SUKOCO - SUNY Korea Community: Final Report

Team Members:

Gyuri Kim

Sungwoong Park

1. Introduction

a. Product description

Many new students have a problem finding information about their colleges including SUNY Korea. After being a SUNY Korea student for two years, I felt the absence of the school community among SUNY Korea students, because of this, I was not able to know which courses by which professors are better. This inconvenience disturbed me to choose classes to take.

Sukoco is a web application designed to help communication between SUNY Korea students, and manage their timetable.

- Ability to create timetable and add courses by their course number
- Ability to view and create the review of professors

b. Scope

This product is to support students to make decisions for which courses they should take. Product will not suggest any information or opinions to users directly. Our product is to give only the public opinion. Our data collection will not be connected to the SBU system to get a grade they got, but all information should be given when they are volunteering.

The product will be running on Chrome Desktop Version with various resolutions.

c. Users

The product's primary users are SUNY Korea students including SBU and FIT, who want to get information about professors and courses, and use and share their timetable. Since our users are college students, we are expecting them to be familiar with the common user interface. Currently, we are limiting our target to SUNY Korea because of our workload, but various users will be using it.

d. User feedback

We got feedback from professors. These are:

1. Indication where the users are very weak.
2. The input of reviews are not explicitly explained.
3. There are possible factors that we can monetize.
4. There is no user feedback when login has failed

For these feedbacks, we noticed that it was the fault of our mockup process. We tried to think and make it look perfect, but we never thought in the way of the new users. Therefore, the change of the mockup was inevitable. For the first feedback, we changed our mockup to indicate what page they are in. Secondly, we changed the text explanation to be more descriptive on the "Write review" page. For the third one, we are trying to think whether the monetization could be a moral hazard, but it is definite that we need a business model.

e. Existing Alternative

There are existing alternative apps which are Everytime, RatemyProfessor, and Campus-pick. The Everytime application handles the timetable and community service, and the RateMyProfessor website shows reviews of the professors. However, when we use timetable service in Everytime, we have to type every single class name, classroom, and instructor manually. Also, there are no reviews of professors from SUNY Korea in the RateMyProfessor. Therefore, we try to make a website that doesn't need to type all the information about the class to add in a timetable. Also, we are going to provide professor reviews for SUNY Korea students.

f. References

1. Our UI Mockup
https://www.figma.com/file/gYfUOC8VJASMSwOnGk7pZf/CSE416_Prototype?node-id=0%3A1&t=AfgzczaCEki2VnD3-1

2. Ph.D kim net to refer to the review system.

https://phdkim.net/

3. Reference for the school community system

https://everytime.kr/

4. Chatting System Reference

https://www.kakaocorp.com/page/service/service/KakaoTalk?lang=en

5. Reference for the rating the professor

https://www.ratemyprofessors.com/

2. Requirements

a. Functional Requirements

Common Interface

- Must Level
    - Function to “Register”
    - Function to “Log in”
    - Function to “Log out”
    - Function to “View the review of professors”
- Should Level
    - Function to “Edit its profile
        - Editing user Id or email should require more authentication.
        - This includes almost every information of its information such as email, id, pwd, and email

Common Users

- Must Level
    - Function to “add courses to their table”
    - Function to “edit or remove courses on their timetable”
    - Function to “leave and remove(edit) a review for professors”
- Should Level
    - Function to “find the course by its information”

Faculty

- Must Level
    - Function to “request to remove the reviews”
        - This is limited to only when students leave an insulting message which is not beneficial to the community.
    - Function to “modify its information”
        - This includes the office hour and office room number of professors

Administrator

- Must Level
    - Function to “remove the reviews”
    - Function to “add courses to the database”

The requirements related to the chatroom are deleted. The main reason for this deletion is because of the lack of time. However, the priority of the chatting system was not essential because the workload required to implement is much higher than any other function because we never learned about the websocket, and the required functions to regulate the chatroom are so many. Profile image is also deleted because the main purpose of profile image is to show their image on the chatting system.

At first, we decided to show the grades and semesters that students took. However, the reviews are anonymous and students should not be identified by their information to the professor. Therefore, we decided to delete students’ grades and semester. We got a feedback from the professor that if there is no information about the written date, students may think those reviews are not useful. So, we sorted reviews in latest order.

b. Non-functional Requirements

1. Users should be able to complete use cases and perform all function requirements in the web application via Chrome.
2. All users' private data should be obtainable and accessible only when necessary.
3. On this web application, all communication should be secured with SSL.
4. This web application should handle at least 200 connections and secure the average load time in 1 second.
5. The web application will be given in English.

c. Use Cases

Use Case	Edit timetable
Primary Actor	Common User
Priority	Essential
Scenario	1. User try edit(add/delete) timetable. 2. User add course(s) using course ID. 3. System checks course ID is valid and provides alert.
Extensions	If the course ID is invalid or does not exist, provide an alert.

Use Case	Edit Profile
Primary Actor	Common User
Priority:	Essential
Scenario:	1. User edit password. 2. Users save the password. 3. System checks if the password is valid or not and if it's valid, provides an alert.
Extensions:	If the password is invalid form, provide an alert to fix the password in the right form.

Use Case	Add courses
Primary Actor	Common Users
Priority	Essential
Scenario	1. User logs in the website 2. User is redirected to the timetable website. 3. User press + button. 4. User types the course number and press add course. 5. The system shows the changed time table.
Extensions	4-1. If the course number doesn't exist, it will not accept and give feedback to the user that course number doesn't exist.

Use Case	Write a review
Primary Actor	Common User
Priority	Essential
Scenario	1. User logs in the website. 2. User presses "Reviews" to go to the review page. 3. User find professor by major and name 4. User presses "write a review" 5. User fills the information requested. 6. User presses the submit button, and the system will redirect to the review page of the professor.
Extensions	5-1. On the recommended level, it will automatically block bad words with some language filters.

d. User Interface

<First draft>

Figure1: Main Page

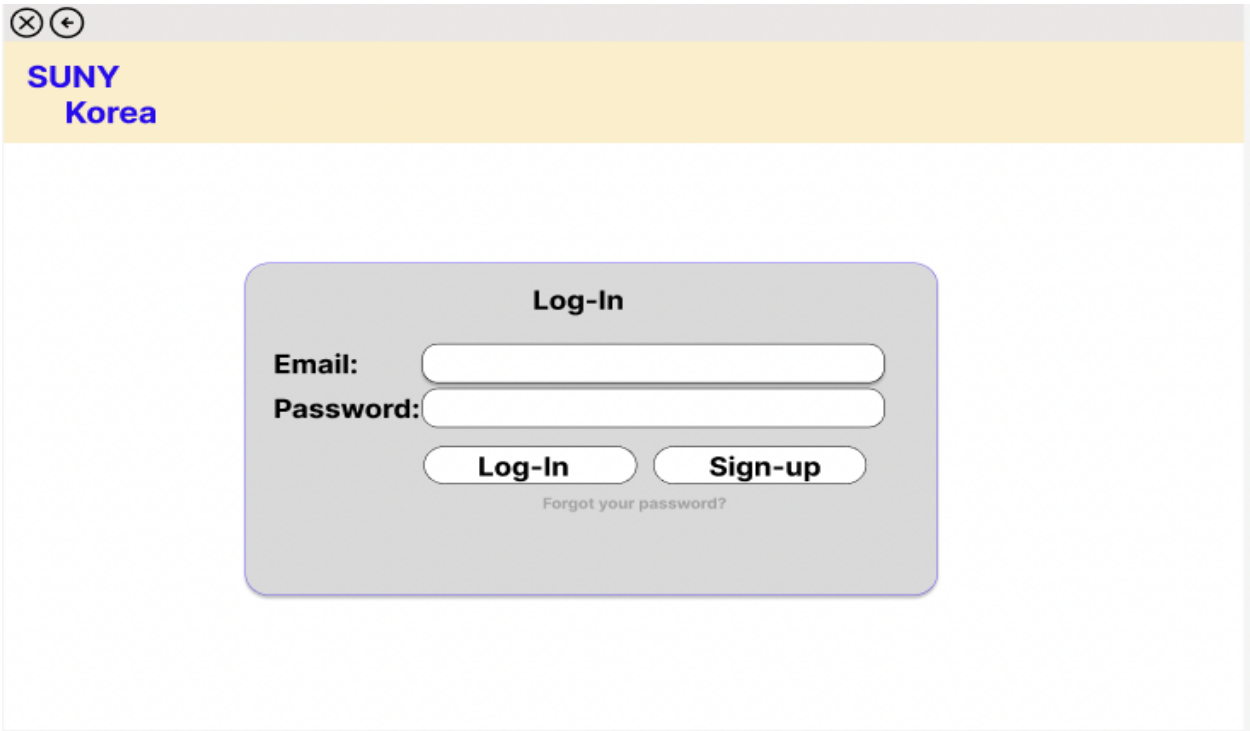


Figure2: Sign-up Page

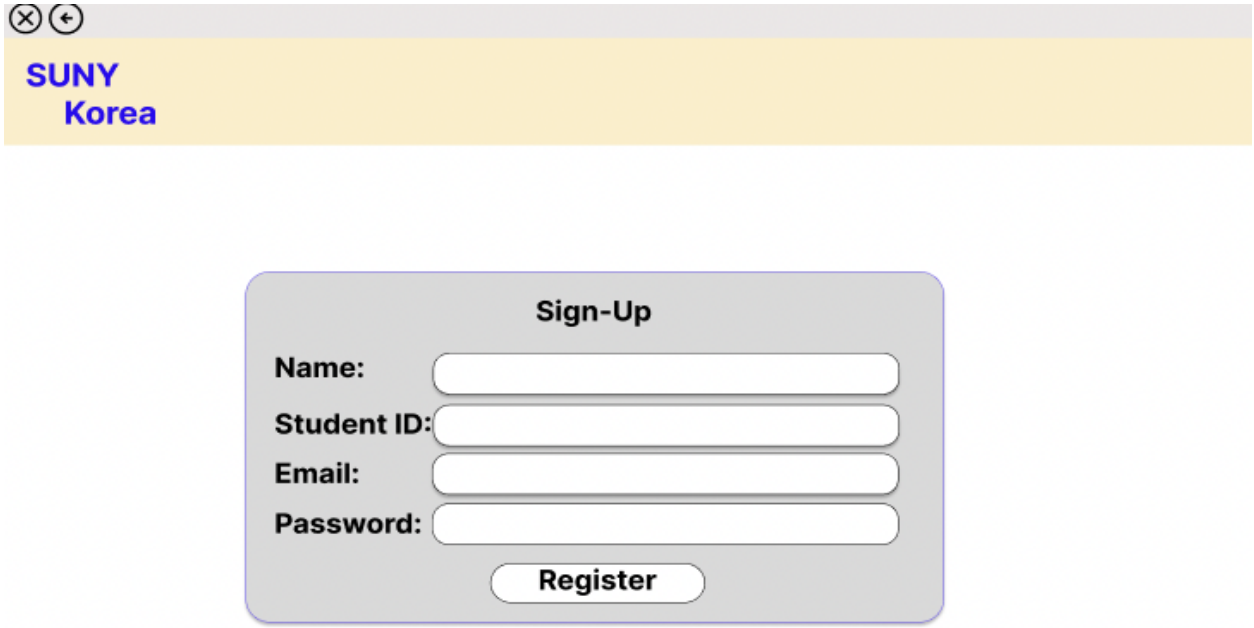


Figure 3: MainPage with login User

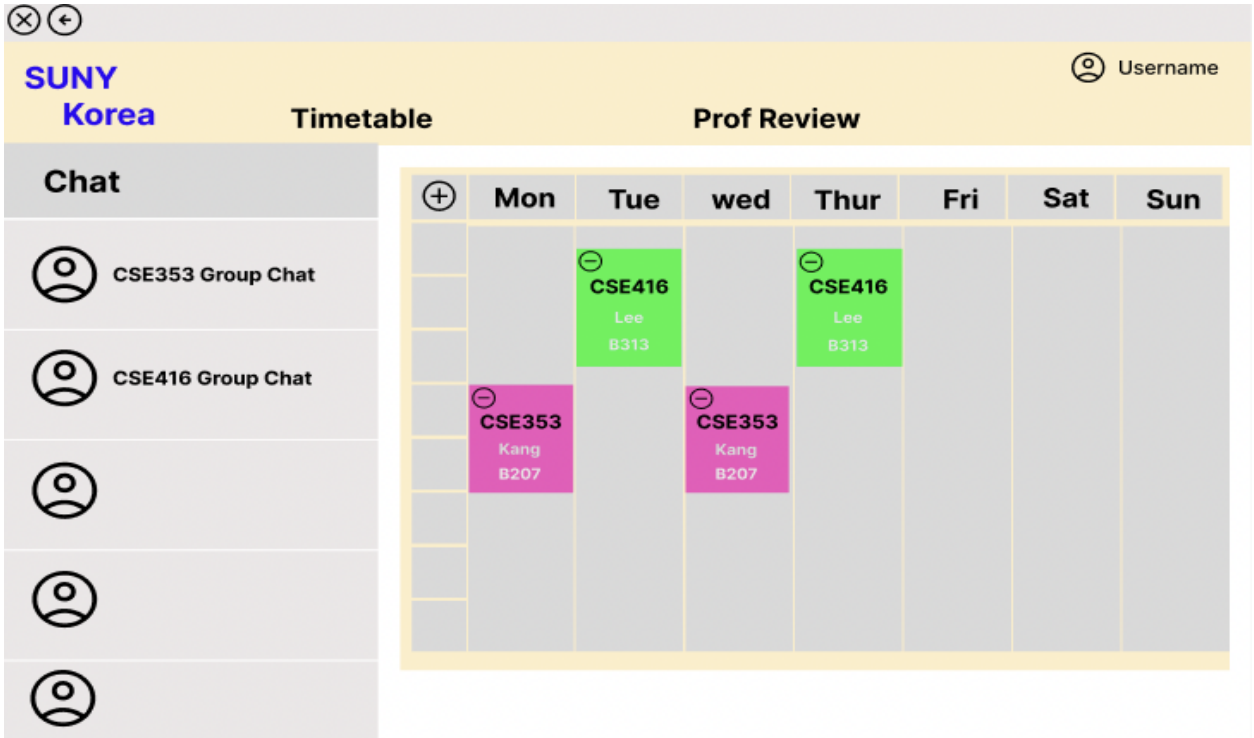


Figure 4: Add Course Page

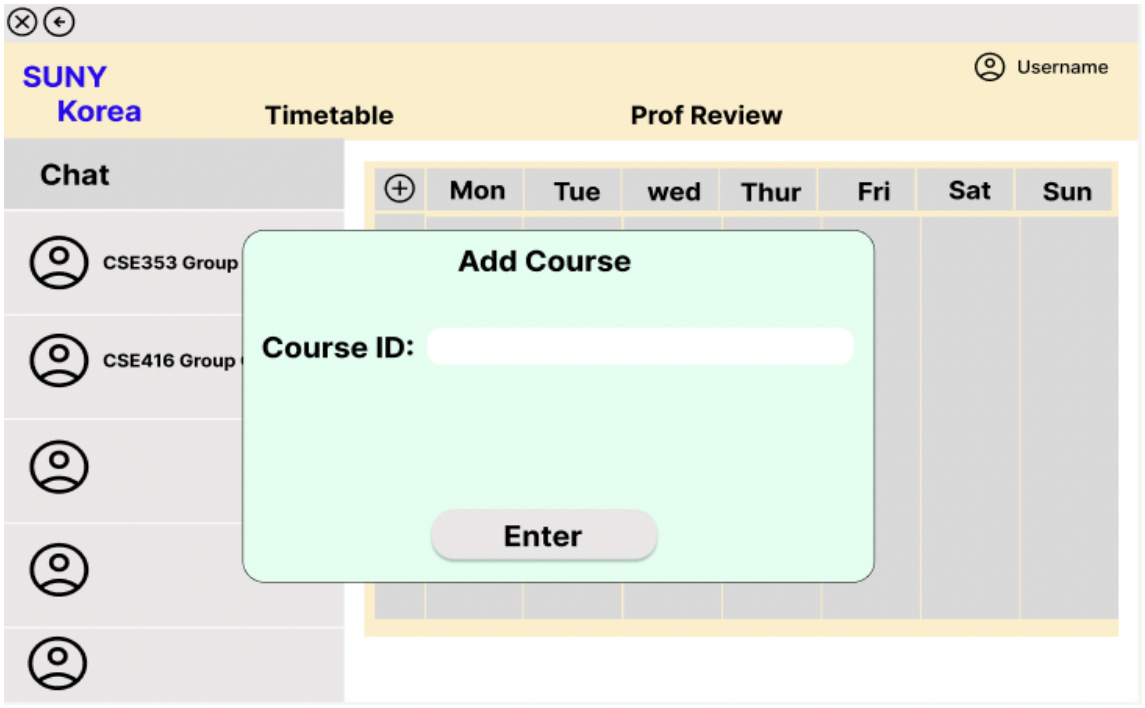


Figure 5: Chatting Page

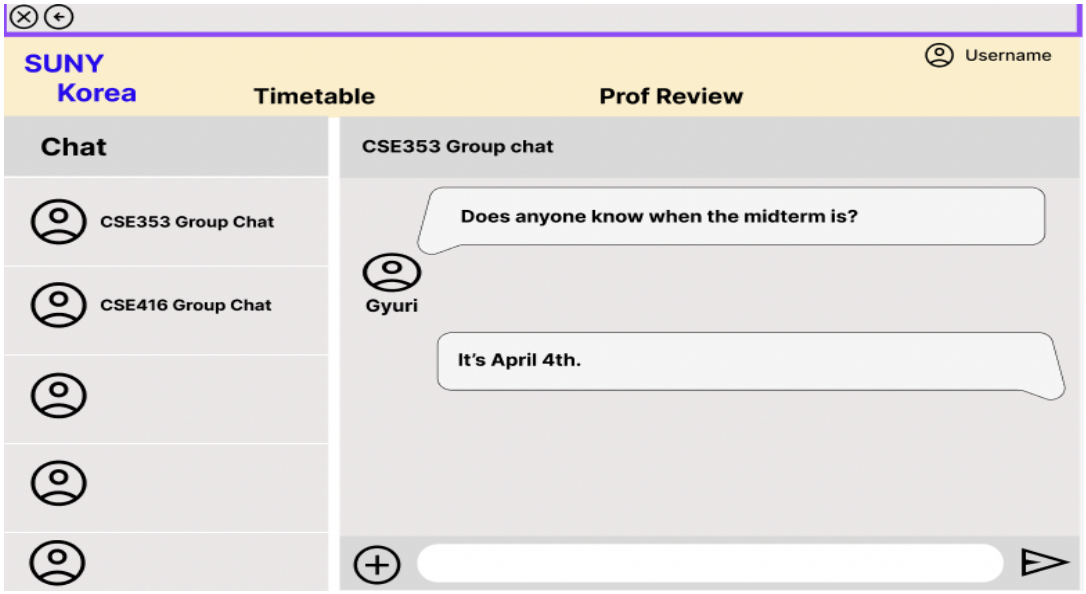


Figure 6: Edit Profile Page

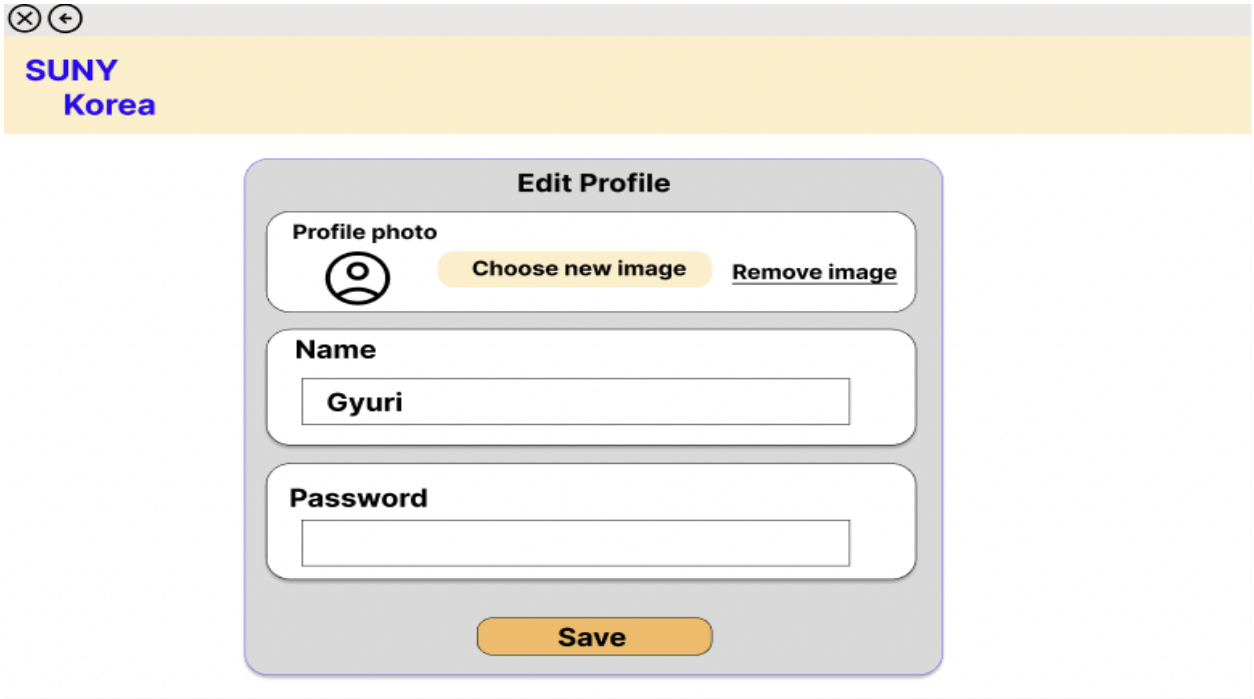


Figure 7: Prof Review Page

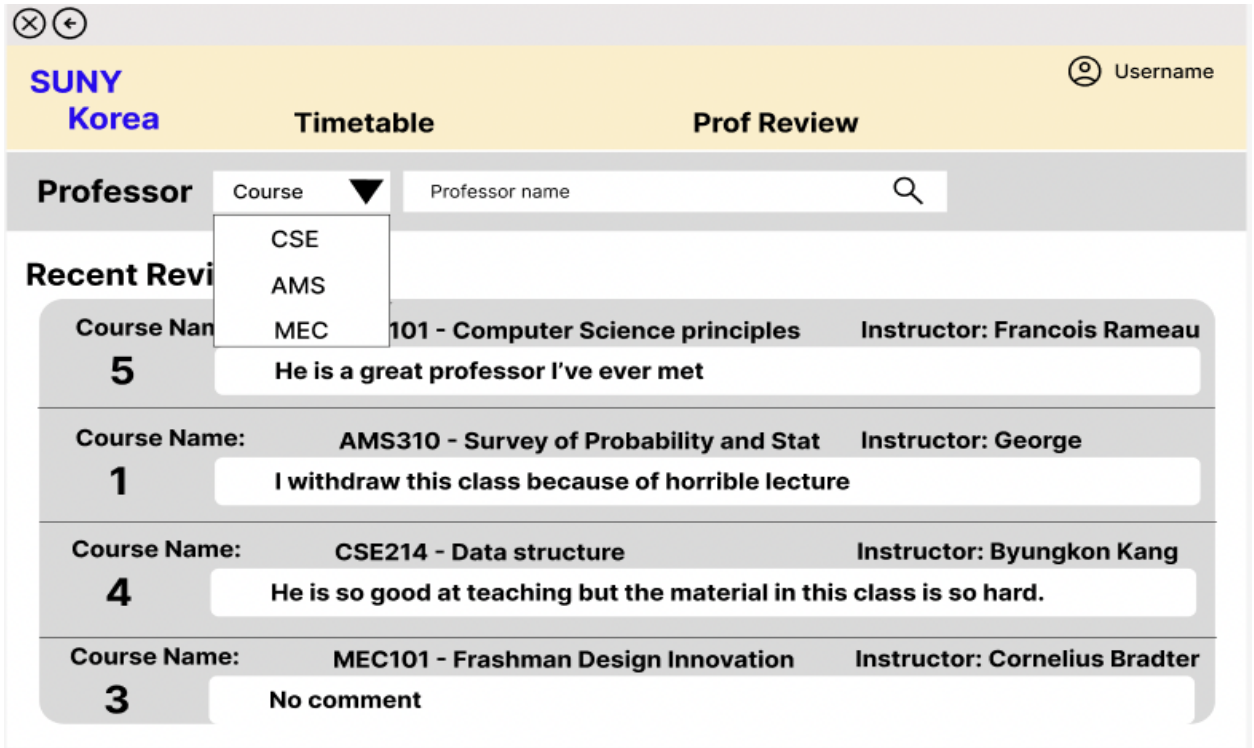


Figure 7.1:

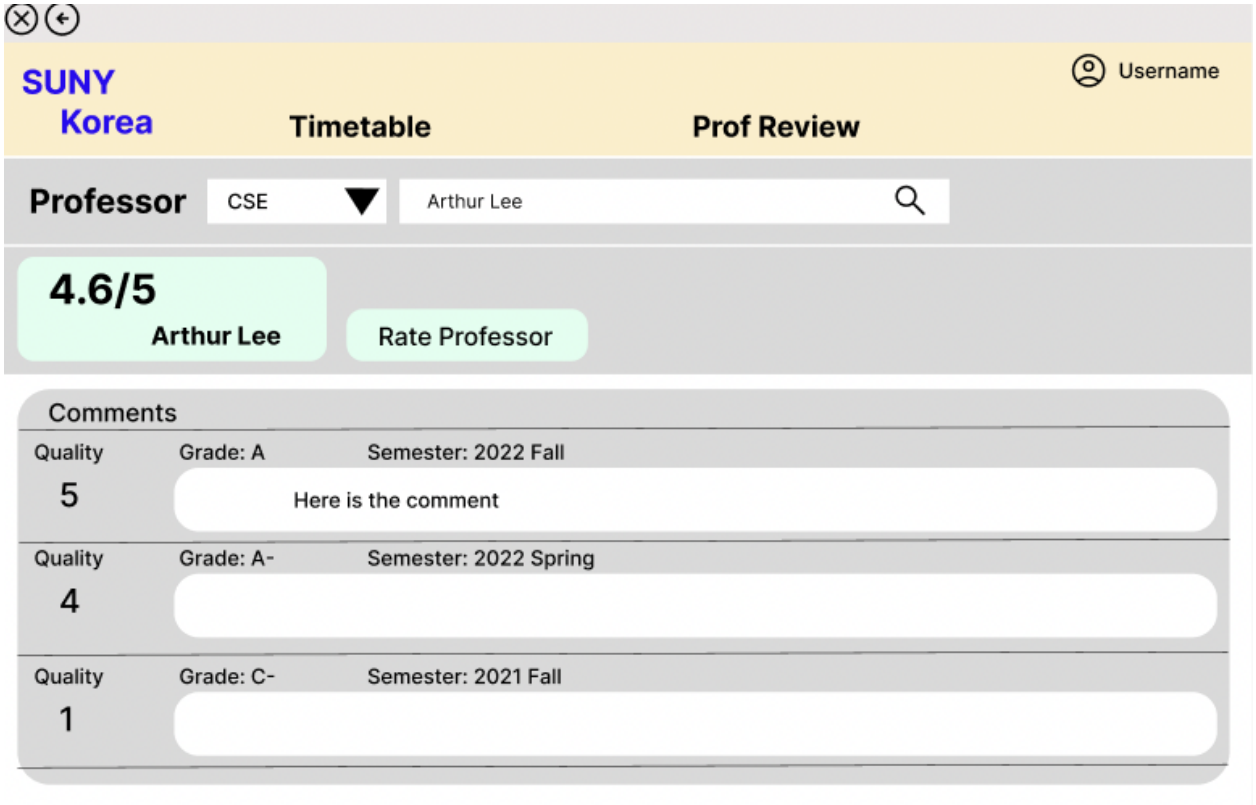
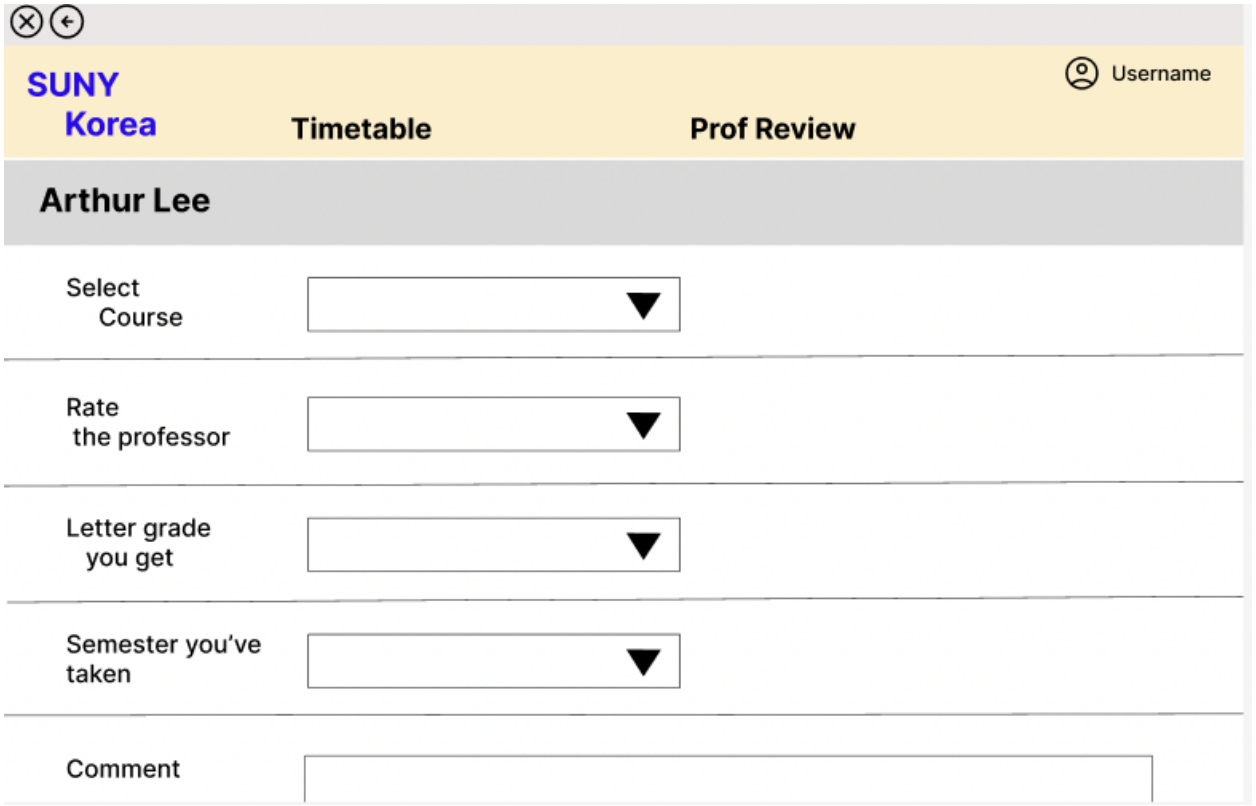


Figure 8: Rate Professor Page



<Final>

Figure1: Main Page

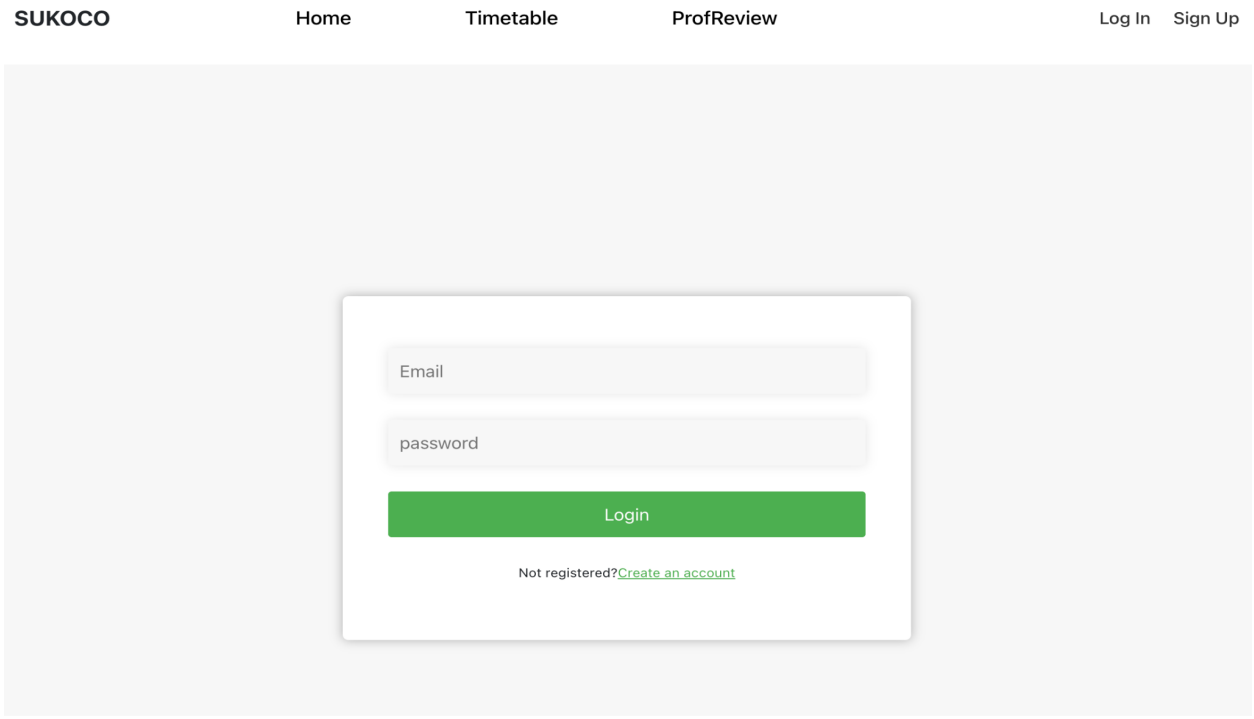


Figure2: Sign-up Page

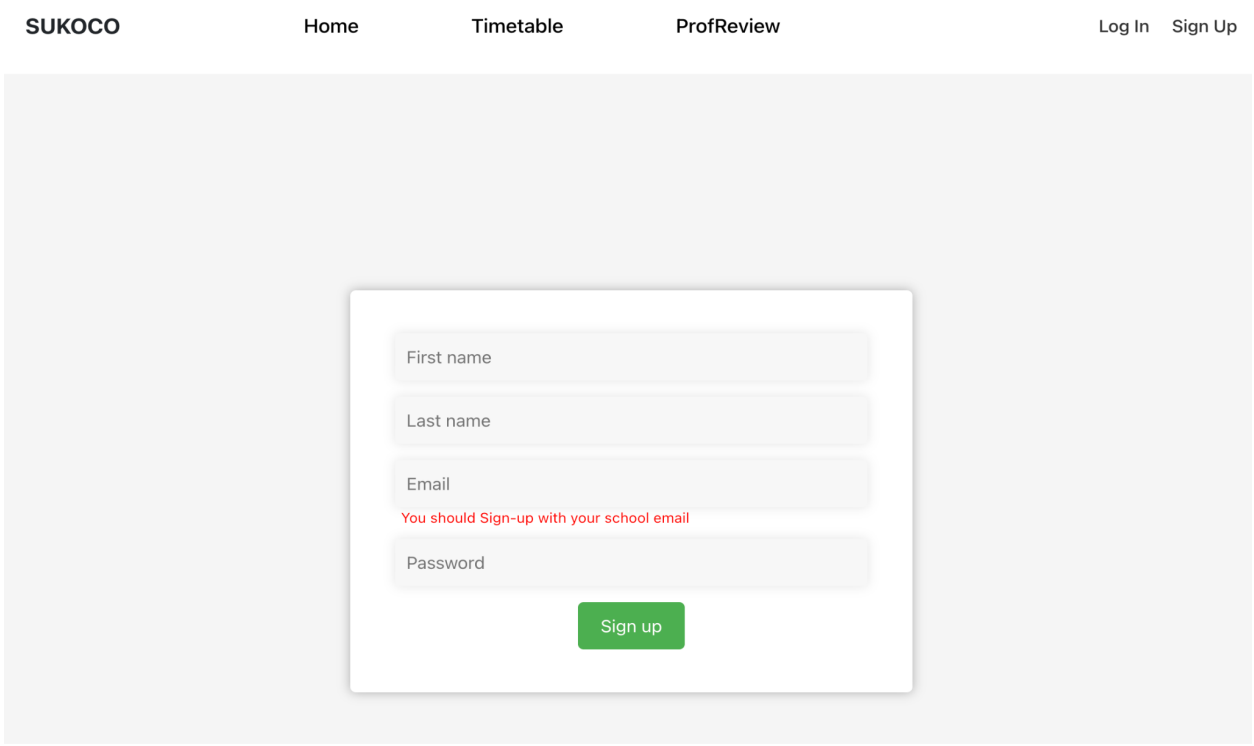


Figure 3: MainPage with login User

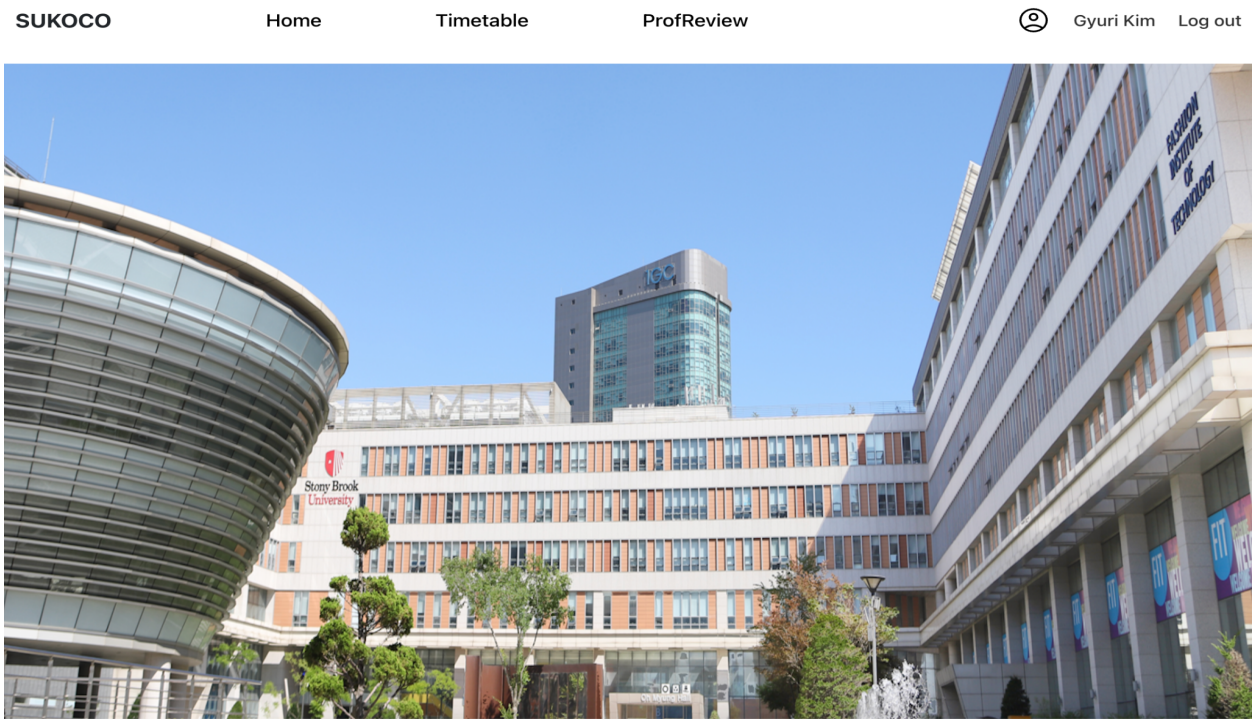


Figure 4: Add Course Page

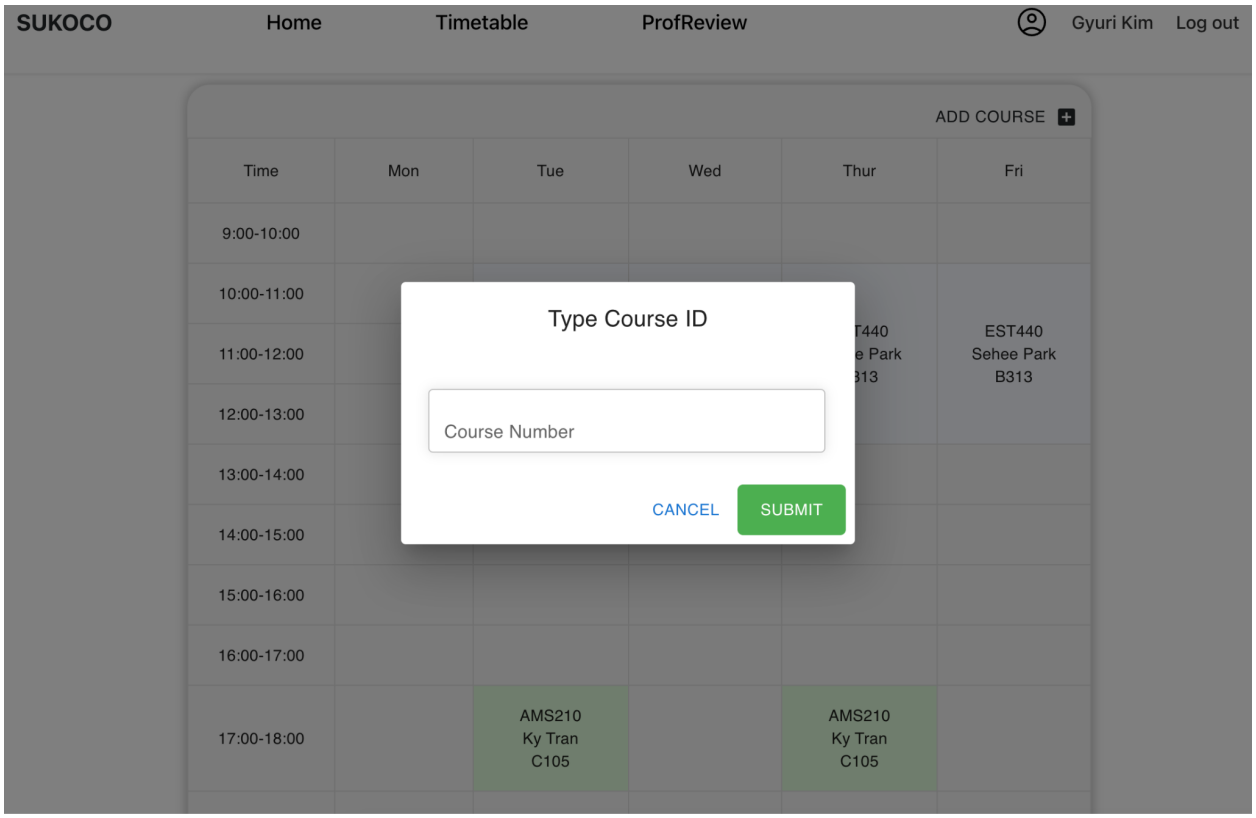


Figure 5: Chatting Page

Dropped this feature

Figure 6: Edit Profile Page

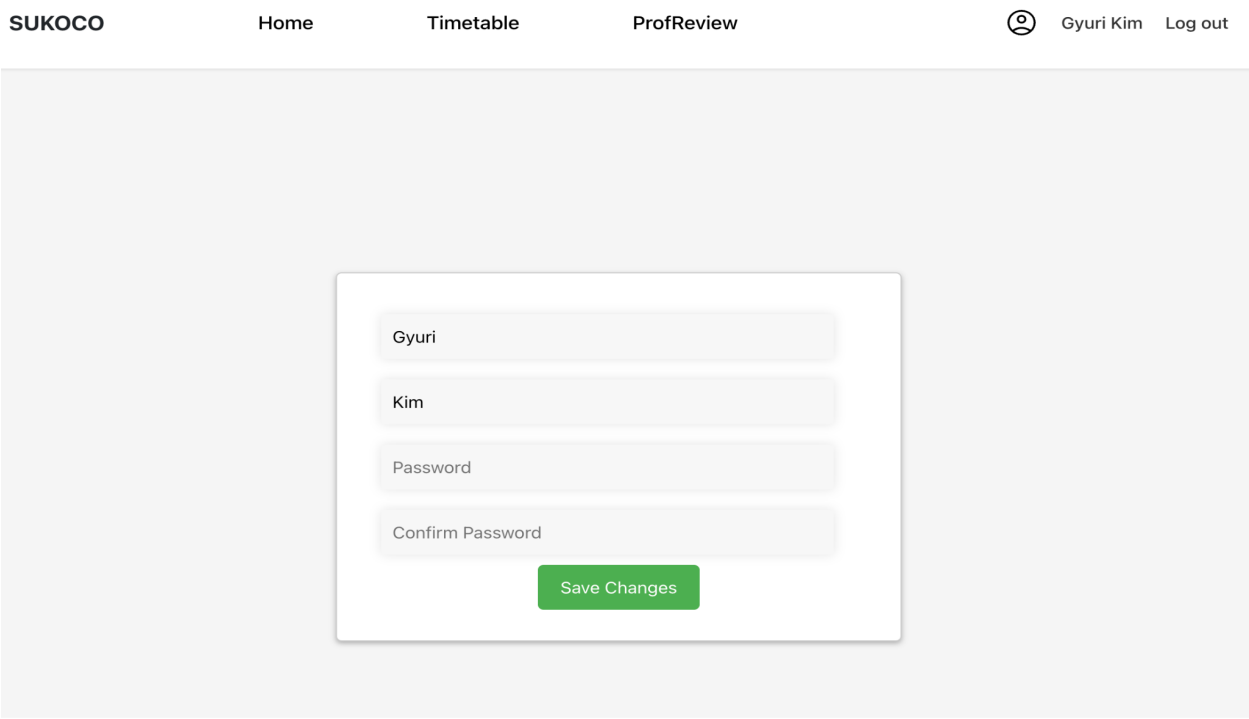


Figure 7: Prof Review Page

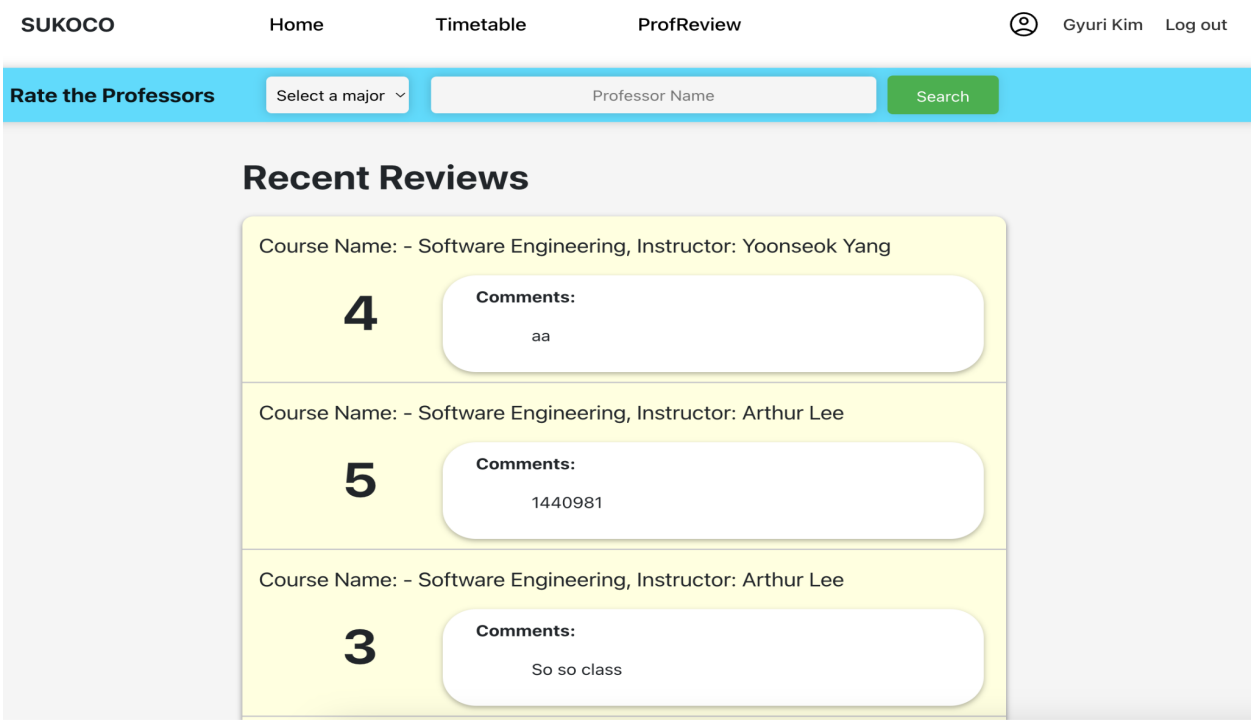


Figure 7.1:

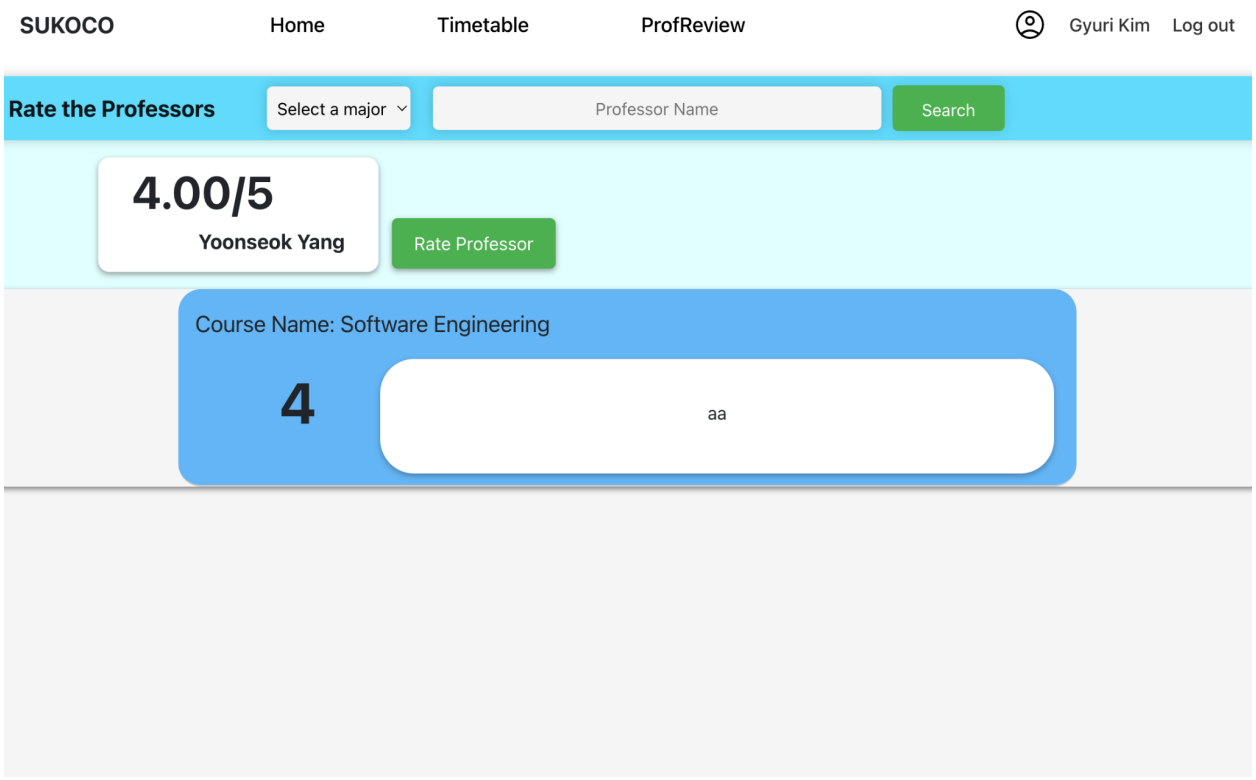
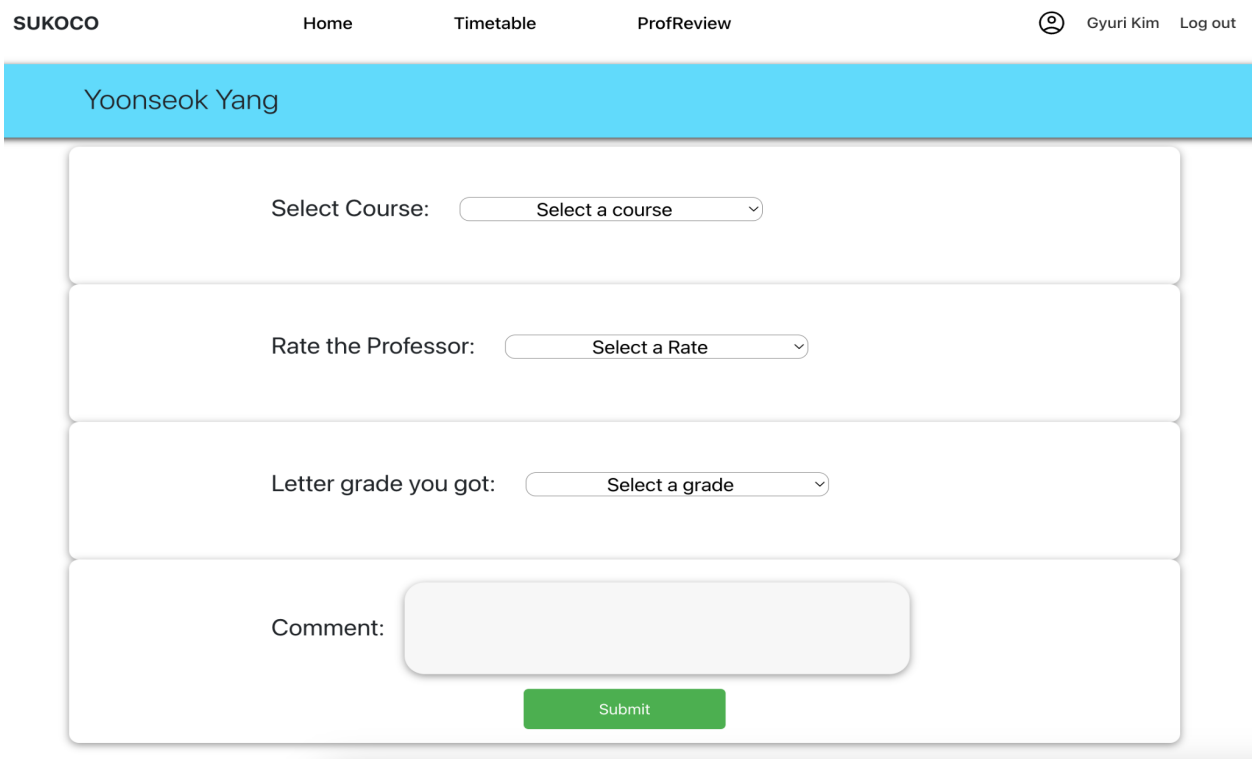


Figure 8: Rate Professor Page



<Admin>

Figure1: Main Page

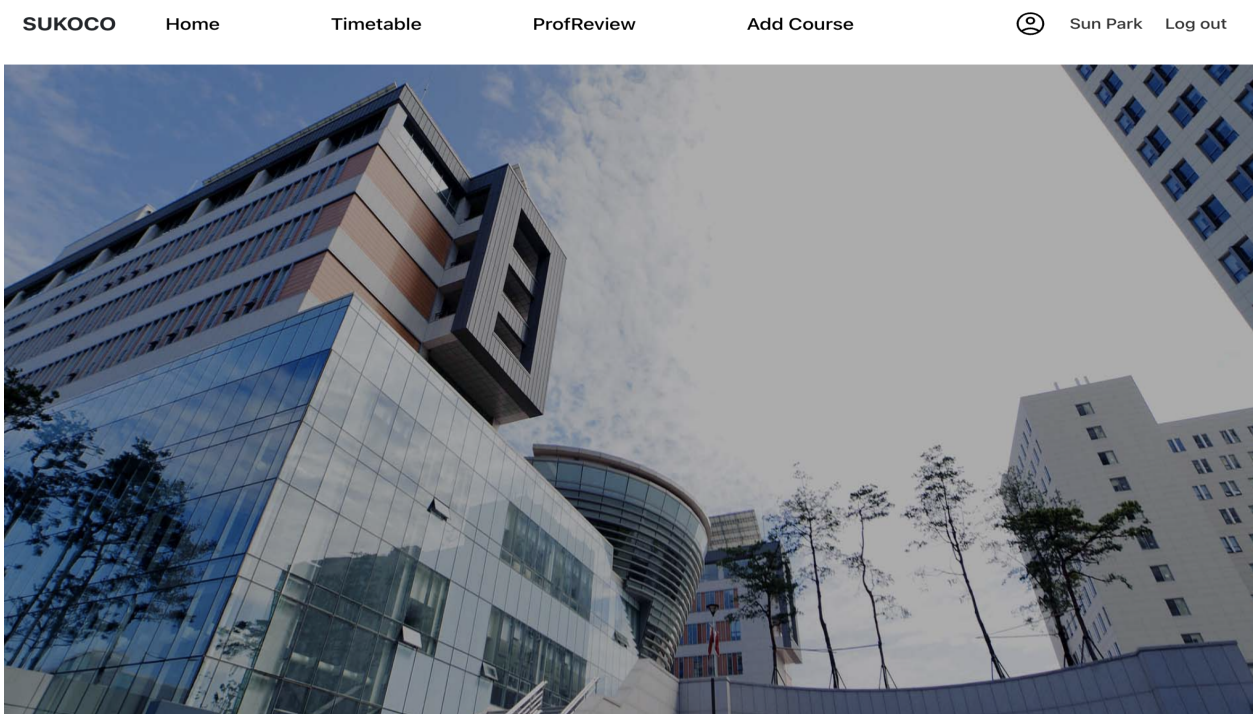


Figure2: Add Courses

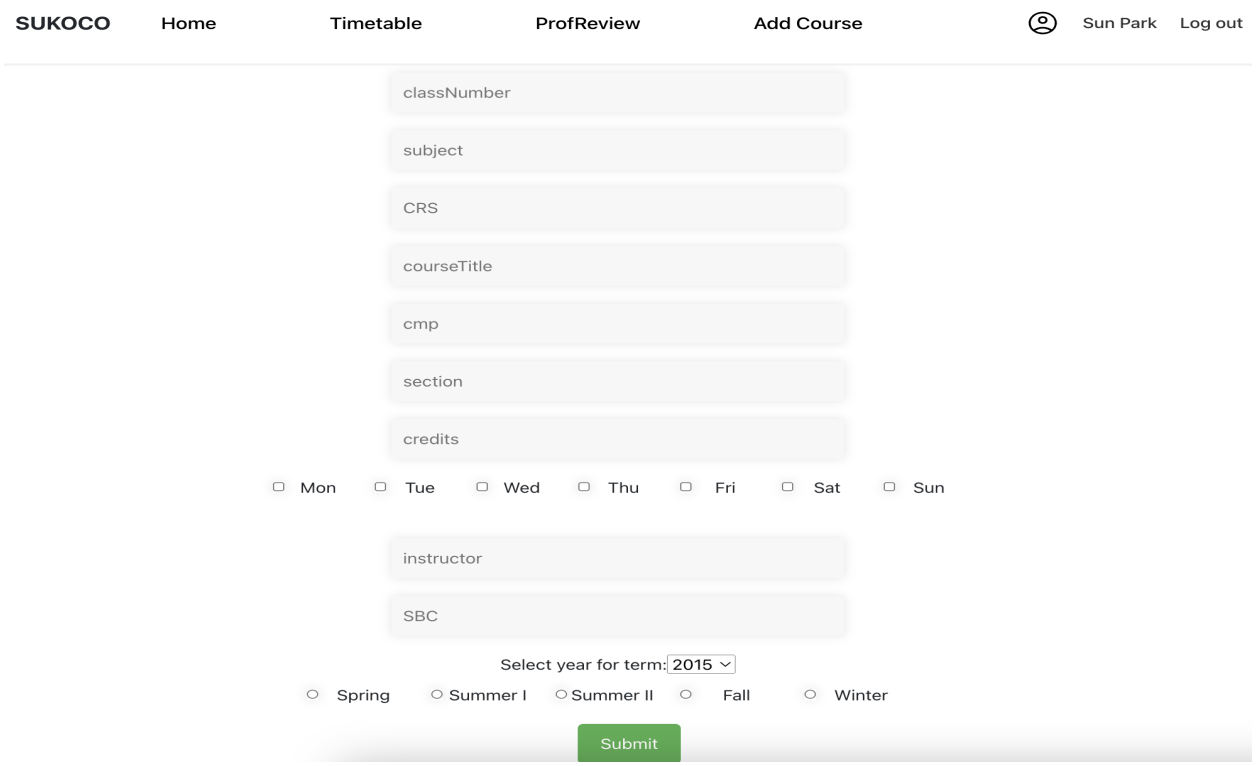
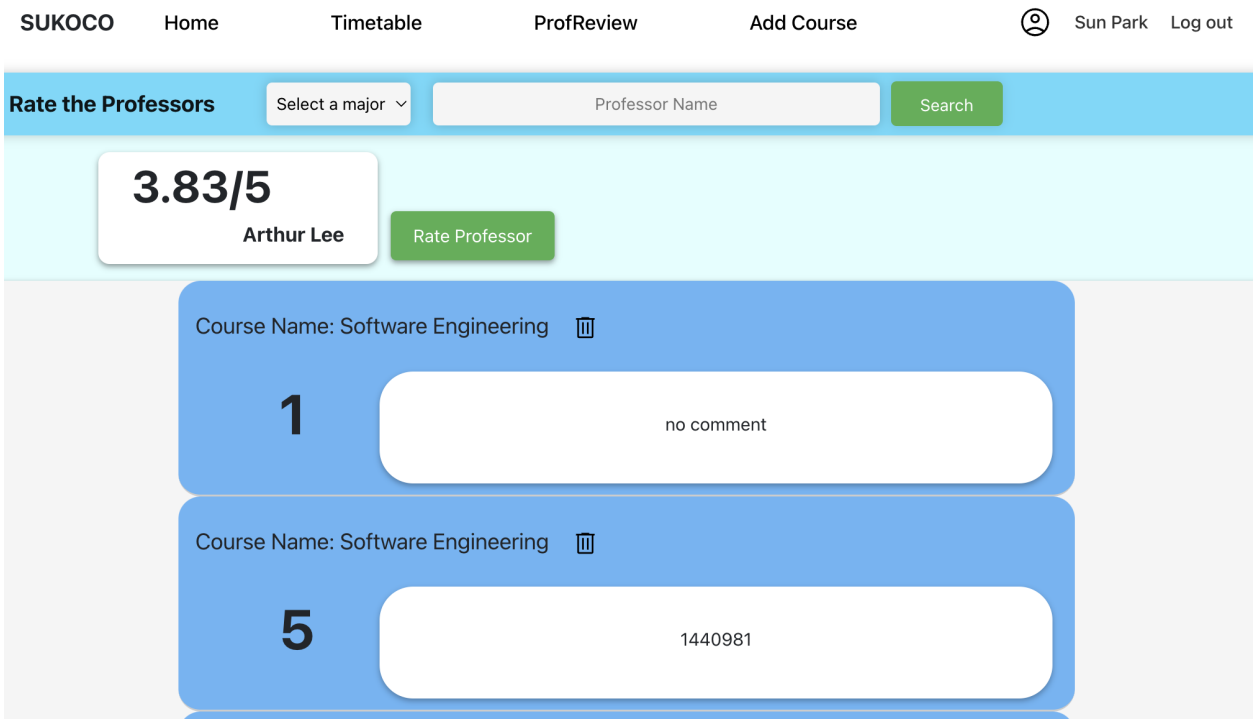


Figure3: Prof rate page



e. Non-functional Requirements

1. Users should be able to complete use cases and perform all function requirements in the web application via Chrome.
2. All users' private data should be obtainable and accessible only when necessary.
3. On this web application, all communication should be secured with SSL.
4. This web application should handle at least 200 connections and secure the average load time in 1 second.
5. The web application will be given in English.

3. Software Design

a. Overview

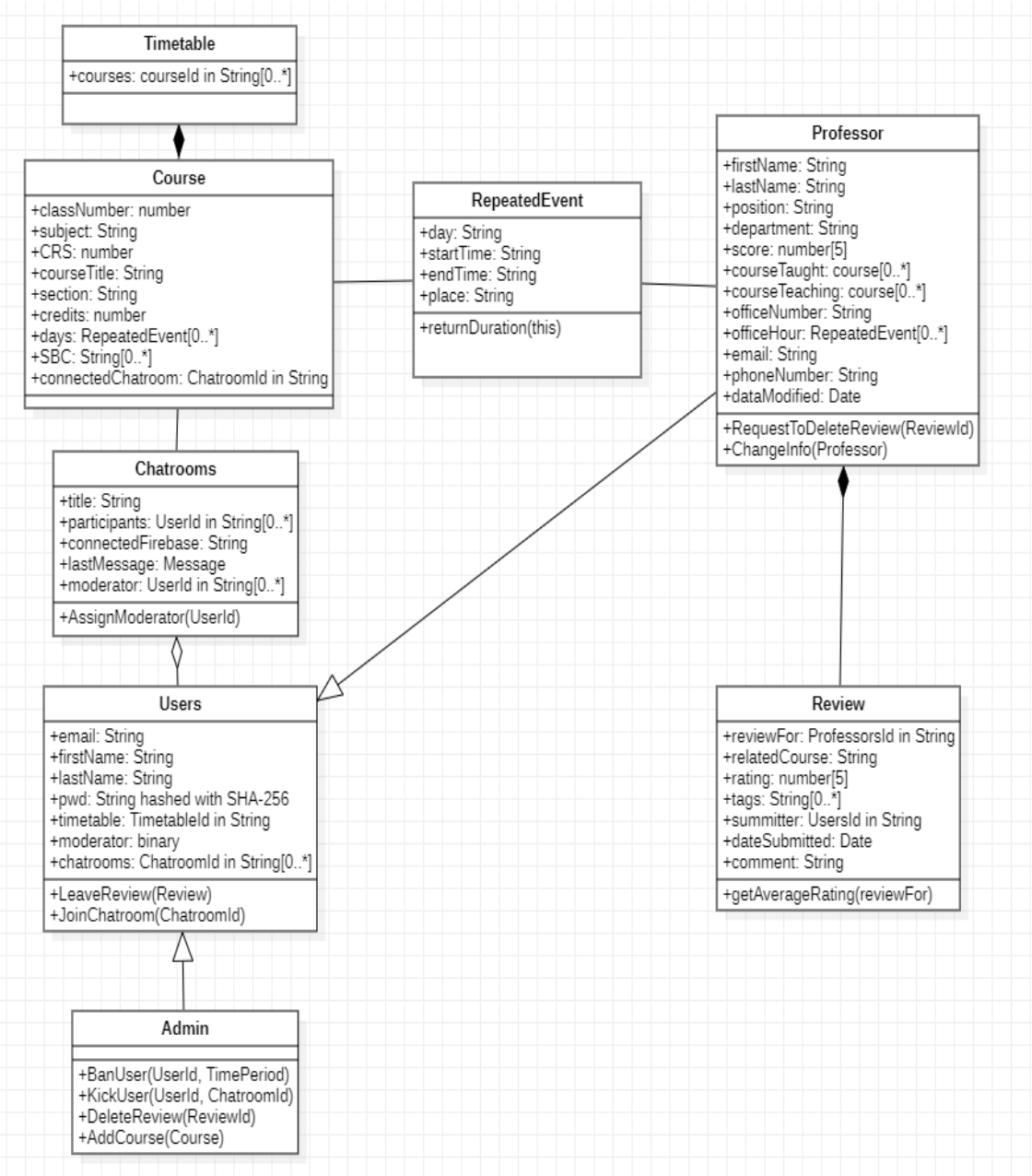
We are using three main tools which are MongoDB, React.js, and Express.js with Typescript, also we are following MVC architecture with no dynamic webpage.

For the front end with React.js, we imported react-dom-router, the uri controller, and navigator, which helps to keep state on uri change, and Axois to adapt API calls with ease, and websocket for real time communication of chatting systems.

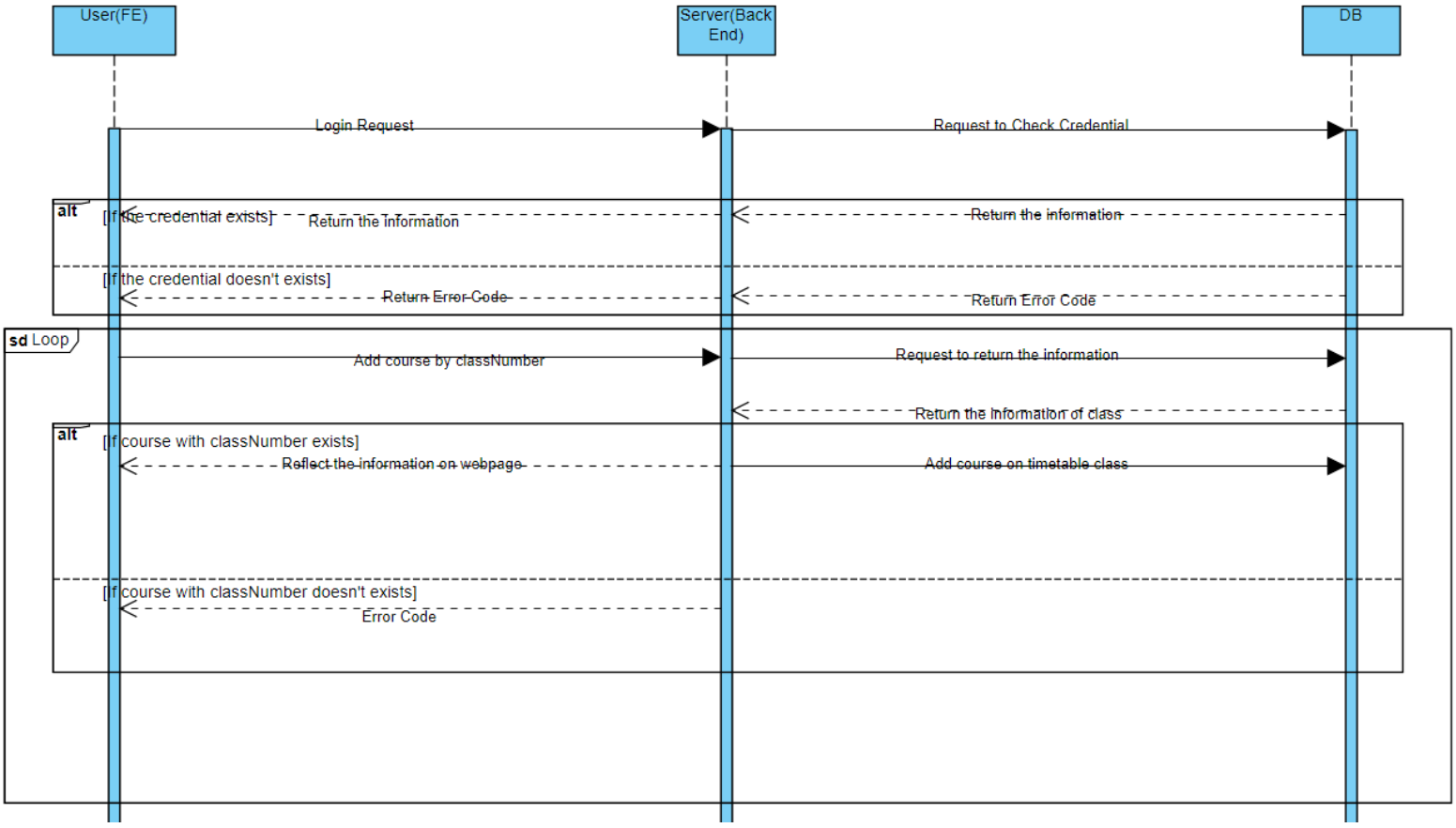
For the back end with Express.js, we used typescript specifically to keep type consistency with all data. For libraries, we used connect-mongo, which helps us stabilize the connection between backend and database, and nodemon for development ease, and websocket for chatting system, and express-session to keep users logged in for a few time although they leave the website.

b. Data Design

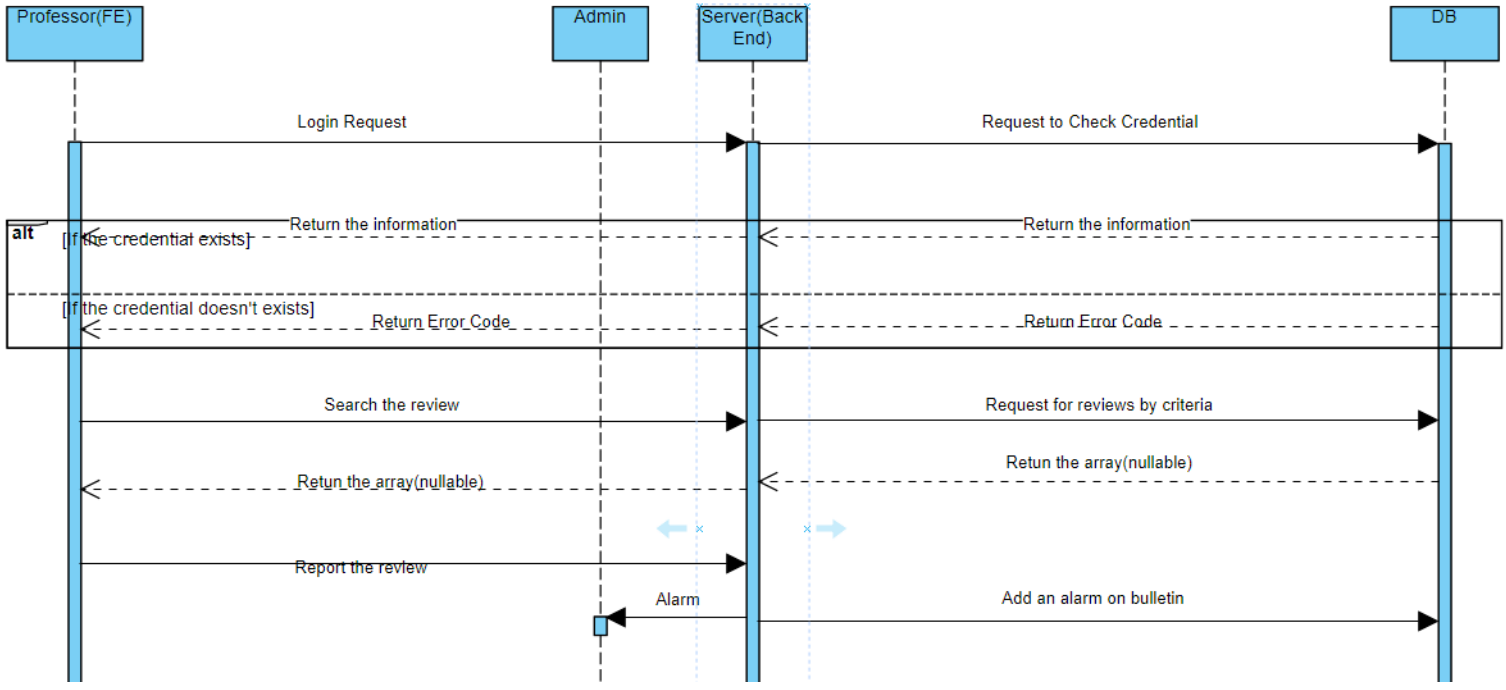
For mongoDB, we make schemas for each collection. Please watch the below image.


Timetable
+courses: courseId in String[0..*]
Course
+classNumber: number
+subject: String
+CRS: number
+courseTitle: String
+section: String
+credits: number
+days: RepeatedEvent[0..*]
+SBC: String[0..*]
+connectedChatroom: ChatroomId in String
RepeatedEvent
+day: String
+startTime: String
+endTime: String
+place: String
+returnDuration(this)
Professor
+firstName: String
+lastName: String
+position: String
+department: String
+score: number[5]
+courseTaught: course[0..*]
+courseTeaching: course[0..*]
+officeNumber: String
+officeHour: RepeatedEvent[0..*]
+email: String
+phoneNumber: String
+dataModified: Date
+RequestToDeleteReview(ReviewId)
+ChangeInfo(Professor)
Chatrooms
+title: String
+participants: UserId in String[0..*]
+connectedFirebase: String
+lastMessage: Message
+moderator: UserId in String[0..*]
+AssignModerator(UserId)
Users
+email: String
+firstName: String
+lastName: String
+pwd: String hashed with SHA-256
+timetable: TimetableId in String
+moderator: binary
+chatrooms: ChatroomId in String[0..*]
+LeaveReview(Review)
+JoinChatroom(ChatroomId)
Review
+reviewFor: ProfessorsId in String
+relatedCourse: String
+rating: number[5]
+tags: String[0..*]
+summitter: UsersId in String
+dateSubmitted: Date
+comment: String
+getAverageRating(reviewFor)
Admin
+BanUser(UserId, TimePeriod)
+KickUser(UserId, ChatroomId)
+DeleteReview(ReviewId)
+AddCourse(Course)

c. UML Sequence



This is the UML Sequence diagram for the instance that users login and add the course in timetable.



This is the UML Sequence diagram for the instance that Professor logs in and reports the review.

d. API Design

For RESTful APIs, we set a few uri to communicate with the front and back end. These are all tables that show the intended functionality.

	GET	POST	PUT	Delete
/review	Retrieve all reviews.	Create a new review. User status should be valid(not banned)	Bulk update of reviews. Authorization needed	Remove all reviews. Authorization needed
/review/1	Retrieve the details for review 1(Class Number)	Error	Update the details of review 1 if it exists	Remove review .Authorization needed.(Whether original writer or admin)
/review/find	Retrieve the reviews by the information of req.body.review	Error	Error	Error
/review/report	Error	Create an alarm to	Error	Error

/course

	GET	POST	PUT	Delete
/course	Retrieve all courses.	Create a new course. Authorization needed	Error	Error

/course/1	Retrieve the details for course 1(Class Number)	Error	Error	Error

/user

	GET	POST	PUT	Delete
/user/login	Error	Get the information of the user based on the credential, and also session information	error	error
/user/register	error	Register the user based on the credential	error	error
/user/logout	error	error	error	Delete the session information in DB.
/user/editProfile/:email	error	error	Edit the profile based on request	

/timetable

	GET	POST	PUT	Delete
/timetable/:email	Get a related courses based on the email given	error	error	error
/timetable/add/:email	error	error	Add a class number in the timetable	error
/timetable/delete/:email	error	error	Delete the designated class number in the timetable	error

e. Deployment

We use Netlify to deploy Frontend and Heroku to deploy Backend.

Frontend deployment procedure using Netlify:

1. Connect your project to a Git repository: Netlify integrates well with popular version control systems like Git. Connect your frontend project to a Git repository hosting service like GitHub.
2. Create a new site on Netlify: Once you're logged in to Netlify, click on the "New site from Git" button on the dashboard. Choose the Git provider where your project's repository is hosted and authorize Netlify to access it. Select the repository and branch you want to deploy.
3. Configure build settings: Netlify needs to know how to build your frontend project. In the build settings, specify the build command, which is typically something like npm run build or yarn build, depending on your project's setup. Set the publish directory to the folder where the built files will be generated, often called dist or build.
4. Monitor the deployment: Netlify will initiate the build process and deploy your frontend project. You can monitor the progress in real-time on the deploy logs page. Once the deployment is complete, Netlify provides a unique URL where your site is hosted.
5. Test and access your deployed frontend: Visit the URL provided by Netlify to see your deployed frontend project. Test all the functionality to ensure everything is working as expected.

Backend deployment procedure using Heroku:

1. Compile and Upload on git: Our backend server is written on typescript and tsc, which is typescript compiler for testing purposes doesn’t fit into the environment of the actual server. We should compile it into javascript and push the repository.
2. Make a server on the cloud of Heroku: Heroku supports making an individual server. This server automatically connects to the url, but needs to change the port

into 80.

3. Connect Github Repository: Connect the Github specific branch of the repository. This will automatically upload the file in the branch and run the instruction in package.json.
4. Make it an automatic deployment: Furthermore, we can connect the Github branch, and we can make it build a server when I push any git on the branch

5. Individual Progress Update

a. Gyuri:

Milestone 1 - Complete the log-in, register, and change the profile info page. There was no hard time to implement those functions. Those functions were not connected to the database at that time (milestone1). However, we connected to the database now.

Milestone 2, 3 - Original plan was completing the timetable but there are dependencies, so instead of doing timetable, I completed Rate the professor page. Sungwoong is now working on the backend to connect with the database.

Milestone 4, 5, 6 - Original plan was working on the chatting system but since we decided to remove the chatting system, I'm working on the timetable page. I had a hard time implementing the timetable, so I looked for the references and I found a good resource to follow. Therefore, I followed the resource and changed some parts to be fit in our project.

b. Sungwoong:

Milestone 1: This week is mainly about learning stuff and defining what to do. I learned about the Typescript and made class schemas.. Basic routings are created to let Gyuri know which route should be called for each instance. I also learned about https encryption with OpenSSL.

Milestone 2 and 3: I wasn't able to work on because of health problems.

Milestone 4: Https encryption has been implemented for connection between front and back end. Also, I prototyped API calls to remove dependencies on Gyuri's work. Session login is completed on the backend only, and login and register is done.

Milestone 5: Login with Session and Register frontend and backend implementation, profile page integration, Technically, I worked to change front mockup to actual connected app with backend.

Milestone 6: Currently, I am working on the rate of the professor for both front and back end. It is expected to be done by Sunday, and the system test will be on Monday.

6. Group Progress Update:

We were running behind schedule. For functional requirements, we met the requirements for login and register. Timetable and rate the professor features are on the stage of integration. Chatting system is to be abandoned because of practical obstacles.

a. Reason explained why Chatting System is Abandoned

The first reason is that the schedule has already been delayed. As a result, this means we have limited time and resources available for this project. By prioritizing many features, the chat system has been abandoned for some reason.

The reason that the chat system is not prioritized is that there are already well-made functions to adapt. Practically, it is hard to make a better interface and provide a better user experience than other products. Therefore, we will use Kakaotalk open chat system. Therefore, the url of the open chat room will be provided instead of the chat system.

b. Reason of delay

The reason for this delay is because of improper measurement of learning steps and no work distribution of integration. In the early stage of the project, we were thinking about using Typescript instead of Javascript. The pros of Typescript are error free from type error. However, I failed to estimate the difficulty of Typescript. It was not only about learning new stuff, but lack of documentation compared to Javascript and spending more time defining types on different libraries manually.

a. Extra reason: Before actual development, we had discussed the work distribution. It was mainly about dividing the frontend and backend, but nothing about integration.

7. Links

Presentation:

https://docs.google.com/presentation/d/1i3Em00uWkUmOltYHTZzAIdgs5c-ExZvHfFYDOzZONZQ/edit?usp=sharing

Deploy: https://cse416-sukoco.netlify.app/