## horizontal line



Group Chat App

**Submitted By: Group 9**

1. Abhishek Kumar - 17114005
2. Gagan Kumre - 17114028
3. Gandhi Ronnie - 17114029
4. Hemant Singh - 17114038
5. Kishan Kumar - 17114046
6. Siddhant Nayak - 17114071

## horizontal line

# Contribution Summary

## Siddhant Nayak (17114071) Pg 2-3

Worked on the Backend Architecture for User Model and Database. Also integrated the user model to the public\_chat page route.

## Hemant Singh (17114038) Pg 3-4

Worked on the backend routing which included integration of socket.io with the rest of the backend of the app, the user model to the login page route and register page route, and also took care of logout on the public chat page.

## Gagan Kumre (17114028) Pg 5-6

Worked on implementation of signup, authentication function, validation of college email address and hashing of users password with the use of passport.js and mongodb

## Abhishek Kumar (17114005) Pg 7-8

Worked on emitting events (send and receive any events with any data) from client to server and broadcasting (emit the event from the server to the rest of the users).

## Kishan Kumar (17114046) Pg 9-10

Designed complete UI and some of the frontend backend integration.

## Gandhi Ronnie (17114029) Pg 11-14

Coding the front-end (UI) of the app and connecting the backend with the frontend in Express.JS.

**Source Code :** [**https://github.com/gagankumre/ACN-Project**](https://github.com/gagankumre/ACN-Project)

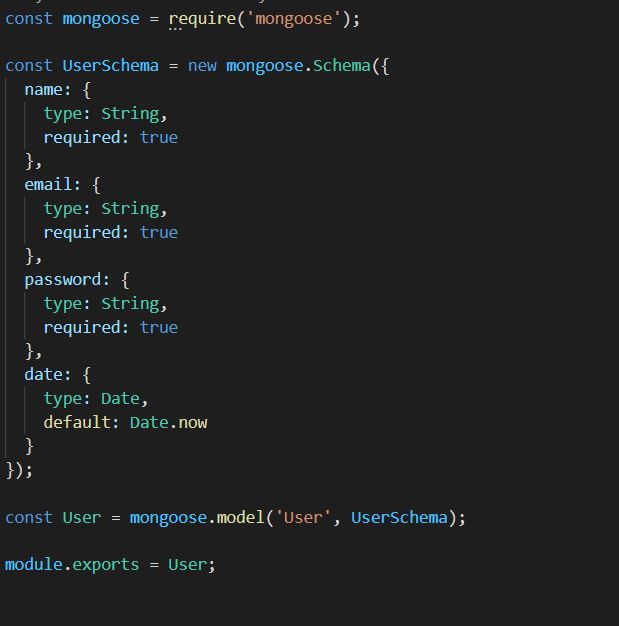
## horizontal line

**Introduction**

We have used Socket programming to build a platform for reliable conveyance and smooth flow of information from the administration to the students alongside functionality of students being able to communicate with each other as well. The platform (Group chat app) has an authentication feature as well so only relevant people (Students of same college) can participate and use the platform.

**Backend Architecture for User Model + Database**

* The app has a User model that defines basic information of a user like name, email, password and date.
* The information for all users is stored in a MongoDB database

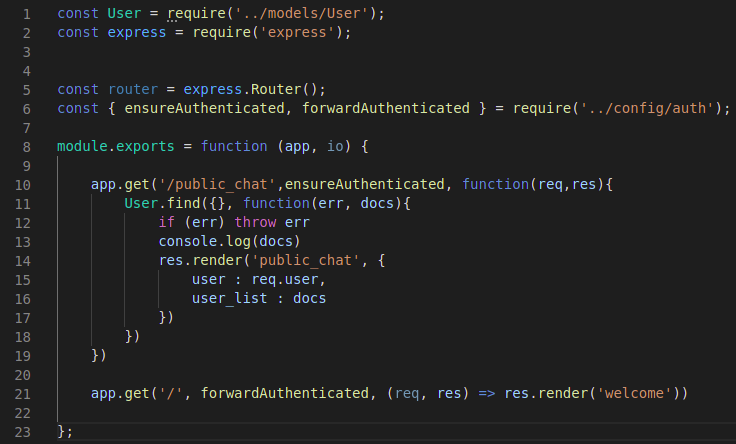


## 

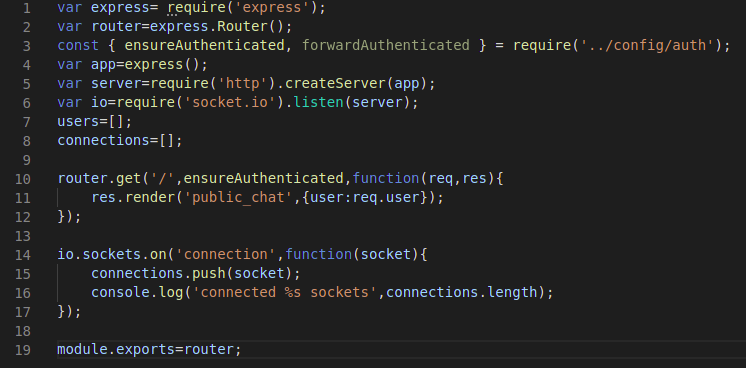
**Backend Architecture for creating routes**

There are following route for each purpose:

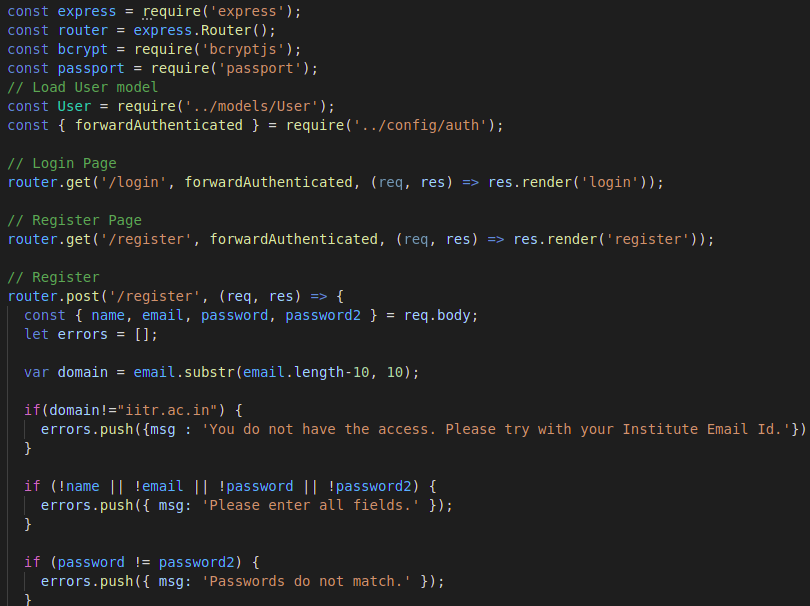
* **Index.Js** : It integrates the user model to the public\_chat page route.



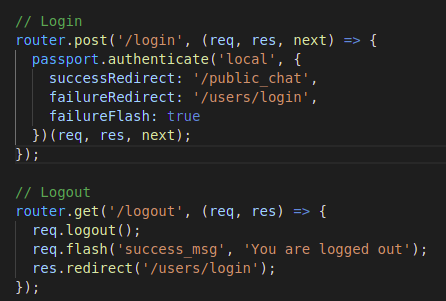
* **public\_chat.Js**: It handles integration of socket.io with the rest of the backend of the app by building a connection.



* **user.js** : It integrates the user model to the login page route, register page route, and also takes care of logout on the public chat page.



## 

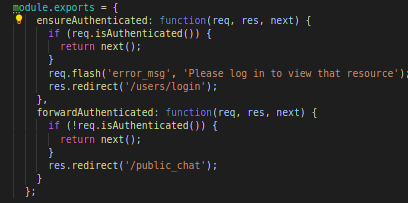


**Authentication**:

* Passport.js : provides encryption with serializers and deserializer functions and makes login a secure process for the users

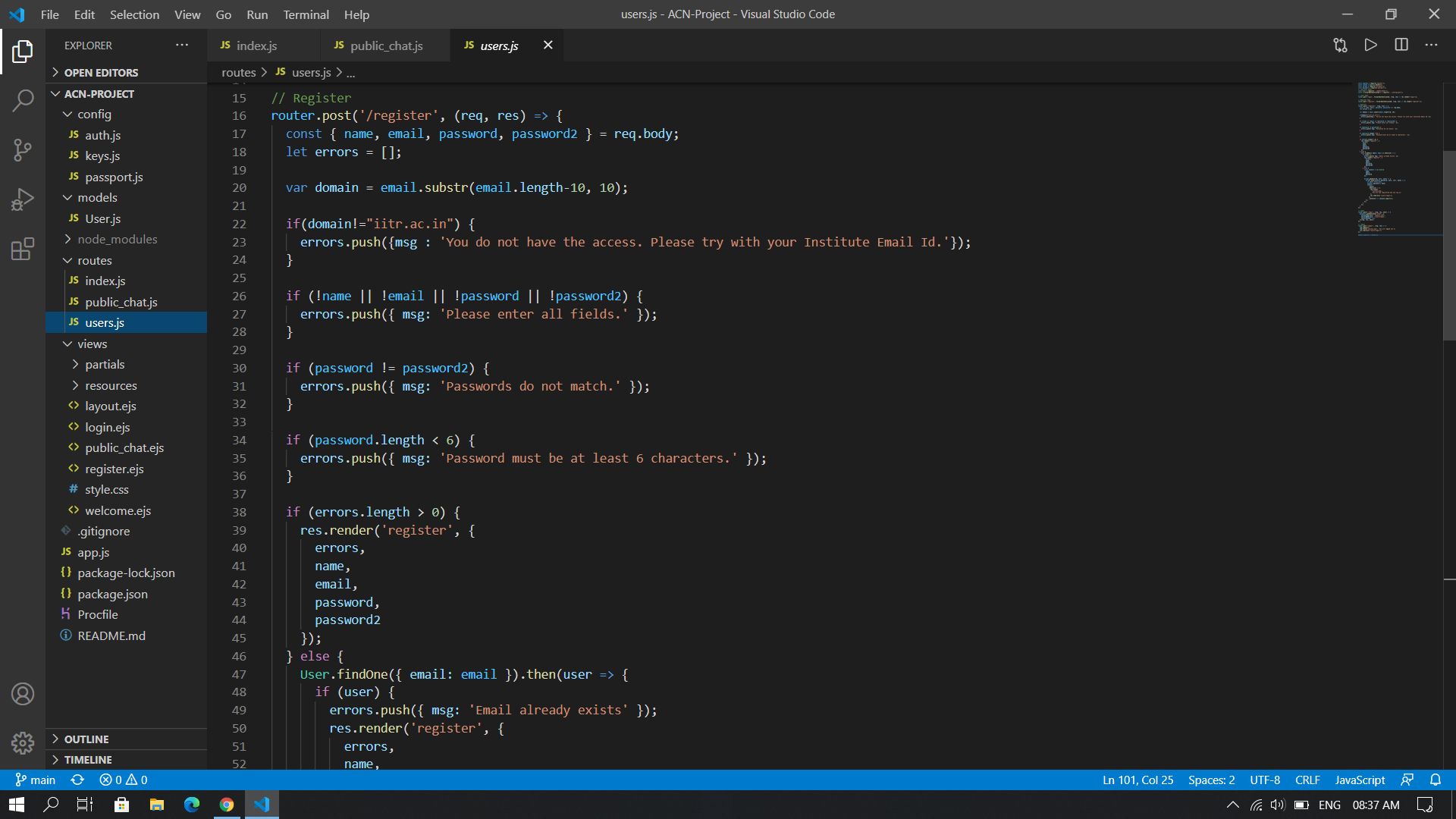


* auth.js: it makes sure only trusted and authenticated emails are allowed else it will show an error message.

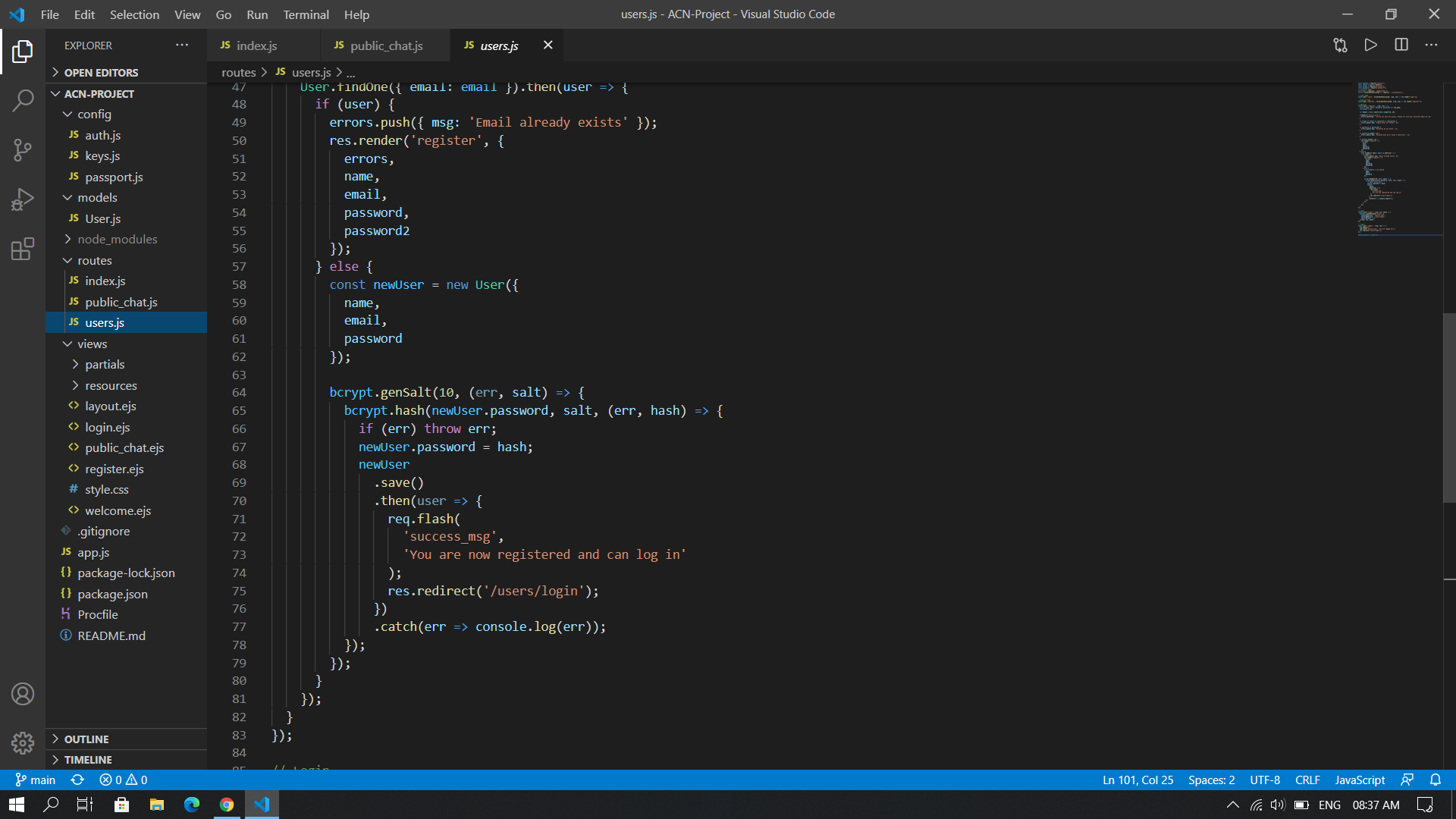


* users.js : used in registering the new user and validation of his details (checking of password length, checking the domain of email, comparison of confirm password and password, checking if user already exists etc.)

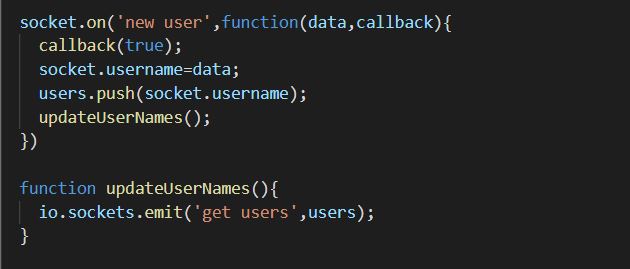
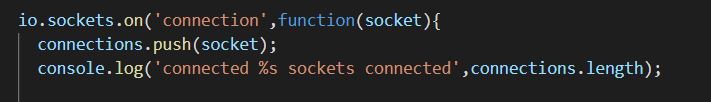
Validation of domain-



Hashing of password-



**Socket Programming**

1. Set the environment variable PORT to tell the web server what port to listen on.  
     
   
2. Created an event to send the username to the server.  
     
   
3. Created the server to accept the new username sent by client and update all users.  
     
   
4. Created the server to accept connections from the client.   
     
   

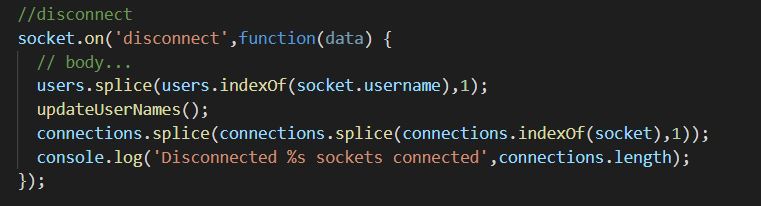
## 

1. Created the link between server and client for messaging.

(i) Client Side for the messaging



(ii) Server Side of the messaging  


1. Created an event to disconnect clients from the server.  
     
   

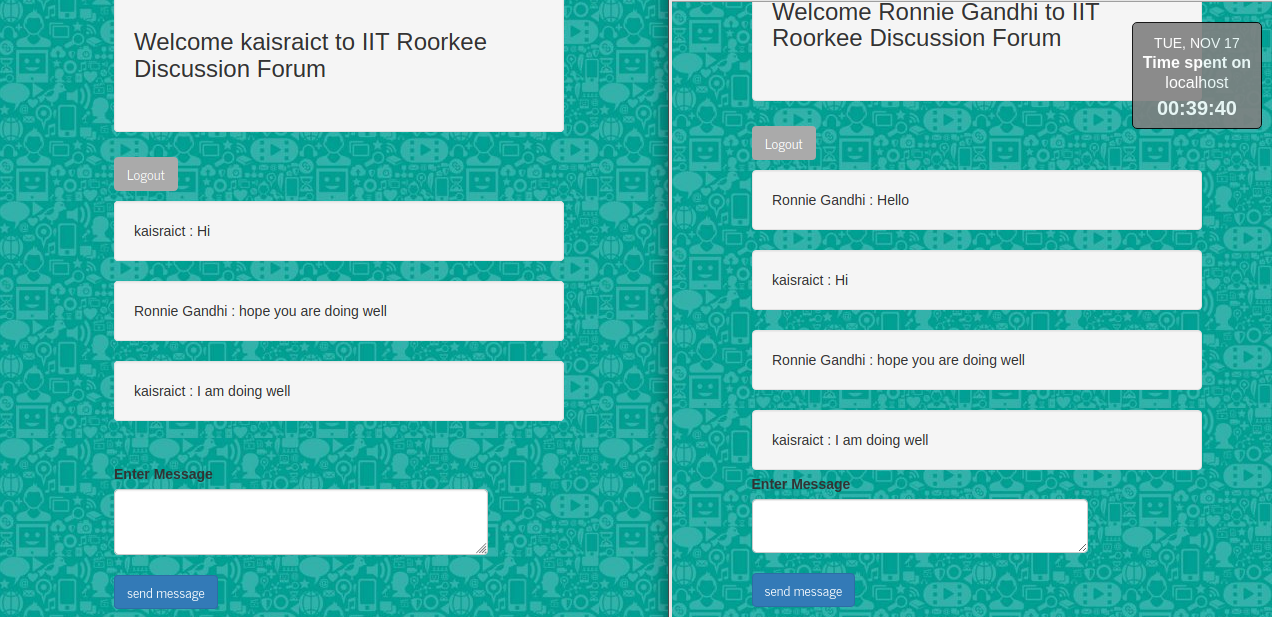
## 

**User Interface Design**

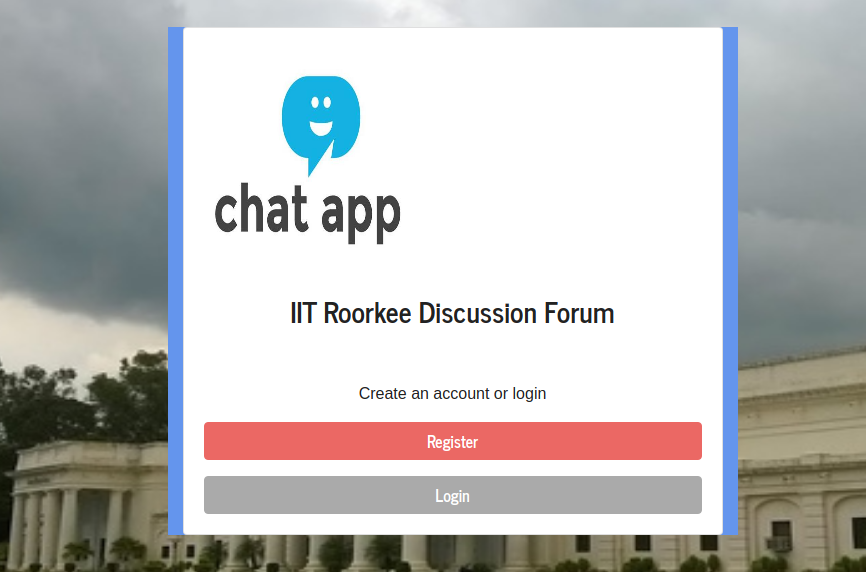
The UI for this app consists of:

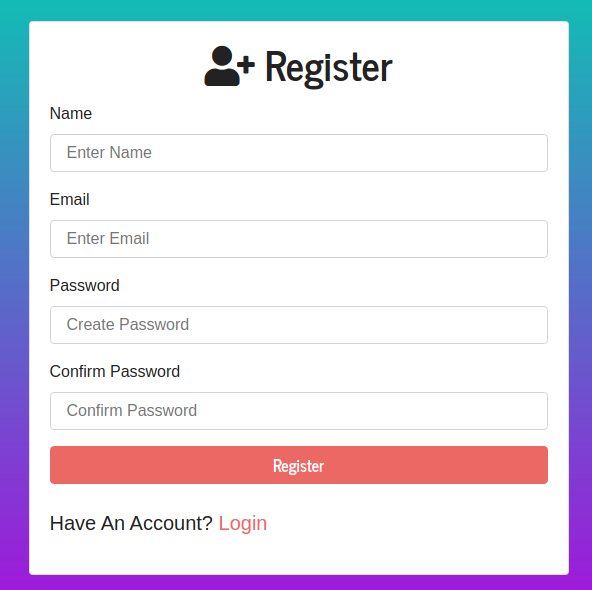
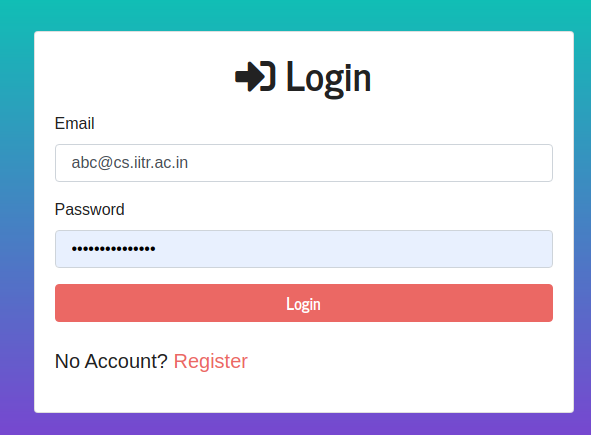
1. A welcome page asking if you are a new user or want to login.
2. Then there is a register pager or login page based on what user chooses. In case user registers, he/she is forwarded to login
3. After the user logs in, the public chat page is shown. Here every user
   1. can see messages posted from other users
   2. post messages themselves also.

Design is as follows:



## 

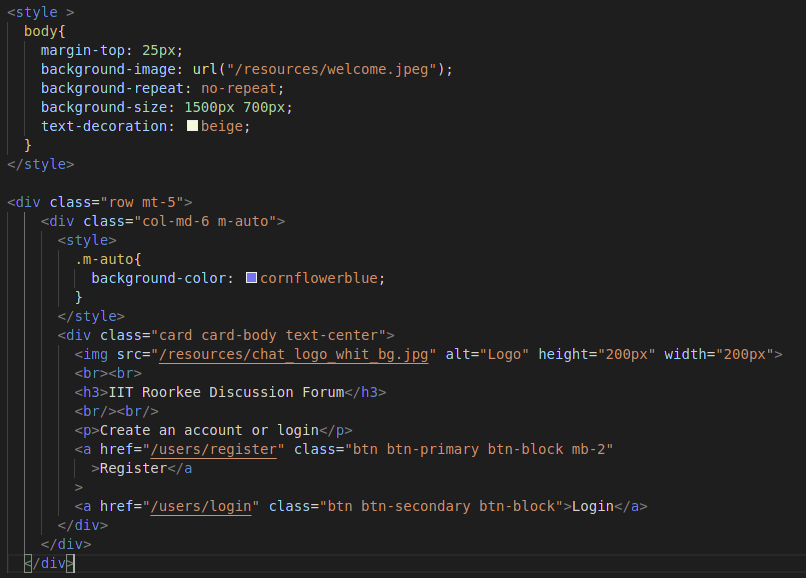


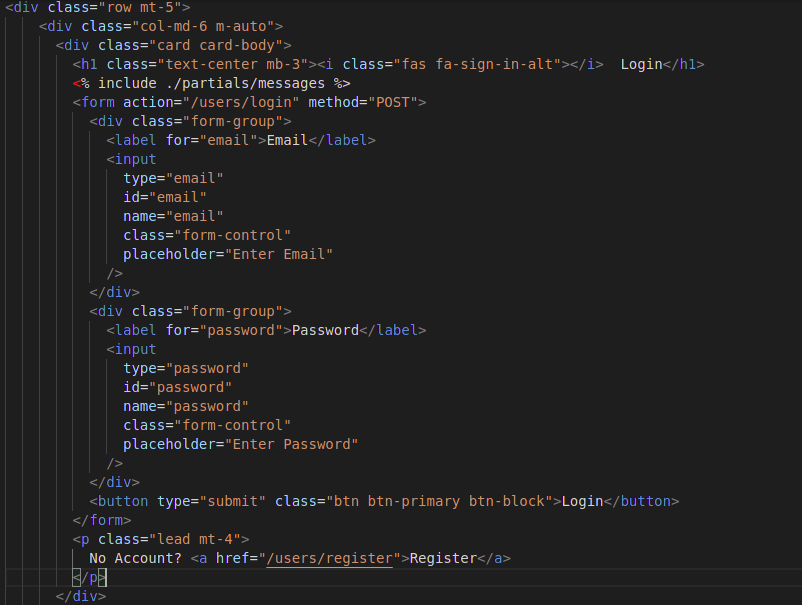
**User Interface Code(Frontend) + Some backend Integration**

So the frontend contains following items:

* **welcome.ejs**: Created the welcome page with container having chat app logo + title + asking user to register or login as shown above.



* **login.ejs: It asks already registered users to login with a username, i.e. their email and password and sends them to public\_chat page after they press the login button. If username is not registered then ask the user for the same and send him to the registration page.**



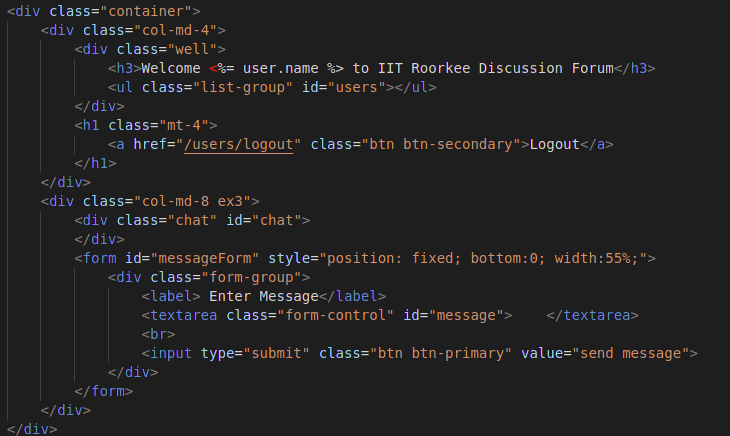
## 

* **register.ejs: It takes care of registering only authenticated, with iitr email , users and stores their data in the database by connecting with the backend user model. If user already has an account, it asks user for the same and sends user to login page.**



* **public\_chat.ejs: Creating the chatting area, with a welcome note for the user. It has a scrollable chat messages container and a “Enter Message” textbox with a send button and a logout button. The socket.io code to access the newly received message and send a new message to the client is applied here.**

## 

****