# Name : Akanksha Gupta Redg : 11714899

Project : Web Chat Application Submitted To : Neha Sharma Github Link :

Web Chat Application

## Functionality

1. User Can create their chat rooms by simply entering the user name and unique room name.
2. Users can join the respectives rooms by entering their user names and the exact room name which they want to join
3. Finally room members can able to chat in real time web application.

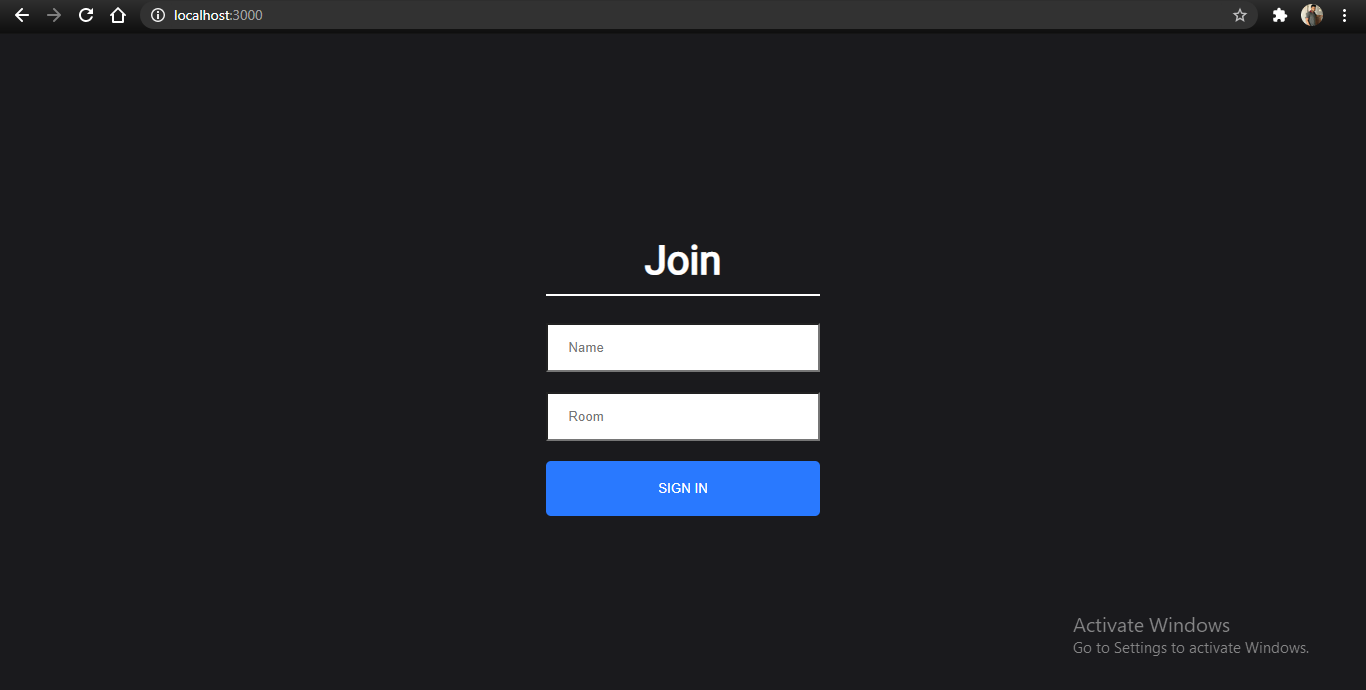
Technology used for real time environment

Socket.io  Writing a chat application with popular web applications stacks like LAMP (PHP) has normally been very hard. It involves polling the server for changes, keeping track of timestamps, and it’s a lot slower than it should be.

Sockets have traditionally been the solution around which most real-time chat systems are architected, providing a bi-directional communication channel between a client and a server.

This means that the server can push messages to clients. Whenever you write a chat message, the idea is that the server will get it and push it to all other connected clients.

1. Join Chat Room Page



Code for Server.js

const http = require('http');

const express = require('express'); const socketio = require('socket.io'); const cors = require('cors');

const { addUser, removeUser, getUser, getUsersInRoom } = require('./users'); const router = require('./router');

const app = express();

const server = http.createServer(app); const io = socketio(server);

app.use(cors()); app.use(router);

io.on('connect', (socket) => {

socket.on('join', ({ name, room }, callback) => {

const { error, user } = addUser({ id: socket.id, name, room });

if(error) return callback(error);

socket.join(user.room);

socket.emit('message', { user: 'admin', text: `${user.name}, welcome to room

${user.room}.`});

socket.broadcast.to(user.room).emit('message', { user: 'admin', text: `${user

.name} has joined!` });

io.to(user.room).emit('roomData', { room: user.room, users: getUsersInRoom(us er.room) });

callback();

});

socket.on('sendMessage', (message, callback) => { const user = getUser(socket.id);

io.to(user.room).emit('message', { user: user.name, text: message });

callback();

});

socket.on('disconnect', () => {

const user = removeUser(socket.id);

if(user) {

io.to(user.room).emit('message', { user: 'Admin', text: `${user.name} has l eft.` });

io.to(user.room).emit('roomData', { room: user.room, users: getUsersInRoom( user.room)});

}

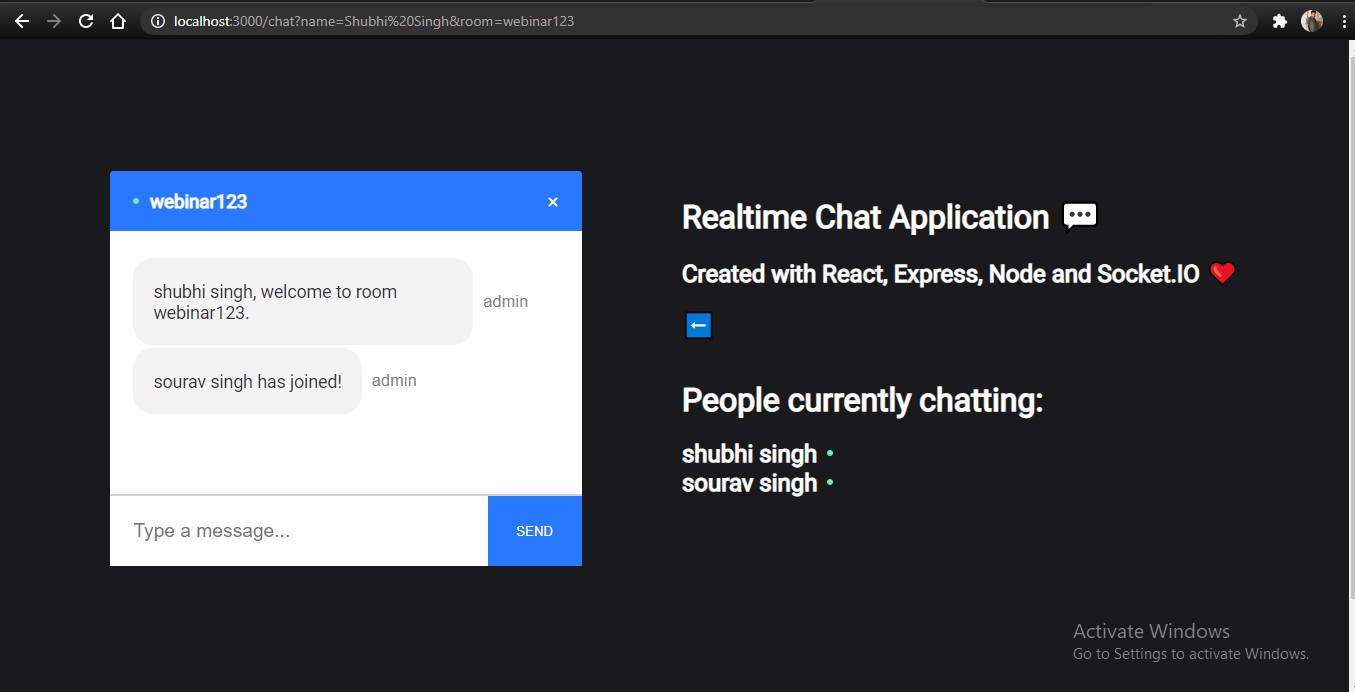
})

});

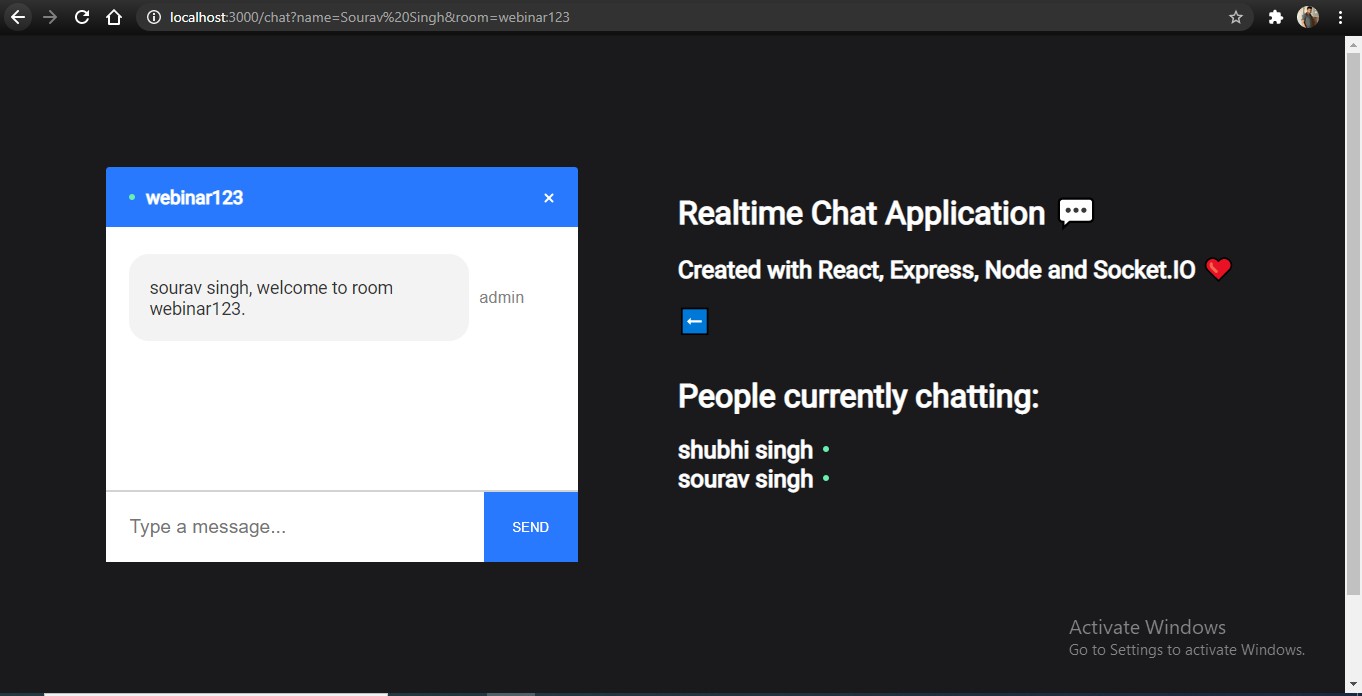
server.listen(process.env.PORT || 5000, () => console.log(`Server has started.`))

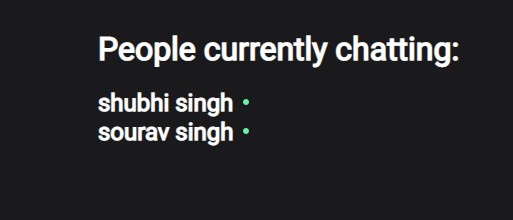
;

## User chat page [first user]



1. User chat Page [second user]



1. Online status for active users

Users validation to add userto chat box Page and chat room joining logic

const users = [];

const addUser = ({ id, name, room }) => { name = name.trim().toLowerCase();

room = room.trim().toLowerCase();

const existingUser = users.find((user) => user.room === room && user.name === n ame);

if(!name || !room) return { error: 'Username and room are required.' }; if(existingUser) return { error: 'Username is taken.' };

const user = { id, name, room }; users.push(user);

return { user };

}

const removeUser = (id) => {

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

if(index !== -1) return users.splice(index, 1)[0];

}

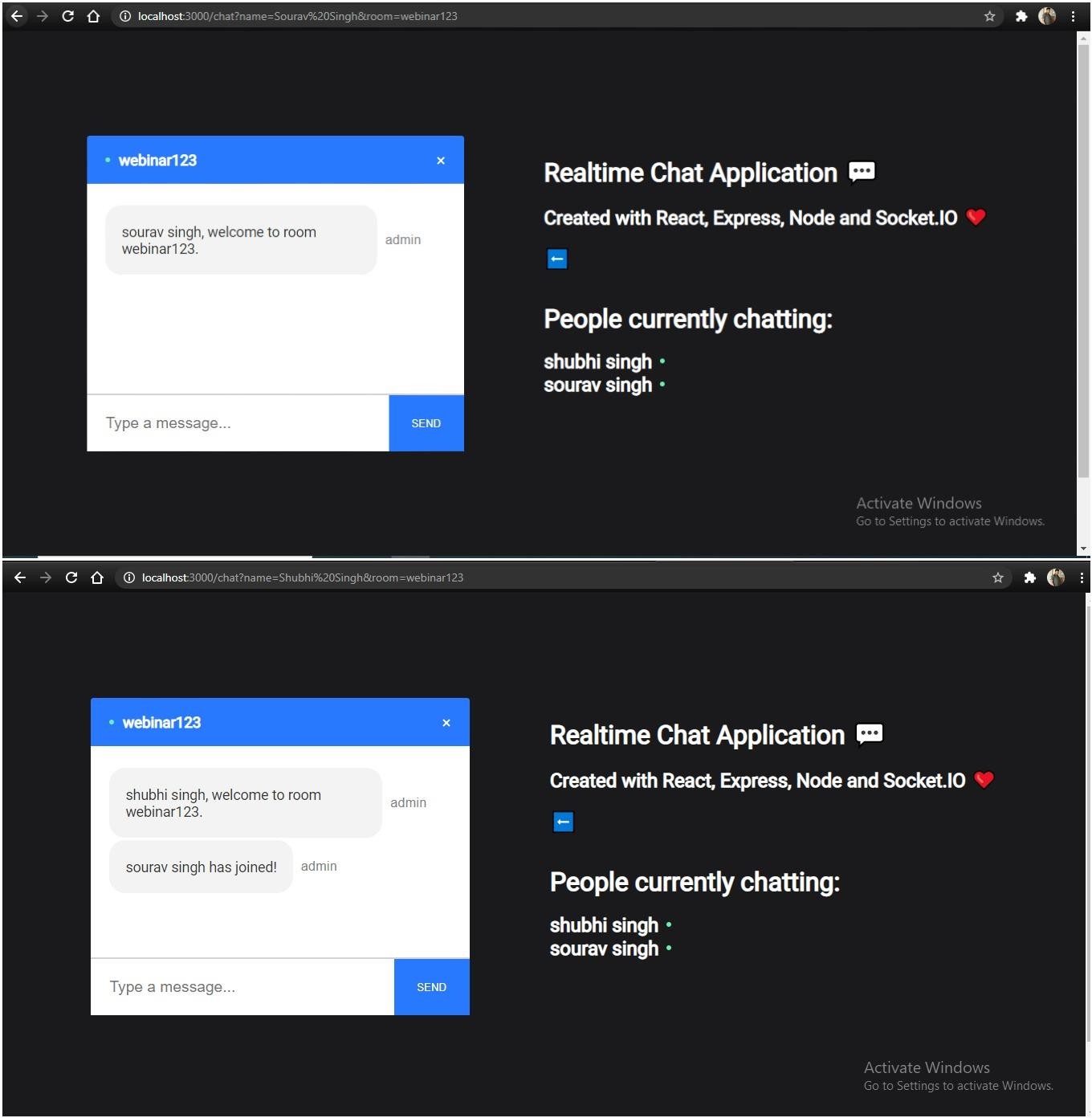
const getUser = (id) => users.find((user) => user.id === id);

const getUsersInRoom = (room) => users.filter((user) => user.room === room); module.exports = { addUser, removeUser, getUser, getUsersInRoom };

CORS  **Cross-Origin Resource Sharing (**[**CORS**](https://developer.mozilla.org/en-US/docs/Glossary/CORS)**) is a mechanism that uses additional** [**HTTP**](https://developer.mozilla.org/en-US/docs/Glossary/HTTP) **headers to tell browsers to give a web application running at**

**one** [**origin**](https://developer.mozilla.org/en-US/docs/Glossary/origin)**, access to selected resources from a different origin. A web application executes a cross-origin HTTP request when it requests a resource that has a different origin (domain, protocol, or port) from its own.**

1. Real time web chat Page from both ends



Thankyou