CHAT-ROOMS

Project Overview

Introduction

The Chat Room Application is a real-time chat platform designed to facilitate communication among multiple users. Users can create and join chat rooms, send messages, and interact in a shared environment. The application includes features such as room creation, joining, exiting, and deletion (restricted to the room owner). The backend is powered by Node.js, Express, and MongoDB, while the frontend is built with React and styled using React-Bootstrap.

Objective

The primary objective of this project is to develop a responsive, user-friendly chat application that supports real-time communication and room management. The application should handle multiple users, ensure data persistence, and maintain a clean and modern UI.

Technologies Used

- **Frontend:**
  - React
  - React-Bootstrap
  - Axios
  - Socket.IO-client
- **Backend:**
  - Node.js
  - Express
  - Socket.IO
  - MongoDB (Mongoose)
- **Styling:**
  - React-Bootstrap
  - Custom CSS

Project Structure

Frontend

- `src/`
  - `components/`
    - `JoinRoom.js`
    - `RoomList.js`
    - `Chat.js`
  - `App.js`

    - `index.js`
    - `Chat.css`

Backend

- `server.js`

Functionalities

1. Room Creation and Joining

- Users can create a new chat room or join an existing one by providing a username and room name.
- The room creator becomes the owner and gains the ability to delete the room.

2. Real-time Messaging

- Users can send and receive messages in real-time within the joined room.
- Messages are displayed in a chat box with the sender's username.

3. Room Management

- Only the room owner can delete the room.
- Users can exit the room, which will notify others in the room.

User Interface

Home Screen

- Displays options to create or join a room.
- Users input their username and room name.

Chat Screen

- Displays the room name and a chat box.
- Messages are shown with the sender's username.
- Includes buttons for sending messages, exiting the room, and (for the owner) deleting the room.

User Roles

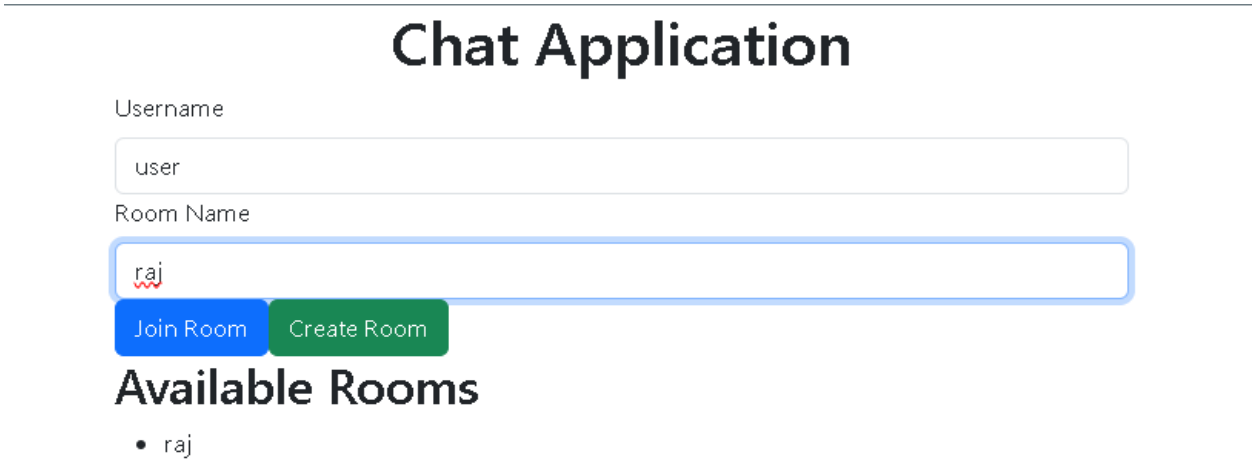
- **Room Creator (Owner):**
    - Can delete the room.
    - Can send and receive messages.

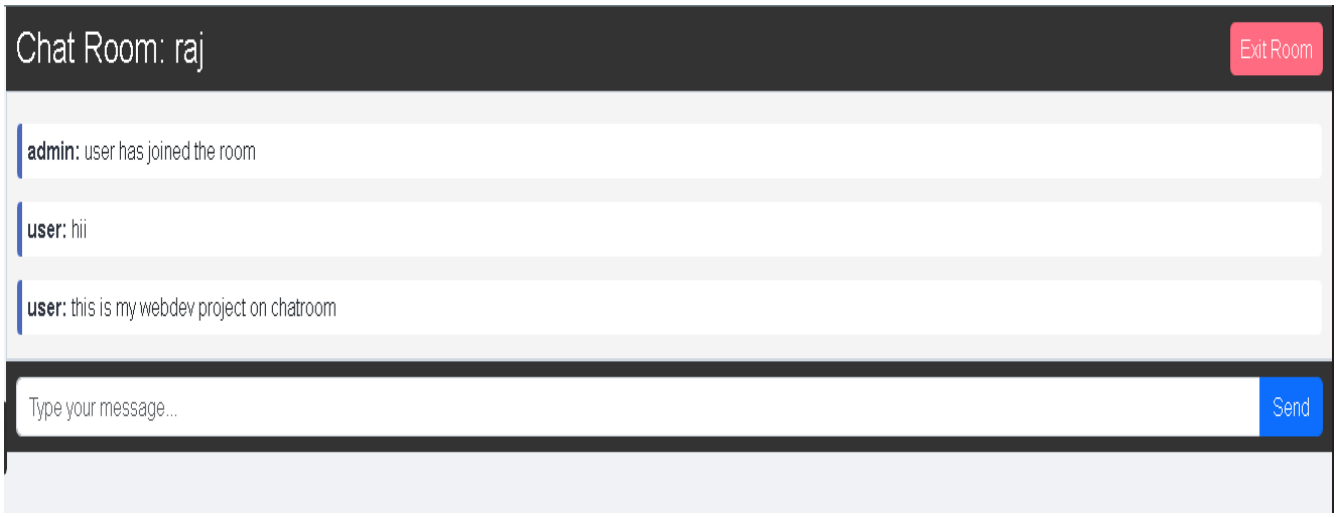
- **Regular User:**
  - Can send and receive messages.
  - Can exit the room.

Flow of Data

1. **Room Creation/Joining:**
   - The user inputs their username and room name.
   - If creating a room, a POST request is made to the backend to create a new room in the database.
   - If joining a room, the user is added to the room's user list.
2. **Messaging:**
   - Messages are sent via Socket.IO to the backend.
   - The backend broadcasts the message to all users in the room.
3. **Room Deletion:**
   - The room owner can delete the room, which removes it from the database and notifies all users.

Implementation





Security and Error Handling

- Usernames within a room are unique to prevent conflicts.
- Error messages are displayed for duplicate usernames, existing room names, and unauthorized room deletion attempts.

Conclusion

The Chat Room Application is a robust and scalable platform for real-time communication. It leverages modern web technologies to provide a seamless user experience with efficient data handling and real-time updates. The project's structure supports further enhancements and scalability, making it a suitable foundation for more complex chat applications or additional features.