Group Travel Assistant Application

COMP8047 – Major Project

Liang Li – A01072475
9-27-2024

Table of Contents

1. Student Background

The author of this proposal is Liang Li. He moved to Canada in 2018. Before he came here, he had worked as a software engineer for about 10 years, and currently he has continued his professional occupation in Canada for almost 5 years already. Since he wants to continue pursuing local education, Liang started his CST BTech program in 2020.

1.1. Education

BCIT - CST BTech (part time), Sep 2020 to present

Nankai University - Software Engineering (part time), Mar 2013 – Jan 2015

Tianjin University of Technology - Electronic Information Engineering, Sep 2004 – July 2008

Yaohua High School - Sep 2001 – July 2004

1.2. Work Experience

ZEMA Global Data Corporation, Vancouver, BC Canada, Software Developer, May 2023 – present

Autozen, Vancouver, BC Canada, Full-stack Developer, Dec 2020 – Mar 2023

Kefelan Solutions, Vancouver, BC Canada, Backend Software Engineer, Jan 2019 – Dec 2020

Chinanx Financial Services Group, Tianjin, China, Senior Software Engineer, May 2017 – Dec 2017

Tansun Technology Co., Ltd, Beijing, China, Senior Software Engineer, Jul 2013 – May 2017

ISoftstone, Beijing, China, Software Engineer, Jul 2008 – Jul 2013

2. Project Description

This project is the major project of BCIT CST BTech program. Liang will participate in the entire SDLC including proposal, design, development, test and deployment individually.

The goal of this project is to build a web-based Group Travel Assistant application named Let's GO designed to simplify and centralize the management of planning travel, organizing travel group members, polling for making decisions, communication and summarizing expenses. (In the report proposal below, the application will be called Group Travel Assistant). It is used as an assistant tool for users to make plans and gather a group of members to attend activities like travelling, watching concerts or soccer games etc. It is also a tool to conduct efficient communication during the process. Regarding the name of the application: the "us" in "Let's" can either mean the user and their close friends if they want to use it for a private travel or other new friends that share the same interests; since this application is used for managing travel, "GO" literally means "go out and have fun". However, the "travel" here means not only a trip to another country, but also a brilliant soccer game, a famous singer's concert or a family time on cruise.

The application allows users to create customized travel, games or show plans and connect with other travelers who share the same itinerary and interests. If they want to set the travel plan as "private", only invited members can join the plan; but if the user wants to publish their plan to "public", other users who want to join the travel and know new friends can apply to the plan and get invited if the creator accepts the application.

The application offers multiple features. Users can create travel plans based on several criteria, such as the topic of the plan, the origin/destination of the travel, transport, the gender or age limitation of the members, number of the members, communication languages, estimated budget and optional expenses. Users can be the owner of a new travel plan and invite specific user to the plan if the plan is set to "private"; they can also publish their plan if the plan is set to "public", so that other applicants who meet the requirements can apply and get invited into the plan. Users can be applicants of other users' plans filtered for them and those they are interested in. After the plan is finally settled, plan members can start to prepare their trip. During the trip, plan members can create polls using the polling feature to make decisions efficiently. Real-time communication can be achieved in the chat room feature. Expense features can make notes and split shared expenses during the trip, so that plan members may know the expense details and how much they spend clearly.

By leveraging functionalities such as Travel Plan searching, polling system, real-time collaboration tool and shared expenses management, the application is an efficient and centralized tool that could enhance the user travel experience significantly.

3. Problem Statement and Background

Traveling is one of the most popular activities globally, allowing individuals to explore new cultures, environments, and experiences. However, organizing travel for a group can often be a cumbersome and complex process, particularly when coordinating among multiple travelers to make the plan, vote for decisions, efficient communication and sharing expenses etc. In today's fast-paced world, many travelers seek more than just a destination. Sometimes, travelers don't have a specific target destination. Since the best season to visit a place is different from others, people usually want to find the best places to go at that time. Besides, there're variety of games, concerts, events and shows happening all around the world, lovers and fans want to join their community to watch the game or enjoy the show together. There are some problems when people want to travel or attend some events with others, and this application is the tool to help solve these problems and facilitate travel experience.

The first problem that can be addressed here is how to find people with the same interest or organize a travel group efficiently when people are preparing for their trip. From my personal experience, I love travel, but there're many places that I have never been to before. I would love to see people's comments and their sharing about the experiences. Some travel lovers would like to organize group travel with other people who share the same interests and prepare for the trip very well before they go. New travelers like me would consider following and joining the group to visit the destination that I have been eager to see. Another example, I am a soccer lover, so most often, I would like to find the fans of my supporting club and watch the game together. Therefore, the travel plan feature of this application helps travelers to create the travel plan before they go, so that the plan members can prepare the travel in advance, get together and have fun. No matter who you want to invite, you can either add people that you know into the plan or share the plan with the public so others can join. In the application, for public plan, many criteria can be set when the owner creates the plan. For example, the plan type can be a trip, an NBA game or concert, so applicants can search for their desired type of plans efficiently with plan type and title. Besides, the more detailed information the owner provides, the more attractive the plan is to the potential applicants. Requirement filter is also a useful feature of tailoring the desired applicants. The owner can set limitations of the group size, gender, age and spoken language etc. Lastly,

the origin and destination, type of transport, accommodation and other potential events can be added to the plan, so people can always know clearly if this is a suitable travel plan for themselves before they apply for it. After the plan has been created, all members are filled, they can start to purchase their flight tickets, book the hotel etc. offline, and get started based on the schedule.

Secondly, efficient communication is essential during group travel. Without this application, travelers usually need to add each other's number into their contact, so they would use instant messaging applications to communicate. However, not everyone is willing to share their personal information with others, especially new friends that they meet for the first time. Therefore, dedicated communication tools are useful and safe to use specifically for certain activities. Real-time collaboration during the trip or event is also an important feature that this application can provide. The polling system allows plan members to create polling questions like: what are we going to eat tonight? A. Hamburger B. Pizza C. Hamberger and Pizza. There's an active time range for the poll, and all members can receive notification to attend the poll within the effective time range. They can select their answer, so that it is a fair and fun way for the group to make decisions. Besides, real-time communication can be conducted in chat room function. All plan members can chat simultaneously or leave messages to others. Users can receive email notifications if they have unread messages left in the chat room.

Lastly, except for the fixed budget for the trip like expenses for transport, accommodation, food and tickets etc., there are many other potential expenses during the trip. Some of these expenses are shared among group members, such as sharing taxis, having a meal within the group or buying more souvenirs to get discounted. Using the shared expense feature, travelers don't need separate applications to make notes and calculate each member's share of the expense. This function allows the user to make notes of the shared expenses like the purpose and total amount and select the members involved. Then it can calculate and display the portion of each corresponding member, so it's very easy to use and efficient to track the shared expense. Thus, one member can always do the payment in advance, then the others give him/her their portion based on the record.

This application does not involve any payment or transaction system, so users still need to use other third-party websites to book flights, hotel and tickets etc. However, it is a very user-friendly application for travelers to find interesting travel plans, manage group members, communicate and collaborate effectively and make notes for shared expenses.

4. Scope and Depth

The scope and depth of this project encompass multiple dimensions, integrating various technologies while addressing specific problems faced by travelers. This section defines the boundaries of the project and highlights the key areas of focus.

4.1. Scope and depth of the Project

The minimum scope of the project is to create an online web-based application that allows users to create or find a travel plan that they are interested in before they start to prepare for the trip. Multiple members can join the plan so they can travel together. After the trip has started, they can use this application as an efficient communication and collaborating tool with other plan members.

4.1.1. Functional requirement

Users can create an account in the application using the register function. Email of the account should be unique within the application, and it can also be used to receive system notification, such as unread message. The other account profile information includes first name, last name, gender, age, phone number, main language, city, country etc.

Users can login with the account to use the application.

Travel Plan is the core concept of this application. There are a few "plan status" of the Travel Plan:

Created ——> In progress ——> Completed

└> Cancelled

"User-plan status" is used to indicate the relationship of current user and the Travel Plan:

Owned, Applied —> A-accepted, Invited —> I-accepted

└——> A-refused └——> I-refused

There are two main sections of the application:

(1) Travel Plan List:
   a) There's a search bar that allows users to search related Travel Plans using keywords
   b) users can browse and apply the Travel Plan that they are interested in if they haven't created a new Travel Plan, and the plan status is "created/in progress"; the plan status should all be "created" inside the Travel Plan List
   c) only one Travel Plan can be applied by current user at the same time, and it will be listed in "Pending" sub section under "My Plans" section; plan status is "created", user-plan status is "applied"; if he owner of the Travel Plan accepts or refuses the application, the user-plan status changes to "A-accepted/A-refused"; if it's "A-accepted", then current user can wait for the Travel Plan to get started, if it's "A-refused", the current user cannot see the Travel Plan anymore.
   d) if current user has created a Travel Plan, it will be listed in "Pending" sub section under "My Plans" section; the user can only create a new Travel Plan if they haven't applied any other plan and the plan status is in "created/in progress"; if the plan type is "public", it will be also listed in the Travel Plan List so that other users can apply; plan status is "created", user-plan status is "owned"
   e) if current user has been invited to join an existing Travel Plan by another owner, the Travel Plan will be listed in "Pending" sub section under "My Plans" section; the user can accept or refuse the invitation, then the user-plan status will be changed to "I-accepted/I-refused" correspondingly
(2) My Plans: there are three sub sections under this section
   a) Active: there can be zero or only one Travel Plan listed in this sub section; the plan status is "in progress", user-plan status is "owned/accepted"; the Travel Plan can be completed automatically by system when it has reached the scheduled end time, then the status changes to "completed" and the Travel Plan goes to History sub section

b) Pending: there can be zero or only one Travel Plan listed in this sub section; the plan status should be “created”; the owner can either accept or refuse any application, or remove and add new members, if the Travel Plan has not been filled with members, when it has reached two days before the start time, the Travel Plan will be cancelled automatically by system, then the plan status changes to “cancelled” and the Travel Plan goes to History sub section; the owner can also cancel the Travel Plan any time before the Travel Plan has been started; after members are filled, the Travel Plan can be started automatically by system when it has reached the scheduled start time, then the plan status changes to “in progress” and the Travel Plan goes to Active sub section
c) History: there can be multiple plans in “cancelled/completed” status
d) Create function is used by users to create a new Travel Plan
e) the user can either create or apply only one plan at the same time; so that at most only one plan can exist in both “Created” and “Applied” sub sections

Inside the Active sub section, users can see the current Travel Plan in progress. Then there are three main functions that can be used for this Travel Plan, after joining the Travel Plan (created, been invited/accepted):

(1) Chat Room: all members can start to send and receive messages inside the Chat Room,
(2) Polling: all members can use the Polling function to create questions or answers,
(3) Shared Expense: all members can add shared expense items in Share Expense function,

no matter what plan status the Travel Plan will change, so it can be used as a group activity channel in the future.

4.1.2. Non-functional requirement

Non-functional requirements are also essential to this application. There are a couple of aspects that are considered when implementing this application.

Performance: the system shall respond to 95% of user requests within 2 seconds under normal operating conditions; there should be 99% of the up time over a rolling 30-day period

UI/UX: the user interface should be easy to use, and there are always buttons that can be used by user to navigate to other page; reasonable warnings or notifications should be shown to user when their usage is not correct or there’s risk of losing data etc.

Date security: although the banking system is not used or stored in the application. Sensitive user information such as personal information should be stored and secured in databases; password should be encrypted as well.

4.2. Content or features out of scope

Due to resource limitations, such as time and development workload, some of the functionalities cannot be implemented in the first version of this application. However, it is always good to think of any improvements or more features to make the application better. Some of content or features out of scope will be mentioned below:

(1) Only text messages are allowed in the Chat Room function now; therefore, no pictures, audios, videos or other media are supported

(2) No transaction, payment, banking APIs or other external booking systems are involved in the development of this application
(3) Multilingual support is not available in this version, only English is used in logo, text or UI design for developing application
(4) Implementing features to manage multi-currency conversions, taxes, or complex financial tracking is beyond scope. That means only Canadian Dollar is supported in the Shared Expense function.

5. Test Plan

The detailed test plan for this project outlines the procedures, methodologies, tools, and criteria to ensure that the application meets its functional and non-functional requirements. The aim is to thoroughly test all components of the system, and the whole function should work without bugs.

5.1. Test Procedures and Methodology

5.1.1. Types of Test Methodology

Unit Test: Usually, unit test is being used during development. It focuses on testing individual component or functional module (e.g., when the user creates a new Travel Plan, based on the type and other properties, the corresponding status should be displayed correctly, and the Travel Plan should exist in the correct section in the UI)

Integration Test: Ensure that all modules of the application work together seamlessly.

Regression Test: Ensure that after new changes or features are added into the application, they do not negatively impact the existing functionality of the application.

User Acceptance Test (UAT): Engage end-users to test the application's usability, functionality, and performance to ensure it meets their needs.

5.1.2. Test Tools and Frameworks

JUnit/Spring Test Framework: For unit and integration testing of the back-end code in Spring Boot.

Postman/Swagger: For testing APIs and verifying request/response handling.

Manual testing: To simulate the real scenario of a new user to use this application.

5.1.3. Verification Criteria

Functional Requirements: All test cases related to the core functionalities must pass successfully, including account creation, Travel Plan creation, invite or accept new member to the Travel Plan, Polling function, Chat Room function and Shared Expense function etc.

Performance: The application should handle at least 1000 concurrent users without significant performance degradation (response time < 2 seconds).

Security: No critical vulnerabilities should be present. Sensitive data transmissions should be encrypted.

5.1.4. Pass/Fail Criteria

Pass: The test case is marked as a pass if the application performs the expected functionality without errors or deviations.

Fail: The test case fails if the application does not meet the expected outcome, behaves unexpectedly, or encounters any critical errors.

5.2. Test Cases

Description	Test Case	Passing Criteria
Test new user registration	Click the register button to be redirected to the register page; fill in all required and non-required information; click Submit button	If all information is correct, there should be a successful notification shown to the user
Test duplicate Email	Email should be unique within the application; entering an existing Email address when registering new account, a warning should be shown	If the Email being submitted is same as anyone Email address in the application database, a warning should be displayed to user
Test user login	After the user enters the Email and corresponding password, the page should be redirected to the main page, the Travel Plan List page	If the username and password are correct, after login, the page should be directed to the main Travel Plan List page
Test search bar in Travel Plan List	Keyword can be used in search bar to search for related Travel Plans	The user can enter any keyword, and the keyword can be used in different criteria to search for related Travel Plan results
Test user to create new Travel Plan	User can create a new Travel Plan when the user is not applying any Travel Plan	If the user hasn't applied to any existing Travel Plan, the user can create a new Travel Plan with Create button
Validate the Travel Plan displayed in the Travel Plan List is based on filtered criteria	Any Travel Plan that can be seen by current users in Travel Plan List should match current user's profile, such as gender, age and language etc. if the Travel Plan has specified these properties	If the Travel Plan has specified gender, age etc. so only the correctly filtered Travel Plan based on current user's profile can be displayed to current user in Travel Plan List
Validate User can apply for only one Travel Plan from Travel Plan List	User can apply for only one Travel Plan from Travel Plan List; if there's already one Travel Plan in Pending or Active section, then it	If there's no Travel Plan listed in Active/Pending List, the user can apply for at most one another Travel Plan

	doesn't allow user to apply for other Travel Plan	
Validate Invitation function	Owner of a Travel Plan can invite any other user by email; the invited user can accept or refuse the invitation; the invitation can only be sent to user without Pending/In progress plan, otherwise, the owner will receive warning message	After the invitation has been sent by the owner, an email notification should be sent to invited user, and a Travel Plan should be displayed in Pending sub section of the invited user; but if the invited user already has Pending/In progress plan, the invitation cannot be successful, then a warning will be displayed to the owner
Test Polling function	Users can create a new Polling item; users can create question and answer options for the Polling item	Users can create a new Polling item by adding new questions and answers inside the Travel Plan
Test Chat Room function	Users for the Travel Plan can send/receive messages in Chat Room function	Users can send/receive messages in the Chat Room function; if there are unread messages, Email notification will be sent to User's Email address
Validate Share Expenses function	Initiate a payment of $100 among four plan members	Each of the members should be noted as $25 for this shared expense item
Validate data change can be synchronized among users	For example, if the owner of the Travel Plan has invited another user to the Travel Plan	Existing members can see the newly invited user to join the Travel Plan
Test for SQL injection vulnerability in form submission	Attempt SQL injection attack via login form input	Application does not execute malicious queries

This test plan covers a comprehensive approach to validating the functionality, performance, security, and usability of the application. By systematically testing each component and feature, the project aims to deliver a robust, user-friendly, and secure travel planning solution.

6. Methodology

This project will utilize a comprehensive methodology and approach to ensure a well-structured development process. The primary focus will be on building a responsive, secure, and user-friendly web application using modern technologies. Below is a detailed description of the methodology, approach, and technologies that will be used in the project.

6.1. Methodology

Waterfall methodology will be applied in the development process of this application. It is surprisingly beneficial for personal solo projects, especially if the project has a clear goal and a defined scope.

For this application, the requirements, timelines and deliverables have been planned in this project proposal before the development. Therefore, Waterfall methodology can help the developer to focus on the development and reduce scope creep. There are two main sections of this application, the “Travel Plan List” and “My Plans”, and under “My Plans” section, there are three sub sections namely “Active”, “Pending” and “History”. Since the detailed functionalities of each section have been confirmed, the scope should not be changed during the development. Therefore, using Waterfall methodology will make it easy to estimate how long each function needs and what resources will be required. Besides, the goal is clear, so it becomes easier that the result can be achieved within predictable timeline.

6.2. Approach

6.2.1. Requirement Analysis and Planning

Conduct in-depth analysis to gather functional and non-functional requirements from potential users and stakeholders. This step has been completed during the requirement design process. Based on surveys from potential users, the final features of the application have been confirmed and designed before development. Basically, this application is used as a tool during two different phases of an activity, commonly referred to travelling, attending concerts and games etc. The first phase means planning and creating the activity group, and the second phase means the process after the activity has started. All functionalities are based on common software design and gathered from user stories.

6.2.2. Architecture Design

Monolithic Architecture is applied in developing this application. Compared to Microservice Architecture, Monolithic Architecture is easier to design, develop and deploy. Everything is contained in a single codebase, and there’s less setup. It is faster when testing and debugging the application since the developer doesn’t need to mock other services and set up inter-process communications. Because this is a personal project with relatively few features, using Monolithic Architecture allows the developer to focus on developing business logic and requirements and avoid unnecessary bugs.

6.2.3. Core Functions and Algorithms

The “Travel Plan List” is one of the core features of this application. In order to display proper Travel Plans in the list, current user’s profile data will be used as input parameters when displaying the desired plan list. Based on the input parameters, such as gender, age, language, location and other preferences, the algorithm will query the Travel Plan table to get matched data. If some of the optional information is not provided, the query will ignore those criteria.

As mentioned about, two dimensions of status will be applied on the Travel Plan: “plan status” and “user-plan status”. There is a “user-plan relationship” table used to connect user and relative Travel Plans. In this table, there is a column named “user-plan status”. Different

combinations of these two statuses will be the results of different operations on the Travel Plan. For example:

(1) Newly created Travel Plan by current user: p status: created; u-p status: owned
(2) Applied Travel Plan that has not been started: p status: created; u-p status: applied
(3) Refused Travel Plan that has been cancelled: p status: cancelled; u-p status: refused
(4) Accepted Travel Plan that is in progress: p status: in progress; u-p status: accepted

WebSockets will be Utilized to enable real-time communication among users when sending and receiving messages in the Chat Room.

The scheduled job could be run to scan databases and automatically start and complete Travel Plans.

Build a responsive front-end using React.js to deliver a seamless user experience across devices.

Focus on creating an intuitive and user-friendly interface that simplifies the operation of Travel Plan List browsing and creation, Polling creation, instant chatting and making notes for shared expenses.

6.3. Technologies

Trello: Use task boards to manage this project.

GitHub: Code management.

Visual Paradigm: Design and draw project diagrams.

IntelliJ IDEA community edition: Development IDE.

Chrome: Used for front-end development and testing.

Java 8+: For developing the core logic and handling business processes.

Spring Boot v3.3.4: A robust framework for developing Java-based application with integrated support for building secure, scalable RESTful APIs.

React.js v18: For building a dynamic, responsive web interface that offers an interactive user experience.

Tailwind CSS v3.4: For styling and creating modern, visually appealing components.

Npm: Runtime JavaScript package manager.

WebSockets: For real-time, bidirectional communication between the server and clients.

OAuth 2.0 and JWT: For secure user authentication and authorization.

MySQL v9.0: A perfect solution for relational data storage, especially when the data structure has been confirmed.

7. System/Software Architecture Diagram

An initial draft design of technical architecture is shown in Diagram 7.1.

7.1. Frontend Client

The web application will be developed using React.js to provide a highly interactive, responsive, and user-friendly experience. The design focuses on intuitive navigation, dynamic data rendering, and seamless interaction to meet the needs of users who are using this application as an efficient travel assistant tool.

Presentation layer: This is the layer where all the UI components are. This layer is responsible for rendering data and for interacting with users. Forms, buttons, pictures and textual content are on this layer.

Business logic layer: This is the layer that focuses on the core logic of the application. It contains the rules, algorithms, and processes that drive the application's functionality. To enhance reusability, this layer is typically implemented as a set of custom Hooks.

Data access layer: This layer is responsible for retrieving data from the API.

7.2. Backend Service

In the back-end server side, the project uses Monolithic MVC (Model-View-Controller) architecture. There are mainly three layers of architecture: web API controller layer, business logic service layer and data persistence layer. Besides, Entities are used to define data model.

7.2.1. Web API controller layer

The controller layer routes requests to corresponding API. This layer accepts client API calls, receives query and path parameters. Then it validates the parameters and provides corresponding operation based on the validation result: whether to respond error information or continue to process request data with service layer functions. For example: when the users create a new account in the application, the information of user profile, such as first name, last name and phone number, is submitted and passed as parameters or request body to the API, those parameters will be validated and passed to user creation function inside service layer.

7.2.2. Service layer

The service layer handles core business logic of each functionality. It receives request parameters passed from controller layer, processes the data with core algorithm, then calls repository layer to do CRUD (Create, Read, Update and Delete) operation with processed data. By using Spring Boot framework, manual dependency injection is not needed. Which mean developers can use powerful and effective annotations like @RestController, @Service, @Autowired to inject service instances into controller layer. Besides, @Repository, @Entity and @Override etc. are some commonly used annotations to help develop the application more efficiently.

7.2.3. Repository layer

The repository layer is responsible for CRUD operation of database. It acts as an intermediary layer between business logic and data storage. It abstracts the details of data access and centralizes common queries or operations. For example, abstract method:

```
public TravelPlan findTravelPlanById(final String id)
```

can be defined in repository interface to find required Travel Plan record using its id.

7.3. Database

MySQL will be used in this project. It's a widely used RDBMS (Relational Database Management System), free and open source. MySQL is optimized for speed and efficiency, making it suitable for handling large databases and supporting high-performance applications. Meanwhile, it has extensive online documentation, forums and community support, so it's easier for developers to find solutions for common issues and learn about best practices.

The reason to prefer RDBMS rather than NoSQL here is because the data structure is clear and the relationship between objects is maintained very well. For example, there are three main tables to represent one of the core concepts of this project:

USER, TRAVEL_PLAN, USER_PLAN_STATUS

There are two FKs in USER_PLAN_STAUS table which are USER_ID and PLAN_ID, and there's also a column named status in this table, each record in this table simply means the relationship between one User and one Travel Plan, such as Owned, Applied, Accepted or Refused.

Therefore, RDBMS is an appropriate choice of database for this application.

7.4. Third party service integration

G Suite is a suite of collaborative productivity apps that offers the user professional email, shared calendars, online document editing and storage, video meetings and more features. Email notification is a convenient and effective way to send system messages to users. Therefore, in this application, G Suite is used an easy approach to send notification.

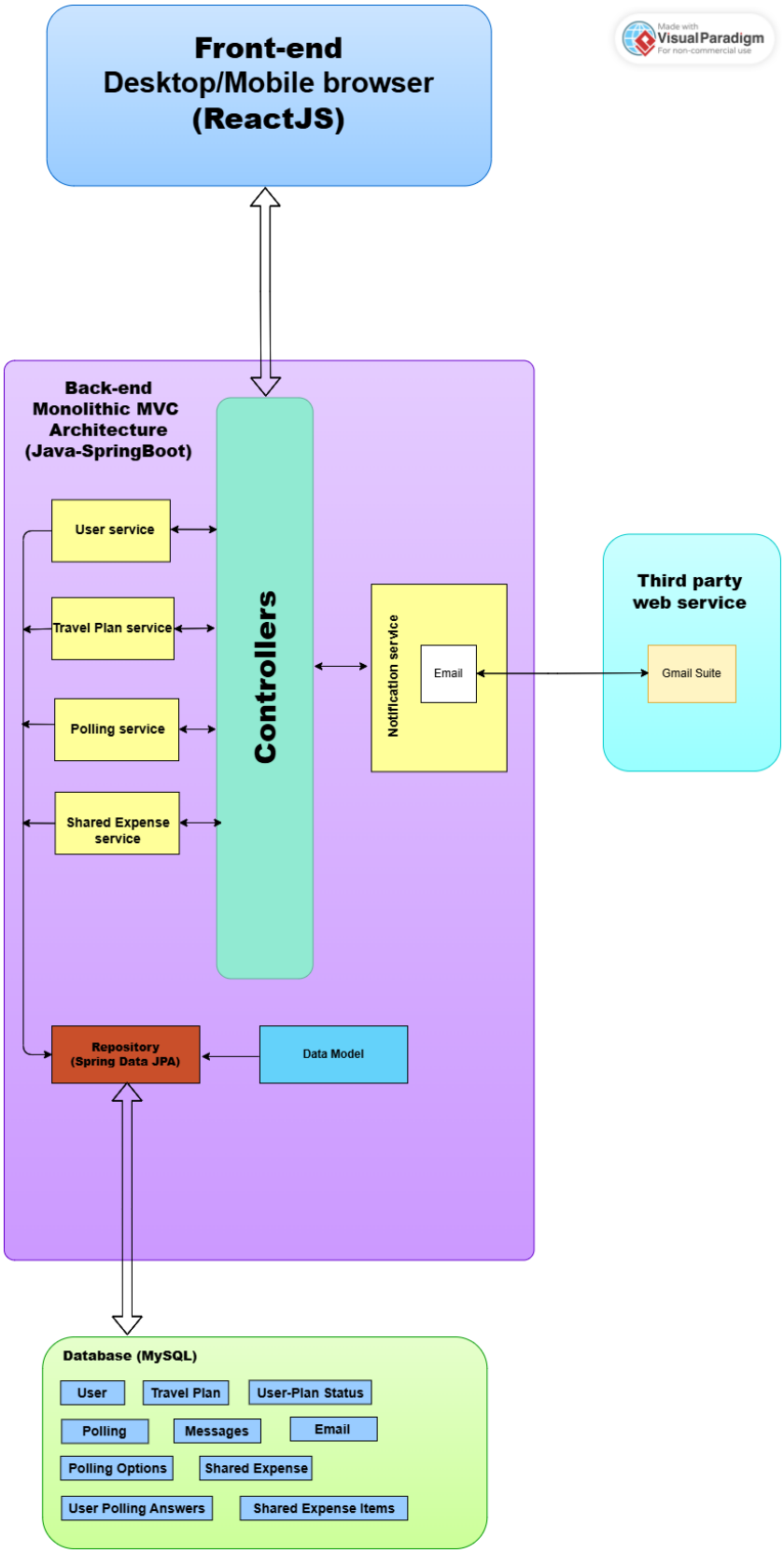

Front-end
Desktop/Mobile browser
(ReactJS)
Made with Visual Paradigm
For non-commercial use
Back-end
Monolithic MVC
Architecture
(Java-SpringBoot)
User service
Travel Plan service
Polling service
Shared Expense service
Controllers
Notification service
Email
Third party web service
Gmail Suite
Repository (Spring Data JPA)
Data Model
Database (MySQL)
User
Travel Plan
User-Plan Status
Polling
Messages
Email
Polling Options
Shared Expense
User Polling Answers
Shared Expense Items

Diagram 7.1

8. Innovation

Incorporating innovative, experimental, or exploratory features and concepts is crucial for advancing the user experience and technical capabilities of the Group Travel Assistant web application. By leveraging modern technologies, such as Spring Boot and React.js framework, Java and JavaScript programming language and MySQL database, this project not only addresses current challenges in group travel organizations but also integrates efficient real time communication and collaboration tools to bring smooth, convenient and innovative user experience in group activities. Technically speaking, the tech stack applied in the development is just widely used modern technology, but some design and concept of the features and the integration of them are absolutely new. Below are the key innovative components that will be implemented in the project.

8.1. Personalized Itinerary Planning

In Group Travel Assistant app, the first core feature is to allow user to create personalized travel plan and publish their plan to allow other users with same interest to join. Many criteria of the trip can be added into the plan, such as the origin and destination, the routine and transport of each day, the estimated budget of the entire activity and some other preferences. One innovative feature here is that not only travel, but also other activities like attending a family event, such as wedding, watching a soccer game or concert can be used as topic of the plan. Therefore, it can be widely used for daily activity planning.

Travel applications like Intrepid always provide fixed travel itinerary. Therefore, users cannot customize their trip using these apps. Sometimes, travelers only want part of the existing itinerary or change some routine based on their interests, so those traditional travel apps cannot meet this requirement. However, Group Travel Assistant can make this possible. Users will create their own travel plan, then the plan will attract other people with similar interests to join it.

8.2. Filter Algorithm

One of the other key challenges in group travel planning is finding specific group of people with same travel preferences. With the filter algorithm, user profile can be used as filter parameters, such as gender, age range, language, estimated budget and time range of the plan. Only those users whose profile matches the filter criteria can see the published Travel Plan. Therefore, people can clearly make their decision whether to join a group activity to make new friends and have fun. Besides, without using the filter, private Travel Plan can also be created and only be used within close friends or family members.

In the traditional application like G Adventures, people can only apply for existing itinerary and they could not know who will travel together with them, like the gender, age and other backgrounds, but by using Group Travel Assistant, the plan owner can set certain criteria so that they can make sure the attendants of the travel group are people that they would like to travel with.

8.3. Real-time Collaboration and Polling System

One of the unique challenges of group travel is the need for real-time collaboration, communication and consensus among group members. The Chat Room feature can enable group members communicate and collaborate in real-time. Besides, polling on activities can be enabled so that group members can propose and vote on activities, accommodations, and itineraries. By using this

application, a separate IM app is no longer needed. These integrated features could provide convenient user experience during the group activity.

8.4. Share expense notes

In this app, a convenient feature for making note of shared expenses is introduced. Very often, shared expenses happen during the trip or other activity, such as the group members sharing a meal, buying a birthday cake for another lucky person or sharing a taxi. By using this unique feature, users can make notes of those shared items and check corresponding members, then they can split the expense. All members can reference these records, so it's easy and clear for all members to know the amount of the extra shared expenses.

9. Complexity

The development of a Group Travel Assistant web application with innovative and exploratory features poses challenges that make the project non-trivial and well-suited for a BTech-level project, rather than a diploma-level one. The complexity arises from both the advanced concepts and requirement needed to implement features and the technical requirement that go beyond basic application development. In order to be involved in the entire SDLC, the developer must have enough knowledge and experience covering design, development, testing and deployment of the application. A diploma student may have knowledge of a certain type of development technology, but handling the entire SDLC requires much more than a diploma student can learn from their courses. A BTech-level student has attended courses including front-end and back-end development technologies, databases, network and other technologies required for this project, so this project should be done as a BTech-level project. The knowledge required to complete this project is more like from a full stack developer with a couple of years' working experience.

9.1. Frontend UI Design

Frontend UI is often the first interaction a user has with the application. Good UI enhances usability. Responsiveness and adaptivity allow the flexible layouts to work seamlessly on multiple screen sizes and devices. The developer does not have plenty of experience in UI design. Therefore, proficiently using Tailwind CSS to design and develop UI requires a learning curve for the developer.

9.2. Backend Architecture and System Integration

The backend service will be developed in MVC architecture which involves multiple layers to build the entire application. Comprehensive configuration is needed to make the application work smoothly and efficiently. Server, data source, JPA, logging and G Suite configuration should all be added and configured correctly, so that the backend service could provide accurate and high performance. Besides, a third-party service, G Suite, would be integrated into the backend service. The appropriate usage of the external API arises communicating complexity. Debugging the API and adding logic to handle exceptions, errors, or timeout scenarios would require comprehensive understanding of modern API development.

9.3. Security and Authentication

Security is crucial for a web application. Proper authentication is absolutely needed to control access to the web APIs in this application. Spring Security framework is chosen to be used in development

of user authentication of this Group Travel Assistant application. Only registered users with valid authorization can access resources in the application. Otherwise, 401 Unauthorized errors should be returned to the sender. Certain approach increases the complexity of development.

9.4. Database design

One of the key factors of optimizing application performance is proper database design. For example, covering each scenario when the users are using this application is not easy, because the combination of multiple dimensions of status represents the different status of the Travel Plan, and the relationship between user and it. Therefore, how to properly define the columns of each table, and assign PKs, FKs or indexes to them also arises complexity in database design.

9.5. User Experience and Feedback

Although the developer will use waterfall methodology for the development, some feedback from potential users during development will be very useful for adjusting the functions to provide better user experience. Certain approaches will increase the complexity during development, since how to effectively arrange tests and not to impact deadlines requires some effort. For example, applying WebSockets into the Spring Boot application is new to the developer. Therefore, after implementing the Chat Room feature, the developer will ask for help from some friends to simulate the real scenario on using this feature and get immediate feedback from them.

In conclusion, to implement this project, the developer needs comprehensive knowledge and experience regarding software development which is required for a BTech-level student. Some of the challenges, such as designing well organized UI and integrating WebSockets into Spring Boot project, require some research, learning and experiment by the developer.

10. Technical Challenges

The Group Travel Assistant project is technically challenging due to the complexity of building from the frontend client, backend service to database. There are some key areas where the project demands knowledge and skills that are not covered fully by typical classroom or lab content.

The first challenge is building an SPA (single page application) with React.js. There is still a learning curve for the developer to proficiently create reusable components and hooks in the frontend project. Besides, using Tailwind CSS to design well-organized layouts is also a big challenge. The developer first needs to get a comprehensive understanding of the syntax and commonly used classes, then applies them in building the UI. Keeping the layout simple and intuitive, while using visual hierarchy to guide the user's attention and usage requires some design strategies. In this application, functions are split into two main sections based on different phases of the Travel Plan, and there is multiple sub functions listed inside each main section. Where to put the search bar and how to redirect the page to the result list and go back or continue needs good thinking and reference from other successful applications.

Secondly, triggering scheduled CRON job to automatically start and complete a Travel Plan involves consideration of time zone. Trigger time should be using the destination time zone, but users may live in different time zones before the Travel Plan gets started. How to display the schedule clearly and not make users confused is also a design challenge. Whether displaying local time or destination time should

be consistent within entire application to avoid confusion, so that users could know when to expect the start and end of the Travel Plan.

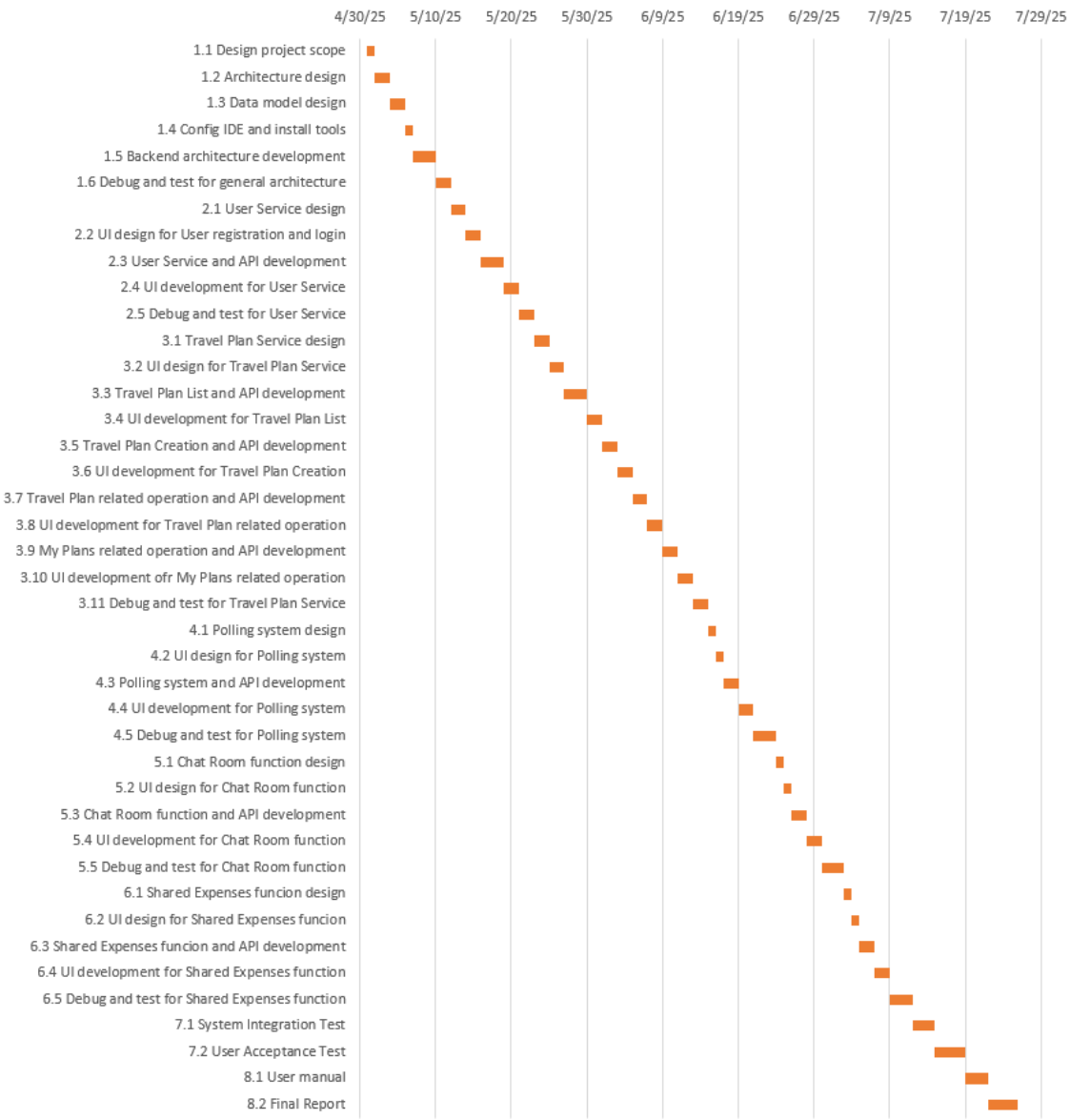
Thirdly, integration of real-time communication is another challenge for the developer. Implementing real-time communication introduces concurrency and synchronization issues. WebSockets for live updates and interactions is a good solution for this scenario. However, how to provide smooth user experience in the Chat Room feature for all involved users is not easy. Especially with the increasing number of group members, how to handle the performance over this single TCP connection requires some optimization.

Lastly, date security and privacy are also important aspects of the project. The challenge is to ensure that the application complies with data laws and employes strong encryption and security measures for user data, including account password and chatting messages etc.

In conclusion, the project presents technical challenges across several domains, from frontend UI design, real-time communication technology integration, time zone application to database design. Students will be required to research, experiment, and apply modern technologies to complete the project.

11. Development Schedule and Milestones

Development	**405 hrs**		**5/1/25**	**7/26/25**
Phase 1	**55 hrs**	**11**	**5/1/25**	**5/12/25**
1.1 Design project scope	5 hrs	1	5/1/25	5/2/25
1.2 Architecture design	10 hrs	2	5/2/25	5/4/25
1.3 Data model design	10 hrs	2	5/4/25	5/6/25
1.4 Config IDE and install tools	5 hrs	1	5/6/25	5/7/25
1.5 Backend architecture development	15 hrs	3	5/7/25	5/10/25
1.6 Debug and test for general architecture	10 hrs	2	5/10/25	5/12/25
Phase 2	**55 hrs**	**11**	**5/12/25**	**5/23/25**
2.1 User Service design	10 hrs	2	5/12/25	5/14/25
2.2 UI design for User registration and login	10 hrs	2	5/14/25	5/16/25
2.3 User Service and API development	15 hrs	3	5/16/25	5/19/25
2.4 UI development for User Service	10 hrs	2	5/19/25	5/21/25
2.5 Debug and test for User Service	10 hrs	2	5/21/25	5/23/25
Phase 3	**115 hrs**	**23**	**5/23/25**	**6/15/25**
3.1 Travel Plan Service design	10 hrs	2	5/23/25	5/25/25
3.2 UI design for Travel Plan Service	10 hrs	2	5/25/25	5/27/25
3.3 Travel Plan List and API development	15 hrs	3	5/27/25	5/30/25
3.4 UI development for Travel Plan List	10 hrs	2	5/30/25	6/1/25
3.5 Travel Plan Creation and API development	10 hrs	2	6/1/25	6/3/25
3.6 UI development for Travel Plan Creation	10 hrs	2	6/3/25	6/5/25
3.7 Travel Plan related operation and API development	10 hrs	2	6/5/25	6/7/25
3.8 UI development for Travel Plan related operation	10 hrs	2	6/7/25	6/9/25
3.9 My Plans related operation and API development	10 hrs	2	6/9/25	6/11/25
3.10 UI development ofr My Plans related operation	10 hrs	2	6/11/25	6/13/25
3.11 Debug and test for Travel Plan Service	10 hrs	2	6/13/25	6/15/25
Phase 4	**35 hrs**	**9**	**6/15/25**	**6/24/25**
4.1 Polling system design	5 hrs	1	6/15/25	6/16/25
4.2 UI design for Polling system	5 hrs	1	6/16/25	6/17/25
4.3 Polling system and API development	10 hrs	2	6/17/25	6/19/25
4.4 UI development for Polling system	10 hrs	2	6/19/25	6/21/25
4.5 Debug and test for Polling system	5 hrs	3	6/21/25	6/24/25
Phase 5	**40 hrs**	**9**	**6/24/25**	**7/3/25**
5.1 Chat Room function design	5 hrs	1	6/24/25	6/25/25
5.2 UI design for Chat Room function	5 hrs	1	6/25/25	6/26/25
5.3 Chat Room function and API development	10 hrs	2	6/26/25	6/28/25
5.4 UI development for Chat Room function	10 hrs	2	6/28/25	6/30/25
5.5 Debug and test for Chat Room function	10 hrs	3	6/30/25	7/3/25
Phase 6	**35 hrs**	**9**	**7/3/25**	**7/12/25**
6.1 Shared Expenses funcion design	5 hrs	1	7/3/25	7/4/25
6.2 UI design for Shared Expenses funcion	5 hrs	1	7/4/25	7/5/25
6.3 Shared Expenses funcion and API development	10 hrs	2	7/5/25	7/7/25
6.4 UI development for Shared Expenses function	10 hrs	2	7/7/25	7/9/25
6.5 Debug and test for Shared Expenses function	5 hrs	3	7/9/25	7/12/25
Phase 7	**35 hrs**	**7**	**7/12/25**	**7/19/25**
7.1 System Integration Test	15 hrs	3	7/12/25	7/15/25
7.2 User Acceptance Test	20 hrs	4	7/15/25	7/19/25
Phase 8	**35 hrs**	**7**	**7/19/25**	**7/26/25**
8.1 User manual	15 hrs	3	7/19/25	7/22/25
8.2 Final Report	20 hrs	4	7/22/25	7/26/25



The entire project has been split into several phases, and in each phase, Waterfall methodology will be applied. Therefore, in each phase, schedules for design, development and test are added. The total estimated development time is 405 hours.

12. Deliverables

The deliverables of this project are as follows:

- Frontend web application (React.js)
- Backend server in monolithic MVC architecture (Java/Spring Boot)
- Sample database in MySQL
- Final Report: final report on this practicum course
- User Manual: user manual to show how to use this application

13. Conclusion and Expertise Development

In this project, I will improve my technical skills and experience in web application development. For completing the entire SDLC, I need to analyze user requirement, design the architecture, implement each feature and test the functionalities. Although there are many aspects that might require me to learn more and apply them into real project development, I believe after this project, I can gain more experience and become a better software developer.

14. References

1. Madushanka, K. (2023, August 30). Building robust React apps with 3-tier architecture and custom hooks. Medium. https://medium.com/@kavindumadushanka972/building-robust-react-apps-with-3-tier-architecture-and-custom-hooks-799e24a8d740
2. MongoDB. (2021, April 14). MongoDB is fantastic for logging. MongoDB Blog. https://www.mongodb.com/blog/post/mongodb-is-fantastic-for-logging
3. Google. (2023). About Google Groups settings and permissions. Google Workspace Admin Help. https://support.google.com/a/answer/6047848?hl=en
4. Twilio. (n.d.). Messaging. Twilio. https://www.twilio.com/en-us/messaging
5. Stripe. (n.d.). Payment processing platform. Stripe. https://stripe.com/en-ca
6. Spring. (2015, July 14). Microservices with Spring. Spring Blog. https://spring.io/blog/2015/07/14/microservices-with-spring
7. Elim, E. (2023, June 14). Building real-time apps using WebSockets. Medium. https://emily-elim04.medium.com/building-real-time-apps-using-websockets-dc137ccdd34b
8. Okta. (n.d.). 8 ways to secure your microservices architecture (Whitepaper). Okta. https://www.okta.com/resources/whitepaper/8-ways-to-secure-your-microservices-architecture/
9. Asana. (n.d.). What is project scope? Asana. https://asana.com/resources/project-scope
10. PractiTest. (n.d.). How to write a test plan. PractiTest. https://www.practitest.com/resource-center/article/write-a-test-plan/

15. Change Log

1. Job title in work experience has been revised. (Page 3)

2. Section 2 has been rewritten entirely to make the project description clearer to readers. (Page 4)

3. Section 3 has been rewritten entirely to address some comments and questions from the committee. (Page 5-6)

> (1) In the problem statement, it just vaguely says that this is for facilitating group travel. It just lists a wide range of problems that arise when trying to arrange a group trip, but it isn't clear how a single piece of software will address all of these.
> (2) What does "join the trip" mean? Strangers go to the same place? Share the same hotel?
> (3) There are lots of places such as section 3.3 where it just lists a problem, with no indication how it is solved or what it means concretely for the software.

(4) This is the problem/This is the precise software that we are going to implement to solve it.

4. Section 4 has been rewritten entirely to remove unnecessary features or unrealistic implementations for this project. Only essential functionalities exist in the application and are explained. So, some of the questions in the feedback can be addressed here. (Page 6-9)

(1) What is the payment system used with? Are there partnerships with booking agencies? Do you imagine the software just storing credit card info and then sharing? This does not seem like a good idea.
(2) Section 4.2 lists so many features, it is hard to see how they will all be implemented.
(3) There are frequently times when it is mentioned that "an algorithm" will be used... or even "sophisticated algorithms". But there is no hint at all what this means... do you have an algorithm in mind for group matching? What are the inputs/outputs? Why is it sophisticated?
(4) Perhaps it could include some mockups of what the interface might look like

5. Section 5 has been rewritten to make the test plan clearer, and that addresses some problems from the feedback below: (Page 9-11)

(1) The test plan is very vague, which is expected since I don't know exactly what the software will do.
(2) Page 9 to 13 is essentially a list of bulleted items with little explanation

6. Section 6 has been rewritten. Waterfall Methodology will be applied in the development of this application, so it will entirely replace the Agile Scrum Methodology in section 6 from the previous version of the project proposal; besides, micro-service architecture has been replaced by monolithic architecture since it's easier to develop and debug, so the developer can focus on the developing business requirement. Some of the problems mentioned in the comments can be addressed below. (Page 11-13)

(1) It is not clear what scrum methodology means for an individual project.
(2) The technology stack doesn't actually commit to definite technologies... it says "react or similar technologies". For a proposal, you don't need to be done – but you need to have a concrete plan of the technology to be used.

7. In section 7, the architecture diagram has been redesigned and regenerated. More details and explanations with examples have been added to provide clearer descriptions for each architectural component. A comment below can be addressed by these changes. (Page 13-17)

(1) Perhaps it could include some mockups of what the interface might look like

8. Most of section 8 has been rewritten to describe the innovative part of key features more clearly. (Page 17-18)

9. Some of section 9 has been modified. (Page 18-19)

10. Section 10 has been rewritten to make clear description of challenges from frontend, backend implementation to database design. (Page 19-20)

11. Section 11 and 12 have been revised. (Page 21-23)