Sukoco: Software Design Document

Team members

Gyuri Kim

Sungwoong Park

Table of Contents

1. Introduction

1.1. 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. For example, I was not able to know which courses by which professors are better because of this. This inconvenience disturbed me to choose classes to take.

So we are introducing Sukoco. Sukoco is a web application designed to help communication between SUNY Korea students, and manage their timetable. We mainly serve three functions

- Ability to create timetable and add courses by their course number
- Ability to view and create the review of professors
- Ability to chat with classmates through chatting service.

1.2. 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 and a range of devices will be smartphone(Android and iOS), and desktop(Chrome).

1.3. Users

The product's primary users are SUNY Korea students including SBU and FIT, who want to get information about professors and courses. 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.

1.4. 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.

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 "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.

1.5 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 don't need to type all the information about the class to add timetable. Also, we are going to provide professor review for SUNY Korea students.

1.6 Definition

No special word.

1.7 References

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

https://phdkim.net/

https://everytime.kr/

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

https://www.ratemyprofessors.com/

2. Requirements

2.1 Functional Requirements

Common Interface
- Must Level
    - Function to "Sign-up"
    - 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 process.
        - 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”
        - This automatically adds the designated chatroom
    - Function to “edit or remove courses on their timetable”
        - ~~This automatically edit or remove the designated chatroom~~
    - ~~Function to “message and view of the chatroom”~~
- Should Level
    - Function to “find the course by its information”

Faculty Moderator

- 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.
        - Too many frequent and unreasonable requests will cause a ban from this system, but they should be able to contact us with inconvenient way.
    - 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 “ban users for short period or permanently”
    - Function to “add courses to the database”
    - ~~Function to “moderate the chat rooms” such as muting~~
        - ~~This function will be given to Student Moderator in further development.~~
- ~~Should Level~~
    - ~~Function to “assign common user as Student Moderator”~~

2.2. 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. System Architecture

3.1 Overview

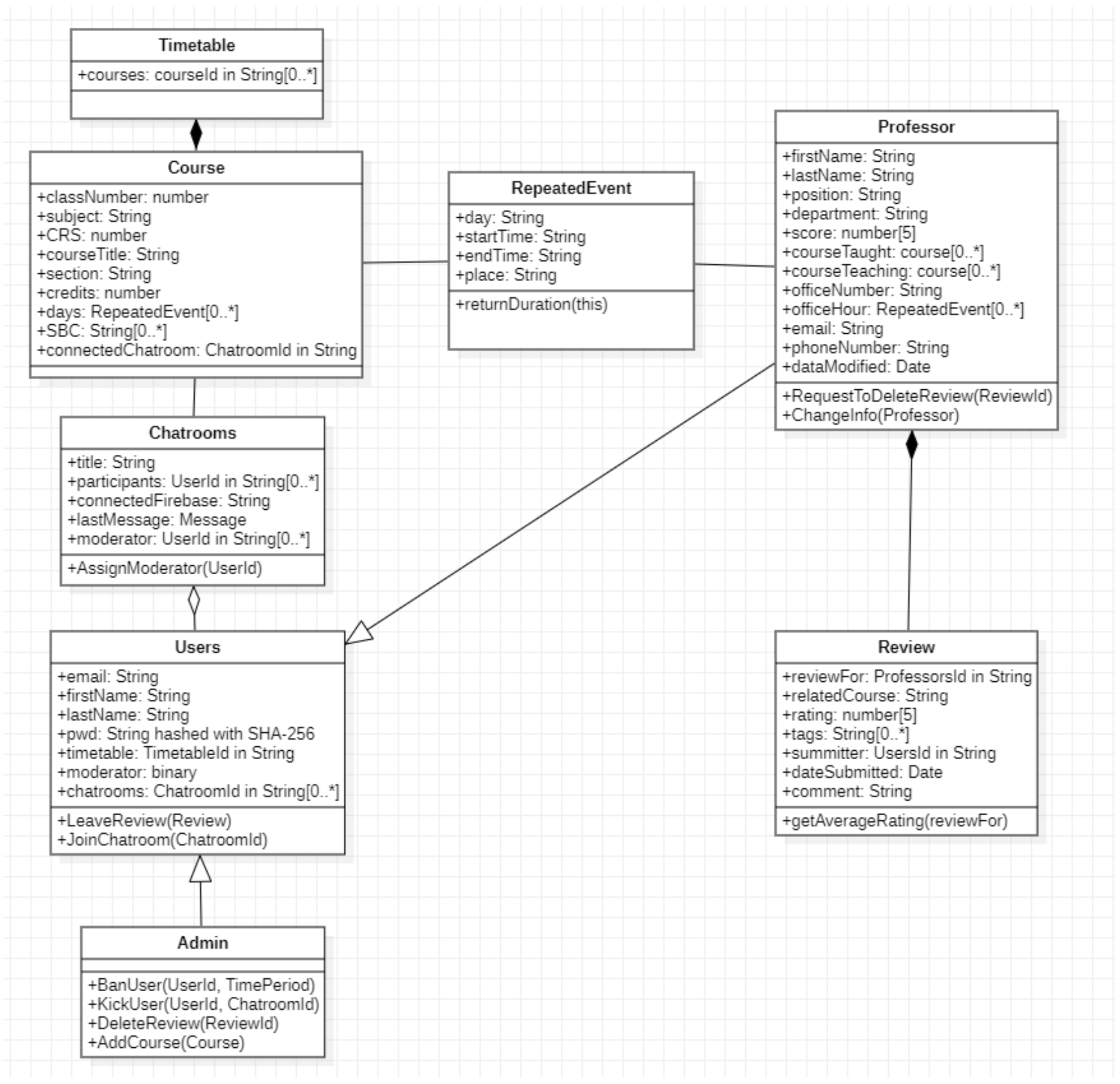
We are using three main tools which are MongoDB, React.js, and Express.js with Typescript, also we are following MVC architecture with no dynamic webpage.

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.

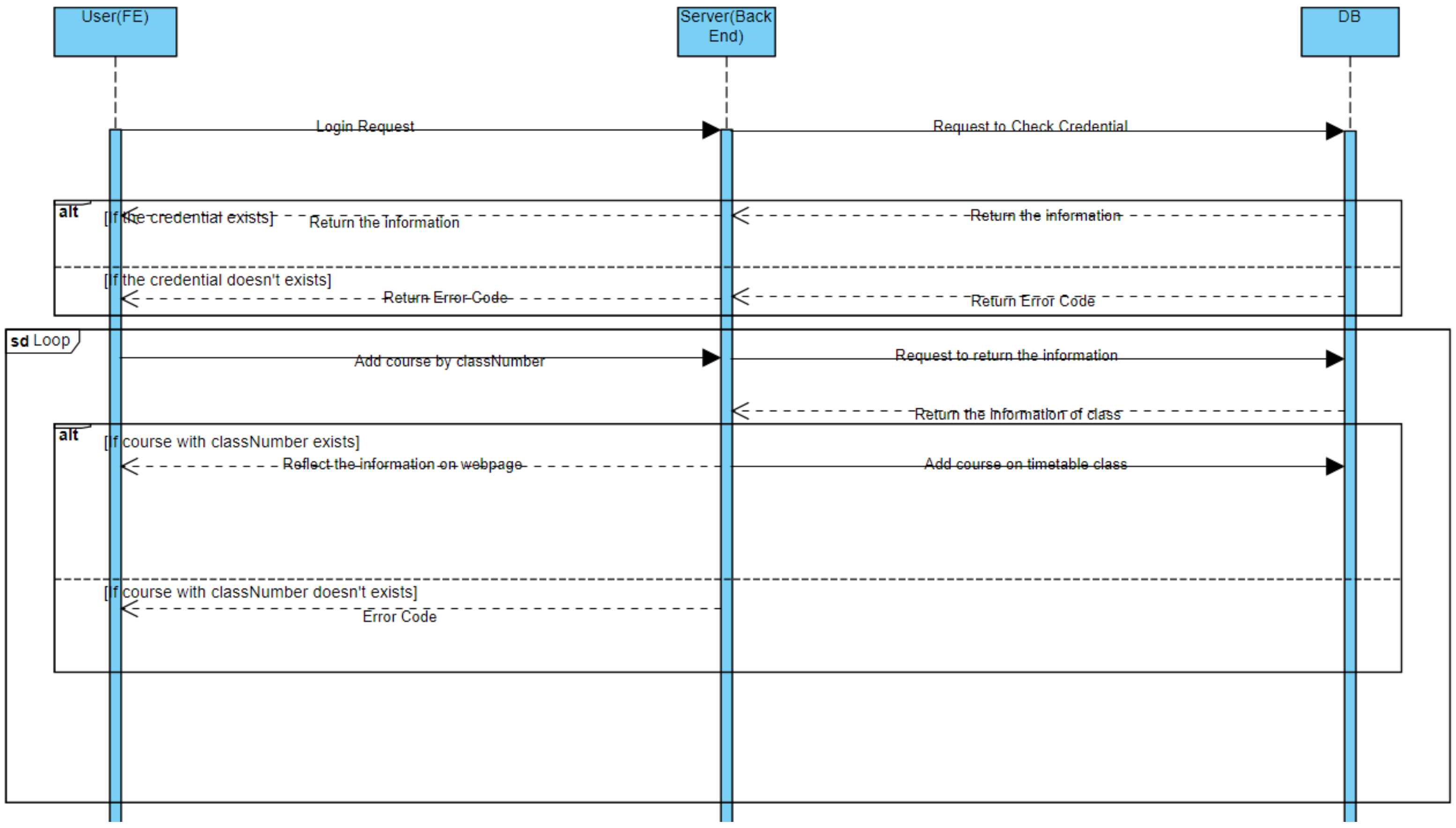
For the back end with Express.js, we used typescript specifically to keep type consistency with all data. For libraries, we used connect-mongo, which helps us stabilize the connection between backend and database, and nodemon for development ease, and websocket for chatting system, and express-session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

3.2 Data Design

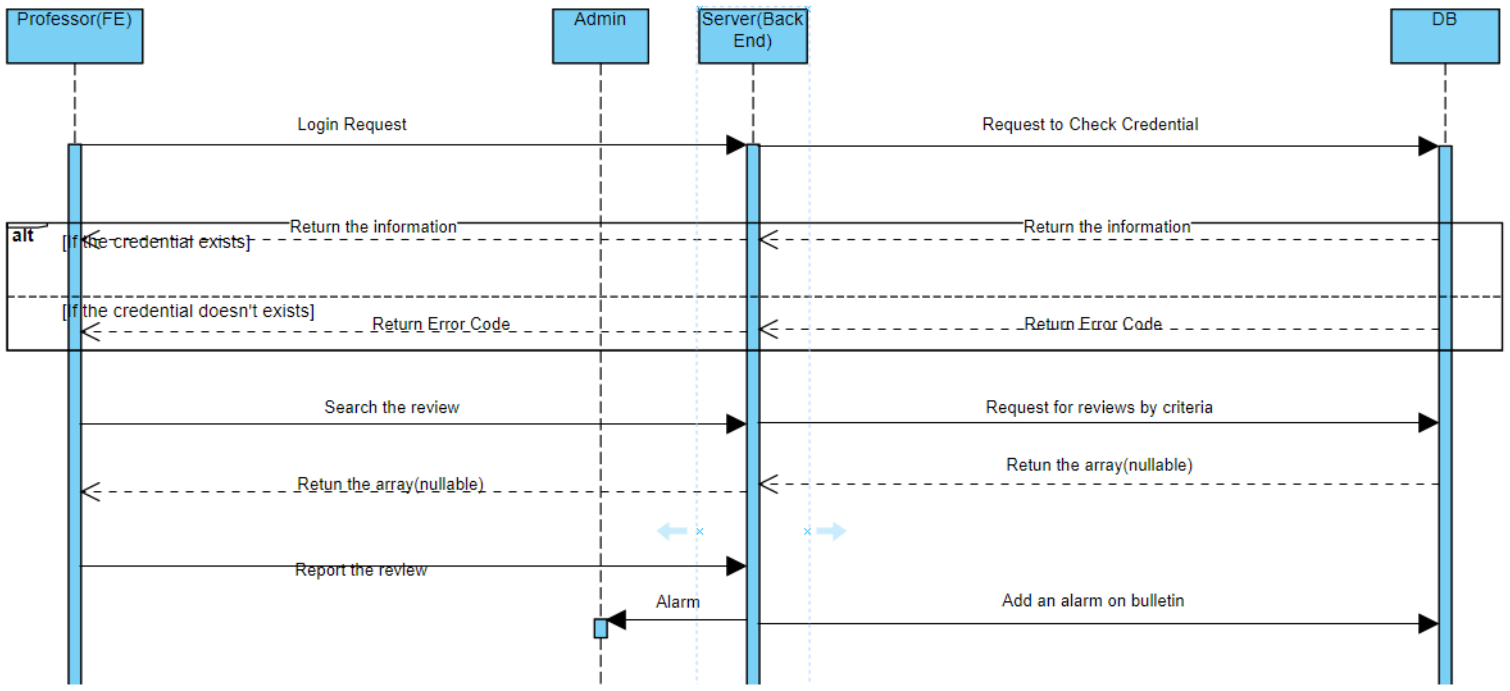
For mongoDB, we make schemas for each collection. Please watch the below image.


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)

3.3. UML Sequence


User(FE)
Server(Back End)
DB
Login Request
Request to Check Credential
alt
[If the credential exists]
Return the information
Return the information
[If the credential doesn't exists]
Return Error Code
Return Error Code
sd Loop
Add course by classNumber
Request to return the information
Return the information of class
alt
[If course with classNumber exists]
Reflect the information on webpage
Add course on timetable class
[If course with classNumber doesn't exists]
Error Code

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 login and report the review.

3.4. 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.	Bulk update of customers	Remove all customers

		Authorization needed		
/course/1	Retrieve the details for course 1(Class Number)	Error	Update the details of customer 1 if it exists	Remove course 1
/course/find	Retrieve the courses by the information of req.body.course	Error	Error	Error

/report

	GET	POST	PUT	Delete
/report	Get all reports that the user sent	Post the report for admin and send an alarm	error	Refute all the reports that the user sent.
/report/1	Get the information of report 1	Error	Update the information of report 1	Refute the report 1 that user sent

/user

	GET	POST	PUT	Delete
/user/login	Get the information of the user based on the credential, and also session information	error	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.

/admin

	GET	POST	PUT	Delete
/admin/course	Get all the course	Add a course in DB based on the req.body. Req.body, should include the credential of admin	error	Delete all the courses in DB.
/admin/review/1	Get the information of review 1	error	error	Delete the review
/admin/viewAlarm	Get the data for all alarm	error	error	Delete all alarm

3.5. Deployment

We are going to use netlify to deploy our website. Netlify is a hosting service that developers can easily publish the web application. The developer can connect with Github.

3.6. Code Convention

We are going to follow the code convention:
https://www.w3schools.com/js/js_conventions.asp

4. Schedule

	Gyuri	Sungwoong
Milestone 1 (4/25):	Complete design, Login function, timetable	Complete API for login with session and timetable
Milestone 2 (5/2):	Work on rate the professors	Complete API for review of professors, and search system.
Milestone 3 (5/9):	Work on chat system	Complete API for admin management system.

Milestone 4 (5/16 - Beta release):	Complete chat system	Complete API and socket for chatting system.

5. Contributions

Gyuri Kim: Write the document, Discuss and search for code convention and deployment, presentation.

Sungwoong Park: Write the document, Discuss and search for code convention and deployment, presentation.