The CUHK Society Promotion Platform

Final Report

Document Version Number
1.0.0
Printing Date
30/4/2021

Group ID: D4
Member names and SID:
Chung Ka Ho Ryan 1155125428
Cheng Ka Pui 1155125534
Lam Puy Yin 1155126240
Lau Tsz Man 1155108216
Wong Tsz Him Iknos 1155124886

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

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Project Overview

This project, CUHK Society Promotion Platform (CUSocPro), is aimed to provide an all-in-one events promotion platform to CUHK stakeholders. The website is developed by using the MERN stack, which stands for MongoDB, Expressjs, Reactjs and Nodejs. The server is powered by Express and MongoDB for offering the API routing and data storage service respectively. With the interactive design, dynamic subscription system and personalized user experience provided by the access control mechanism, it is expected to provide a comprehensive user experience to both the event holders and potential participants.

1.2 Problem Statement

CUHK is a big society with various events held each day, however, there does not have a standardised event collaboration platform for student acknowledgement that the student can only rely on social media to gather such information. The current CUHK Mass Mail System is presented to serve its purpose of public promotions and announcements, yet, the current state of the system is not optimised, and could not meet today's standard with the following concerns:

- Excessive information is presented in the mail without a clear classification and filtering, and the event notifications are displayed in a rough and junky format. It is costly for students to read every line of the announcement list, and some of them may just randomly skim through the list and miss important announcements.
- Limited multimedia elements available in the Mass Mail System makes it hard to get attention from the readers and difficult to search for interesting events among the events list.
- Lack of existing channels to cover the promotion needs of the activity of CUHK clubs and societies. Regarding there is no common platform for the clubs to list out their event currently, all society is promoting their events solely by Facebook and peers referrals, with the limited social circle of every exco-members may have, it is difficult for them to reach the activity to the potential member.

In the light of this, the CUHK Society Promotion Platform is proposed to alleviate the observed drawbacks in the current CUHK Mass Mails system, as well as increasing coverage of every event to students, and increase the overall participation rate.

1.3 Objective

The CUHK Society Promotion Platform aims to supplement the existing Mass Mail System for events delivery for society, as well as managing a user-friendly interface with a dynamic subscription and customized calendar tracker system for students, in order to provide a comprehensive and well-organized event collaboration system for CUHK stakeholders.

1.4 Expected Customers and Market

The targeted user groups of the CUHK Society Promotion Platform are CUHK students and student societies. It is expected that students can have a quick understanding of all CUHK's available events by the well-organized event catalogue. Meanwhile, all societies can increase their exposure by having an open and fair opportunity to promote their organized events in a unified platform.

1.5 Highlights

The CUHK Society Promotion Platform enables CUHK societies to promote their upcoming events to CUHK students. To ensure the account is registered by a CUHK student, CUHK webmail (student-id@link.cuhk.edu.hk) has to be presented to sign up for an account. Meanwhile, the society accounts will be assigned and allocated by the CUSocPro management team. While the system provides an access control mechanism that allows a different user experience regarding their user type, the credibility and reliability of the system will be strengthened.

Regarding the functions provided to student users, the system provides a customized experience by having a subscription system that student users can subscribe to different societies and receive unified event notification. The user can also view the interactive calendar in an effective format, which shows all the events of the interested society. The system may also provide a weekly digest of activities that an email will shortlist all the upcoming activities for the following week. For effective events and societies browsing, filtering and searching functions are provided: students can filter the tag or dates in order to focus on their area of interest, or locate their targeted events by inputting keywords directly.

Regarding the functions provided to society users, the system can post events with a visual illustration attached. Event details can be edited and deleted anytime by using the same account. In addition, society can review their subscription number for their reference.

Also, the system provides a seminar feature that the society can host a real-time seminar for users to join, while both participants and the host can chat with each other. Moreover, for the

better management of the seminar, the host is able to mute any participant if there are any inappropriate activities happening during the seminar.

The systems encourage interactive reaction between students and the societies. Apart from the seminar function, all users can leave comments on the post and reply to the other comments in the post to express their views.

Last but not least, a responsive web design has been implemented that the system can be accessed seamlessly on the mobile device and provides an excellent user experience.

1.6 Project Statistics

1.6.1 Front End

Statistic in JavaScript

(The following statistics are calculated by a tool called Plato)

Filename	Maintainability	Complexity	Source Lines of code	Estimated Errors	Difficulty
App.test.js	80.08	1	8	0.04	5.82
config.js	70.9	1	9	0.06	2.50
index.js	75.54	1	21	0.08	4.55
EventCalendar.js	71.99	3	81	0.48	14.88
Home.js	76.36	1	35	0.10	76.36
GetStart.js	65.41	1	36	0.15	5.36
Login.js	68.9	5	123	0.64	15.45
Signup.js	74.91	3	163	0.81	16.78
NotFound.js	96.33	1	21	0.06	6.09
PostBrowser.js	78.64	4	127	0.69	13.48
PostContent.js	69.2	27	344	2.70	30.39
PostCreate.js	86.24	2	40	0.19	10.41
PostForm.js	66.67	16	336	2.82	47.30
StudentProfile.js	71.36	6	87	0.52	32.43
SeminarCreate.js	81.86	3	45	0.21	11.44
SeminarEntrance.js	76.16	7	136	0.81	16.93
SeminarForm.js	76.18	5	202	1.19	32.43
SocietyBrowser.js	73.92	1	31	0.12	7.27
DataPagination.js	87.56	4	42	0.33	40.00
NavMenu.js	67.9	4	76	0.44	16.51

SocietyList.js	81.94	4	91	0.50	22.18
AccountSystem.js	74.33	8	93	0.63	30.12
Post.js	78.92	4	104	0.71	24.58
SearchEngine.js	87.09	1	38	0.26	18.50
Seminar.js	83.6	2	22	0.13	9.57
Society.js	75.77	3	68	0.49	17.52
SocietyInfo.js	85.59	4	50	0.37	21.70
Student.js	83.03	4	84	0.50	25.28
StudentInfo.js	83.6	3	39	0.23	19.43
Tag.js	96.09	1	11	0.08	7.33
service-worker.js	78.1	8	99	0.55	19.85
serviceWorkerRegistration.js	78.63	18	137	0.71	26.69
setupTests.js	100	1	5	0	1.00

Summary

Total / Average Lines	Average Maintainability
2804 / 84	78.87

1.6.2 Back End

Statistic in JavaScript

(The following statistics are calculated by a tool called Plato)

Filename	Maintainability	Complexity	Source Lines of code	Estimated Errors	Difficulty
config.js	64.33	1	16	0.05	2.67
authMiddlewares.js	88.61	1	20	0.10	4.36
postMiddlewares.js	80.74	14	174	1.78	38.72
seminarMiddlewares.js	78.5	7	80	0.71	25.71
subscriptionMiddleware.js	76.98	13	119	1.09	30.48
userInfoMiddleware.js	89.28	2	68	0.52	15.70
userMiddlewares.js	84.22	1	75	0.54	12.05
PostModel.js	43.82	1	43	0.40	10.83
SeminarModel.js	53.29	1	18	0.16	7.11
UserModel.js	40.14	1	58	0.54	12.97
Post.js	71.08	40	397	3.12	56.80
Society.js	69.5	22	255	1.79	50.24
Student.js	69.2	27	356	2.29	51.21
Tag.js	81.44	2	26	0.13	9.50
User.js	67.62	12	211	1.33	30.58
mongo.js	83.6	2	51	0.35	10.82
postRouter.js	55.6	1	41	0.25	5.20
seminarRouter.js	67.85	1	12	0.10	5.00
subscriptionRouter.js	65.28	1	23	0.10	3.52
userInfoRouter.js	65.2	1	24	0.10	3.70
userRouter.js	65.2	1	22	0.10	3.70

server.js	100	3	47	0.40	8.47

Summary

Total / Average Lines	Average Maintainability
2147 / 97	70.98

1.6.3 Statistics Metric in Javascript

- Maintainability: A value between 0 and 100 represents the relative ease of maintaining the code. A high value means better maintainability.
- Complexity: This metric counts the number of distinct paths through a block of code. Lower values are better.
- Lines of code: Sources line of code/ Logical Lines of Code.
- Estimated Errors: Halstead's delivered bugs is an estimate for the number of errors in the implementation.
- Difficulty: The difficulty measure is related to the difficulty of the program to write or understand.

2 SYSTEM ARCHITECTURAL DESIGN

2.1 System Architecture

2.1.1 Architecture diagram

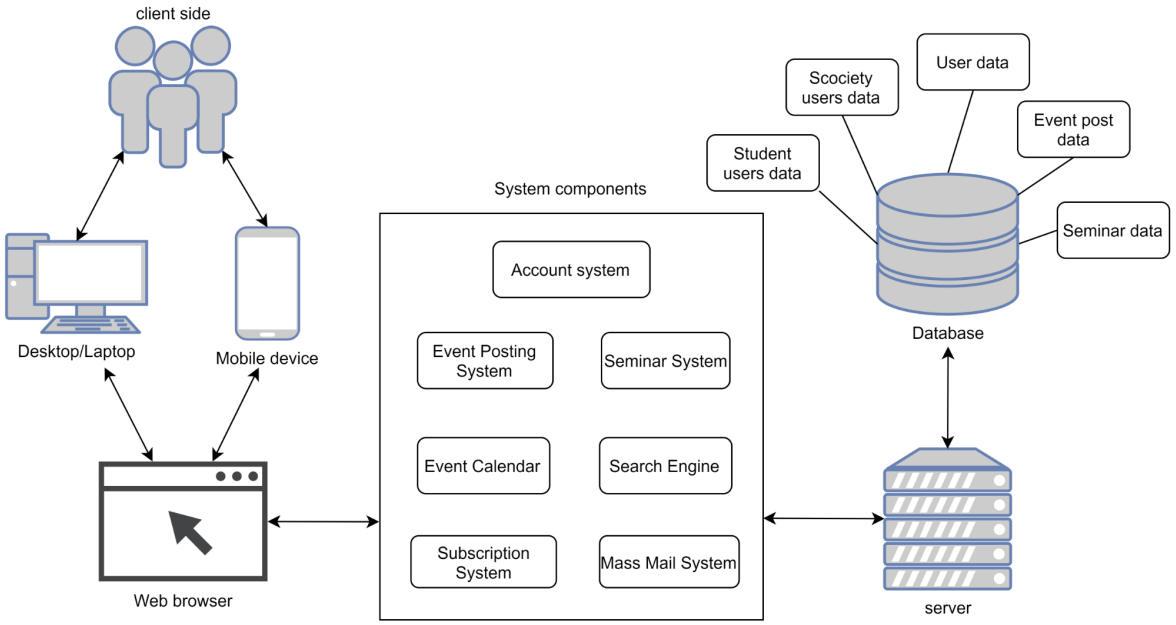

Figure 1. Architecture Diagram

The architecture diagram (Figure 1) presents the system architecture blueprint of the CUSocPro system. The end-users can access CUSocPro by visiting the progressive web app and interact with the function through the user interface, which triggers the Restful APIs to perform record insert, read, update, and delete in the backend database server.

2.1.2 System Components

2.1.2.1 Account System

A student account can be created by setting up the username, password and a CUHK email, i.e. student-id@link.cuhk.edu.hk. For verifying the identity of the email address, a verification email that contains an account activation link, will be sent to the email. By clicking the activation link, the backend system will validate the activation token and activate the account. Users will then be able to login into their account using their userID and password.

2.1.2.2 Event Calendar

The calendar will retrieve the subscription information from the database and displays the event schedule of the subscribed societies of the student user. While the component is only available to the student users, society users will be redirected back to the Home page upon their visits to the page. The calendar can be resized by selecting the data frame size like monthly or weekly view, so as to achieve a personalized experience.

2.1.2.3 Event Posting System

The event posting system is able to handle different operations such as creation, edit, deletion and update of the events posts, which brings a more flexible experience to event promotion. An event can only be created by verified society users while the event edition and deletion are only authorized to the post creator. For better discussion and communication, both society and student users are able to write their comments to the post for expressing their thoughts or enquiries. Also, statistics such as the number of viewers can be retrieved from each post, the society can use the data for further analysis of their performance, as well as their customer reaching strategy.

2.1.2.4 Seminar System

To reinforce the interactive experience, a feature, seminar system, is provided for the society to hold a seminar for the users to join, which enables a real-time textual interaction between different parties. A seminar moderator is offered with the seminar room management functions which include “mute” and “kick”. A muted participant will be disabled from sending the chat message in the room while a kicked participant will be forced to quit the room and disable for entering the same room within 20 minutes. These seminar room management functions enable a proper social manner are upheld by every participant in this social exchange environment.

2.1.2.5 Event dashboard with the advanced search engine system

The event dashboard provides a quick review of all the events held by all societies, filtering and searching functions are provided for effective events browsing. The search engine performs keyword searching as well as event tags and dates filtering by retrieving the data from the database via query and data matching.

2.1.2.6 Subscription System

The subscription system allows the student users to subscribe to their interested societies. Whenever the subscribed society creates a new post, a notification email shall be sent to the student's email. Moreover, the events held by the subscribed societies shall be displayed in the student's calendar. It is believed that all of these functions provided will bring convenience to the student users in receiving the news from their subscribed societies timely.

2.1.2.7 Mail System

The Mail System provides an email delivery service for the new event notification, weekly digest and account verification. For the new event notification, a notification email shall be sent to the societies' subscribers automatically after the creation of a post. While the weekly digest contains the abstracts of the new posts created by the subscribed societies in the last week. The weekly digest shall be automatically sent to the student's email every weekend. The mail system is implemented in the backend server and it will retrieve the user or post information from the database for generating the customized email for every student user.

2.2 Data Flow Diagram

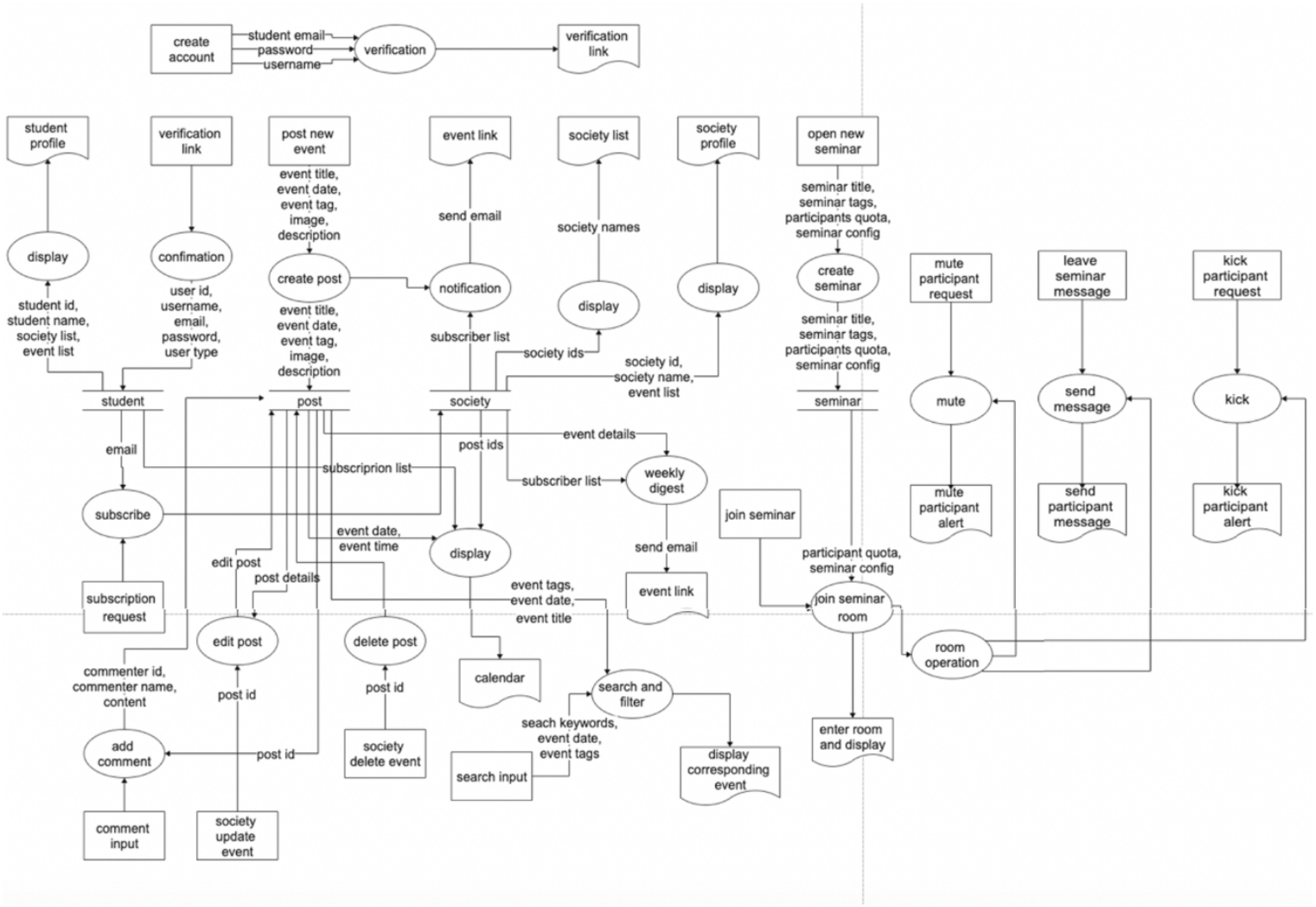

Figure 2. Data Flow Diagram

The data flow diagram shows the flow of the data between the components in the system.

3 DETAILED DESCRIPTION OF COMPONENTS by UML

3.1 Use-case Diagram

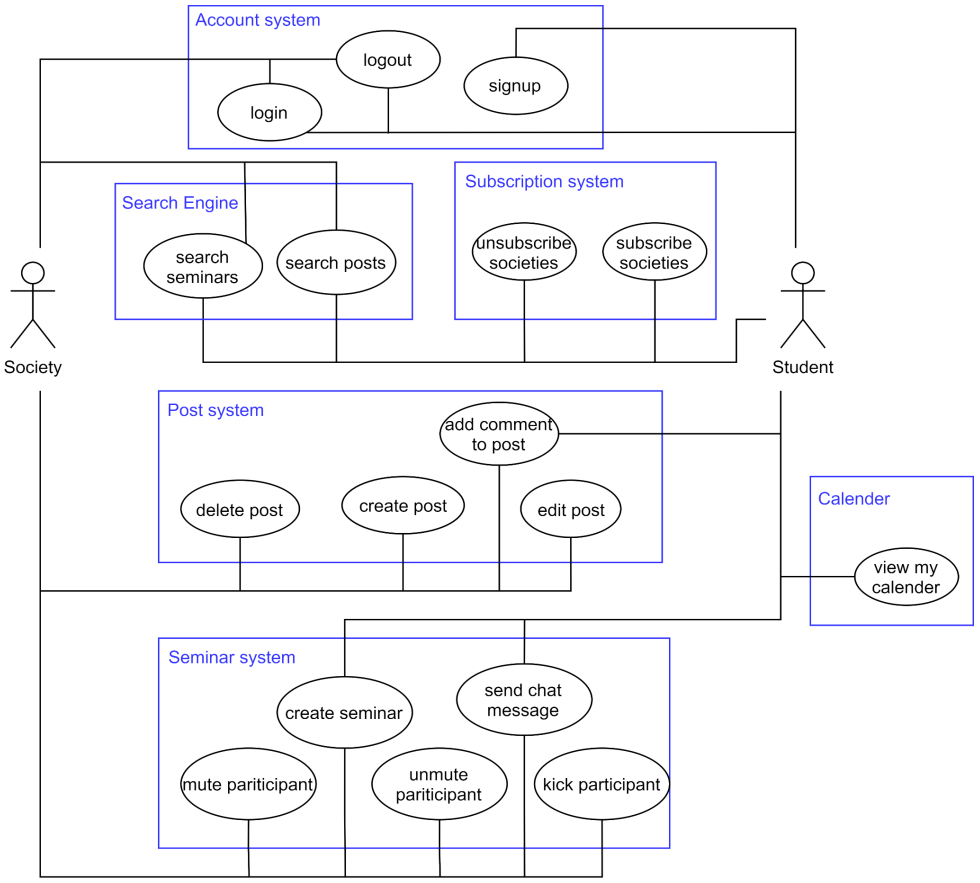

Figure 3. Use-case Diagram

The use-case diagram shows the user journey of the end-users. Start with the sign-up function for verifying the identity of students via CUHK email. While the account registration is done by the management team society users, requires verification to society's certificate for allocating an account. Both student users and society users use the login function to login into their accounts and use the logout function to log out. After logging in, they shall be able to use other applicable functions corresponding to their user type.

Some application functions can be generally used by both society and student users, such as attaching comments to posts, sending chat messages in the seminars and the search function in the event dashboard. For the others, the subscription function and calendar function are specifically used by student users, while the event posting and seminar management functions are specifically used by society users. The rest of the functions and components such as the mass mail system, which are not present in the graph, are not in touch with the end-users because they are run internally within the systems, but not activated by the users directly.

3.2 Class Diagram

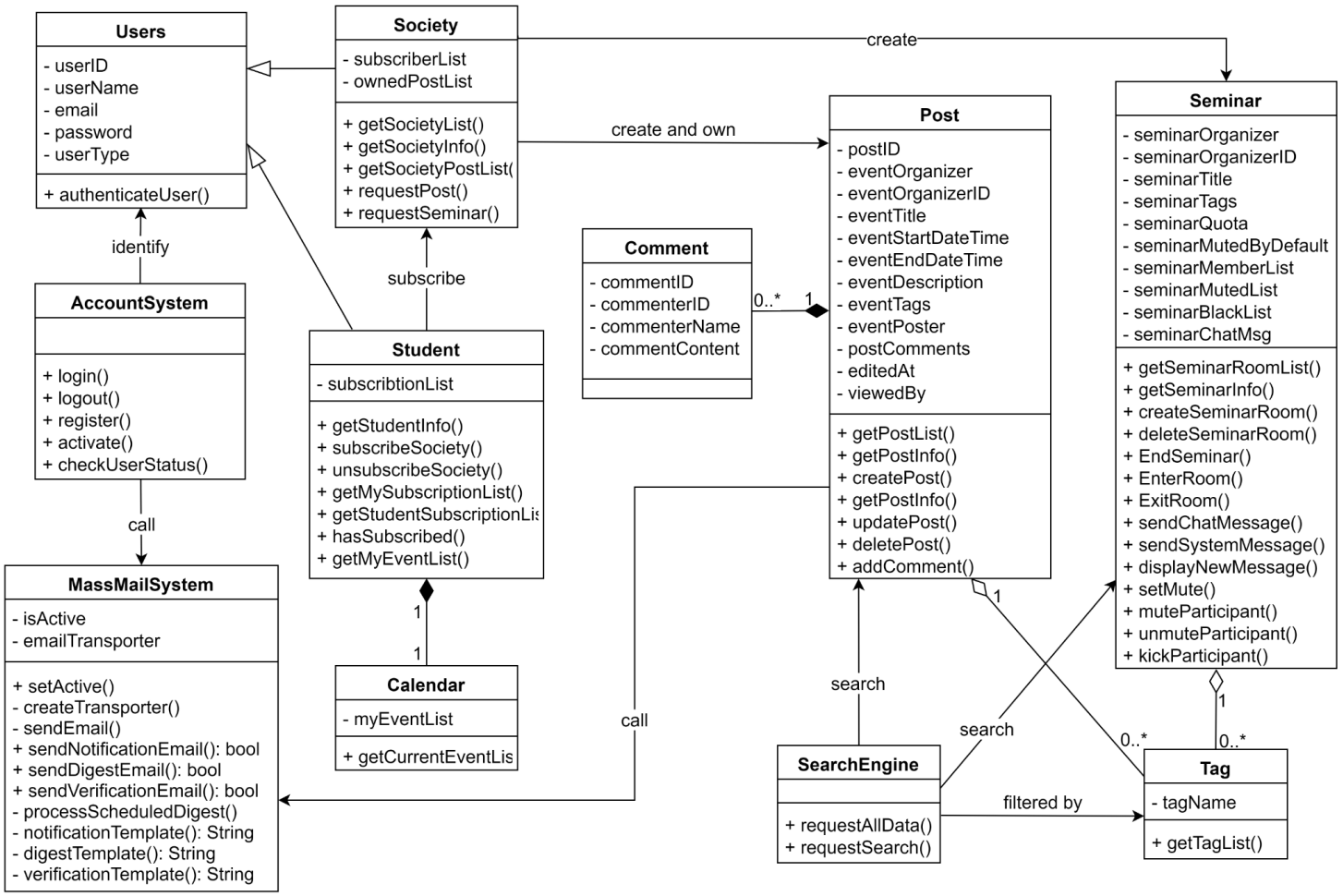

Figure 4. Class diagram

The class diagram shows the relations, as well as the coherence and cohesion, between various relations that builds up the system.

3.3 Sequence Diagram

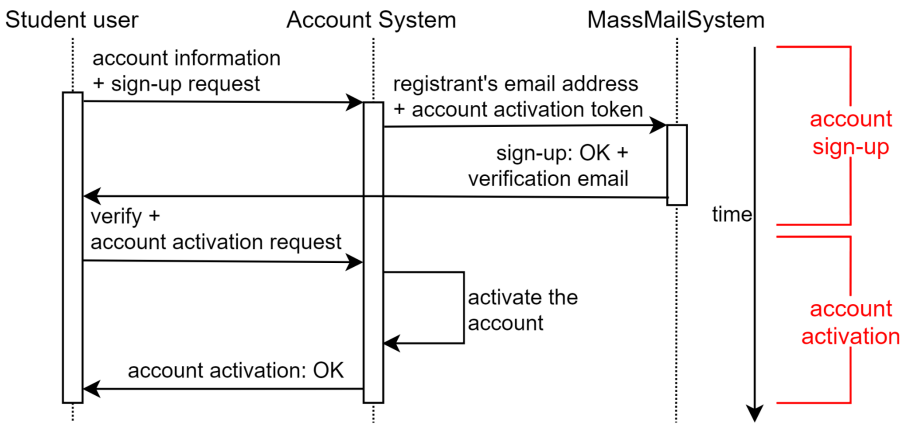

Figure 5. Sequence Diagram for the account sign-up and account activation

Step-by-step description:

1. A student user clicks on the “sign up” hyperlink on the login page and route to the sign-up page.
2. The student user enters the user name, CUHK email address, password and the confirm password in the relevant input boxes on the sign-up form.
3. The student user clicks the “Sign Up” button to request sign-up with the account information.
4. The server signs a JSON web token used for the account activation and constructs an URL routing to the account activation page for the signed up account.
5. The server sends a verification email that contains the account activation URL to the student user’s email address.
6. The student user opens the account activation page from the received email.
7. The student user clicks on the “Activate” button.
8. The server verifies the JSON web token parsed from the URL and activates the account upon a valid token.
9. The signed up account is activated successfully and a button for returning to the login page is shown.

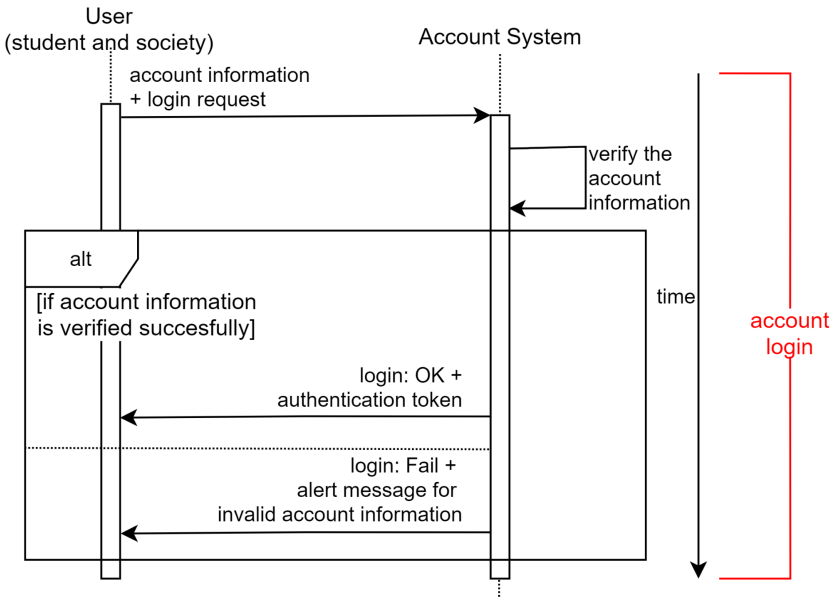

Figure 6. Sequence Diagram for the account login

Step-by-step description:

1. A user clicks on the “Get Started” button on the welcome page and route to the login page.
2. The user enters the user ID and the password to the relevant input boxes in the login form.
3. The user clicks the “Sign in” button to request login with the account information.
4. The server verifies the submitted account information.
5. If the submitted account information is valid, the server signs a JSON web token used for the authentication and sends it back to the user’s browser, otherwise, an alert message for the invalid account information shall be shown up.

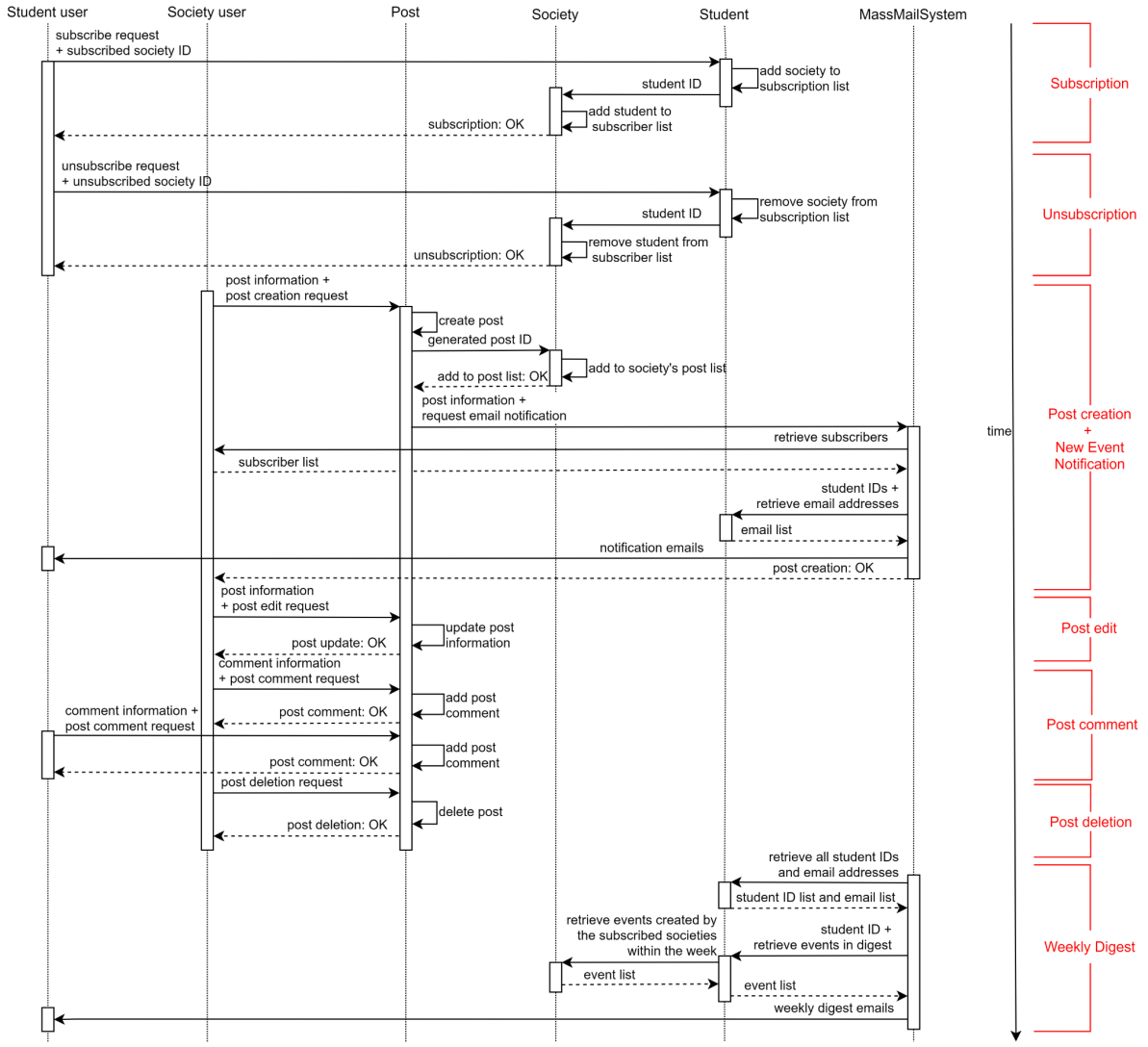

Figure 7. Sequence Diagram for the subscription system, event posting system, new event notification and weekly digest

Step-by-step description:

1. A student user clicks on the “Subscribe” button on the profile page of an interested society.
2. The subscribed society is added to the subscription list of the student.
3. The student is added to the subscriber list of the subscribed society.
4. When the subscription succeeds, the “Subscribe” button on the society profile page will be changed to the “Unsubscribe”.
5. The student user clicks on the “Unsubscribe” button on the society profile page of a subscribed society.
6. The record of subscribed society is removed from the subscription list of the student.
7. The record of the subscribing student is removed from the subscriber list of the society.

8. Upon the unsubscription succeeds, the “Unsubscribe” button on the profile page is changed to the “Subscribe”.
9. The society user clicks on the “Create Post” button on the event browser page.
10. The user enters the event title, dates, tags, description and selects the image file of the event poster to the relevant input boxes on the post creation form.
11. The user clicks the “Submit” button to activate the post creation function with the submitted event post information.
12. The server creates the post records in the database.
13. The created post is added to the post list of the society user.
14. The server sends a notification email which contains the event title and the URL of the event post to the email of the subscribers of the society user.
15. The society user clicks on the “edit” button on the post.
16. The society user may update the event title, event dates, event tags, event description or selects a new image file of the event poster in the relevant input boxes on the post edit form.
17. The user clicks the “Submit” button to activate the post edit function with the submitted event post information.
18. The server updates the post record in the database.
19. All other users may enter a comment to the input box in the comment section under the post.
20. The user clicks the “Submit” button to activate the comment adding function.
21. The server pushes the comment to the post’s record in the database.
22. Upon the success of adding the comment to the post, the added comment is displayed in the comment section of the post.
23. The society user clicks on the “delete” button on the post to activate the post deletion.
24. The server deletes the post record in the database.
25. Upon the success of the post deletion, the post will be discarded and the society user will be redirected to the post browser page.
26. The server retrieves the student IDs and email addresses of all student users from the database.
27. The server retrieves the events whose post is created by the subscribed societies of each student within the week.
28. The server sends a weekly digest email, which contains the list of event titles and URLs of the event posts created by the subscribed societies of the student within the week, to each student user’s email address.

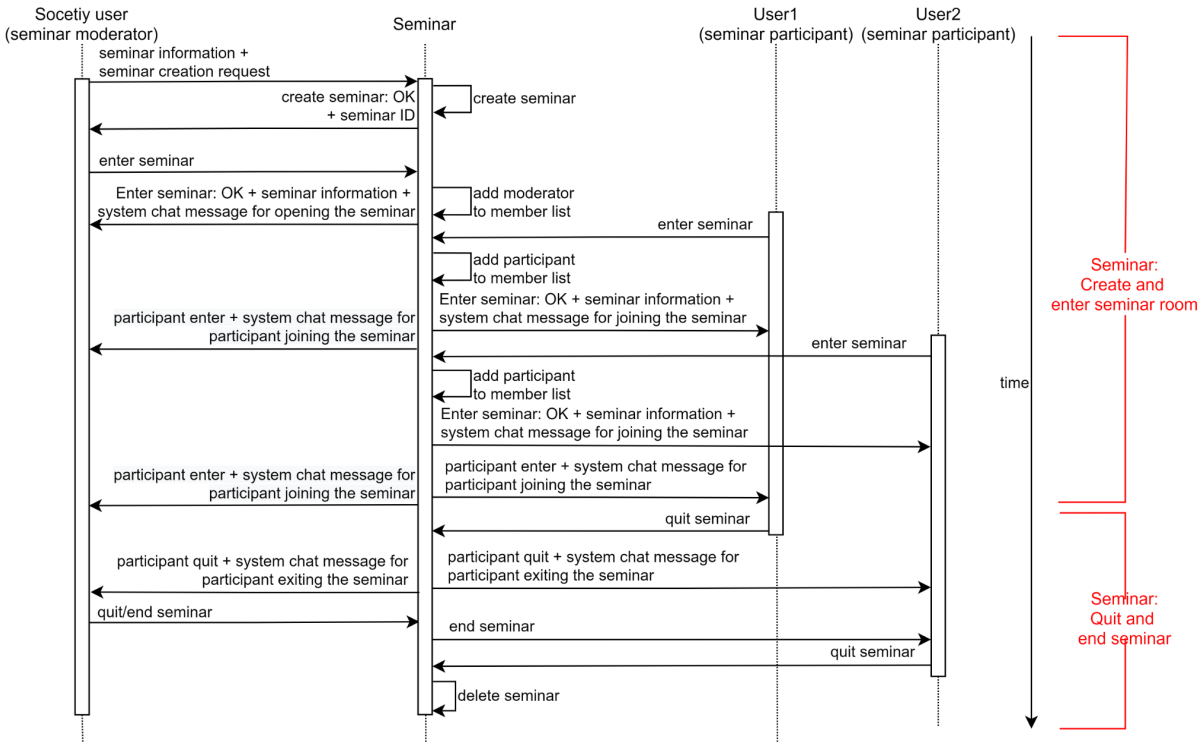

Figure 8. Sequence Diagram for creating, entering, quitting and ending the seminar

Step-by-step description:

1. The society user clicks on the “Create Seminar” button on the seminar home page.
2. The society user enters the seminar title, quota, tags to the relevant input boxes and checks the box for deciding whether the seminar policy of muting the participants should be implemented by default in the seminar creation form.
3. The society user clicks the “Submit” button to activate the seminar creation with the seminar information.
4. The server generates a seminar ID and creates the seminar record in the database.
5. The generated seminar ID for the newly created seminar will be returned to the society user.
6. The society user will be redirected and enters the seminar room corresponding to the seminar ID.
7. The server constructs a session for the seminar and adds the society user who creates the seminar, as well as the user in the member list of the seminar room with the member type “moderator”.
8. A signal will be returned to the society user upon the successful connection to the seminar room. An automatic chat message indicating the success of opening the seminar is displayed in the chatbox of the seminar room.
9. Other users can enter the seminar by clicking on the seminar widget on the seminar entrance page.

10. The server adds the user to the member list of the seminar room and registers the member type as “participant”.
11. A system message will be broadcasted to all members in the seminar room upon the successful connection of the participating user to the seminar room.
12. The user with the member type “participant” can quit the room by clicking on the “Quit” button on the seminar room or just closes the seminar room page.
13. A signal will be broadcasted to all members in the seminar room upon the disconnection of any participant in the seminar room and a system chat message for a participant who exited the seminar will be displayed in the chatbox of all other members in the seminar room.
14. The society user with the member type “moderator” will end the seminar by clicking on the “Quit” button on the seminar room or closing the seminar room page.
15. An ending seminar signal will be broadcasted to all members in the seminar room upon the disconnection of the moderator.
16. All participants in the seminar room are forced to quit the seminar and will be redirected back to the seminar entrance.
17. The server destroys the session and the relevant data for the seminar and deletes the seminar’s record in the database.

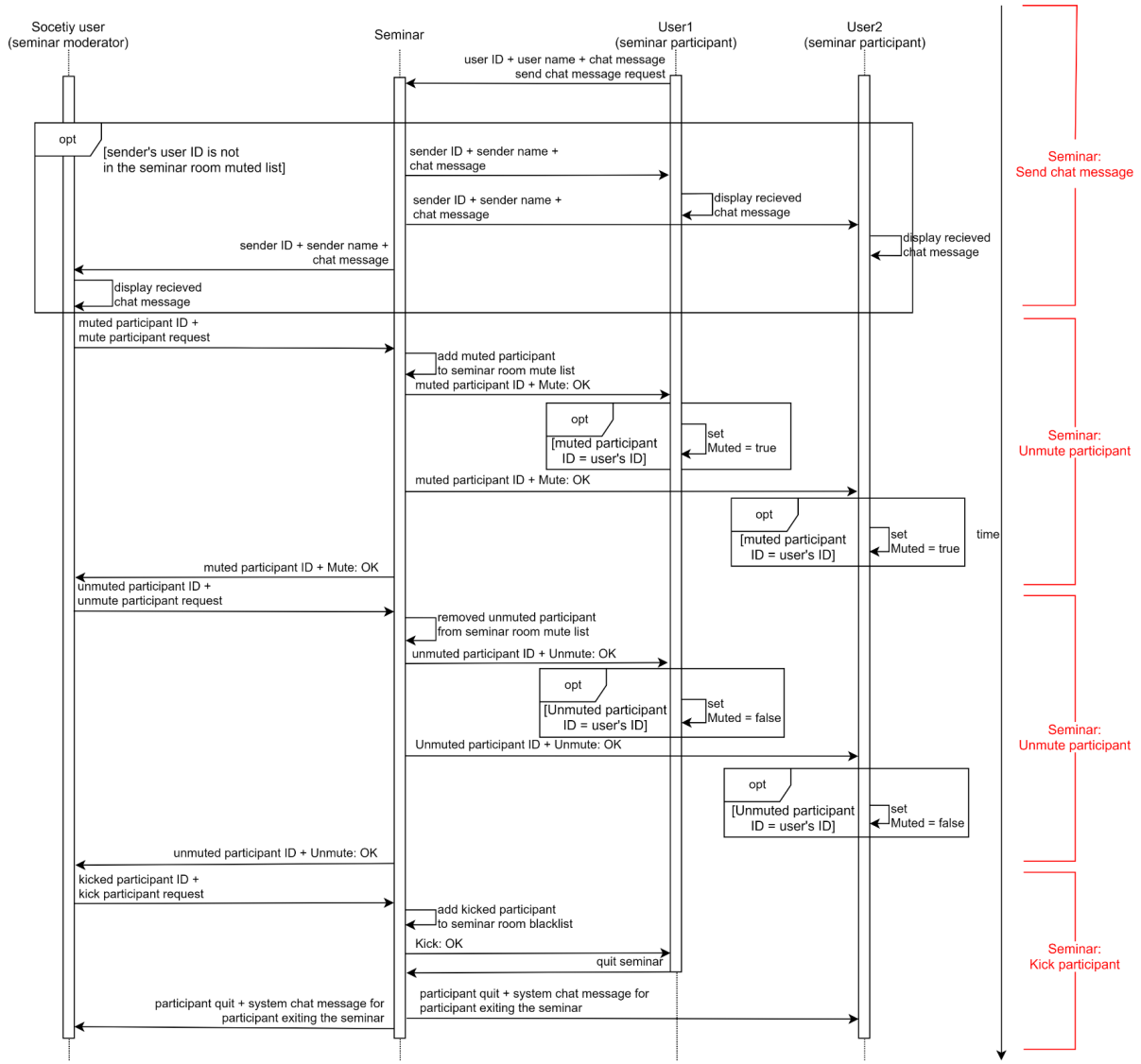

Figure 9. Sequence Diagram for sending a chat message, muting, unmuting and kicking participant in the seminar

Step-By-Step Description:

1. A seminar participant inputs a chat message to the chat box in the seminar room.
2. The seminar participant clicks on the “send” button to activate sending the inputted chat message.
3. The server will check if the sender of the message is in the muted list of the seminar room.
4. If the message sender is not in the muted list, the chat message will be sent to all members in the seminar room and displayed in their chatbox.
5. The seminar moderator may right click on the name of any participants on the seminar room’s member list and click on the “Mute” button on the popover to request for muting the participant.

6. The server adds the muted participant to the seminar room's mute list and emits a signal of muting the participant with the user ID of the muted participant to all members in the seminar room.
7. If the userID of the seminar member equals the user ID of the muted participant stated in the received signal, the status of the seminar member is set to be muted.
8. Upon muting the participant, a muted icon is displayed beside the name of the muted participant on the seminar room's member list and the "Mute" button on the popover is changed to the "Unmute" button.
9. The seminar moderator right-clicks on the name of the unmuted participants on the seminar room's member list and clicks on the "Unmute" button on the popover to request unmuting the participant.
10. The server removes the unmuted participant from the seminar room's mute list and emits a signal of unmuting the participant with the user ID of the unmuted participant to all members in the seminar room.
11. If the userID of the seminar member equals the user ID of the unmuted participant stated in the received signal, the status of the seminar member is set to be unmuted.
12. Upon unmuting the participant, the muted icon beside the name of the muted participant on the seminar room's member list is hidden and the "Unmute" button on the popover is changed to the "Mute" button.
13. The seminar moderator right clicks on the name of the muted participants on the seminar room's member list and then clicks on the "Kick" button on the popover to request kicking the participant.
14. The server adds the kicked participant to the seminar room's blacklist and emits a signal of kicking the participant with his/her user ID to all members in the seminar room.
15. If the userID of the seminar member equals the user ID of the kicked participant stated in the received signal, the seminar member will be forced to quit the seminar and directed back to the seminar entrance.
16. Upon kicking the participant, a signal is broadcasted to all members in the seminar room upon the disconnection of the kicked participant from the seminar room and a system chat message for the participant exiting the seminar is displayed in the chatbox of all members in the seminar room.

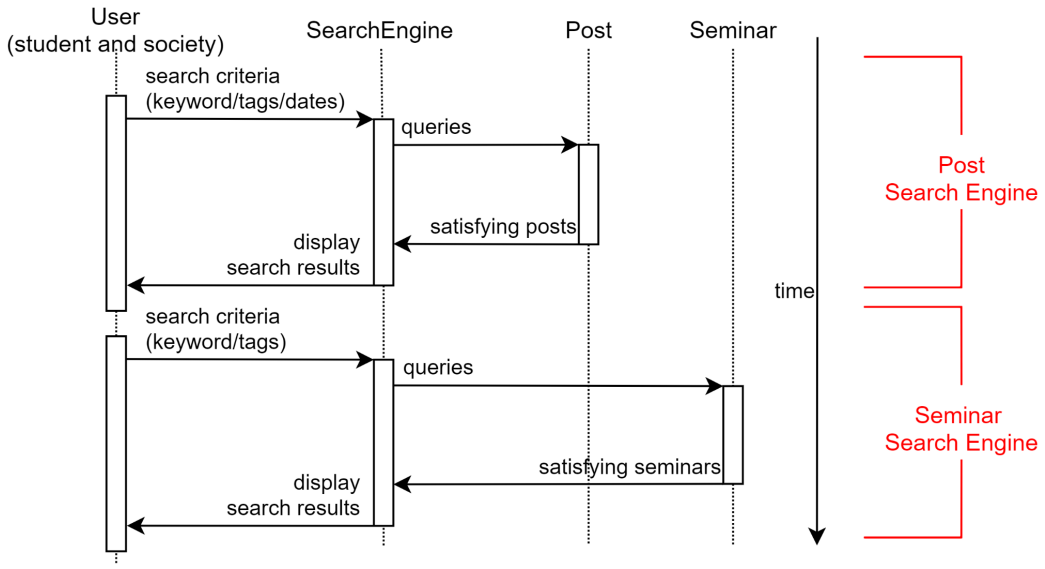

Figure 10. Sequence Diagram for the search engine

Step-By-Step Description:

1. The user may enter a word to the input box in the search panel on the post browser page, and selects event tags, and the range of the event dates on the advanced search panel.
2. The user clicks on the “search” button to activate a post search for the query.
3. The server retrieves posts which match the query conditions from the database.
4. The retrieved post list is returned back to the user and displayed on the post browser page.
5. The user enters the keyword to the input box in the search panel on the seminar entrance page, and selects event tags on the advanced search panel.
6. The user clicks on the “search” button to activate a seminar search for the query.
7. The server retrieves seminars which match the query conditions from the database.
8. The retrieved seminar list is returned back to the user and displayed on the seminar entrance page.

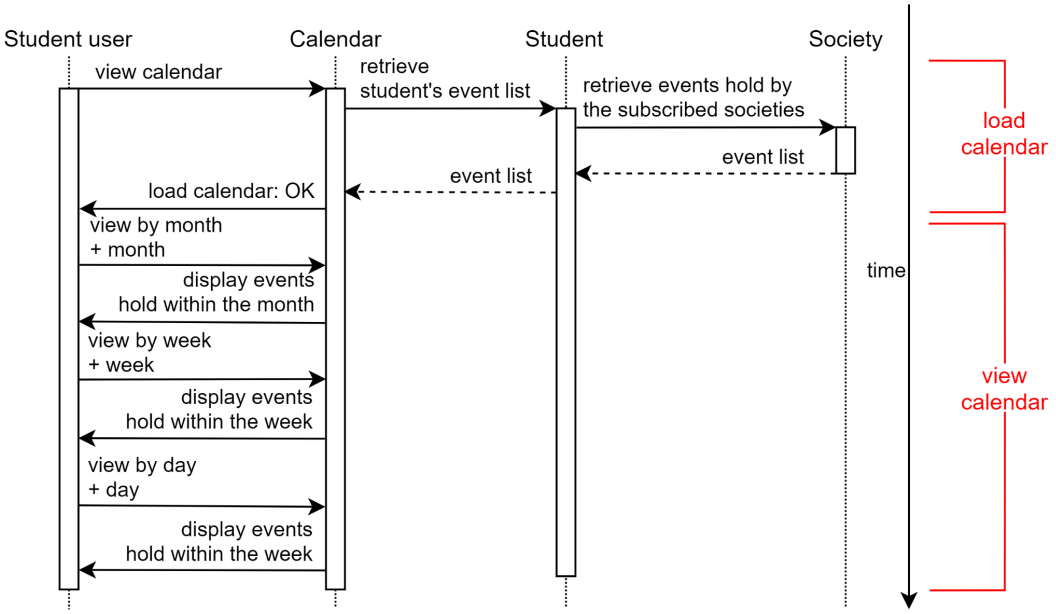

Figure 11. Sequence Diagram for viewing student calendar

Step-By-Step Description:

1. The student user clicks on the “View My Calendar” button to route to the calendar page.
2. Upon entering the calendar page, the server retrieves the events of the subscribed societies of the student user from the databases.
3. The retrieved event list is returned back and displayed on the calendar.
4. The student user clicks on the “month” button on the calendar to view the calendar by month.
5. The student user clicks on the “week” button on the calendar to view the calendar by week.
6. The student user clicks on the “day” button on the calendar to view the calendar by day.

4 USER INTERFACE DESIGN

4.1 Welcome Page

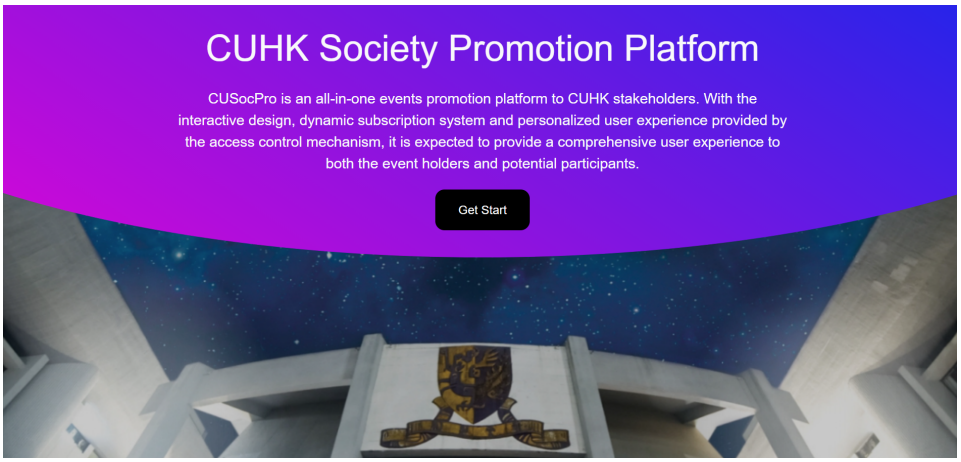

Figure 12. Welcome Page

When the user firstly arrives at the Welcome Page, a warm greeting and purpose of the system will be shown.

4.2 Login and Sign-up Page

4.2.1 Login Page

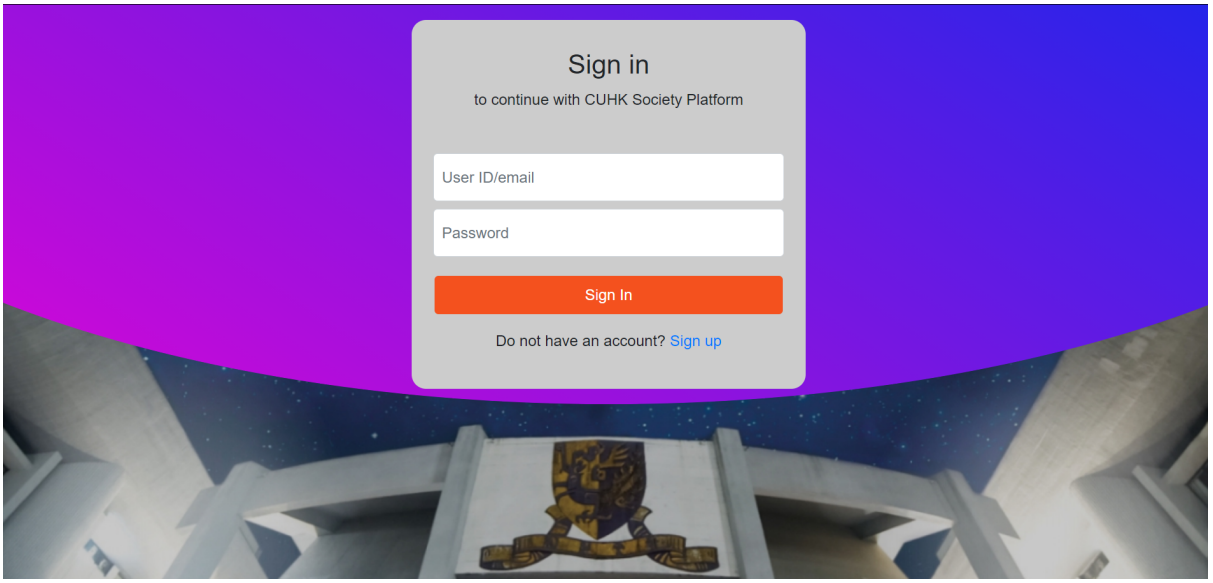

Figure 13. Login Page

Upon clicking the “Get Started” button on the Welcome page, users will be directed to this Login page. Students or society users can login into their CUSocPro account by their user ID and password. As long as the server has verified their account credentials, users will be redirected to the correspondent home page regarding their account type.

For the users who access CUSocPro the first time, they could click on the “Sign up” link underneath the sign-in box and get redirected to the account registration page.

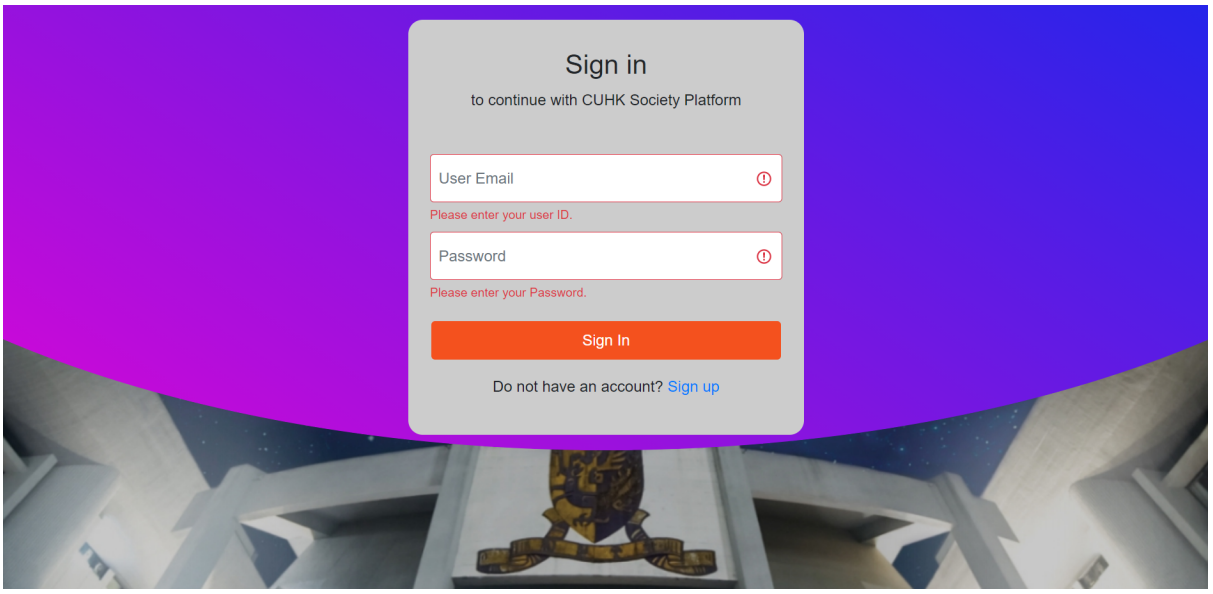

Figure 14. Invalid login

If any of the input boxes are null, the username or that the password is not match with the record in the server, a warning message will be returned below the input boxes.

4.2.2 Sign-up Page

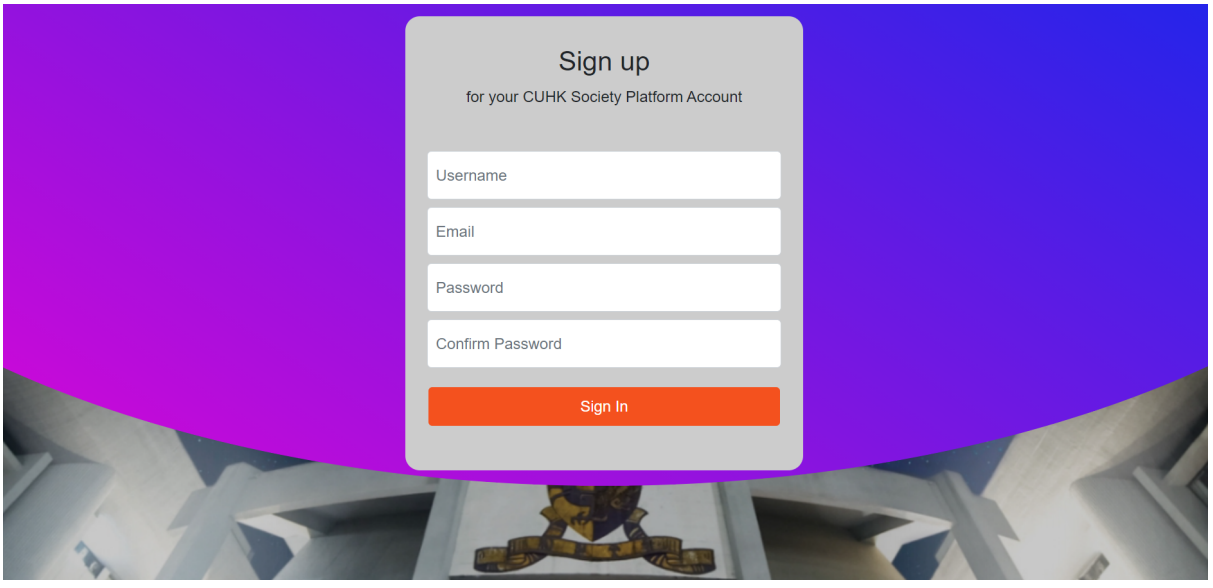

Figure 15. Sign-up Page

On the sign-up page, students can create an account by setting up their username, CUHK email address; as well as a password which must be at least 8 characters long.

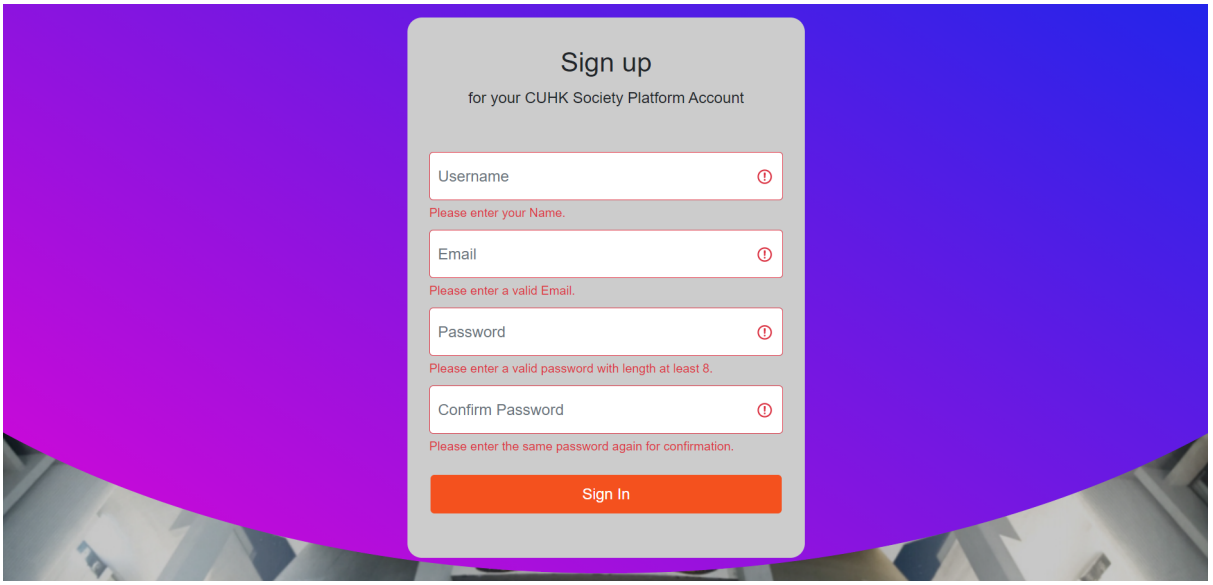

Figure 16. Invalid sign-up

If a CUHK email address is not given, or the any of the requirement is not fulfilled, the registration will not be accepted and a warning messages will be returned below the input boxes.

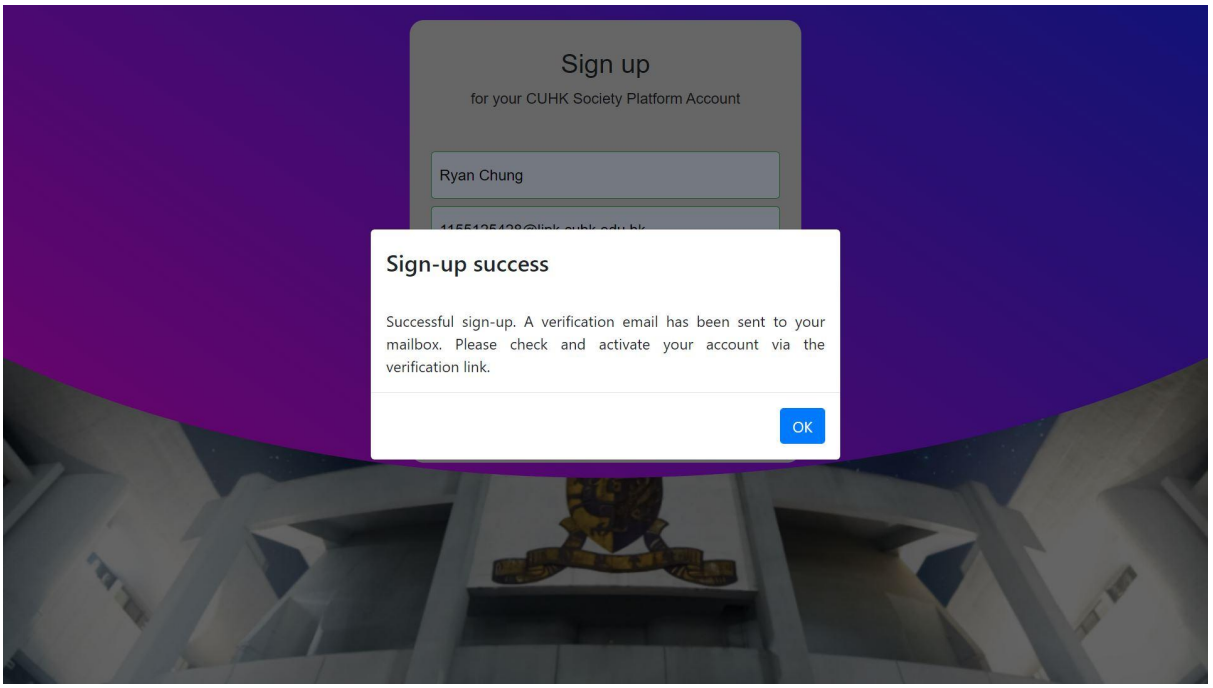

Figure 17. Valid sign-up (with notification)

If the registration is successful, a notification box will be shown on screen and requires the student to verify their CUHK email to confirm their registrations.

4.2.3 Account Activation Page

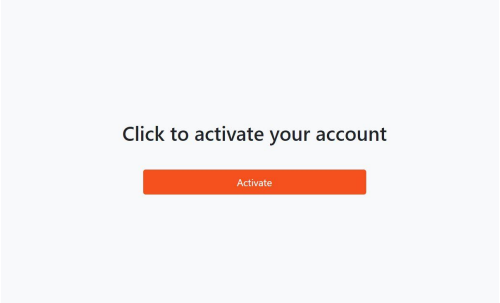

Figure 18. Account Activation Page

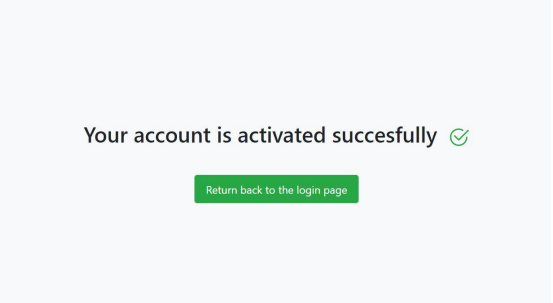

Figure 19. Account activation successful

When the student clicked the link “Activate your account” in their received mail, they will be redirected to this Account Activation page. Students are required to confirm the activation of the account by clicking the “Activate” button.

A success message will be returned once the student clicks the “Activate” button, and students may return to the Login page if they click the “Return back to the login page” button.

4.3 Home Page

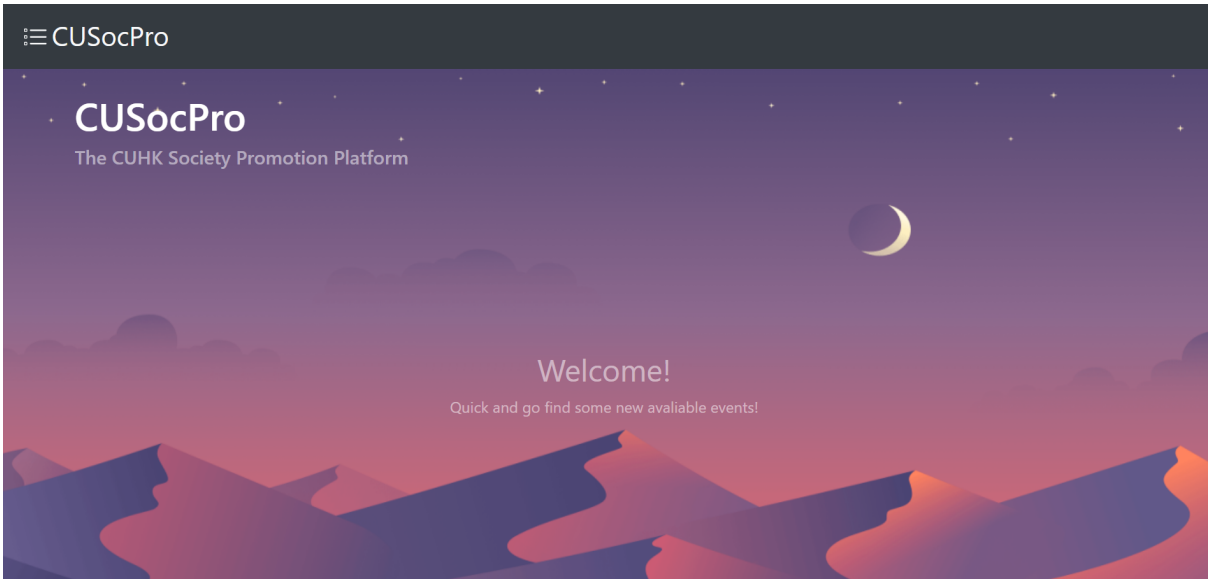

Figure 20. Home Page

In the home page, it contains welcome messages and a slider button for the menu on the top left corner.

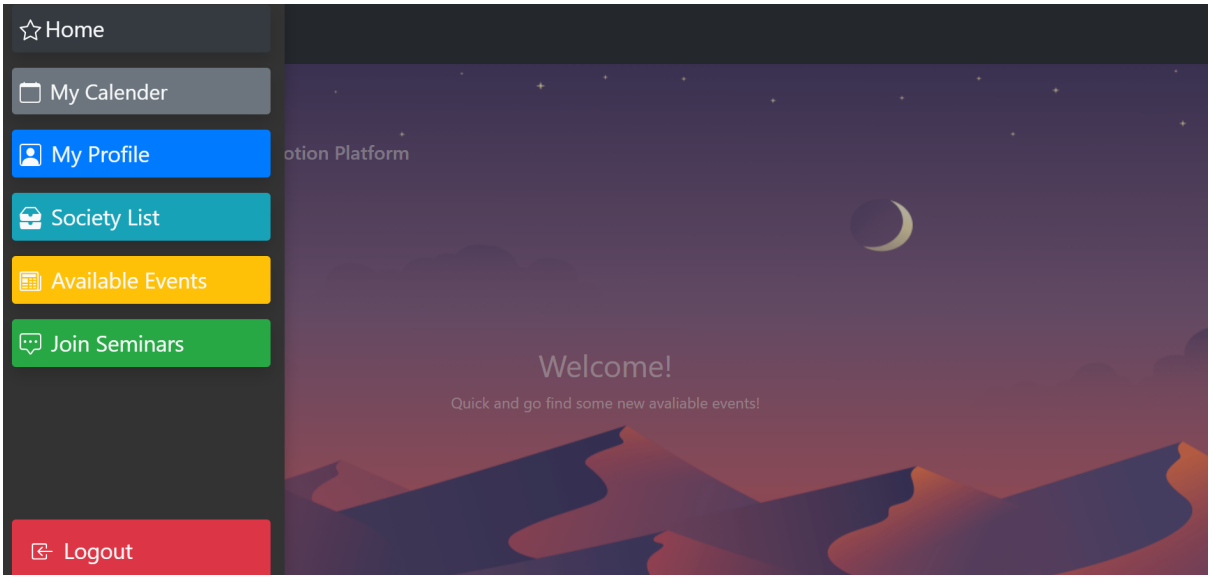

Figure 21. Navigation Menu (Student)

Upon clicking the Menu button, a slider interface will be shown on the left. It contains various buttons that links to different pages, which includes “My Calendar”, “My Profile”, “Society List”, “Available Events”, “Join Seminars”, and at last a “Logout” function that will activitate a logout function.

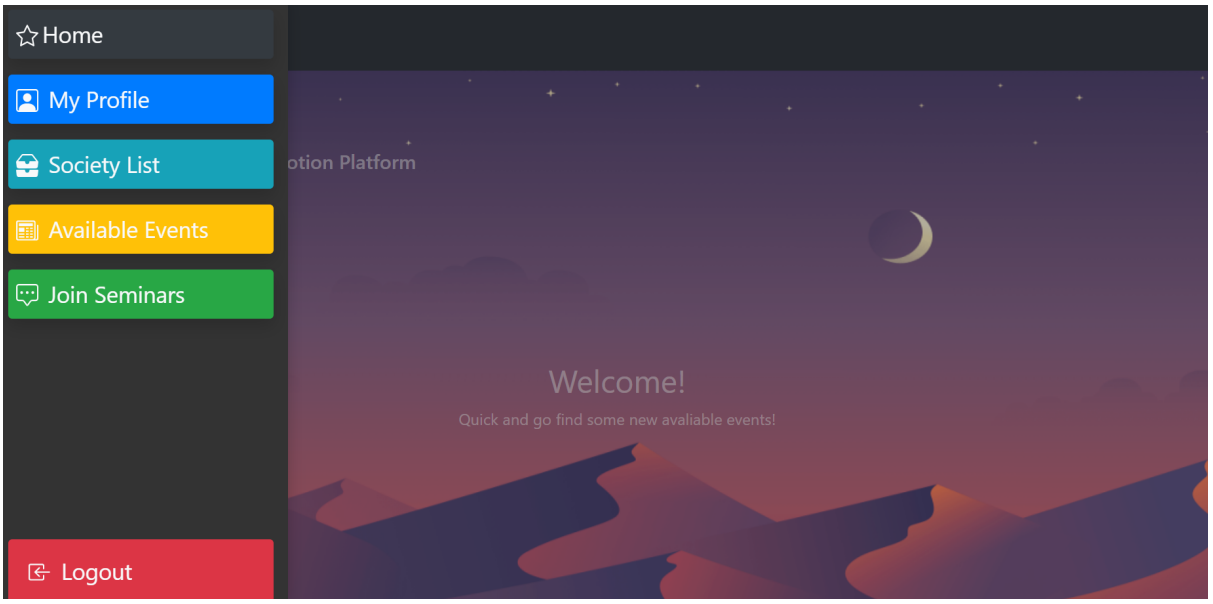

Figure 22. Navigation Menu (Society)

Similarly, society users also have a Navigation Menu that is similar to the student's point of view while the “My Calendar” function has been disabled.

4.4 Calendar Page

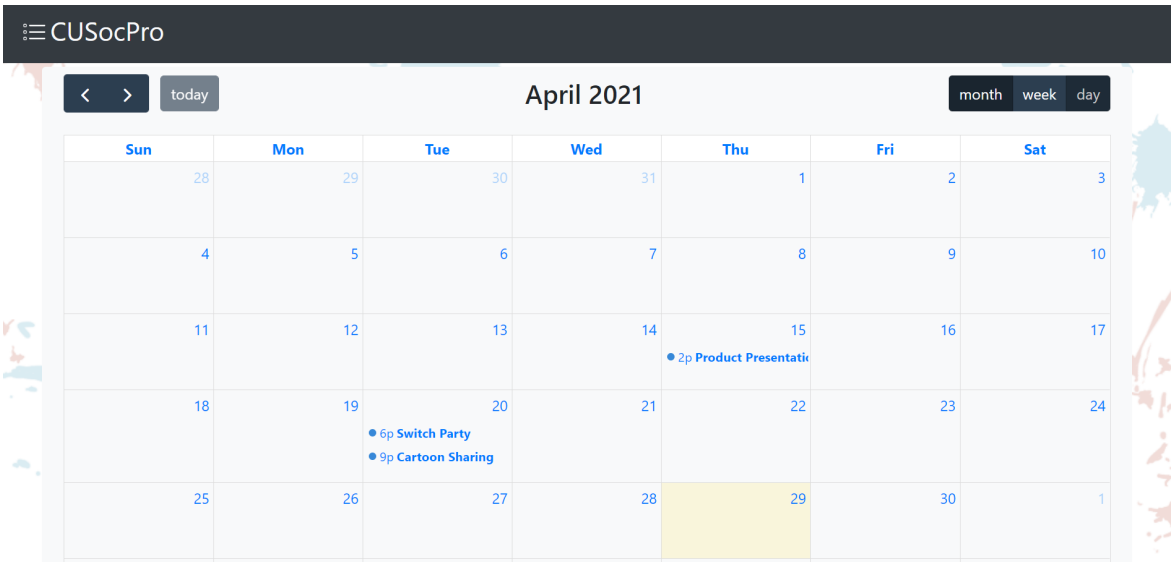

Figure 23. Calendar Page (in month)

In the Calendar Page, the date of this month, the current date, as well as the subscribed society events' date and time will be shown. Students can change the data frame to previous or next month with the left and right buttons placed on the top left corner. Students can also toggle the Calendar in terms of month, week or day from the boxes on the top right corner, or jump back to the current day via the today button.

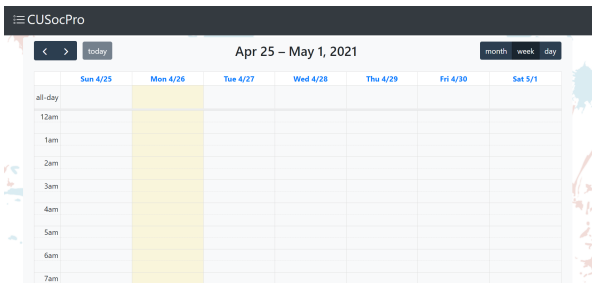

Figure 24. Calendar Page (in week)

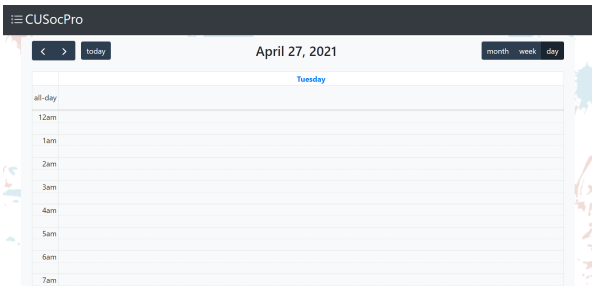

Figure 25. Calendar Page (in day)

4.5 Profile Page

4.5.1 Student Page

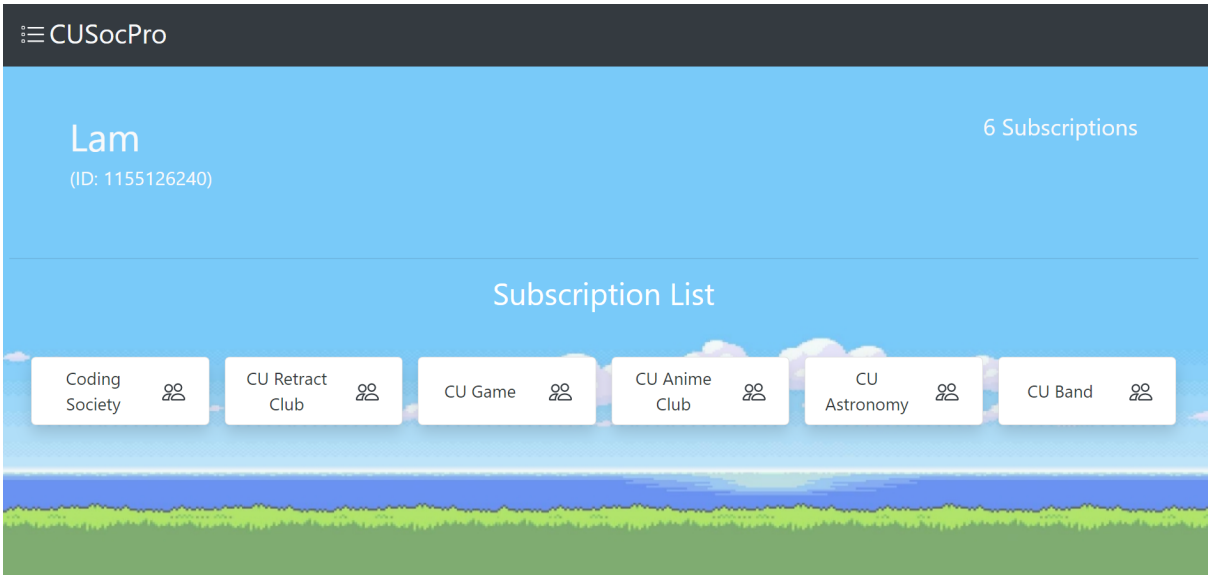

Figure 26. Student Profile Page

In the Profile page from the student's point of view, the username and its ID number on the top left corner, the number of societies the student had subscribed to on the top right corner, as well as the subscription list that listed all the names of the subscribed societies will be shown. Students can click on the society card and redirect to its society profile page for more information.

4.5.2 Society Page

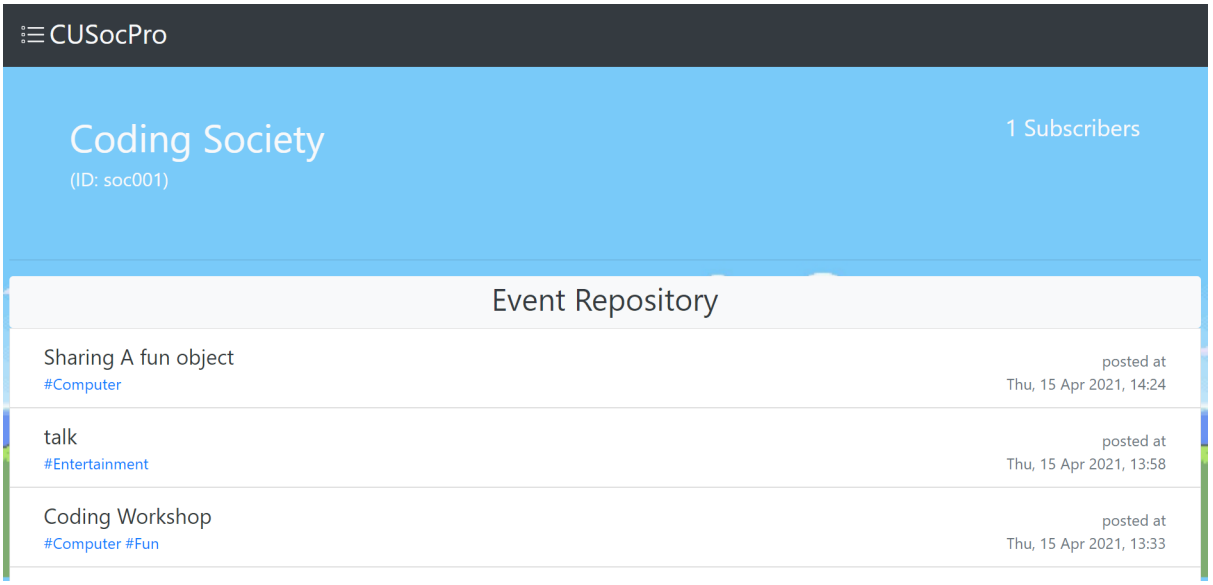

Figure 27. Society Profile Page

In the society profile page, society name and its ID number will be shown on the top left corner, the number subscribe will be shown on the top right corner. Event repository that

listed all the previous and upcoming events of this society will be shown and users can click the event name for more information.

4.6 Society List Page

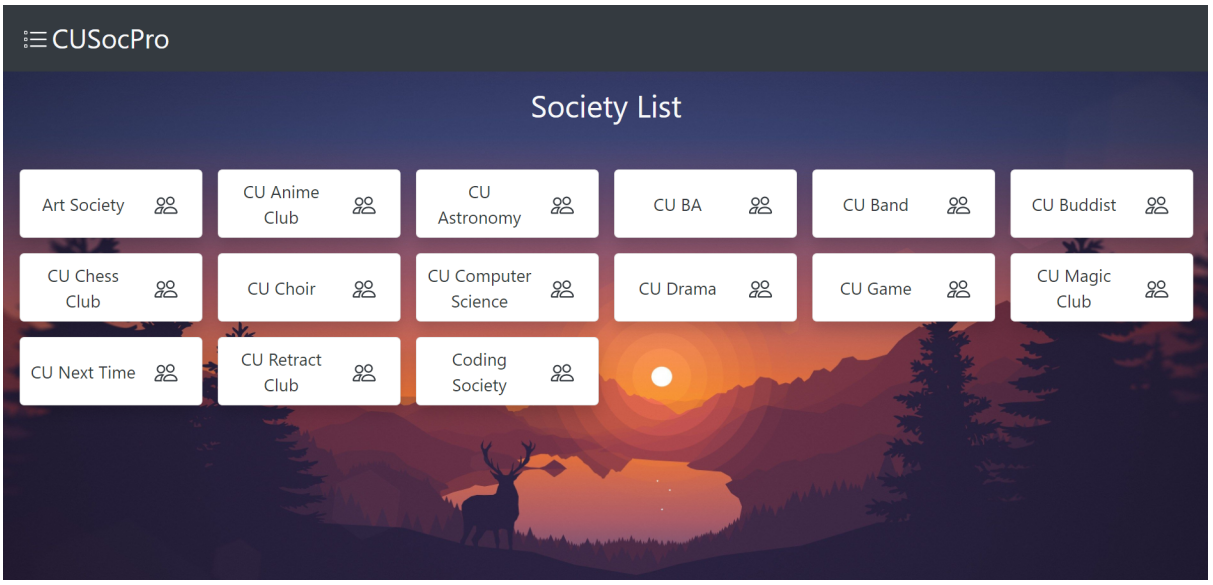

Figure 28. Society List Page

In the Society List Page (available to all users), the names of all the registered societies queried from the database server will be displayed. The student users can browse the respective society's profile page by clicking on the widget of the corresponding society.

4.7 Activity List Page

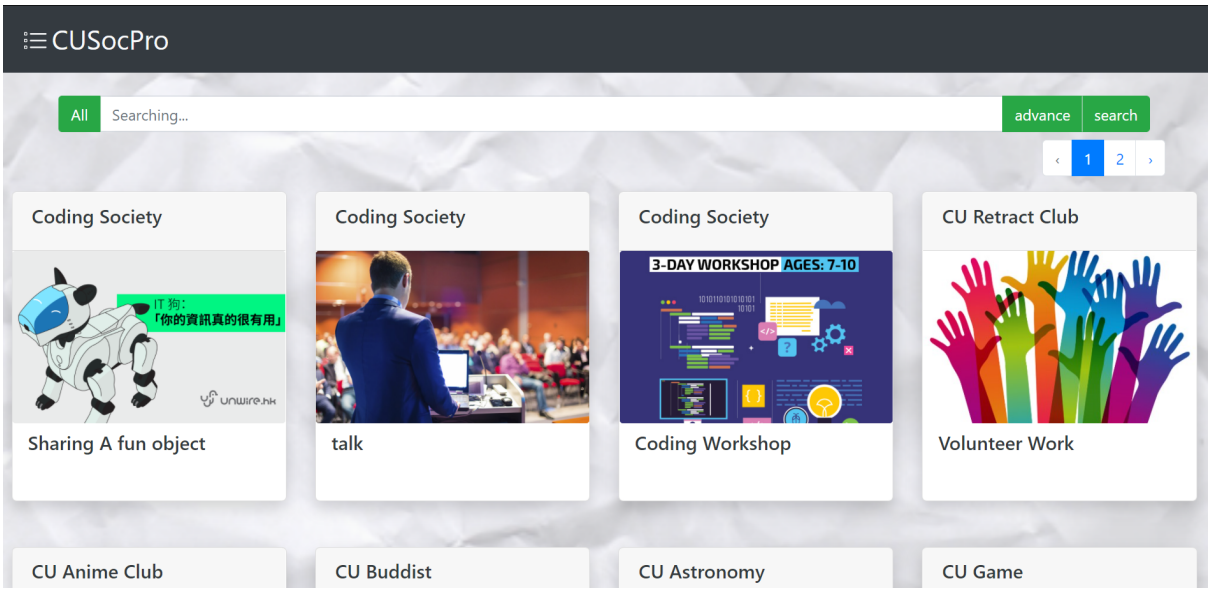

Figure 29. Activity List Page

In the Activity List Page, it shows all the events created by different societies. The student users can browse the event content page by clicking on the event widget.

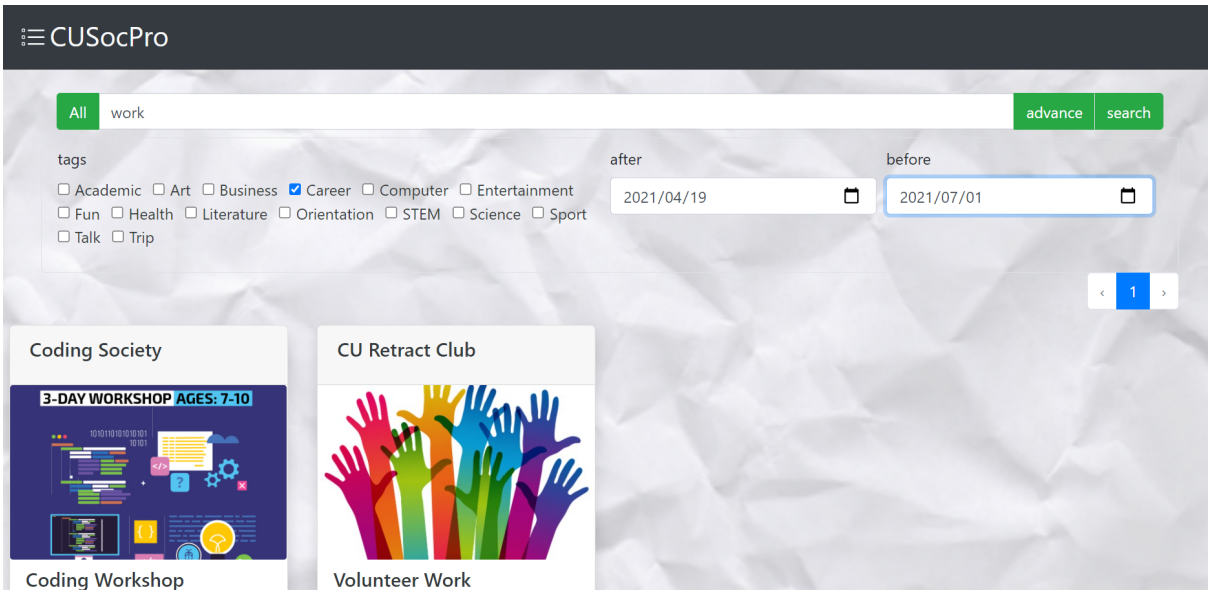

Figure 30. Activity List Page (Searching function)

A searching function is also available for students to find specific events by entering the keywords, or using advanced search functions for more detailed search.

4.7.1 Event Content Page

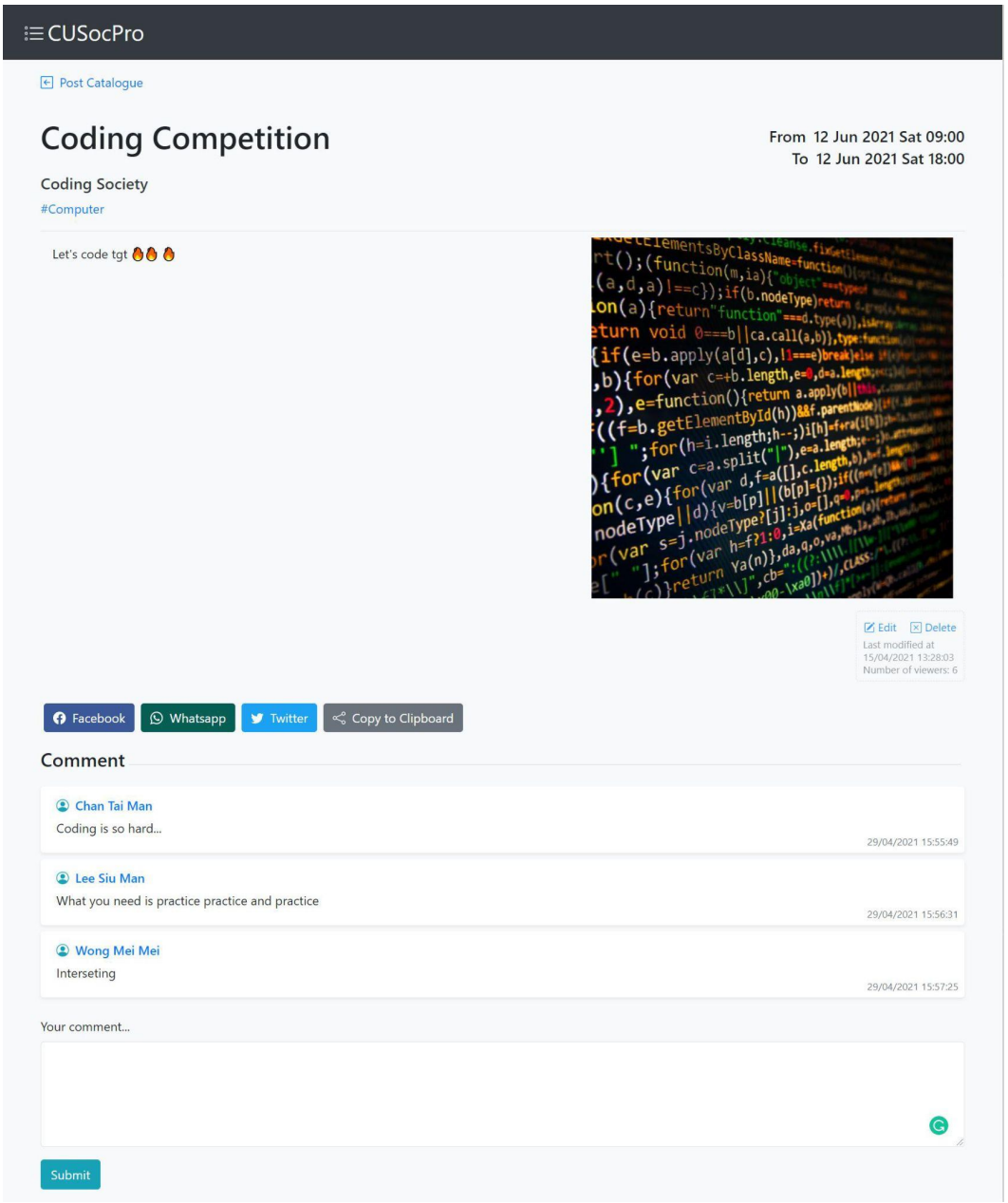

Figure 31. Event Content Page

In the Event Content Page, event content like event title, event-holding society, description, poster images, the date of the event, tags, and the comment section will be shown. All users can leave comments on the post and share the event via Facebook, Whatsapp, Twitter or by copying from clipboard. The edit and delete post functions are only available to the society account who created the post. Nevertheless, the time from the last modification, as well as the number of viewers will also be shown in the event details.

4.7.2 Create Event Page

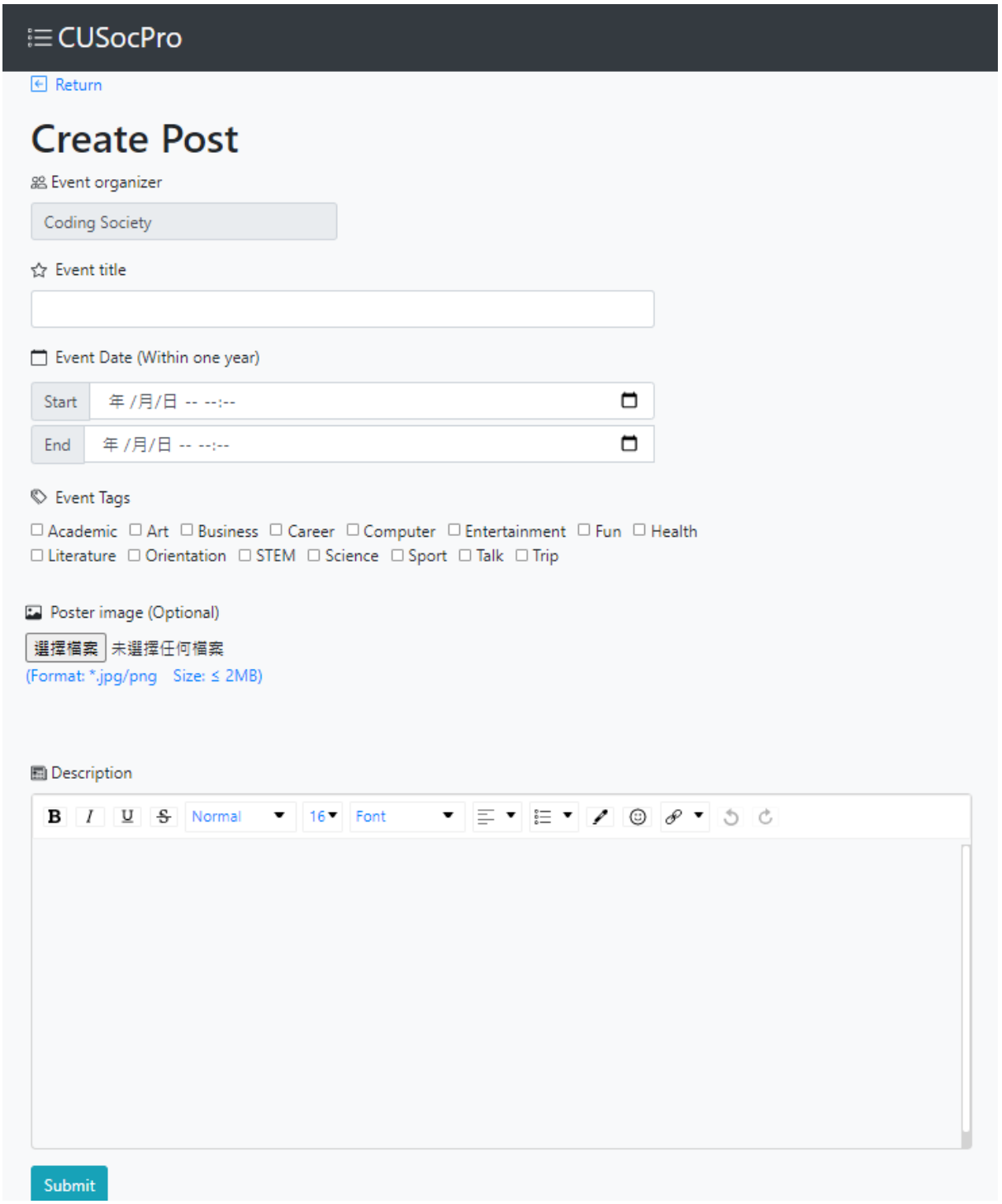

Figure 32. Create Post Page

In the Create Post page, society can post details about their upcoming event including event title, event date, description and poster through this page (poster image has to be less than 2MB), and name tags for searching events in the Activity List page. Students who have subscribed to the society would automatically receive an email notification about this event.

4.7.3 Edit Event Page

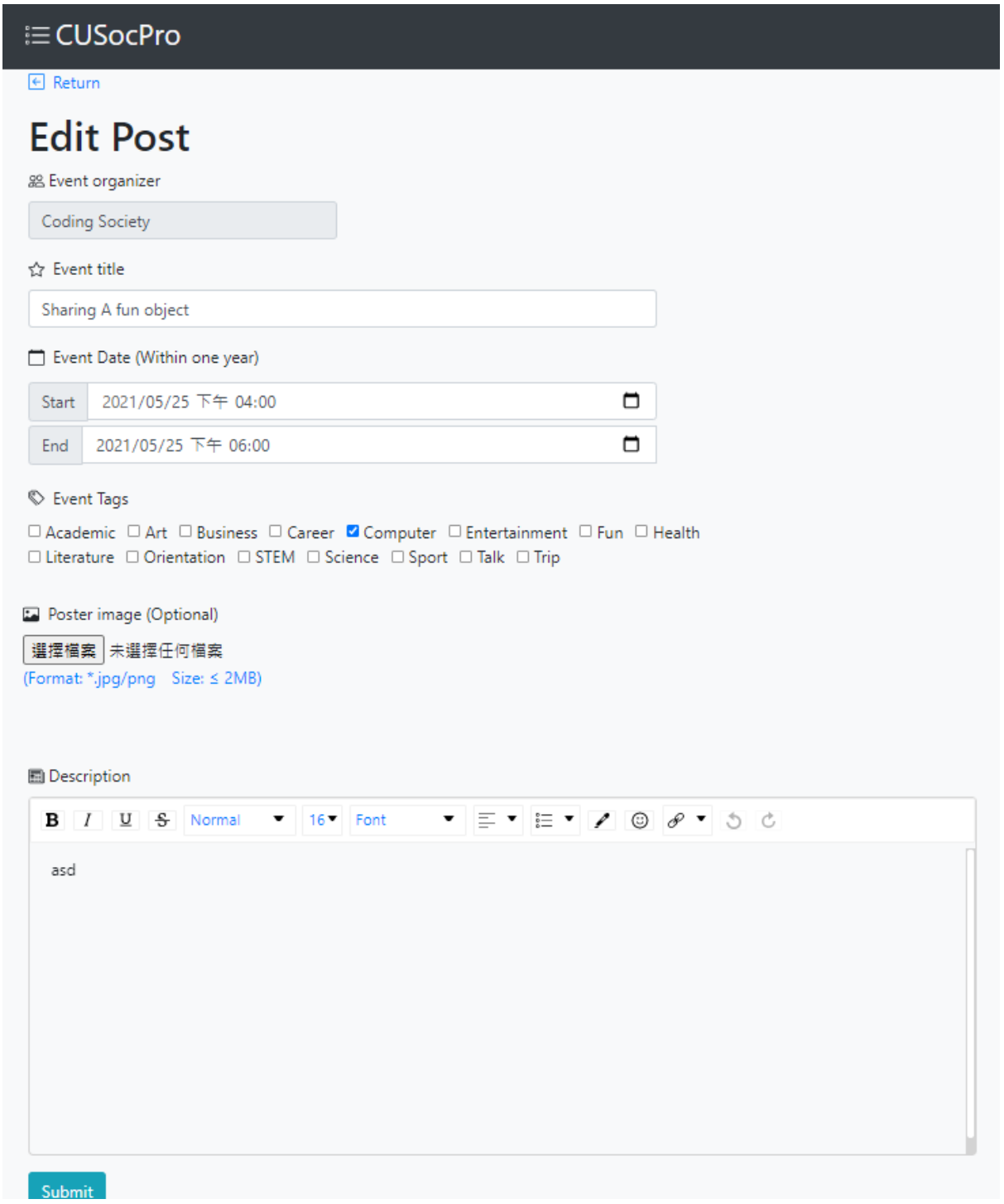

Figure 33. Edit Post Page

In the Edit Post page, the society can update a post by modifying the details of the event content. After the edit is done, an email notification will be sent to the subscribed students to notify the changes. The latest modification date will also be updated on the event post.

4.8 Seminar Page

4.8.1 Seminar Content Page

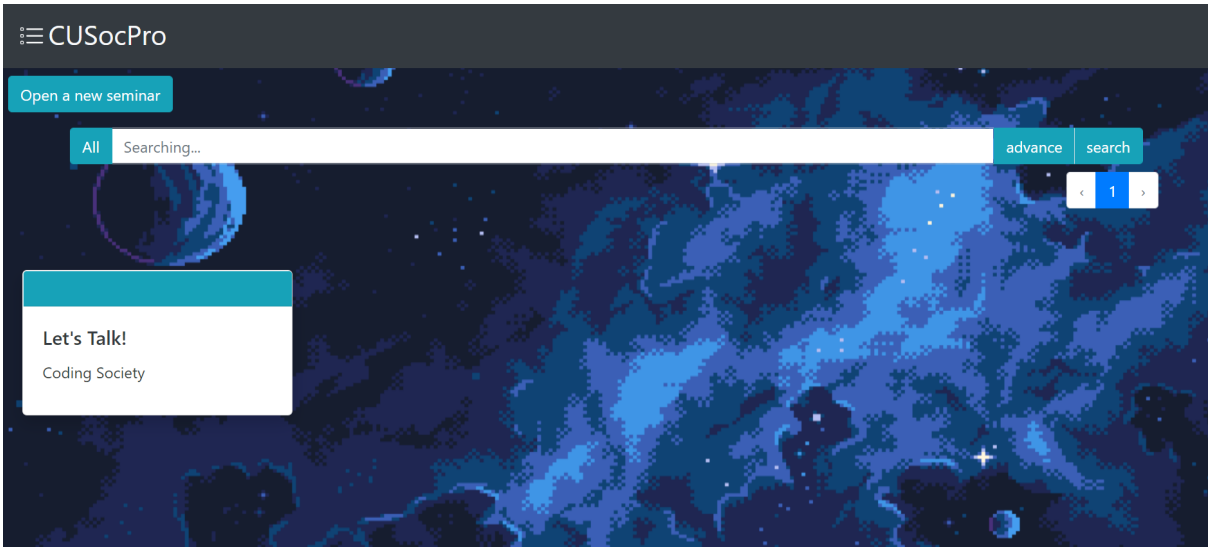

Figure 34. Seminar Content Page

In the Seminar Content Page, users can browse all the available seminars that are currently opened by societies. Moreover, the users can use the search function to find the desired seminar in the search engine via keywords filters or tags.

4.8.2 Create Seminar Page

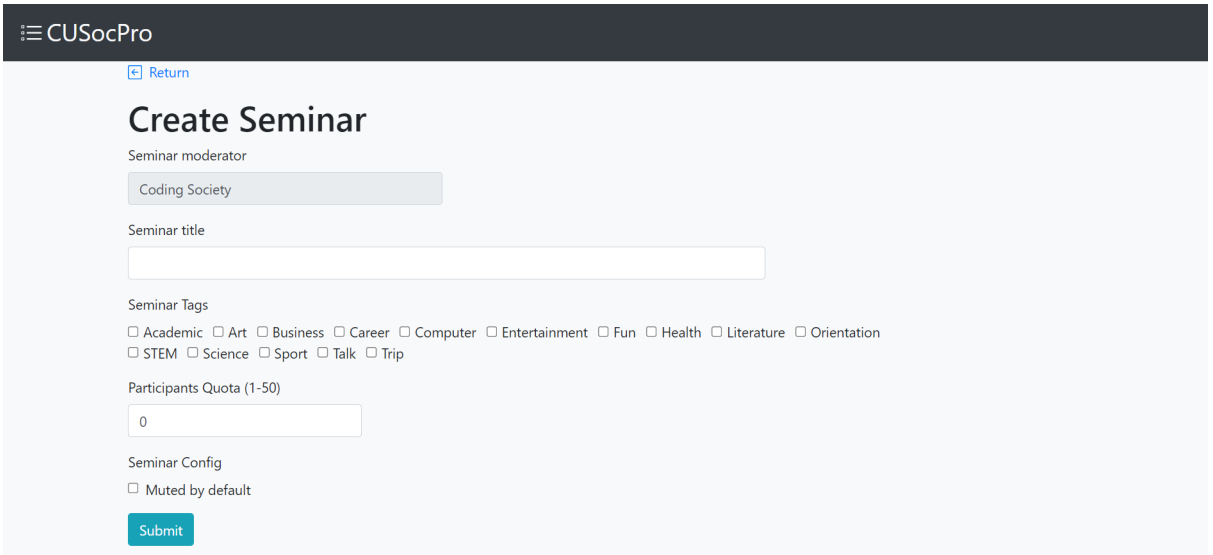

Figure 35. Create Seminar Page

In the Create Seminar Page, society users can create a seminar with the title and tags that help the students to search for the desired content. Furthermore, the setting of the seminar, number of quotas and other configurations, can be customized.

4.8.3 Seminar Room Page

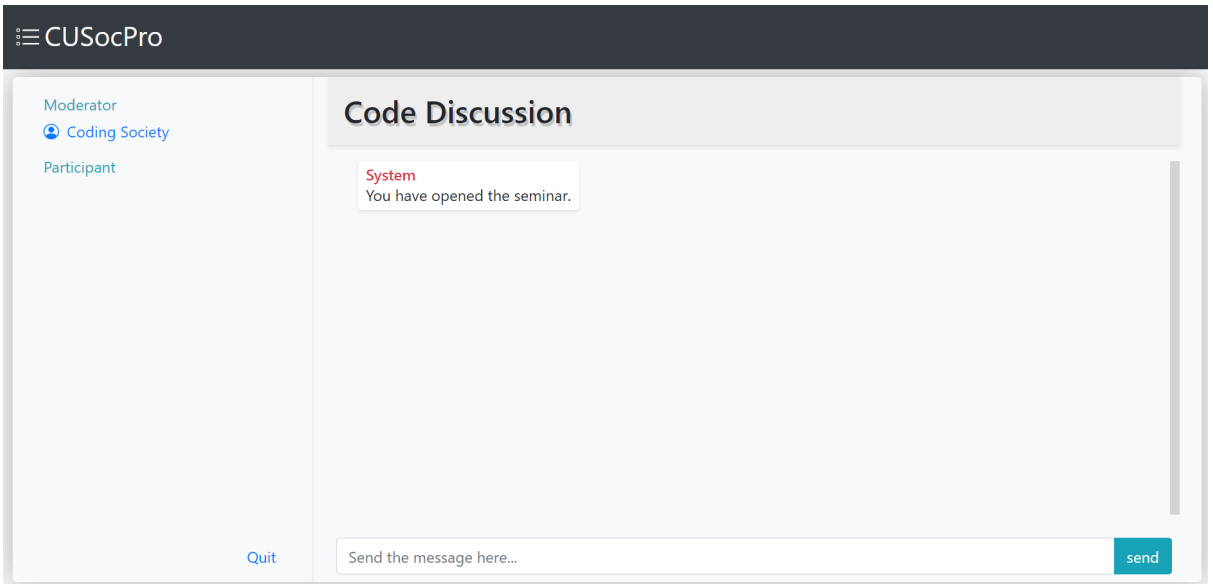

Figure 36. Seminar Room Page

In the Seminar Room Page, it allows real-time communication between both the participants (students) and the host (society).

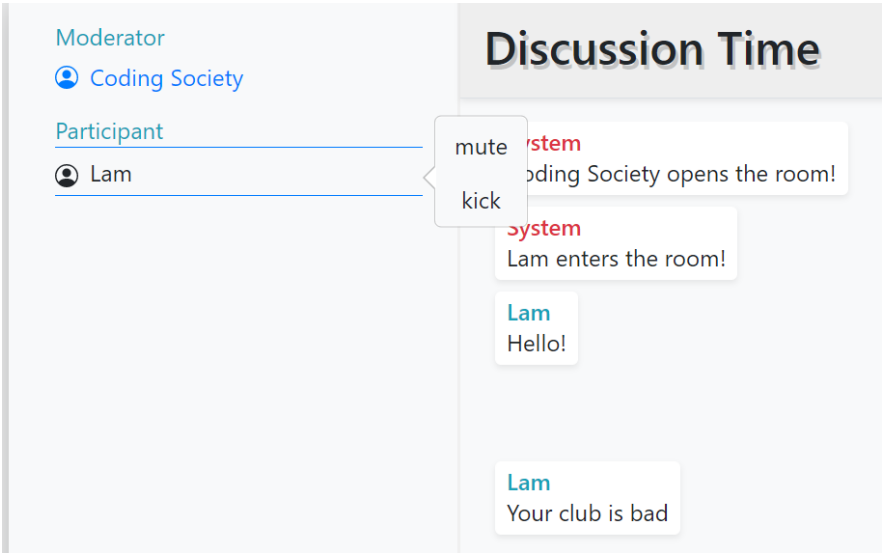

Figure 37. Seminar Room Page (Moderator functions)

The host (i.e. moderators) can monitor and control the seminar room by muting or kicking participants who have inappropriate behaviours.

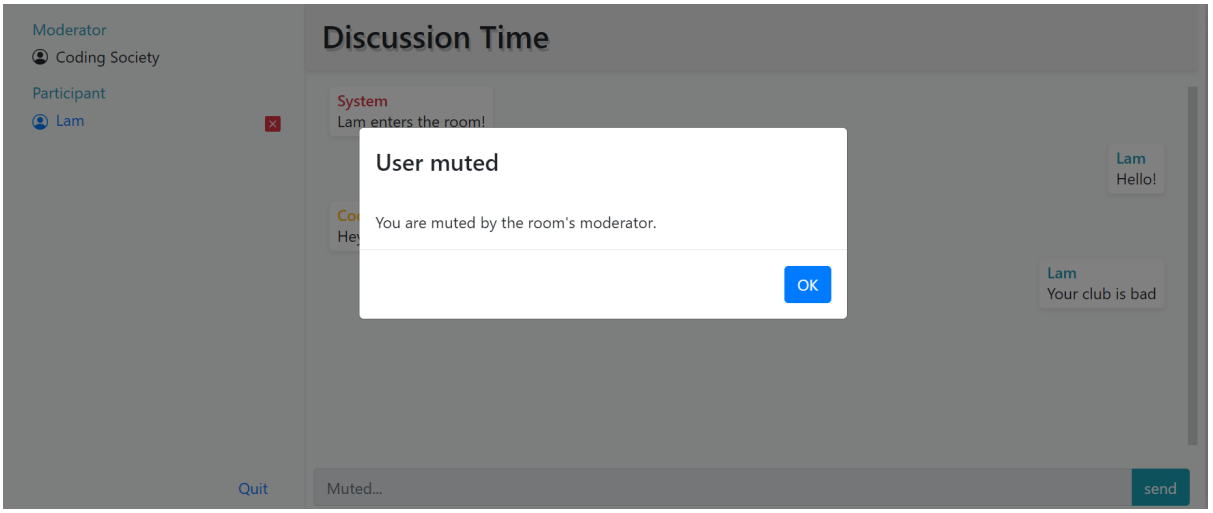

Figure 38. Seminar Room Page (Student get muted)

Situation when a student who has been temporarily muted and could not type in any messages until he got unmuted.

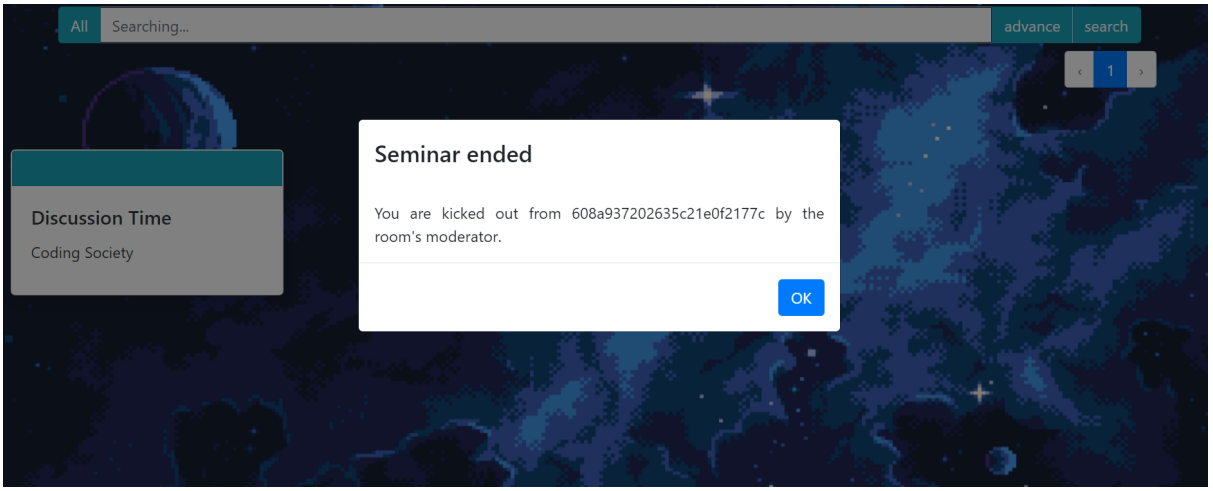

Figure 39. Seminar Room Page (Student get kicked)

Situation when a student was kicked from the seminar room and could not join the room again.

5 TEST

5.1 SIGNUP

5.1.1 Purpose

To test the sign-up function and the verification performed according to the following conditions:

- The inputted username, email, password and the confirm password must be non-empty
- The inputted email must be unused by the existing user
- The inputted email must be ended with @link.cuhk.edu.hk and have the prefix constructed by 10 digits
- The length of the inputted password must be at least 8 and at most 20
- The inputted confirm password must be equal to the inputted password

5.1.2 Inputs and expected outputs

Input	Expected output	Remarks
userName: “” email: “” password: “” confirmPassword: “”	Error messages shown under the input boxes for the user name, email, password and confirm password	The inputs are empty
userName: “Ryan” email: “testing@example.com” password: “testpassword” confirmPassword: “testpassword”	Error message shown under the input boxes for the email	Valid username, password and confirm password. The email address must be a CUHK @link email address, which ends with @link.cuhk.edu.hk
userName: “Ryan” email: “115512542@link.cuhk.edu.hk” password: “testpassword” confirmPassword: “testpassword”	Error message shown under the input boxes for the email	Valid username, password and confirm password. The email prefix must be constructed by 10 digits because it is expected to be a student ID
userName: “Iknos” email: “1155124886@link.cuhk.edu.hk” password: “passwordtest” confirmPassword: “passwordtest”	Message pops up for the failed sign-up because the email address has already been used	Valid username, password and confirm password. The email address must be unused so that each student can only apply for an account with their own CUHK email

		address
userName: “Ryan” email: “1155125428@link.cuhk.edu.hk” password: “testpw” confirmPassword: “testpw”	Error message shown under the input boxes for the password	Valid username, email address and confirm password. The minimum length of the password is 8
userName: “Ryan” email: “1155125428@link.cuhk.edu.hk” password: “testtesttesttesttesttest” confirmPassword: “testpw”	Error message shown under the input boxes for the password	Valid username, email address and confirm password. The maximum length of the password is 20
userName: “Ryan” email: “1155125428@link.cuhk.edu.hk” password: “testpassword” confirmPassword: “anotherpassword”	Error message shown under the input boxes for the confirm password	Valid username, email address and password. The confirmed password must be equal to the inputted password
userName: “Ryan” email: “1155125428@link.cuhk.edu.hk” password: “testpassword” confirmPassword: “testpassword”	Message pops up for the successful sign-up	Valid account information

5.2 ACCOUNT ACTIVATION

5.2.1 Purpose

To test the account activation function, there are several conditions to be considered according to the following:

- The json web token parsed from the URL must be valid
- The json web token must be unexpired when activating the account

5.2.2 Inputs and expected outputs

Input	Expected output	Remarks
activation token: (Invalid) abcdefghijklmnop.ItIsAnInvalidJsonWebToken...	The alert pops up for the unsuccessful account activation	The activation token is invalid
activate token: (Expired) eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InJ5YW5jaHVuZyI...	uThe alert pops up for the unsuccessful account activation	The activation token is expired
activate token: (valid) yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6IlJ5YW4gQ2h1bmci...	Message displayed for the successful account activation	The valid and unexpired activation token

5.3 LOGIN

5.3.1 Purpose

To test the login function according to the following conditions:

- The inputted userID/email and password must be non-empty
- There must be an existing user account whose user ID equals the inputted user ID or email address equals the inputted email address
- The inputted password must be matched with the password of the existing user account whose user ID equals the inputted user ID or email address equals the inputted email address

5.3.2 Inputs and expected outputs

For testing, the account information of a valid existing account is as follows:

- user ID = “Ryan”
- email: “1155125428@link.cuhk.edu.hk”
- password = “testpassword”

Input	Expected output	Remarks
userID/email: “” password: “”	Error messages shown under the input boxes for the user name, email, password and confirm password	The inputs are empty
userID/email: “Ryan” password: “wrongfortest”	Message pops up for the unsuccessful login because the account information is invalid	Valid useID/email. The inputted password doesn’t match the password of the corresponding user with the inputted user ID or email address
userID/email: “wrongfortest” password: “testpassword”	Message pops up for the unsuccessful login because the account information is invalid	Valid password.. The user with the inputted user ID or email address is not found
userID/email: “Ryan” password: “testpassword”	The browser is directed to the location, /Home, and a cookie with the name “authToken”, which stores a json web token, is set.	Valid account information (valid user ID and password)
userID/email: “1155125428@link.cuhk.edu.hk” password: “testpassword”	The browser is directed to the location, /Home, and a cookie with the name	Valid account information (valid email address and password)

	“authToken”, which stores a json web token, is set.	

5.4 LOGOUT

5.4.1 Purpose

To test the logout function.

5.4.2 Inputs and expected outputs

Input	Expected output	Remarks

cookie: {
authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
}
Click on the “Logout” button on the navigation menu
	The cookie with the name “authToken” in the currently used browser is deleted and the browser is directed to the login page at the location /login.	Successful logout from an active user account

5.5 CALENDAR

5.5.1 Purpose

To test the calendar function by viewing in month/week/day view

5.5.2 Inputs and expected outputs

Input	Expected output	Remarks

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
	The browser is directed back to the Home page at the location ./Home	Society user is disabled from using the calendar function

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
	The calendar displays the events of the subscribed societies of the student user (in the month view by default)	Valid user

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
viewOption: by month
	The calendar displays the events of the subscribed societies of the student user in the month view	Valid user and viewing calendar by month

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
viewOption: by week
	The calendar displays the events of the subscribed societies of the student user in the week view	Valid user and viewing calendar by week

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
viewOption: by day
	The calendar displays the events of the subscribed societies of the student user in the day view	Valid user and viewing calendar by day

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
	The calendar displays the events, which is held in the current month, in the month view	Valid user and viewing the calendar for the current month

viewOption: by month Click on the “today” button		

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
viewOption: by month
Click on the “＞” button
	The calendar displays the events, which is held in the next month from the currently viewed month, in the month view	Valid user, flipping the calendar towards the future and viewing by month

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
viewOption: by month
Click on the “＜” button
	The calendar displays the events, which is held in the previous month from the currently viewed month, in the month view	Valid user, flipping the calendar towards the past and viewing by month

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
viewOption: by week
Click on the “today” button
	The calendar displays the events, which is held in the current week, in the week view	Valid user and viewing the calendar for the current week

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
viewOption: by week
Click on the “＞” button
	The calendar displays the events, which is held in the next week from the currently viewed week, in the week view	Valid user, flipping the calendar towards the future and viewing by week

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
viewOption: by week
Click on the “＜” button
	The calendar displays the events, which is held in the previous week from the currently viewed week, in the week view	Valid user, flipping the calendar towards the past and viewing by week

authToken: { userID: “1155125428” userName: “Ryan”	The calendar displays the events, which is held today, in the day view	Valid user and viewing the calendar for today

userType: “student” } viewOption: by day Click on the “today” button		

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
viewOption: by day
Click on the “＞” button
	The calendar displays the events, which is held in the next day from the currently viewed day, in the day view	Valid user, flipping the calendar towards the future and viewing by day

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
viewOption: by day
Click on the “＜” button
	The calendar displays the events, which is held in the previous day from the currently viewed day, in the day view	Valid user, flipping the calendar towards the past and viewing by day

5.6 SOCIETY LIST

5.6.1 Purpose

To test the navigation to the society profile page in the society list

5.6.2 Inputs and expected outputs

Input	Expected output	Remarks
Click on the society widget corresponding to the society with societyName: “Coding Society” societyID: “soc001”	The browser is directed to the society profile of the Coding Society at the location, /SocietyProfile/soc001	Route to the society profile upon clicking the society widget

5.7 CREATE POST

5.7.1 Purpose

To test the post creation function and the verification performed, there are several conditions to be considered according to the following:

- The post creation function is only authorized to the society user
- The inputted event title, event starting datetime and event ending datetime must be non-empty
- The length of the event title is at most 50
- The event starting datetime and the event ending datetime must be equal to or later than the datetime of the post creation form submission
- The event starting datetime and the event ending datetime must be within the following one year from the datetime of the post creation form submission
- The event starting datetime must be equal to or before the event ending datetime, and the event ending datetime must be equal to or after the event starting datetime
- The selected file of the event poster must have a size at most 2MB = 2097152 byte
- The format of the selected file of the event poster must be either JPEG or PNG

5.7.2 Inputs and expected outputs

Assume the datetime of the post creation form submission is 30th April, 23:59.

Input	Expected output	Remarks

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
eventOrganizer: “Ryan”
eventTitle: “Personal Introduction”
eventStartDatetime:
“2021-05-20T09:00”
eventEndDatetime: “2021-05-20T13:00”
eventTags: [“Fun”]
eventPoster: arrayBuffer(
name: “self.jpg”,
size: 1012853,
type: "image/jpg"
)
eventDescription: “Teach programming”
	Error message for rejection of the post creation request because of the invalid userType	The requesting user is a student user, which is not authorized to perform the post creation

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
	Error messages shown under the input boxes for the event title, event starting datetime and event ending datetime	The inputted event title, event starting datetime and event ending datetime are empty

eventOrganizer: “Coding Society” eventTitle: “” eventStartDatetime: “” eventEndDatetime: “” eventTags: [] eventPoster: undefined eventDescription: “”		

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
eventOrganizer: “Coding Society”
eventTitle: “101010101010...” (51 chars)
eventStartDatetime:
“2021-05-20T09:00”
eventEndDatetime: “2021-05-20T13:00”
eventTags: [“Computer”]
eventPoster: arrayBuffer(
name: “poster.jpg”,
size: 1409778,
type: "image/jpeg"
)
eventDescription: “Teach programming”
	Error messages shown under the input boxes for the event title	The length of the event title exceeds 50

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
eventOrganizer: “Coding Society”
eventTitle: “Coding Workshop”
eventStartDatetime:
“2021-04-27T09:00”
eventEndDatetime: “2021-04-27T13:00”
eventTags: [“Computer”]
eventPoster: arrayBuffer(
name: “poster.jpg”,
size: 1409778,
type: "image/jpeg"
)
eventDescription: “Teach programming”
	Error messages shown under the input boxes for the event starting datetime and the event ending datetime	The event starting datetime and the event ending datetime are before the datetime of the post creation form submission

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
eventOrganizer: “Coding Society”
eventTitle: “Coding Workshop”
eventStartDatetime:
“2048-01-01T14:00”
eventEndDatetime: “2048-01-01T18:00”
eventTags: [“Computer”]
eventPoster: arrayBuffer(
name: “poster.jpg”,
size: 1409778,
type: "image/jpeg"
)
eventDescription: “Teach programming”
	Error messages shown under the input boxes for the event starting datetime and the event ending datetime	The event starting datetime and the event ending datetime are not within the following one year from the datetime of the post creation form submission

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
eventOrganizer: “Coding Society”
eventTitle: “Coding Workshop”
eventStartDatetime:
“2021-06-05T09:00”
eventEndDatetime: “2048-06-04T13:00”
eventTags: [“Computer”]
eventPoster: arrayBuffer(
name: “poster.jpg”,
size: 1409778,
type: "image/jpeg"
)
eventDescription: “Teach programming”
	Error messages shown under the input boxes for the event starting datetime and the event ending datetime	The event starting datetime is before the event ending datetime

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
eventOrganizer: “Coding Society”
eventTitle: “Coding Workshop”
	Error messages shown under the input boxes for the event poster	The selected file of the event poster has size
		= 5242880 byte
		= 5MB,
		which exceeds the restricted maximum

eventStartDatetime: “2021-05-20T09:00” eventEndDatetime: “2021-05-20T13:00” eventTags: [“Computer”] eventPoster: arrayBuffer(name: “poster.jpg”, size: 5242880, type: "image/jpeg") eventDescription: “Teach programming”		file size, which is 2MB

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
eventOrganizer: “Coding Society”
eventTitle: “Coding Workshop”
eventStartDatetime:
“2021-05-20T09:00”
eventEndDatetime: “2021-05-20T13:00”
eventTags: [“Computer”]
eventPoster: arrayBuffer(
name: “P_TU1.pdf”,
size: 664905,
type: "application/pdf"
)
eventDescription: “Teach programming”
	Error messages shown under the input boxes for the event poster	The selected file is a pdf file, which is not accepted

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
eventOrganizer: “Coding Society”
eventTitle: “Coding Workshop”
eventStartDatetime:
“2021-05-20T09:00”
eventEndDatetime: “2021-05-20T13:00”
eventTags: []
eventPoster: undefined
eventDescription: “”
	A post ID is generated and returned to the user; The browser is directed to the post page corresponding to the returned post ID, which displays the event information	Valid post and event information with no event tags, event poster and event description

authToken: {	A post ID is generated	Valid post and event

userID: “soc001” userName: “Coding Society” userType: “society” } eventOrganizer: “Coding Society” eventTitle: “Coding Workshop” eventStartDatetime: “2021-05-20T09:00” eventEndDatetime: “2021-05-20T13:00” eventTags: [“Computer”] eventPoster: arrayBuffer(name: “poster.jpg”, size: 1409778, type: "image/jpeg") eventDescription: “Teach programming”	and returned to the post creator; The browser is directed to the post page corresponding to the returned post ID, which displays the event information	information

5.8 EDIT POST

5.8.1 Purpose

To test the post edit function and the verification performed according to the following conditions:

- The inputted event title, event starting datetime and event ending datetime must be non-empty
- The event starting datetime and the event ending datetime must be equal to or later than the datetime of the post creation form submission
- The event starting datetime and the event ending datetime must be within the following one year from the datetime of the post creation form submission
- The event starting datetime must be equal to or before the event ending datetime, and the event ending datetime must be equal to or after the event starting datetime
- The selected file of the event poster must have a size at most 2MB = 2097152 byte
- The format of the selected file of the event poster must be either JPEG or PNG

5.8.2 Inputs and expected outputs

Assume the datetime of the post edit form submission is 1st May, 12:00.

Input	Expected output	Remarks

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
eventOrganizer: “Coding Society”
eventTitle: “”
eventStartDatetime: “”
eventEndDatetime: “”
eventTags: [“Computer”]
eventPoster: undefined
eventDescription: “”
	Error messages shown under the input boxes for the event title, event starting datetime and event ending datetime	The inputted event title, event starting datetime and event ending datetime are empty

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
eventOrganizer: “Coding Society”
eventTitle: “101010101010...” (51 chars)
eventStartDatetime:
“2021-05-20T09:00”
eventEndDatetime: “2021-05-20T13:00”
eventTags: [“STEM”, “Computer”]
eventPoster: undefined
	Error messages shown under the input boxes for the event title	The length of the event title exceeds 50

eventDescription: “Teach arduino”		

authToken: {
  userID: “soc001”
  userName: “Coding Society”
  userType: “society”
}
eventOrganizer: “Coding Society”
eventTitle: “STEM Workshop”
eventStartDatetime:
“2021-04-27T09:00”
eventEndDatetime: “2021-04-27T13:00”
eventTags: [“STEM”, “Computer”]
eventPoster: undefined
eventDescription: “Teach arduino”
	Error messages shown under the input boxes for the event starting datetime and the event ending datetime	The event starting datetime and the event ending datetime are before the datetime of the post creation form submission

authToken: {
  userID: “soc001”
  userName: “Coding Society”
  userType: “society”
}
eventOrganizer: “Coding Society”
eventTitle: “STEM Workshop”
eventStartDatetime:
“2048-01-01T14:00”
eventEndDatetime: “2048-01-01T18:00”
eventTags: [“STEM”, “Computer”]
eventPoster: undefined
eventDescription: “Teach arduino”
	Error messages shown under the input boxes for the event starting datetime and the event ending datetime	The event starting datetime and the event ending datetime are not within the following one year from the datetime of the post creation form submission

authToken: {
  userID: “soc001”
  userName: “Coding Society”
  userType: “society”
}
eventOrganizer: “Coding Society”
eventTitle: “STEM Workshop”
eventStartDatetime:
“2021-06-05T09:00”
eventEndDatetime: “2048-06-04T13:00”
eventTags: [“STEM”, “Computer”]
eventPoster: undefined
eventDescription: “Teach arduino”
	Error messages shown under the input boxes for the event starting datetime and the event ending datetime	The event starting datetime is before the event ending datetime

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
eventOrganizer: “Coding Society”
eventTitle: “STEM Workshop”
eventStartDatetime:
“2021-05-27T09:00”
eventEndDatetime: “2021-05-27T13:00”
eventTags: [“STEM”, “Computer”]
eventPoster: arrayBuffer(
name: “stem.png”,
size: 3145728,
type: "image/png"
)
eventDescription: “Teach arduino”
	Error messages shown under the input boxes for the event poster	The selected file of the event poster has size
		= 3145728 byte
		= 3MB,
		which exceeds the restricted
		maximum file size, which is 2MB

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
eventOrganizer: “Coding Society”
eventTitle: “STEM Workshop”
eventStartDatetime:
“2021-05-27T09:00”
eventEndDatetime: “2021-05-27T13:00”
eventTags: [“STEM”, “Computer”]
eventPoster: arrayBuffer(
name: “P_TU1.pdf”,
size: 664905,
type: "application/pdf"
)
eventDescription: “Teach arduino”
	Error messages shown under the input boxes for the event poster	The selected file is a pdf file, which is not accepted

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
eventOrganizer: “Coding Society”
eventTitle: “Coding Workshop”
	The browser is directed back to the edited post page, which displays the updated event information. The event poster remains unchanged	Valid post and event information with no event poster uploaded

eventStartDatetime: “2021-05-27T09:00” eventEndDatetime: “2021-05-27T13:00” eventTags: [“STEM”, “Computer”] eventPoster: undenied eventDescription: “Teach arduino”		

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
eventOrganizer: “Coding Society”
eventTitle: “Coding Workshop”
eventStartDatetime:
“2021-05-27T09:00”
eventEndDatetime: “2021-05-27T13:00”
eventTags: [“STEM”, “Computer”]
eventPoster: arrayBuffer(
name: “stem.png”,
size: 1782579,
type: "image/png"
)
eventDescription: “Teach arduino”
	The browser is directed back to the edited post page, which displays the updated event information	Valid post and event information

5.9 COMMENT ON POST

5.9.1 Purpose

To test the post comment function and the verification performed according to the following conditions:

- The inputted comment must be non-empty
- The inputted comment must have a length at most 180

5.9.2 Inputs and expected outputs

Input	Expected output	Remarks
commenterID: 1155125428 commenterName: Ryan commentContent: “”	Message pops over the input box for the comment content for the empty comment content	The inputted comment content is empty
commenterID: 1155125428 commenterName: Ryan commentContent: “buh bah buh bah buh bah buh ... ” (200 chars)	Only the prefix of the comment content with length = 180 is extracted to be the comment content. The extracted comment is added to the post and displayed in the comment section on the post content page	The inputted comment content has a length of 200 characters, which exceeds the maximum length, which is 180
commenterID: 1155125428 commenterName: Ryan commentContent: “I am looking forward to the event!”	The comment is added to the post and displayed in the comment section on the post content page	The inputted event title, event starting datetime and event ending datetime are empty

5.10 DELETE POST

5.10.1 Purpose

To test the post deletion function according to the following conditions:

- The deletion of the post can only be performed by the post creator

5.10.2 Inputs and expected outputs

Input	Expected output	Remarks

eventOrganizerID: soc001
authToken: {
userID: “soc005”
userName: “CU Chess Club”
userType: “society”
}
	The message of rejecting the post deletion request	The society user with the user ID = soc005 is not the post creator

eventOrganizerID: soc001
authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
	The post is deleted and the browser is directed back to the post browser page at the location ./browsePosts	Matched the user’s identity with the post creator.

5.11 VIEW POST

5.11.1 Purpose

To test entering the post according to the following conditions:

- The “delete” and “edit” buttons on the post content page are only shown to the post creator of the viewed post
- For the first time the user visits the post, the number of viewers is incremented by 1

5.11.2 Inputs and expected outputs

Input	Expected output	Remarks

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
eventOrganizerID: soc005
number of viewers: 4
viewedBy: [“soc001”, “soc005”, “1155156830”, “1155124368”]
	The post page without the “delete” and “edit” button;
	
	the revise information:
	number of viewers: 4
	viewedBy: [“soc001”, “soc005”, “1155156830”, “1155124368”, “1155125428”]
		The viewing user is not the creator of the viewed post

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
eventOrganizerID: soc001
number of viewers: 0
viewedBy: []
	The post page with the “delete” and “edit” button;
	
	the revise information:
	number of viewers: 1
	viewedBy: [“soc001”]
		The viewing user is the creator of the viewed post

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
eventOrganizerID: soc001
number of viewers: 1
viewedBy: [“soc001”]
	The post page with the “delete” and “edit” button;
	
	The number of viewers and viewedBy remains unchanged
		The viewing user is the creator of the viewed post and the post has already been visited by the viewing user before

5.12 SUBSCRIBE SOCIETY

5.12.1 Purpose

To test the subscription function according to the following condition:

- The subscribing user must be the requesting user, so the passed student ID of the subscribing user must equal the user ID of the requesting user
- The user type of the requesting user, i.e. subscribing user, must be a student
- There exists a student user with the inputted student ID and a society user with the inputted society ID

5.12.2 Inputs and expected outputs

Input	Expected output	Remarks

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
studentID: 1155124886
societyID: soc001
Post request via the api, /api/subscribe
	{message: "Only the subscribing student is authorized for the operation", status: 3}	Invalid authToken
		The requesting user is not the subscribing user

authToken: {
userID: “soc005”
userName: “CU Chess Club”
userType: “society”
}
studentID: soc005
societyID: soc001
Post request via the api, /api/subscribe
	{message:"Only student users are authorized for the operation", status: 3}	Invalid authToken and studentID
		The requesting user is not a student user

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
studentID: 1155125428
societyID: 1155124886
Post request via the api, /api/subscribe
	{message:"Society not found", status: 2}	Invalid societyID
		The society user with the society ID = 1155124886 is not found

authToken(fake): { userID: “1155000000”	{message:"Student not found", status: 2}	Invalid studentID The student user with the

userName: “Wrong” userType: “student” } studentID: 1155000000 societyID: soc001 Post request via the api, /api/subscribe		student ID = 1155000000 is not found

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
studentID: 1155125428
societyID: soc001
Post request via the api, /api/subscribe
	{message: “Subscription succeeds”, status: 0}	Valid authToken, studentID and societyID

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
studentID: 1155125428
societyID: soc001
Click on the “Subscribe” button on the profile page of the society
	The button “Subscribe” on the society profile page is changed to “Unsubscribe” and the number of subscribers is incremented by 1.	Valid authToken, studentID and societyID

5.13 UNSUBSCRIBE SOCIETY

5.13.1 Purpose

To test the unsubscription function according to the following condition:

- The unsubscribing user must be the requesting user, so the passed student ID of the unsubscribing user must equal the user ID of the requesting user
- The user type of the requesting user, i.e.,unsubscribing user, must be a student
- There must exist a student user with the inputted student ID and a society user with the inputted society ID

5.13.2 Inputs and expected outputs

Input	Expected output	Remarks

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
studentID: 1155124886
societyID: soc001
Post request via the api, /api/unsubscribe
	{message: "Only the unsubscribing student is authorized for the operation", status: 3}	Invalid authToken
		The requesting user is not the unsubscribing user

authToken: {
userID: “soc005”
userName: “CU Chess Club”
userType: “society”
}
studentID: soc005
societyID: soc001
Post request via the api, /api/unsubscribe
	{message:"Only student users are authorized for the operation", status: 3}	Invalid authToken and studentID
		The requesting user is not a student user

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
studentID: 1155125428
societyID: 1155124886
Post request via the api, /api/unsubscribe
	{message:"Society not found", status: 2}	Invalid societyID
		The society user with the society ID = 1155124886 is not found

authToken(fake): { userID: “1155000000”	{message:"Student not found", status: 2}	Invalid studentID The student user with the

userName: “Wrong” userType: “student” } studentID: 1155000000 societyID: soc001 Post request via the api, /api/unsubscribe		student ID = 1155000000 is not found

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
studentID: 1155125428
societyID: soc001
Post request via the api, /api/unsubscribe
	{message: “Unsubscription succeeds”, status: 0}	Valid authToken, studentID and societyID

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
studentID: 1155125428
societyID: soc001
Click on the “Unsubscribe” button on the profile page of the society
	The button “Unsubscribe” on the society profile page is changed to “Subscribe” and the number of subscribers is decremented by 1.	Valid authToken, studentID and societyID

5.14 CREATE SEMINAR

5.14.1 Purpose

To test the seminar creation function and the verification performed according to the following conditions:

- The inputted seminar title and seminar quota must be non-empty
- The seminar quota must be at least 1 and at most 50

5.14.2 Inputs and expected outputs

Input	Expected output	Remarks
seminarModerator: “Coding Society” seminarTitle: “” seminarTags: [] seminarQuota: “” seminarMutedByDefault: false	Error messages shown under the input boxes for the seminar title and seminar quota	The inputted seminar title and seminar quota are empty
seminarModerator: “Coding Society” seminarTitle: “1010101010...” (51 chars) seminarTags: [“Computer”] seminarQuota: “5” seminarMutedByDefault: false	Error messages shown under the input boxes for the event title	The length of the seminar title exceeds 50
seminarModerator: “Coding Society” seminarTitle: “Developer conference” seminarTags: [“Computer”] seminarQuota: “0” seminarMutedByDefault: false	Error messages shown under the input boxes for the seminar quota	Invalid seminar quota. The inputted seminar quota is less than 1
seminarModerator: “Coding Society” seminarTitle: “Developer conference” seminarTags: [“Computer”] seminarQuota: “100” seminarMutedByDefault: false	Error messages shown under the input boxes for the seminar quota	Invalid seminar quota. The inputted seminar quota is more than 50
seminarModerator: “Coding Society” seminarTitle: “Developer conference” seminarTags: [] seminarQuota: “100” seminarMutedByDefault: false	A seminar ID is generated and returned to the seminar moderator and the browser is directed to the seminar room page corresponding to the returned seminar ID	Valid seminar information with no seminar tags and seminarMutedByDefault = false
seminarModerator: “Coding Society” seminarTitle: “Developer conference”	A seminar ID is generated and the	Valid seminar information with

seminarTags: [“Computer”] seminarQuota: “100” seminarMutedByDefault: true	seminar session data storage is constructed in the server; The seminar ID is returned to the seminar moderator and the browser is directed to the seminar room page corresponding to the returned seminar ID	seminar tag and seminarMutedByDefault = true

5.15 JOIN SEMINAR

5.15.1 Purpose

To test joining the seminar according to the following conditions:

- There must exist a seminar room with the inputted seminar ID
- The joining user must not be on the blacklist of the seminar room
- The current number of members participating in the room must be less than the maximum seminar participant quota
- The returned user role in the seminar for the seminar moderator is “moderator” and that for the seminar participants is “participant”

5.15.2 Inputs and expected outputs

Assume there exists a seminar with seminar ID = “6089191a3b40df3bec655a40”

Input	Expected output	Remarks
emittedSignal: “knock-room” seminarID: “wrongseminarID” userID: “1155125428” userName: “Ryan” moderatorID: undefined blackList(in server): undefined quota: 50 currentMemberNum(in server): undefined	The browser is redirected to the Not Found page	The seminar with the seminar ID = “wrongseminarID” is not found
emittedSignal: “knock-room” seminarID: “6089191a3b40df3bec655a40” userID: “1155125428” userName: “Ryan” moderatorID: “soc001” blackList(in server): [“1155125428”, “1155124886”] quota: 50 currentMemberNum(in server): 1	An alert pops up for rejecting the user from joining the seminar room because the user is already on the blacklist	The userID “1155125428’ is added to the black list of the seminar room with seminar ID = “6089191a3b40df3bec655a40”
emittedSignal: “knock-room” seminarID: “6089191a3b40df3bec655a40” userID: “1155125428” userName: “Ryan” moderatorID: “soc001” blackList(in server): [“1155125428”, “1155124886”] quota: 50	An alert pops up for rejecting the user from joining the seminar room because the seminar room is full	The seminar room is full that the current number of members participating in the seminar reaches the maximum

currentMemberNum(in server): 50		
emittedSignal: “knock-room” seminarID: “6089191a3b40df3bec655a40” userID: “soc001” userName: “Coding Society” moderatorID: “soc001” blackList(in server): [] quota: 50 currentMemberNum(in server): 0	receivedSignal: “connection-succeeds”; returned seminar info; returned user role in the seminar: “moderator”; The user is added to the member list on the seminar room page and a system chat message is displayed in the chat room informing for the user opening the seminar; The current number of the seminar members is incremented by 1	Valid seminar ID and user, who is the seminar moderator
emittedSignal: “knock-room” seminarID: “6089191a3b40df3bec655a40” userID: “1155125428” userName: “Ryan” moderatorID: “soc001” blackList(in server): [] quota: 1 currentMemberNum(in server): 50	receivedSignal: “connection-succeeds”; returned seminar info; returned user role in the seminar: “participant”; The user is added to the member list on the seminar room page and a system chat message is displayed in the chat room for informing the user to open the seminar The current number of the seminar members is incremented by 1	Valid seminar ID and user, who is the seminar participant

5.16 QUIT SEMINAR

5.16.1 Purpose

To test quitting the seminar according to the following conditions:

- The seminar is ended upon the seminar moderator quits the seminar room

5.16.2 Inputs and expected outputs

Input	Expected output	Remarks
emittedSignal: “disconnect” userRole: “participant”	The user is removed from the member list on the seminar room page and a system chat message is displayed in the chat room informing for the user quitting the seminar; The current number of the seminar members is decremented by 1	The seminar participant quits the room
emittedSignal: “disconnect” userRole: “moderator”	All seminar members in the seminar room received a “seminar-ended” signal, which forces them to quit the seminar and be directed to the seminar entrance; The seminar room is closed	The seminar moderator quits the room

5.17 SEND CHAT MESSAGE

5.17.1 Purpose

To test the chat function in the seminar according to the following conditions:

- The inputted chat message must be non-empty
- The inputted chat message must have a length at most 500
- The message sender must not be in the mute list of the seminar room

5.17.2 Inputs and expected outputs

Assume there exists a seminar with seminar ID = “6089191a3b40df3bec655a40”

Input	Expected output	Remarks
emittedSignal: “send-msg” seminarID: “6089191a3b40df3bec655a40” userID: “1155125428” userName: “Ryan” chatMessage: “” muteList (in server): []	Nil	The inputted chat message is empty
emittedSignal: “send-msg” seminarID: “6089191a3b40df3bec655a40” userID: “1155125428” userName: “Ryan” chatMessage: “bah buh bah buh bah buh bah …” (501 words) muteList (in server): []	receivedSignal: “recieve-msg”; returned userID and userName of the message sender; Only the prefix of the comment content with length = 500 is extracted for the chat message; The message block which contains the username of the message sender and the chat message content, is displayed in the chat box in the seminar room	The inputted comment content has a length of 501 characters, which exceeds the maximum length, which is 500
emittedSignal: “send-msg” seminarID: “6089191a3b40df3bec655a40” userID: “1155125428” userName: “Ryan” chatMessage: “How are you?” muteList (in server): [“1155125428”]	The message of rejecting the user from sending the chat message	The user (message sender) is in the muted list of the seminar room

emittedSignal: “send-msg” seminarID: “6089191a3b40df3bec655a40” userID: “1155125428” userName: “Ryan” chatMessage: “How are you?” muteList (in server): []	receivedSignal: “recieve-msg”; returned userID and userName of the message sender; The message block which contains the user name of the message sender and the chat message content, is displayed in the chat box in the seminar room	Valid chat message and user

5.18 MUTE PARTICIPANT

5.18.1 Purpose

To test the mute function in the seminar according to the following conditions:

- The mute function is only authorized by the seminar moderator
- There must exist a participant with the inputted mutedUserID in the seminar room
- Only the participants of the seminar can be muted

5.18.2 Inputs and expected outputs

Input	Expected output	Remarks

emittedSignal: “mute-parti”
seminarID: “6089191a3b40df3bec655a40”
userID: “soc005”
userName: “CU Chess Club”
userRole: “participant”
mutedUserID: “1155125428”
memberLists: {
moderator: [{userID:“soc001”,...}],
participant: [{userID:“1155125428”,...},
{userID:“soc005”,...}]
}
muteList (in server): []
	Nil	The user role of the requesting user is “participant”, so the emitted signal results in no response

emittedSignal: “mute-parti”
seminarID: “6089191a3b40df3bec655a40”
userID: “soc001”
userName: “Coding Society”
userRole: “moderator”
mutedUserID: “1155123456”
memberLists: {
moderator: [{userID:“soc001”,...}],
participant: [{userID:“1155125428”,...},
{userID:“soc005”,...}]
}
muteList (in server): []
	The message of rejecting the user’s request because the member with the userID = “1155123456” is not participating in the seminar room.	The user with the user ID = “1155123456” is not found in the member lists of the seminar room

emittedSignal: “mute-parti” seminarID: “6089191a3b40df3bec655a40” userID: “soc001” userName: “Coding Society” userRole: “moderator” mutedUserID: “1155125428” memberLists: { moderator: [{userID:“soc001”,...}],	Nil	The muted user has already been the seminar room muted list

participant: [{userID:“1155125428”,...}, {userID:“soc005”,...}] } muteList (in server): [“1155125428”]		

emittedSignal: “mute-parti”
seminarID: “6089191a3b40df3bec655a40”
userID: “soc001”
userName: “Coding Society”
userRole: “moderator”
mutedUserID: “soc001”
memberLists: {
moderator: [{userID:“soc001”,...}],
participant: [{userID:“1155125428”,...},
{userID:“soc005”,...}]
}
muteList (in server): [“1155125428”]
	The message of rejecting the user’s request because it is unavailable to mute the seminar moderator	The muted user is the seminar moderator

emittedSignal: “mute-parti”
seminarID: “6089191a3b40df3bec655a40”
userID: “soc001”
userName: “Coding Society”
userRole: “moderator”
mutedUserID: “soc005”
memberLists: {
moderator: [{userID:“soc001”,...}],
participant: [{userID:“1155125428”,...},
{userID:“soc005”,...}]
}
muteList (in server): [“1155125428”]
	receivedSignal:
	“user-muted”;
	returned userID of the muted user;
	returned message for the mute operation;
	The chatting input box of the muted user is disabled;
	A “Muted” icon is shown beside the name of the muted user on the member list
	the revised muteList =
	[“1155125428”, “soc005”]
		Valid muter and muted user

5.19 UNMUTE PARTICIPANT

5.19.1 Purpose

To test the unmute function in the seminar according to the following conditions:

- The unmute function is only authorized to the seminar moderator
- There must exist a participant with the inputted unmutedUserID in the seminar room

5.19.2 Inputs and expected outputs

Input	Expected output	Remarks

emittedSignal: “unmute-parti”
seminarID: “6089191a3b40df3bec655a40”
userID: “1155125428”
userName: “Ryan”
userRole: “participant”
unmutedUserID: “1155125428”
memberLists: {
moderator: [{userID:“soc001”,...}],
participant: [{userID:“1155125428”,...},
{userID:“soc005”,...}]
}
muteList (in server): [“1155125428”]
	Nil	The user role of the requesting user is “participant”, so the emitted signal results in no response

emittedSignal: “unmute-parti”
seminarID: “6089191a3b40df3bec655a40”
userID: “soc001”
userName: “Coding Society”
userRole: “moderator”
unmutedUserID: “1155123456”
memberLists: {
moderator: [{userID:“soc001”,...}],
participant: [{userID:“1155125428”,...},
{userID:“soc005”,...}]
}
muteList (in server): [“1155125428”]
	The message of rejecting the user’s request because the member with the userID = “1155123456” is not participating in the seminar room.	The user with the user ID = “1155123456” is not found in the member lists of the seminar room

emittedSignal: “unmute-parti” seminarID: “6089191a3b40df3bec655a40” userID: “soc001” userName: “Coding Society” userRole: “moderator” unmutedUserID: “1155125428” memberLists: { moderator: [{userID:“soc001”,...}], participant: [{userID:“1155125428”,...},	Nil	The unmuted user is not in the seminar room muted list

{userID:“soc005”,...}] } muteList (in server): []		

emittedSignal: “unmute-parti”
seminarID: “6089191a3b40df3bec655a40”
userID: “soc001”
userName: “Coding Society”
userRole: “moderator”
unmutedUserID: “soc005”
memberLists: {
moderator: [{userID:“soc001”,...}],
participant: [{userID:“1155125428”,...},
{userID:“soc005”,...}]
}
muteList (in server): [“1155125428”,
“soc005”]
	receivedSignal: “user-unmuted”;
	returned userID of the unmuted user;
	returned message for the unmute operation;
	The chatting input box of the unmuted user is enabled;
	The “Muted” icon beside the name of the unmuted user on the member list is hidden
	the revised muteList = [“soc005”]
		Valid unmuter and unmuted user

5.20 KICK PARTICIPANT

5.20.1 Purpose

To test the kick function in the seminar according to the following conditions:

- The kick function is only authorized to the seminar moderator
- There must exist a participant with the inputted kickedUserID in the seminar room
- Only the participants of the seminar can be kicked

5.20.2 Inputs and expected outputs

Input	Expected output	Remarks

emittedSignal: “kick-parti”
seminarID: “6089191a3b40df3bec655a40”
userID: “soc005”
userName: “CU Chess Club”
userRole: “participant”
kickedUserID: “1155125428”
memberLists: {
moderator: [{userID:“soc001”,...}],
participant: [{userID:“1155125428”,...},
{userID:“soc005”,...}]
}
blackList (in server): []
	Nil	The user role of the requesting user is “participant”, so the emitted signal results in no response

emittedSignal: “kick-parti”
seminarID: “6089191a3b40df3bec655a40”
userID: “soc001”
userName: “Coding Society”
userRole: “moderator”
kickedUserID: “1155123456”
memberLists: {
moderator: [{userID:“soc001”,...}],
participant: [{userID:“1155125428”,...},
{userID:“soc005”,...}]
}
blackList (in server): []
	The message of rejecting the user’s request because the member with the userID= “1155123456” is not participating in the seminar room.	The user with the user ID = “1155123456” is not found in the member lists of the seminar room

emittedSignal: “kick-parti” seminarID: “6089191a3b40df3bec655a40” userID: “soc001” userName: “Coding Society” userRole: “moderator” kickedUserID: “soc001” memberLists: { moderator: [{userID:“soc001”,...}],	The message of rejecting the user’s request because it is unavailable to kick the seminar moderator	The kicked user is the seminar moderator

participant: [{userID:“1155125428”,...}, {userID:“soc005”,...}] } blackList (in server): []		

emittedSignal: “kick-parti”
seminarID: “6089191a3b40df3bec655a40”
userID: “soc001”
userName: “Coding Society”
userRole: “moderator”
kickedUserID: “1155125428”
memberLists: {
moderator: [{userID:“soc001”,...}],
participant: [{userID:“1155125428”,...},
{userID:“soc005”,...}]
}
blackList (in server): []
	receivedSignal: “seminar-ended”;
	returned message for the kick operation;
	The kicked user is forced to quit the seminar room and directed to the seminar entrance.
	the revised blackList = [“1155125428”]
		Valid kicker and kicked user

5.21 SEND VERIFICATION EMAIL

5.21.1 Purpose

To test the delivery of the verification email for the signed up account activation.

5.21.2 Inputs

- Successful sign-up
- The sample account information for signing up is as follows:

userName	“Ryan Chung”
email	“1155125428@link.cuhk.edu.hk”
password	“testpassword”
confirmPassword	“testpassword”

5.21.3 Expected outputs

Email sent to 1155125428@link.cuhk.edu.hk : 250 2.0.0 OK 1619621956 w134sm50671pfd.173 - gsmtp

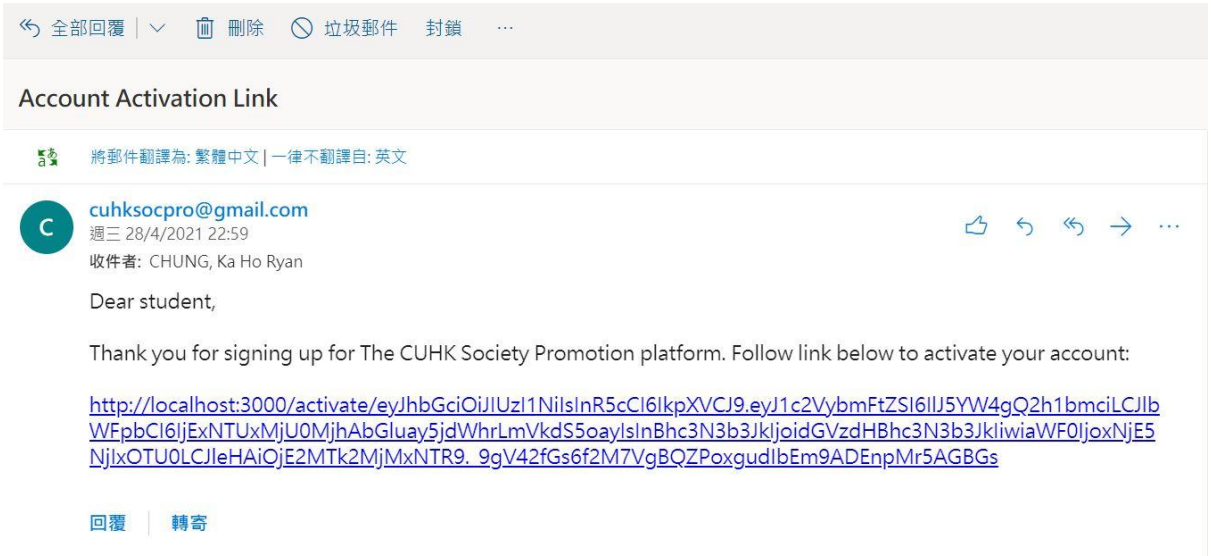

Figure 40. Account Activation

5.22 SEND NOTIFICATION EMAIL

5.22.1 Purpose

To test the delivery of the notification email for the newly created post.

5.22.2 Inputs

- Successful post creation
- The sample event information for the post creation is as follows:

eventOrganizer	“Coding Society”
eventTitle	“Coding Workshop”
eventStartDatetime	“2021-07-28T09:00”
eventEndDatetime	“2021-07-28T13:00”
eventTags	[“Computer”]
eventPoster	undefined
eventDescription	“Teach Python”

5.22.3 Expected outputs

Email sent to 1155125428@link.cuhk.edu.hk : 250 2.0.0 OK 1619622775 mv14sm1942060pjb.57 - gsmtp

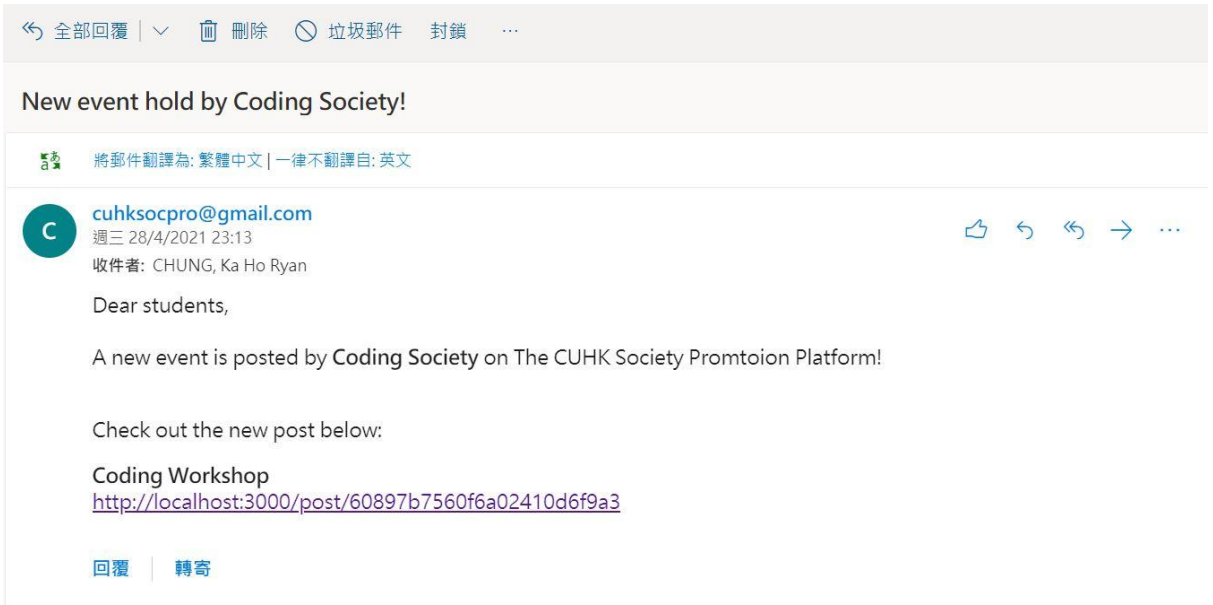

Figure 41. Event Notification

5.23 SEND WEEKLY DIGEST EMAIL

5.23.1 Purpose

To test the delivery of the weekly digest email.

5.23.2 Testing Procedure

Change the system date and times to any Sunday and 12:00.

Figure 42. Testing Procedure (1)

Figure 43. Testing Procedure (2)

5.23.3 Expected outputs

Email sent to 1155125428@link.cuhk.edu.hk : 250 2.0.0 OK 1619626097 d21sm149719pfo.200 - gsmtp

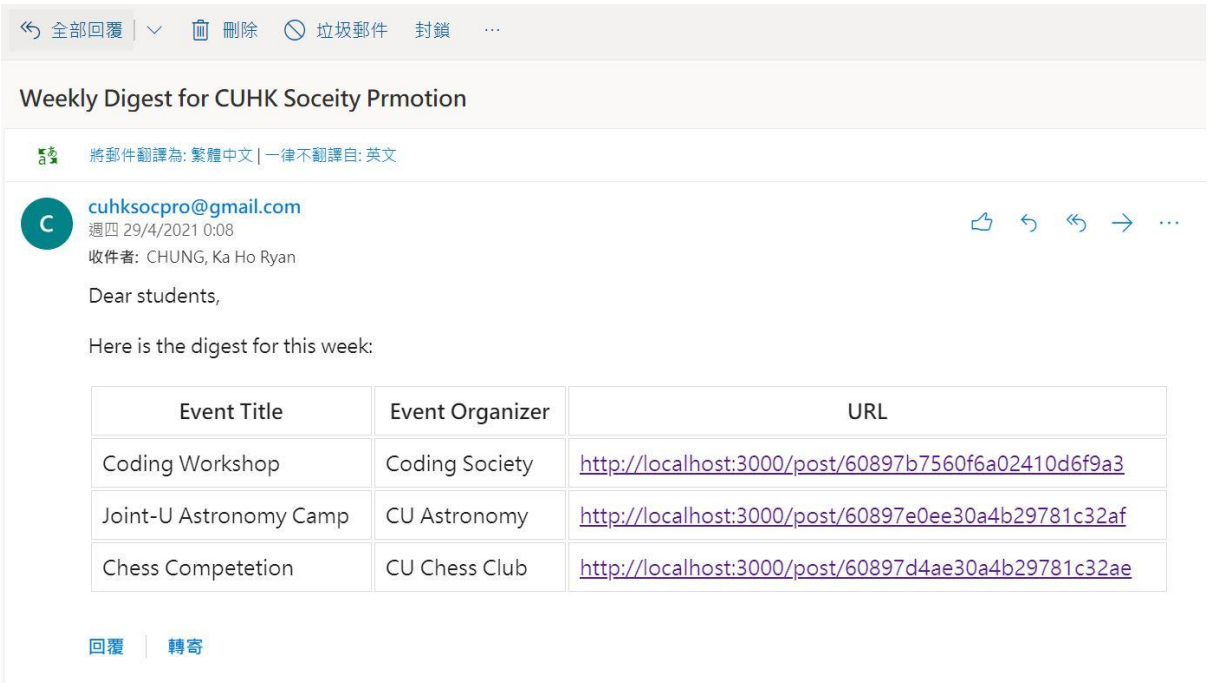

Figure 44. Weekly Digest Notification

5.24 NAVIGATION MENU

5.24.1 Purpose

To test the navigating function of the navigation menu. Note that the Calendar function is only provided for the student users, so the “My Calendar” button on the navigation menu is hidden for the society user.

5.24.2 Inputs and expected outputs

Input	Expected output	Remarks

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
Click on the “Home” button on the navigation menu
	The browser is navigated to:
	/Home
		Navigate to the home page

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
Click on the “My Calendar” button on the navigation menu
	The browser is navigated to:
	/viewCalendar
		Navigate to the calendar page

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
Click on the “My Profile” button on the navigation menu
	The browser is navigated to:
	/StudentProfile/1155125428
		Navigate to the student profile page of the user as his user type is “student”

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
Click on the “Society List” button on the navigation menu
	The browser is navigated to:
	/browseSocieties
		Navigate to the society list page

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
Click on the “Available Events” button on the navigation menu
	The browser is navigated to:
	/browsePosts
		Navigate to the post browser page

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
Click on the “Join Seminars” button on the navigation menu
	The browser is navigated to:
	/browseSeminars
		Navigate to the seminar entrance page

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
	Navigation menu with no “My Calendar” button	The user type of the user is “society” so the calendar component is disabled

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
Click on the “Home” button on the navigation menu
	The browser is navigated to:
	/Home
		Navigate to the home page

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
Click on the “My Profile” button on the navigation menu
	The browser is navigated to:
	/SocietyProfile/soc001
		Navigate to the calendar page

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
Click on the “Society List” button on the navigation menu
	The browser is navigated to:
	/browseSocieties
		Navigate to the society profile page of the user as his user type is “society”

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
Click on the “Available Events” button on the navigation menu
	The browser is navigated to:
	/browsePosts
		Navigate to the society list page

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
Click on the “Join Seminars” button on the navigation menu
	The browser is navigated to:
	/browseSeminars
		Navigate to the post browser page

5.25 URL ROUTING

5.25.1 Purpose

To test the url redirection according to the following conditions:

- The app must route to a component according to the location
- The society user is disabled from routing to the Calendar component at the location /viewCalendar
- There must be an existing post with the postID for routing to the PostContent component at the location /post/:postID
- The user type of the user must be “society” for routing to the PostCreate component at the location /createPost
- The user must be the creator of the post with the postID for routing to the PostEdit component at the location /post/edit/:postID
- There must be an existing seminar with the seminarID for routing to the SeminarRoom component at the location /seminar/roomID
- The user type of the user must be “society” for routing to the SeminarCreate component at the location /createSeminar
- There must be an existing society user with the societyID for routing to the SocietyProfile component at the location /SocietyProfile/:societyID
- There must be an existing student user with the studentID for routing to the StudentProfile component at the location /StudentProfile/:studentID
- The user must own a valid authToken in the cookie for routing to the location included in the authPath stated in the config file, otherwise, the user is redirected to the login page to perform login
- The user is disabled from routing to the login page and signup page at the location /login and /signup respectively if he/she has already logged in to an account

5.25.2 Inputs and expected outputs

Input	Expected output	Remarks

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
location.pathname:
/
	Welcome page	Route to the GetStarted component

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
location.pathname:
/home
	Home page	Route to the Home component

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
location.pathname:
/viewCalendar
	Calendar page	Route to the EventCalendar component

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
location.pathname:
/browsePosts
	Post browser page	Route to the PostBrowser component

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
location.pathname:
/post/60772b7d2555b342d091a1fe,
where 60772b7d2555b342d091a1fe is the id of the post
	Post page	Route to the PostContent component

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
location.pathname:
/createPost
	Post creation page	Route to the PostCreate component

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
location.pathname:
/post/edit/60772b7d2555b342d091a1fe,
where 60772b7d2555b342d091a1fe is the id of the post
	Post edit page	Route to the PostEdit component

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
location.pathname:
/browseSeminars
	Seminar entrance page	Route to the Seminar Entrance component

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
location.pathname:
/createSeminer
	Seminar creation page	Route to the SeminarCreate component

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
location.pathname:
/seminar/608992110ccf9b31c45f1237,
where 608992110ccf9b31c45f1237 is the id of the seminar
	Seminar room page	Route to the SeminarRoom component

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
location.pathname:
/browseSocieties
	Society browser page	Route to the SocietyBrowser component

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
location.pathname:
/societyProfile/soc001,
where soc001 is the user ID of the society user
	Society profile page of the society user with the user ID = “soc001”	Route to the SocietyProfile component

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
location.pathname:
/studentProfile/1155125428,
where 1155125428 is the user ID of the student user
	Student profile page of the student user with the user ID = “1155125428”	Route to the StudentProfile component

authToken: undefined location.pathname: /login	Login page	Route to the Login component
authToken: undefined location.pathname: /signup	Signup page	Route to the Signup component
authToken: undefined location.pathname: /activate	Account activation page	Route to the Activation component

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
location.pathname:
/nonexistingpath
	Not Found page	Route to the NotFound component

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
location.pathname:
/viewCalendar
	Home page	Redirect to /home because the calendar component is disabled from the society user

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
location.pathname:
	Not Found page	Redirect to /post?notfound=true because the post with the post ID = “nonexistingpost” doesn’t exist

/post/nonexistingpost, where nonexistingpost is the id of the post		

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
location.pathname:
/createPost
	Post browser page	Redirect to /browsePosts because the student user is unauthorized from the post creation function

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
eventOrganizerID: soc001
location.pathname:
/post/edit/60772b7d2555b342d091a1fe,
where 60772b7d2555b342d091a1fe is the id of the post
	Post page	Redirect to /post/60772b7d2555b342d091a1fe because the user is not the creator of the post with the post ID = “60772b7d2555b342d091a1fe”

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
location.pathname:
/seminar/nonexistingseminar,
where nonexistingseminar is the id of the seminar
	Not Found page	Redirect to /seminar?notfound=true because the post with the seminar ID = “nonexistingseminar” doesn’t exist

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
location.pathname:
/createSeminar
	Seminar entrance page	Redirect to /browseSeminars because the student user is unauthorized from the seminar creation function

authToken: { userID: “1155125428”	Not Found page	Redirect to /SocietyProfile?notfound=tr

userName: “Ryan” userType: “student” } location.pathname: /SocietyProfile/nonexistingsociety, where nonexistingsociety is the id of the seminar		ue because the society with the society ID = “nonexistingsociety” doesn’t exist

authToken: {
userID: “1155125428”
userName: “Ryan”
userType: “student”
}
location.pathname:
/SocietyProfile/nonexistingstudent,
where nonexistingstudent is the id of the seminar
	Not Found page	Redirect to /StudentProfile?notfound=true because the student with the student ID = “nonexistingstudent” doesn’t exist

authToken: undefined authPath: ['home', 'browseSocieties', 'browsePosts', 'post', 'post/edit', 'createPost', 'viewCalendar', 'browseSeminars', 'createSeminar', 'seminar', 'SocietyProfile', 'StudentProfile'] nonAuthPath: ['activate', 'login', ‘signup’], location.pathname: /home	Login page	Redirect to /login because the user has not logged into any account and has no authToken

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
authPath: ['home', 'browseSocieties', 'browsePosts', 'post', 'post/edit', 'createPost', 'viewCalendar', 'browseSeminars', 'createSeminar', 'seminar', 'SocietyProfile', 'StudentProfile']
nonAuthPath: ['activate', 'login', ‘signup’],
location.pathname:
	Home page	Redirect to /home because the user has already logged in to an account

/login		

authToken: {
userID: “soc001”
userName: “Coding Society”
userType: “society”
}
authPath: ['home', 'browseSocieties', 'browsePosts', 'post', 'post/edit', 'createPost', 'viewCalendar', 'browseSeminars', 'createSeminar', 'seminar', 'SocietyProfile', 'StudentProfile']
nonAuthPath: ['activate', 'login', ‘signup’],
location.pathname:
/signup
	Home page	Redirect to /home because the user has already logged in to an account

6 LESSON LEARNED

In this course project, we have learned to build a full-stack system from sketch and experienced a full development cycle: all the way from planning, design, implementation, testing to maintenance.

Following the methodology taught in lectures, as well as considering the limitation of time compatibility of the project, we decided to implement our project with a waterfall model that it is easy to manage our working plan and structure the development cycle. Then we decided on our topic and vision by brainstorming what interesting product may be useful and helpful to the CUHK buddies, we came up with the idea of The CUHK Society Promotion System. With the consensus agrees on the idea, we then have a discussion on how to make things happen. We draw the Data Flow Diagrams (DFD) to showcase the data flow and function provided, Entity-Relationship (ER) Diagrams to organise how the tables are organized, and the Architecture diagram to illustrate how the system is presented. Also, Use-case Diagrams and Sequence Diagrams are drawn to have a better understanding of how the user experience will be presented to the actual user.

After the planning and design stage, we decided to use the MERN stack, which stands for MongoDB, Expressjs, Reactjs and Nodejs, for the project development and manage the project on GitHub. We have practice and learn how to organise and modularize the codebase structure according to the website structure, as well as classified different function pages into folders for easy management. As we are developing the system in parallel, it is more vital for us to learn using the Git tools in order to sync up the progress we have made on each separated task. Meanwhile, we have faced some blockers on *git pull* after merging, as different libraries are used by different teammates and we faced some update conflicts in the node modules.

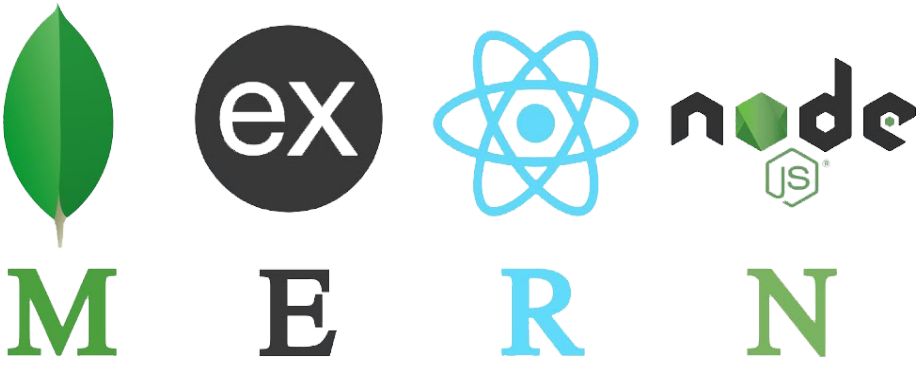

Figure 45. The MERN stack

Afterwards, we set up a database server in MongoDB, we created the data relation with reference to the class diagram we designed, set up the backend APIs for the system to fetch

data from the database via express and mongoose and used Postman to test the functionality. In our origin plan, we separated the task according to the function that the person-in-charge should be responsible for writing both the front-end and backend API. However, we realized the coding style of each teammate varies and some APIs written are duplicated, which causes a high maintenance cost and redundant API usage, so we have spent extra effort to unify and reorganise the API.

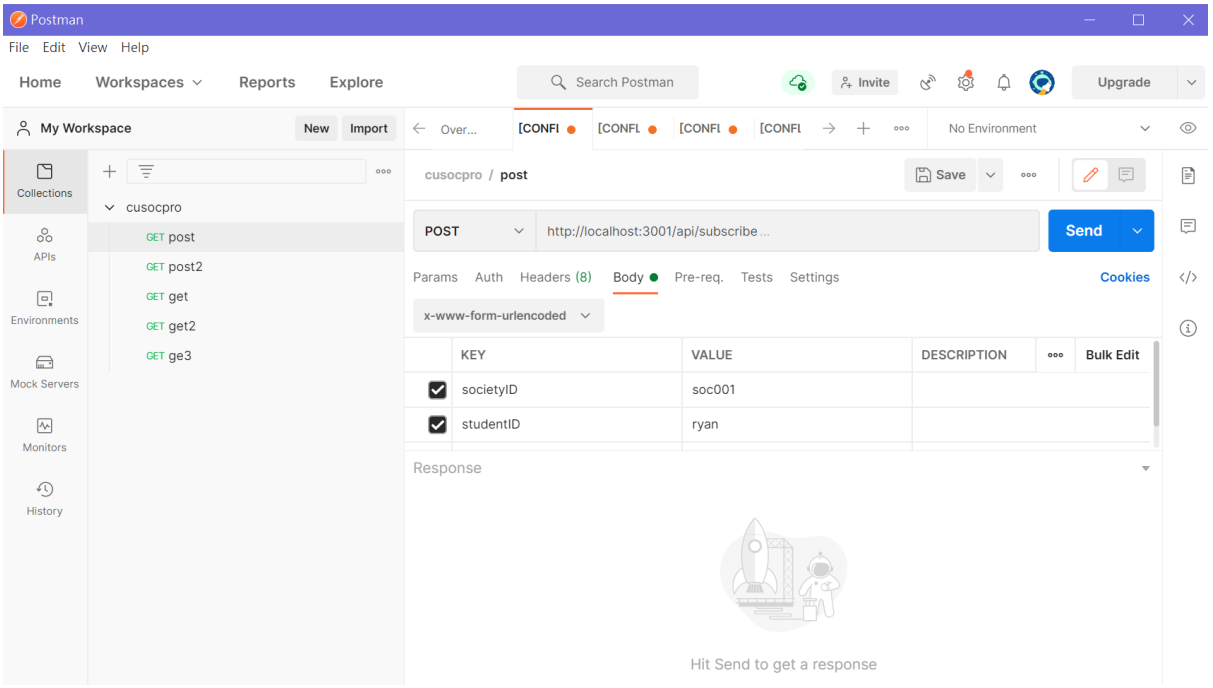

Figure 46. Postman

Throughout the development process, we have done a lot of reading on javascript tutorial with different frameworks, document specification of different libraries, as well as some online discussion platform for self-learning; we have figure out and conclude the effective way to learn coding is to carry the actual development first, so that we will realize what misconception we have in mind.

The testing phase of the project had a few minor hic-ups like displaying issues of the interface's size, or some interaction bugs between different functions like registration and notifications. Yet we have managed to resolve them all thanks to our collaborative efforts.

Although some bugs and unexpected errors are found during the development cycle, we tried our best and solved the problems collaboratively. The CUSocPro project that we have built may not be the best among the class, but we have spared all our efforts and our ideas have indeed turned into reality, which has fulfilled our initial thoughts and met the expectations towards our final goal.

7 CONCLUSION

In the very first beginning of the semester, we observed the student's need on an unified event promotion platform for society, we decided to work on this society promotion platform in order to supplement the mass mail system and hope to provide students with an efficient gateway to gather the information that they are interested in. After a myriad of analysis and testing, a user-friendly and convenient application is deployed. CUSocPro benefits both societies and student participants. To societies, activities are able to attract students more easily due to the variety and quality of the promotion materials. Photos or even videos can be attached, making the promotion well polished. And thanks to the interactive comment system, societies can now promptly improve their activities based on the feedback. To students, they no longer need to search through social media to find activities, thus saving time in finding activities to join. And even during busy periods like the midterm exam periods, email notifications ensure that students could gather their favourite societies' newest information in the nick of time. With CUSocPro, we are sure that a balanced lifestyle can be promoted in CUHK.

8 ACKNOWLEDGMENT

We would like to take this opportunity to express our sincere gratitude to our CSCI3100 course professor, Professor LYU Rung Tsong Michael, and course TAs for their assistance, advice and guidance in this course project. The structured course design and the elaborated tutorials given has motivated our thoughts to push forward and to try out new designs during the deployment of The CUHK Society Promotion System. We are very fortunate to receive such a precious chance and resource to facilitate our wish in implementing something we are fascinated in.

9 REFERENCES

[1] Documentation, Socket.io. (2021). Socket.io. Available: https://socket.io/docs/v4/

[2] Expressjs (2021). API Reference, Express. Available: https://expressjs.com/zh-tw/4x/api.html

[3] Facebook. (2021). Reactjs. Available: https://reactjs.org

[4] FullCalendar. (2021). Full Calendar. Available: https://fullcalendar.io/

[5] Jarrod Overson. (2012). Plato Version 1.7.0. Available: https://github.com/es-analysis/plato

[6] MongoDB. (2021). MongoDB. Available: https://www.mongodb.com/2

[7] MongoDB. (2021). What is the MERN Stack?. Available: https://www.mongodb.com/mern-stack

[8] StackOverflow. (2021). StackOverflow. Available: https://stackoverflow.com/

[9] Stephen J. Collings, Matthew Honnibal, Pieter Vanderwerff. (2014). React-Bootstrap. Available: https://react-bootstrap.github.io/

[10] StrongLoop, IBM. (2017). Expresss.js. Available: https://expressjs.com/

[11] w3schools. (2021). w3schools. Available: https://w3schools.com

10 APPENDIX

10.1 Student User Account

Some student user accounts are provided below for trial use:

- userName: Chan Tai Man
  userID: stu001
  password: stu001

- userName: Lee Siu Man
  userID: stu002
  password: stu002

- userName: Wong Mei Mei
  userID: stu003
  password: stu003

*Please note that the email delivery functions are not supported for the provided accounts

10.2 Society User Account

Some society user accounts are provided below for trial use:

- userName: Coding Society
  userID: soc001
  password: soc001

- userName: CU Chess Club
  userID: soc005
  password: soc005

- userName: CU Astronomy
  userID: soc005
  password: soc005