SC2006: Software Engineering

Software Requirements Specification

WanderMap

Ang Yvette (U2220779L)
Chua Sym Eyan (U2221430B)
Tan Hui Ling (U2222280A)
Dan He (N2304782D)

Content Page

1 Introduction

1.1 Purpose

Our team aims to tackle the problems faced by tourists when traveling in Singapore. We will develop a web application for those tourists to create their own itineraries, upload and share their itineraries for a day in Singapore, and browse through other itineraries to build on and review others itineraries. This will help tourists easily find more well-planned itineraries that cater to their preferences.

1.2 Documentation Convention

The following conventions will be used throughout this document

Headings	Arial font size 16 for titles (H2) Arial font size 14 for subtitles (H3)
Content	Arial font size 11
Key terms	Key terms will be bolded to increase readability of document

1.3 Scope

Wander map will offer a range of functions for both itinerary creators and itinerary viewers

For users who have **yet to sign in**

1) **Browse all itineraries**: View all published itineraries, filtering them based on their categories, recency, budget and popularity
2) **View itinerary details:** View the images, attractions, route between attractions and reviews posted for that itinerary

For users who have **signed in**, on top of the two functions above, they will be able to:

1) **Create itineraries:** Plan their own itinerary, picking from the range of attractions available.
2) **Review itineraries:** Comment, rate (on a scale of 1-5), and upload images to review itineraries
3) **Edit itineraries:** Edit the title, attractions, and description of the user's own itineraries
4) **Bookmark itineraries:** Bookmark interesting itineraries to find it easily in bookmarked itineraries
5) **Delete itineraries:** Delete user's own existing itineraries

1.4 Intended audience and Reading suggestions

This document is intended for developers, testers, and users of this product. While reading the full documentation will provide the most comprehensive view of our application, certain sections may be more useful to different audiences.

For **Users,** those who intend to use Wandermap to plan their itinerary in Singapore, section 1 - Introduction will be the most useful for them to make full use of the functionalities in this website.

For **Developers and Testers**, the diagrams, non-functional requirements, architecture design, technology stack and testing may be more important to them.

1.5 References

API used:

1. "Google Maps Maps API Documentation," Google Maps platform documentation | Maps API | google developers. [Online]. Available: https://developers.google.com/maps/documentation/javascript
2. "Google Maps Routes API Documentation," Google Maps platform documentation | routes API | google developers. [Online]. Available: https://developers.google.com/maps/documentation/routes

Documentation used:

1. "Mantine Documentation," Mantine. [Online]. Available: https://mantine.dev/
2. "React Documentation," React. [Online]. Available: https://react.dev/

2 Overall Description

2.1 Product Perspective

Wandermap is a crowdsourced travel itinerary planner offering a wide range of user generated itineraries. It helps make travel in Singapore easy and fuss free with its built in routes generator created through Google Maps API. It is highly customisable, allowing users to pick and plan the itineraries. Users are also able to find the perfect itinerary for them, by narrowing down the itineraries to help them find one that caters to both their interest and budget.

2.2 Product functions

1. **Login User**
   Users will be able to authenticate through Google OAuth by logging in with their google accounts.

2. **Create itineraries**
   The application will allow logged in users to create their own itineraries. Each itinerary will comprise a title, description, and a list of attractions. The application will generate the

budget and category of the itinerary based on the attractions selected. Users can then decide if they want to publish the itinerary or save it as a draft.

3. **Browse Itineraries**
   The application will allow users to view all public itineraries uploaded by other users. Users can filter these itineraries based on the category, budget, popularity or recency of the itineraries. When selected, more details of each itinerary will be displayed, for example the reviews, images, attractions and route between attractions.

4. **Browse User itineraries**
   The application will allow users to view itineraries created by them. Users can choose to edit or delete their saved drafts and delete their published itineraries.

5. **Review itineraries**
   Users will be able to review other itineraries by leaving a comment, rating, or images on the itinerary.

6. **Edit itineraries**
   Users will be able to edit the title, description, and attractions of the itineraries in their saved drafts.

7. **Delete itineraries**
   Users will be able to delete itineraries that they have created and the itinerary will be removed from the database.

8. **Bookmark itineraries**
   Users will be able to bookmark itineraries that interest them. This allows them to refer back to these itineraries easily in through the button “bookmarked itineraries” in the navigation bar.

2.3 Operating Environment

The following describe the software components of our application. Our project involves no hardware components

Presentation layer:
- **NextJS:** Building user interfaces - **CSS:** Styling - **Mantine**: UI library
Application layer:
- **Nextauth.js and Google OAuth:** Secure authentication of users - **Vercel**

- **NPM:** Used for managing dependencies
Persistence layer:
- **Prisma:** used for data modeling and schema syntax
Database layer:
- **PostgreSQL:** Database management system - **Vercel Blob:** Storage for images

2.4 User classes and characteristics

1. **Itinerary creators**
   These are users who create new itineraries from scratch. These itineraries can then be uploaded to the public database to share to other users, or saved privately for future editing. They are required to be logged in to the application. They will be able to use the Create Itinerary feature to help them easily create an itinerary based on categories that they are interested in. Itinerary creators will also have access to all other features such reviewing and bookmarking other itineraries.

2. **Itinerary browsers**
   These are users who only wish browse itineraries uploaded by others to gather Inspiration. They are not required to be logged in to browse itineraries. They will not have access to features like creating, reviewing, and bookmarking itineraries.

2.5 User Documentation

The user interface is intuitive and easy to use. However, for further demonstration, a video will be available to showcase the usage of the application. The video can be found on https://github.com/ey4n/wandermap-final/blob/main/Final%20Submission/Team%20One%20Demo.mp4

2.6 Assumptions and Constraints

Assumptions:
Users have a device that is able to connect to the internet.

Constrains:

1. **Language constraints:**
   The website is only available in English, language location support may be available in future.
2. **Location constraints:**

The attractions are specifically catered to Singapore for now as our team is more familiar with things to do in Singapore, future expansion is possible as the application is easily scalable.

2.7 Dependencies

1. **NextAuth.js**
   NextAuth.js is an authentication solution for Next.js applications. We have chosen to use a built-in provider for NextAuth, specifically Google OAuth, allowing users to log in through their Google accounts.

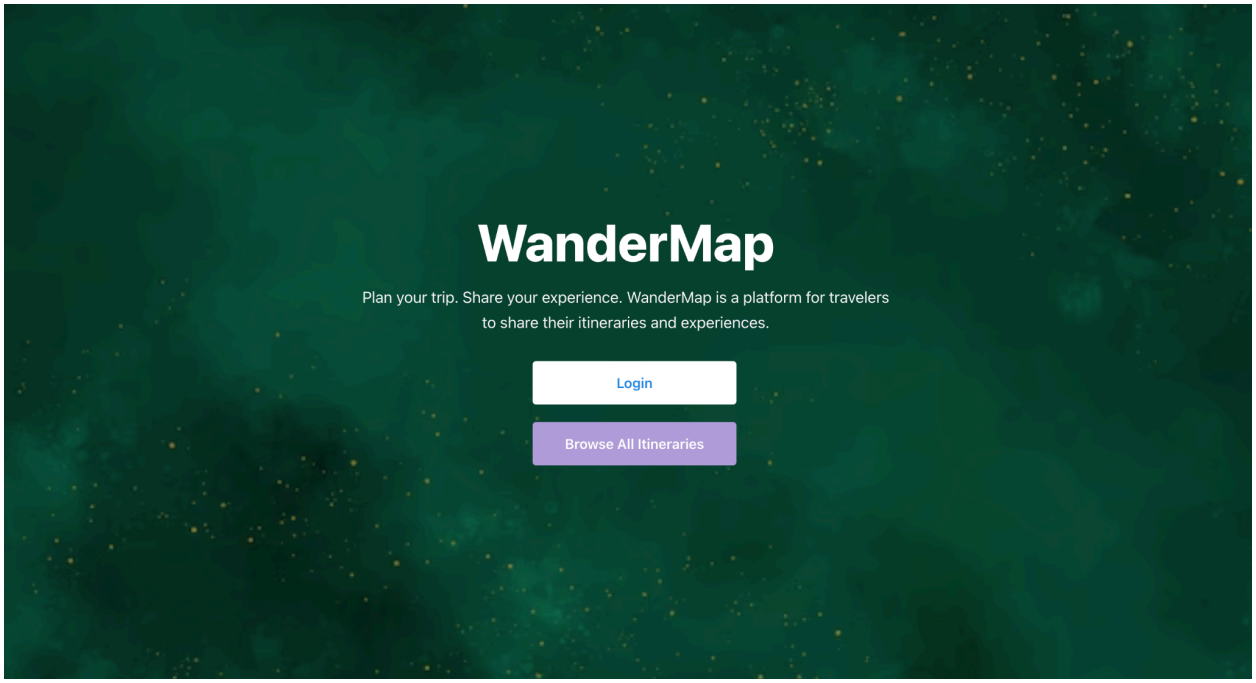
2. **Google Maps API**
   In order to display a map on our route page for an intuitive and convenient overview of the route, we will rely on Google Maps API.

3. **Google Maps Routes API**
   In order to generate a route with descriptive and clear directions for each part of the route, we will rely on Google Maps Routes API. For our application, we have also provided users with the option of different travel modes - by car or by public transport. The Google Maps Routes API will provide us with different routes for different travel modes.

3 External Interface Requirements

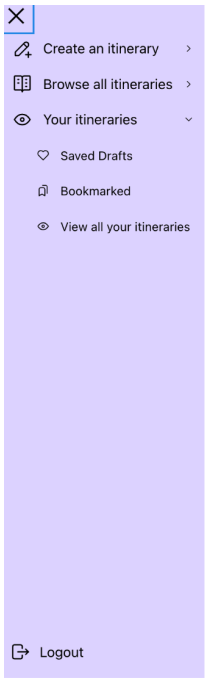
3.1 User interfaces

Login page



When a User first opens Wandermap, they will be brought to this landing page. If the User plans to use all the features of the app, they will have to Login, which prompts them to log in with their Google account. Else, the User can choose to browse all itineraries without logging in to look through the published itineraries.

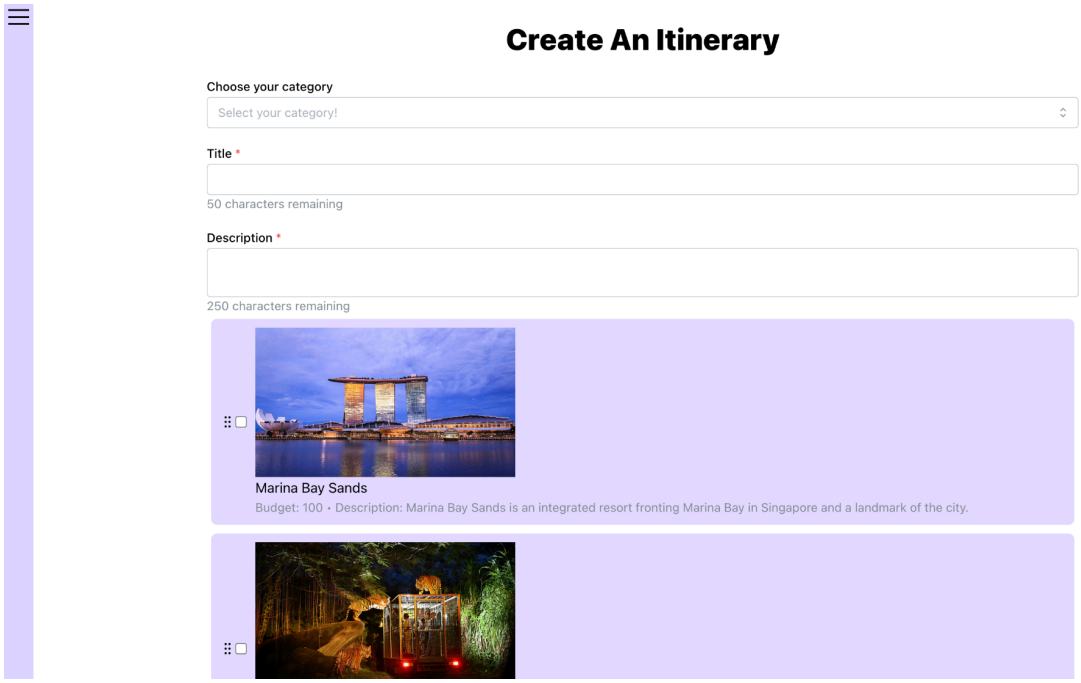
Landing page

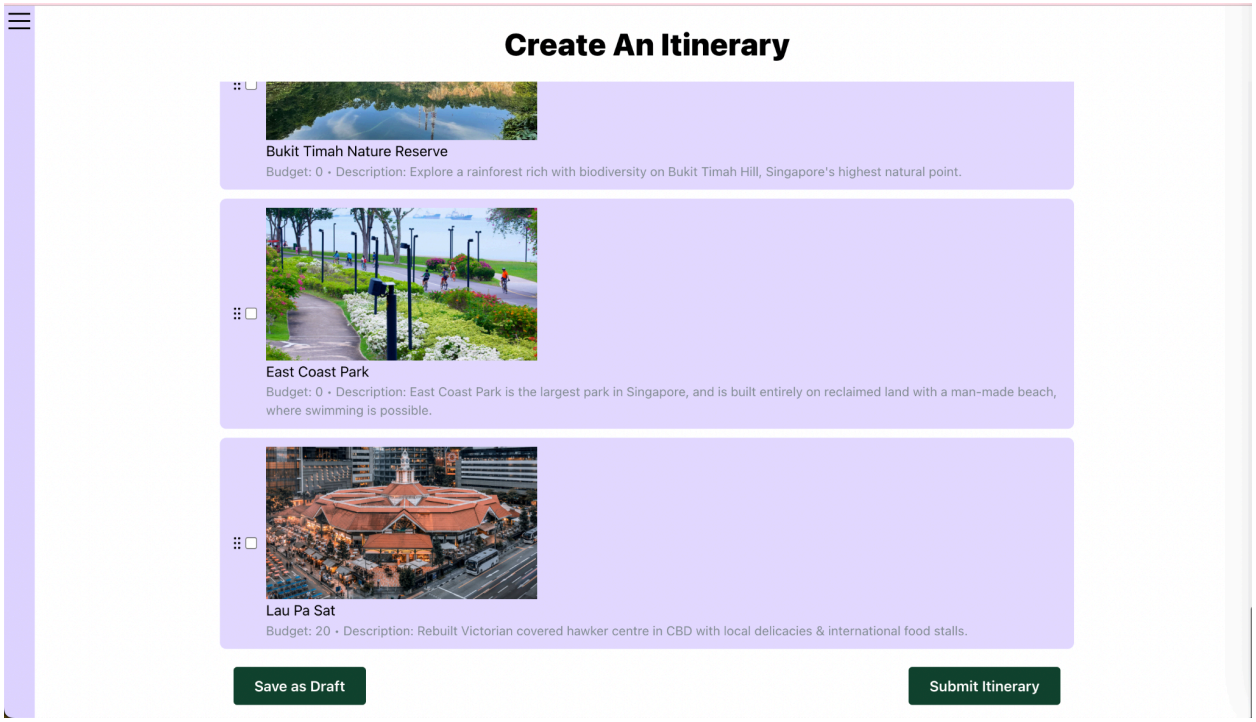


Welcome, Eyan Chua!

If the User has successfully logged in, they will gain access to all these functions in the navigation bar.

Create itinerary page

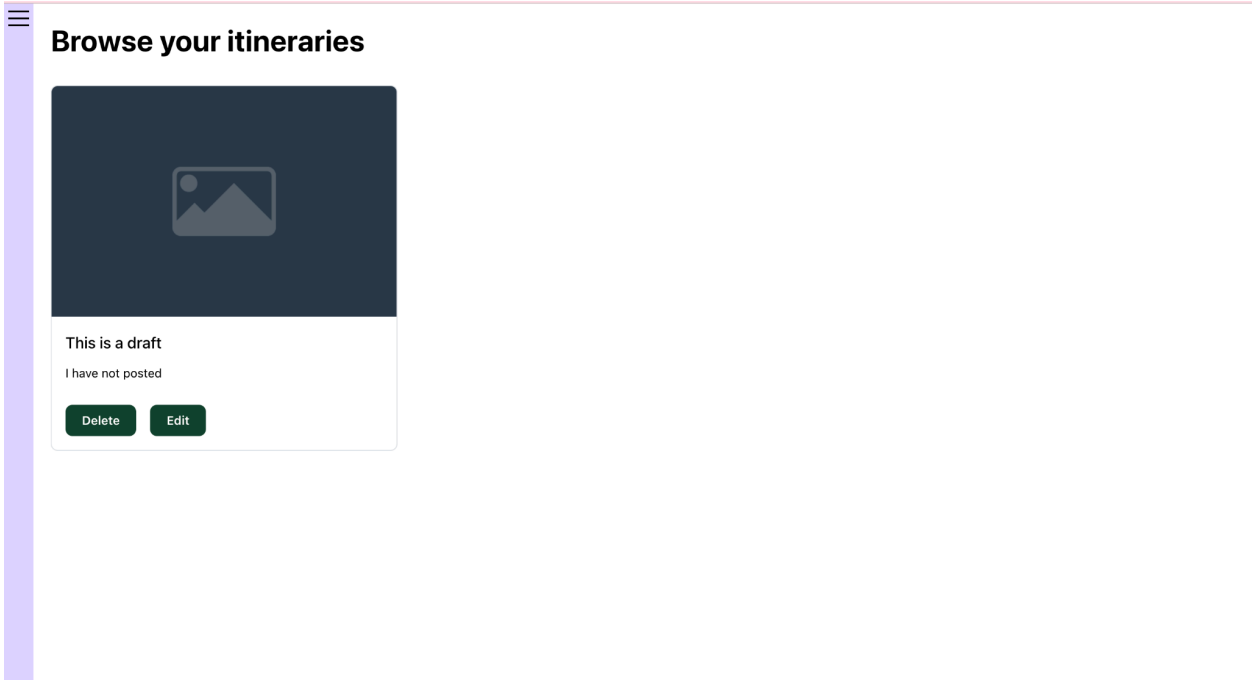


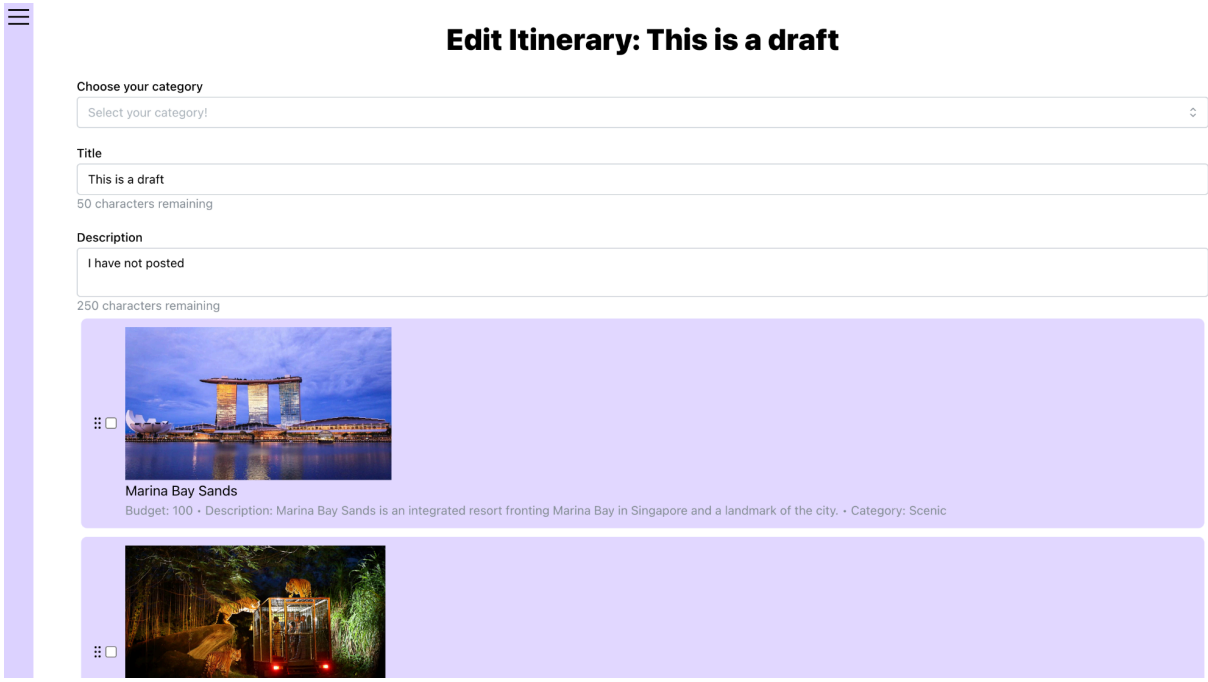


Upon clicking on Create an Itinerary in the navigation bar, the User will be brought to the Create Itinerary page. Here, the User can narrow down the attractions available by selecting one or more categories. The user will then have to fill up the title and description of the itinerary. The user will then use the drag and drop elements to select the attractions they wish to include and

shuffle them into the order they wish to visit them. At the bottom of the page, users are given 2 choices, to save the itinerary as a draft, or to submit itinerary and publish it directly.

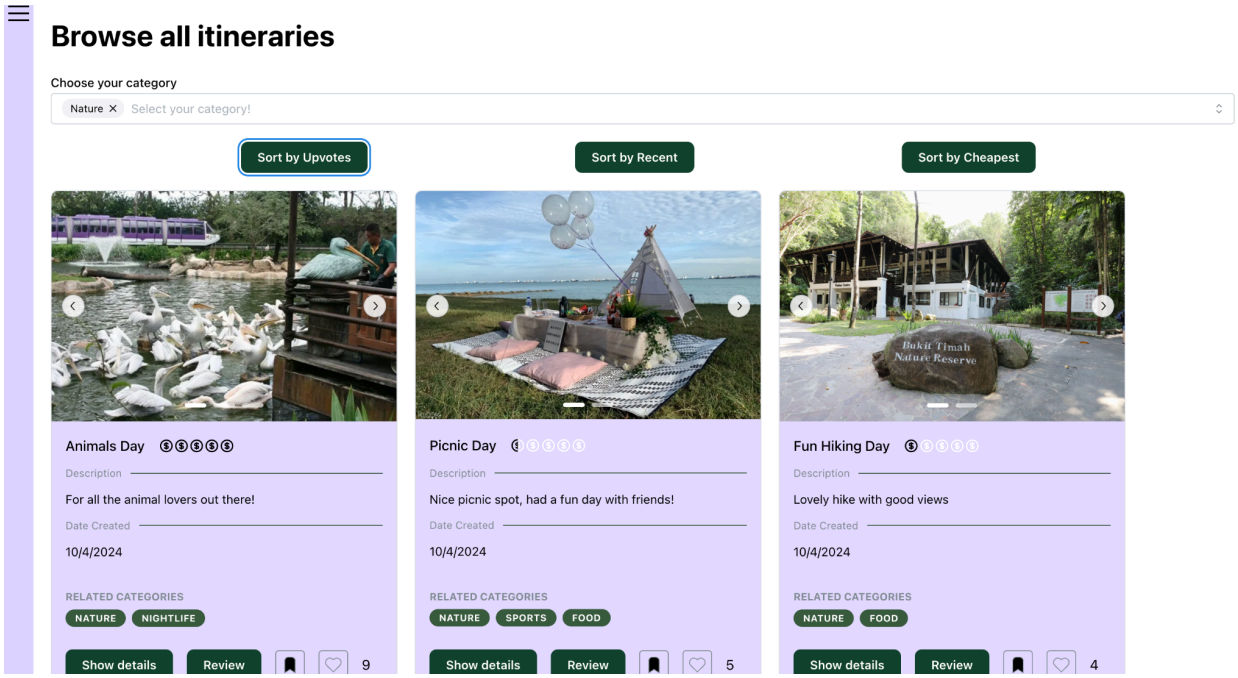
Edit draft





Users are able to edit itineraries in their saved drafts by clicking on the edit button. This will lead them to the Edit Itinerary page where they are able to edit the title, description and attractions.

Browse



Users are able to browse all the published itineraries on the browse page. The filter functions will help them narrow down their searches to help them easily sift through posted itineraries to find one that interests them. Users can select one or more categories to view the itineraries related to that category. Users can also sort the itineraries by popularity (upvotes), recency, and budget.

3.2 Software interfaces

1. **Google Maps Platforms APIs**
   Our application uses Google Maps Platform APIs to find the information users need. We use HTTP requests to communicate with these APIs, which are built on the RESTful architecture. The APIs then send back responses in JSON format, giving us the data in a structured way.

   The 2 Google Maps Platform APIs we require for this application would be Google Maps API and and Google Maps Routes API.

   The Google Maps API enables the application to display maps by communicating with web clients through JavaScript libraries to render maps and handle user interactions. It is used to render a map centered around the 2 attractions that Users would like to view the route between.

   The Google Maps Routes API is used for directions and routing. It takes the 2 attractions that Users have selected to view the route between and returns the ideal route based on the travel mode that the user has selected. These detailed instructions returned by the API will be beneficial to tourists who are unfamiliar with traveling around.

2. **Mantine**
   Mantine Core is used to enhance the functionality and user experience of the website. Complimentary Mantine packages were also used. Mantine Forms was used to manage forms and user input validations. This ensures smooth interactions while creating date ideas. Common reusable logic is abstracted through the use of Mantine Hooks, allowing for implementation of various hooks and customized functionalities that can be reused throughout the application.

3.3 Hardware interfaces

1. The user needs to have a device that can run a **web browser**.
2. The user's device needs to have an **active connection to the Internet** to use the application.
3. The application was designed for use on **computers and tablets.** At the moment, responsive design for mobile viewports was not implemented. As such, our application is not suited for mobile webpages.

3.4 Communications Interfaces

1. **Hypertext Transfer Protocol (HTTP)**
   HTTP is used to communicate between the web client and Google Maps Platform. Both the Google Maps API and Google Maps Routes API require an API key to be passed with the HTTP request. These sensitive **API keys** are stored in our environment variables instead of being hard-coded in the source code. This protects this information

from accidental exposure even when the codebase is shared. This also allows us to easily configure the variables for different environments.

2. **Web Browser**
   Our application has to be viewed on a web browser which supports JavaScript. The application is built using TypeScript, a superset of JavaScript. It compiles to Javascript while allowing type checking by the compiler.

4 System Features

4.1 Login User

Description and Priority

1. All users must be able to log into their account before they are able to plan or post itineraries. This is to provide legitimacy for the itineraries uploaded and protect the quality of posted itineraries.
2. Priority is high as only users are required to perform this step every time they visit the site.

Stimulus/Response Sequences

1. Users are given the option to sign in.
2. The web page redirects the user to Google's O-Auth
3. The user signs in with their existing Google account

Functional Requirements

REQ-1: “Log in” button must be able to redirect from “home” page to “Login” page.
REQ-2: Database must be able to process the login credentials and reply with “success” or “unsuccessful”.
REQ-3: Database must be able to store the session after User has successfully signed in

4.2 Create travel itinerary

Description and Priority

1. User can input the title of the itinerary, description of the itinerary, list of attractions they like to visit as well as the order they like to visit the attractions in if more then one attraction is selected.
2. Priority is medium as not all users might want to create their own itineraries from scratch

Stimulus/Response Sequences

1. User selects the ‘Create an itinerary’ option in the navbar.
2. User inputs the title of the itinerary.
3. User inputs the description of the itinerary.
4. User inputs list of attractions they like to visit.

5. If more than one attraction is chosen, user can select the order they wish to visit the attractions.
6. User can choose to save as draft or submit and share the planned itinerary.

Functional Requirements

REQ-1: Application must be able to check if the User is currently logged in

REQ-2: "Create Itinerary" button must be able to redirect from any page to "Create Itinerary" page.

REQ-3: All input fields must only accept valid data

REQ-4: Database must be able process the title, description, and attractions fields of the input form and reply with "success" or "unsuccessful".

4.3 Get routes

Description and Priority

1. Retrieve the route between two attractions in the travel itinerary on an integrated map feature.
2. Priority is high as tourists will often require instructions from one attraction to the next.

Stimulus/Response Sequences

1. User selects 'View route to next attraction' on an itinerary page.
2. If Google Maps Routes API can find the imputed coordinates and a valid route, it returns the details of route to the system.
3. The system displays the route returned by Google Maps Routes API on the Integrated Map.
4. The system displays instructions on how to follow the route step by step.

Functional Requirements

REQ-1: "View Route" button must be able to redirect them to the "View Route" page

REQ-2: Google Maps API must render the map widget on the "View Route" page.

REQ-3: Google Maps Route API must return the correct instructions for the directions between the selected attractions

4.4 Publish travel itinerary

Description and Priority

1. Users can upload their travel itineraries onto the website for other users to view and interact with. The travel itinerary includes the following information: title of the itinerary, description of the itinerary, list of attractions they like to visit as well as the order they like to visit the attractions in if more then one attraction is selected.
2. Priority is medium as not all users will want to publish their itineraries

Stimulus/Response Sequences

1. User has a filled in itinerary from the create itinerary of edit itinerary page
2. The user selects to upload the travel itinerary.
3. The system processes the information.

4. The system uploads the travel itinerary and adds it to its database.
5. The website displays a confirmation message to the user.

Functional Requirements

REQ-1: “Publish itinerary” button must be able to redirect them to the “Review itinerary” page as Users have to review the itinerary before successfully posting it.

REQ-2: Database has to update the published status of the itinerary and reply with “success” or “unsuccessful”.

4.5 Edit draft itinerary

Description and Priority

1. Users can edit draft itineraries by editing the title, attractions and description.
2. Priority is medium as only itinerary creators have to use this function.

Stimulus/Response Sequences

1. User selects an itinerary that they have created and saved but not yet uploaded.
2. User now has the option to edit the itinerary.
3. The user can choose to make amendments to the itinerary through adding attractions, deleting attractions, changing the order of attractions, or editing the title and description of the itinerary.
4. User can choose to save draft or submit and share the planned itinerary.

Functional Requirements

REQ-1: “Edit itinerary” Button has to redirect them to “Edit itinerary” page.

REQ-2: All input fields must only accept valid data

REQ-4: Database must be able process the title, description, and attractions fields of the input form and update the edited itinerary with these fields and reply with “success” or “unsuccessful”.

4.6 Delete itinerary

Description and Priority

1. Users can delete their itineraries.
2. Priority is medium as only itinerary creators have to use this function.

Stimulus/Response Sequences

1. User selects an itinerary that they have created.
2. User now has the option to delete the itinerary.
3. The website sends a confirmation message to delete the itinerary.
4. The website then processes the delete request.
5. The website displays a success message if the request has been processed.

Functional Requirements

REQ-1: “Delete itinerary” Button has send a confirmation form before deleting the itinerary

REQ-2: “No” Button on the confirmation form should redirect the user back to the previous page

REQ-3: “Yes” button on the confirmation form should delete the itinerary

REQ-4: Database must delete the itinerary and reply with “successful” or “unsuccessful”.

4.7 Browse uploaded travel itineraries

Description and Priority

1. User can browse travel itineraries uploaded by others. User can filter itineraries according to preset categories (Nightlife, Nature, Scenice, Sport, Food, Culture), upvotes, most recent upload, and cheapest itinerary.
2. Priority is high as most Users would want to browse through itineraries.

Stimulus/Response Sequences

1. User selects the "Browse all itineraries" option.
2. User applies filters in the form of categories (Nightlife, Nature, Scenice, Sport, Food, Culture), upvotes, most recent upload, and cheapest itinerary to narrow down the list of itinerary ideas.
3. The website retrieves and displays itinerary ideas based on the applied categories

Functional Requirements

REQ-1: “Browse all itineraries” button must redirect User from any page to “Browse” page
REQ-2: Filter buttons must be able to sort the itineraries correctly

4.8 Review travel itinerary

Description and Priority

1. Users can review others' itineraries by uploading a comment, rating, and images
2. Priority is medium as not all users will be willing to leave reviews

Stimulus/Response Sequences

1. User selects an itinerary that they see when browsing itineraries.
2. User now has the option to upload a text review, a rating and a photo.
3. The website processes the review data and stores it in the database under the itinerary.
4. User receives a success message if the review has been posted.

Functional Requirements

REQ-1: Application must be able to check if the User is currently logged in
REQ-2: “Review Itinerary” button must be able to redirect User to “Review Itinerary” page.
REQ-3: All input fields must only accept valid data
REQ-4: Database must be able process the comment, rating and image fields of the input form and reply with “success” or “unsuccessful”.

4.9 View Itinerary

Description and Priority

1. Users can click into each itinerary to view the title, description, budget, attractions, routes and images of the itinerary.
2. Priority is high as most Users will be browsing other Users itineraries and would want to view details of these itineraries.

Stimulus/Response Sequences

1. User selects ‘Show details’ of an itinerary they are interested in.
2. The website retrieves itinerary information of the selected itinerary
3. The website displays the information to the user

Functional Requirements

REQ-1: “View Details” button must redirect User to the page with the itinerary details.
REQ-2: Website must be able to display the images, description, reviews, budget and attractions for this itinerary

4.10 Save draft itinerary

Description and Priority

1. Users can save their itinerary as a draft after creating it.
2. Priority is medium as not all Users will create itineraries.

Stimulus/Response Sequences

1. User has a filled in itinerary from the create itinerary of edit itinerary page
2. The user selects to save the travel itinerary.
3. The system processes the information.
4. The system uploads the travel itinerary and adds it to its database.
5. The website displays a confirmation message to the user.

Functional Requirements

REQ-1: Database must be able to update the published status of the itinerary to false and reply with “success” or “unsuccessful”.

5 Non-Functional Requirements

1. **Usability**
   - 1.1. A new user should be able to complete the account authentication and creation processes within 5 minutes of entering the web application.
   - 1.2. The system should provide error messages with < 25 words for invalid email formats and passwords.
   - 1.3. A new user should be able to create and upload their first itinerary within 10 minutes of entering the web application.
   - 1.4. A new user should be able to find and use the search functions, including the filter function, within 5 minutes of entering the web application.

2. **Reliability**
   - 2.1. The system shall not have planned downtime of more than 1 hour per year.

3. **Performance**
   - 3.1. The system must respond to user authentication requests within 5 seconds.
   - 3.2. The system must generate travel itineraries, including routes, within 10 seconds.
   - 3.3. The system should return results for the search functions within 5 seconds.
   - 3.4. The system should be able to handle a minimum of 1000 simultaneous user sessions without significant degradation in performance.

4. **Supportability**
   - 4.1. The code should be well organised with relevant documentation, to allow for easy readability and maintenance.

5. **Scalability**
   - 5.1. The system must be able to accommodate a 20% annual increase in user interactions.

6. **Security**
   - 6.1. User data must be securely stored using encryption techniques such as AES-256.

6.2. The system should implement measures to prevent unauthorised access, with account lockouts after 5 failed login attempts.

6.3. User will be automatically logged out after 30 minutes of inactivity.

6 Appendix

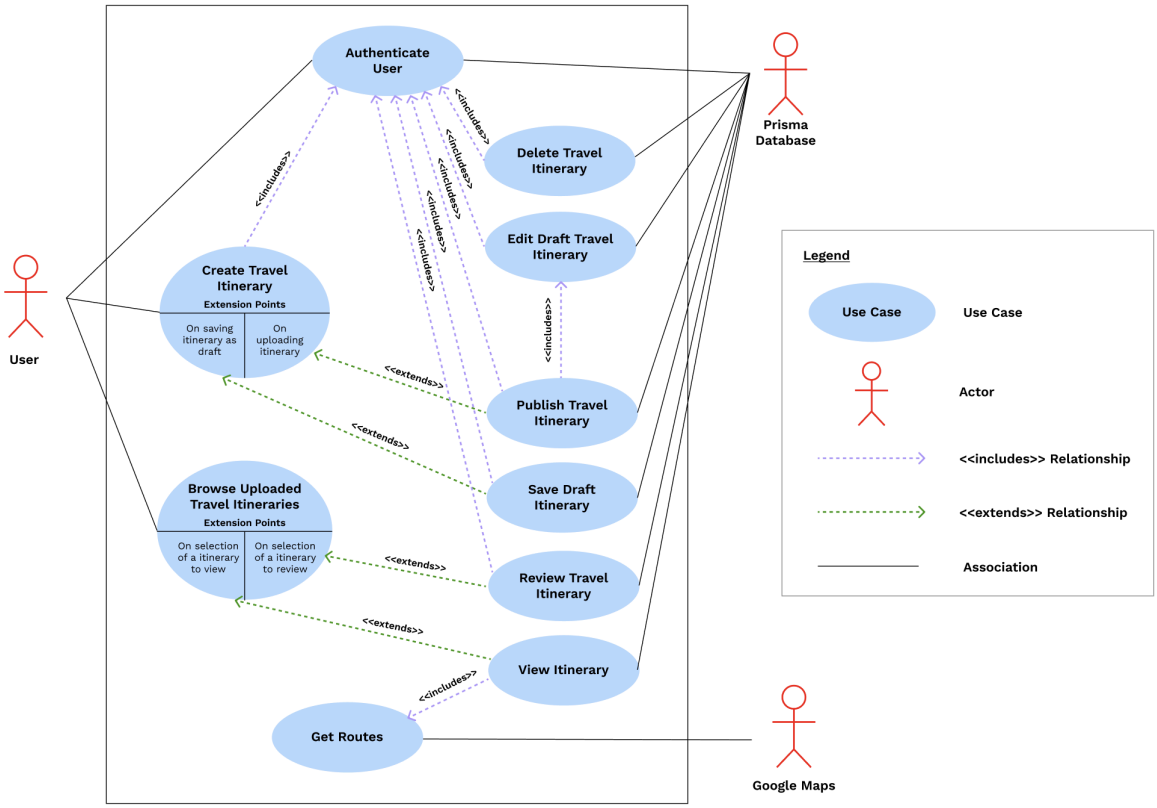
Appendix A: Data Dictionary

Terms	Definition
Itinerary	A planned schedule of the attractions that the user has selected, that outlines the location and timing for each activity.
Category	Grouping of the attraction based on their shared characteristics. For example, outdoor activities, scenic activities, etc.
Plan	Based on the activities that the user has selected, the app comes up with the sequence and timing for each activity.
Attraction	A point of interest uploaded to the web application that is contains information of the name of location, address, and category of activity.
Upvote	A function of the web page that allows users to cast a vote in support of a given route by pressing a button, which in turn would affect how the route is recommended to other users.

Appendix B: Analysis Models

Use Case Model

Use Case Diagram



Use Case Description

Use Case ID:	001		
Use Case Name:	Authenticate user		
Created By:	Eyan Chua	Last Updated By:	Eyan Chua
Date Created:	01/02/2024	Date Last Updated:	01/02/2024

Actor:	Users, Prisma Database
Description	All users must be able to log into their account before they are able to plan or post itineraries. This is to provide legitimacy for the itineraries uploaded and protect the quality of posted itineraries.
Precondition:	User has not been authenticated.
Postcondition:	User has successfully signed in.
Priority:	High
Frequency of Use:	Frequent
Flow of Events:	1. Users are given the option to sign in. 2. The web page redirects the user to Google's O-Auth 3. The user signs in with their existing Google account
Alternative Flows:	AF - S3 If no valid Google account is found Google O-Auth will prompt users to create a Google account. The system returns to S3.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and issues:	-

Use Case ID:	002		
Use Case Name:	Create travel itinerary		
Created By:	Ang Yvette	Last Updated By:	Ang Yvette
Date Created:	01/02/2024	Date Last Updated:	01/02/2024

Actor:	User
Description:	User can input the title of the itinerary, description of the itinerary, list of attractions they like to visit as well as the order they like to visit the attractions in if more then one attraction is selected.
Precondition:	User has the website opened and an account created.
Postcondition:	User has a fully created itinerary with all forms filled in.
Priority:	Medium
Frequency of Use:	Frequent
Flow of Events:	1. User selects the ‘Create an itinerary’ option in the navbar. 2. User inputs the title of the itinerary. 3. User inputs the description of the itinerary. 4. User inputs list of attractions they like to visit. 5. If more than one attraction is chosen, user can select the order they wish to visit the attractions. 6. User can choose to save as draft or submit and share the planned itinerary.
Alternative Flows:	AF - S6 If the ‘title’ field is empty, an error message is displayed prompting users to add a ‘title’. System returns to step 2. If the ‘description’ field is empty, an error message is displayed prompting users to add a ‘description’. System returns to step 3.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and issues:	-

Use Case ID:	003		
Use Case Name:	Get routes		
Created By:	Tan Hui Ling	Last Updated By:	Tan Hui Ling
Date Created:	01/02/2024	Date Last Updated:	01/02/2024

Actor:	Google Maps
Description:	Retrieve the route between two attractions in the travel itinerary on an integrated map feature.
Precondition:	User has inputted a minimum of 2 attractions into their travel itinerary.
Postcondition:	System successfully displays the route on the integrated map feature to the user.
Priority:	High
Frequency of Use:	Frequent
Flow of Events:	1. User selects ‘View route to next attraction’ on an itinerary page. 2. If Google Maps Routes API can find the imputed coordinates and a valid route, it returns the details of route to the system. 3. The system displays the route returned by Google Maps Routes API on the Integrated Map. 4. The system displays instructions on how to follow the route step by step.
Alternative Flows:	AF - S3 If Google Maps Routes API cannot find a valid route between attractions, the system will return an error message. The system will return to step 1.
Exceptions:	-
Includes:	-
Special Requirements:	Google Maps Location API
Assumptions:	-
Notes and issues:	-

Use Case ID:	004		
Use Case Name:	Publish travel itinerary		
Created By:	Tan Hui Ling	Last Updated By:	Tan Hui Ling
Date Created:	01/02/2024	Date Last Updated:	01/02/2024

Actor:	User, Prisma Database
Description:	Users can upload their travel itineraries onto the website for other users to view and interact with. The travel itinerary includes the following information: title of the itinerary, description of the itinerary, list of attractions they like to visit as well as the order they like to visit the attractions in if more then one attraction is selected.
Precondition:	User is logged into the system
Postcondition:	Travel itinerary is successfully uploaded onto the website
Priority:	High
Frequency of Use:	Frequent
Flow of Events:	1. User has a filled in itinerary from the create itinerary of edit itinerary page 2. The user selects to upload the travel itinerary. 3. The system processes the information. 4. The system uploads the travel itinerary and adds it to its database. 5. The website displays a confirmation message to the user.
Alternative Flows:	AF - S4 If the system is unsuccessful in uploading the travel itinerary, the website will return an error message and prompt the user to try again. The system returns to step 1.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and issues:	-

Use Case ID:	005		
Use Case Name:	Edit draft travel itinerary		
Created By:	Eyan Chua	Last Updated By:	Eyan Chua
Date Created:	01/02/2024	Date Last Updated:	01/02/2024

Actor:	Users, Prisma Database
Description:	User can edit draft itineraries.
Precondition:	User is successfully logged in and has previously uploaded draft itineraries.
Postcondition:	Itinerary is successfully edited.
Priority:	Medium
Frequency of Use:	Seldom
Flow of Events:	1. User selects an itinerary that they have created and saved but not yet uploaded. 2. User now has the option to edit the itinerary. 3. The user can choose to make amendments to the itinerary through adding attractions, deleting attractions, changing the order of attractions, or editing the title and description of the itinerary. 4. User can choose to save draft or submit and share the planned itinerary.
Alternative Flows:	AF - S3 If the ‘title’ field is empty, an error message is displayed prompting users to add a ‘title’. System returns to step 2. If the ‘description’ field is empty, an error message is displayed prompting users to add a ‘description’. System returns to step 3.
Exceptions:	-
Includes:	1. Authentication of user as a user can only edit their own itineraries.
Special Requirements:	-
Assumptions:	-
Notes and issues:	-

Use Case ID:	006		
Use Case Name:	Delete travel itinerary		
Created By:	Eyan Chua	Last Updated By:	Eyan Chua
Date Created:	01/02/2024	Date Last Updated:	01/02/2024

Actor:	Users, Prisma Database
Description:	User can delete their itineraries.
Precondition:	User is successfully logged in and has previously uploaded itineraries.
Postcondition:	Itinerary is successfully deleted.
Priority:	Medium
Frequency of Use:	Seldom
Flow of Events:	1. User selects an itinerary that they have created. 2. User now has the option to delete the itinerary. 3. The website sends a confirmation message to delete the itinerary. 4. The website then processes the delete request. 5. The website displays a success message if the request has been processed.
Alternative Flows:	AF - S4 If the website is unable to process the request, an error message will be displayed and the website returns back to step 2.
Exceptions:	-
Includes:	1. Authentication of user as a user can only delete their own itineraries.
Special Requirements:	-
Assumptions:	-
Notes and issues:	-

Use Case ID:	007		
Use Case Name:	Browse uploaded travel itineraries		
Created By:	Ang Yvette	Last Updated By:	Ang Yvette
Date Created:	01/02/2024	Date Last Updated:	01/02/2024

Actor:	User
Description:	User can browse travel itineraries uploaded by others. User can filter itineraries according to preset categories (Nightlife, Nature, Scenice, Sport, Food, Culture), upvotes, most recent upload, and cheapest itinerary.
Precondition:	User has a registered account.
Postcondition:	User views a list of itineraries based on applied categories and views further information on the interested travel itinerary.
Priority:	High
Frequency of Use:	Frequent
Flow of Events:	1. User selects the "Browse all itineraries" option. 2. User applies filters in the form of categories (Nightlife, Nature, Scenice, Sport, Food, Culture), upvotes, most recent upload, and cheapest itinerary to narrow down the list of itinerary ideas. 3. The website retrieves and displays itinerary ideas based on the applied categories.
Alternative Flows:	AF - S3 If no itinerary ideas match the applied filters, the website displays ‘No itinerary found’. Website returns to Step 2.
Exceptions:	-
Includes:	1. Get shared itinerary from other users
Special Requirements:	-
Assumptions:	-
Notes and issues:	-

Use Case ID:	008		
Use Case Name:	Review travel itinerary		
Created By:	Eyan Chua	Last Updated By:	Eyan Chua
Date Created:	13/02/2024	Date Last Updated:	13/02/2024

Actor:	Users, Prisma Database
Description:	Users can review others' itineraries.
Precondition:	User is successfully logged in.
Postcondition:	Review is successfully posted and tagged to the itinerary.
Priority:	Medium
Frequency of Use:	Seldom
Flow of Events:	1. User selects an itinerary that they see when browsing itineraries. 2. User now has the option to upload a text review, a rating and a photo. 3. The website processes the review data and stores it in the database under the itinerary. 4. User receives a success message if the review has been posted.
Alternative Flows:	AF - S3 If the website is unable to process the review (e.g. wrong file type), an error message will be displayed and the website returns back to step 2.
Exceptions:	-
Includes:	Authentication of users as only registered users can upload reviews.
Special Requirements:	-
Assumptions:	-
Notes and issues:	-

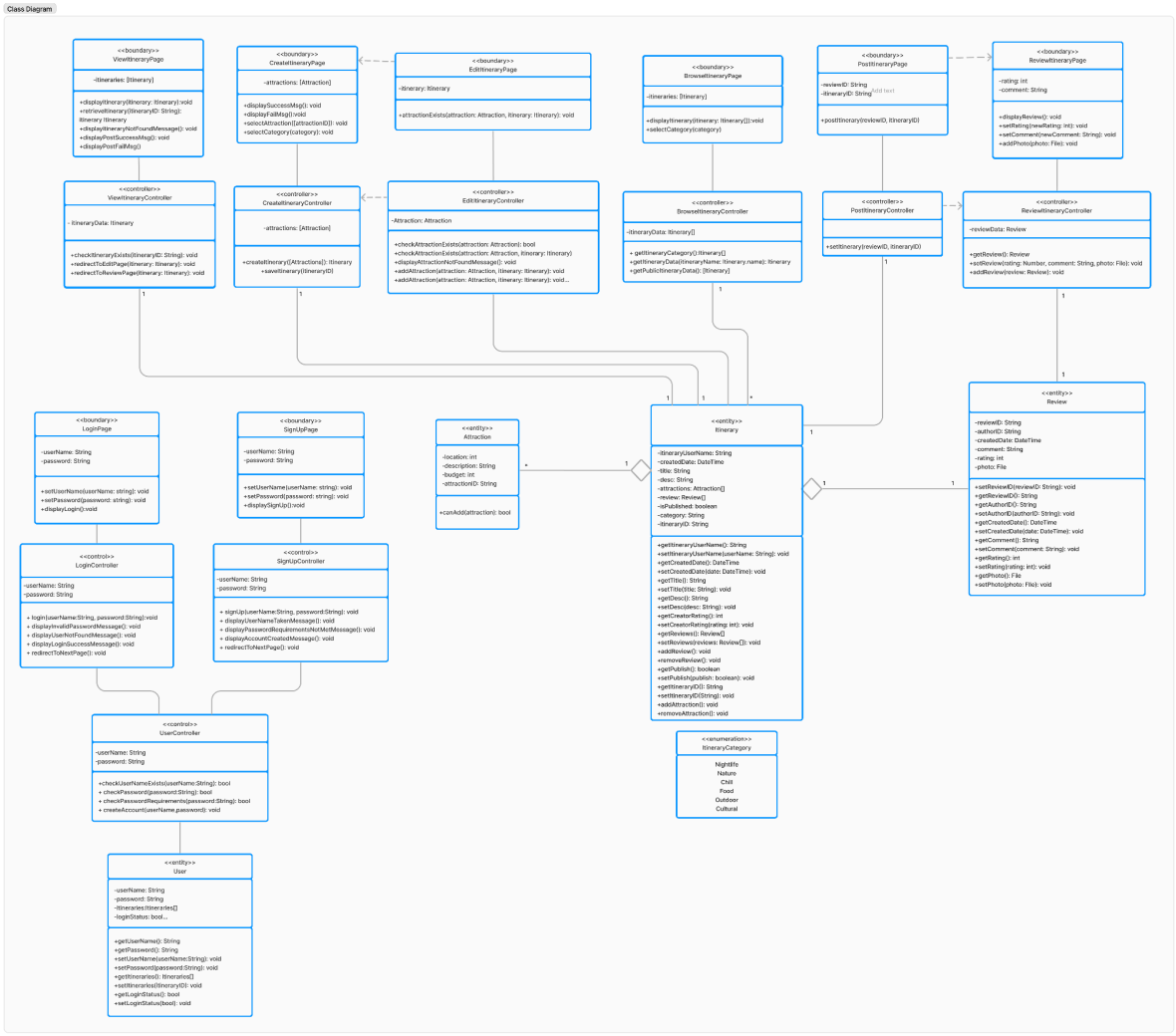
Use Case ID:	009		
Use Case Name:	View Itinerary		
Created By:	Eyan Chua	Last Updated By:	Eyan Chua
Date Created:	01/02/2024	Date Last Updated:	01/02/2024

Actor:	Users, Prisma Database
Description:	User can click into each itinerary to view the title, description, budget, attractions, routes and images of the itinerary.
Precondition:	User is successfully logged in.
Postcondition:	User can view the information of the itinerary
Priority:	High
Frequency of Use:	Frequent
Flow of Events:	1. User selects ‘Show details’ of an itinerary they are interested in. 2. The website retrieves itinerary information of the selected itinerary 3. The website displays the information to the user
Alternative Flows:	AF - S3 The website is unable to retrieve the information of the selected itinerary and displays an error. Website returns to step 1.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and issues:	-

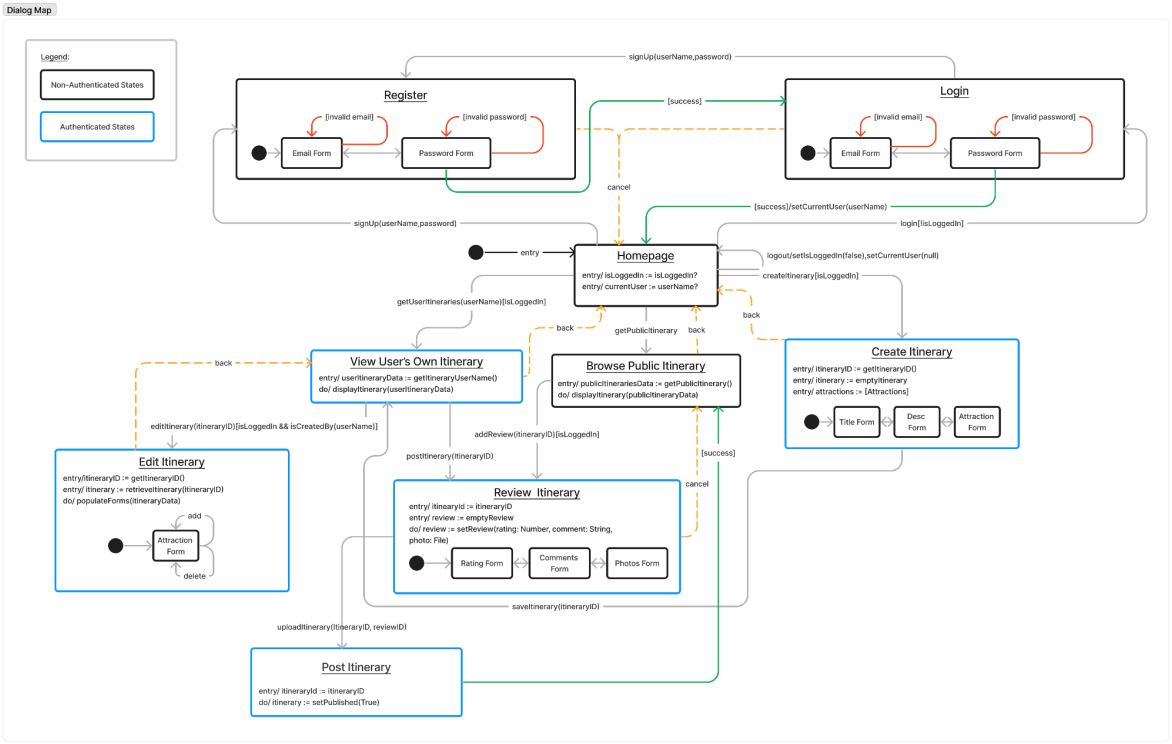
Use Case ID:	010		
Use Case Name:	Save Draft Itinerary		
Created By:	Eyan Chua	Last Updated By:	Eyan Chua
Date Created:	01/02/2024	Date Last Updated:	01/02/2024

Actor:	Users, Prisma Database
Description:	Users can save their itinerary as a draft.
Precondition:	User is successfully logged in.
Postcondition:	User itinerary is successfully saved as a draft.
Priority:	High
Frequency of Use:	Frequent
Flow of Events:	6. User has a filled in itinerary from the create itinerary of edit itinerary page 7. The user selects to save the travel itinerary. 8. The system processes the information. 9. The system uploads the travel itinerary and adds it to its database. 10. The website displays a confirmation message to the user.
Alternative Flows:	AF - S4 If the system is unsuccessful in uploading the travel itinerary, the website will return an error message and prompt the user to try again. The system returns to step 1.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and issues:	-

Class Diagram

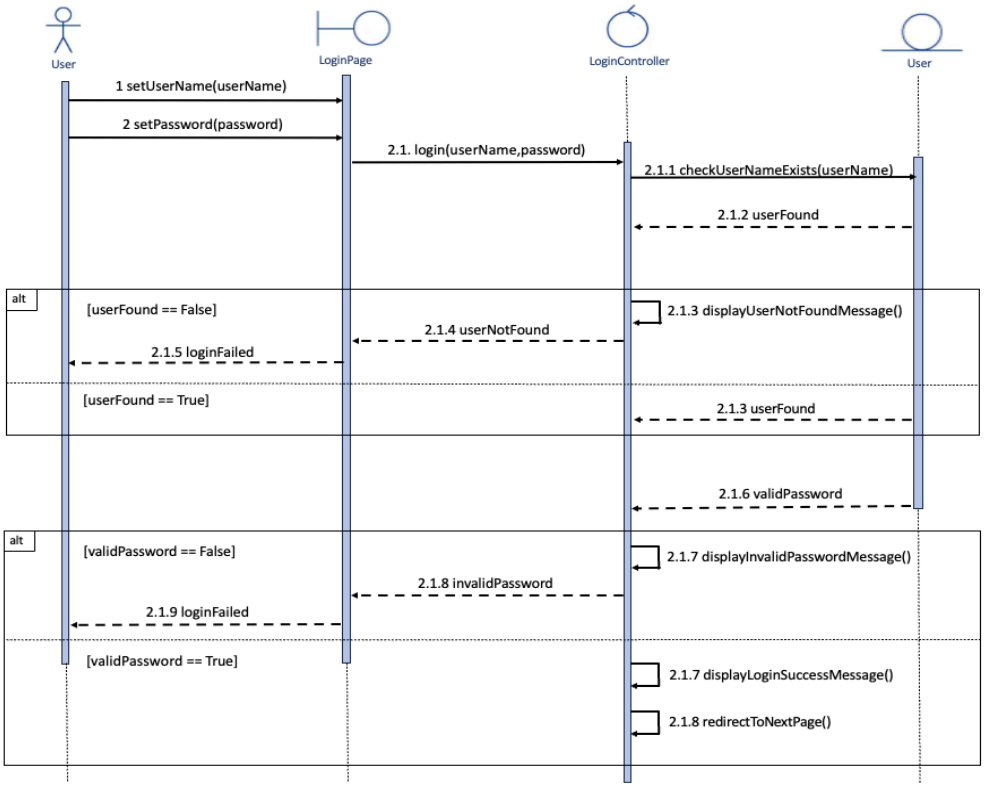


Dialogue Map

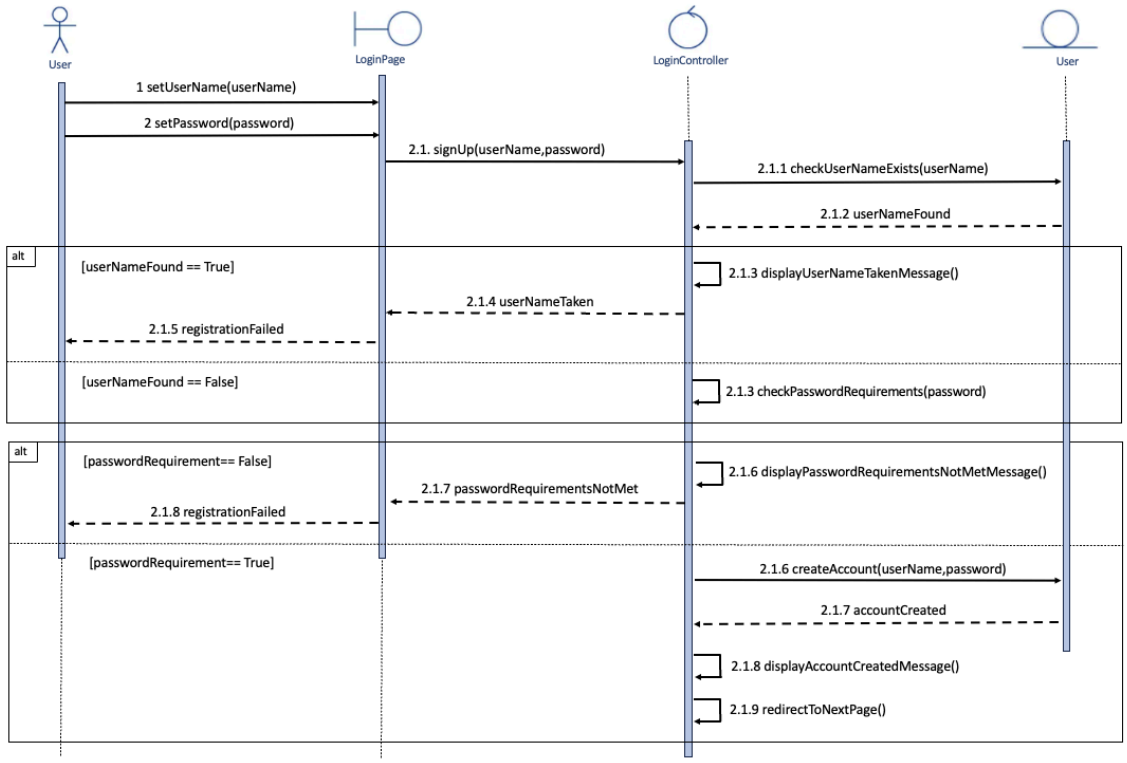


Sequence Diagrams

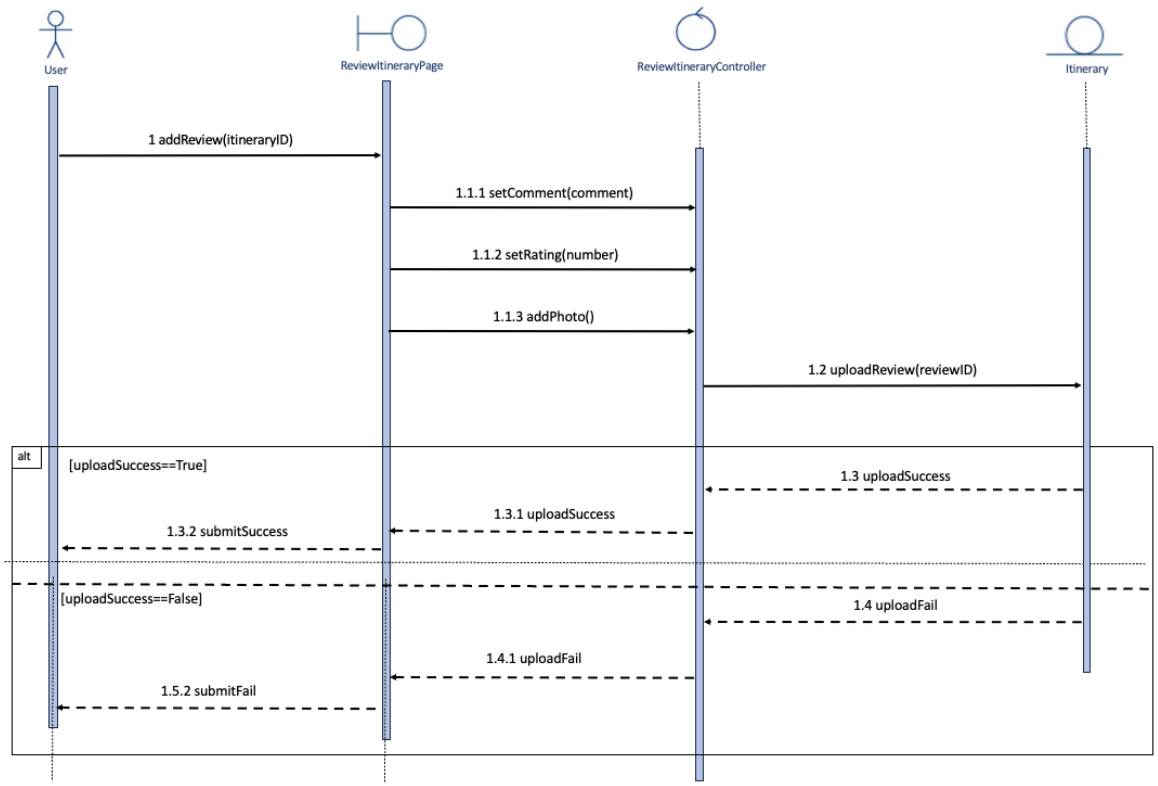
Login



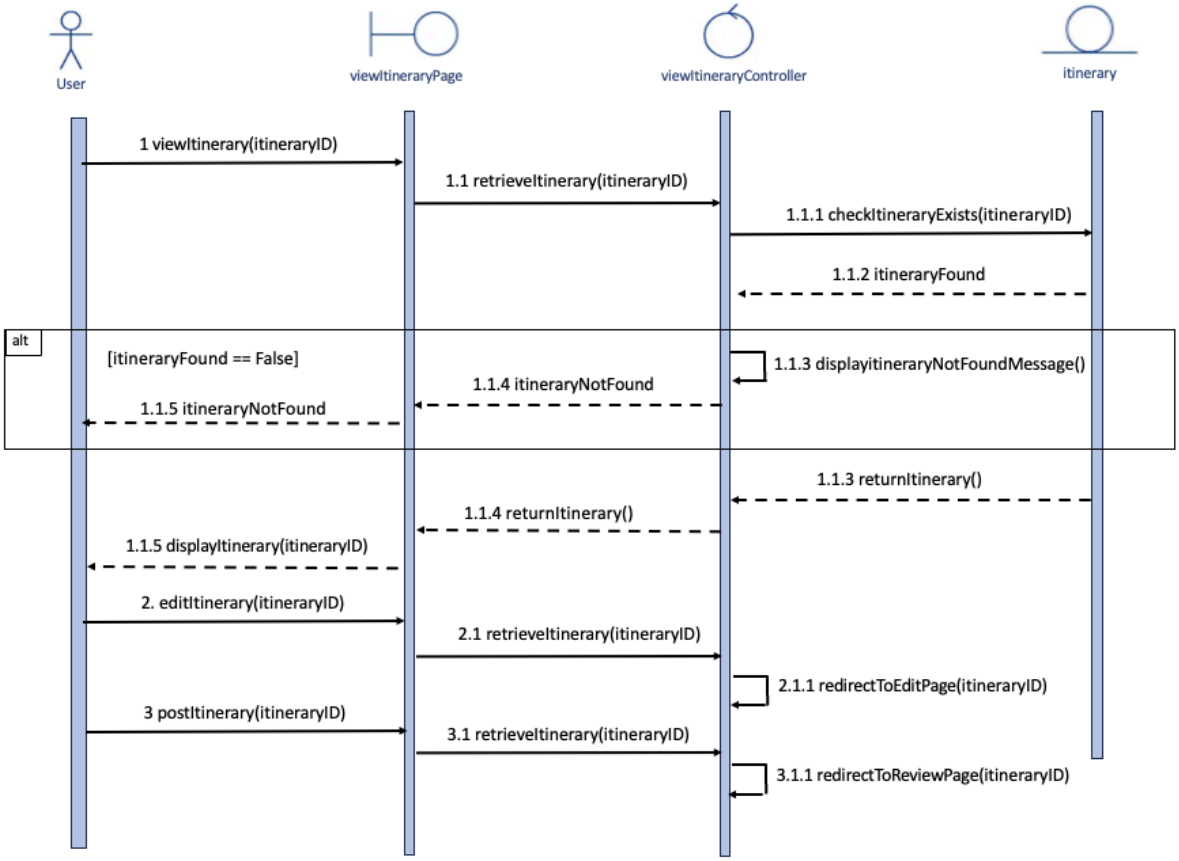
Sign up


User
LoginPage
LoginController
User
1 setUserName(userName)
2 setPassword(password)
2.1. signUp(userName,password)
2.1.1 checkUserNameExists(userName)
2.1.2 userNameFound
alt
[userNameFound == True]
2.1.3 displayUserNameTakenMessage()
2.1.4 userNameTaken
2.1.5 registrationFailed
[userNameFound == False]
2.1.3 checkPasswordRequirements(password)
alt
[passwordRequirement== False]
2.1.6 displayPasswordRequirementsNotMetMessage()
2.1.7 passwordRequirementsNotMet
2.1.8 registrationFailed
[passwordRequirement== True]
2.1.6 createAccount(userName,password)
2.1.7 accountCreated
2.1.8 displayAccountCreatedMessage()
2.1.9 redirectToNextPage()

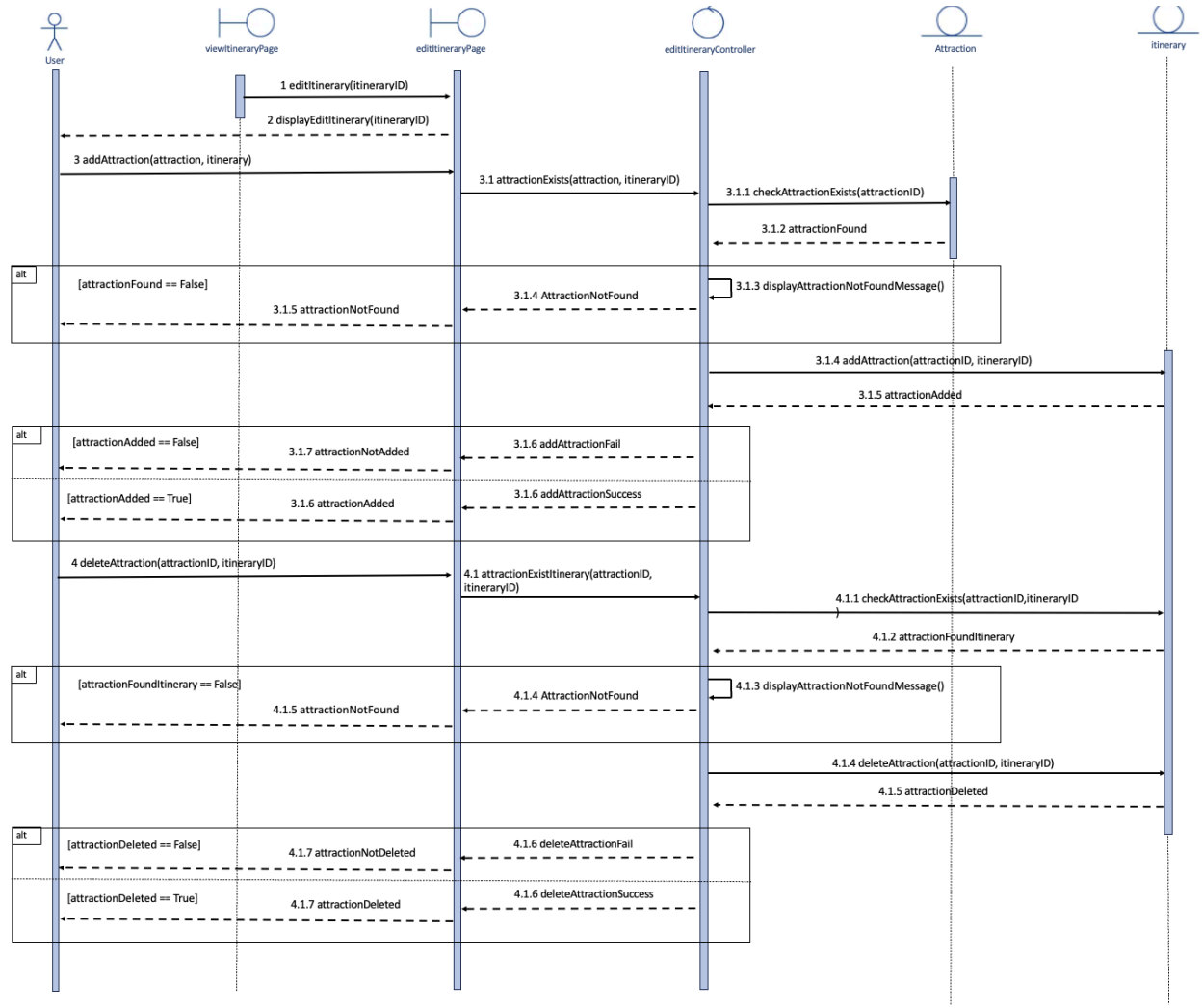
Review Itinerary



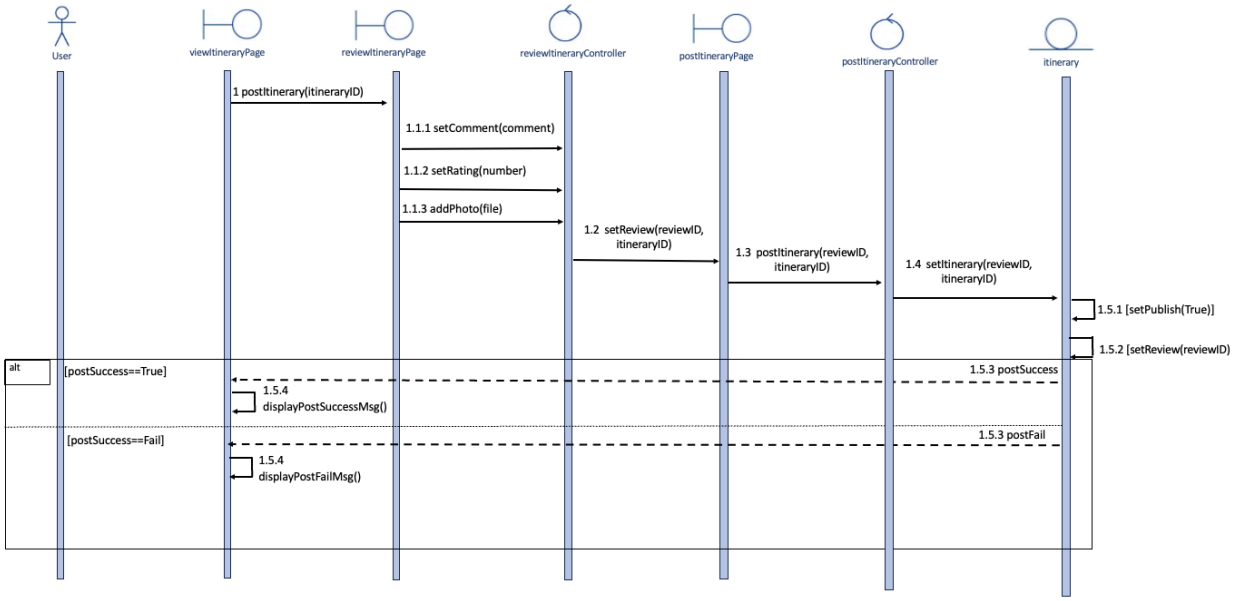
View Itinerary



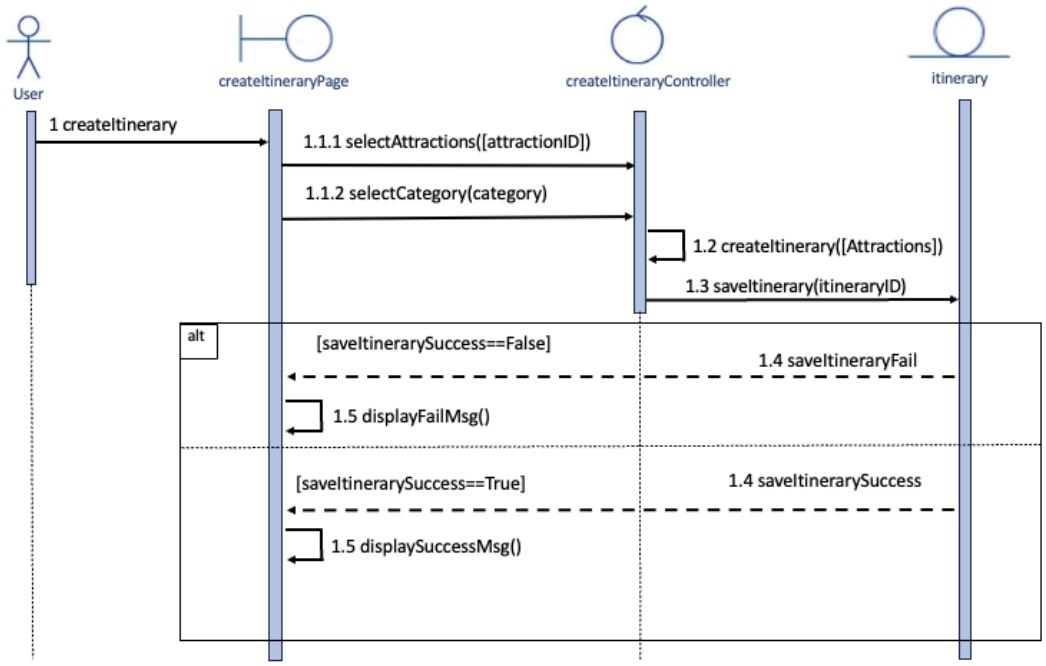
Edit Itinerary



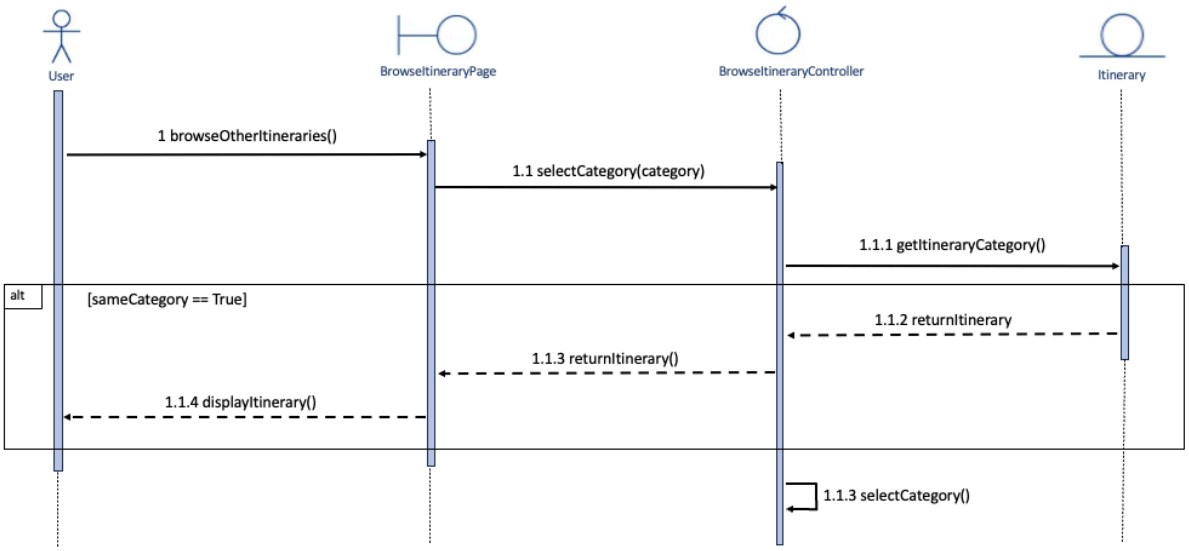
Post Itinerary



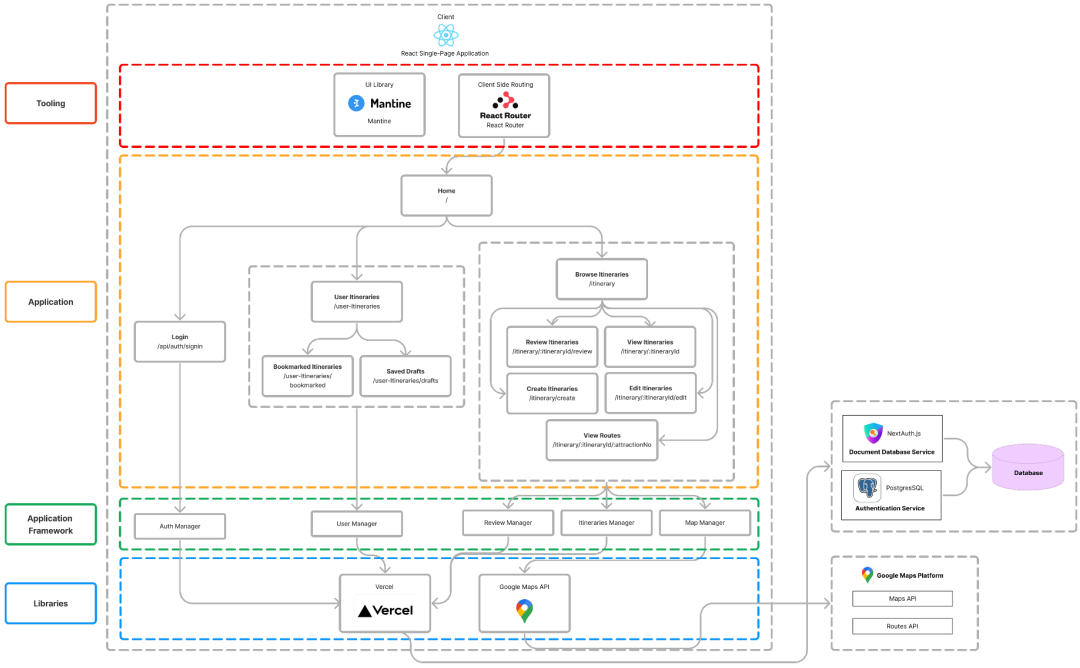
Create Itinerary



Browse Itinerary



System Architecture



Appendix C: Testing

Black-Box

Equivalence class and boundary value testing

1. **Create itinerary feature**

Title Control Class

Valid Equivalence Class	Invalid Equivalence Class
{Any string that is between 1 - 50 characters}	{Empty strings or strings with more then 50 characters}

Description Control Class

Valid Equivalence Class	Invalid Equivalence Class
{Any string that is between 1 - 250 characters}	{Empty strings or strings with more then 250 characters}

Valid Equivalence Class	Invalid Equivalence Class
{Any selection of 1 or more attractions provided in the list of attractions}	{A selection of 0 attractions}

Title	Description	Attraction	Expected Output	Actual Output
‘’	‘test description’	‘Singapore Zoo’, ‘Merlion’	Invalid	Invalid
‘Lorem ipsum dolor sit…’ (over 50 characters)	‘test description’	‘Singapore Zoo’, ‘Merlion’	Invalid	Invalid
‘test title’	‘’	‘Singapore Zoo’, ‘Merlion’	Invalid	Invalid
‘test title’	‘Lorem ipsum dolor sit…’ (over 250 characters)	‘Singapore Zoo’, ‘Merlion’	Invalid	Invalid
‘test title’	‘test description’	‘’	Invalid	Invalid
‘test title’	‘test description’	‘My House’	Invalid	Invalid
‘test title’	‘test description’	‘Singapore Zoo’, ‘Merlion’	Accepted	Accepted

2. **Review itinerary feature**

Rating Control Class

Valid Equivalence Class	Invalid Equivalence Class
1 2 3 4 5	{Any real number that is not an integer between 1 and 5 (inclusive)}

Review Control Class

Valid Equivalence Class	Invalid Equivalence Class
{Any string that is between 1 - 250 characters}	{Empty strings or strings with more then 250 characters}

Image Control Class

Valid Equivalence Class	Invalid Equivalence Class
{Any image file in the form of a png or jpg}	{Any files not in the form of a png or jpg}

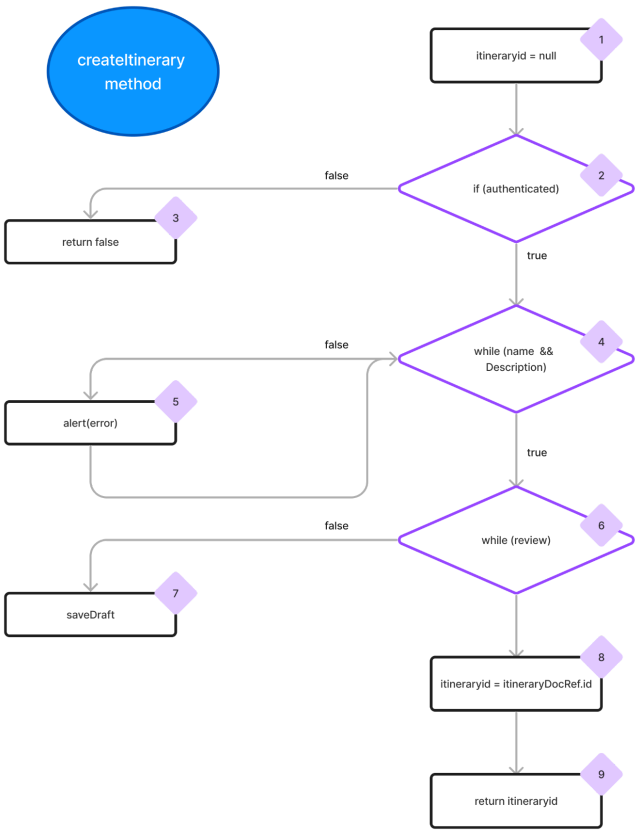
Comment	Rating	Image	Expected Output	Actual Output

‘test comment’	6	png file	Invalid	Invalid
‘test comment’	2.5	png file	Invalid	Invalid
‘’	4	png file	Invalid	Invalid
‘Lorem ipsum dolor sit…’ (over 250 characters)	4	png file	Invalid	Invalid
‘test comment’	4	pdf file	Invalid	Invalid
‘test comment’	4	png file	Accepted	Accepted

White-Box

Basis Path Testing

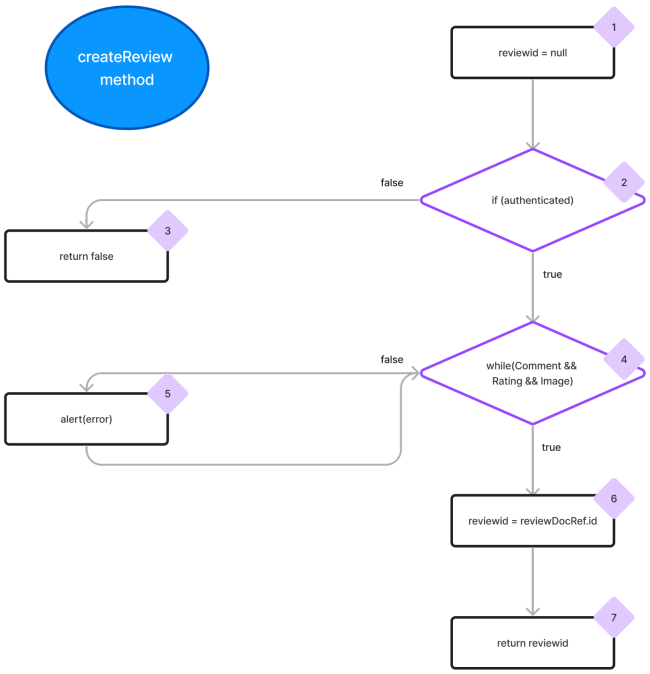
1. createItinerary



Cyclomatic Complexity = 2

No.	Test Cases	Basis Path
1	Title: 'Trip to Singapore Zoo' Description: 'Highly recommended for animal lovers!'	1,2,4,6,8,9
2	**Initial input** Title: 'Trip to Singapore Zoo' Description: ''	1,2,4,5,4,6,8,9
	Updated input at second occurrence of 4 Title: 'Trip to Singapore Zoo' Description: 'Highly recommended for animal lovers!'	

2. createReview



Cyclomatic Complexity = 2

No.	Test Cases	Basis Path
1	Comment: ‘Lovely trip, highly recommend!’ Rating: ‘5’ Images: png file	1,2,4,6,7
2	**Initial input** Comment: ‘Lovely trip, highly recommend!’ Rating: ‘’ Images: png file	1,2,4,5,4,6,7
	Updated input at second occurrence of 4 Comment: ‘Lovely trip, highly recommend!’ Rating: ‘5’ Images: png file	