First, the server file

Code:-

const express = require('express');

const http = require('http');

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

const path = require('path');

const app = express();

const server = http.createServer(app);

const io = socketIo(server);

const PORT = process.env.PORT || 3000;

app.use(express.static(path.join(\_\_dirname, 'public')));

const users = new Map();

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

console.log('New user connected');

socket.on('join', (username) => {

users.set(socket.id, username);

io.emit('userJoined', username);

io.emit('userList', Array.from(users.values()));

});

socket.on('chatMessage', (message) => {

const username = users.get(socket.id);

io.emit('message', {

username,

text: message,

timestamp: new Date().getTime()

});

});

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

const username = users.get(socket.id);

socket.broadcast.emit('userTyping', username);

});

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

const username = users.get(socket.id);

users.delete(socket.id);

io.emit('userLeft', username);

io.emit('userList', Array.from(users.values()));

console.log('User disconnected');

});

});

server.listen(PORT, () => console.log(`Server running on port ${PORT}`));

Next, the HTML file:-

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Real-time Chat Application</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div id="chat-container">

<div id="chat-header">

<h2>Chat Room</h2>

</div>

<div id="user-list"></div>

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

<div id="typing-indicator"></div>

<form id="chat-form">

<input type="text" id="username-input" placeholder="Your name" required>

<input type="text" id="message-input" placeholder="Type your message..." required>

<button type="submit">Send</button>

</form>

</div>

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

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

</body>

</html>

The CSS file:-

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

background-color: #f0f0f0;

}

#chat-container {

width: 600px;

height: 800px;

background-color: white;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

display: flex;

flex-direction: column;

}

#chat-header {

background-color: #4CAF50;

color: white;

padding: 10px;

text-align: center;

border-top-left-radius: 10px;

border-top-right-radius: 10px;

}

#user-list {

padding: 10px;

background-color: #e6e6e6;

border-bottom: 1px solid #ccc;

}

#chat-messages {

flex-grow: 1;

overflow-y: auto;

padding: 10px;

}

#typing-indicator {

padding: 5px;

font-style: italic;

color: #666;

}

#chat-form {

display: flex;

padding: 10px;

}

#username-input,

#message-input {

flex-grow: 1;

margin-right: 10px;

padding: 5px;

}

button {

background-color: #4CAF50;

color: white;

border: none;

padding: 5px 10px;

cursor: pointer;

}

button:hover {

background-color: #45a049;

}

.message {

margin-bottom: 10px;

padding: 5px;

border-radius: 5px;

background-color: #e6e6e6;

}

.message .username {

font-weight: bold;

margin-right: 5px;

}

.message .timestamp {

font-size: 0.8em;

color: #666;

}

.system-message {

font-style: italic;

color: #666;

}

Finally, the JavaScript file:-

const socket = io();

const chatForm = document.getElementById('chat-form');

const chatMessages = document.getElementById('chat-messages');

const userList = document.getElementById('user-list');

const usernameInput = document.getElementById('username-input');

const messageInput = document.getElementById('message-input');

const typingIndicator = document.getElementById('typing-indicator');

let username = '';

chatForm.addEventListener('submit', (e) => {

e.preventDefault();

if (username === '') {

username = usernameInput.value.trim();

if (username) {

socket.emit('join', username);

usernameInput.value = '';

usernameInput.disabled = true;

}

} else {

const message = messageInput.value.trim();

if (message) {

socket.emit('chatMessage', message);

messageInput.value = '';

}

}

});

messageInput.addEventListener('input', () => {

socket.emit('typing');

});

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

displayMessage(message);

});

socket.on('userJoined', (user) => {

displaySystemMessage(`${user} has joined the chat`);

});

socket.on('userLeft', (user) => {

displaySystemMessage(`${user} has left the chat`);

});

socket.on('userList', (users) => {

userList.innerHTML = `<strong>Online Users:</strong> ${users.join(', ')}`;

});

socket.on('userTyping', (user) => {

typingIndicator.textContent = `${user} is typing...`;

setTimeout(() => {

typingIndicator.textContent = '';

}, 3000);

});

function displayMessage(message) {

const div = document.createElement('div');

div.classList.add('message');

div.innerHTML = `

<span class="username">${message.username}:</span>

<span class="content">${message.text}</span>

<span class="timestamp">${new Date(message.timestamp).toLocaleTimeString()}</span>

`;

chatMessages.appendChild(div);

chatMessages.scrollTop = chatMessages.scrollHeight;

}

function displaySystemMessage(message) {

const div = document.createElement('div');

div.classList.add('system-message');

div.textContent = message;

chatMessages.appendChild(div);

chatMessages.scrollTop = chatMessages.scrollHeight;

}

These files should be saved in their respective locations as mentioned earlier. Once you have these files in place, you can proceed with initializing the Node.js project and installing the necessary dependencies.

Certainly! I'll guide you through the process of running your server and accessing your chat application:

 Open a terminal in your project directory:

 On Windows:

 Open File Explorer and navigate to your project folder.

 Hold Shift and right-click in an empty area of the folder.

 Select "Open PowerShell window here" or "Open command window here".

 On macOS:

 Open Finder and navigate to your project folder.

 Right-click (or Control-click) on the folder.

 Select "New Terminal at Folder" or "Services > New Terminal at Folder".

 On Linux:

 Open your file manager and navigate to your project folder.

 Right-click in the folder and select "Open in Terminal" (the exact wording may vary depending on your Linux distribution).

Alternatively, you can open a terminal and use the cd command to navigate to your project directory:

   cd path/to/your/real-time-chat-app

 Run node server.js to start the server:

 In the terminal that you just opened in your project directory, type the following command and press Enter:

     node server.js

- If everything is set up correctly, you should see a message like:

     Server running on port 3000

- This means your server is now running and listening for connections on port 3000.

 Open a web browser and navigate to http://localhost:3000:

 Open your preferred web browser (Chrome, Firefox, Safari, Edge, etc.).

 In the address bar at the top of the browser window, type:

     http://localhost:3000

- Press Enter to navigate to this address.

If your server is running correctly and your files are in the right place, you should now see your chat application interface in the browser.

Additional notes:

 Keep the terminal window open while you're using the application. Closing the terminal will stop the server.

 If you make changes to your server.js file, you'll need to stop the server (by pressing Ctrl+C in the terminal) and start it again with node server.js for the changes to take effect.

 If you see any error messages in the terminal when trying to start the server, make sure all your files are in the correct locations and that you've installed all necessary dependencies (express and socket.io) by running npm install in your project directory.

Certainly! I'll guide you through initializing the Node.js project and installing the necessary dependencies. Here's the step-by-step process:

 Open a terminal in your project directory.

 Initialize a new Node.js project:

   npm init -y

This command creates a package.json file with default values.

 Install the necessary dependencies:

   npm install express socket.io

This command installs Express and Socket.io, saving them as dependencies in your package.json file.

 After the installation is complete, your package.json file should now include these dependencies. You can verify this by opening the file in a text editor.

 (Optional) If you want to use a tool like nodemon for automatic server restarts during development, you can install it as a dev dependency:

   npm install --save-dev nodemon

 If you installed nodemon, you can add a start script to your package.json. Open the file and add or modify the "scripts" section:

   "scripts": {

     "start": "node server.js",

     "dev": "nodemon server.js"

   }

 Your project is now initialized and has all necessary dependencies installed.

To run your server:

 If you didn't add the scripts or install nodemon, use:

  node server.js

- If you added the scripts, you can use:

  npm start

Or for development with automatic restarts:

  npm run dev

 Once your server is running, open a web browser and navigate to http://localhost:3000 to see your chat application.

Remember, you need to keep the terminal window open while using the application. If you close the terminal, it will stop the server.