



CA3 - Project

On

Chat Application

Submitted by:

Name	Reg.no	Section /Roll.no
B.V.Hanumantha Reddy	11715087	KM012 / 27

Submitted to: Ms. Neha Sharma

Subject : INT 222.

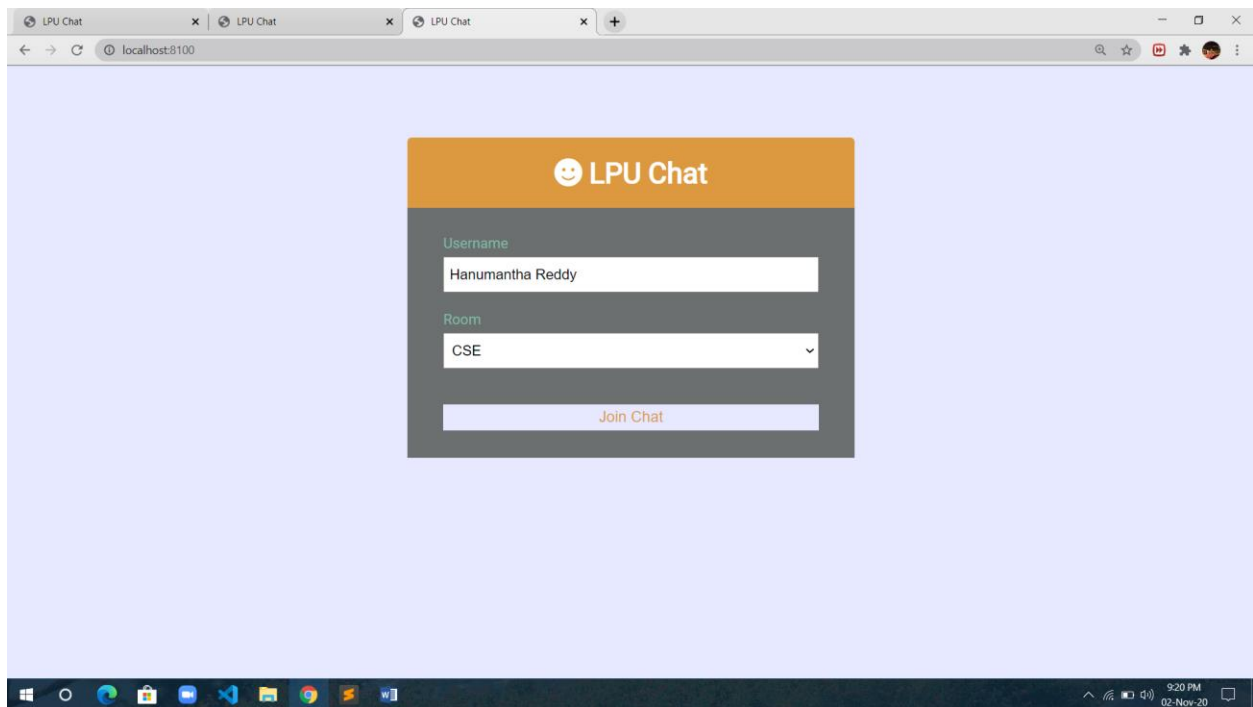
Date : 2nd Nov 20.

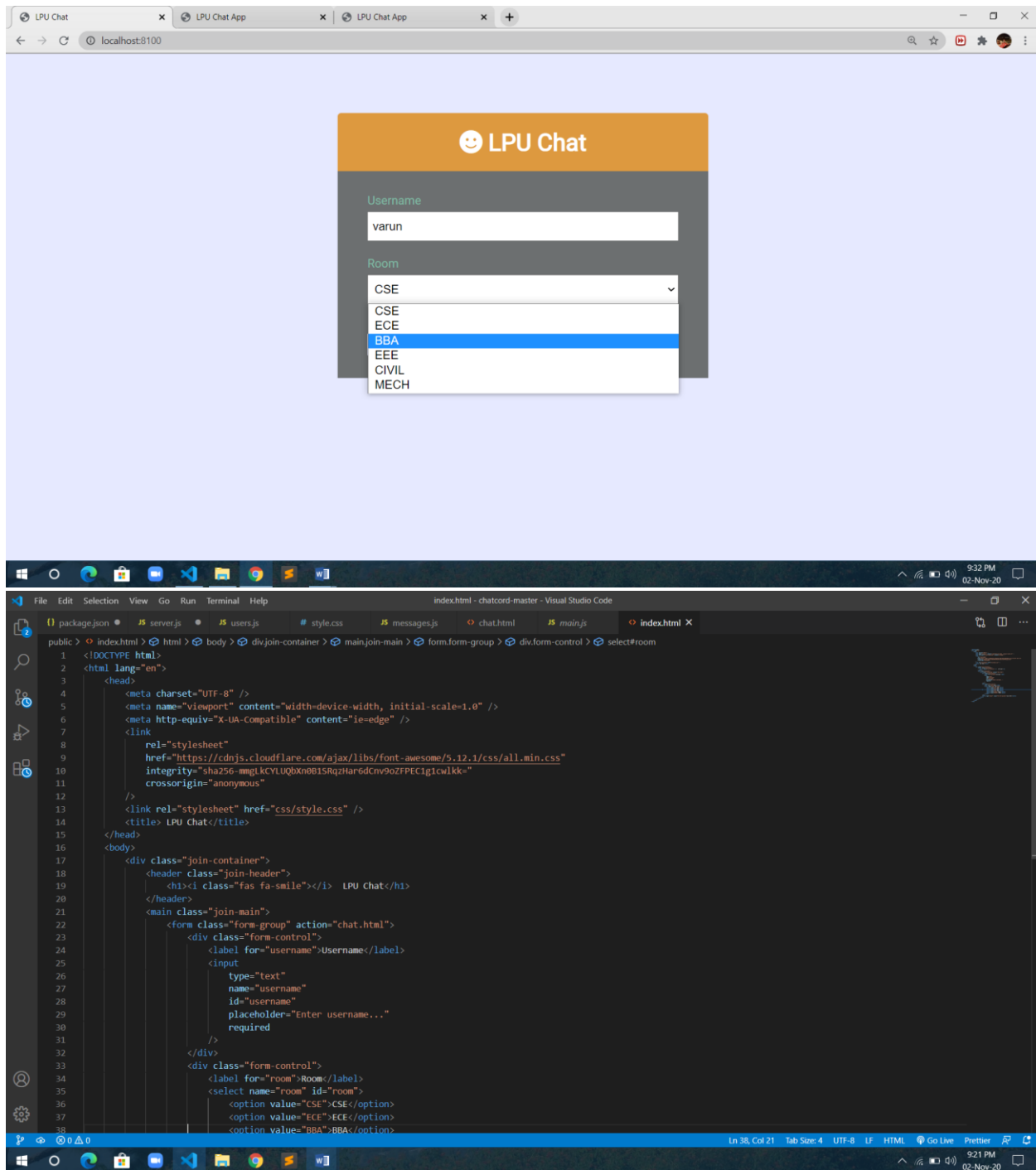
Github link:

<https://github.com/Hanureddy94/ChatApp-NodeJs/>

Code snippets and Outputs:

Joining Chat :





Chat page:

```
File Edit Selection View Go Run Terminal Help chat.html - chatcord-master - Visual Studio Code
public > chat.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8" />
5 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6 <link
7   rel="stylesheet"
8   href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.12.1/css/all.min.css"
9   integrity="sha256-mmglkCYLUQbXn0B1SRqZHar6dCny90ZFPEC1g1cwlkk="
10  crossorigin="anonymous"
11 />
12 <link rel="stylesheet" href="css/style.css" />
13 <title>LPU Chat App</title>
14 </head>
15 <body>
16 <div class="chat-container">
17   <header class="chat-header">
18     <h1><i class="fas fa-smile"></i> LPU chat</h1>
19     <a href="index.html" class="btn">Leave Room</a>
20   </header>
21   <main class="chat-main">
22     <div class="chat-sidebar">
23       <h3><i class="fas fa-comments"></i> Room Name:</h3>
24       <h2 id="room-name"></h2>
25       <h3><i class="fas fa-users"></i> Users</h3>
26       <ul id="users"></ul>
27     </div>
28     <div class="chat-messages"></div>
29   </main>
30   <div class="chat-form-container">
31     <form id="chat-form">
32       <input
33         id="msg"
34         type="text"
35         placeholder="Enter Message"
36         required
37         autocomplete="off"
38       />
39     </form>
40   </div>
41 </div>
42 </body>
43 </html>
```

```
File Edit Selection View Go Run Terminal Help chat.html - chatcord-master - Visual Studio Code
public > chat.html > ...
21 <main class="chat-main">
22 <div class="chat-sidebar">
23   <h3><i class="fas fa-comments"></i> Room Name:</h3>
24   <h2 id="room-name"></h2>
25   <h3><i class="fas fa-users"></i> Users</h3>
26   <ul id="users"></ul>
27 </div>
28 <div class="chat-messages"></div>
29 </main>
30 <div class="chat-form-container">
31 <form id="chat-form">
32 <input
33   id="msg"
34   type="text"
35   placeholder="Enter Message"
36   required
37   autocomplete="off"
38 </input>
39 <button class="btn"><i class="fas fa-paper-plane"></i> Send</button>
40 </form>
41 </div>
42 </div>
43 </main>
44 <script
45   src="https://cdnjs.cloudflare.com/ajax/libs/qs/6.9.2/qs.min.js"
46   integrity="sha256-TDxxjkAUay70ae/QJBEpgckpvs1Xai#hayk1IVg1FRt4="
47   crossorigin="anonymous"
48 ></script>
49 <script src="/socket.io/socket.io.js"></script>
50 <script src="js/main.js"></script>
51 </body>
52 </html>
53
```

(main.js)

```
File Edit Selection View Go Run Terminal Help main.js - chatcord-master - Visual Studio Code

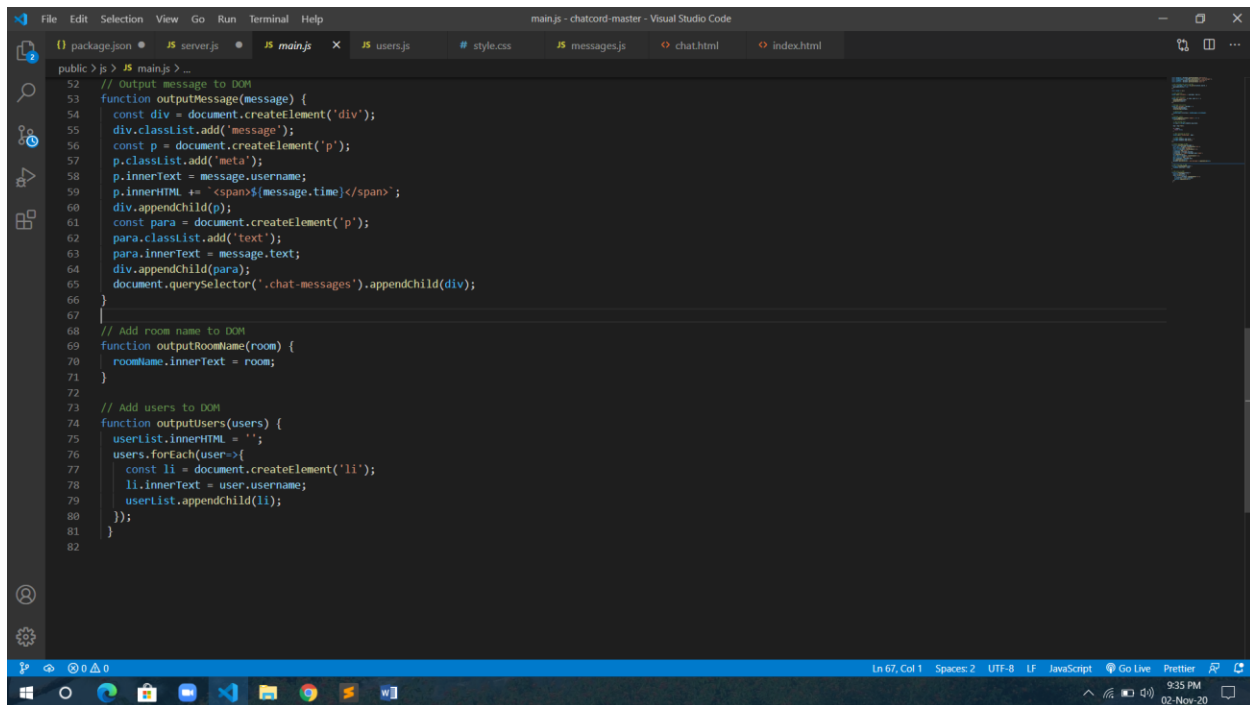
package.json JS server.js JS main.js JS users.js # style.css JS messages.js chat.html index.html

public > js > JS main.js > ...
1 const chatForm = document.getElementById('chat-form');
2 const chatMessages = document.querySelector('.chat-messages');
3 const roomId = document.getElementById('room-name');
4 const userList = document.getElementById('users');
5
6 // Get username and room from URL
7 const { username, room } = Qs.parse(location.search, {
8   ignoreQueryPrefix: true
9 });
10
11 const socket = io();
12
13 // Join chatroom
14 socket.emit('joinRoom', { username, room });
15
16 // Get room and users
17 socket.on('roomUsers', ({ room, users }) => {
18   outputRoomName(room);
19   outputUsers(users);
20 });
21
22 // Message from server
23 socket.on('message', message => {
24   console.log(message);
25   outputMessage(message);
26
27   // Scroll down
28   chatMessages.scrollTop = chatMessages.scrollHeight;
29 });
30
31 // Message submit
32 chatForm.addEventListener('submit', e => {
33   e.preventDefault();
34
35   // Get message text
36   let msg = e.target.elements.msg.value;
37
38   msg = msg.trim();
```

```
File Edit Selection View Go Run Terminal Help main.js - chatcord-master - Visual Studio Code

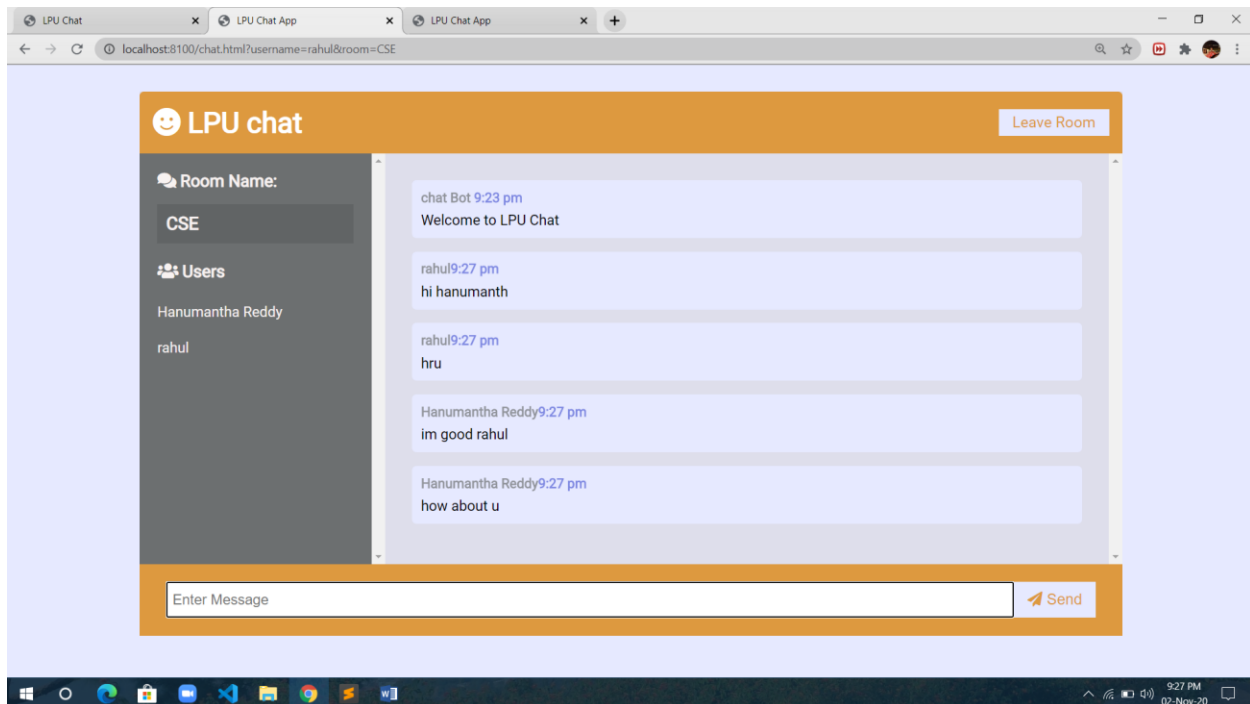
package.json JS server.js JS main.js JS users.js # style.css JS messages.js chat.html index.html

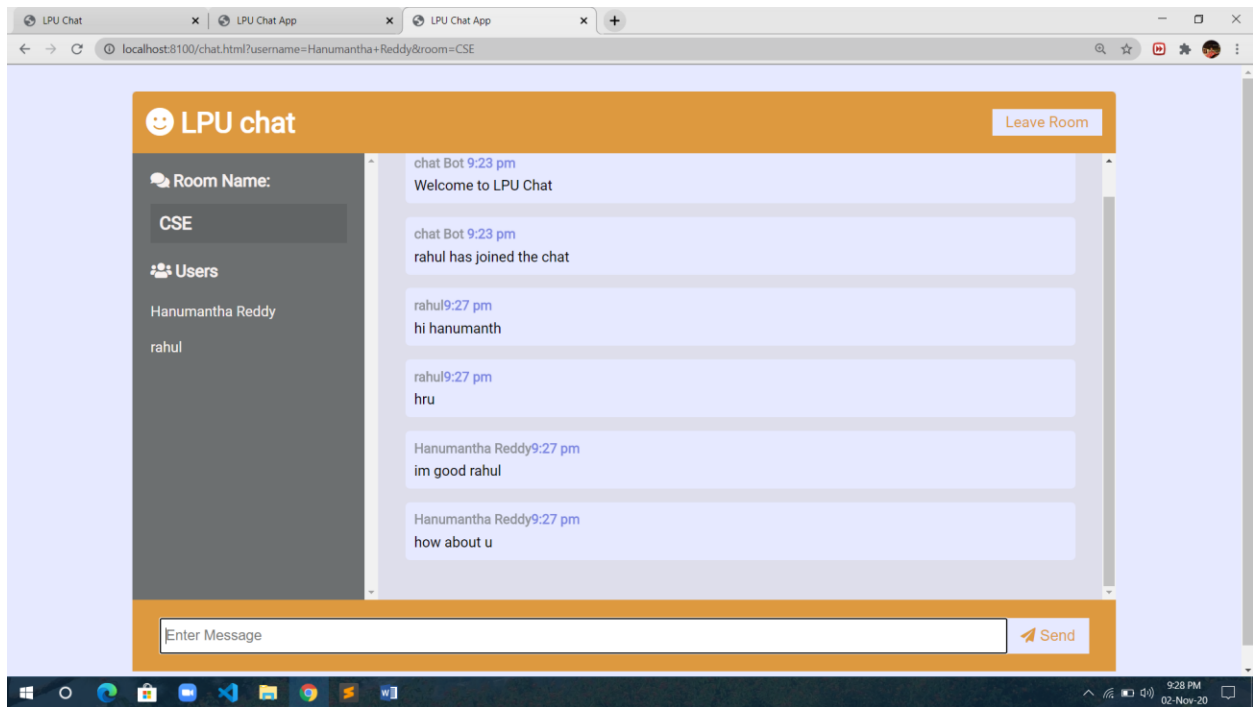
public > js > JS main.js > ...
37
38   msg = msg.trim();
39
40   if (!msg) {
41     return false;
42   }
43
44   // Emit message to server
45   socket.emit('chatMessage', msg);
46
47   // Clear input
48   e.target.elements.msg.value = '';
49   e.target.elements.msg.focus();
50 });
51
52 // Output message to DOM
53 function outputMessage(message) {
54   const div = document.createElement('div');
55   div.classList.add('message');
56   const p = document.createElement('p');
57   p.classList.add('meta');
58   p.innerText = message.username;
59   p.innerHTML += `<span>${message.time}</span>`;
60   div.appendChild(p);
61   const para = document.createElement('p');
62   para.classList.add('text');
63   para.innerText = message.text;
64   div.appendChild(para);
65   document.querySelector('.chat-messages').appendChild(div);
66 }
67
68 // Add room name to DOM
69 function outputRoomName(room) {
70   roomId.innerText = room;
71 }
72
73 // Add users to DOM
74 function outputUsers(users) {
```



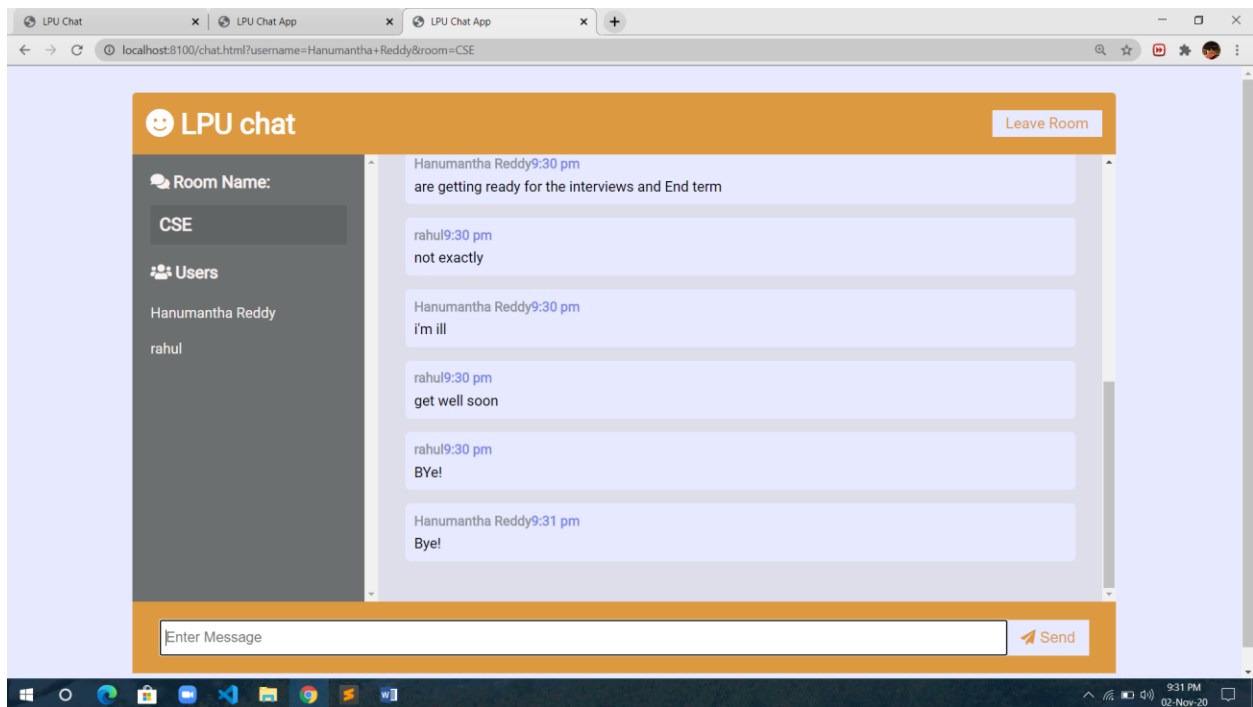
```
public > js > main.js > ...
52 // Output message to DOM
53 function outputMessage(message) {
54   const div = document.createElement('div');
55   div.classList.add('message');
56   const p = document.createElement('p');
57   p.classList.add('meta');
58   p.innerText = message.username;
59   p.innerHTML += `<span>${message.time}</span>`;
60   div.appendChild(p);
61   const para = document.createElement('p');
62   para.classList.add('text');
63   para.innerText = message.text;
64   div.appendChild(para);
65   document.querySelector('.chat-messages').appendChild(div);
66 }
67
68 // Add room name to DOM
69 function outputRoomName(room) {
70   roomName.innerText = room;
71 }
72
73 // Add users to DOM
74 function outputUsers(users) {
75   userList.innerHTML = '';
76   users.forEach(user=>{
77     const li = document.createElement('li');
78     li.innerText = user.username;
79     userList.appendChild(li);
80   });
81 }
82 }
```

Messaging output:

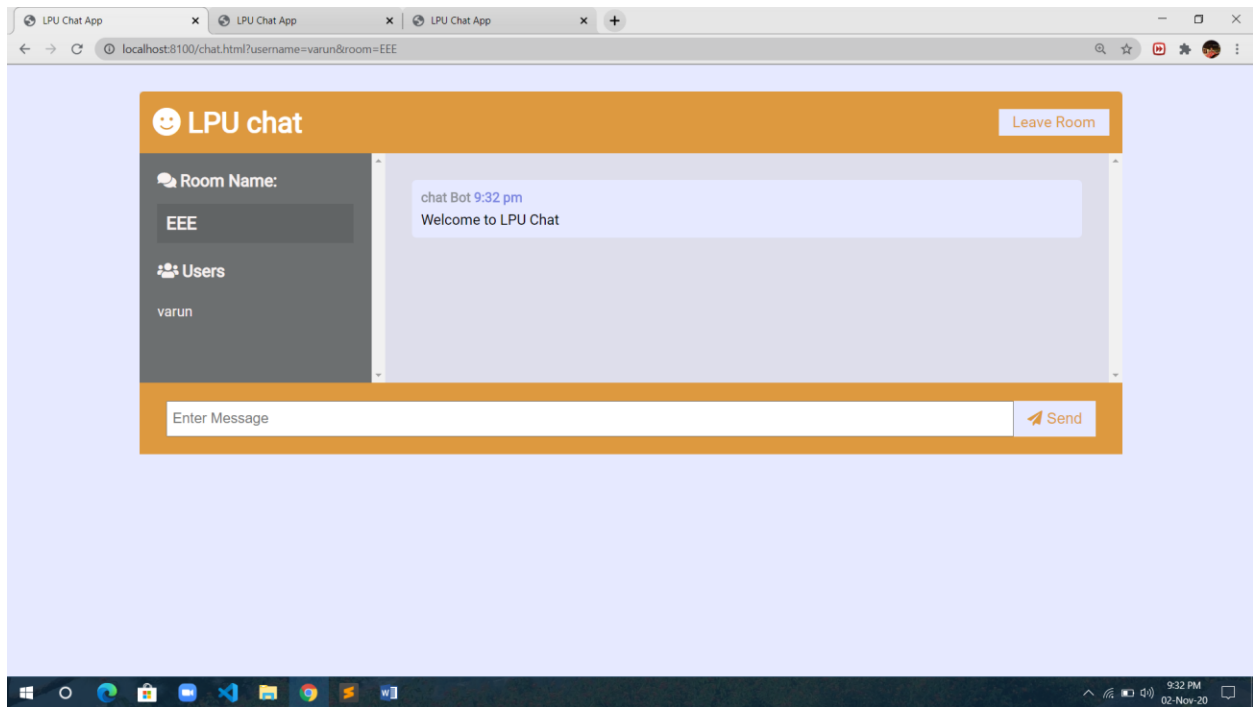




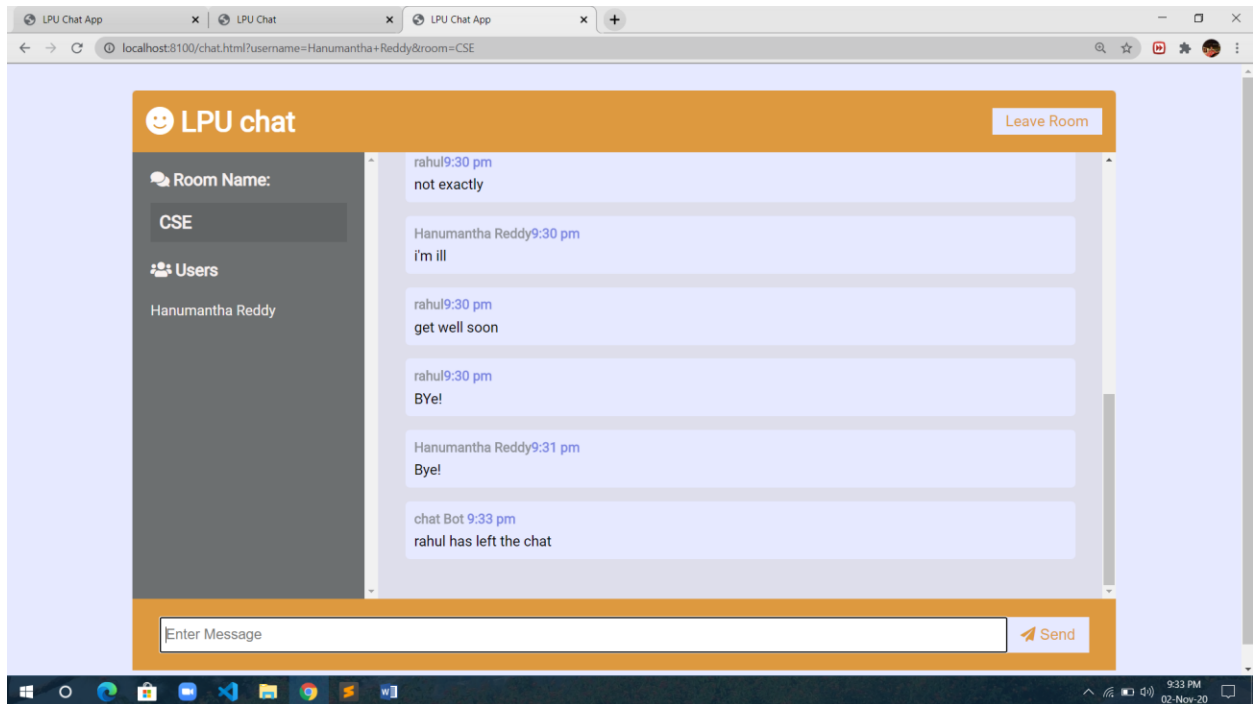
SCROLL bar:



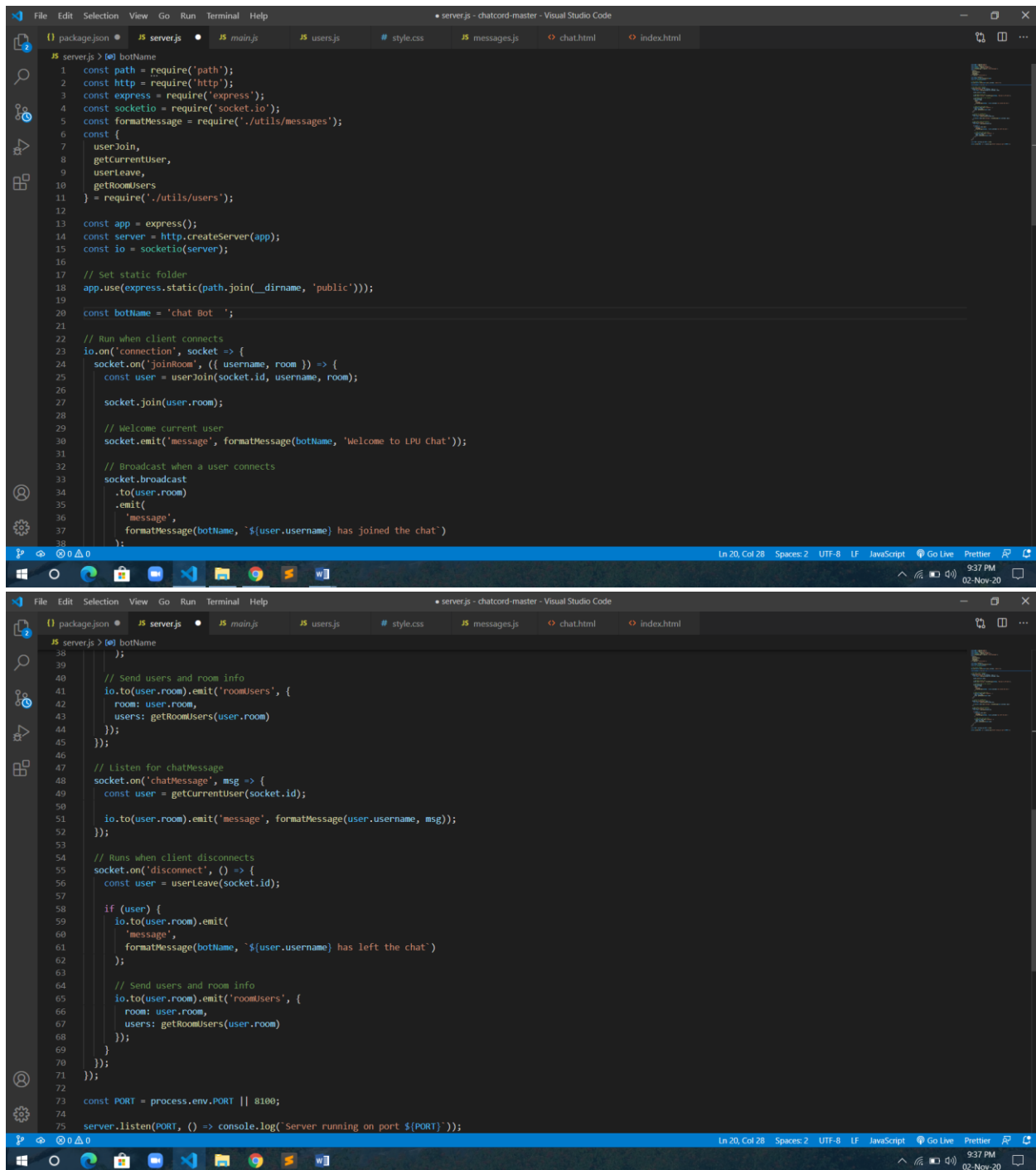
ROOMS:



After a user left Chat:



Server.js :



```
server.js > botName
1  const path = require('path');
2  const http = require('http');
3  const express = require('express');
4  const socketio = require('socket.io');
5  const formatMessage = require('./utils/messages');
6  const {
7    userJoin,
8    getCurrentUser,
9    userLeave,
10   getRoomUsers
11 } = require('./utils/users');
12
13 const app = express();
14 const server = http.createServer(app);
15 const io = socketio(server);
16
17 // Set static folder
18 app.use(express.static(path.join(__dirname, 'public')));
19
20 const botName = 'chat Bot';
21
22 // Run when client connects
23 io.on('connection', socket => {
24   socket.on('joinRoom', ({ username, room }) => {
25     const user = userJoin(socket.id, username, room);
26
27     socket.join(user.room);
28
29     // Welcome current user
30     socket.emit('message', formatMessage(botName, 'Welcome to LPU Chat'));
31
32     // Broadcast when a user connects
33     socket.broadcast
34       .to(user.room)
35       .emit(
36         'message',
37         formatMessage(botName, `${user.username} has joined the chat`)
38       );
39   });
40
41   // Send users and room info
42   io.to(user.room).emit('roomUsers', {
43     room: user.room,
44     users: getRoomUsers(user.room)
45   });
46
47   // Listen for chatMessage
48   socket.on('chatMessage', msg => {
49     const user = getCurrentUser(socket.id);
50
51     io.to(user.room).emit('message', formatMessage(user.username, msg));
52   });
53
54   // Runs when client disconnects
55   socket.on('disconnect', () => {
56     const user = userLeave(socket.id);
57
58     if (user) {
59       io.to(user.room).emit(
60         'message',
61         formatMessage(botName, `${user.username} has left the chat`)
62       );
63
64       // Send users and room info
65       io.to(user.room).emit('roomUsers', {
66         room: user.room,
67         users: getRoomUsers(user.room)
68       });
69     }
70   });
71 });
72
73 const PORT = process.env.PORT || 8100;
74
75 server.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Package.json:

```
1 {
2   "name": "lpu chat",
3   "version": "1.0.0",
4   "description": "Realtime chat app with rooms",
5   "main": "server.js",
6   "scripts": {
7     "start": "node server.js",
8     "dev": "nodemon server"
9   },
10  "author": "HANUMANTHA REDDY",
11  "license": "MIT",
12  "dependencies": {
13    "express": "^4.17.1",
14    "moment": "^2.24.0",
15    "socket.io": "^2.3.0"
16  },
17  "devDependencies": {
18    "nodemon": "^2.0.6"
19  }
20 }
21
```