ClubHouse

System Design Document

Dhruv Patel

Priyank Dave

Amy Li

Arailym Mussilim

Noah Cristino

Tharuth Attanayake

Faraz Kaleem Malik

Contents

Front-End

Back-End

Software Architecture

Front-end

React Component(s): Club Admin Edit Profile Component	
Parent Class (if any): N/A **Subclasses (if any):** N/A	
Responsibilities: ● Allow club admin to edit the profile page including, contact info, description of club, and profile picture.	**Collaborators:**

React Component(s): Club Admin View Profile Component	
Parent Class (if any): N/A **Subclasses (if any):** N/A	
Responsibilities: ● Displays the current profile page of the club associated with the club admin	**Collaborators:**

React Component(s): Club Admin Profile Page	
Parent Class (if any): N/A **Subclasses (if any):** N/A	
Responsibilities: ● Allow the Club Admin to go into edit mode or remain in view profile mode.	**Collaborators:** ● Club Admin View Profile Component ● Club Admin Edit Profile Component

React Component(s): Event Form (Student View)	
Parent Class (if any): N/A **Subclasses (if any):** N/A	
Responsibilities: ● Knows the logged in club's name ● Knows the event name input ● Knows the event image input ● Knows the event start time input ● Knows the event end time input ● Knows the event location input ● Knows the event description input ● Knows the event's attendees ● Knows the event tags input ● When submit button is clicked, a post request gets submitted using api endpoint in events router, and then redirect to club profile ● When delete button is clicked, redirect to previous page	**Collaborators:**

React Component(s): StudentEventCard	
Parent Class (if any): N/A **Subclasses (if any):** N/A	
Responsibilities: ● Displays eventName, organizer's name, start and end times, location ● Reformats date, start and end times to display in "mmmm dS, yyyy" and "HH:mm" (shortTime), respectively	**Collaborators:**

React Component(s): RegisterForm	
Parent Class (if any): React.Component **Subclasses (if any):** N/A	
Responsibilities: ● Lets user input username and password ● Shows verification input field when required ● Allows user to input verification code ● Once verified, redirects user to the login screen ● Perform basic verification on registration details ● If any error, show them on corresponding fields at tiny subtext	**Collaborators:** ● Login

React Component(s): Login	
Parent Class (if any): React.Component **Subclasses (if any):** N/A	
Responsibilities: ● Allows the user to input an email and password and login ● Logs user in and sets cookie if credentials match database ● Knows the current state of the register form ● Redirects user to home page if already logged in ● Displays error message if password is incorrect or if the user doesn't exist	**Collaborators:**

React Component(s): NavBar	
Parent Class (if any): React.Component **Subclasses (if any):** N/A	
Responsibilities: ● Sets navbar components ● Changes color when clicked on different components	**Collaborators:**

React Component(s): Clubs (Student View)	
Parent Class (if any): React.Component **Subclasses (if any):** N/A	
Responsibilities: ● Provides megaview for club ● Check database for clubs ● Display club information from above ● Consists of: ○ Club Banner ○ Club Name ○ Club Phone Number ○ Club Email ○ And a View Club button	**Collaborators:**

React Component(s): ClubCard	
Parent Class (if any): React.Component **Subclasses (if any):** N/A	
Responsibilities: ● Format for the Club map in “Clubs” page ● Includes minimal pseudo-styling	**Collaborators:**

Back-end

Class Name: User	
Parent Class (if any): **Subclasses (if any):**	
Responsibilities: ● Knows its email and password ● Know what type of account credentials are attached to	**Collaborators:** ●

Class Name: Student	
Parent Class (if any): **Subclasses (if any):**	
Responsibilities: ● Knows its real name ● Knows associated User Object ● Knows all ids of clubs she/he a general member of ● Knows all ids of clubs he/she follows	**Collaborators:** ● User

Class Name: Club	
Parent Class (if any): **Subclasses (if any):**	
Responsibilities: ● Knows its name ● Keeps track of its general members ● Knows its categories/tags ● Keeps track of	**Collaborators:** ●

Class Name: Event	
Parent Class (if any): **Subclasses (if any):**	
Responsibilities: ● Knows its organizer's name ● Knows its name ● Knows its image ● Knows its start time ● Knows its end time ● Knows its location ● Knows its description ● Knows its attendees ● Knows its tags/categories	**Collaborators:** ●

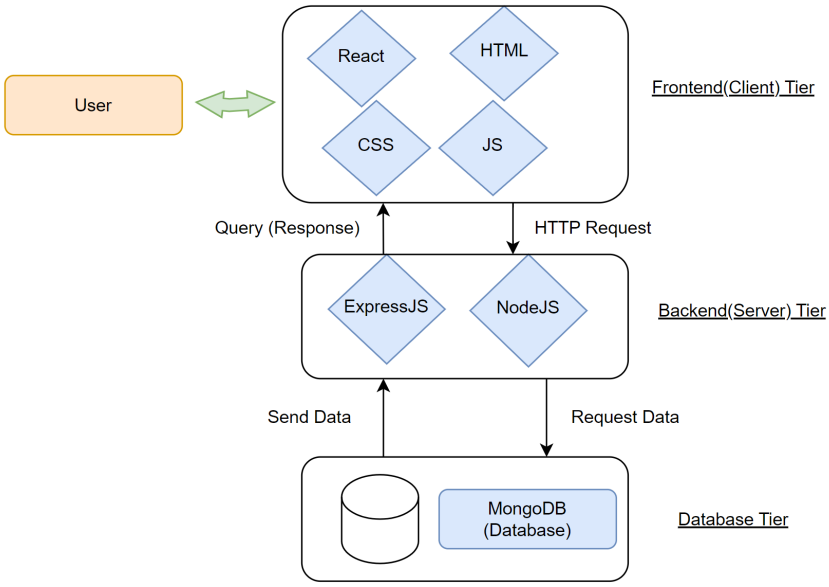
Class Name: usersDAO	
Parent Class (if any): **Subclasses (if any):**	
Responsibilities: ● Performing database operations on users and potentialUsers ● inserting ● finding ● deleting	**Collaborators:**

Class Name: adminProfileDAO	
Parent Class (if any): **Subclasses (if any):**	
Responsibilities: ● Performing database operations on clubs and club specific events ● Inserting image strings ● Querying club profile information ● Fetching event details specific to the club	**Collaborators:**

Class Name: emailWrapper	
Parent Class (if any): **Subclasses (if any):**	
Responsibilities: ● Sending a verification email	**Collaborators:**

Three Tier Architecture

This project utilized the Three Tier Architecture (MERN Stack). The user interacts with React (the client) frontend of the web application. Any user requests will enable React to send a HTTP request to the application server which is written using NodeJS and ExpressJS. The server then serves the client by accessing the MongoDB database.



System Decomposition

The system architecture has three separate parts: the database, the frontend, and the backend. The frontend is what the users will be interacting with. The frontend is developed using React and its components. We also used Material UI for universal styling and creating smoother working components. We also call the API endpoints that the backend provides using the fetch call. This way we can allow for CRUD properties to be introduced in our application. The backend of the app was built using Node.js and Express.js. Express is a popular node framework and has a lot of benefits. One of the ones we used is for writing handlers for http requests at different URL paths (routes). This is how we set up the API endpoints that interact with our database (MongoDB).In our database, we have set up different collections for different objects such as clubs, events, and users. We have implemented input sanitation code to make sure we don't receive invalid inputs. And for invalid images, we have implemented pop-up alerts that show up on the user's screen.