CSCI3100 Project Report

rFriend - an Event Explorer

Group C2

Ma Ying Kei	1155127385
Lai Chun Yin	1155125168
Yu Man Ho	1155127657
Mak Wing Chit	1155125179
Leong Chon Hou	1155113489

Version 1.0

Date of printing: May 6, 2022

Department of Computer Science and Engineering, The Chinese University of Hong Kong

Table of Contents

1. Introduction

1.1 Project Overview

rFriend, named after a Cantonese slang for gathering friends together, aims to provide a more convenient and efficient way to initiate and participate in events around your social circle.

The majority of psychological research indicates loneliness may lead to dissatisfaction in life in general and raises risks of depression. The situation gets worse with the strict social distancing and quarantine policies, and as a result, people feel more isolated than ever. Solutions are needed to connect people with either known friends or new friends, in daily activities such as dining, or hobbies such as hiking.

In this project, we specifically target meet-ups with known friends, as many available hobby meet-up sites do not connect known friends, and address the common trouble of scheduling events: finding friends with the same availability. We hope this tool can help broadcast events, vacancies and availability, set event visibility, discover events around you, filter out those you are unavailable, and leave comments for discussion.

Whether you are a student who wants a new study partner, a group of hikers who wish to recruit new members, or someone who is looking to expand their social circle through meet-ups, the unique but intuitive mechanisms of this project will save you the hassle of reaching out, negotiating time, or whatever else it takes to arrange an event.

1.2 Objective

The objective of our software, rFriend, is to provide a simple but effective event organization service tailor-made for known friends. Nowadays, the communication overhead has always been a problem when organizing an event, given many meet-up applications in the market, there is no proper solution to help users to manage small or casual events between friends. Therefore, rFriend is designed to tackle this pain point in the market. In rFriend, people can connect, interact, and bring friends together with just a few clicks, they can explore more events around them in the “discovery view”, and manage their schedule in the “my event view”. Overall, rFriend serves as an easy, yet comprehensive friends meet-up platform to assist friend-based event organizations in a productive and user-friendly way.

1.3 Highlights

Our software provides a platform for users who want to find participants for their events or find some events to join. Therefore, our system could be highlighted into three main types of features: event manager, event browser, and friend system.

Event Manager

The event manager concludes the system features that handle the user’s events. Users can create their events which will be displayed on the user calendar which can help the users to plan and manage their

event schedule. Moreover, users can further edit their events anytime or they can delete the events if they want.

Event Browser

The most important feature of our software is the event browser. The event browser consists of two views which are the grid view and the map view. In the grid view, all events will be displayed on a card deck. Users browse through the card deck to find the events they are interested in. Optionally, we have a working filter in which users could filter the card deck with options such as event category, event time, and privacy setting. Furthermore, users could also direct search for an event with a search bar if they know what event they are looking for.

There is also a map view to browse the event. The map view utilizes Google Maps and map markers to display the event location and users can click on the markers to see what events will be hosted in those locations which helps the users to find events by location preference.

Friend system

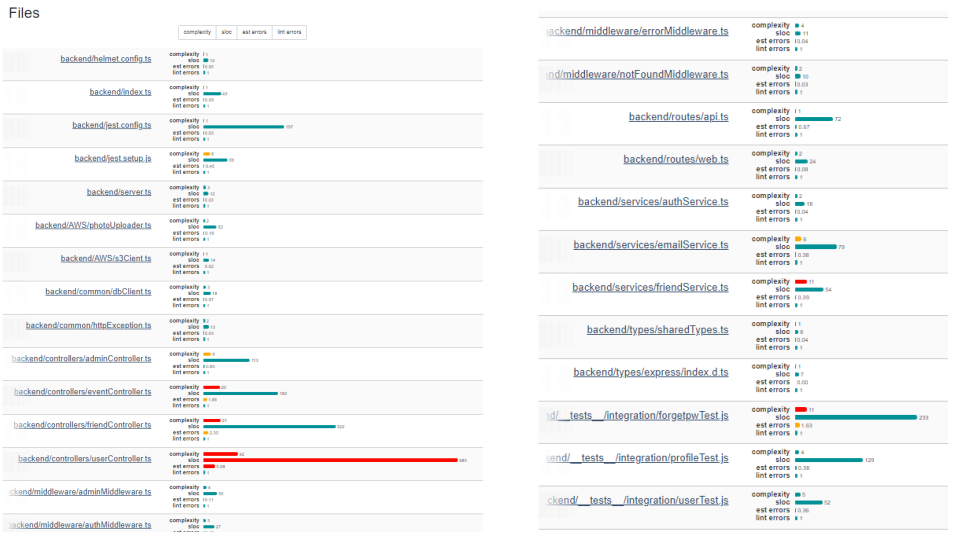
Our software consists of a fully functional friend system in which users could find and add new friends. The worth mentioning feature or importance of the friend system is that after the other users accepted your friend requests and officially became your friends, you will be able to browse their events in the “friend-only” and “friends-of-friends” privacy settings, which is essential for users to join non-public events or set up some events for their friends to join.

1.4 Project Statistics

The statistics reports are generated using “ts-plato”.

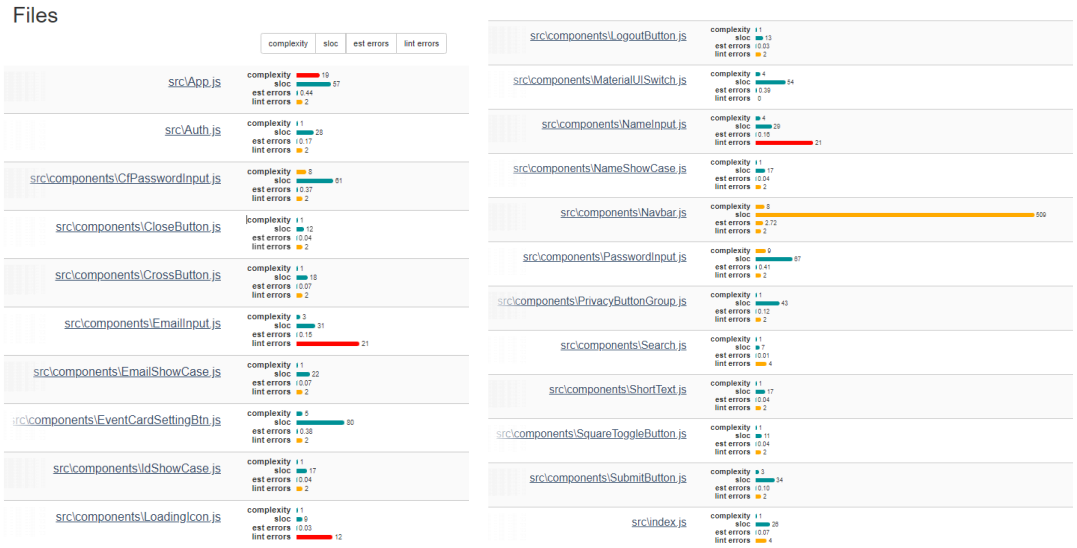
1.4.1 Backend

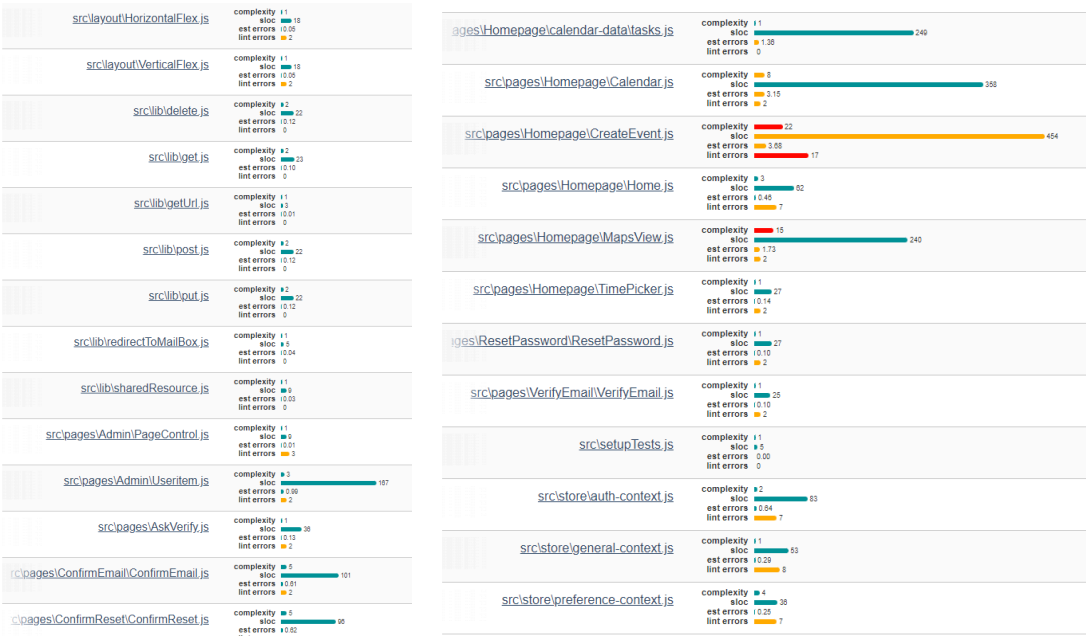
Total/Average Lines	2457/91
Average Maintainability	66.38



1.4.2 Frontend

Total/Average Lines	3332/69
Average Maintainability	76.39





2. System Architecture Design

2.1 System Architecture

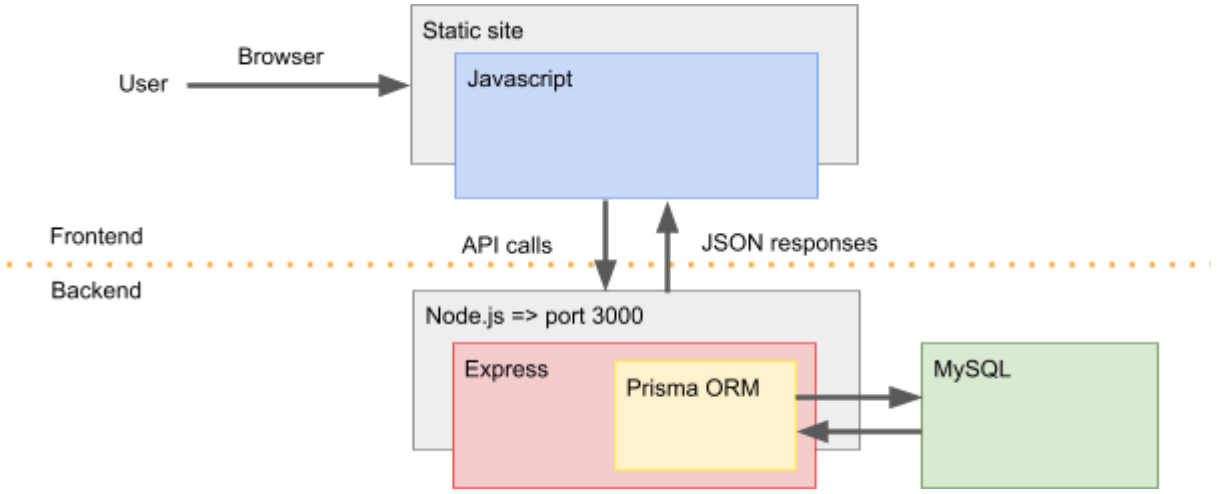
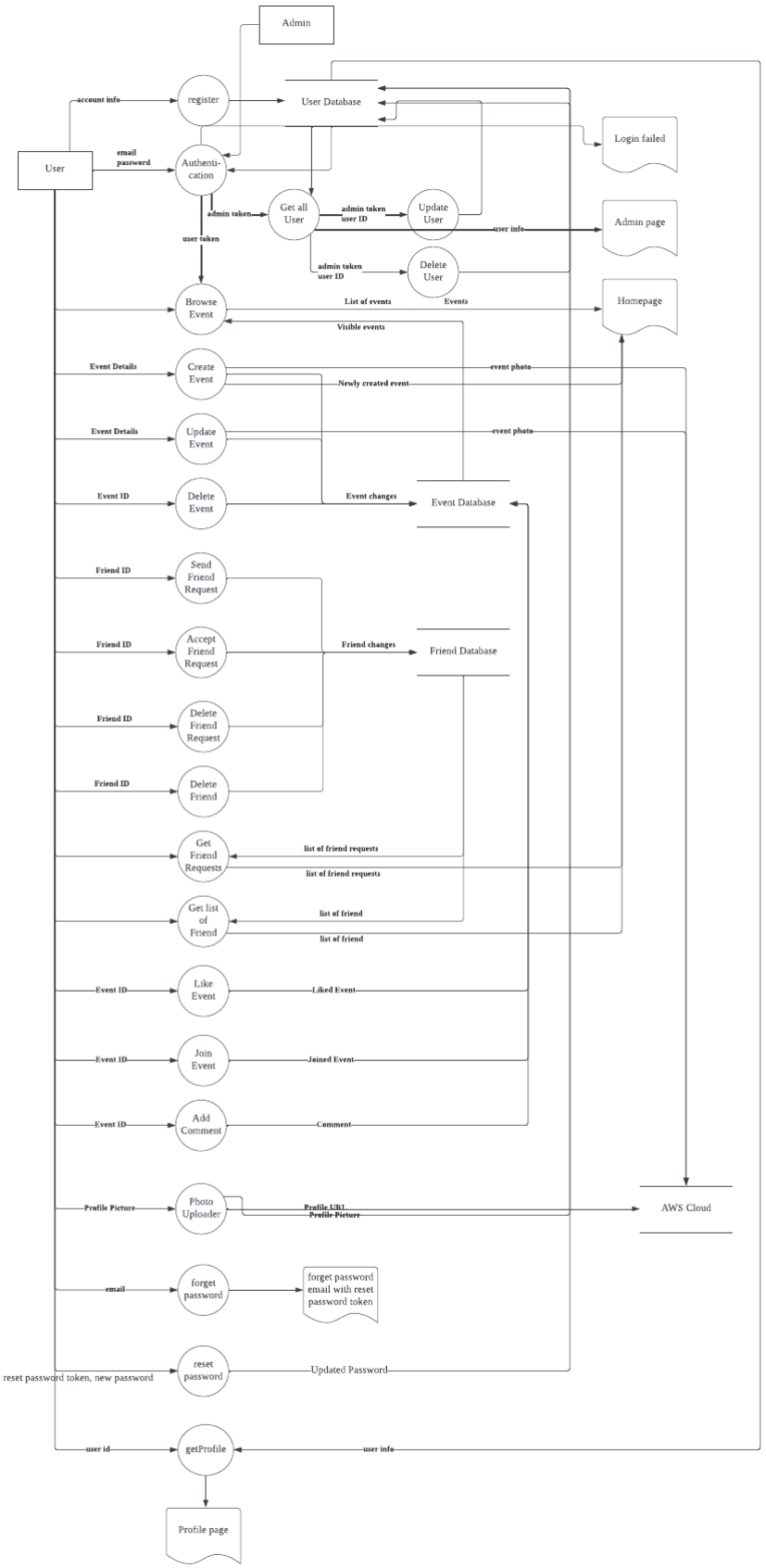


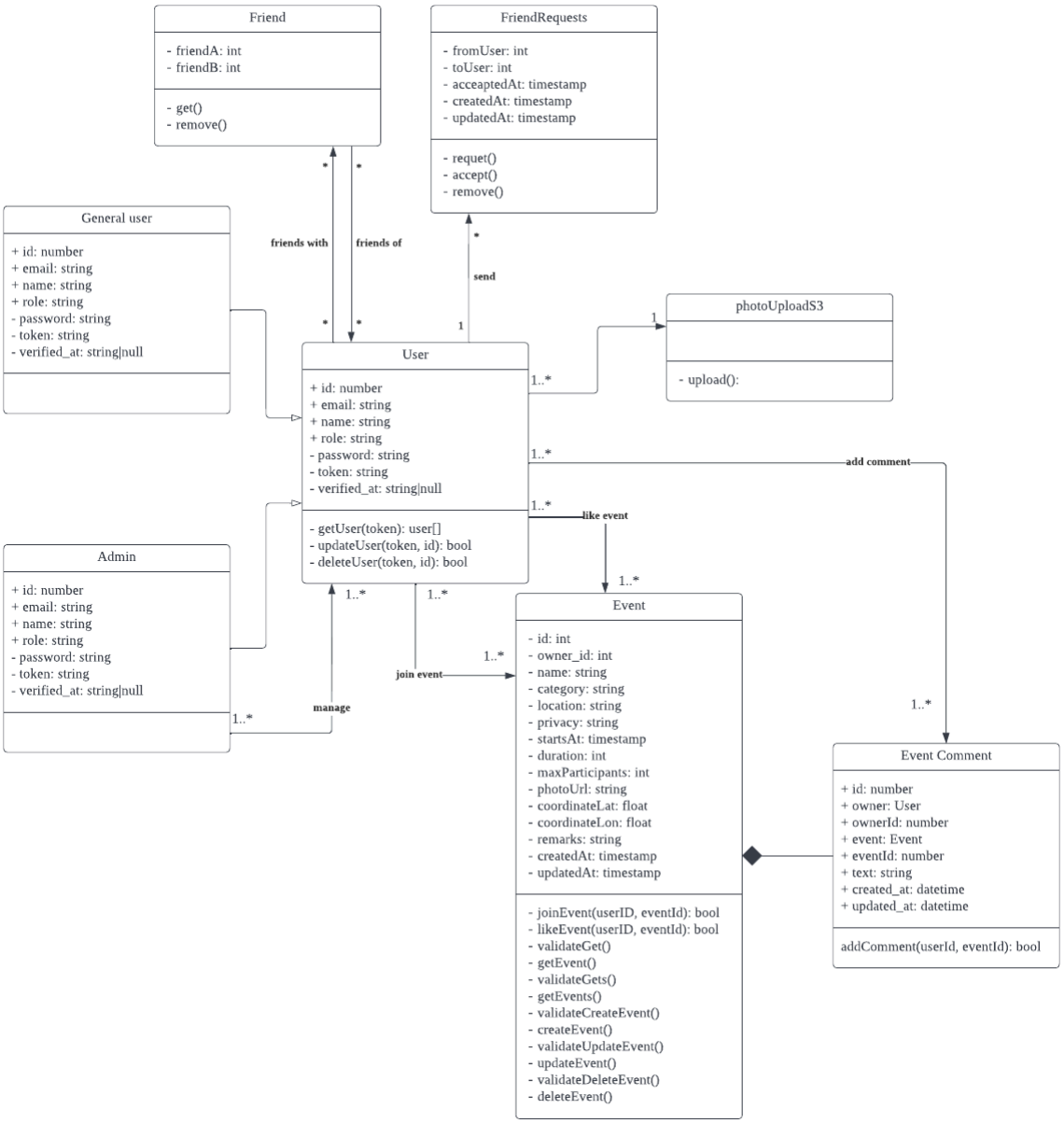
Figure X: Architecture Diagram

As shown above, the system is divided into frontend and backend. The frontend is a single-page app written with React JS, and the backend is built with Express JS with a MySQL database. They communicate via the HTTP API the backend exposes.

2.2 DFD of the Project



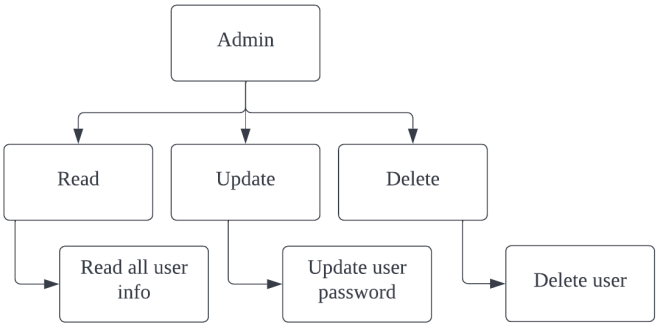
2.3 UML Class Diagram of Project



3. Detailed Description of Components

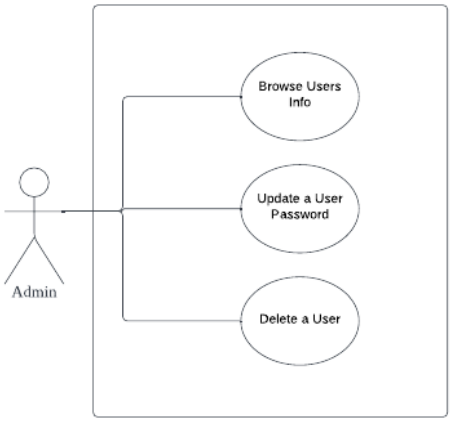
3.1 Admin

3.1.1 Structural Diagram

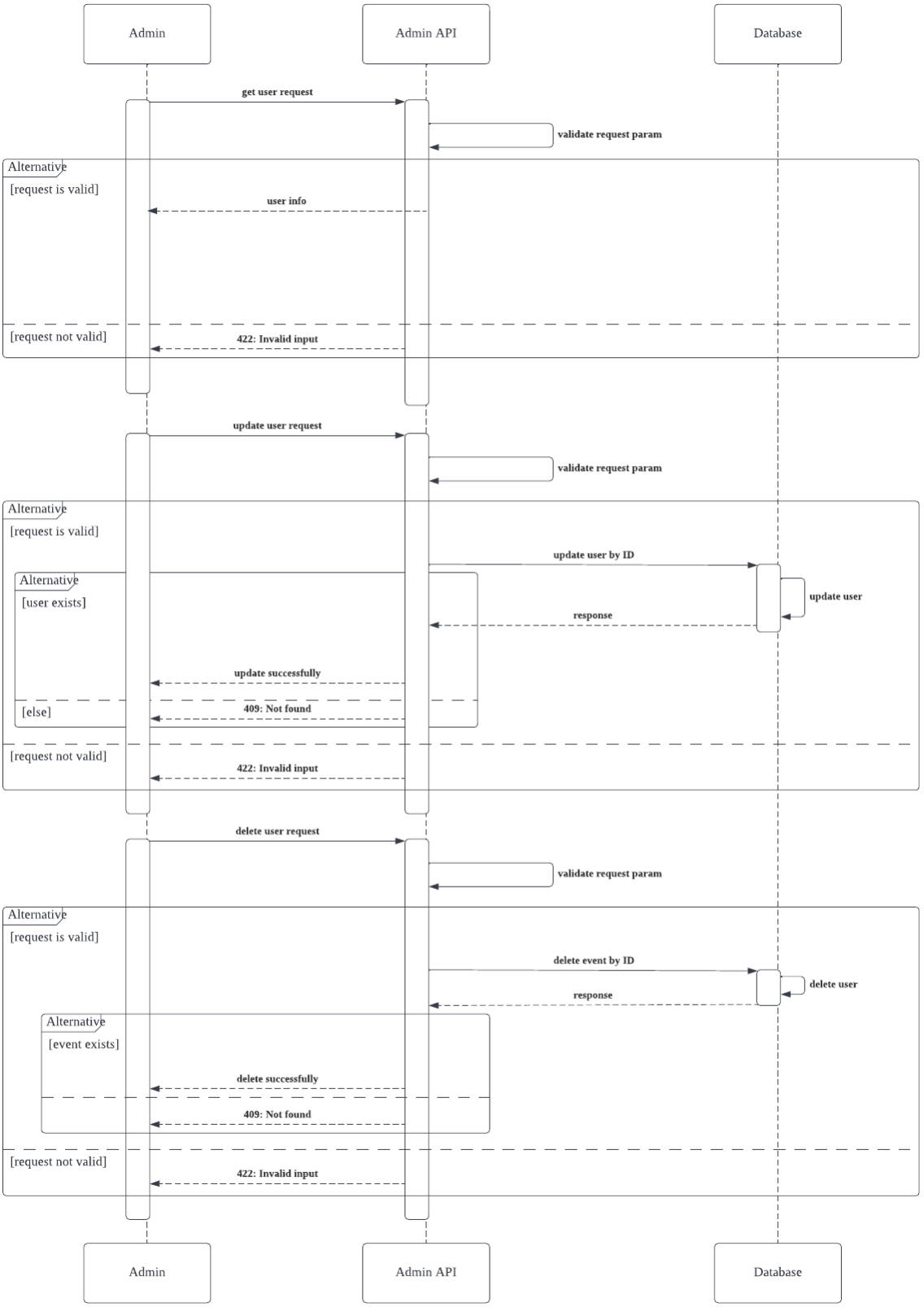


3.1.2 UMLs

Use Case Diagram:



Sequence Diagram:



3.1.3 Functionality

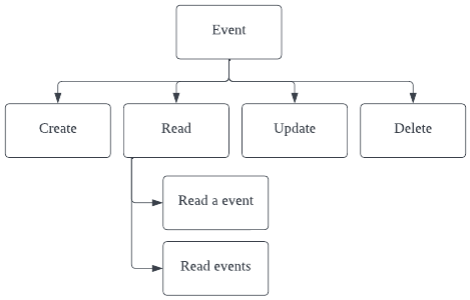
This module is responsible for handling admin-related actions, including reading, updating, and deleting users.

3.1.4 Procedures and Functions

Function/Procedure	Description
validateAdmin()	This function is a middleware function to validate the input token whether it is an admin role.
getUser()	This function return all the user informations
updateUser()	This function update a user information given the user ID
deleteUser()	This function delete the user given the user id

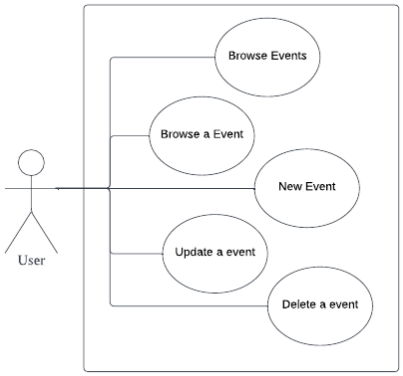
3.2 Event

3.2.1 Structural Diagram

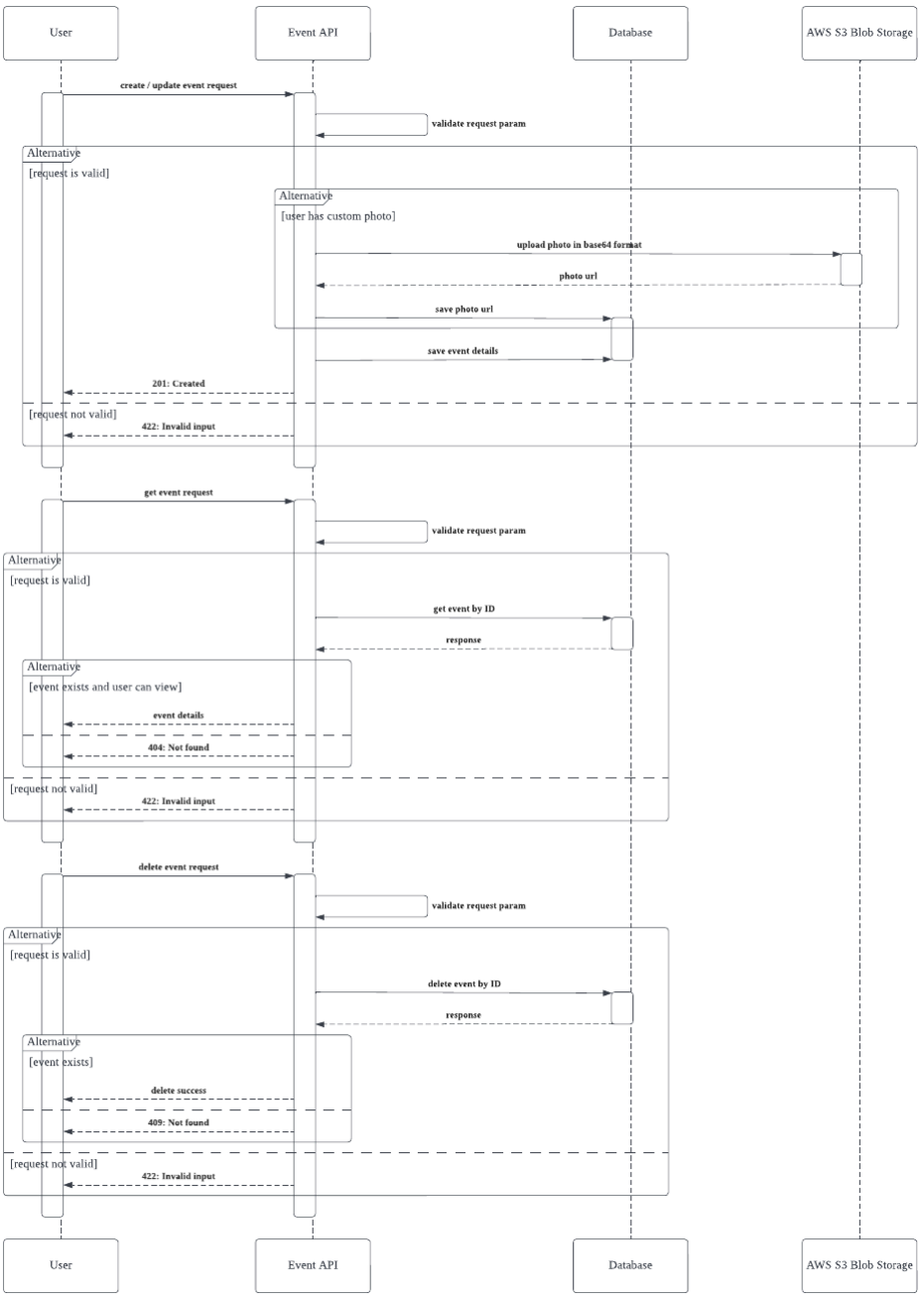


3.2.2 UMLs

Use Case Diagram:



Sequence Diagram:



3.2.3 Functionality

This module is responsible for handling event-related actions, including creating, reading, updating, and deleting.

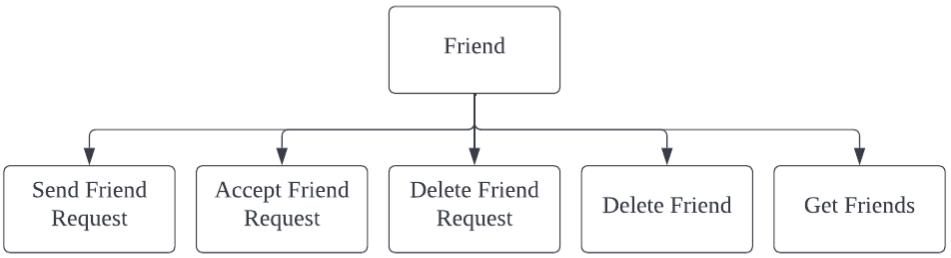
3.2.4 Procedures and Functions

Function/Procedure	Description

validateUpsert()	This function is a middleware function that returns an array of input validation functions for the upsert function. The validation result, which is appended to the request object, can be handled by the upsert function.
upsert()	This function serves both the create and update functionality. The function will insert a new event to the database if the event id is not given, otherwise, it would update the specified record. Depending on the user input, if an event photo is supplied by the user, it will be uploaded to AWS S3.
validateGet()	This function is a middleware function that returns an array of input validation functions for the get function. The validation result, which is appended to the request object, can be handled by the get function.
get()	This function returns the event or events requested.
validateRemove()	This function is a middleware function that returns an array of input validation functions for the remove function. The validation result, which is appended to the request object, can be handled by the remove function.
remove()	This function deletes the event requested.

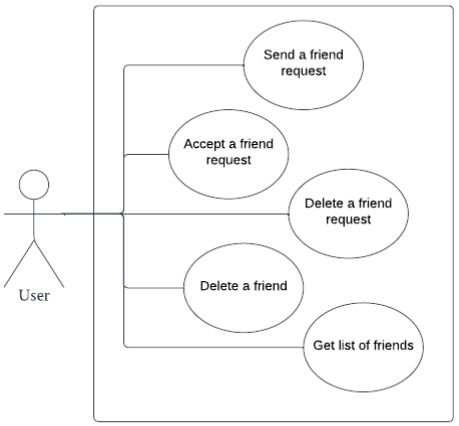
3.3 Friend

3.3.1: Structural Diagram

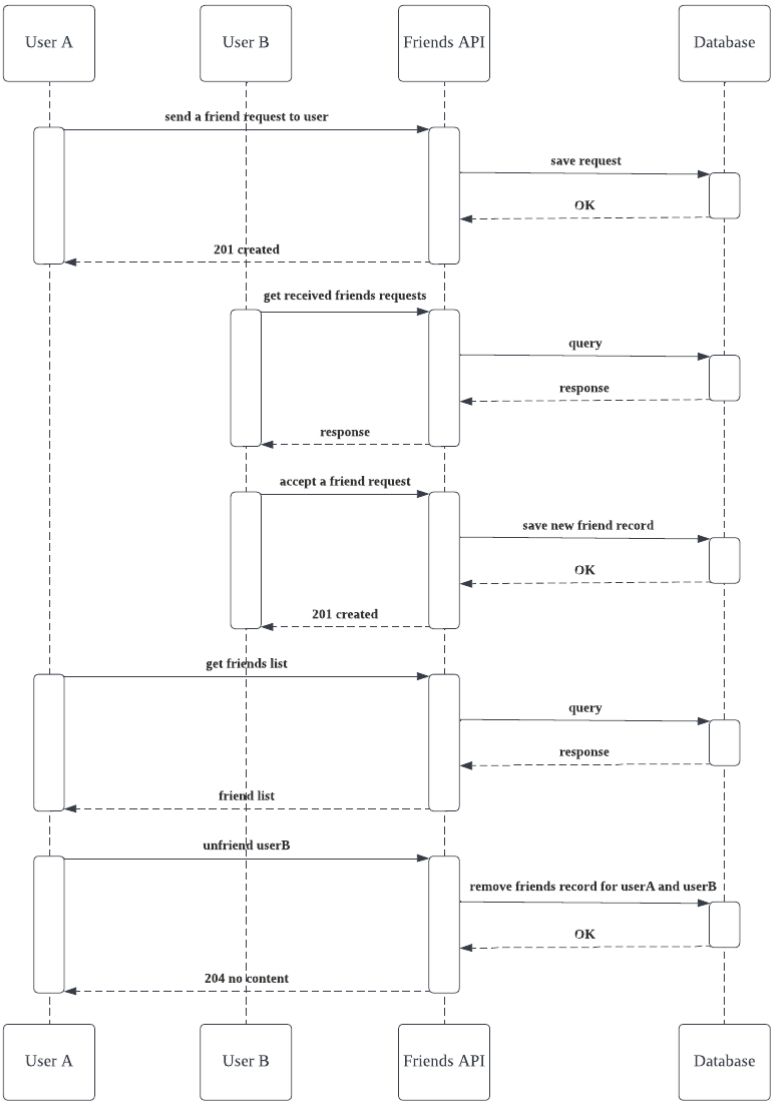


3.3.2: UMLs

Use Case Diagram:



Sequence Diagram:



3.3.3 Functionality

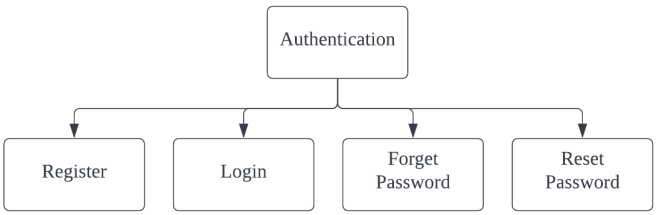
This module is designed to facilitate any befriending-related actions, including sending friend requests, viewing friend requests, accepting friend requests, querying friends, and deleting friends.

3.3.4 Procedures and Functions

Function/Procedure	Description
validate()	This function is a shared middleware function that returns an array of input validation functions for the remove function. The validation result, which is appended to the request object, can be handled by the following functions: newFriendRequest, acceptFriendRequest, deleteFriendRequest, deleteFriend.
newFriendRequest()	This function accepts a user ID as input and creates a new friend request record on behalf of the requesting user.
acceptFriendRequest()	This function accepts a friend request and creates a new friend record between the two users.
deleteFriendRequest()	This function deletes a friend request.
getFriends()	This function returns a list of friends that the user is connected to.
deleteFriend()	This function removes the friend record between the two users.

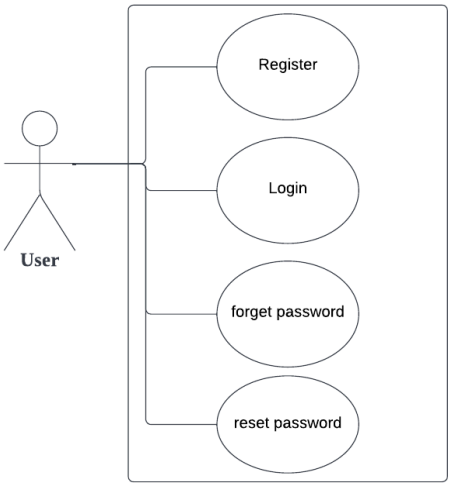
3.4 Authentication

3.4.1 Structural Diagram



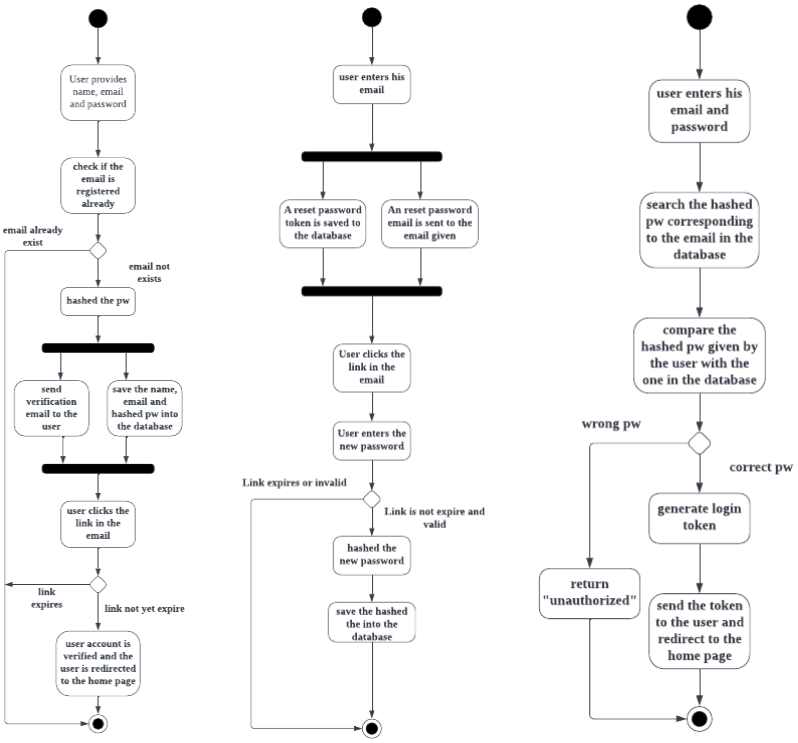
3.4.2 UMLs

Use Case Diagram:



Sequence Diagram:

Activity Diagrams



3.4.3 Functionality

This module handles functionalities related to user authentication, including registering a new account, logging in to an account, and changing the password.

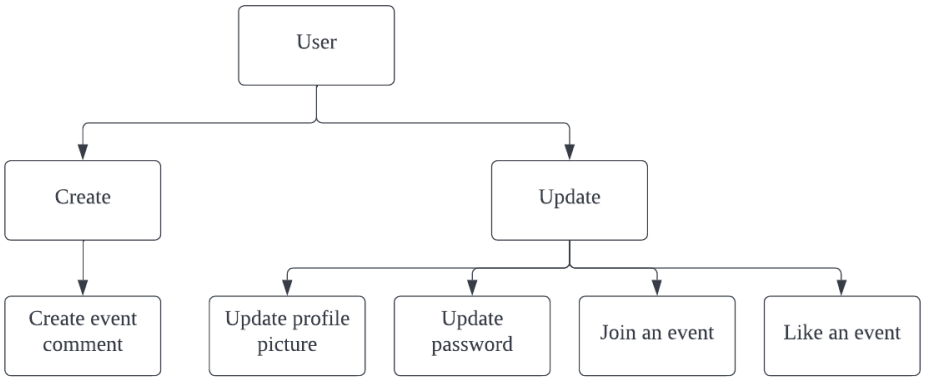
3.4.4 Procedures and Functions

Function/Procedure	Description
getUser	This function first checks if the email provided by the user is already registered. If not, the user inputs, including the user name, email and the hashed password will be saved to the database. At last, a verification email will be sent to the user's email address. The user has to click the link in the email to complete the registration process.
updateUser	The user provides his email and password. The server then checks if the given email and password pair is valid. If it is valid, the user will receive a login token that expires in a day and will be redirected to the main page. If it is invalid, an error message will be shown.
deleteUser	The user provides his email. Then, the server checks if the email exists in the database. If it exists, an email containing a link to the password reset page will be sent to the given address. The link is associated with a special token that is different each time when the link is generated. The link will expire in an hour and if the user calls the function multiple

	times, only the latest link received will be valid.
pw_reset	Once the user enters the new password on the password reset page. The new password and the special token associated with the password reset page link will be sent to the server. The server will first check if the token is valid, meaning that it is not yet expired and matches the one stored in the database. If the token is valid, the new password will be hashed and stored in the database.

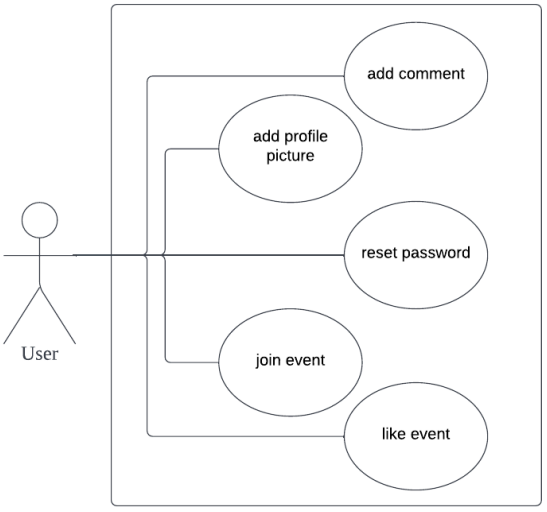
3.5 User

3.5.1 Structural Diagram

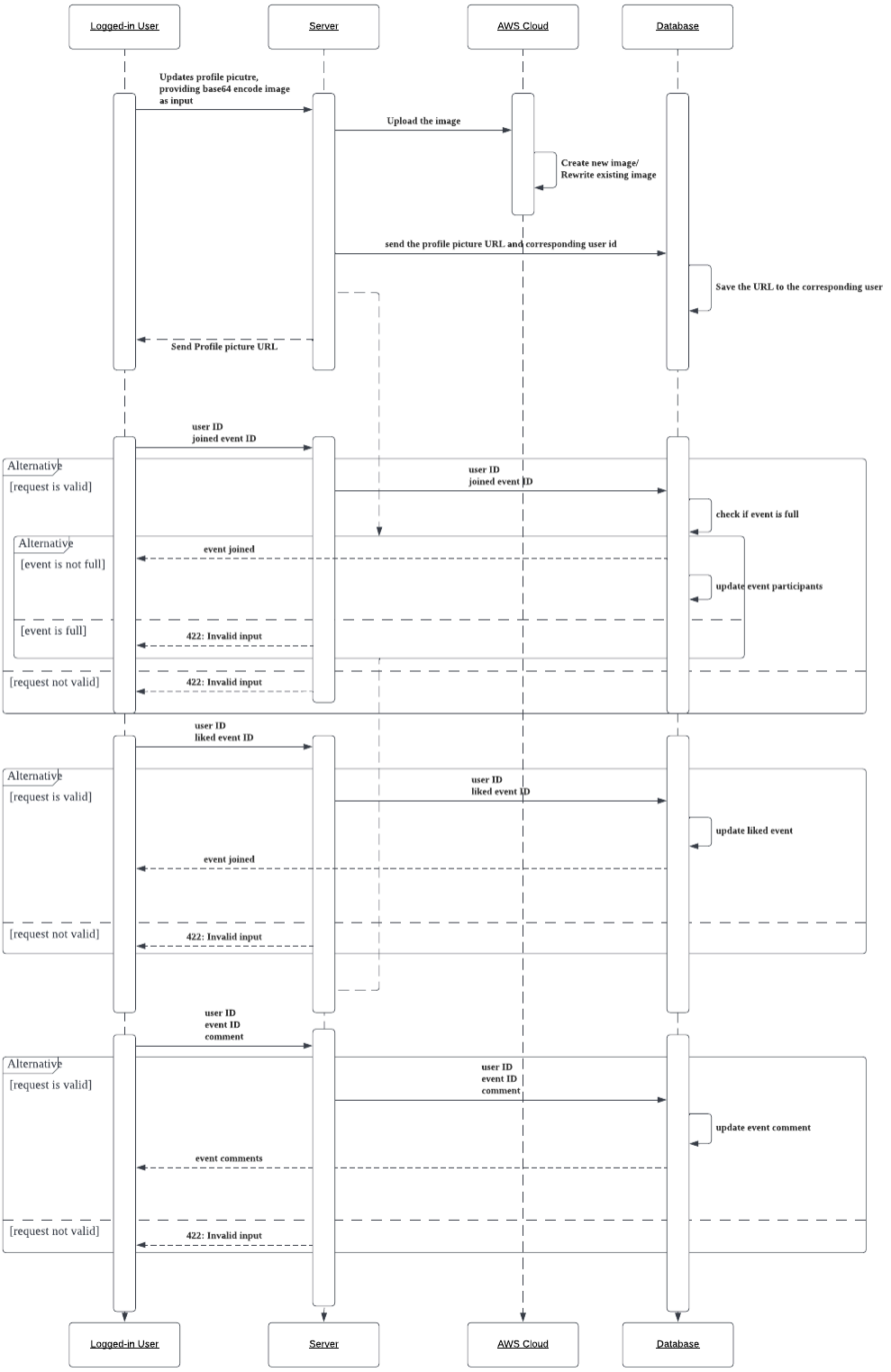


3.5.2 UMLs

Use Case Diagram:



Sequence Diagram:



3.5.3 Functionality

This module handles functionalities related to user interaction, including updating the user profile picture, getting user information of other users, joining an event, liking an event, and adding comments to an event.

3.5.4 Procedures and Functions

Function/Procedure	Description
getProfile	This function returns information of the user corresponding to the given user id, including his name, email, joined events, and the URL for his profile picture.
updateProfile	This function receives a base64 encoded image as input. The image will be stored in the AWS cloud. Each image can be accessed through a unique URL, which will also be stored in the database. The existing image will be rewritten.
joinEvent	This function takes the user token and event id as input, the function will first check if the event is full, then the user id will be added to the participant list if the list is not full, and return the joined event if the user joined the event successfully.
likeEvent	This function takes the user token and event id as input, the user ID will be added to the liked event list, and return the liked event.
addComment	This function takes the user token, event id, and comment as input, the user id, name, and comment will be stored in the EventComment database, and return all the comments of the event.

4. User Interface Design

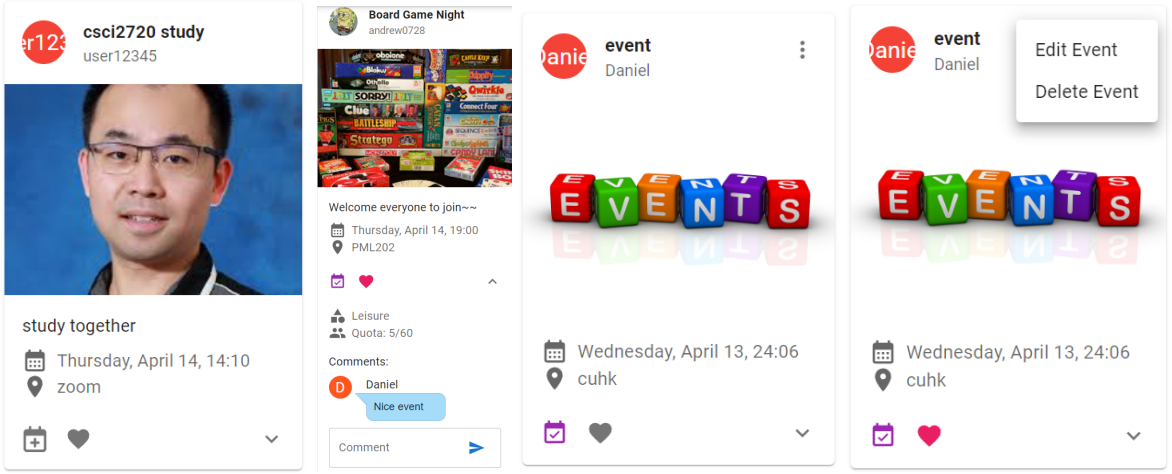
4.1 Description of the User Interface

Other than relying on vanilla HTML, CSS, and JavaScript, the user interface is mainly developed by MUI, which is a UI framework that implements material design standards. Therefore, by utilizing MUI components as building blocks, it becomes possible for frontend developers to achieve a modern and interoperable UI without the aid of professional web designers.

The UI is responsive, which means it works well for desktop screens, tablet screens, and mobile screens, which are achieved by the flexbox and grid system.

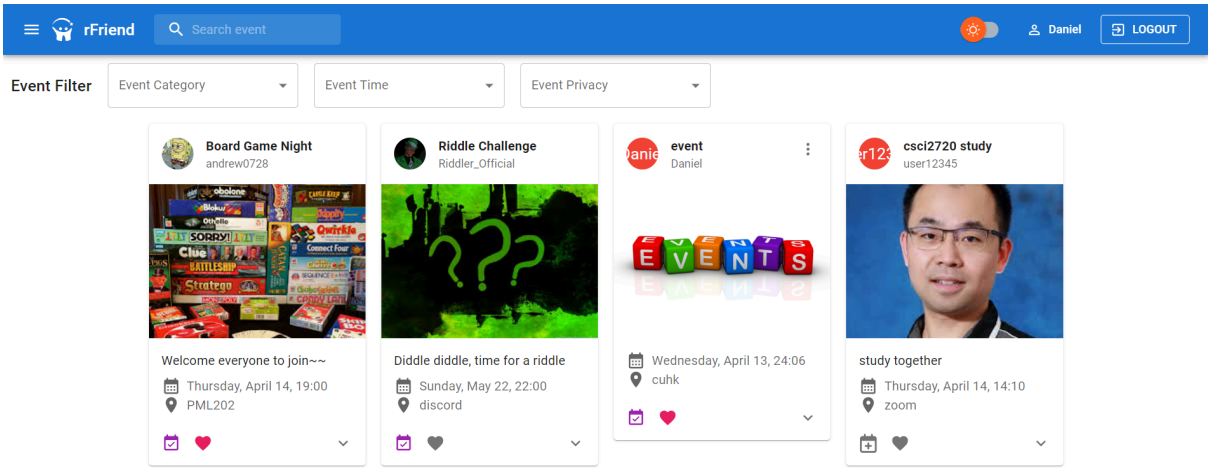
4.2 Screen Images

Event Card



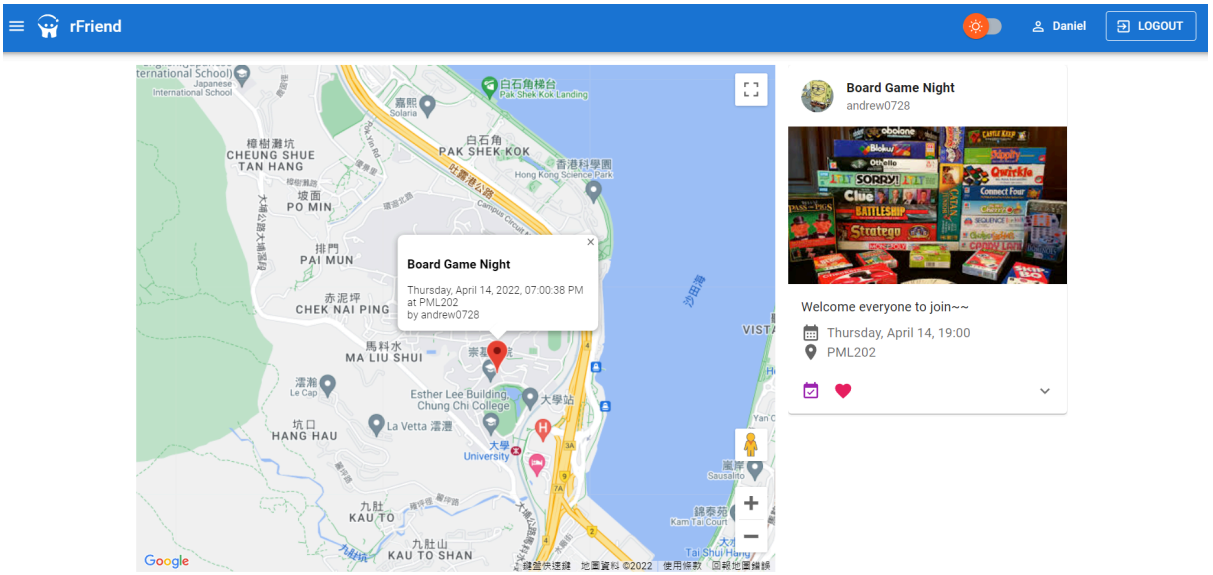
The events created by the users will be displayed through event cards. The default event card will be the unexpanded one. It will display the event name, the username of the host, the event picture, event description, location, and time. The event picture can be uploaded by the user by their preference, if the user did not upload one, it will use the default picture based on the event category. Users can join the event by clicking the add event button or like the event by clicking the heart button to save it. The users can also expand the event card by clicking the arrow button to view more information about the event such as the event category, the maximum number of participants, the current number of participants, and the comment section. The user can view other users' comments and post their own comments in the comment section. If the user is the host of the event, the event card will have a config button on the top-right hand corner and it contains the options to edit or delete the event.

Event Browser (Grid View)



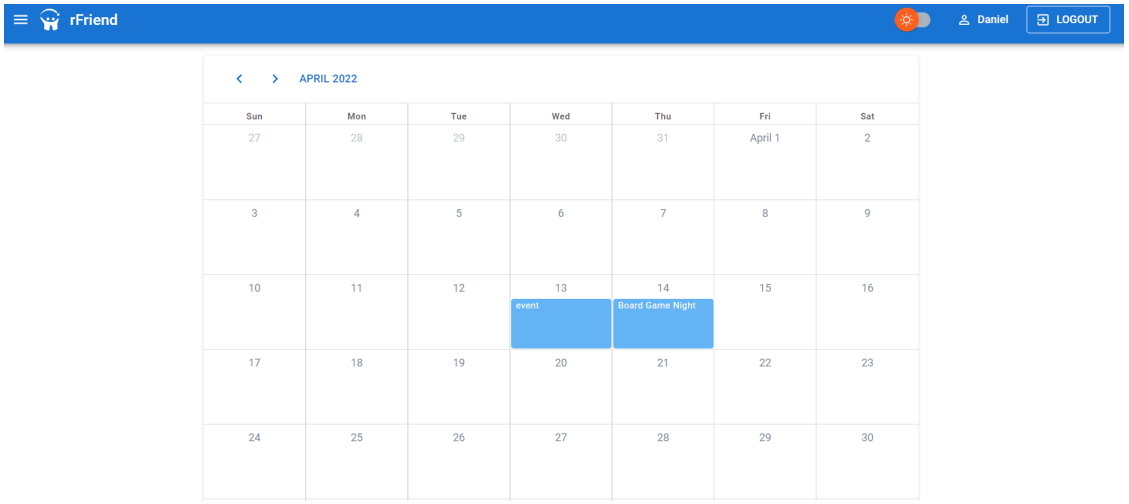
This is the event browser in grid view, all the event cards will be shown with a grid layout. On this page, users can browse all events with public, friends-only, and friends-of-friends privacy levels plus an addition of the events created by the user him/herself no matter what the privacy level is. Then, the user can filter the events with 3 options: event category, event time, and privacy level. For example, when setting the filter to "Dining", "Night", and "Friends", the browser will only show the dining events at night that is only shown to friends. Furthermore, the user can also directly find an event by searching the event name.

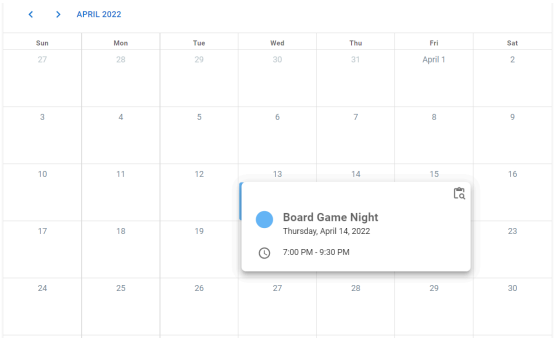
Event Browser (Map View)

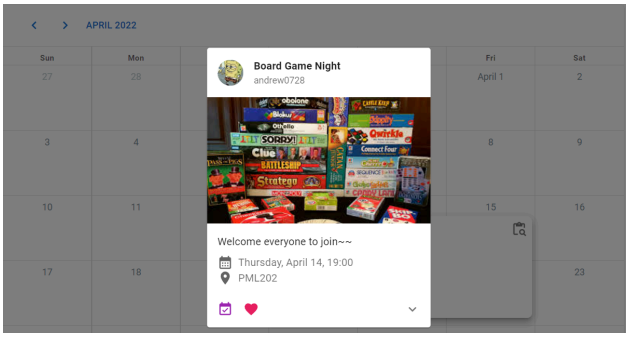


This is the event browser in map view. It implemented the google map service and will show different events on the map with map markers. When clicking a map marker, it will show the corresponding event card. The map view is particularly helpful for users to find events with their location preference.

My Events (Event Calendar)

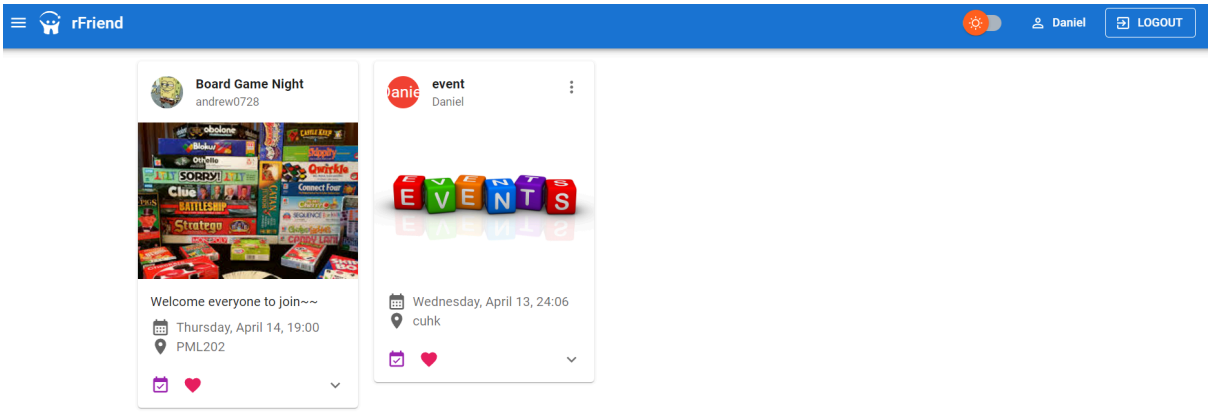




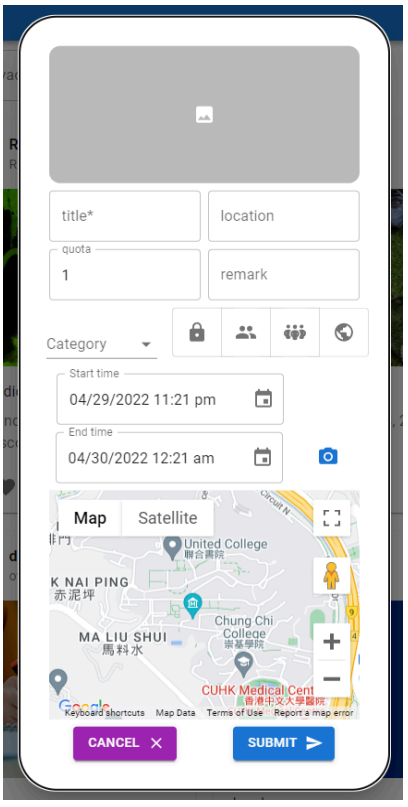


A calendar is implemented to display the events that the users have joined. The calendar will show the events in corresponding time slots. The users can view the event time by clicking the event time slot. Then, users can further look into the event details by clicking the button and calling up the event card. The calendar will be a great tool for users to organize their schedules.

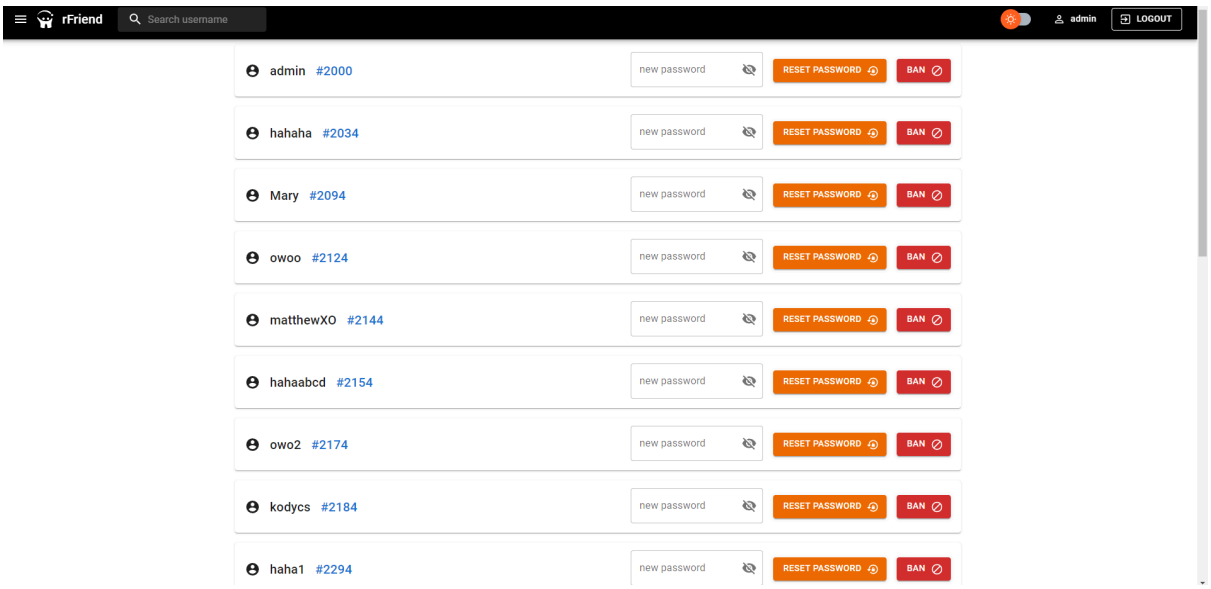
Liked Event Page



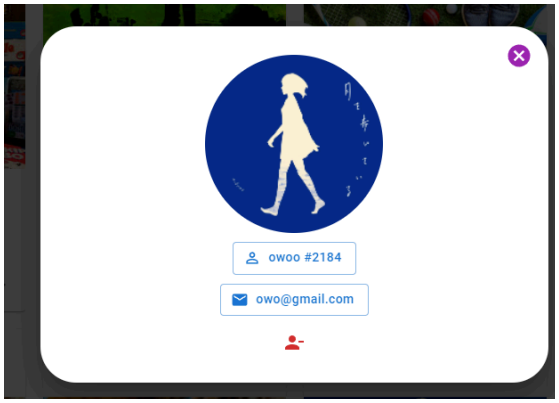
Other than actually joining an event in order to save it, we provided an alternative which is to like an event. After a user has liked an event, the event will be automatically saved, and the user can view it on the liked event page. On the page, users can decide whether they join the events or they can unlike the event to remove it from the page.

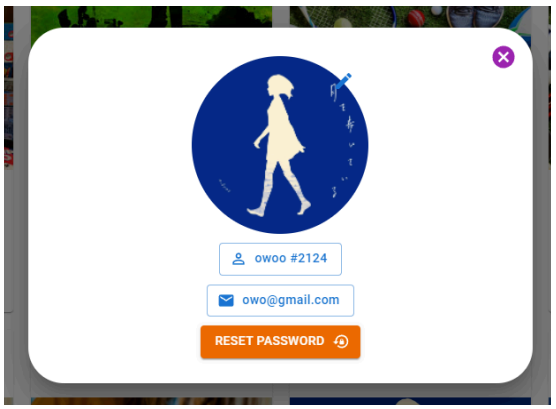


This panel is for users to create/edit events. The following details can be input by users: title, location, quota, remark, category, privacy, start time, end time, event image, map location.

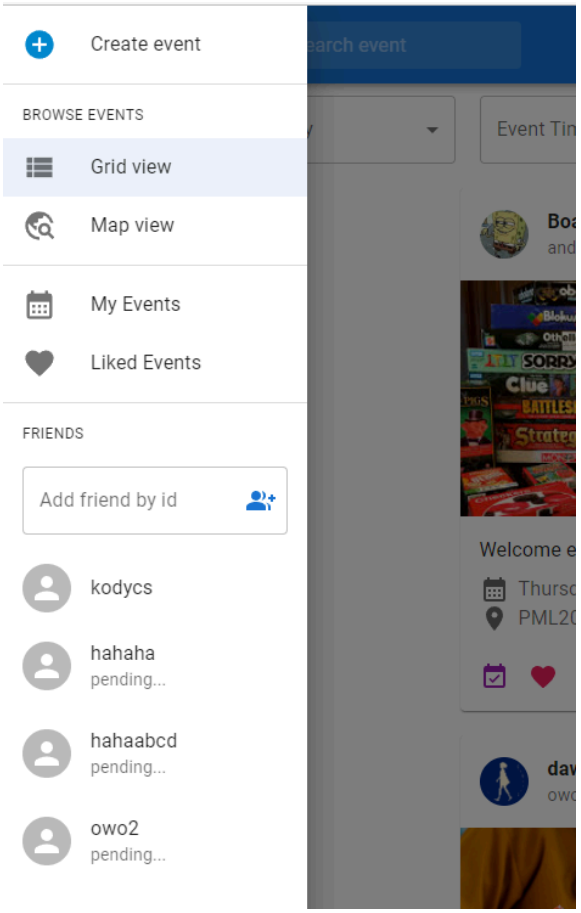


This page is for the admin to reset users' passwords, and ban users.





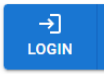
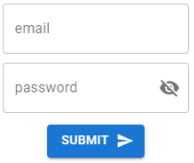
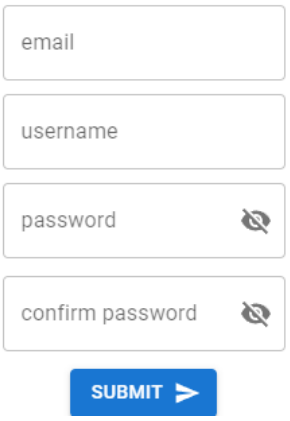
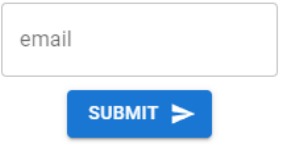
These are 2 examples of user profiles.



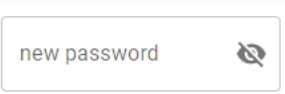
Many functions are stored in the toggle left drawer.

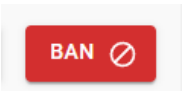
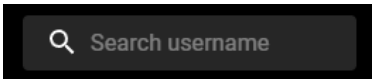
4.3 Objects and Actions

Authentication Page

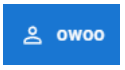
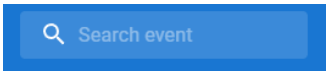
Press to start login	
Type email and password to login	
Press to start register	
Type username, email, password, and confirmed password to register	
Press to start getting back your password	
Type your email to get back your password	
Toggle whether the password is masked	

Admin Page

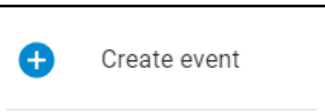
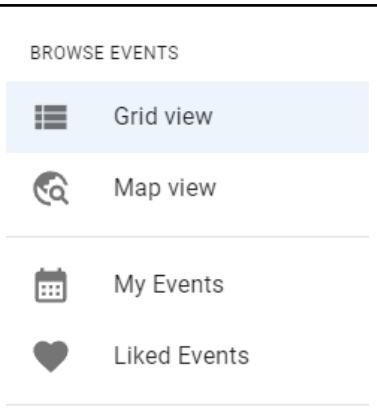
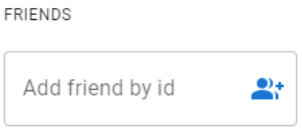
Type the new password of a user	
Reset the password of a user to the new password	

Ban a user	
Type to search user according to username	

Navbar

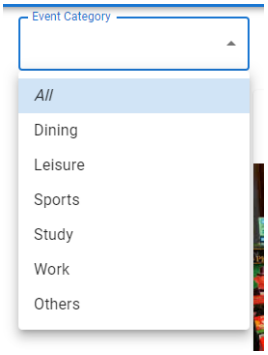
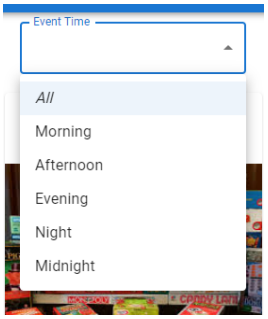
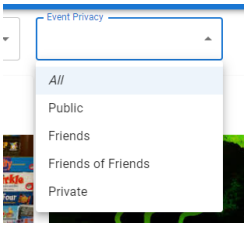
Click to view user profile	
Click to logout	
Click to switch between light mode and dark mode	
Click to open the left drawer	
Type to search event according to event name	

Left drawer

Click to create an event	
Click to switch between grid view, a map view, my events, liked events	
Type friend id and then click the button to add friend	

Click to view the profile of a friend	

Event filter

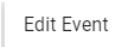
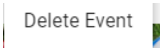
Click to filter event by category	
Click to filter event by time	
Click to filter event by privacy	

Map view

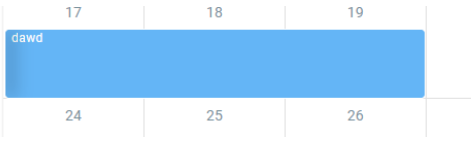
Click to view the event card of the event.	
Drag onto the map to enable street view	

Click to zoom in or zoom out	

Event card

Press to join Event	
Event joined	
Press to like/save event	
Press to unlike event	
Press to show or hide more info section of the event card	
Press to show the options menu for edit and delete event (Only available to event host)	
Type comment and click the button to submit a comment to an event	
Click to edit event details	
Click to delete the event created by you	

Event Calendar

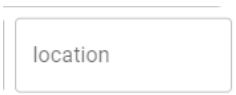
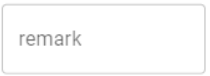
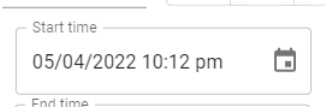
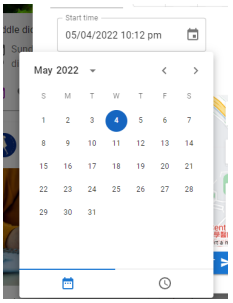
Press to view the detail of an event	
Press to open the event card modal for the event	

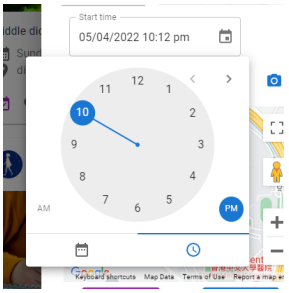
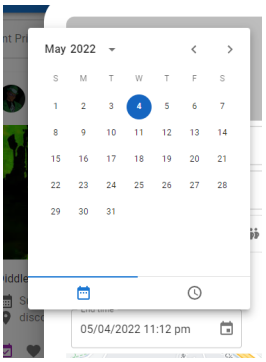
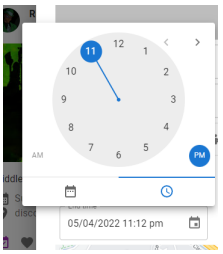
User Profile

Click to email the user, by a system provided mail app.	

Click to reset the password of a user. Only available when you are viewing your own profile	
Click to change the avatar. Only available when you are viewing your own profile	
Click to close the profile	
Click to delete a friend. Only available when you are a friend of the user	

Create/Edit Event Panel

Type to enter the title of the event	
Type to enter the location of the event	
Type or click to adjust the quota of the event	
Type to enter the remark for viewers and participants	
Click to choose the category of the event	
Click to select privacy of the event	
Click to open calendar for choosing start time	
Click to select the start date of the event	

Click to select the start time of the event	
Click to select the end date of the event	
Click to select end time of the event	

5. Lessons Learned

5.1 Frontend

This project brings an opportunity for us to appreciate the beauty of software engineering and improve our web dev skills.

5.2.1 What We Learned - Web development

Web dev skills are important for all kinds of programmers since many interesting applications can only be unlocked when they are able to utilize the internet. Unfortunately, web dev is not a required course for many IT-related students in CUHK. It's lucky to have this project, to give challenges and motivations for us so that we can grab this treasure.

5.2.1 What We Learned - Software Engineering

Although we didn't practice many software engineering theories due to time constraints, looking back, we realize the importance of software engineering. For example, we met certain troubles due to some unstandardized software processes. We didn't standardize everyone to develop based on the Github repository, so it brings trouble when merging the local projects together. On the other hand, we also gained benefits from standardized software processes and interoperability. For example, we did an

API design document, so that frontend developers and backend developers can cooperate timely and correctly by taking a glance at the documentation.

5.2 Backend

5.2.1 Designing the Project Structure

For the backend, since the non-opinionated framework, Express JS, is used, the challenge is to set up a structure that makes sense for the project and provides a decent coding experience for developers. Express JS does not have restrictions regarding how to set up a project, which means one could put all code into a single file, which would work but would lead to very poor maintainability and frustrating developing experience in the long term. It does not even enforce any pattern, which means it is very easy to write messy code with duplicating logic everywhere. With the great flexibility and free choice of the project structure comes great responsibility of creating a structure that is easy to maintain, read, and develop.

A lot of time has been put into picking good practices/structures from popular, well-iterated opinionated frameworks such as PHP Laravel and Ruby on Rails, which are based on the Model-View-Controller pattern, and translating them into JavaScript. Because of the difference in the programming language, the translation is not perfect and compromises need to be made. Careful decisions must be made to ensure the smoothness of development work.

5.2.2 Cloud Deployment on Day One

If a developer has no deployment strategies in mind, it is easy for one to write undeployable code. For example, a developer could implement the image upload function that stores images to a local folder, but it is not deployable because some cloud provider, such as Heroku or other container-based deployments, does not provide writable directories for storage. Although the code works perfectly locally, it would need refactoring before it would work on the cloud. The later the cloud deployment strategies are set and known by developers, the more “technical debt” would accumulate, which could lead to huge refractory costs or even failed delivery.

Foreseeing the possible build-up of the technical debt, cloud deployment is set up on day one and advocated such that the code can be validated continuously, and problems will arise sooner.

6. Conclusion

rFriend is built to serve as a known-friend-focused event organization platform. We are dedicated to making it a user-friendly and powerful tool to settle tedious friends meet-up in our daily life. Besides, we added extra features and functionality to make it more than just a meet-up application in the market. For example, we provide a discovery view, a map view, a calendar view, and a liked event view to make rFriend more comprehensive and versatile in dealing with event exploration and time schedules. Not to mention the friends-of-friends filters, profile picture, and comment features to increase interactions between users.

On behalf of our team, this project has been a challenging yet fruitful experience. It is a long journey to create a worked product from scratch. All of us took part in brainstorming ideas, designing structures, building the software module by module, and finally enhancing the product. It is quite rare to be involved in every part of the process, and we enjoyed it very much. We believe all these little steps contribute to what rFriend has achieved, and we hope rFriend could facilitate friends’ meetups, making a better social life for everyone.

7. Appendix - Tests

7.1 Test Overview and Test Plan

To ensure the project runs smoothly and correctly, different tests are designed for both the frontend and the backend. Both white-box testing and black-box testing are used when designing the tests. In particular, for the frontend, which does not handle data manipulation, black-box testing is used. And for the backend, both methods are used.

7.2 Backend Test 1 - Test of Admin API

Test Set 1

Purpose	To test if we can browse user info with admin role
Inputs	With the following records in the database, User 1 = {id: 1, email: “a@a.com”, name: “user1”, password: “passworda”, profile_url: null, role = “Admin”} User 2 = {id: 2, email: “b@a.com”, name: “user2”, password: “passwordb”, profile_url: null, role = “User”} User 3 = {id: 2, email: “b@a.com”, name: “user2”, password: “passwordb”, profile_url: null, role = “User”} Test case 1 = {id: 1, token: token} Test case 2 = {id: 2, token: token}
Expected Outputs	For the test case id {id: 1, token: token}, we expect the function send a response of all user info given User 1 is the admin. For the test case id = 2 , we expect the function send 401 as User 2 is not an admin.
Pass/Fail Criteria	The test result if any output does not match the expected outputs

Test Set 2

Purpose	To test if we can update user’s password with admin role
Inputs	With the following records in the database,

	User 1 = {id: 1, email: “a@a.com”, name: “user1”, password: “passworda”, profile_url: null, role = “Admin”} User 2 = {id: 2, email: “b@a.com”, name: “user2”, password: “passwordb”, profile_url: null, role = “User”} User 3 = {id: 2, email: “b@a.com”, name: “user2”, password: “passwordb”, profile_url: null, role = “User”} Test case 1 = {id: 1, token: token, target_id: 3, new_password: newpassword} Test case 2 = {id: 2, token: token, target_id: 3, new_password: newpassword} Test case 3 = {id: 1, token: token, target_id: 5, new_password: newpassword}
Expected Outputs	For the test case 1 = {id: 1, token: token, target_id: 3, new_password: newpassword}, we expect the function update the password of User 3 to “newpassword” in the database. For the test case 2 = {id: 2, token: token, target_id: 3, new_password: newpassword}, we expect the function send 401 as User 2 is not an admin. For the test case Test case 3 = {id: 1, token: token, target_id: 5, new_password: newpassword}, we expect the function send 401 as User with id = 5 does not exist.
Pass/Fail Criteria	The test result if any output does not match the expected outputs

Test Set 3

Purpose	To test if we can delete user with admin role
Inputs	With the following records in the database, User 1 = {id: 1, email: “a@a.com”, name: “user1”, password: “passworda”, profile_url: null, role = “Admin”} User 2 = {id: 2, email: “b@a.com”, name: “user2”, password: “passwordb”, profile_url: null, role = “User”} User 3 = {id: 2, email: “b@a.com”, name: “user2”, password: “passwordb”, profile_url: null, role = “User”} Test case 1 = {id: 1, token: token, target_id: 3} Test case 2 = {id: 2, token: token, target_id: 3} Test case 3 = {id: 1, token: token, target_id: 5}
Expected Outputs	For the test case 1 = {id: 1, token: token, target_id: 3}, we expect the function delete the user with id = 3.

	For the test case 2 = {id: 2, token: token, target_id: 3}, we expect the function send 401 as User 2 is not an admin. For the test case Test case 3 = {id: 1, token: token, target_id: 5}, we expect the function send 401 as User with id = 5 does not exist.
Pass/Fail Criteria	The test result if any output does not match the expected outputs

7.3 Backend Test 2 - Tests of Getting User Information

Purpose	To test if we can get the correct user information, including user name, email, the URL to his profile picture and the events he joined, given an user id
Inputs	With the following records in the database, User 1 = {id: 1, email: “a@a.com”, name: “user1”, password: “passworda”, profile_url: null} User 2 = {id: 2, email: “b@a.com”, name: “user2”, password: “passwordb”, profile_url: “https://csci3100project.s3.ap-southeast-1.amazonaws.com/img2”} Test set = {id = 1, id = 2, id = 10}
Expected Outputs	For the test case id = 1, we expect the function send a response containing the name, email and URL of profile picture of the corresponding user For the test case id = 2 , we expect the function send a response containing the name and email of the corresponding user For the test case id = 10, we expect to receive the status code 401 with message “Cannot find user.”

7.4 Backend Test 3 - Tests of Forget and Reset Password

Purpose	To test if the user can change the password under different conditions, including the reset password link is expired and the reset password link is not the latest one. There are 3 tests in this part in total.
Inputs	In the database,

	User 1 = {id: 1, email: “a@a.com”, name: “user1”, password: “passworda”, profile_url: “https://csci3100project.s3.ap-southeast-1.amazonaws.com/img1”} User 2 = {id: 2, email: “b@a.com”, name: “user2”, password: “passwordb”, profile_url: “https://csci3100project.s3.ap-southeast-1.amazonaws.com/img2”} User 3 = {id: 3, email: “c@a.com”, name: “user3”, password: “passwordc”, profile_url: null} Test case 1 = {newpassword = “newpassword0”} For this test case, we try to generate a reset password link and use it to change password immediately. Then, we try to log in using the new password. Test case 2 = {newpassword = “newpassword1”} For this test case, we try to generate multiple reset password links and use the oldest one to change password. Then, we try to log in using the new password. Test case 3 = {newpassword = “newpassword2”} For this test case, we first mock the function being test such that it generates a reset password link that will expire in a extremely short period of time. Then, we generate a link and wait a few seconds. At last, we try to log in using the new password. The image below shows an example.
Expected Outputs	In the first test, we expect to receive status 200 when logging in with the new password. In the second and the third test, we expect to receive status 401 when logging in with the new password, but with message "Invalid forget password token" and "Forget password token expired" respectively.

7.5 Backend Test 4 - Test of User-Event API

Test Set 1

Purpose	To test if user can view events of friends and friends-of-friends based on the privacy setting of the events
Inputs	In the database, User 1 = {id: 1, email: “1111@abc.com”, name: “user1”, password: “passworc”, profile_url: null} User 2 = {id: 2, email: “2222@abc.com”, name: “user2”, password: “password”, profile_url: null} User 3 = {id: 3, email: “3333@abc.com”, name: “user3”, password:

	“password”, profile_url: null} User 4 = {id: 4, email: “4444@abc.com”, name: “user4”, password: “password”, profile_url: null} Event 1 = { eventId=1, "name": "1", "location": "hk", "category": "study", "privacy": "friends", "time": 14, "duration": 10, "max_participants": 4, ownerId=1, participants=[] } Event 2 = { eventId=2, "name": "2", "location": "hk", "category": "study", "privacy": "only-me", "time": 14, "duration": 10, "max_participants": 4, ownerId=2, participants=[] } Event 3 = { eventId=3, "name": "2", "location": "hk", "category": "study", "privacy": "friends-of-friends", "time": 14, "duration": 10, "max_participants": 4, ownerId=3, participants=[] } Friends = {{1,2}, {2,3}, {1,4}} Test set = {id = 1, id = 2, id = 3, id=4}
Expected Outputs	For the test case id = 1, we expect the API return Event 1 and Event 3 as user 1 is user 3 friends of friends. For the test case id = 2 , we expect the API return Event 1, Event 2, and Event 3 as Event 2 is private and User 2 is friends of both User 1 and User 3 For the test case id = 4, we expect the API return Event 1 as User 4 is friends of User 1, but not return Event 3 as User 4 is friends of friends of friends of User 3.
Pass/Fail Criteria	The test result if any output does not match the expected outputs

Test Set 2

Purpose	To test if user can join the event correctly given the participant limit.
Inputs	In the database, User 1 = {id: 1, email: “1111@abc.com”, name: “user1”, password: “passworc”, profile_url: null} User 2 = {id: 2, email: “2222@abc.com”, name: “user2”, password: “password”, profile_url: null} User 3 = {id: 3, email: “3333@abc.com”, name: “user3”, password: “password”, profile_url: null} Event 1 = { eventId=1, "name": "1", "location": "hk", "category": "study", "privacy": "public", "time": 14, "duration": 10, "max_participants": 2, ownerId=1, participants=[1] } Test set = {{id = 1, eventId=1}, {id = 2, eventId=1}, {id = 3, eventId=1}}
Expected Outputs	For the test case {id = 1, eventId=1}, we expect the API return 401 as the User 1 already joined the event.

	For the test case {id = 2, eventId=1} , we expect the API return 200 and User 2 can join the event For the test case {id = 3, eventId=1}, we expect the API return 422 as the event is full and User 3 cannot join the event.
Pass/Fail Criteria	The test result if any output does not match the expected outputs

Test Set 3

Purpose	To test if user can add a comment to an event with user id and event id.
Inputs	In the database, User 1 = {id: 1, email: “1111@abc.com”, name: “user1”, password: “passworc”, profile_url: null} User 2 = {id: 2, email: “2222@abc.com”, name: “user2”, password: “password”, profile_url: null} Event 1 = { eventId=1, "name": "1", "location": "hk", "category": "study", "privacy": "public", "time": 14, "duration": 10, "max_participants": 2, ownerId=1, participants=[1] } Event 2 = { eventId=2, "name": "2", "location": "hk", "category": "study", "privacy": "only-me", "time": 14, "duration": 10, "max_participants": 4, ownerId=2, participants=[] } Test set = {{id = 1, eventId=1, comment: “new comment1”}, {id = 2, eventId=1, comment: “new comment2”}, {id = 1, eventId=2, comment: “new comment3”}}
Expected Outputs	For the test case {id = 1, eventId=1}, we expect the API create a comment: “new comment1” with owner id = 1 and relate it with event id = 1. For the test case {id = 2, eventId=1} , we expect the API create a comment: “new comment2” with owner id = 2 and relate it with event id = 1. For the test case {id = 3, eventId=1}, we expect the API create a comment: “new comment3” with owner id = 1 and relate it with event id = 2.
Pass/Fail Criteria	The test result if any output does not match the expected outputs

7.7 Backend Test 6 - Tests of User Account APIs

Test Set 1

Purpose	To test the user account creation API

Inputs	Case 1:
	To test if user account can be created successfully with no email collision
	`{email: “a@a.com”, password: “00000000”, name: “personA”}`
	
	Case 2:
	To test if user account creation is rejected with email collision.
	With the email a@a.com being used in a record, input:
	`{email: “a@a.com”, password: “00000000”, name: “personA”}`
	
	Case 3:
	To test if user account creation is rejected with invalid inputs
	`{email: “a.com”, password: “00000000”, name: “personA”}`

Expected Outputs	Case 1: Return 200, along with user details and login token Case 2: Return 409, along with a error message Case 3: Return 422, along with a error message
Pass/Fail Criteria	The test result passes if and only if the API returns the corresponding status code, the output meets the API definition

Test Set 2

Purpose	To test the user account login API

Inputs	Case 1:
	To test if user can successfully login with correct credentials
	`{email: “a@a.com”, password: “correct-pw”}`
	
	Case 2:
	To test if login is rejected with incorrect credentials, with the email belong to an existing user
	With the email a@a.com belong to one of the user, input:
	`{email: “a@a.com”, password: “not-correct-pw”}`
	
	Case 3:
	To test if login is rejected with incorrect credentials, with the email does not belong to any user.
	`{email: “b@b.com”, password: “00000000”}`

Expected Outputs	Case 1: Return 200, along with user details and a login token Case 2: Return 401, along with a error message Case 3:

	Return 401, along with a error message
Pass/Fail Criteria	The test result passes if and only if the API returns the corresponding status code, the output meets the API definition

7.8 Backend Test 8 - Tests of Events APIs

Test Set 1

Purpose	To test the event creation API

Inputs	Case 1:
	To test if a event can be created successfully on correct input
	{
	"name": "Lunch",
	"category": "dining",
	"time": "1651769118",
	"duration": "3600",
	"location": "WS Can 3",
	"privacy": "friends",
	"max_participants": 2,
	"coordinate_lat": "22.4242023",
	"coordinate_lon": "114.1868048",
	}
	
	Case 2:
	To test if the event creation is rejected on incorrect input
	{}

Expected Outputs	Case 1: Return 200, along with the newly created event Case 2: Returns 422, with an error message
Pass/Fail Criteria	The test result passes if and only if the API returns the corresponding status code, the output meets the API definition

Test Set 2

Purpose	To test the event update API
Inputs	Case 1: To test if a event can be updated successfully on correct input With an existing event 1 in the database, input:

	{
	"id": "1",
	"name": "Lunch",
	"category": "dining",
	"time": "1651769118",
	"duration": "3600",
	"location": "WS Can 3",
	"privacy": "friends",
	"max_participants": 2,
	"coordinate_lat": "22.4242023",
	"coordinate_lon": "114.1868048",
	}
	
	Case 2:
	To test if the event update is rejected on incorrect input
	{ "id": "1" }
	
	Case 3:
	To test if the event update is rejected when the event does not exist
	{ "id": "9999" }

Expected Outputs	Case 1: Return 200, along with the newly created event Case 2: Returns 422, with an error message Case 3: Returns 422, with an error message
Pass/Fail Criteria	The test result passes if and only if the API returns the corresponding status code, the output meets the API definition

Test Set 3

Purpose	To test the event delete API
Inputs	Case 1: To test if a event can be deleted successfully on correct input With an existing event 1 in the database, input: { "id": "1" } Case 2: To test if the event delete is rejected when the event does not exist { "id": "9999" }
Expected Outputs	Case 1: Return 200, along with the newly created event Case 2: Returns 409, with an error message
Pass/Fail Criteria	The test result passes if and only if the API returns the corresponding status

	code, the output meets the API definition

7.9 Backend Test 9 - Tests of Friends APIs

Test Set 1

Purpose	To test the new friend request API

Inputs	Case 1:
	To test if user can send a friend request to another user when they are not friends
	`{userID: 3}`
	
	Case 2:
	To test if a friend request to themselves is rejected
	With the userID of request is 1, input:
	`{userID: 1}`
	
	Case 3:
	To test if a friend request to an existing friend is rejected
	With the user 1 is already friends with user 2, input:
	`{userID: 2}`
	
	Case 4:
	To test if a friend request is rejected if there is an existing friend request
	With a friend request from the user 1 to user 2, input:
	`{userID: 2}`

Expected Outputs	Case 1: Return 201, along with the request details Case 2: Return 409, along with an error message Case 3: Return 409, along with an error message Case 4: Return 409, along with an error message
Pass/Fail Criteria	The test result passes if and only if the API returns the corresponding status code, the output format meets the API definition

Test Set 2

Purpose	To test the accept friend request API

Inputs	Case 1:
	To test if a friend request can be accepted.
	With a friend request from the user 4 to user 1, input:
	`{userID: 4}`
	
	Case 2:
	To test if a friend request can not be accepted if there is no such friend request
	With no friend request from the user 1 to user 2, input:
	`{userID: 2}`

Expected Outputs	Case 1: Return 200, along a list of user's friends Case 2: Return 409, along with an error message
Pass/Fail Criteria	The test result passes if and only if the API returns the corresponding status code, the output format meets the API definition

Test Set 3

Purpose	To test the delete friend request API

Inputs	Case 1:
	To test if a friend request can be deleted.
	With a friend request from the user 4 to user 1, input:
	`{userID: 4}`
	
	Case 2:
	To test if a friend request can not be accepted if there is no such friend request
	With no friend request from the user 1 to user 2, input:
	`{userID: 2}`

Expected Outputs	Case 1: Return 200, along with a success message Case 2: Return 404, along with an error message
Pass/Fail Criteria	The test result passes if and only if the API returns the corresponding status code, the output format meets the API definition

Test Set 4

Purpose	To test the get friends API
Inputs	Case 1: To test if the friends of a user can be returned

	`{userID: 1}`
	
	Case 2:
	To test when the given user ID does not exist
	`{userID: 9999}`

Expected Outputs	Case 1: Return 200, along with a list of friends Case 2: Return 404, along with an error message
Pass/Fail Criteria	The test result passes if and only if the API returns the corresponding status code, the output format meets the API definition

Test Set 5

Purpose	To test the delete friends API
Inputs	Case 1: To test if a friends of the user can be deleted. With user ID 2 being friends with user 1, input: `{userID: 2}`
Expected Outputs	Case 1: Return 200, along with the latest list of friends
Pass/Fail Criteria	The test result passes if and only if the API returns the corresponding status code, the output format meets the API definition

7.10 Frontend Test 1 - Tests of Frontend Routing

Test Set 1

Purpose	To test if the user is redirected correctly when they login
Inputs	User 1 = {email: “kody@example.com”, password: “12345678”} User 2 = {email: “kody@example.com”, password: “1234567”}
Expected Outputs	For the test case User 1, we expect the user will be redirected to homepage, as it’s a valid normal user. For the test case User 2, we expect the user will be kept at the authentication page, as it’s a invalid normal user.
Pass/Fail Criteria	The test result passes if and only if the expected outputs above are met

Test Set 2

Purpose	To test if the admin is redirected correctly when they register
Inputs	Admin 1 = {email: “admin@example.com”, password: “12345678”} Admin 2 = {email: “admin@example.com”, password: “1234567”}
Expected Outputs	For the test case User 1, we expect the user will be redirected to the homepage, as it’s a valid admin. For the test case User 2, we expect the user will be kept at the authentication page, as it’s a invalid admin.
Pass/Fail Criteria	The test result passes if and only if the expected outputs above are met

Test Set 3

Purpose	To test if the a user is redirected correctly when they visit pages by entering url in the address bar
Inputs	User 1 = {isLoggedIn: true, inputUrl: “/”} User 2 = {isLoggedIn: false, inputUrl: “homepage”}
Expected Outputs	For the test case User 1, we expect the user will be kept at the homepage, as a valid user is not supposed to be viewing the authentication page. For the test case User 2, we expect the user will be kept at the authentication page, as a valid user is not supposed to be viewing the homepage.
Pass/Fail Criteria	The test result passes if and only if the expected outputs above are met

Test Set 4

Purpose	To test if the a user is redirected correctly when logout
Inputs	User 1 = {isLoggedIn: true, clickLogoutButton: true}
Expected Outputs	For the test case User 1, we expect the user will be redirected to the authentication page.
Pass/Fail Criteria	The test result passes if and only if the expected outputs above are met

Test Set 5

Purpose	To test if the an admin is redirected correctly when logout
Inputs	Admin 1 = {isLoggedIn: true, clickLogoutButton: true}
Expected Outputs	For the test case Admin 1, we expect the user will be redirected to the authentication page.
Pass/Fail Criteria	The test result passes if and only if the expected outputs above are met

Test Set 6

Purpose	To test if the an user is restricted from visiting admin page, and an admin is restricted from visiting user page
Inputs	User 1 = {isLoggedIn: true, inputUrl:"/admin"} Admin 1 = {isLoggedIn: true, inputUrl:"/homepage"}
Expected Outputs	For the test case User 1, we expect the user will be kept at the homepage. For the test case Admin 1, we expect the admin will be kept at the admin page.
Pass/Fail Criteria	The test result passes if and only if the expected outputs above are met

7.11 Frontend Test 2 - Tests of Input Fields Component for Authentication

Test Set 1

Purpose	To test if email field is showing error correctly
Inputs	Input set 1 = {email:"owo@exmaple.com"} Input set 2 = {email:"owoexmaple.com"}
Expected Outputs	For the test case Input set 1, we expect no error should be shown. For the test case Input set 2, we expect the error message "Invalid email" should be shown.
Pass/Fail Criteria	The test result passes if and only if the expected outputs above are met

Test Set 2

Purpose	To test if password field is showing error correctly
Inputs	Input set 1 = {password:"12345678"} Input set 2 = {password:"1234567"} Input set 3 = {password:""}
Expected Outputs	For the test case Input set 1, we expect no error should be shown. For the test case Input set 2, we expect the error message "Must have at least 8 characters" should be shown. For the test case Input set 3, we expect the error message "Must have at least 8 characters" should be shown.
Pass/Fail Criteria	The test result passes if and only if the expected outputs above are met

Test Set 3

Purpose	To test if confirm password field is showing error correctly
Inputs	Input set 1 = {password:"12345678", confirmPassword: "12345678} Input set 2 = {password:"12345678", confirmPassword: "12345677"}

Expected Outputs	For the test case Input set 1, we expect no error is shown. For the test case Input set 2, we expect the error message “Doesn’t match” is shown.
Pass/Fail Criteria	The test result passes if and only if the expected outputs above are met

Test Set 4

Purpose	To test if username field is showing error correctly
Inputs	Input set 1 = {username:”hi”} Input set 2 = {username:”owow”} Input set 3 = {username:””}
Expected Outputs	For the test case Input set 1, we expect the error message “Must be at least 4 characters long” is shown For the test case Input set 2, we expect no error message is shown. For the test case Input set 3, we expect the error message “Must be at least 8 characters” is shown.
Pass/Fail Criteria	The test result passes if and only if the expected outputs above are met

7.12 Frontend Test 3 - Test of Event Card Interaction

Test Set 1

Purpose	To test if the join event button is work properly when a user pressed it.
Inputs	Input set 1: Pressing the grey join event button. Input set 2: Pressing the purple event joined button.
Expected Outputs	For test case Input set 1, the grey join event button should turn to purple event joined button and the joined event should appear on the user calendar. For test case Input set 2, as the event has already been joined, the purple event joined button should remain unchanged.
Pass/Fail Criteria	The test result passes if the actual outputs meet the expected outputs

Test Set 2

Purpose	To test if the like event button is work properly when a user pressed it.
Inputs	Input set 1: Pressing the grey like event button. Input set 2: Pressing the pink event liked button.
Expected Outputs	For test case Input set 1, the grey like event button should turn to pink event liked button and the liked event should appear on the liked event page. For test case Input set 2, the pink event liked button should turn to grey like event button and the liked event should disappear on the liked event page.

Pass/Fail Criteria	The test result passes if the actual outputs meet the expected outputs

Test Set 3

Purpose	To test if the option menu button appears only to the host of the event.
Inputs	Input set 1: isHost == true. Input set 2: isHost == false.
Expected Outputs	For test case Input set 1, the grey option menu button should appear on the top right hand corner of the event card and will show the “Edit Event” and “Delete Event” options when pressing it. For test case Input set 2, the grey option menu button should not appear on the top right hand corner of the event card.
Pass/Fail Criteria	The test result passes if the actual outputs meet the expected outputs.

Test Set 4

Purpose	To test if the options inside option menu button work properly.
Inputs	Input set 1: Pressing “Edit Event” button. Input set 2: Pressing “Delete Event” button.
Expected Outputs	For test case Input set 1, a card modal similar to create event should appear on the screen for user to edit the event. For test case Input set 2, the event card should disappear in event browser, calendar and liked event pages.
Pass/Fail Criteria	The test result passes if the actual outputs meet the expected outputs.

Test Set 5

Purpose	To test if the comment section of the event card is able to send comment.
Inputs	Input set 1: Enter the comment in text box and press send button. Input set 2: Empty the comment text box and press send button.
Expected Outputs	For test case Input set 1, the comment text box will be emptyed and the comment should appear in the comment section together with the sender’s avatar. For test case Input set 2, there should be no change in comment section as there is no comment entered.
Pass/Fail Criteria	The test result passes if the actual outputs meet the expected outputs.

7.13 Frontend Test 4 - Test of Create/Edit Event Panel

Test Set 1

Purpose	To test if the upload image button indicating error and success correctly
Inputs	Input set 1 = {image:image} Input set 2 = {image:text file}
Expected Outputs	For test case Input set 1, a sign of success should occur. For test case Input set 2, a sign of warning should occur.
Pass/Fail Criteria	The test result passes if the actual outputs meet the expected outputs

Test Set 2

Purpose	To test if warning sign occurs correctly when the end time is earlier than the start time
Inputs	Input set 1: {startTime:”1651679268”,endTime:”1651679000”} Input set 2: {startTime:”1651679268”:”1651679333”}
Expected Outputs	For test case Input set 1, a warning “” should occur. For test case Input set 2, no warning “” should occur.
Pass/Fail Criteria	The test result passes if the actual outputs meet the expected outputs

Test Set 3

Purpose	To test if the location marker occurs on the right position when user click a location on google map
Inputs	Input set 1: {Coordinate:{x:127,y:138}} Input set 2: {Coordinate:{x:100,y:128}} Input set 3: {Coordinate:{x:0,y:0}} Input set 4: {Coordinate:{x:300,y:300}} Input set 5: {Coordinate:{x:400,y:99}}
Expected Outputs	Output set 1: {Coordinate:{x:127,y:138}} Output set 2: {Coordinate:{x:100,y:128}} Output set 3: {Coordinate:{x:0,y:0}} Output set 4: {Coordinate:{x:300,y:300}} Output set 5: {Coordinate:{x:400,y:99}}
Pass/Fail Criteria	The test result passes if the actual outputs meet the expected outputs

Test Set 4

Purpose	To test if the input field of event quota is preventing negative value
Inputs	Input set 1: {keyboardInput:”-10”}

	Input set 2: {timesOfClickingDecreaseArrow:10}
Expected Outputs	Output set 1: {0} Output set 2: {0}
Pass/Fail Criteria	The test result passes if the actual outputs meet the expected outputs

7.14 Frontend Test 5 - Test of User Management Components for Admin

Test Set 1

Purpose	To test if a warning occurs correctly when an admin enters a new password, which is less than 4 characters, for an user.
Inputs	Input set 1: {password:”123”} Input set 2: {password:”1234”}
Expected Outputs	For test case Input set 1, a warning “” should occur. For test case Input set 2, no warning “” should occur.
Pass/Fail Criteria	The test result passes if the actual outputs meet the expected outputs

Test Set 2

Purpose	To test if ban button is indicating the loading status and error status correctly
Inputs	Input set 1: {doesTheBanningUserExist:true} Input set 2: {doesTheBanningUserExist:false}
Expected Outputs	For test case Input set 1, it should show a loading appearance first, and then show a successful appearance. For test case Input set 2, it should show a loading appearance first, and then show a failed appearance.
Pass/Fail Criteria	The test result passes if the actual outputs meet the expected outputs

7.15 Frontend Test 6 - Test of Friend Components

Test Set 1

Purpose	To test if add friend button is showing error sign correctly
Inputs	Input set 1: {userIdToAdd:”2000”}//2000 is an existing user Input set 2: {userIdToAdd:”2111”}//2111 is an non-existing user
Expected Outputs	For test case Input set 1, it should show a loading appearance first, and

	then show a successful appearance. For test case Input set 2, it should show a loading appearance first, and then show a failed appearance.
Pass/Fail Criteria	The test result passes if the actual outputs meet the expected outputs

Test Set 2

Purpose	To test if add friend id input field only allow inputting numbers
Inputs	Input set 1: {keyboardInput:”sjdawj2dsad”} Input set 2: {keyboardInput:”2000”}
Expected Outputs	Output set 1: {value:”2”} Output set 2: {value:”2000”}
Pass/Fail Criteria	The test result passes if the actual outputs meet the expected outputs

7.16 Frontend Test 7 - Test of Dark Mode

Test Set 1

Purpose	To test if darkMode is working correctly
Inputs	Input set 1: {currentMode:”light”, isButtonClick:true} Input set 2: {currentMode:”night”, isButtonClick:true}
Expected Outputs	For test case Input set 1, the output should be dark mode. For test case Input set 2, the output should be light mode.
Pass/Fail Criteria	The test result passes if the actual outputs meet the expected outputs

Test Set 2

Purpose	To test if darkMode is working correctly under different page
Inputs	Input set 1: {currentMode:”light”, selectView:”AdminView”} Input set 2: {currentMode:”light”, selectView:”FavouriteView”} Input set 3: {currentMode:”light”, selectView:”MapView”} Input set 4: {currentMode:”light”, selectView:”MyEvents”} Input set 5: {currentMode:”light”, selectView:”GridView”} Input set 6: {currentMode:”light”, selectView:”Left drawer”} Input set 7: {currentMode:”light”, selectView:”Authentication pager”} Input set 8: {currentMode:”night”, selectView:”AdminView”} Input set 9: {currentMode:”night”, selectView:”FavouriteView”} Input set 10: {currentMode:”night”, selectView:”MapView”} Input set 11: {currentMode:”night”, selectView:”MyEvents”} Input set 12: {currentMode:”night”, selectView:”GridView”} Input set 13: {currentMode:”night”, selectView:”Left drawer”}

	Input set 14: {currentMode:”night”, selectView:”Authentication pager”}
Expected Outputs	For test case Input set 1 to 7, the output should be light mode. For test case Input set 8 to 14, the output should be dark mode.
Pass/Fail Criteria	The test result passes if the actual outputs meet the expected outputs

7.17 Frontend Test 8 - Test of Event Filter and Search Bar

Test Set 1

Purpose	To test if event filter is working properly
Inputs	Input set 1: {filterCategory:“dinning”, filterTime:“Evening”} Input set 2: {filterCategory:“study”, filterPrivacy:“only-me”} Input set 3: {filterCategory:“all”, filterTime:“all”, filterPrivacy:“all”}
Expected Outputs	For test case Input set 1, the output should be only dinning events start between 5pm to 8:59pm remain on the grid view. For test case Input set 2, the output should be only study events that users created and with privacy level “only-me” remain on the grid view. For test case Input set 3, the output should be all event available to the user are shown on the grip view.
Pass/Fail Criteria	The test result passes if the actual outputs meet the expected outputs

Test Set 2

Purpose	To test if search bar is working properly
Inputs	Input set 1: Search input: “EvEn” Input set 2: Search input: “LOL tournament” Input set 3: Empty search input to “”
Expected Outputs	For test case Input set 1, all the events with name containing “even”(case insensitive) will remain on the grip view, eg. “Dinning event”. For test case Input set 2, all the events with name containing “LOL tournament” will remain on the grip view, eg. “LOL HK tournament” but not “LOLTournament”. For test case Input set 3, all events will be displayed again.
Pass/Fail Criteria	The test result passes if the actual outputs meet the expected outputs