Software Requirements Specification

for

SC2006

Version 1.3 approved

Prepared by

Chin Jun Hao, Mark U2221742F

Zhong Xing Jie U2221276A

Aditya Mani U2222778D

Charmain Ang Wan Theng U2223590D

Nguyen Tien Dat U2220949L

Nanyang Technological University, Team Softwhere EnginHere

14/04/24

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Nguyen Tien Dat	08/04/24	Moving Lab 1 Deliverables to SRS (Other Nonfunctional Requirements)	1.0
Nguyen Tien Dat	09/04/24	Overall Description	1.1
Nguyen Tien Dat	10/04/24	Moving Lab 3 Deliverables to SRS (System Features)	1.2
		Introduction	
		External Interface Requirements	
		References	
		Appendix A	
		Appendix B	
Mark, Mani, Nguyen Tien Dat, Xing Jie, Charmain		Added testing documentation	1.3

1. Introduction

1.1 Purpose

CheckIt is a revolutionary mobile application designed to enhance your daily planning experience. With its personalized and smart planning features, CheckIt goes beyond traditional to-do lists, intelligently curating your schedule to optimize your day. The app leverages APIs to provide location-based navigation, weather insights, and event recommendations.

1.2 Document Conventions

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

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

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

1.2.3 Highlighting
Critical sections are bolded

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

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

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

1.2.7 Revision History
This document follows version 1.0 and was last revised on 14/04/24

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

1.3 Intended Audience and Reading Suggestions

The SRS is intended for various stakeholders involved in the project. Listed below is the type of audience which can read and have a more comprehensive understanding of the project, according to their roles and needs.

1.3.1 Developers
Developers will use the SRS to obtain a deeper understanding of our web application. They may find Functional and Non-functional Requirements most useful, together with the use case diagrams.

1.3.2 Project Managers
Project managers will use the SRS to draw up a timeline to allocate time, resources and manpower to specific parts of the project and achieve their respective goals. They may find the Project Scope and Overall Description most useful.

1.3.3 Marketing staff
Marketing staff will use the SRS to see the web application from a user's perspective through the key features and the user's needs. They may find the Functional and Non-functional Requirements and User Interface most useful.

1.3.4 Users
Users will use the SRS to gain some understanding of the workings of the web application, and its limitations too. They may find the Introduction and Functional and Non-functional Requirements most useful.

1.3.5 Testers
Testers will use the SRS to generate test cases and test them against the functional requirements of the web application. They may find the Functional Requirements most useful.

Document Contents
This SRS contains information relating to the project scope, user documentation, external interface requirements, functional and non-functional requirements, and other important information.

Document Overview
This SRS is organized into the following sections: Introduction, Overall Description, External Interface Requirements, Other Nonfunctional Requirements, and Appendices.

1.4 Product Scope

The web application will utilize the publicly available Google Events SERP, Weather, and Google Maps APIs to achieve its purpose as mentioned in 1.1. OpenWeatherMap API, Google Events, and Google Maps APIs are provided by third parties.

The software frameworks we are using are **HTML5**, **CSS,** and **JavaScript** as they allow us to create dynamic and interactive user interfaces while maintaining their speed and processing power.

The frontend framework we are using is **Bootstrap**. Bootstrap is a free, open-source front-end development framework for creating websites. This was chosen as it allows us to use HTML5, CSS, and JavaScript as the primary languages.

The database and backend framework used for this project is **Firebase**. Firebase is a cloud-based platform that offers real-time database capabilities, allowing seamless integration into web development projects through its NoSQL database. It provides features like authentication, hosting, and analytics alongside its database functionalities.

1.5 References

1.5.1 APIs

1.5.1.1 Routing
1.5.1.1.1 Google Maps
1.5.1.2 Events
1.5.1.2.1 Google Events SERP
1.5.1.3 Weather
1.5.1.3.1 OpenWeatherMap

1.5.2 Frameworks

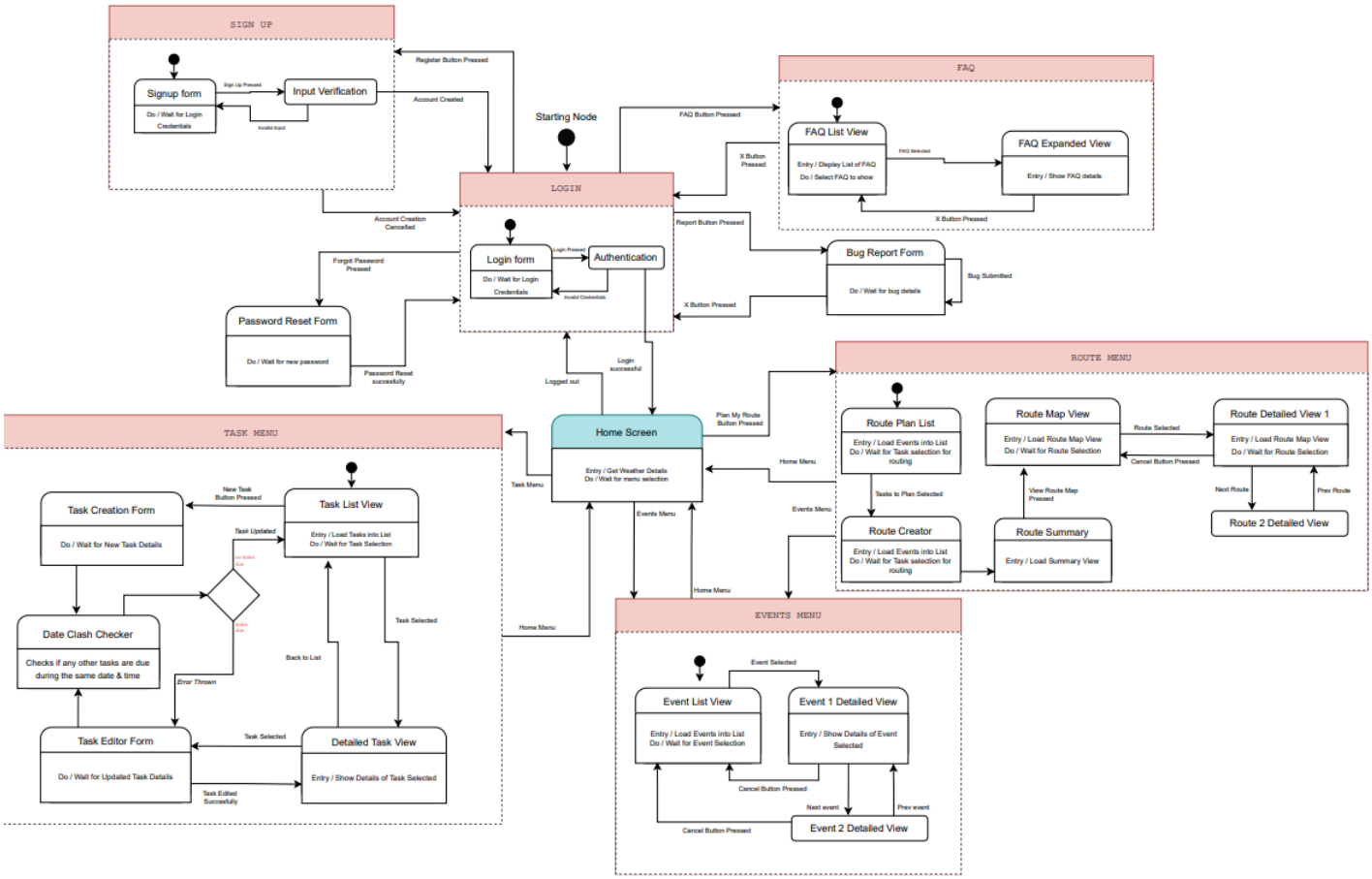
1.5.2.1 Database & Server
1.5.2.1.1 Firebase
1.5.2.2 CSS
1.5.2.2.1 Bootstrap

2. Overall Description

2.1 Product Perspective

Users make use of the publicly available APIs from Google Maps, Google Events, and OpenWeatherMap. The product allows users to schedule tasks, provides search functionality for events, and provides directions using the Google Maps API.

The image below shows the dialogue map for our application.



2.2 Product Functions

2.2.1 The website must provide login services before allowing users to access Todos.
2.2.2 The website must be able to provide location services.
2.2.3 The website must allow the user to create Todos.
2.2.4 The website should have real-time access to weather data of the day.
2.2.5 The website should have real-time access to the list of events happening near a location.
2.2.6 The website should be able to take in user feedback between 1 to 1000 characters.
2.2.7 The website should have an FAQ page.

2.3 User Classes and Characteristics

2.3.1 General User

Any user will be able to access the website using the Google search engine, from their electronic devices. They do not require any level of technical expertise.

2.3.2 Developer

Developers see the same interface from their electronic devices as the users. However, they can access the backend server where they will be able to access the database as well.

2.4 Operating Environment

The website requires an Internet browser that supports at least HTML5 or above, CSS3 or above, and JavaScript. Examples of such browsers include Google Chrome 124.0, Mozilla Firefox 123.0, etc.

The back-end server is hosted by Firebase, a cloud-based system.

2.5 Design and Implementation Constraints

The back-end server is implemented with libraries specifically for Firebase.

The back-end is organized based on the entity classes and each entity should have its repository and controller.

The front end is implemented with the HTML5, CSS, and JavaScript frameworks.

2.6 User Documentation

We have a live demo video showing how our web application works. The link to the video is here. The codebase is available on GitHub.

2.7 Assumptions and Dependencies

2.7.1 Dependencies

2.7.1.1 OpenWeatherMap, Google Events SERP, and Google Maps API

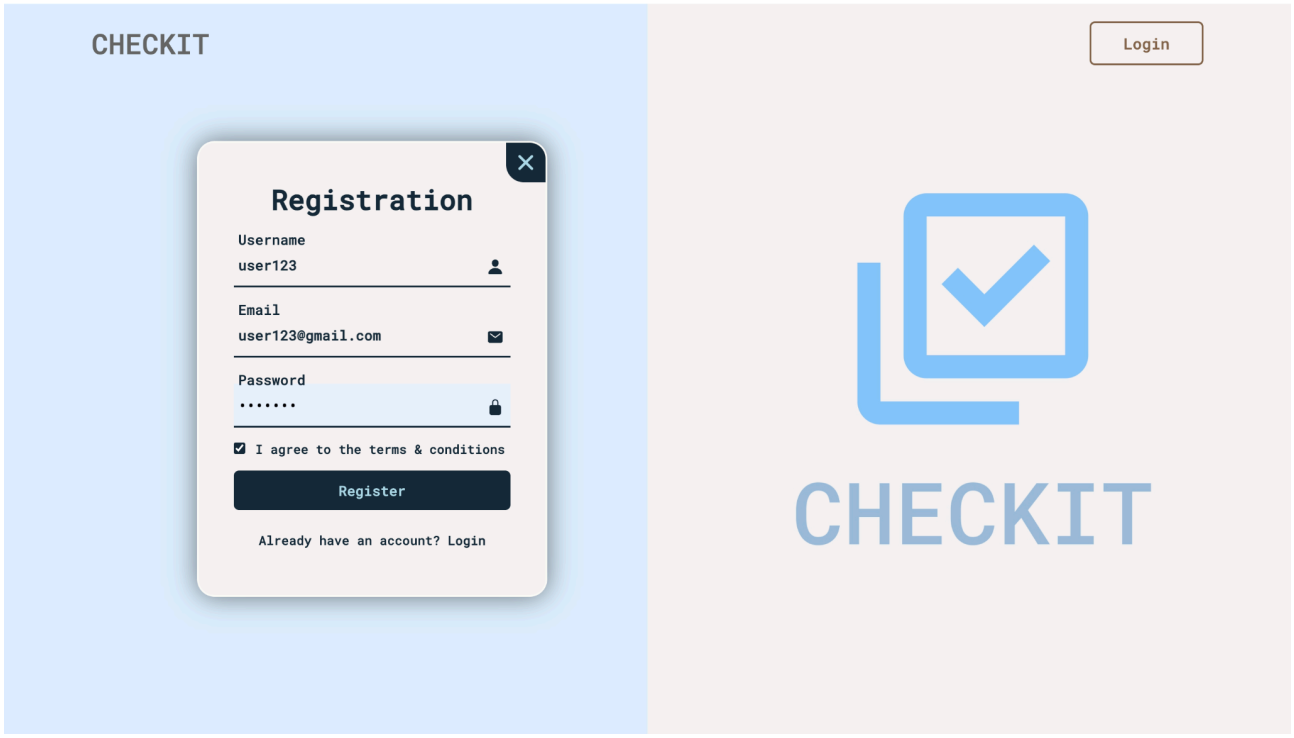
Our application relies on the data availability from third-party websites (OpenWeatherMap, Google Events and Google Maps).

2.7.2 Assumption

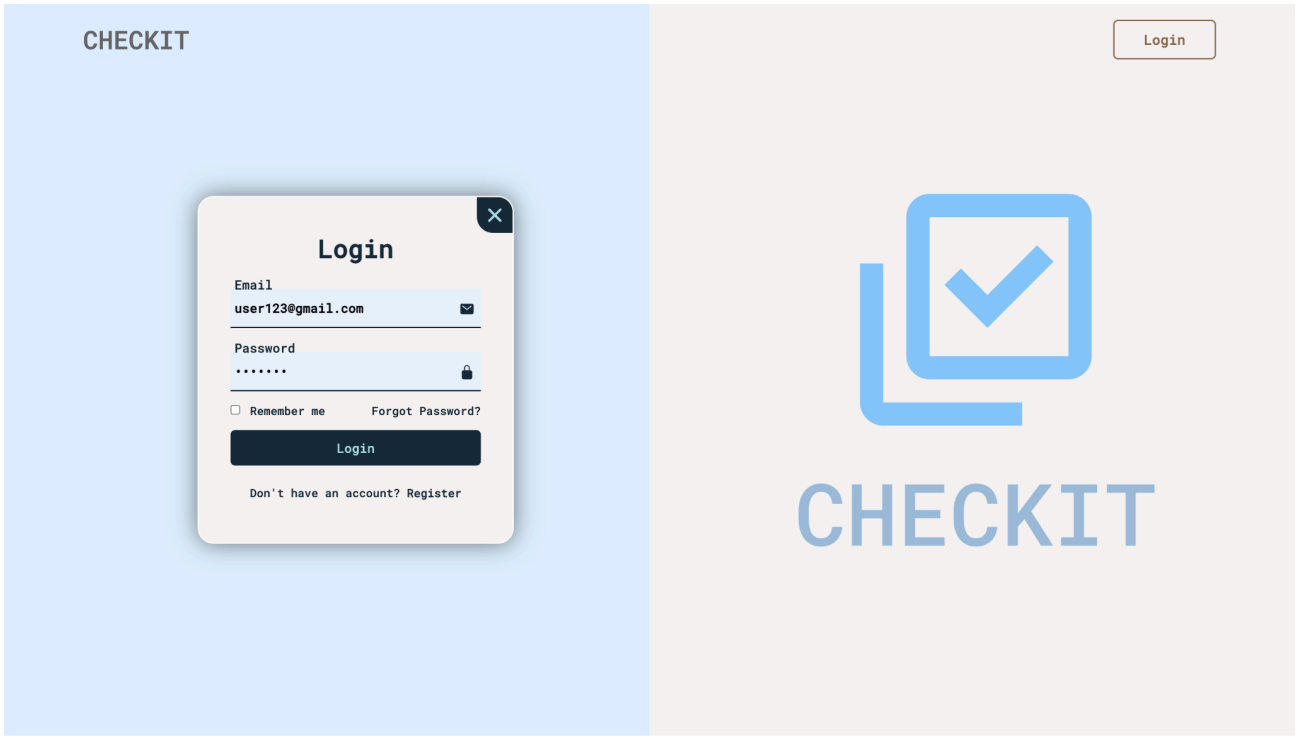
We assume that these APIs are continuously updated, maintained, and available for use by the relevant sectors. Any issues or discontinuation of these APIs will render our website non-functional.

3. External Interface Requirements

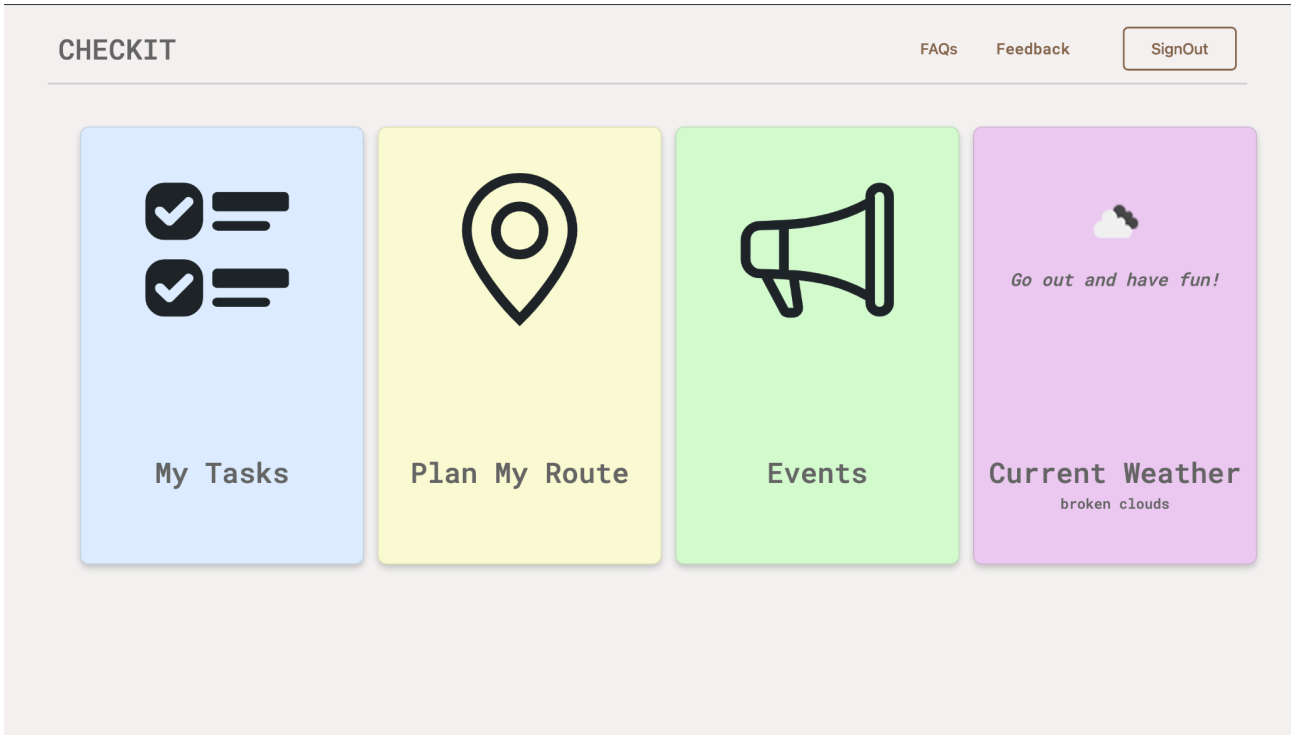
3.1 User Interfaces



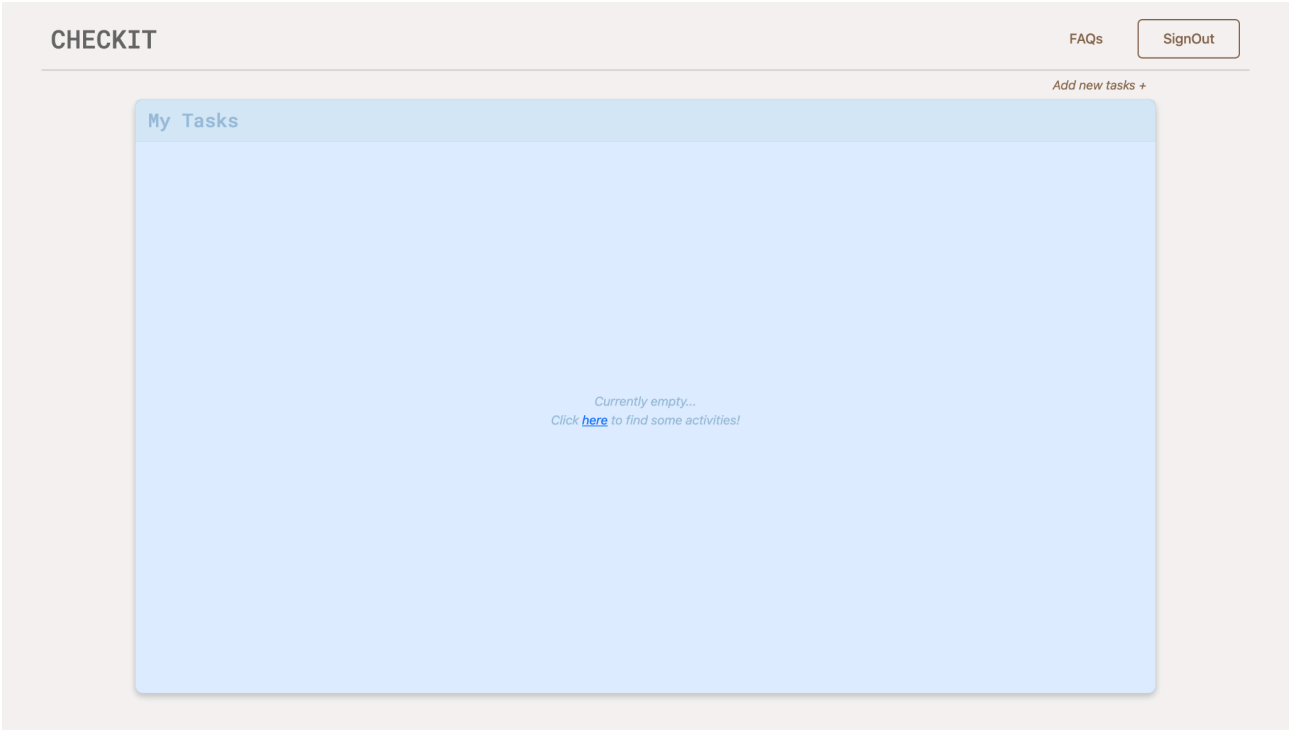
1. Register



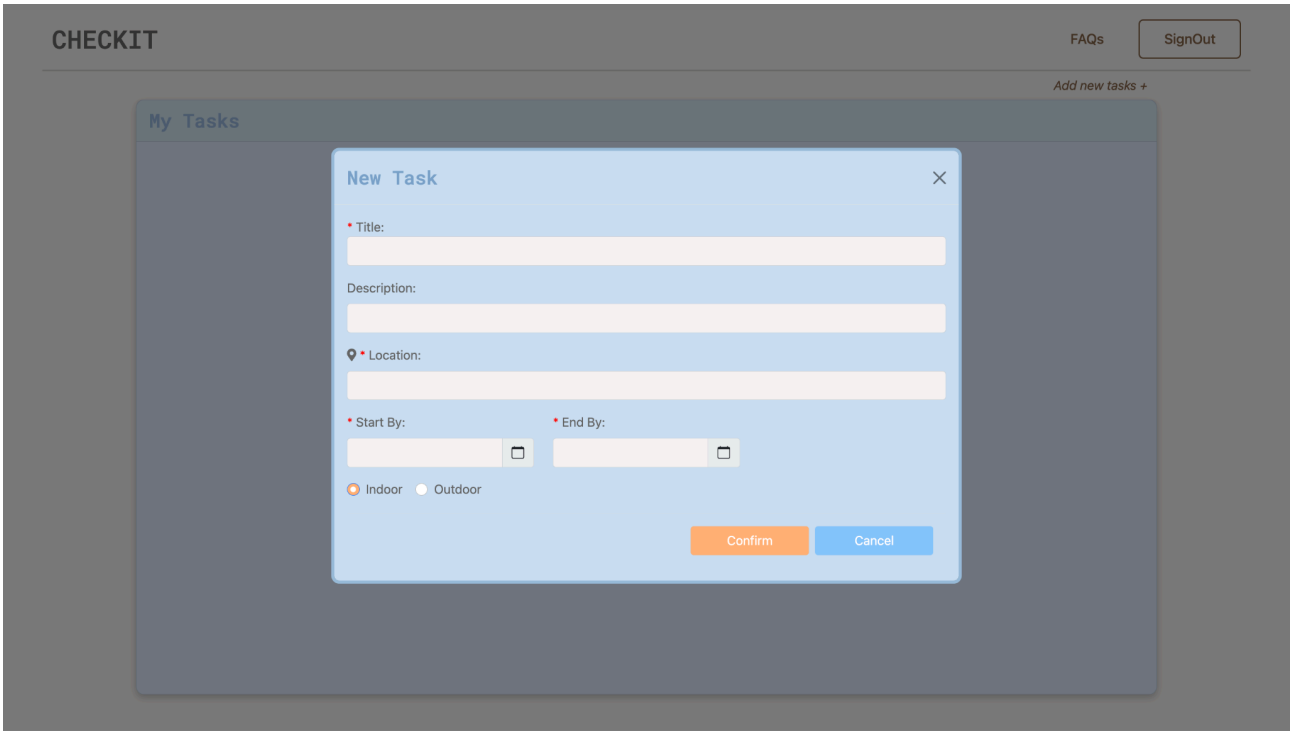
2. Login



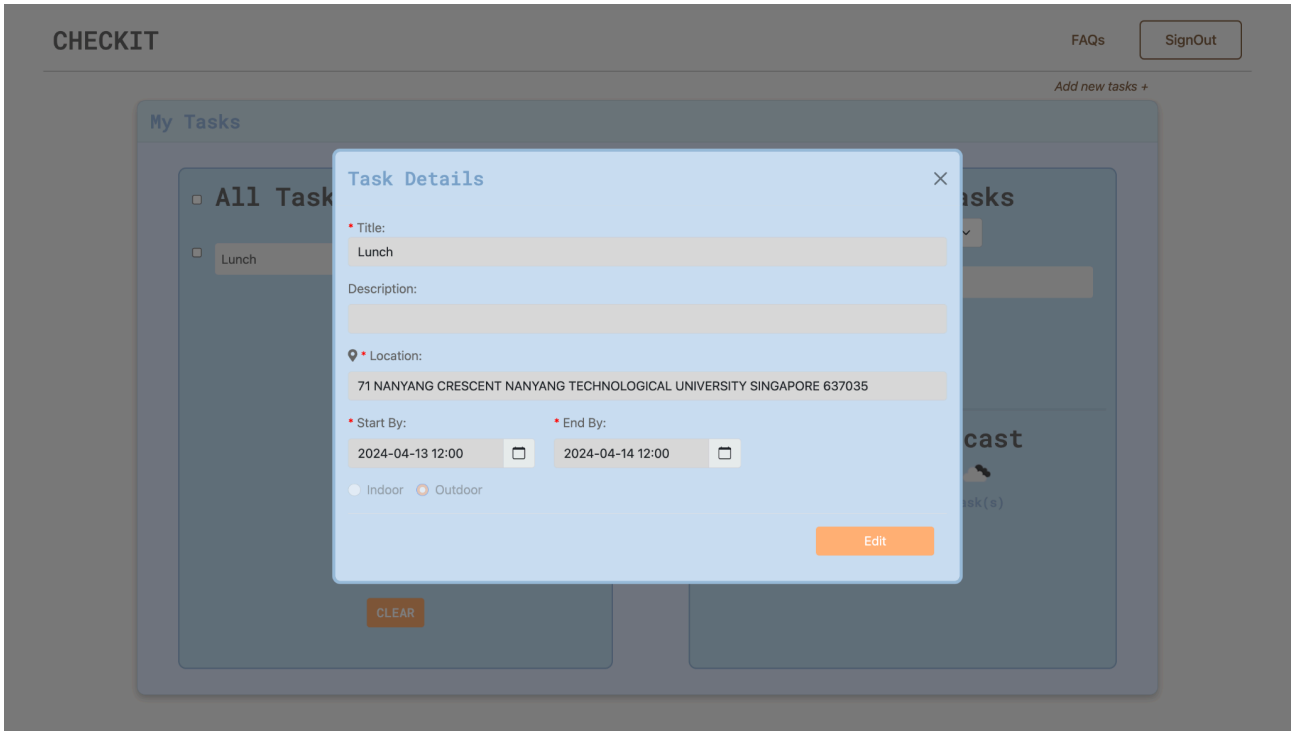
3. Home



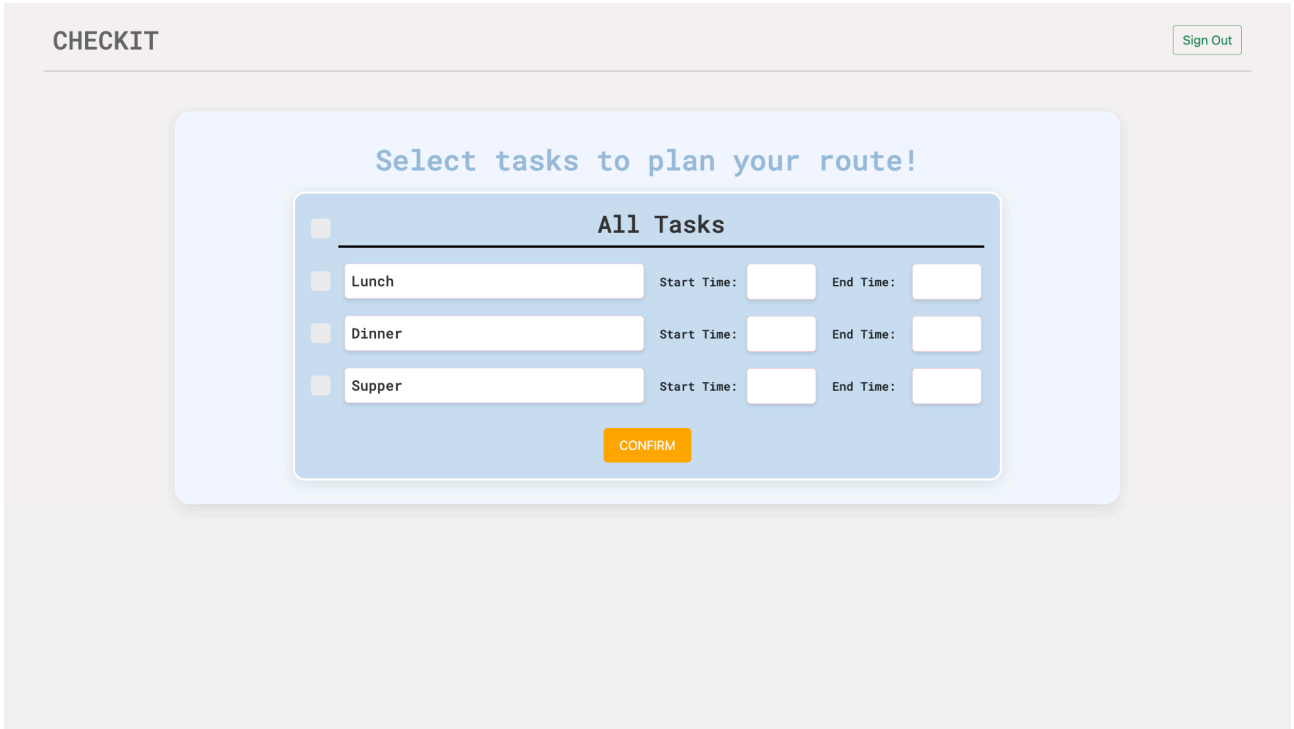
4. Empty Todo List



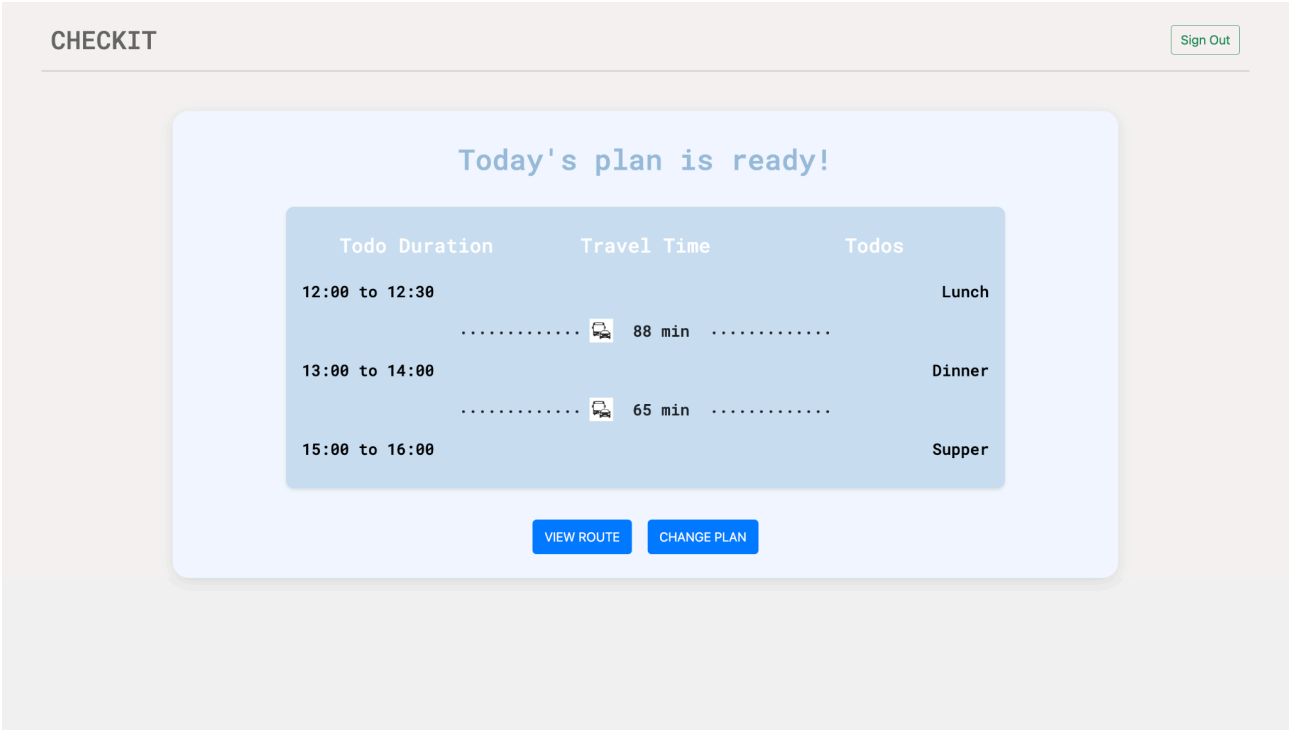
5. Create Todo



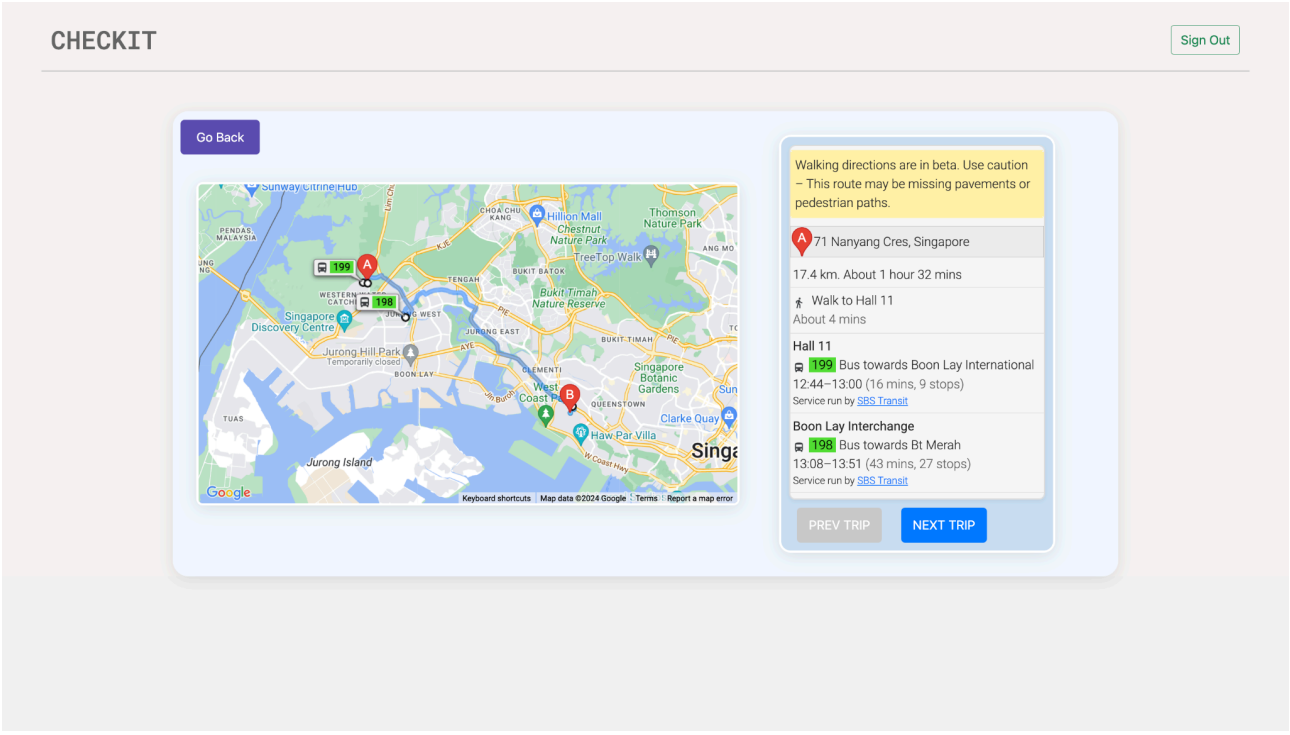
6. Edit Todo



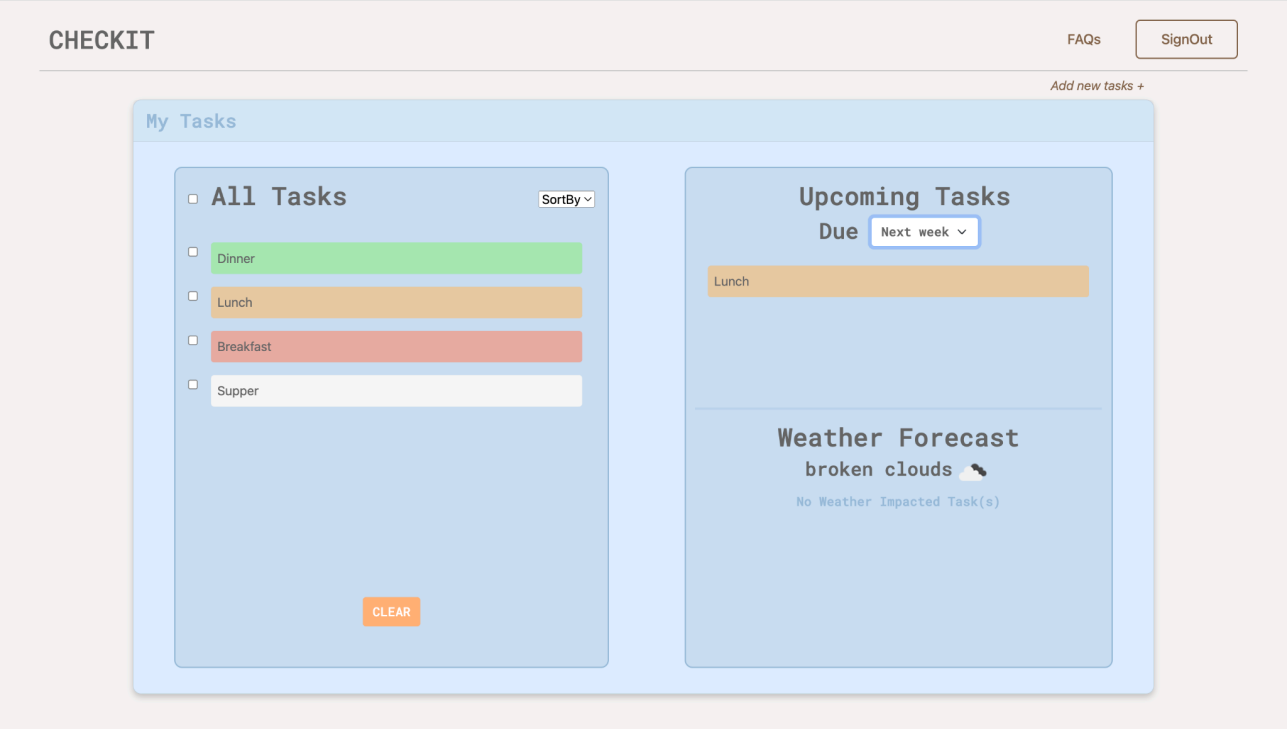
7. Todo Selection for Route Creation



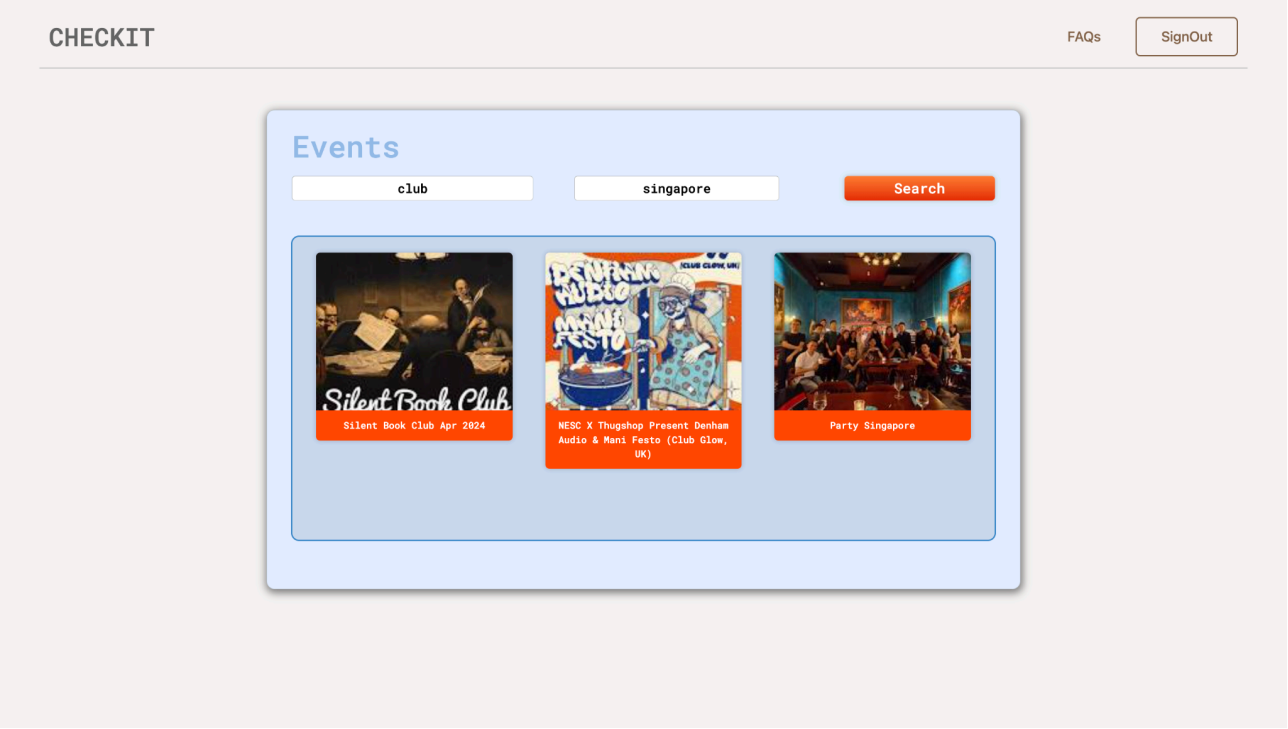
8. Trip Overview



9. Trip Summary



10. Populated Todo List



11. Event Search

3.2 Software Interfaces

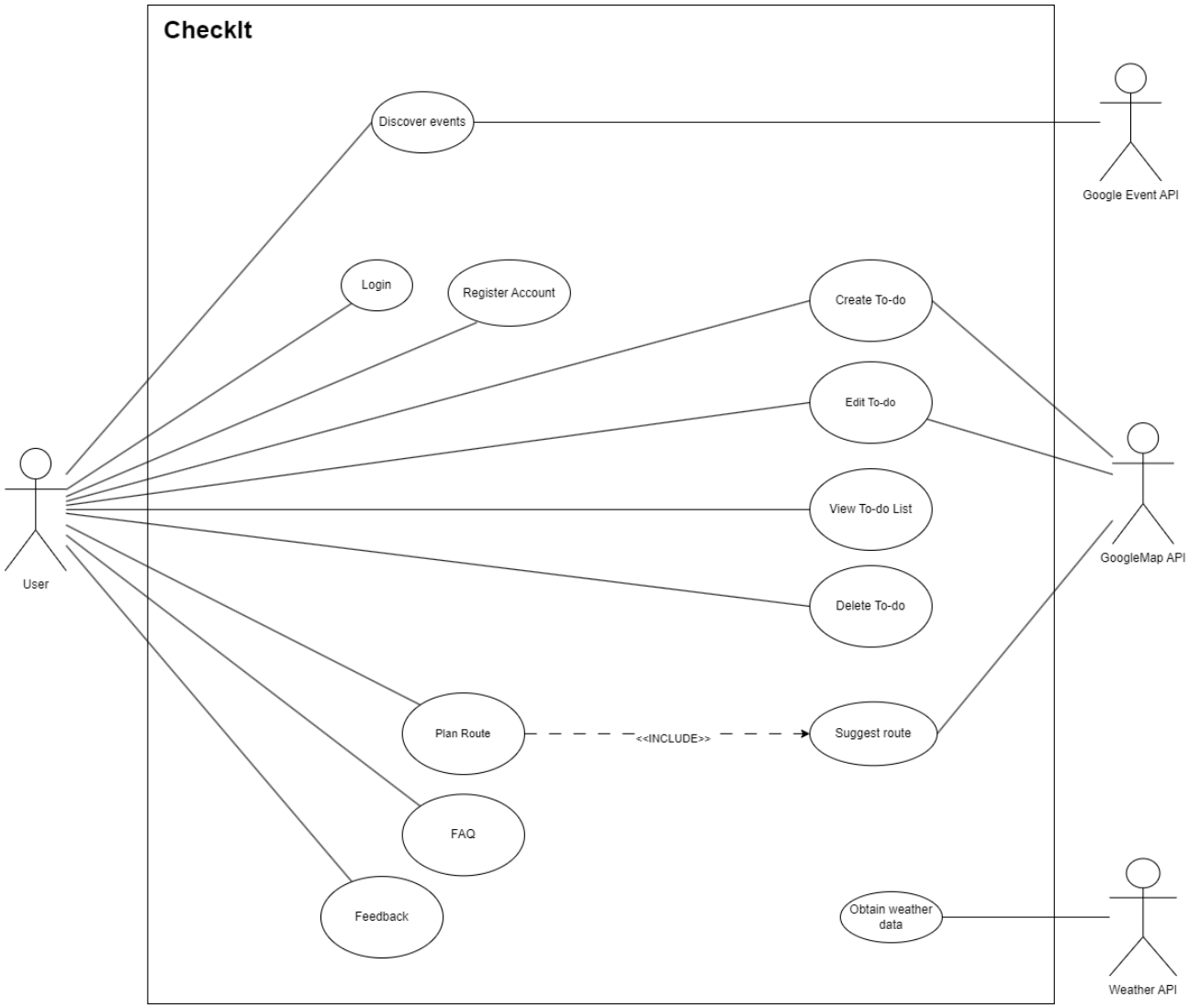
1. HTML5 (version 5)
2. CSS (Version 3)
3. JavaScript (Version ES14)
4. Firebase (Version 10.8.1)

3.3 Communications Interfaces

The main form of communication between the front and backend is via HTTP requests and responses. Data is communicated with the Firebase backend using the Firebase REST API.

4. System Features

Below is our use case diagram of the CheckIt application.

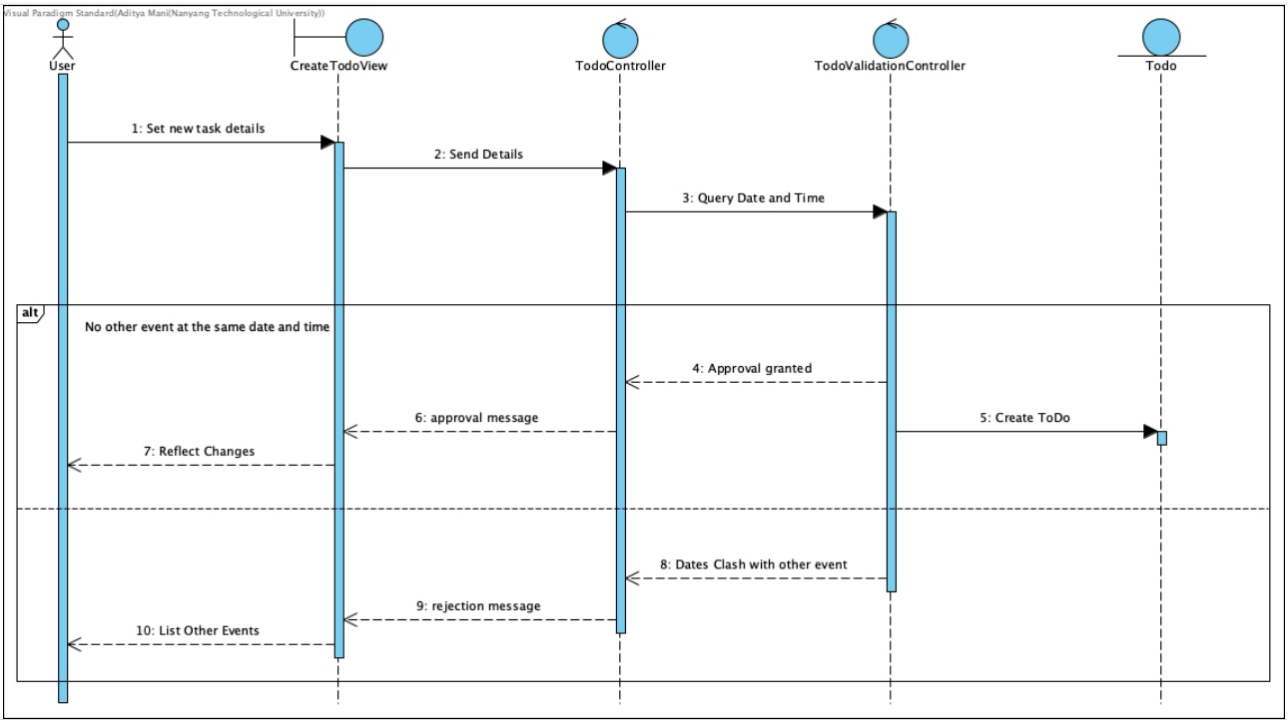


4.1 Create Todo

4.1.1 Description and Priority

This use case allows the user to create a To-do and add it to the To-do list. This feature has high priority.

4.1.2 Stimulus/Response Sequences



4.1.3 Functional Requirements

REQ-1: The user can select the option to create a new Todo.

REQ-2: The system will prompt the user for the Todo details: title, task, starting datetime, ending datetime, destination, indoor/outdoor status.

REQ-3: The user can input the required details and may use the included use case "manual destination input" to input the address of the activity, and "manual timing input" to input the activity time.

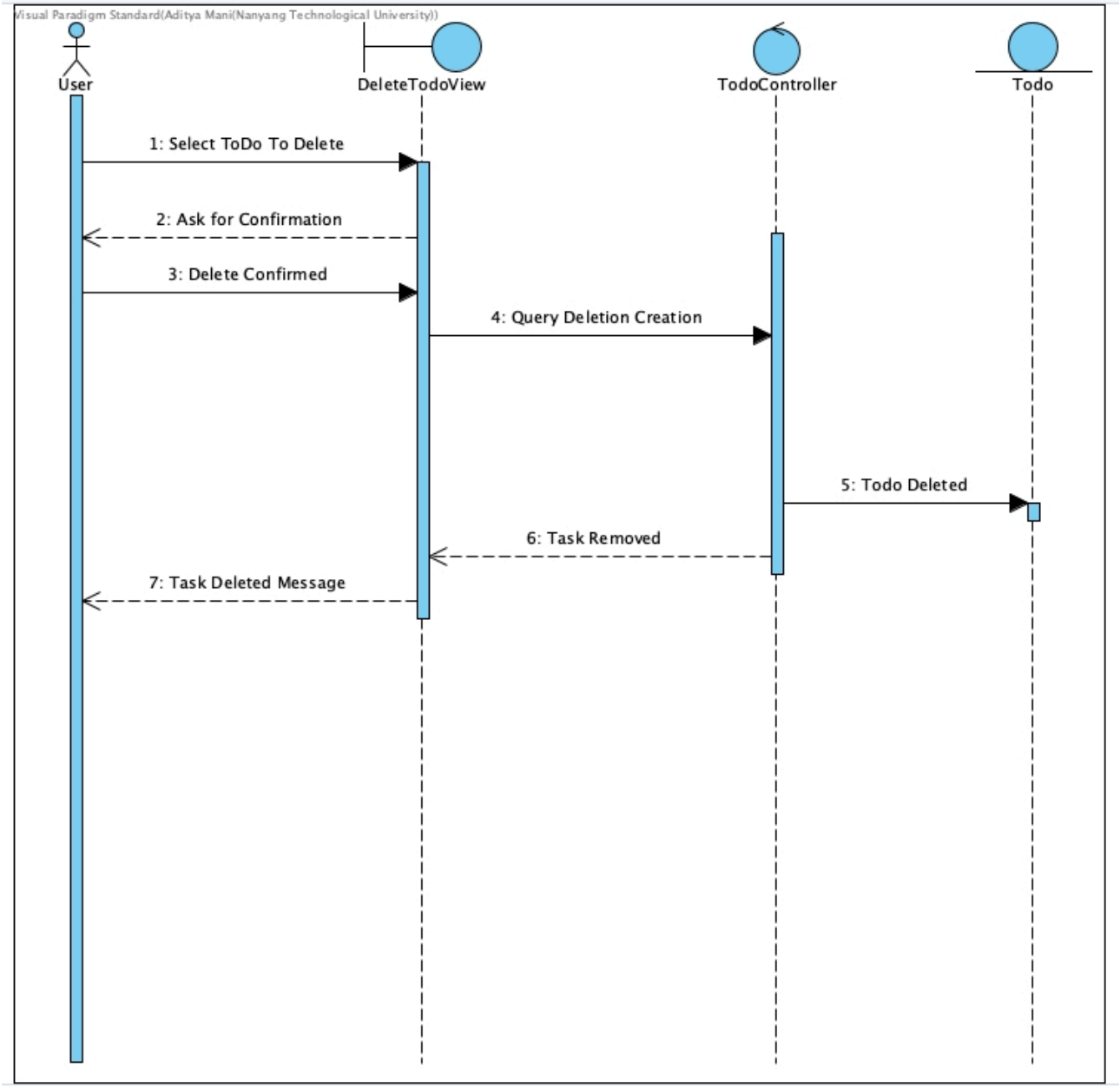
REQ-4: The system includes a use case “check for clashes” to identify if there’s any overlap between activities. If there is, the system will warn the user and ask the user to change the starting or ending datetime.

4.2 Delete Todo

4.2.1 Description and Priority

This use case allows the user to delete the To-do list. This feature has low priority.

4.2.2 Stimulus/Response Sequences



4.2.3 Functional Requirements

REQ-1: The user can select the option to delete a Todo.

REQ-2: The system must promp the user once for confirmation regarding the deletion request.

REQ-3: Upon confirmation, the system must delete the selected Todo from the database

REQ-4: The system must inform the user of the successful deletion in REQ-3.

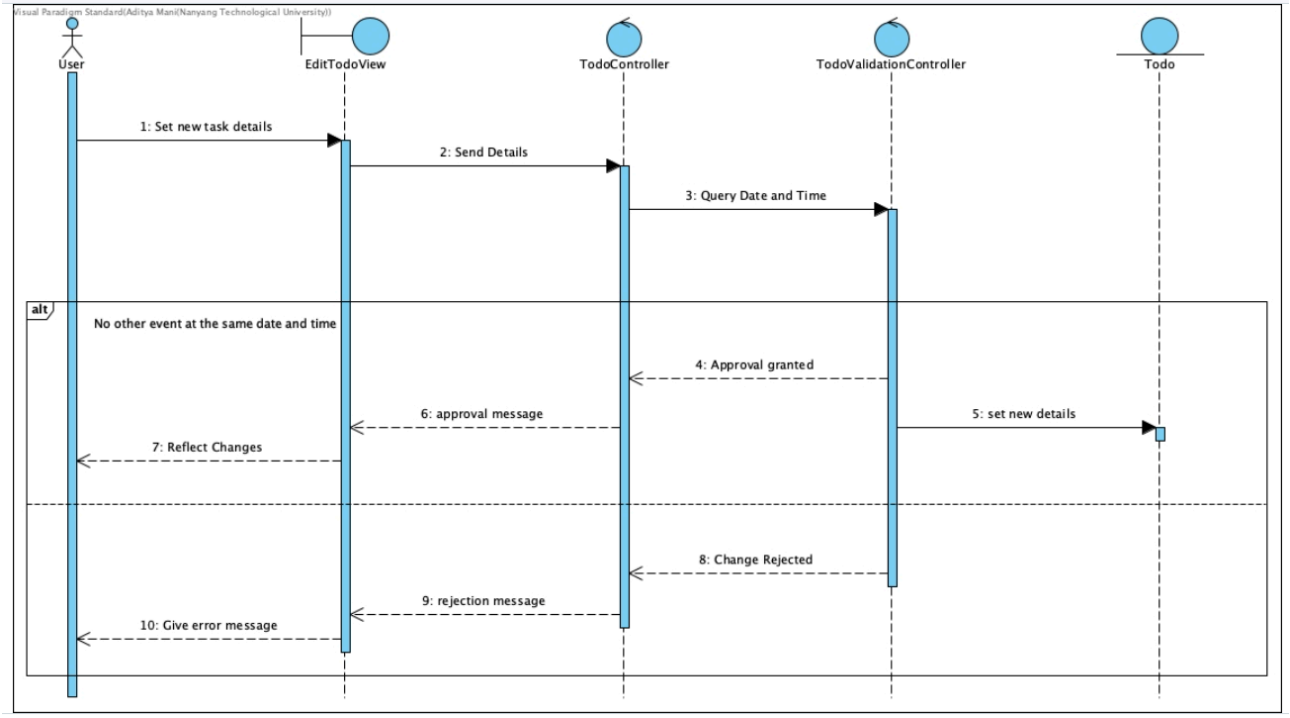
REQ-5: If there are no Todos to delete, the system informs the user that no Todos are available.

4.3 Edit Todo

4.3.1 Description and Priority

This use case allows the user to edit a To-do and update the To-do list. This feature has medium priority.

4.3.2 Stimulus/Response Sequences



4.3.3 Functional Requirements

REQ-1: The user can select the option to edit a specified Todo.

REQ-2: The system must prompt the user for the Todo details: title, task, starting datetime, ending datetime, destination, indoor/outdoor status.

REQ-3: The user can update the details and may use the included use case "manual destination input" to input the address of the activity, "manual timing input" to input the activity time.

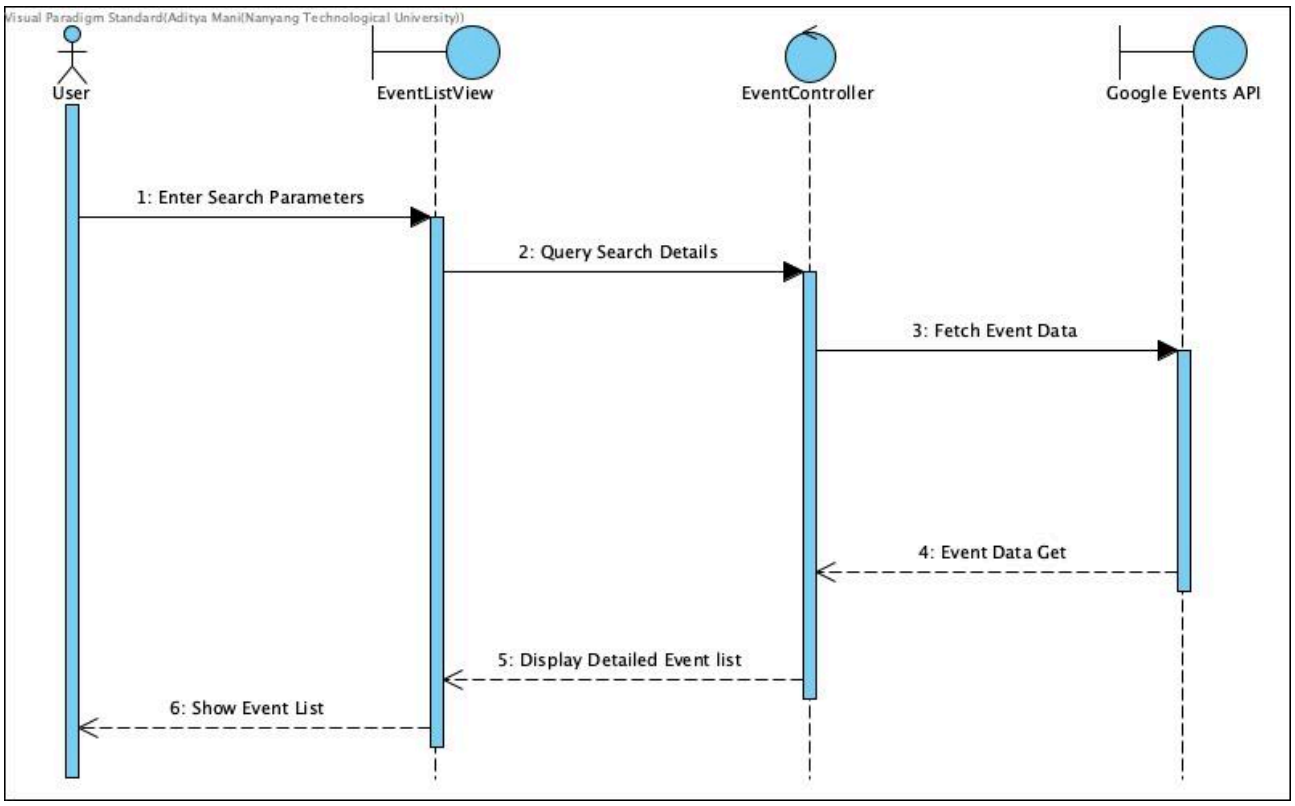
REQ-4: The system includes a use case “check for clashes” to identify if there’s any overlap between activities. If there is, the system must warn the user and ask the user to change the starting or ending datetime.

4.4 Events View

4.4.1 Description and Priority

This use case uses Google Event API to present events happening nearby. This feature has low priority.

4.4.2 Stimulus/Response Sequences



4.4.3 Functional Requirements

REQ-1: The user can select the option to discover events.

REQ-2: The system uses the Google Events API to present the user with events happening nearby, based on preferences and current location.

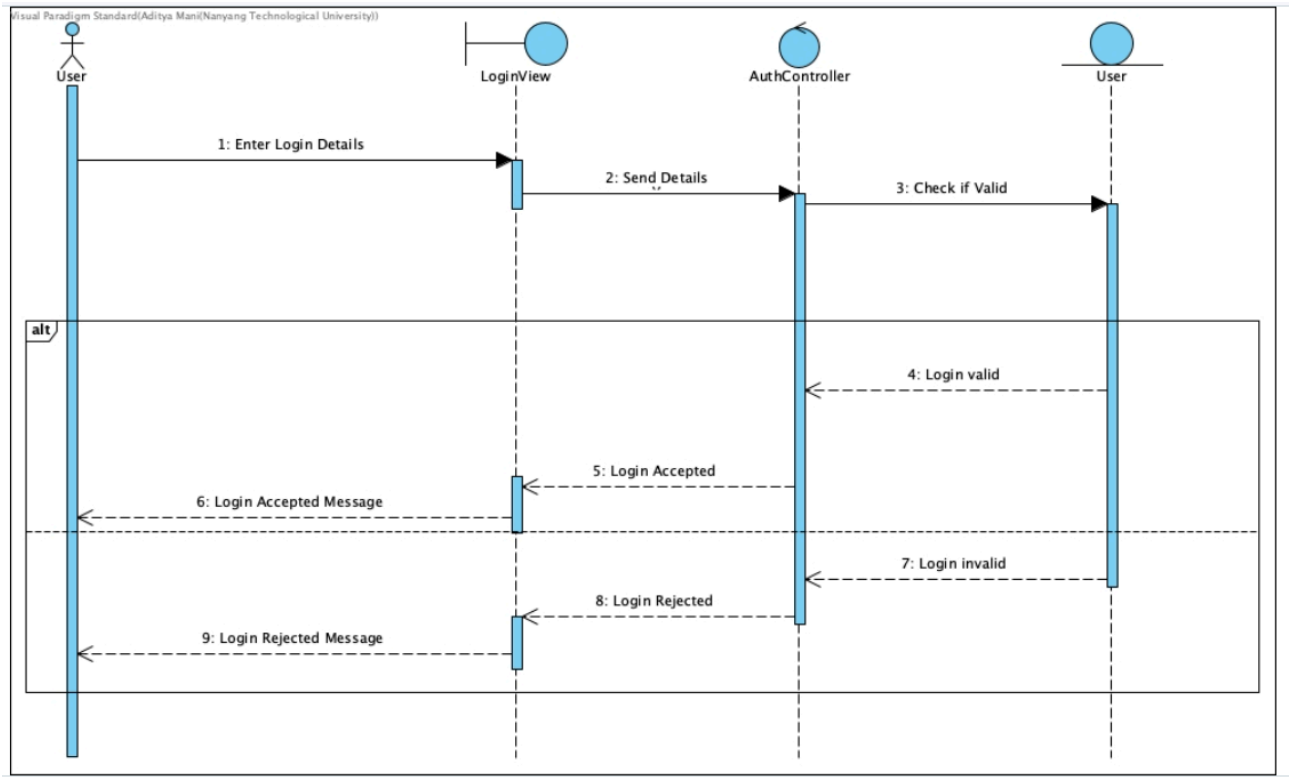
REQ-3: The user can view more information about an event or add it as a Todo.

4.5 Login

4.5.1 Description and Priority

This use case allows a new user to create his/her account. This feature has high priority.

4.5.2 Stimulus/Response Sequences



4.5.3 Functional Requirements

REQ-1: User can select the option to register.

REQ-2: User can provide a username, email, and password.

REQ-3: System must validate the provided information in REQ-2.

REQ-4: System must create a new user account in the database if validation is successful.

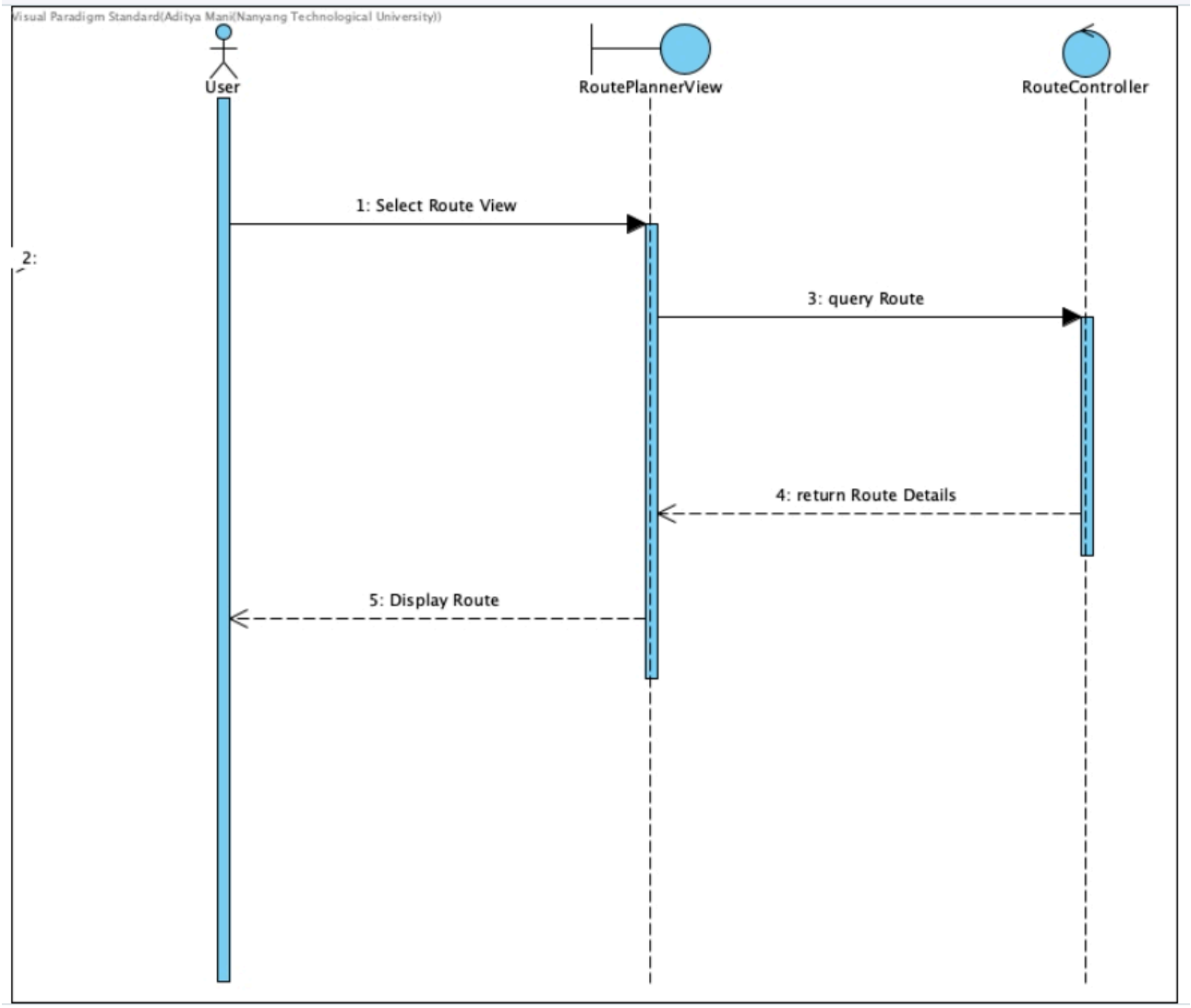
REQ-5: User must receive confirmation of account creation.

4.6 Suggest Route

4.6.1 Description and Priority

This use case retrieves a few best route suggestions from Google Map API. This feature has medium priority.

4.6.2 Stimulus/Response Sequences



4.6.3 Functional Requirements

REQ-1: Based on a selected Todo's destination or during the creation of a Todo, the user can request route suggestions.

REQ-2: The system can use the OneMap API to suggest an optimal path from the user's current location to the destination, considering traffic and public transportation options.

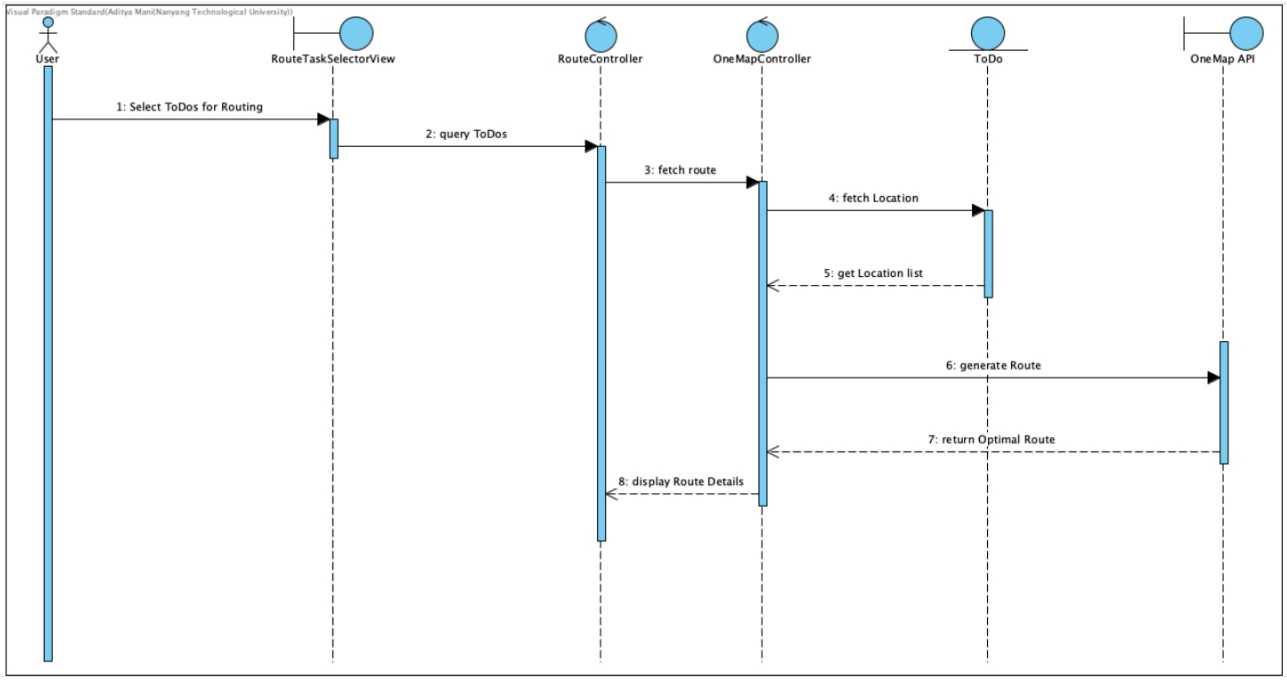
REQ-3: The system is able to display the suggested routes and shows the estimated time taken for each route to the user.

4.7 Plan Route

4.7.1 Description and Priority

This use case retrieves a few best route suggestions from Google Map API. This feature has medium priority.

4.7.2 Stimulus/Response Sequences



4.7.3 Functional Requirements

REQ-1: Based on a selected Todo's destination or during the creation of a Todo, the user can request route suggestions.

REQ-2: The system can use the OneMap API to suggest an optimal path from the user's current location to the destination, considering traffic and public transportation options.

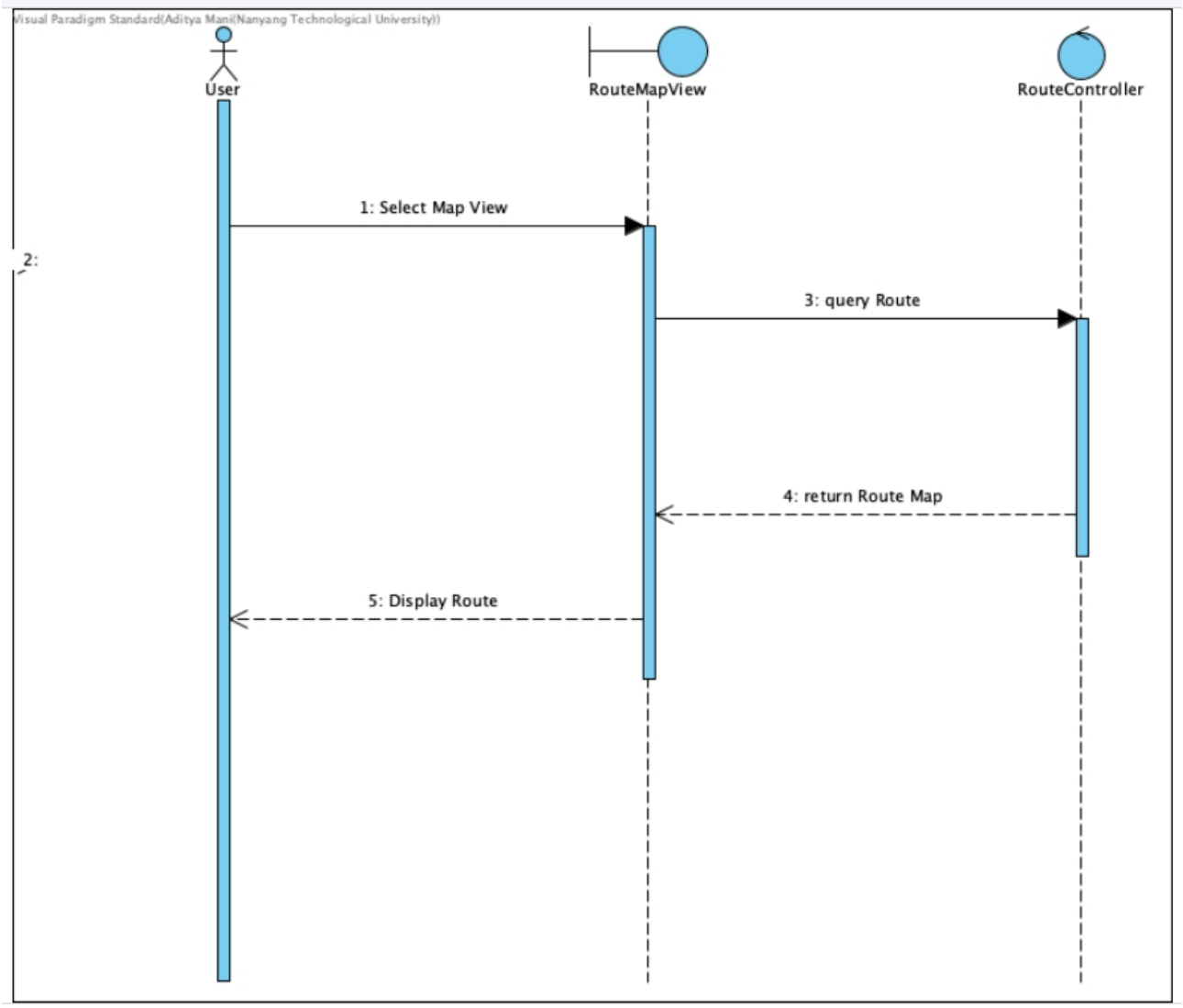
REQ-3: The system must display the suggested routes and show the estimated time taken for each route to the user.

4.8 Route Map

4.8.1 Description and Priority

This use case allows the user to plan a route between the location of the To-dos based on the To-do list he/she has. This feature has medium priority.

4.8.2 Stimulus/Response Sequences



4.8.3 Functional Requirements

REQ-1: The user can select the option to plan a route.

REQ-2: The system must display the dates on which users have created Todos.

REQ-3: Upon selection, the system uses the included "Suggest Route" use case to plan the routes between every continuous two events.

REQ-4: The system must retrieve the destination of the selected Todo and the specified start location.

REQ-5: The system can use “Suggest route” to suggest an optimal path from the start location to Todo's destination

REQ-6: The system must display the suggested routes to the user, including details such as estimated travel time and possible modes of transportation as well as a mini-map.

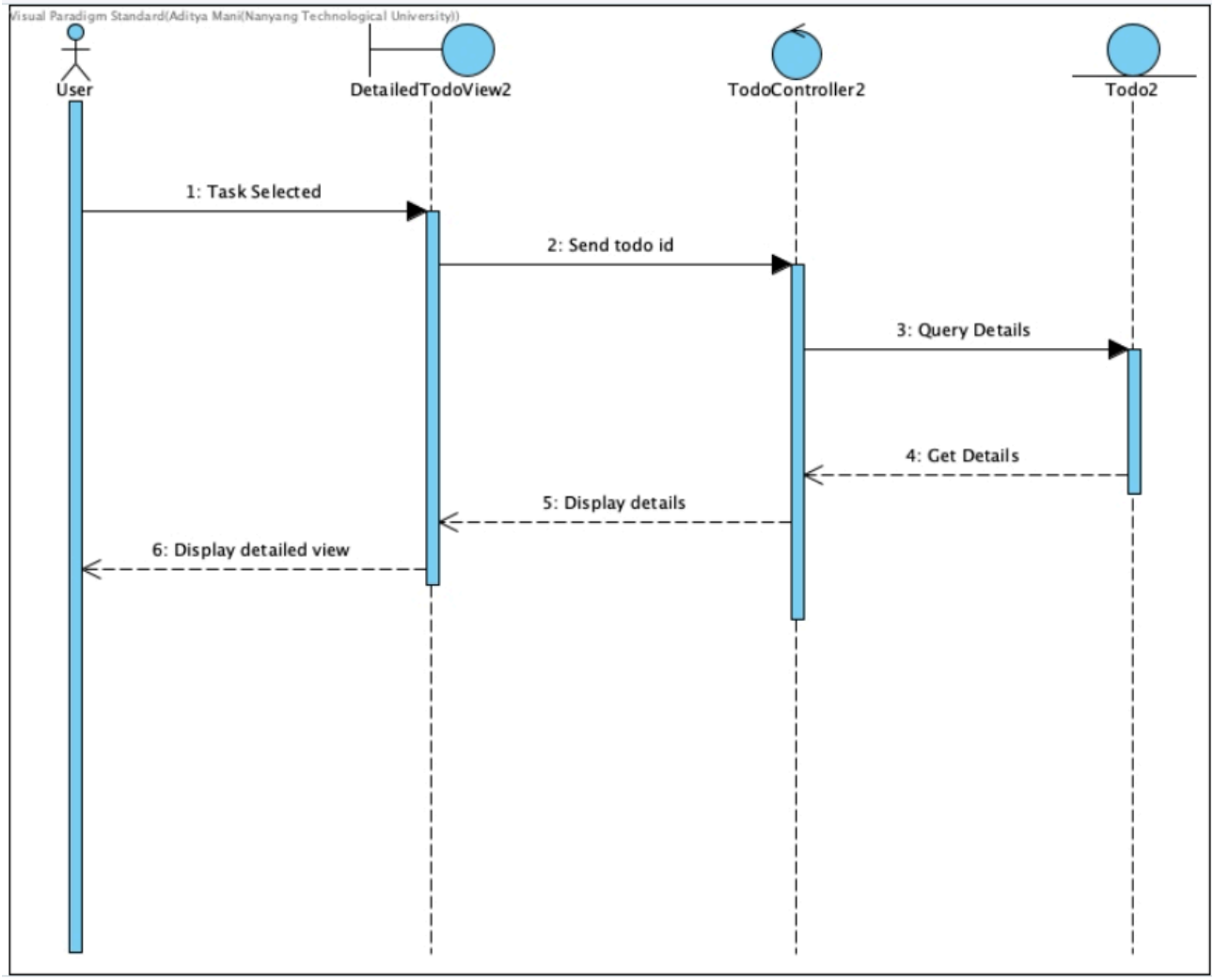
REQ-7: The user can view details of each route by selecting the “next route” and “previous route” buttons.

4.9 View Todo List

4.9.1 Description and Priority

This use case allows the user to view the To-do list. This feature has high priority.

4.9.2 Stimulus/Response Sequences



4.9.3 Functional Requirements

REQ-1: The user can select the option to view the Todos list.

REQ-2: The system must retrieve the Todos list from the database and display it to the user.

REQ-3: The user can select a Todo activity from the list to view its detailed information and edit/delete the specific Todo.

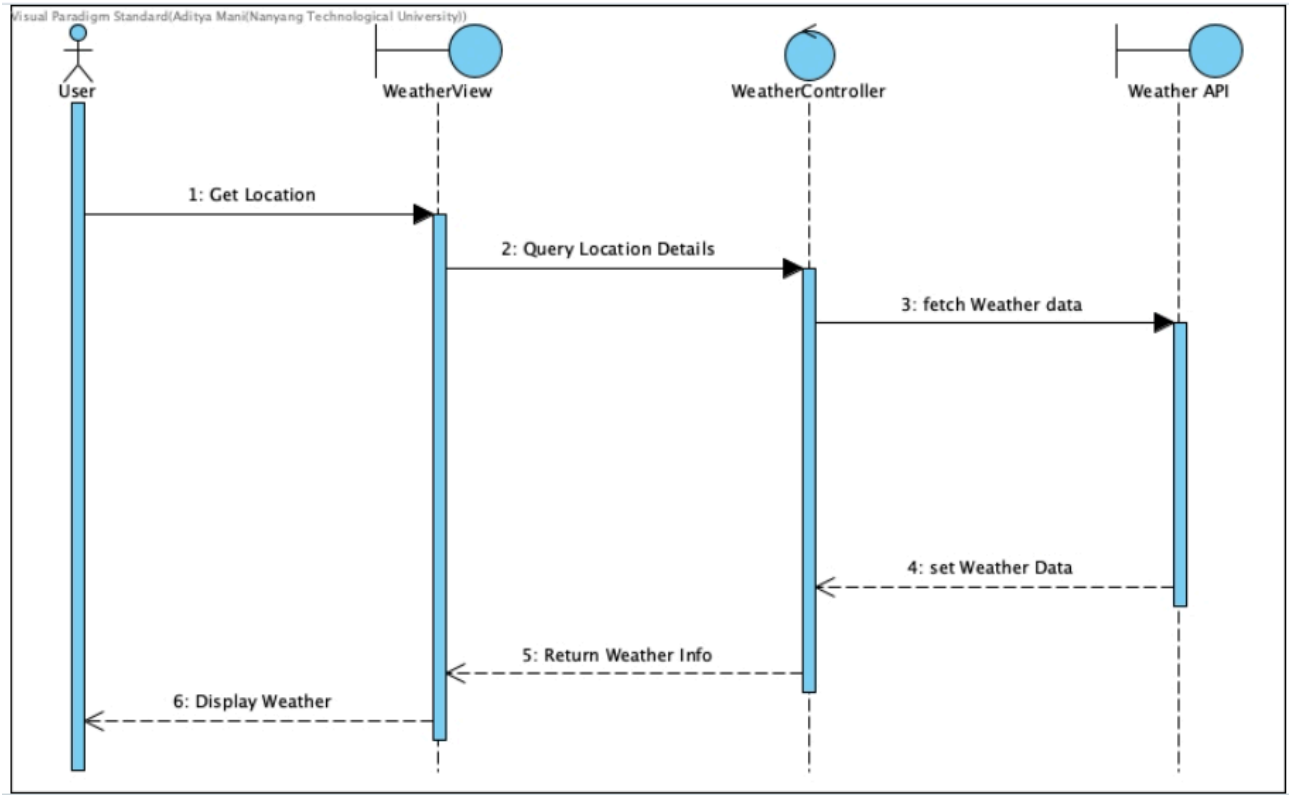
REQ-4: If there are no Todos to display, the system must inform the user that no Todos are available.

4.10 Weather Display

4.10.1 Description and Priority

This use case allows the user to input the timing of To-do. This feature has medium priority.

4.10.2 Stimulus/Response Sequences



4.10.3 Functional Requirements

REQ-1: The system uses the Weather API to obtain real-time weather data for the user's current location and all Todo destinations.

REQ-2: The system must display the weather information to the user or use the information to suggest alternative events or provide alerts.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

5.1.1 The website must load within 3 seconds
5.1.2 The website must respond to the user's actions within 1 second.

5.2 Safety Requirements

5.2.1 The website must clearly inform users of the purpose of collecting location data.

5.2.2 The website must seek permission from users to collect location data.
5.2.3 Users must have the choice to disable location sharing at any time.
5.2.4 Users must have the choice to change their password

5.3 Security Requirements

5.3.1 The website must clearly inform users of the purpose of collecting location data.
5.3.2 The website must seek permission from users to collect location data.

5.4 Software Quality Attributes

5.4.1 Usability
5.4.1.1 More than 95% of users should be able to create a Todo within 10 minutes.
5.4.1.2 Users must be able to edit their Todo after creating one.
5.4.1.3 In-website FAQ guide must be available to address user questions related to the website's functionalities and location settings.

5.4.2 Compatibility
5.4.2.1 Map view must be clearly viewed, thus compatible with a variety of devices and screen sizes (e.g. Mac, Windows, Linux, etc).
5.4.2.2 Schedule view must be clearly viewed, thus compatible with a variety of devices and screen sizes (e.g. Mac, Windows, Linux, etc).

5.4.3 Localization
5.4.3.1 Users must be able to choose their language of preference.
5.4.3.2 Map markers and distance units must be presented in the user's chosen language and regional settings.

5.4.4 Accessibility
5.4.4.1 The map interface must include appropriate labels and alternative text for screen readers.
5.4.4.2 Users with disabilities must be able to interact with the map.

5.4.5 Accuracy
5.4.5.1 The information displayed should be accurate.
5.4.5.2 The events of Todo should be displayed in chronological order.
5.4.5.3 The events suggested by Todo should be displayed within 10km of the user's intended location.

5.4.6 Support
5.4.6.1 Proper documentation must be provided for future developers.
5.4.6.2 The app must include help and feedback buttons for the users.
5.4.6.3 The app must provide FAQs for users who have inquiries.

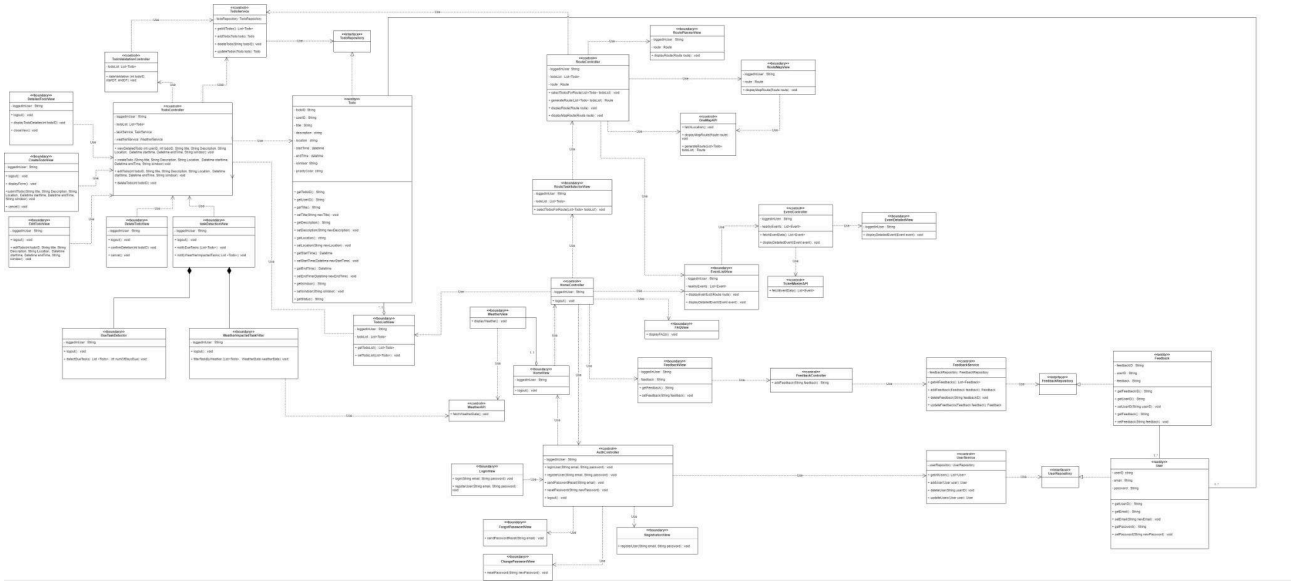
Appendix A: Glossary

Data Dictionary

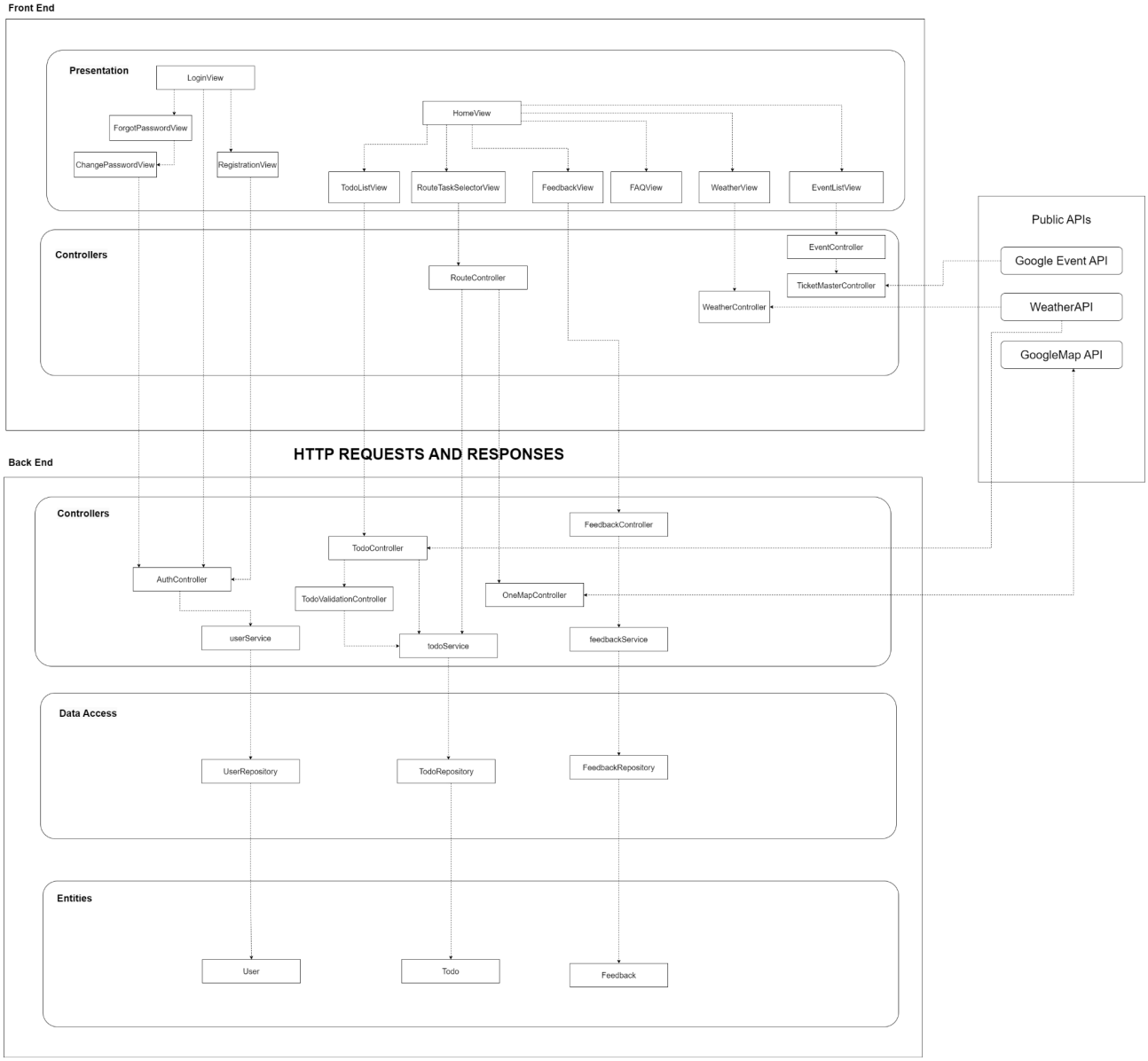
Term	Definition
User	The person currently using the application.
Location Services	A component of the application that can track live locations, view, and place waypoints at locations on a map, and highlight routes to waypoints.
Weather Data	A component of the application that tracks weather situations (updated hourly), along with forecasted weather situations. Data such as temperature, probability of precipitation, and a category corresponding to the weather (e.g. sunny, rainy, cloudy) are provided.
Todo	A unit of information consisting of a Title, Task, Starting Datetime, Estimated End Datetime, Destination, Indoors/Outdoors value, and Route.
Title	The user-given name for a specific Todo.
Task	The user-given description of planned activities for a specific Todo.
Starting Datetime	The user-given date and time a specific Todo starts to be done.
Estimated End Datetime	The user-given date and time a specific Todo is estimated to be finished.
Destination	The user-given destination a specific Todo occurs at.
Indoors/Outdoors value	A value indicating if a specific Todo occurs indoors or outdoors.
Route	The route leading to a Destination of a specific Todo.
Frequently Asked Questions (FAQ)	A list of questions and answers, intended to help users understand how to use a particular function of the application.

Appendix B: Analysis Models

Class Diagram



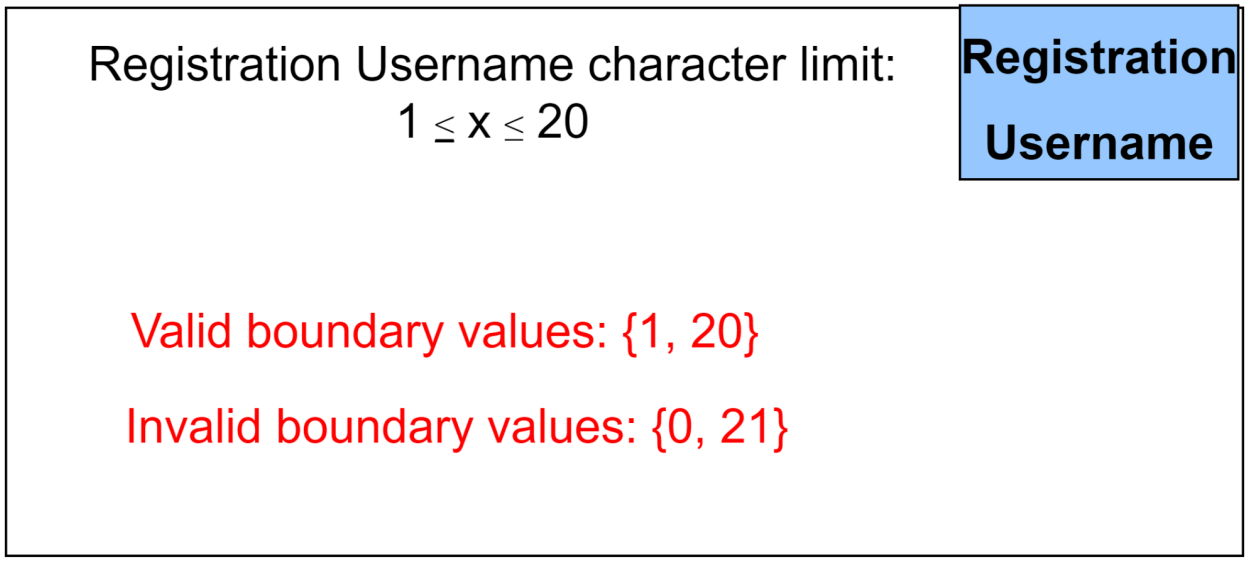
System Architecture

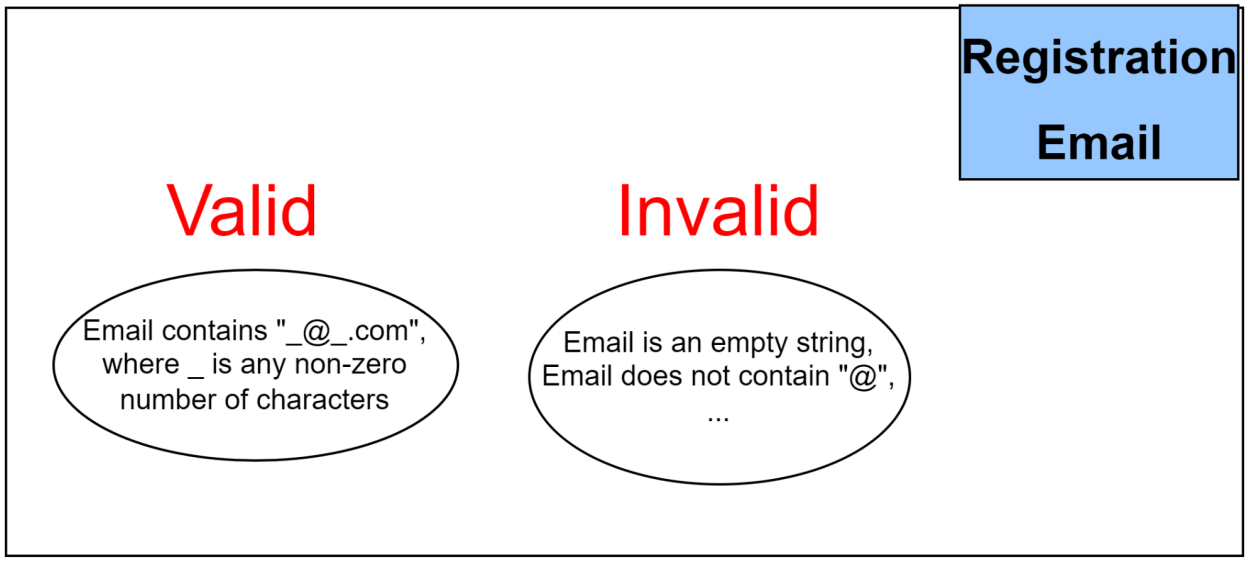


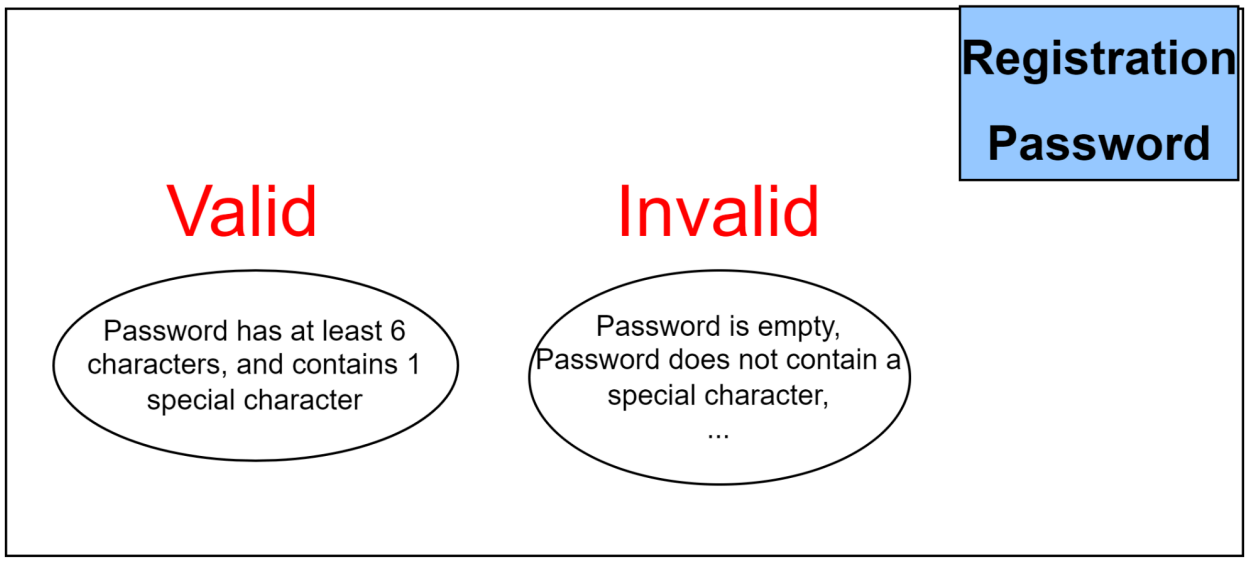
Black Box Testing (Equivalence Class & Boundary Value Testing)

AuthController Class

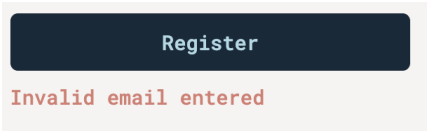
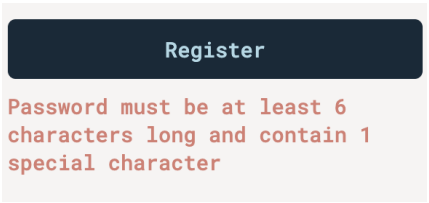
Function: UserAccount Registration





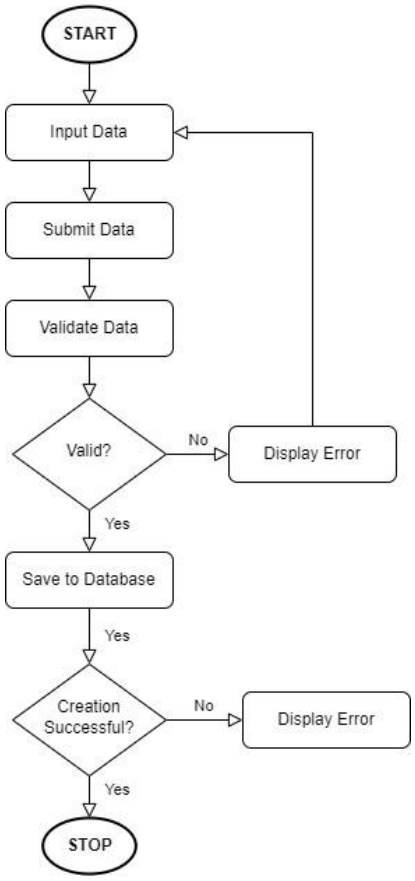


Test Input	Expected Output	Actual Output
Valid Username, Valid Email, Valid Password	New UserAccount is successfully created.	New UserAccount is successfully created. Register Registration successful. Proceed to Login
Invalid Username, Valid Email, Valid Password	Error message: Username must be between 1 to 20 characters	Error message: Username must be between 1 to 20 characters Register Username must be between 1-20 characters
Valid Username, Invalid Email, Valid Password	Error message: Invalid email entered	Error message: Invalid email entered

		
Valid Username, Valid Email, Invalid Password	Error message: Password must be at least 6 characters long and contain 1 special character	Error message: Password must be at least 6 characters long and contain 1 special character 

White Box Testing (Basis Path Testing)

Function: Create Tasks



Cyclomatic Complexity:

Decision Points = 2

CC = 3

Therefore, 3 basis paths to test:

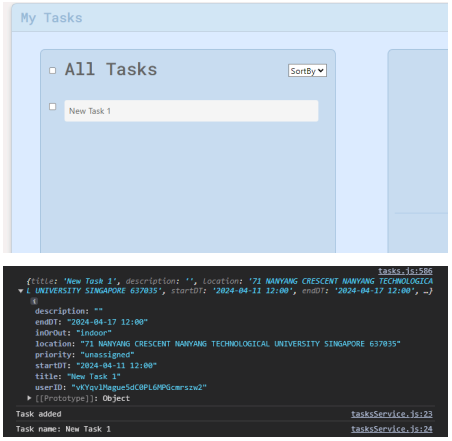
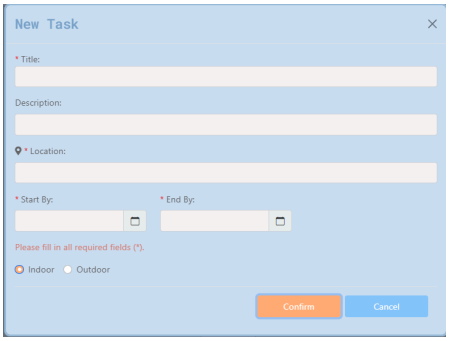
1. Start, Input Data, Submit Data, Validate Data, Valid?, Save to Database, Creation Successful?, Stop (Baseline)
2. Start, Input Data, Submit Data, Validate Data, Valid?, Save to Database, Creation Successful?, Display Error
3. Start, Input Data, Submit Data, Validate Data, Valid?, Display Error, Input Data, Submit Data, Validate Data, Valid?, Save to Database, Creation Successful?, Stop

Test Cases

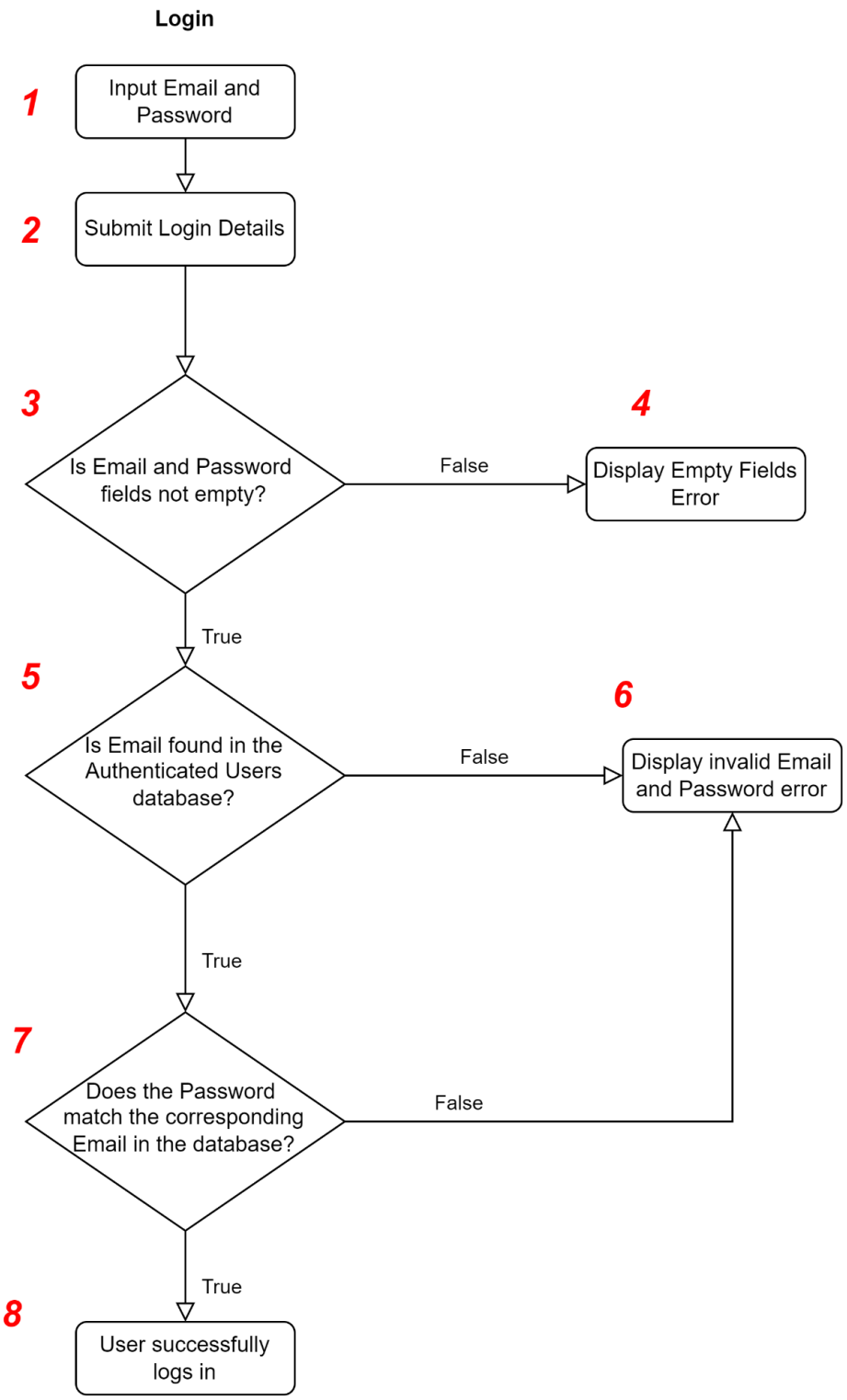
1. Create a Task with Valid Data

2. Create a Task with Missing Required Fields
3. Create a Task with Past Start/End Dates

Results

Test Inputs	Expected Output	Actual Output
Create a Task with Valid Data	The task should be created successfully, and the user should be redirected to the "View All Tasks" page with the new task listed.	
Create a Task with Missing Required Fields	An error message should appear, and the task should not be created.	
Create a Task with Past Start/End Dates	An error message should be displayed indicating the date error, and the task should not be created.	

Function: Create Tasks



Cyclomatic Complexity:
Decision Points = 3

CC = 4

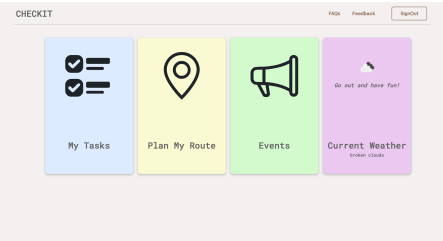
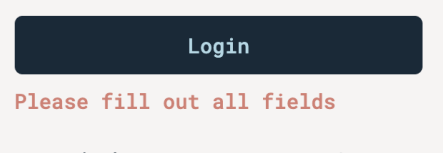
Therefore, 4 basis paths to test:

1. 1, 2, 3, 5, 7, 8 (Baseline)
2. 1, 2, 3, 4
3. 1, 2, 3, 5, 6
4. 1, 2, 3, 5, 7, 6

Test Cases

4. Enter a valid Email and Password
5. Enter a valid Email and empty Password
6. Enter a non-empty Email and Password that has not been registered
7. Enter a non-empty Email and Password, but the Password is incorrect

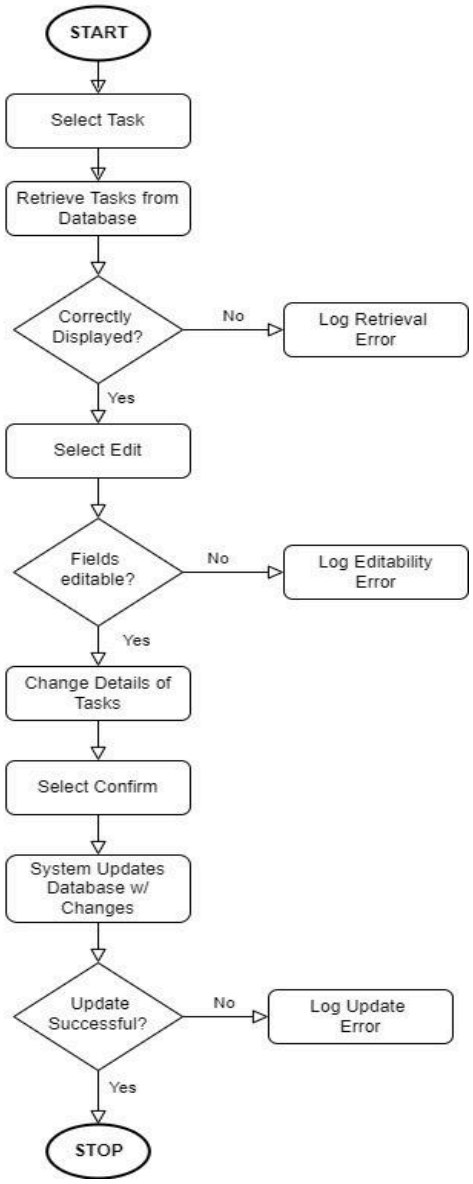
Results

Test Inputs	Expected Output	Actual Output
Enter a valid Email and Password	The user is successfully logged in and brought to the homepage.	The user is successfully logged in and brought to the homepage. 
Enter a valid Email and empty Password	Error message displayed: Please fill out all the fields	Error message displayed: Please fill out all the fields 

Enter a non-empty Email and Password that has not been registered	Error message displayed: Invalid email or password	Error message displayed: Invalid email or password Login Invalid email or password
Enter a non-empty Email and Password, but the Password is incorrect	Error message displayed: Invalid email or password	Error message displayed: Invalid email or password Login Invalid email or password

White Box Testing

Function: View Detailed Task & Edit Function

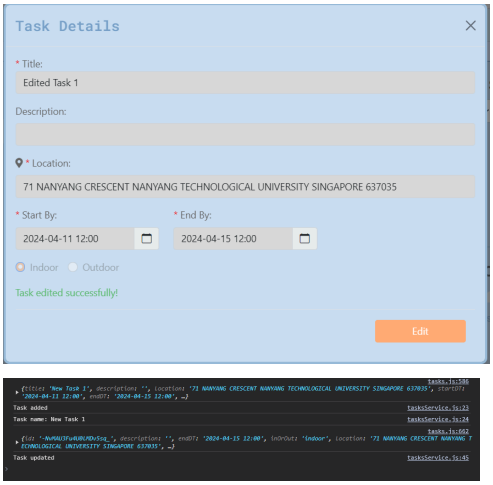
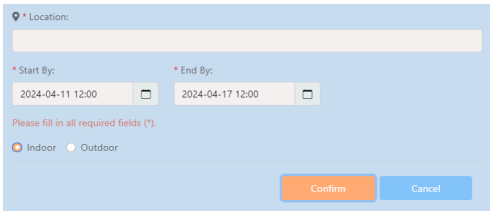
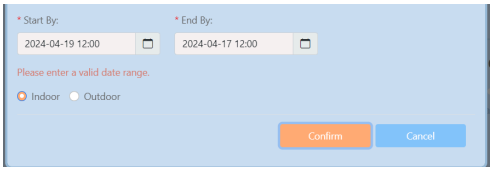


Test Cases

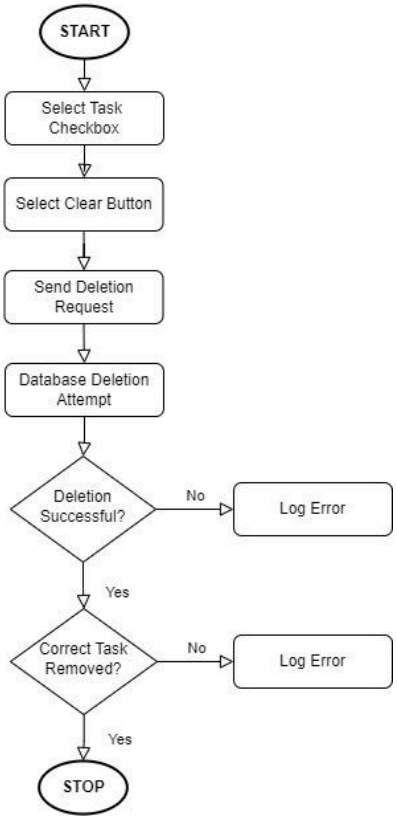
1. Edit Task with Valid Data
2. Edit Task with Missing Required Data
3. Edit Task with Invalid Date Format

Results

Test Inputs	Expected Output	Actual Output

Edit Task with Valid Data	Changes are saved, and the task is updated with new details.	Edited: New Task 1 → Edited Task 1 
Edit Task with Missing Required Data	An error message appears, and the task remains unchanged.	
Edit Task with Invalid Date Format	An error message is displayed, and the task is not updated.	

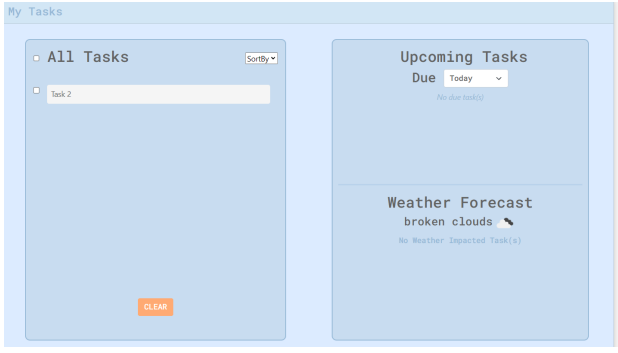
Function: Delete Task

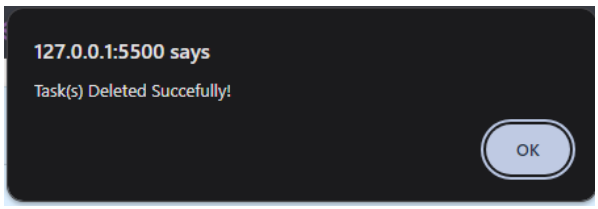
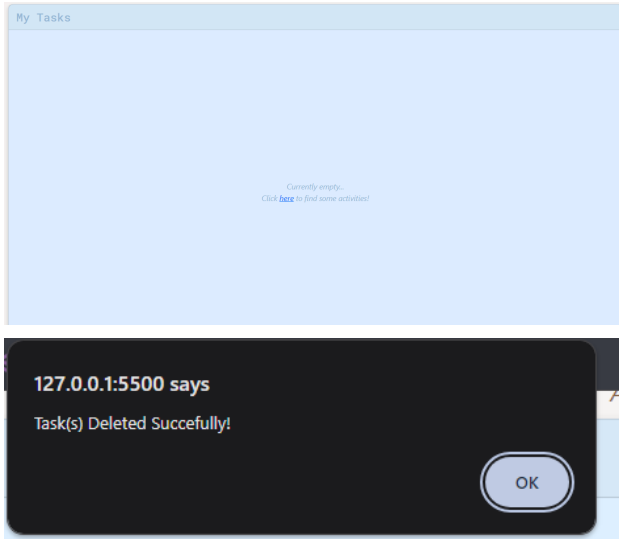
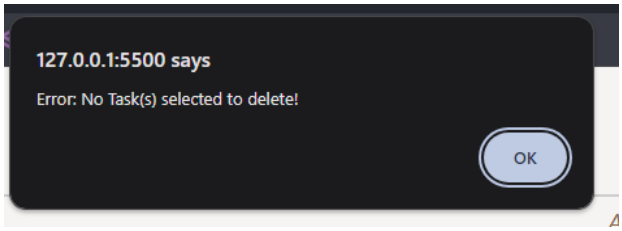


Test Cases

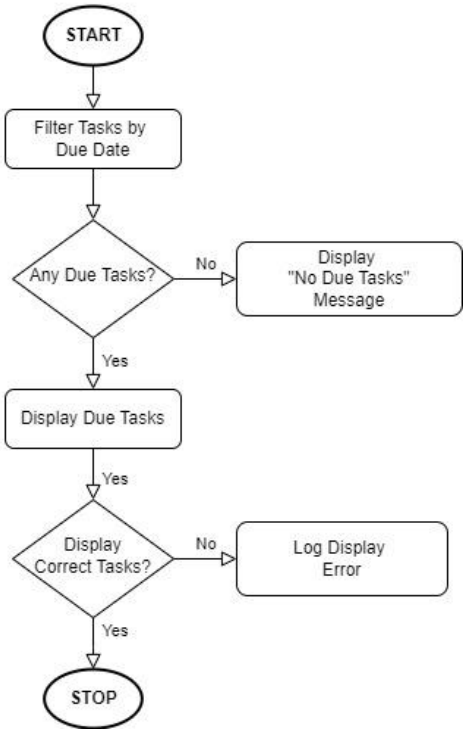
1. Select Single Task to delete
2. Select ALL tasks to delete
3. Click Delete without selecting any tasks

Results

Test Inputs	Expected Output	Actual Output
Select Single Task to delete	The task is deleted and no longer appears in the task list.	Deleted Task 1 

		
Select ALL tasks to delete	All selected tasks are deleted.	
Click Delete without selecting any tasks	An appropriate error message is displayed, and the task list remains unchanged.	

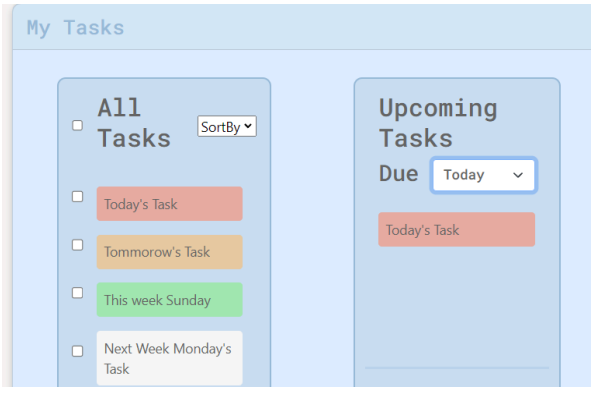
Function: Show Due Tasks

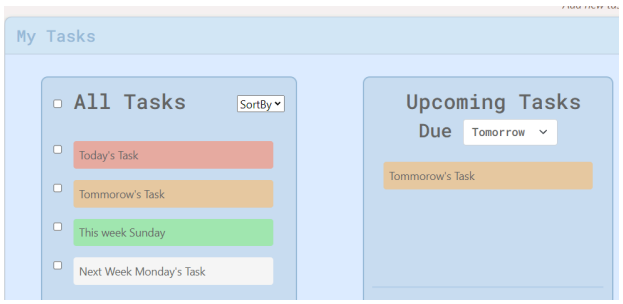
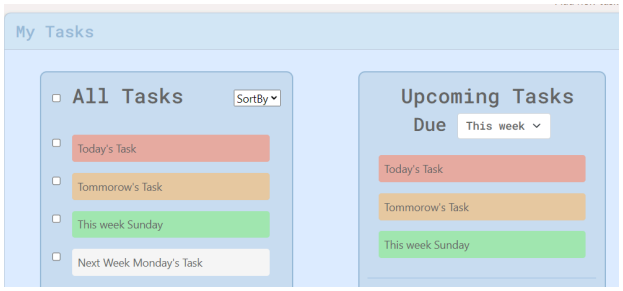
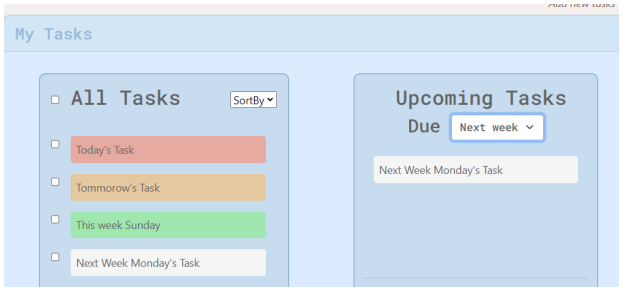
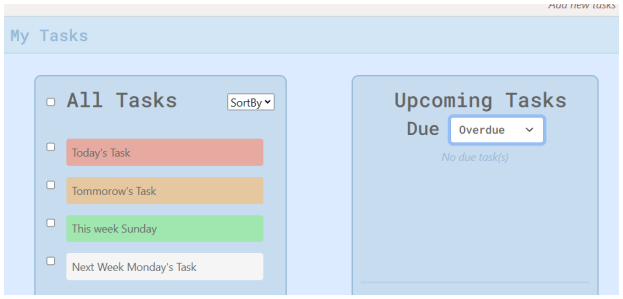


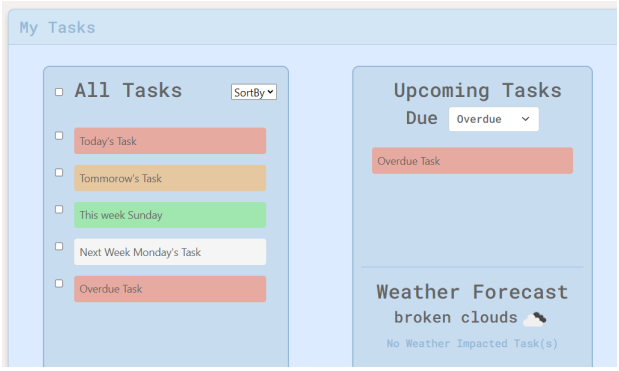
Test Cases

1. Display Tasks Due Today
2. Display Tasks Due Tomorrow
3. Display Tasks Due This Week
4. Display Tasks Due Next Week
5. Display Due Tasks When There Are None
6. Display Overdue Tasks

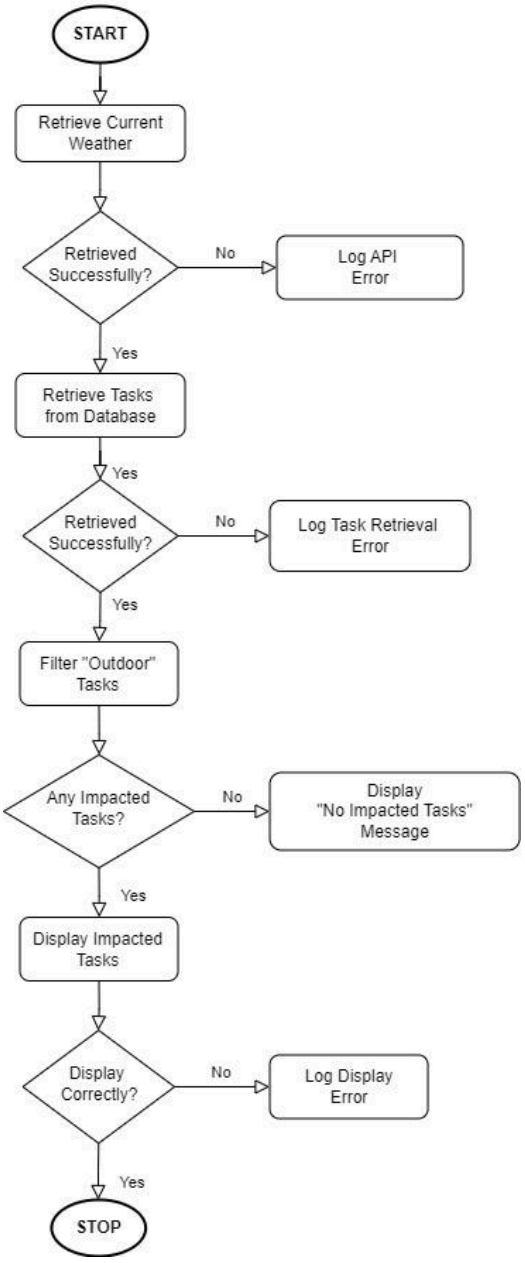
Results

Test Inputs	Expected Output	Actual Output
Display Tasks Due Today	All and only tasks due today are displayed.	

Display Tasks Due Tomorrow	All and only tasks due Tomorrow are displayed.	
Display Tasks Due This Week	All tasks due in the current week are shown.	
Display Tasks Due Next Week	All tasks due in the next week are shown.	
Display Due Tasks When There Are None	A message indicating "No Due Tasks" is displayed.	

Display Overdue Tasks		

Function: Display Impacted Weather Tasks

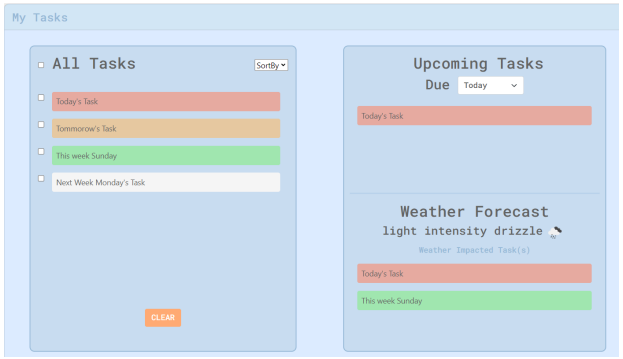
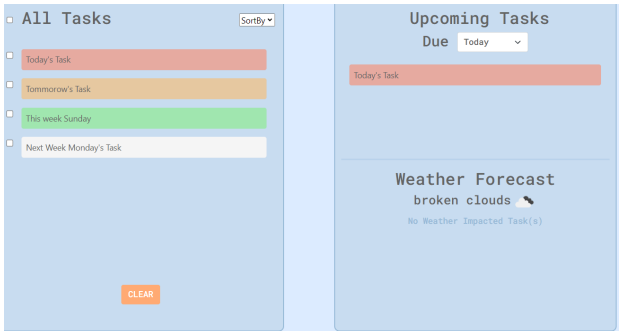


Test Cases

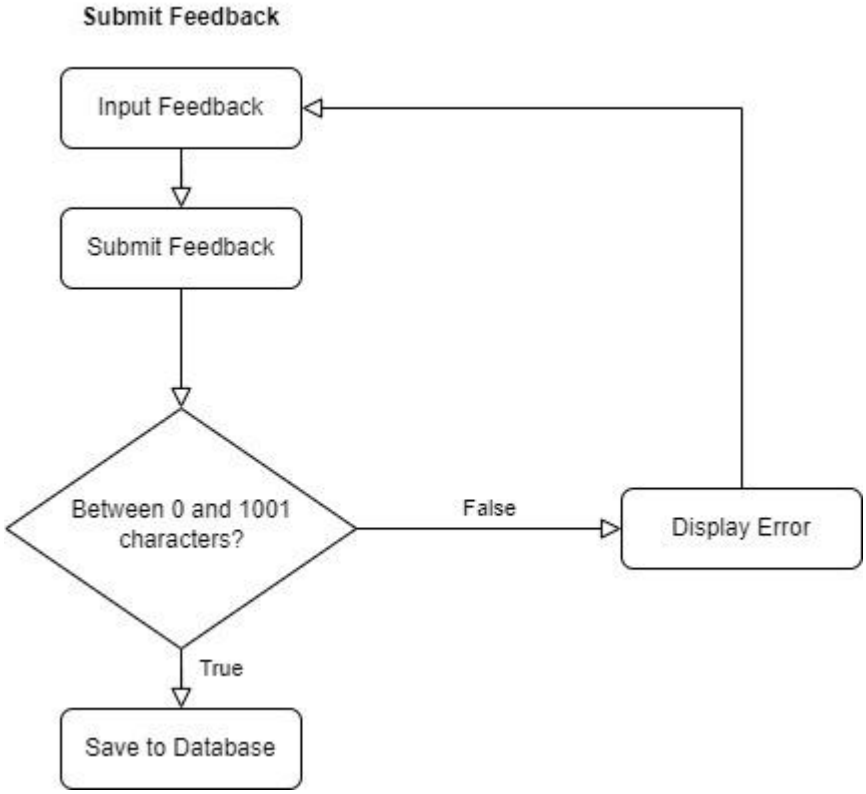
1. Display Currently Active Tasks Affected by Bad Weather Conditions
2. Display Weather Impact When There Is No Adverse Weather

Results

Test Inputs	Expected Output	Actual Output

Display Currently Active Tasks Affected by Bad Weather Conditions	Any active tasks that are affected by the bad weather conditions will be displayed	
Display Weather Impact When There Is No Adverse Weather	A message indicating "No Weather-Impacted Tasks"	

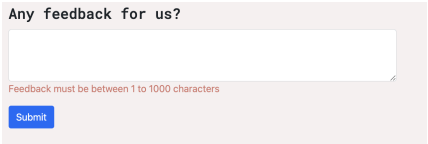
Function: Inputting Feedback



Test Cases

1. Input feedback of 0 characters
2. Input feedback of 1000 characters

Results

Test Inputs	Expected Output	Actual Output
Submit feedback of 0 characters	Error message: Feedback must be between 1 to 1000 characters	Error message: Feedback must be between 1 to 1000 characters 
Submit feedback of 1000 characters	Feedback is accepted	Feedback is accepted