, to the Application's frontend.
Alternative flows	AF-S1: No routes created 1. The application will show a "No Previous Rides" message
Post-conditions	The user should be able to view all routes that have been created by other users.

Exceptions	No Network Connection

4.8.3 Functional Requirements

No.	Requirement	Description	Error handling
1	View all routes	Users should be able to see all routes that have been created on the application	-
2	Sorting	Users should be able to see the routes being sorted by recency or by number of likes received.	-

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Requirement type	Description
Overall performance	● The application shall not crash or close except when authorised by its user ● The application shall always be available to users while they are connected to the internet
Authentication requirements	● The login feature shall be able to authenticate users within 5s ○ Users may be misguided into thinking they forgot their password and try to change it
Route	● The route planning feature shall be able to generate a route within 30s ○ Users will lose their patience if route takes a long time to generate and may close the application ● The application shall be able to upload a user's selected route to their profile within 5 minutes. ○ User's may be confused if they do not see their newly saved routes inside their profile and may crash the system by continually trying to generate and save their routes ● The application shall be able to load more routes when required by the dashboard within 5 seconds ○ Users may lose their patience if they have to continually wait for extended durations while scrolling the most popular routes and close the app

Scalability	● The application shall be able to support a peak of 1500 concurrent users during peak periods such as weekends, public holidays and rush hours ○ This is roughly 1% of all cyclists in singapore ● The application shall be able to support 150 concurrent users during non-peak periods ○ This is roughly 0.1% of all cyclists in singapore

5.2 Safety Requirements

Requirement type	Description
Damage to device	● The application shall not cause a phone to reach 35 degree celsius ○ Application shall not run in the background to prevent strain on the phone's processor
Damage to user	● The application shall not be used when a user is moving too quickly ○ The user may be involved in an accident if they are diverting their attention from the road onto the application

5.3 Security Requirements

Requirement type	Description
Privacy	● The application shall not disclose confidential information about its users. ● The application shall only require user's email and password as verification

Security	• The application shall not grant unauthorised access to any 3rd party application • The application shall not transmit or receive data from 3rd party applications or servers while in use • The application shall have its own security to prevent unauthorised *write/delete* access but will have no restriction on *read* access. • The application shall practise the principle of least privilege for defining access-level requirements of the software system and its associated services. • The application shall detect malicious login attempts and temporarily suspend these accounts

5.4 Software Quality Attributes

Requirement type	**Description**
Availability	• The application shall be supported by android and IOS devices • The application shall be supported on Google chrome, Firefox, safari and Microsoft edge web browsers • The application shall be supported on the newest android and IOS releases
Robustness	• Upgrade all third-party libraries and frameworks to the latest version by 10 working days since their release date
Testability	• The application should have a 99% unit test coverage for both frontend and backend code
Maintainability	• All exceptions thrown shall be logged into a third party system for 12 months and be searchable by the developers

5.5 Business Rules

Requirement type	Description
User	● Users are able to sign up for an account on the sign up page ● Users are able to reset password on the reset password page ● Users are able to log in on the login page ● Users are able to create new routes in the route planning page ● Users are able to like/favourite routes in the dashboard or in their profile page ● Users are able to change their name and profile picture on their profile page

6. Other Requirements

Appendix A: Glossary

Term	Definition
Route	A way or course taken from a starting point to an ending point.
Destination	The end point a user wants to go to.
Weather Conditions	The temperature and prediction of rain of a certain date and time.
API	Application Programming Interface. An interface or communication protocol between different parts of a computer program intended to simplify the implementation and maintenance of software
Dashboard	A user interface page that will display the routes created by other users
User Profile	A display of user's information and cycling history
User Authentication	A process that allows users to sign up, log-in, choose username and set password.
Route Card	A user interface component in the User Profile and Dashboard where users can view the summary of a route
Database	Data storage for all user and route information
I'mFeelingLucky	Automatic route planning service provided by TwoTyred

Appendix B: Analysis Models

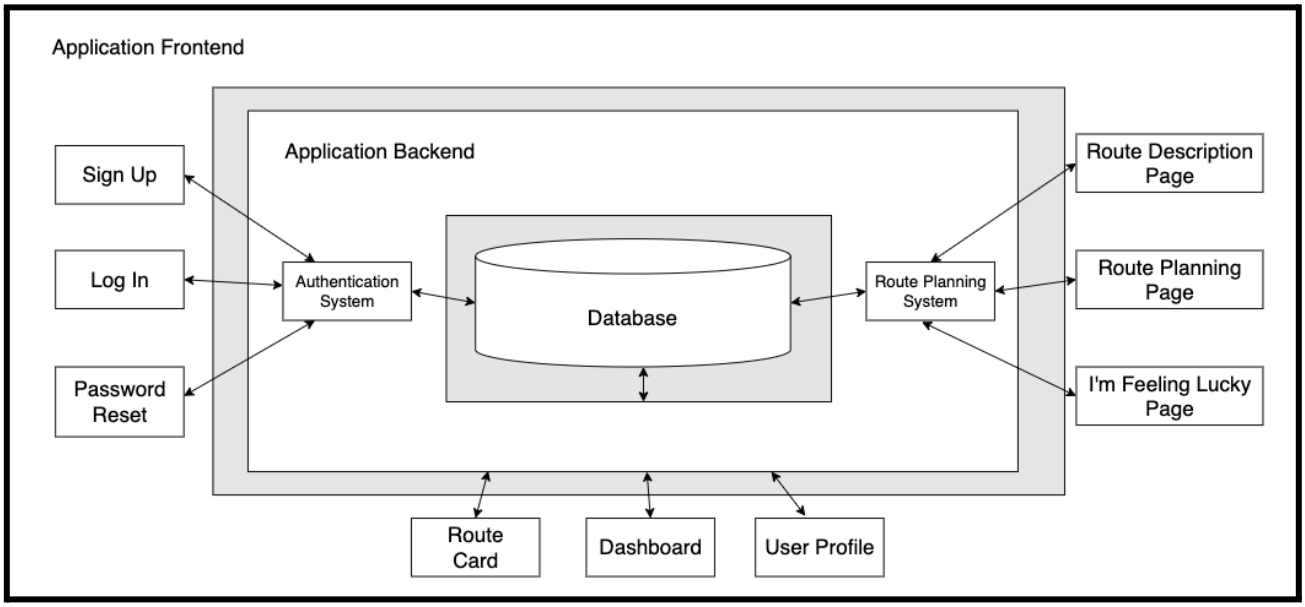


Figure 31: Application Architecture Diagram

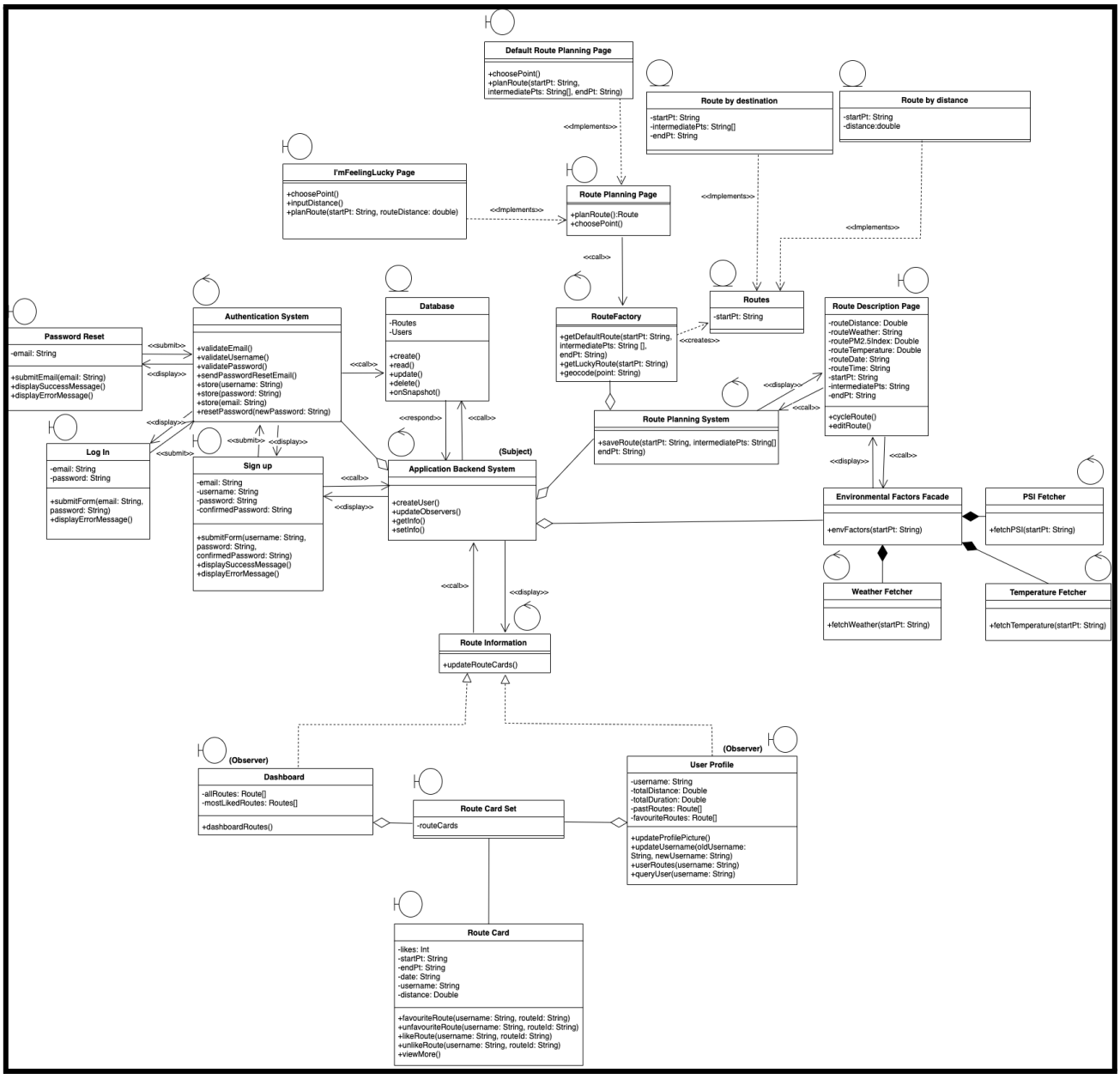


Figure 32: Application Class Diagram

Class	Implementation
Application Backend System	twotyred-backend/server.js
Route Planning System	twotyred-backend/server.js
Route Factory	twotyred-backend/server.js
Environmental Factors Facade	twotyred-backend/server.js
Route Information	twotyred-backend/server.js
Password Reset	twotyred-frontend/src/views/login/ResetCard.jsx
Log In	twotyred-frontend/src/views/login/SignInCard.jsx
Sign Up	twotyred-frontend/src/views/login/RegistrationCard.jsx
Route Card	twotyred-frontend/src/components/RouteCard.jsx
Route Card Set	twotyred-frontend/src/views/dashboard/Dashboard.jsx
Dashboard	twotyred-frontend/src/views/dashboard/Dashboard.jsx
User Profile	twotyred-frontend/src/views/profile/Profile.jsx
Default Route Planning Page	twotyred-frontend/src/views/createroute/RouteSelection.jsx
I'mFeelingLucky Page	twotyred-frontend/src/views/createroute/RouteSelection.jsx
Route Description Page	twotyred-frontend/src/views/createroute//RouteDescription.jsx

Figure 33: Application Class Diagram Implementations

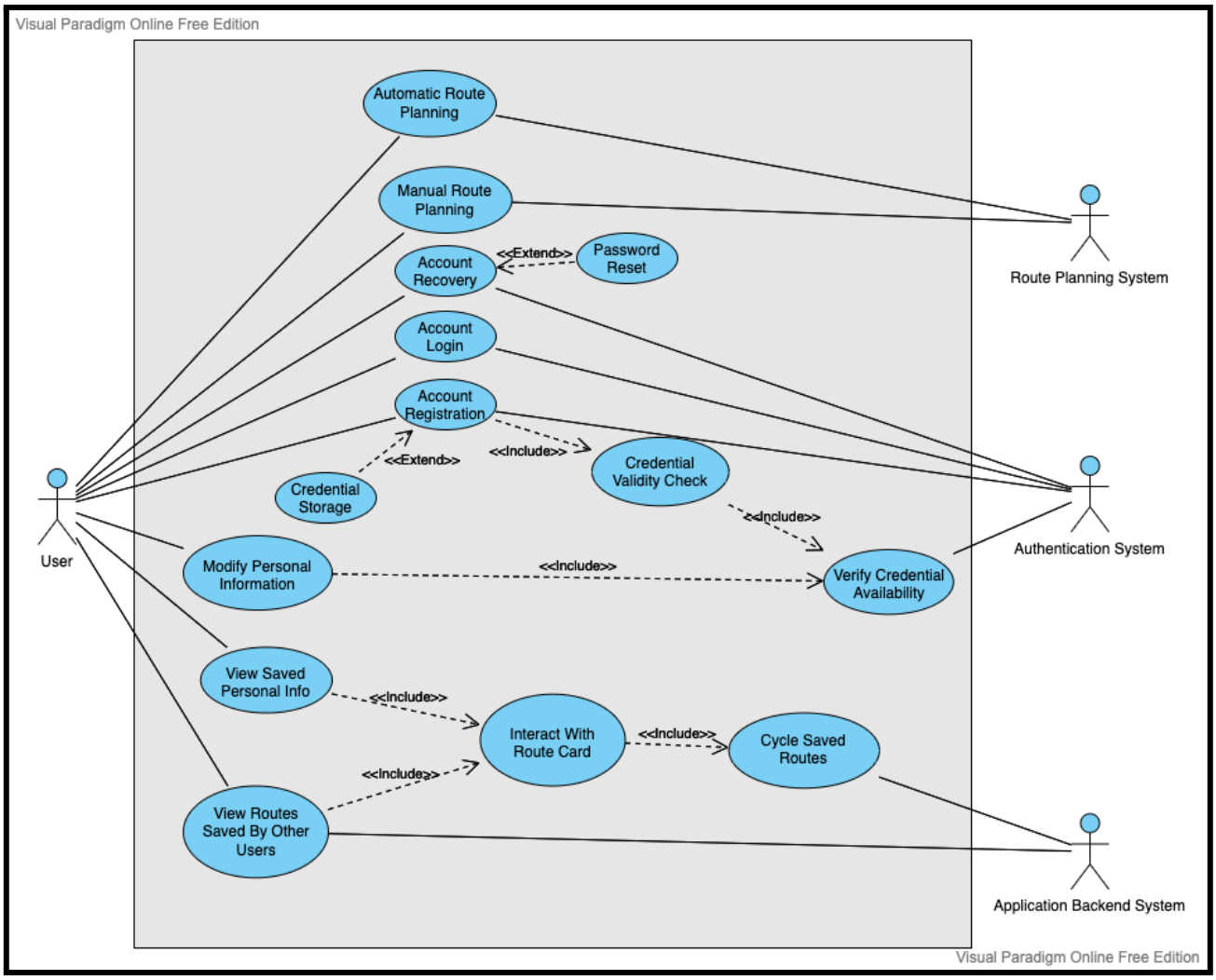


Figure 34: Use Case Model

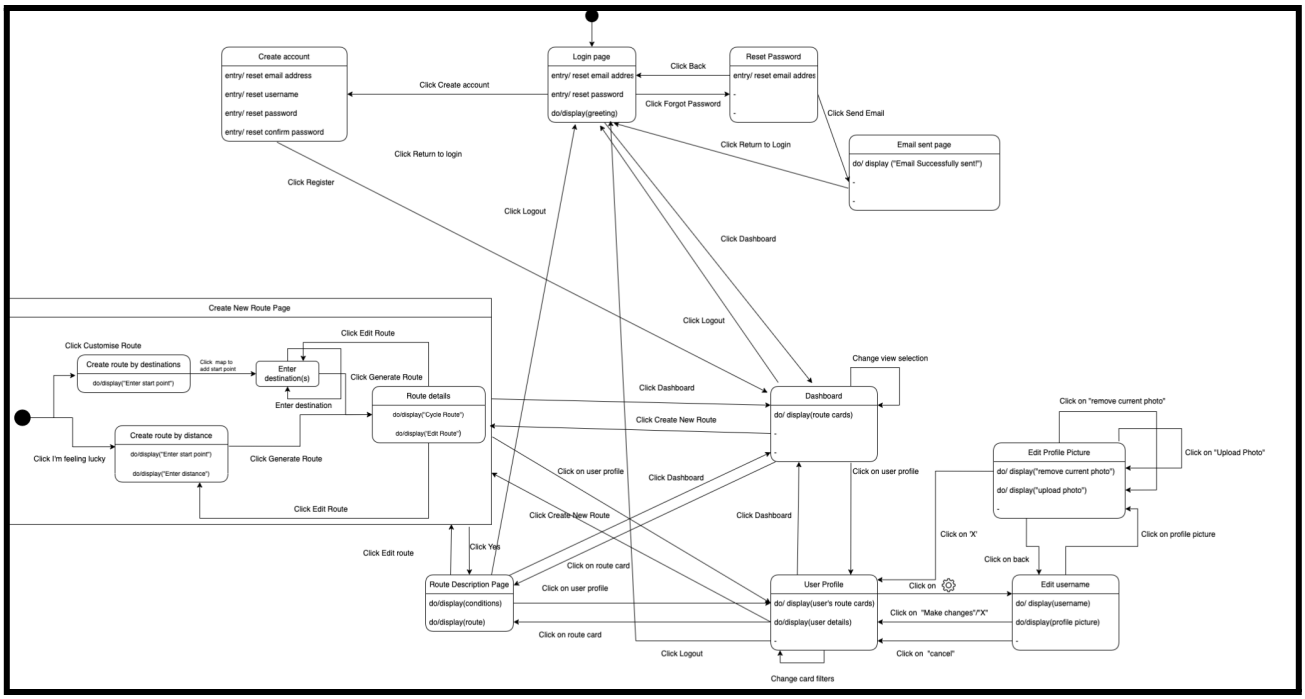


Figure 35: Application Dialog Map

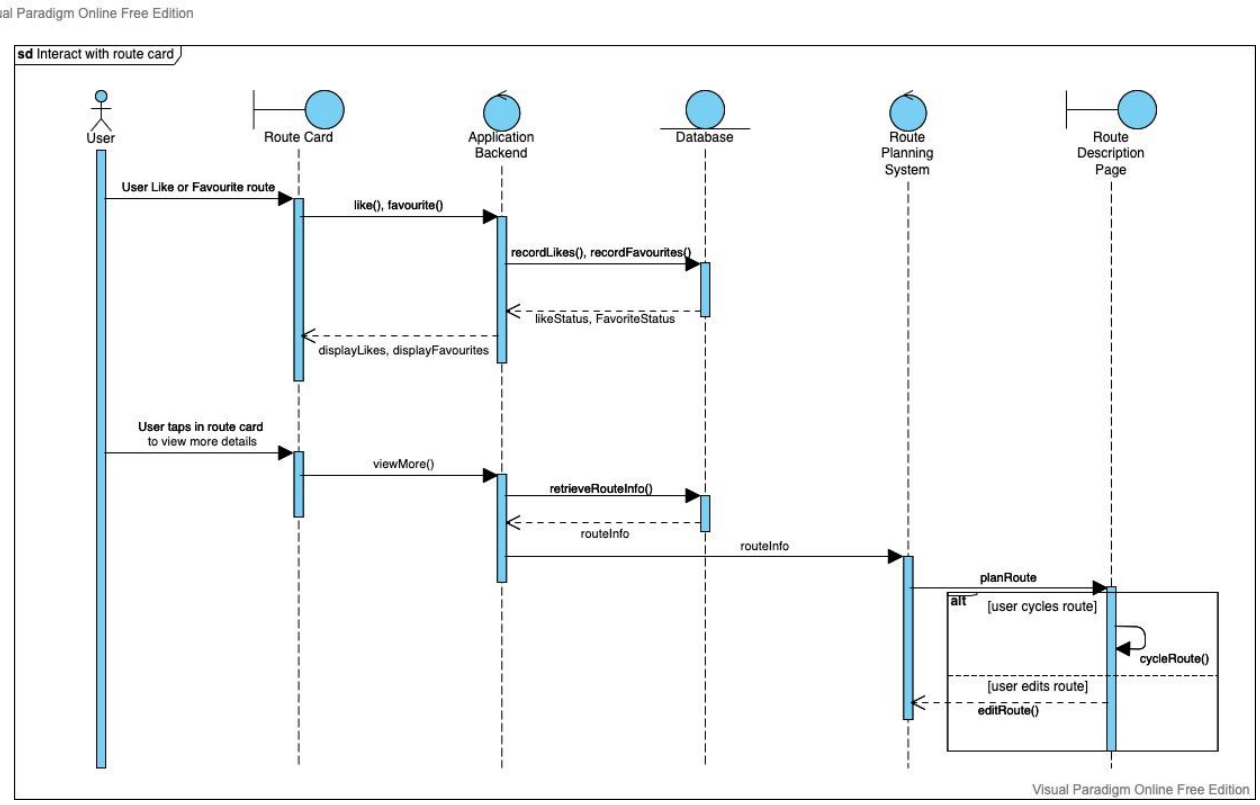


Figure 36: Sequence Diagram for Interacting with route card use case

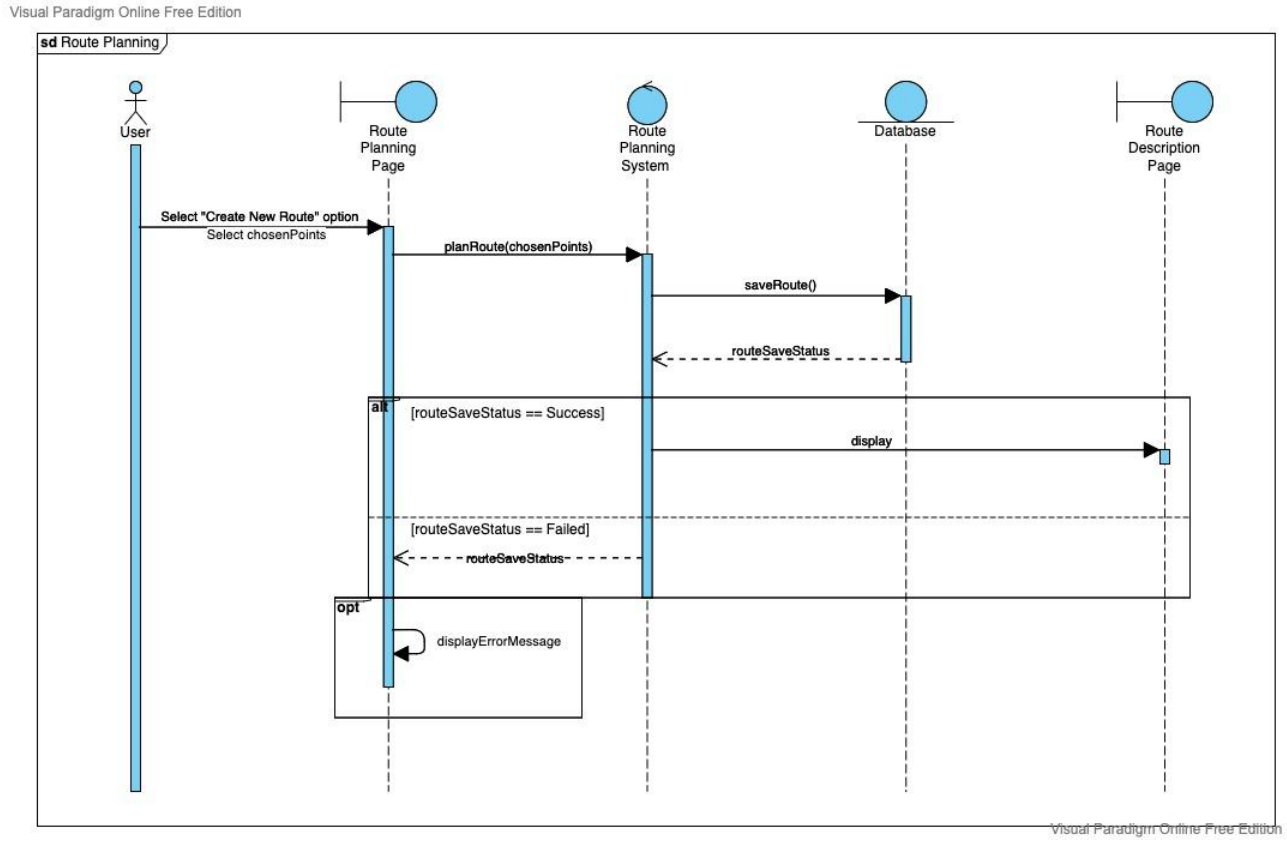


Figure 37: Sequence Diagram for Route planning use case

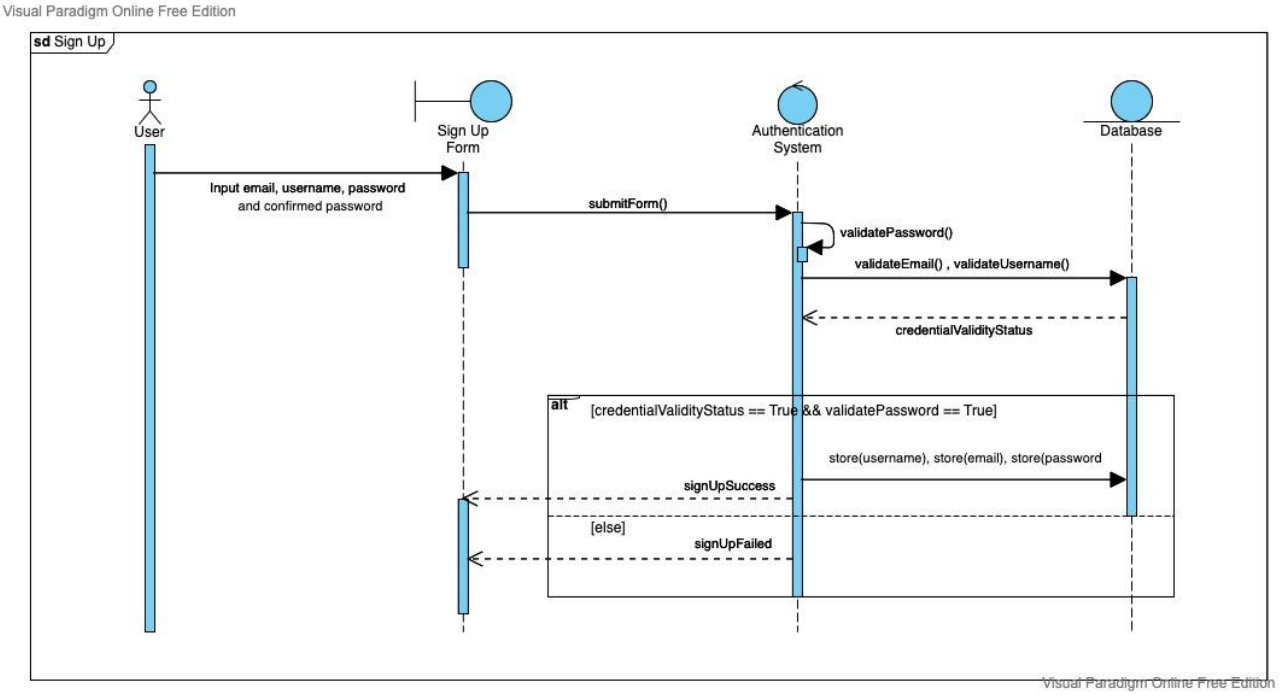


Figure 38: Sequence Diagram for Sign up use case

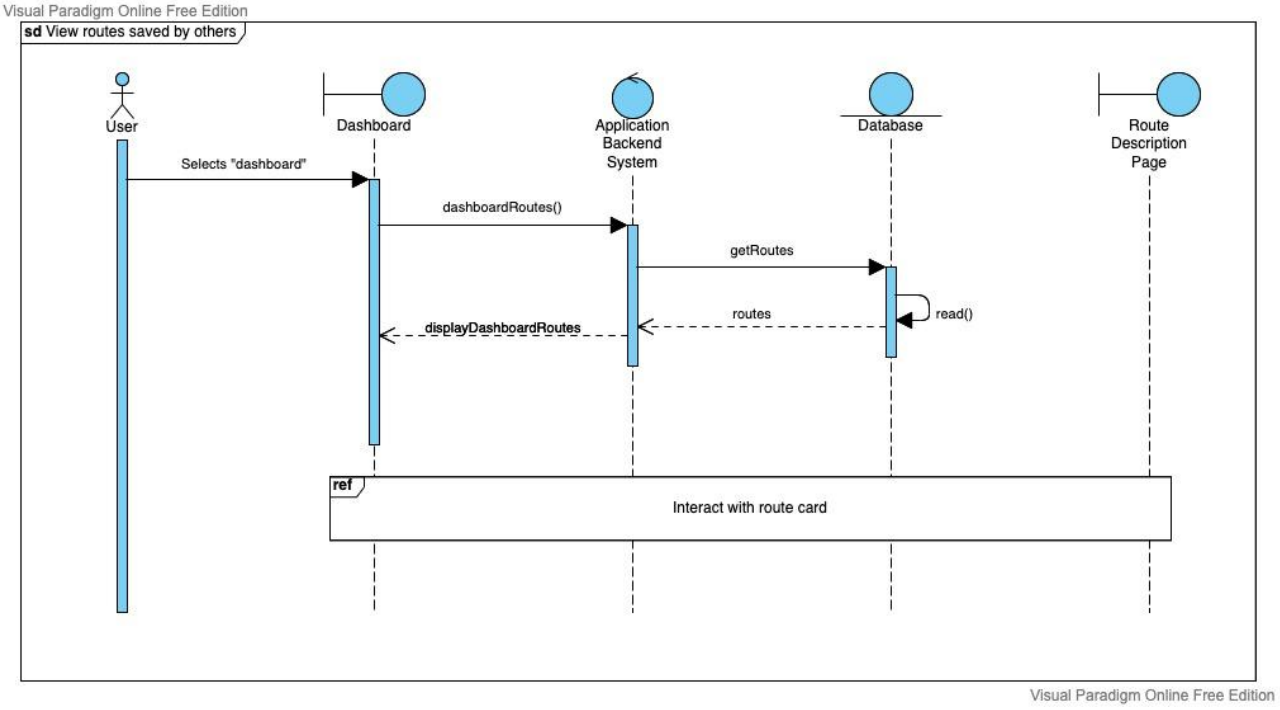


Figure 39: Sequence Diagram for View routes saved by others use case

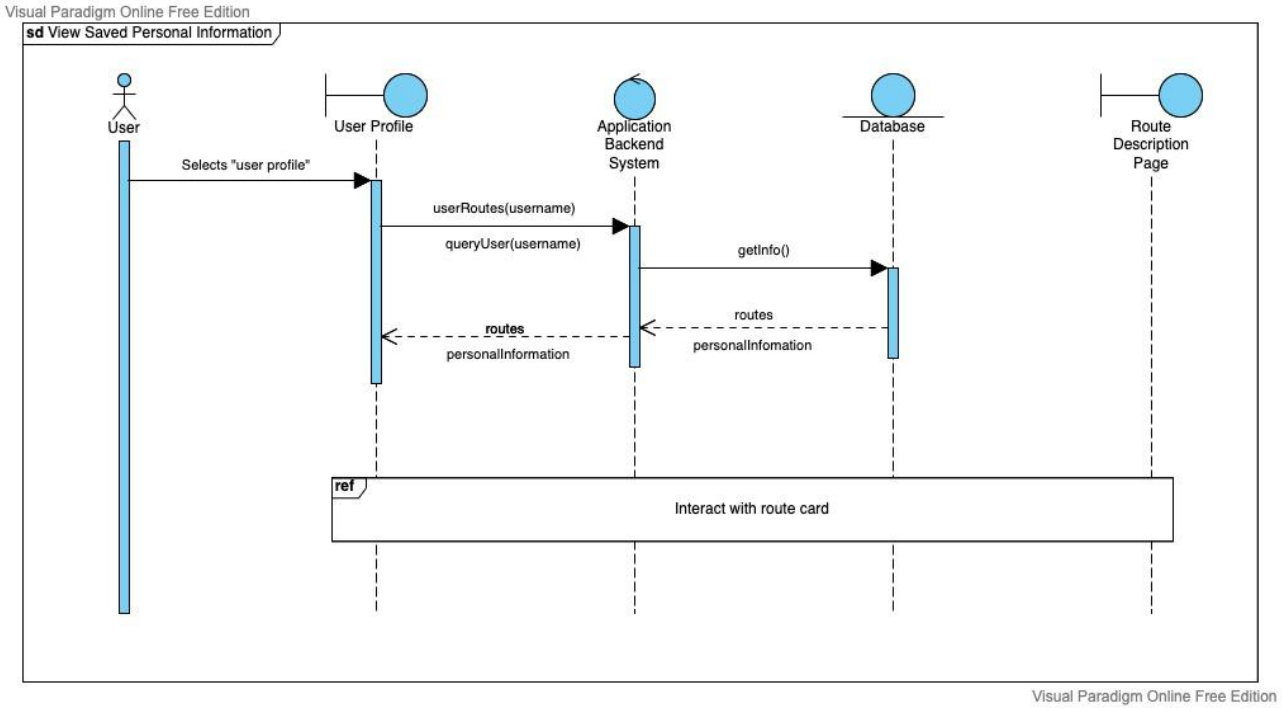


Figure 40: Sequence Diagram for View saved personal information use case

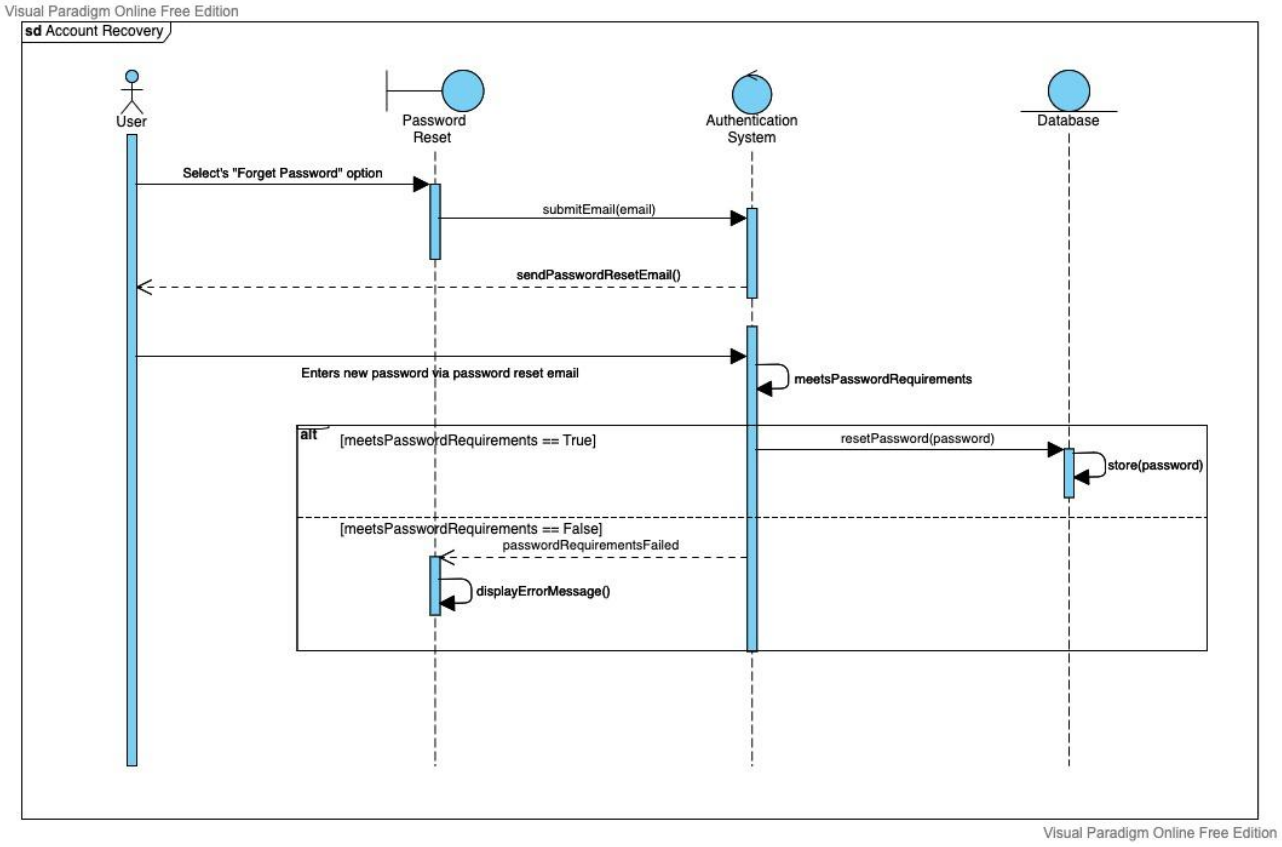


Figure 41: Sequence Diagram for Account recovery use case

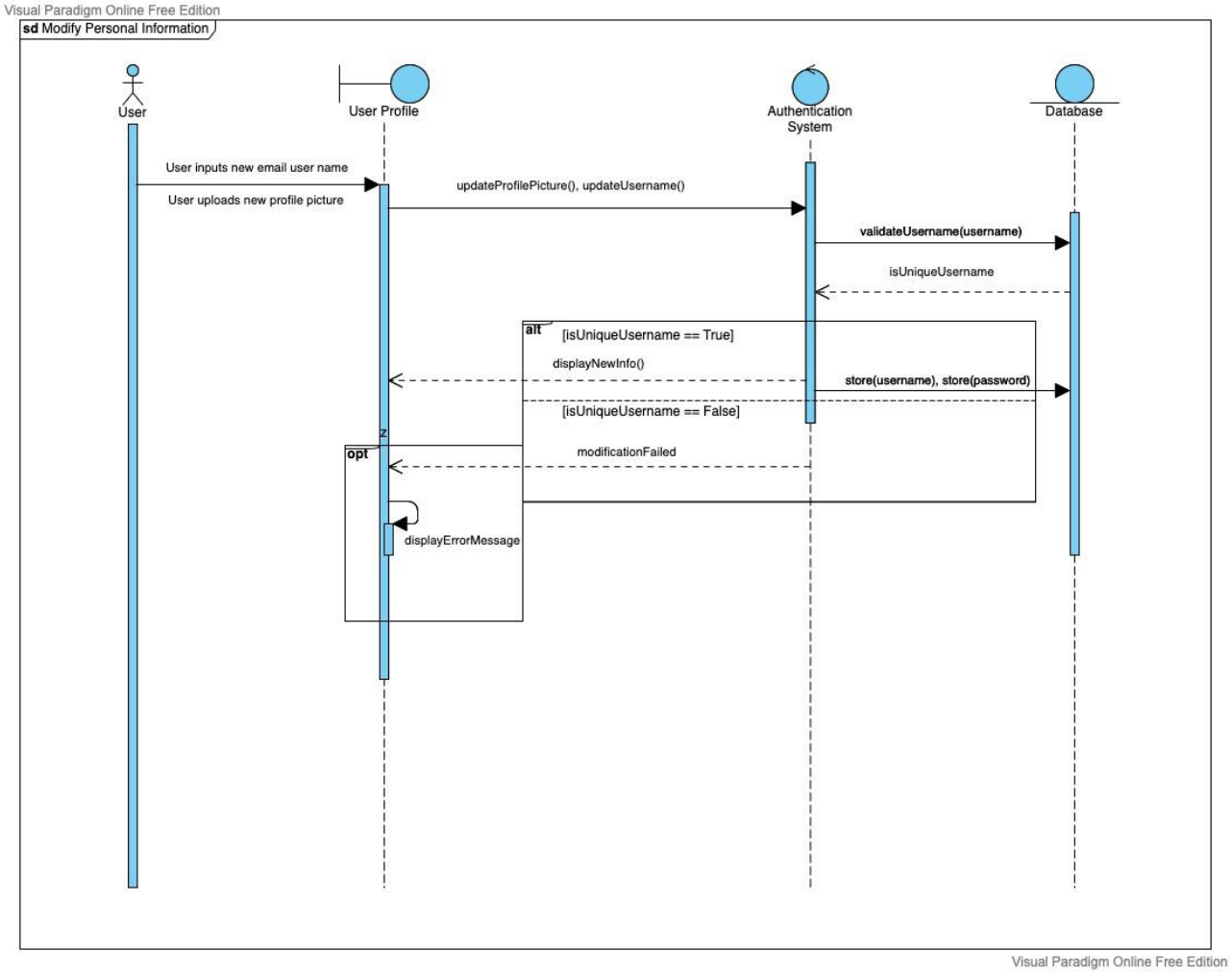


Figure 42: Sequence Diagram for Modify personal information use case