[](https://github.com/santhosh250101/DomesticFlightFareEstimator)**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

(**A Unit of Keshav Memorial Technical Education (KMTES)**

**(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad).**

**A**

**MINI PROJECT REPORT**

**on**

[**Chat Room**](https://github.com/santhosh250101/DomesticFlightFareEstimator)**s**

**BACHELOR OF ENGINEERING**

in

**INFORMATION TECHNOLOGY**

**Submitted by**

**Anirudh Gunda** **2453-18-737-001**

**T V Aditya Vardhan** **2453-18-737-051**

**Tanmay Chamrolia** **2453-18-737-052**

**Under the Guidance of**

**Mr. Naresh, Assistant Professor, IT.**

[](https://github.com/santhosh250101/DomesticFlightFareEstimator)

**DEPARTMENT OF INFORMATION TECHNOLOGY**

Kachawanisingaram Village, Hyderabad, Telangana 500058.

**September 2021**



**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

(**A Unit of Keshav Memorial Technical Education (KMTES)**

**(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad).**

**CERTIFICATE**



|  |
| --- |
| This is to certify that the project report titled “**Chat Room”** is being submitted by **Anirudh Gunda** (2453-18-737-001), **TV Aditya Vardhan** (2453-18-737-051), **Tanmay Chamrolia** (2453-18-0737-052),of IIIrd year B.E.VI Semester Information Technologyis a record of bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.    Internal Guide HOD  External Examiner |

**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

(**A Unit of Keshav Memorial Technical Education (KMTES)**

**(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad).**

**DECLARATION**



|  |
| --- |
| We hereby declare that the Mini Project Report entitled, “**Chat Room”** submitted for the B.E. degree is entirely our work, and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree. |

**Date: - 10 - 2021**

**Anirudh Gunda** **2453-18-737-001**

**T V Aditya Vardhan** **2453-18-737-051**

**Tanmay Chamrolia** **2453-18-737-052**

**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

(**A Unit of Keshav Memorial Technical Education (KMTES)**

**(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad).**

**ACKNOWLEDGEMENT**



We are happy to express our deep sense of gratitude to the principal of the college   **Dr. D Jaya Prakash,** Professor, Neil Gogte Institute of Technology, for having provided us with adequate facilities to pursue our project.

We would like to thank **Dr. Madhuri sri ram,** Professor, and Head, Department of Information Technology,Neil Gogte Institute of Technology, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

We are very grateful to our project guide **Mr. Naresh, Assistant Professor, IT.,** Professor**,** Designation, Department of Information Technology, Neil Gogte Institute of Technology, for his extensive patience and guidance throughout our project work.

We sincerely thank our seniors and all the teaching and non-teaching staff of the Department of Computer Science & Engineering and Information Technology for their timely suggestions, healthy criticism, and motivation during the course of this work.

We would also like to thank classmates for always being there whenever we needed help or moral support. With great respect and obedience, We thank our parents, sisters, and brother who were the backbone behind our deeds.

Finally, We express our immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at the right time for the development and success of this work.

**Abstract**

The chatroom is way of streamlining discussions around various topics. There are different rooms in the chatroom and users can join a specific room to enter the discussion. These rooms act as one place where all the news and information about that topic can be found. There can be rooms for various technologies, anime, a certain event etc.

How Does It Work?

The chatroom works by creating a connection between the end user and the room. Here web sockets are used to improve the efficiency of the message delivery. The user is in the room as long as he is connected to the browser. A user can be part of multiple rooms at the same time.

**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

(**A Unit of Keshav Memorial Technical Education (KMTES)**

**(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad).**

**CONTENTS**



|  |  |  |  |
| --- | --- | --- | --- |
| **S NO.** |  | **TITLE** | **PAGE NO.** |
|  |  | |  |
| **1** | **INTRODUCTION** | | **08** |
|  | 1.1 | Motivation | 08 |
|  | 1.2 | Problem Statement | 08 |
|  | 1.3 | Objective | 08 |
|  | 1.4 | Limitations | 08 |
| **2** | **LITERATURE SURVEY** | | **09** |
|  | 2.1 | Surveys | 09 |
|  | 2.2 Existing System | | 09 |
|  | 2.3 | Disadvantages of Existing System | 09 |
|  | 2.4 | Proposed System | 09 |
|  | 2.5 Advantages of Proposed System | | 09 |
|  | 2.6 Conclusion | | 09 |
| **3** | **ANALYSIS** | | **10** |
|  | 3.1 | Software and Hardware Requirements | 10 |
|  | 3.2 | Content Diagram a) Flow Diagram  b) E-R Diagram | 10  11 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | | |  |
|  |  |  |  |
| **4** | **DESIGN 11 - 12** | |  |
|  | 4.1 | Use Case Diagram 11 |  |
|  | 4.2 | Logical Data Flow Diagram 12 |  |
| **5** | 4.3 State chart Diagram 12  **IMPLEMENTATION 13 - 31** | |  |
|  | 5.1 | Implementation Steps 13 |  |
|  | 5.2 | Source code 13 |  |
| **6** | **TESTING AND VALIDATION 31** | |  |
|  | 6.1 | Testing Types 31 |  |
|  | 6.2 | Project Screenshots 32 |  |
|  | 6.3 | Chat Window 32 |  |
| **7**  8 | **CONCLUSION AND FUTURE ENHANCEMENTS 33**  **REFERENCES 33** | |  |

|  |  |  |
| --- | --- | --- |
|  | **1) INTRODUCTION**   * 1. **Motivation**   Today there are tons of instant messaging applications. These are typically based on one-to-one communication. In the age of internet it is very convenient to chat with anyone at the click of a button. There also arises a need for an application which can ensure all the pros of the instant messaging but aimed at the group messaging.  **1.2 Problem Statement**  The Chatroom is an application which allows users to communicate with a set of users. This application is primarily aimed at texting between a group of people.  **1.3 Objective**  The prime objective of this project is to use web sockets to enable transfer of messages among the users. The use of web sockets also enables the instant transfer of messages unlike the traditional use of databases.  The users are stored based on the socket id which they are assigned upon visiting the website. Users can have one socket id and be a part of multiple chatrooms.  This type of system makes it easy for group texting and also is quick to access and start the conversation.  **1.4 Limitations**  The user should be able to operate the browser provided with a proper internet connection.  The whole system is based on the quality of internet connection at the user end. A bad connection can entirely spoil the user experience and also delay the conversation for the whole group.  The web browser which is being used by the user should also support the application.  **2. LITERATURE SURVEY**  **2.1 Surveys**  The current form of communication is very slow and not convenient for fast and immediate communication. For a group of people to converse, concurrently there is a lot of overhead. There are various solutions put forward to solve this.  **2.2 Existing System**  The different platforms available today for instant communication are typically designed for one-to-one communication like instant messaging.  **2.3 Disadvantages of the existing system**  The one-to-one communication platforms do no enable the simultaneous transfer of messages among a group of people who are communicating on the same topic.  **2.4 Proposed System.**  Building a system based on web sockets which enable instant transfer of messages. The user can access different chat rooms and can participate in the conversation in multiple rooms at once.  **2.5 Advantages of the proposed system**  The primary use of a chat room is to share information via text with a group of other users. The proposed system would enable this with the use of web sockets.  **2.6 Conclusion**  Since we are using the sockets for immediate transfer of messages, there is no need of storing those messages on any database. This would mean less access time for the users to see the messages.  **3.ANALYSIS**  **Software and Hardware Requirements:**   * + 1. **Software Requirements:**   **For System:**   * + Visual Studio Code   + Windows 10 OS   + Good internet connection   + Node JS run-time environment   + NPM Package Installer   **For User:**   * Pc or smartphone * Keyboard or equivalent software * Any web browser * Good Internet Connectivity      * + 1. **Hardware Requirements:** * Browser compatible device.   1. **Content diagram:**   **a) Flow chart Diagram**    **b) E-R Diagram**    **4.DESIGN**  **4.1 Use Case Diagram**    **4.2 Logical Data Flow Diagram**      **4.3 State Flow Diagram**    **5.IMPLEMENTATION**     * 1. **Implementation Steps** * Install visual studio code * Install node.js * Install express.js * Install socket.io * Install moments * Install nodemon * Open terminal and run command “npm run dev” * Open localhost:3000 in the browser   1. **Source code:**     **index.html**  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8" />  <meta name="viewport" content="width=device-width, initial-scale=1.0" />  <meta http-equiv="X-UA-Compatible" content="ie=edge" />  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.12.1/css/all.min.css"  integrity="sha256-mmgLkCYLUQbXn0B1SRqzHar6dCnv9oZFPEC1g1cwlkk=" crossorigin="anonymous"/>  <link rel="stylesheet" href="css/style.css" />  <title>HORIZON INC.</title>  </head>  <body>  <div class="join-container">  <header class="join-header">  <h1><i class="fas fa-language"></i> HORIZON</h1>  </header>  <main class="join-main">  <form action="chat.html">  <div class="form-control">  <label for="username">USERNAME</label>  <input type="text" name="username" id="username" placeholder="Enter username..." required/>  </div>  <div class="form-control">  <label for="room">ROOM</label>  <input type="text" name="room" id="room" placeholder="Enter room name..." required>  </div>  <button type="submit" class="btn">Join Chat</button>  </form>  </main>  </div>  </body>  </html>  **chat.html**  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8" />  <meta name="viewport" content="width=device-width, initial-scale=1.0" />  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.12.1/css/all.min.css"  integrity="sha256-mmgLkCYLUQbXn0B1SRqzHar6dCnv9oZFPEC1g1cwlkk=" crossorigin="anonymous" />  <link rel="stylesheet" href="css/style.css" />  <title>HORIZON</title>  </head>  <body>  <div class="chat-container">  <header class="chat-header">  <h1><i class="fas"></i> HORIZON</h1>  <a id="leave-btn" class="btn">Leave Room</a>  </header>  <main class="chat-main">  <div class="chat-sidebar">  <h3><i class="fas fa-comments"></i> ROOM NAME:</h3>  <h2 id="room-name"></h2>  <h3><i class="fas fa-users"></i> USERS</h3>  <ul id="users"></ul>  </div>  <div class="chat-messages"></div>  </main>  <div class="chat-form-container">  <form id="chat-form">  <input id="msg" type="text" placeholder="Enter Message" required autocomplete="off"/>  <button class="btn"><i class="fas fa-paper-plane"></i> Send</button>  </form>  </div>  </div>  <script src="https://cdnjs.cloudflare.com/ajax/libs/qs/6.9.2/qs.min.js"  integrity="sha256-TDxXjkAUay70ae/QJBEpGKkpVslXaHHayklIVglFRT4=" crossorigin="anonymous"></script>  <script src="/socket.io/socket.io.js"></script>  <script src="js/main.js"></script>  </body>  </html> |  |
|  | **styles.css**  @import url('https://fonts.googleapis.com/css?family=Roboto&display=swap');  :root {  --dark-color-a: #667aff;  --dark-color-b: #7386ff;  --light-color: #e6e9ff;  --success-color: #5cb85c;  --error-color: #d9534f;  }  \* {  box-sizing: border-box;  margin: 0;  padding: 0;  }  body {  font-family: 'Roboto', sans-serif;  font-size: 16px;  background: #93D9A3;  margin: 20px;  }  ul {  list-style: none;  }  a {  text-decoration: none;  }  .btn {  cursor: pointer;  padding: 5px 15px;  background: var(--light-color);  color: var(--dark-color-a);  border: 0;  font-size: 17px;  }  /\* Chat Page \*/  .chat-container {  max-width: 1200px;  background: #fff;  margin: 30px auto;  overflow: hidden;  }  .chat-header {  background: black;  color: #fff;  border-top-left-radius: 5px;  border-top-right-radius: 5px;  padding: 15px;  display: flex;  align-items: center;  justify-content: space-between;  }  .chat-main {  display: grid;  grid-template-columns: 1fr 3fr;  border-right: 3px solid black;  }  .chat-sidebar {  background: #2C2891;  color: #fff;  padding: 15px 15px 50px;  overflow-y: scroll;  width: 85%;  border-left: 3px solid black;  }  .chat-sidebar h2 {  font-size: 20px;  background: rgba(0, 0, 0, 0.1);  padding: 10px;  margin-bottom: 10px;  margin-left: 25px;  width: 80%;  }  .chat-sidebar h3 {  margin-bottom: 15px;  }  .chat-sidebar ul li {  padding: 5px 0;  margin-left: 25px;  }  .chat-messages {  padding: 30px;  max-height: 500px;  overflow-y: scroll;  }  .chat-messages .message {  padding: 10px;  margin-bottom: 15px;  background-color: #F1ECC3;  border-radius: 5px;  overflow-wrap: break-word;  }  .chat-messages .message .meta {  font-size: 15px;  font-weight: bold;  color: var(--dark-color-b);  opacity: 0.7;  margin-bottom: 7px;  }  .chat-messages .message .meta span {  color: #777;  }  .chat-form-container {  padding: 10px 10px;  background-color: #112031;  border: 3px solid black;  border-top: none;  }  .chat-form-container form {  display: flex;  }  .chat-form-container input[type='text'] {  font-size: 16px;  padding: 5px;  height: 40px;  flex: 1;  }  /\* Join Page \*/  .join-container {  max-width: 500px;  margin: 80px auto;  color: #fff;  border: 3px solid black;  border-radius: 5px;  }  .join-header {  text-align: center;  padding: 20px;  background: black;  border-top-left-radius: 5px;  border-top-right-radius: 5px;  }  .join-main {  padding: 30px 40px;  background: #506D84/\*#2C2891\*/  ;  }  .join-main p {  margin-bottom: 20px;  }  .join-main .form-control {  margin-bottom: 20px;  }  .join-main label {  display: block;  margin-bottom: 5px;  }  .join-main input[type='text'] {  font-size: 16px;  padding: 5px;  height: 40px;  width: 100%;  }  .join-main select {  font-size: 16px;  padding: 5px;  height: 40px;  width: 100%;  }  .join-main .btn {  margin-top: 20px;  width: 100%;  }  #joinChat {  display: block;  padding: 8px;  width: 30%;  margin-left: 150px;  margin-top: 40px;  }  @media (max-width: 700px) {  .chat-main {  display: block;  }  .chat-sidebar {  display: none;  }  }  #username,  #room {  border: 3px solid black;  border-radius: 7px;  }  **Package.json**  {  "name": "horizon",  "version": "1.0.0",  "description": "chatrooms for multi-user communication",  "main": "server.js",  "scripts": {  "start": "node server",  "dev": "nodemon server"  },  "author": "",  "license": "MIT",  "dependencies": {  "express": "^4.17.1",  "moment": "^2.24.0",  "socket.io": "^2.4.0"  },  "devDependencies": {  "nodemon": "^2.0.2"  }  }  **main.js**  const chatForm = document.getElementById('chat-form');  const chatMessages = document.querySelector('.chat-messages');  const roomName = document.getElementById('room-name');  const userList = document.getElementById('users');  // Get username and room from URL  const { username, room } = Qs.parse(location.search, {  ignoreQueryPrefix: true,  });  const socket = io();  // Join chatroom  socket.emit('joinRoom', { username, room });  // Get room and users  socket.on('roomUsers', ({ room, users }) => {  outputRoomName(room);  outputUsers(users);  });  // Message from server  socket.on('message', (message) => {  //console.log(message);  outputMessage(message);  // Scroll down  chatMessages.scrollTop = chatMessages.scrollHeight;  });  // Message submit  chatForm.addEventListener('submit', (e) => {  e.preventDefault();  // Get message text  let msg = e.target.elements.msg.value;  msg = msg.trim();  if (!msg) {  return false;  }    // Emit message to server  socket.emit('chatMessage', msg);  // Clear input  e.target.elements.msg.value = '';  e.target.elements.msg.focus();  });  // Output message to DOM  function outputMessage(message) {  const div = document.createElement('div');  div.classList.add('message');  const p = document.createElement('p');  p.classList.add('meta');  p.innerText = message.username;  p.innerHTML += `<span>${message.time}</span>`;  div.appendChild(p);  const para = document.createElement('p');  para.classList.add('text');  para.innerText = message.text;  div.appendChild(para);  document.querySelector('.chat-messages').appendChild(div);  }  // Add room name to DOM  function outputRoomName(room) {  roomName.innerText = room;  }  // Add users to DOM  function outputUsers(users) {  userList.innerHTML = '';  users.forEach((user) => {  const li = document.createElement('li');  li.innerText = user.username;  userList.appendChild(li);  });  }  //Prompt the user before leave chat room  document.getElementById('leave-btn').addEventListener('click', () => {  const leaveRoom = confirm('Are you sure you want to leave the chatroom?');  if (leaveRoom) {  window.location = '../index.html';  } else {  }  });  **server.js**  const path = require('path');  const http = require('http');  const express = require('express');  const socketio = require('socket.io');  const formatMessage = require('./utils/messages');  const {  userJoin,  getCurrentUser,  userLeave,  getRoomUsers,  addRoom  } = require('./utils/users');  const app = express();  const server = http.createServer(app);  const io = socketio(server);  // Set static folder  app.use(express.static(path.join(\_\_dirname, 'public')));  const botName = 'Horizon Bot';  // Run when client connects  io.on('connection', socket => {  socket.on('joinRoom', ({ username, room }) => {  const user = userJoin(socket.id, username, room);  socket.join(user.room);  // Welcome current user  socket.emit('message', formatMessage(botName, 'Welcome to Horizon!'));  // Broadcast when a user connects  socket.broadcast  .to(user.room)  .emit(  'message',  formatMessage(botName, `${user.username} has joined the chat`)  );  // Send users and room info  io.to(user.room).emit('roomUsers', {  room: user.room,  users: getRoomUsers(user.room)  });  });  // Listen for chatMessage  socket.on('chatMessage', msg => {  const user = getCurrentUser(socket.id);  io.to(user.room).emit('message', formatMessage(user.username, msg));  });  // Runs when client disconnects  socket.on('disconnect', () => {  const user = userLeave(socket.id);  if (user) {  io.to(user.room).emit(  'message',  formatMessage(botName, `${user.username} has left the chat`)  );  // Send users and room info  io.to(user.room).emit('roomUsers', {  room: user.room,  users: getRoomUsers(user.room)  });  }  });  });  const PORT = process.env.PORT || 3000;  server.listen(PORT, () => console.log(`Server running on port ${PORT}`));  **messages.js** |  |

const moment = require('moment');

function formatMessage(username, text) {

return {

username,

text,

time: moment().format(' h:mm a')

};

}

module.exports = formatMessage;

**users.js**

const users = [];

// Join user to chat

function userJoin(id, username, room) {

const user = { id, username, room };

users.push(user);

return user;

}

//Add new room

function addRoom(room){

rooms.push(room);

return room;

}

// Get current user

function getCurrentUser(id) {

return users.find(user => user.id === id);

}

// User leaves chat

function userLeave(id) {

const index = users.findIndex(user => user.id === id);

if (index !== -1) {

return users.splice(index, 1)[0];

}

}

// Get room users

function getRoomUsers(room) {

return users.filter(user => user.room === room);

}

module.exports = {

userJoin,

getCurrentUser,

userLeave,

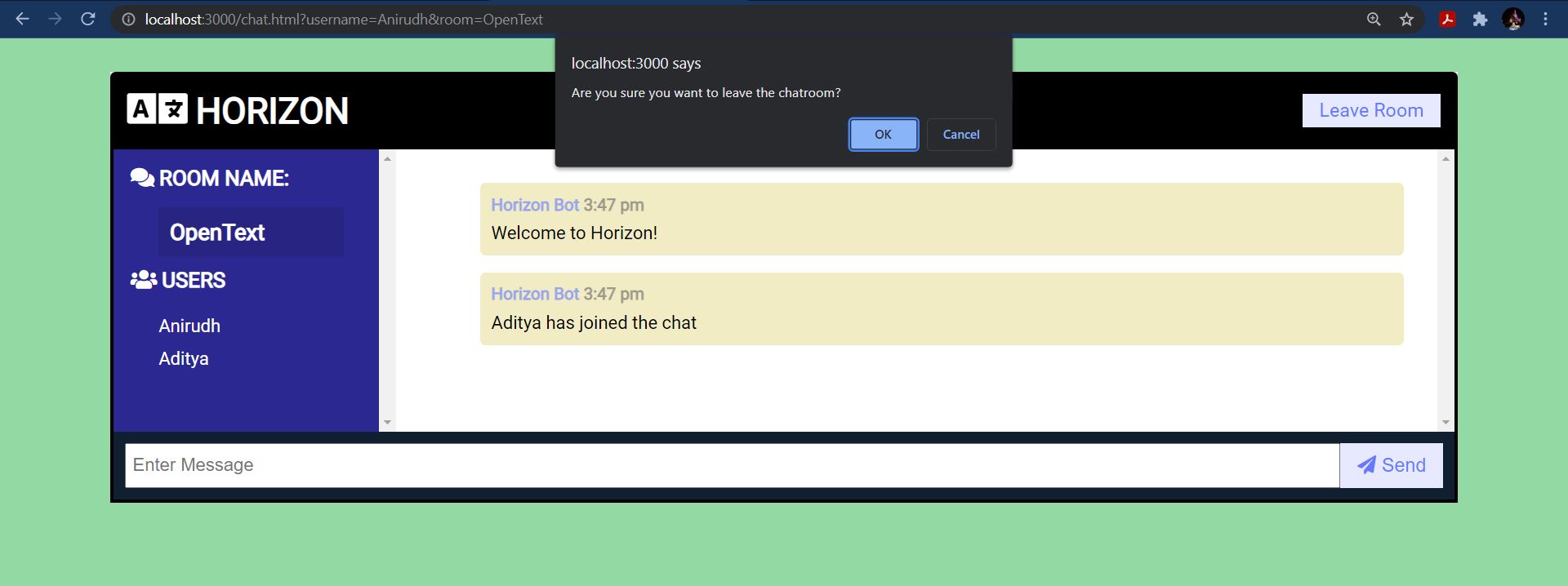
getRoomUsers,

addRoom

};

**6. TESTING AND VALIDATION**

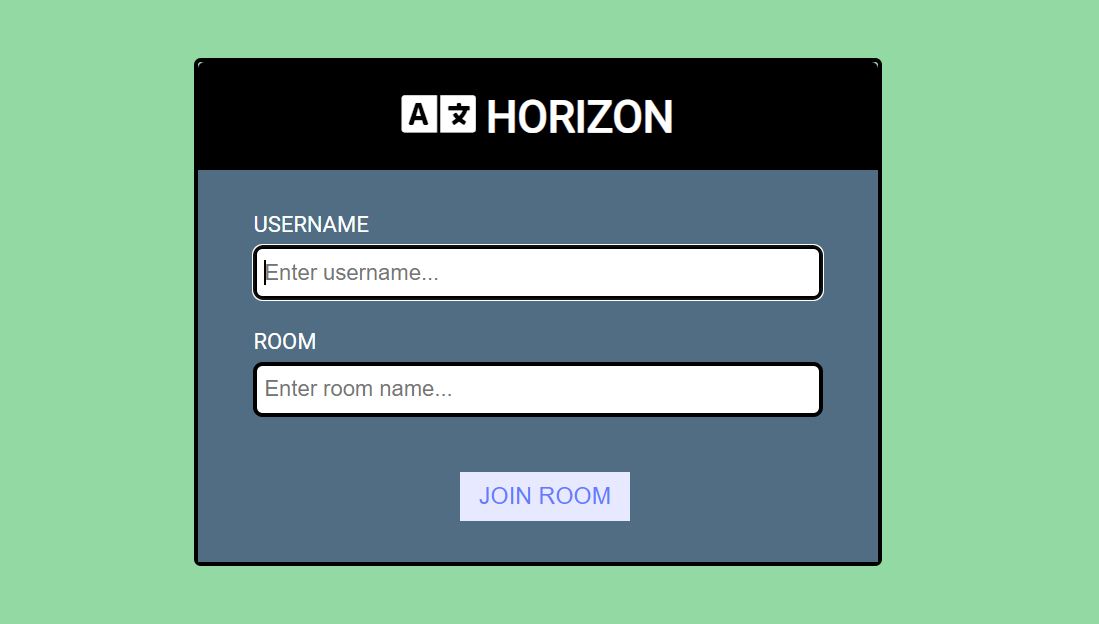
**6.1 Testing Types:**

****

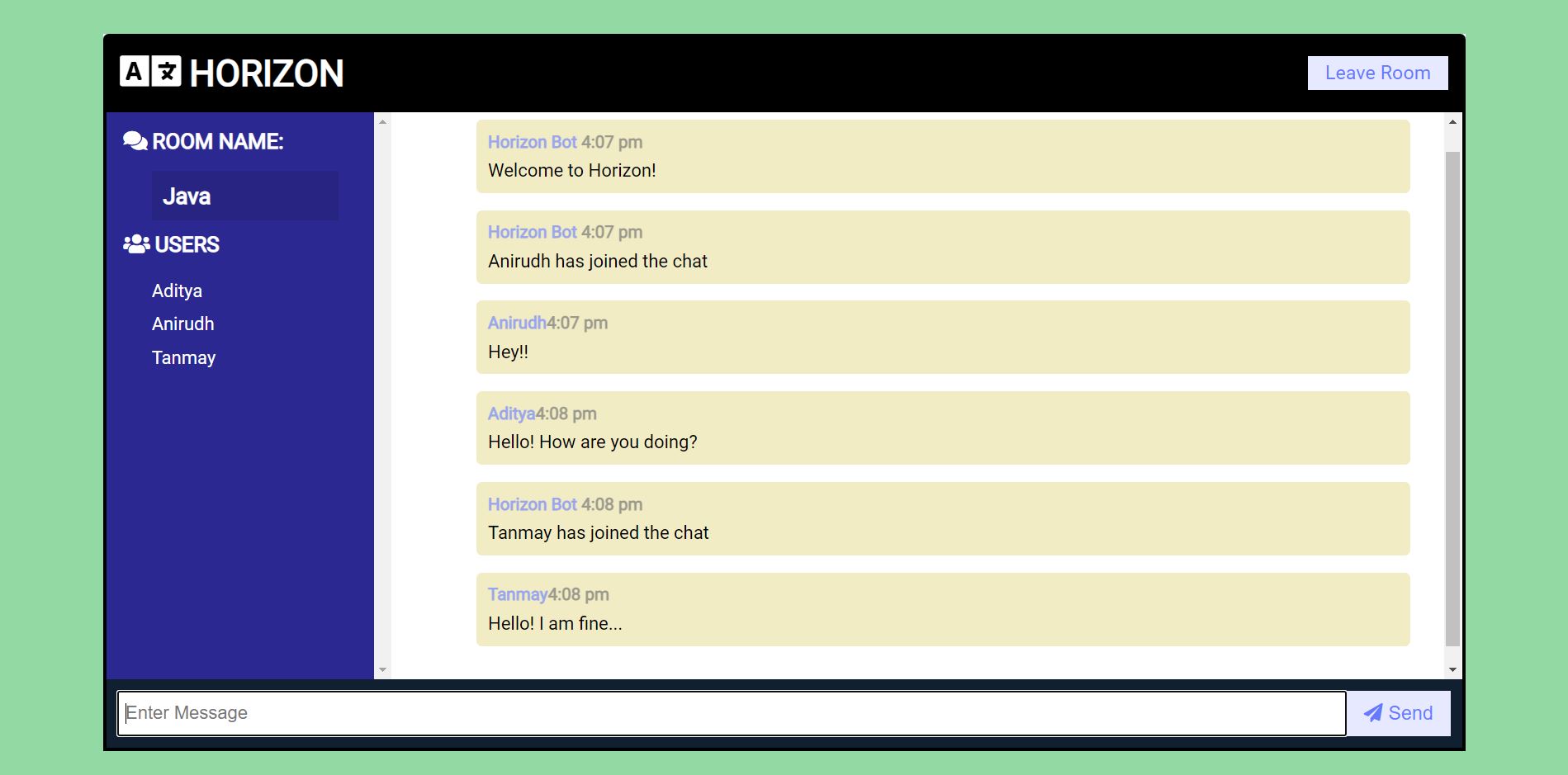
* The page will prompt an alert message to confirm if the user wants to leave the room.
* If the user clicks OK, only then he will leave the room. In this case, the user will be redirected to the homepage (index.html).

**6.2 Project screenshots and results:**

**a) Home Page**



**b) Chat Window:**



**7. CONCLUSION AND FUTURE ENHANCEMENTS:**

**Conclusion:**

* Since we are using Web Sockets, the instantaneous communication among a group of users can be achieved flawlessly.

**Future Enhancements:**

* This application can be made more functional by adding a user authentication and giving the user the option to make rooms private by using authentication and other security measures.
* There can be admins or control groups for each room to maintain the discipline in the chat environment.
* Support for other media like files, music, videos, images etc can be added.

**8. REFERENCES:**

* HTML & CSS: <https://www.w3schools.com/>
* Bootstrap: https://getbootstrap.com
* Socket.io : https://socket.io/docs/v4/
* Moment :https://www.webfx.com/blog/web-design/javascript-dates-moment-js/
* Node JS: https://nodejs.org/api/documentation.html
* Express JS: https://expressjs.com/
* Javascript: https://developer.mozilla.org/en-US/docs/Web/JavaScript
* Font Awesome: https://fontawesome.com/v4.7/icon/language
* Colour Palette: https://colorhunt.co/
* Flaticon: https://www.flaticon.com/uicons/interface-icons
* Draw.io: https://app.diagrams.net/
* Wikipedia: https://en.wikipedia.org/wiki/Chat\_room