# Using for Project Commands

We have created a project structure using this command: mkdir webrtc-chat

We have created json file using this command for package creation : npm init –y

We must install socker using this command: npm install express socket.io

We must install this package for Multi chats: npm install cors

Multiple Chats

# Server Code (server.js)

const express = require("express");

const http = require("http");

const { Server } = require("socket.io"); // Use the recommended `Server` class

const app = express();

const server = http.createServer(app);

const io = new Server(server, {

cors: {

origin: "\*", // Allow all origins; customize as needed

methods: ["GET", "POST"]

}

});

// Serve static files (like HTML, CSS, JS)

app.use(express.static(\_\_dirname)); // Serves files from the directory where this script is located

// WebSocket event handlers

io.on("connection", (socket) => {

console.log("A user connected:", socket.id);

// Listen for incoming chat messages

socket.on("chat message", (msg) => {

console.log("Message received:", msg);

// Broadcast message to all other connected clients

socket.broadcast.emit("chat message", msg);

});

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

console.log("A user disconnected:", socket.id);

});

});

// Start the server

server.listen(3000, () => {

console.log("Server running on <http://192.168.1.12:3000>");

});

# UI code(index.html)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Socket.IO Chat</title>

<style>

body { font-family: Arial, sans-serif; margin: 20px; }

#messages { border: 1px solid #ccc; padding: 10px; height: 200px; overflow-y: scroll; }

#messageInput { width: 80%; padding: 5px; }

#sendButton { padding: 5px; }

</style>

</head>

<body>

<h1>Socket.IO Chat</h1>

<div id="messages"></div>

<input id="messageInput" placeholder="Type a message" />

<button id="sendButton">Send</button>

<script src="/socket.io/socket.io.js"></script>

<script>

const socket = io("http://192.168.1.11:3000"); // Connect to the server

// Handle incoming messages

socket.on("chat message", (msg) => {

const messagesDiv = document.getElementById("messages");

const message = document.createElement("div");

message.textContent = msg;

messagesDiv.appendChild(message);

messagesDiv.scrollTop = messagesDiv.scrollHeight; // Auto-scroll

});

// Send message on button click

document.getElementById("sendButton").addEventListener("click", () => {

const messageInput = document.getElementById("messageInput");

const msg = messageInput.value;

if (msg.trim() !== "") {

socket.emit("chat message", msg); // Send message to server

messageInput.value = ""; // Clear input field

}

});

</script>

</body>

</html>

We must use Admin IP address for multiple chats.

# P2P Connection (For Single Person)

# Server Code(server.js)

const express = require("express");

const http = require("http");

const { Server } = require("socket.io"); // Use the recommended `Server` class

const app = express();

const server = http.createServer(app);

const io = new Server(server, {

cors: {

origin: "\*", // Allow all origins; customize as needed

methods: ["GET", "POST"]

}

});

// Serve static files (like HTML, CSS, JS)

app.use(express.static(\_\_dirname)); // Serves files from the directory where this script is located

// WebSocket event handlers

io.on("connection", (socket) => {

console.log("A user connected:", socket.id);

// Listen for incoming chat messages

socket.on("chat message", (msg) => {

console.log("Message received:", msg);

// Broadcast message to all other connected clients

socket.broadcast.emit("chat message", msg);

});

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

console.log("A user disconnected:", socket.id);

});

});

// Start the server

server.listen(3000, "192.168.1.11", () => {

console.log("Server running on <http://192.168.1.12:3000>");

});

# UI Code(index.html)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Socket.IO Chat</title>

<style>

body { font-family: Arial, sans-serif; margin: 20px; }

#messages { border: 1px solid #ccc; padding: 10px; height: 200px; overflow-y: scroll; }

#messageInput { width: 80%; padding: 5px; }

#sendButton { padding: 5px; }

</style>

</head>

<body>

<h1>Socket.IO Chat</h1>

<div id="messages"></div>

<input id="messageInput" placeholder="Type a message" />

<button id="sendButton">Send</button>

<script src="/socket.io/socket.io.js"></script>

<script>

const socket = io("http://192.168.1.11:3000"); // Connect to the server

// Handle incoming messages

socket.on("chat message", (msg) => {

const messagesDiv = document.getElementById("messages");

const message = document.createElement("div");

message.textContent = msg;

messagesDiv.appendChild(message);

messagesDiv.scrollTop = messagesDiv.scrollHeight; // Auto-scroll

});

// Send message on button click

document.getElementById("sendButton").addEventListener("click", () => {

const messageInput = document.getElementById("messageInput");

const msg = messageInput.value;

if (msg.trim() !== "") {

socket.emit("chat message", msg); // Send message to server

messageInput.value = ""; // Clear input field

}

});

</script>

</body>

</html>

# Me to Me Chat:

# Server Code(server.js)

const express = require("express");

const http = require("http");

const socketIo = require("socket.io");

const app = express();

const server = http.createServer(app);

const io = socketIo(server);

// Serve static files (like HTML, CSS, JS)

app.use(express.static(\_\_dirname));

io.on("connection", (socket) => {

console.log("A user connected");

// Broadcast incoming message to all other clients

socket.on("chat message", (msg) => {

console.log("Message received: " + msg);

socket.broadcast.emit("chat message", msg); // send to all other clients

});

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

console.log("A user disconnected");

});

});

server.listen(3000, () => {

console.log("Server running on <http://localhost:3000>");

});

# UI Code(index.html)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>WebRTC Chat</title>

<style>

#chat {

width: 300px;

height: 400px;

border: 1px solid black;

margin: 20px auto;

padding: 10px;

overflow-y: scroll;

}

#messageInput {

width: 80%;

padding: 5px;

}

#sendButton {

padding: 5px;

width: 15%;

}

</style>

</head>

<body>

<h2>WebRTC Chat</h2>

<div id="chat"></div>

<input type="text" id="messageInput" placeholder="Type a message" />

<button id="sendButton">Send</button>

<script src="/socket.io/socket.io.js"></script>

<script>

const socket = io(); // Connect to the server

// Send message when the send button is clicked

document.getElementById("sendButton").onclick = () => {

const message = document.getElementById("messageInput").value;

if (message) {

socket.emit("chat message", message); // Send message to the server

document.getElementById("messageInput").value = ""; // Clear input field

}

};

// Receive and display messages from the server

socket.on("chat message", (msg) => {

const chatDiv = document.getElementById("chat");

const newMessage = document.createElement("div");

newMessage.textContent = msg;

chatDiv.appendChild(newMessage);

// Automatically scroll to the latest message

chatDiv.scrollTop = chatDiv.scrollHeight;

});

</script>

</body>

</html>