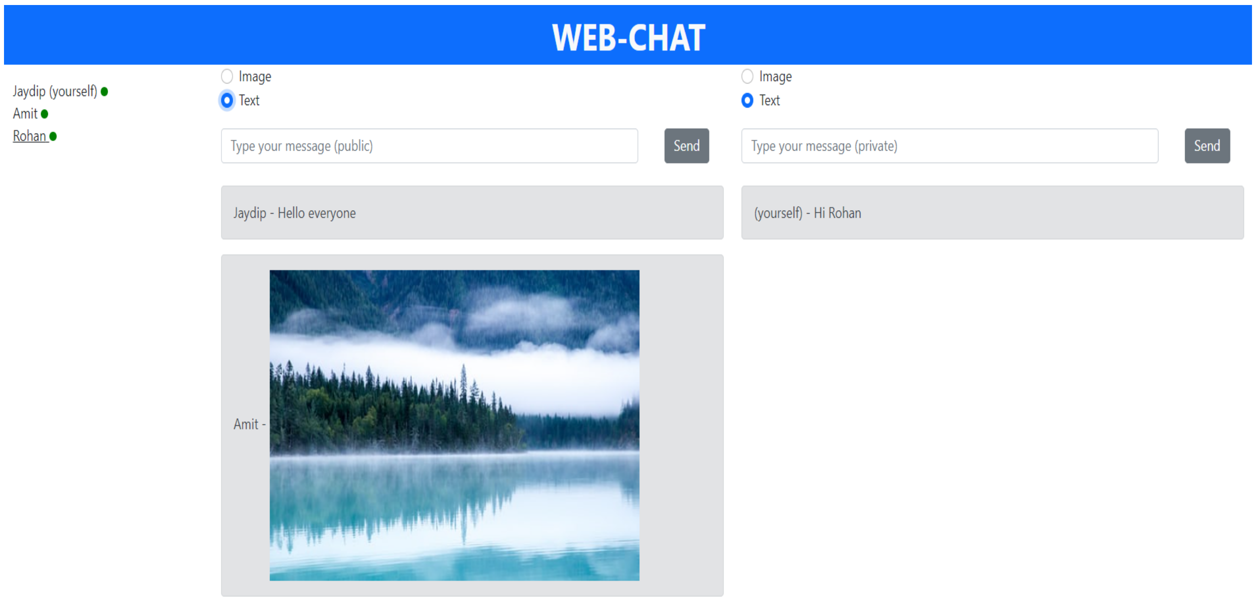

WEB-CHAT

Jaydip (yourself) ●
Amit ●
Rohan ●

Image
[✓ Text]

Type your message (public) Send

Jaydip - Hello everyone

Amit -

Image
[✓ Text]

Type your message (private) Send

(yourself) - Hi Rohan

Chat App

Project Link : https://github.com/jaydip1235/placewit-js/tree/master/WebChat

Jaydip Dey
Jadavpur University
https://linktr.ee/jaydipdey

This is a real-time chat application developed using React for the front-end, Node.js and Express for the back-end, and Socket.io for real-time bi-directional event-based communication, supporting both one-to-one private conversations and many-to-many group chats.

Features:

1. User can do group chat
2. User can do one to one chat
3. Both text and image formats are supported
4. User can see the number of unread message and which user is typing in global chat similar to Whatsapp
5. Notification of a user joining and leaving the room will me shown with the username

BACKEND

server.js

- We import and instantiate an Express.js application. Express.js is a web application framework for Node.js that helps with the organization of your web application into an MVC architecture.
- We enable CORS using the cors module. This allows the server to accept requests from different origins.
- We configure Socket.IO to use the created server and handle real-time communication. We also set the CORS options for Socket.IO, limiting the accepted requests to those from http://localhost:3000 and only allowing GET and POST methods.
- We import and call our chat controller with io (the instance of Socket.IO) as a parameter. This controller is expected to handle all chat-related functionalities like emitting and listening for events.

chat.js

Modules and Functions

This file exports a single function chat, which takes an instance of a Socket.IO server (io) as its argument. This function sets up the necessary middleware and event listeners for the chat server.

Middleware

- A middleware is used to check the username sent in the socket handshake. If a username is not provided, an error is thrown. If a username is provided, it's attached to the socket object for later use.

Event Listeners

The function sets up several event listeners:

- "connection": Fires when a new user connects to the server. It checks for existing users with the same username and if found, the user gets a "username taken" message and the socket is disconnected. If there are no conflicts, the user details are added to the users list and the "user connected" event is broadcasted to all users except the one who just joined.
- "message": Listens for message events from the client side. When a message event is received, it is immediately broadcast to all clients.
- "typing": Fires whenever a user starts typing a message. This event is broadcast to all other clients, signaling them that the user is typing.
- "private message": Fires when a private message is sent. It finds the recipient by their socket id, then sends the private message to that particular user.
- "disconnect": Fires when a user disconnects from the server. A "user disconnected" event is broadcasted to all other users.

FRONTEND

App.js

How to Use the App

1. Joining the chat

Upon loading the application, you are prompted to enter a username. Once you submit your username, the socket connects and you're officially part of the chat.

2. Public and Private Messaging

Once you've joined, you will find two sections: public chat and private chat.

In the public chat section, you can send a message or an image to everyone in the chat. The messages sent will appear for everyone in the chat, including yourself.

The private chat section is slightly different. Firstly, you need to select a user by clicking on a username from the user list on the left side of the page. After selecting a user, you can send them a private message or image. The message you send will only appear for you and the user you selected.

3. Users list

On the left side, you will see a list of all users currently connected to the chat. Your own username will be marked with "(yourself)" text next to it. The users' names will also show their connection status through a dot: green for online users and red for offline users. Users with new unread messages will be indicated with a text "___".

Main Components and Functionalities

This app is primarily built within a single functional component: App. Here are some key aspects of this code:

1. State Variables

The app maintains several pieces of state through the useState hook to keep track of various parts of the application such as the current user's username, connection status, messages, and more.

2. Socket Events

The application uses several socket events to manage different aspects of the chat. For instance, "user connected" and "user disconnected" events manage the addition and removal of users from the chat. The "message" event handles public messages, while "private message" handles private messages between users. The "typing" event provides live updates of users currently typing a message.

3. Event Handlers

Several event handlers manage the user input and interaction within the application, like handleUsername to set up the username, handleMessage and handlePrivateMessage to send public and private messages respectively, and handleUsernameClick to select a user for private messaging.

4. Message Types: Text or Image

The application provides the ability to send both text and image messages. For image messages, an upload input is used to receive an image file, which is then converted into a base64 string using the convertBase64 function. This base64 string can then be sent as a message and displayed as an image in the chat.

5. Rendering

The rendering part of the component shows different forms and messages based on the state of the application. For example, if the user hasn't connected yet, it shows the form to enter a username. After the user connects, it shows the chat interface.