Live Chat Application Docs

1. In this application there will be two servers the one will be the frontend server and the other will be the node server
2. Make an folder named the **js** and init make an file named the **client.js** in that, if required make an css folder and make an file in it and operate the styles of the frontend from there
3. Make all the frontend part completed
4. Than in it make a folder named the nodeServer
5. And than open the terminal and change the folder by writing the **cd nodeServer**
6. And than install the socket.io in it by writing the **npm install socket.io**
7. Than make a new file in the nodeServer folder named the index.js and in it we will write the backend code on the website
8. First initialize the socket.io by writing the following statement:

**const io=require(‘socket.io’)(8000);**

🡪Here instead of the 8000, any of the port can be used

1. Than we are running an socket.io server which is an instance of the http server and it attaches with the http itself
2. Here io.on is a socket.io instance which will listen many of the socket connections like connected by harry, divya and etc
3. socket.on will decide what to do to on a particular connection when some activity is performed by thar particular connection



🡪So that, the **users[socket.id]=name;** will take in the name of the user that joined the chat and store it in the socket.id

🡪And the **socket.broadcast.emit(‘user-joined’, name);** will broadcast the message to the all the users that are already present in the chat that new user joined

🡪Here the new-user-joined is not predefined, its name can be anything wish

1. Than we will make a next function for the sending the message to all, After doing that the code would look like:



🡪And if any one sends the message than we will broadcast to all that the message has been arrived

🡪Here the all the words are not any command or the function, they can be any thing

1. Than to check our code that it is working rightly, we will make the nodemon install by writing the **npm install nodemon** and than we will run it by writing the **nodemon .\index.js** and it it runs and does not crash than our code is right (make sure that the nodemon .\index.js is written in the terminal of the backend folder i.e. in the nodeServer folder and if the terminal is running on the other folder than by the help of the **cd** command change the location of the folder)
2. Now lets come to the frontend part and we will make the connection between the frontend and the node sever, lets us forget the backend/node part for some time
3. Now to make the connection between the backend and frontend, we will make an javascript file to be added in the index.html(of the frontend), that file would be:

**<script defer src=”http://localhost:8000/socket.io/socket.io.js”></script>**

🡪Here the socket.io.js will be included in the package automatically when we install the socket.io in the project

1. And than also include the javascript file that we have made in the js folder and by name client.js, Example

**<script defer src=”js/client.js”></script>**

1. Make sure that the client’s script is after the server’s script, as we have to first make the connection between them, And than use the client side frontend
2. Now lets go to the client.js which is on the js folder and in it we will write the following statements:

**const socket=io(‘http://localhost:8000’);**

**const form=document.getElementById(‘send-container’);**

**const messageInput=document.getElementById(‘messageInp’);**

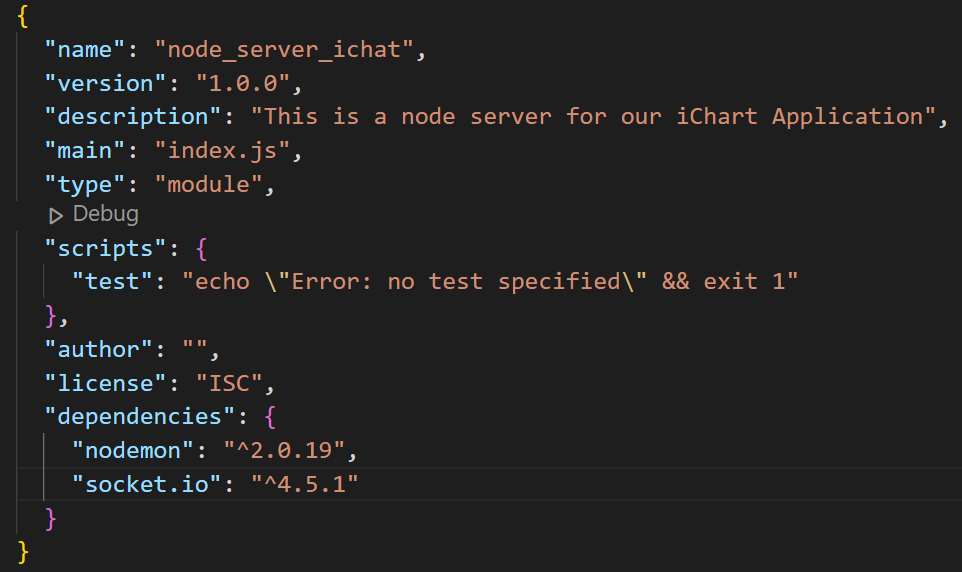
**const messageContainer=document.querySelector(“.container”);**

**const name=prompt(“Enter your name to join”);**

**socket.emit(‘new-user-joined’, name);**

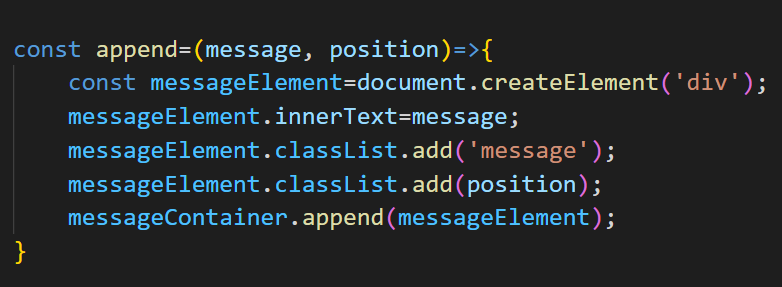
🡪Here, the socket will make an emit to the backend which would be new-user-joined and will make the process that is written in it, and the backend will also get an variable named name

1. Here if the error comes like that of the import is not allowed outside the container than the make some changes in the package.json like as follow:

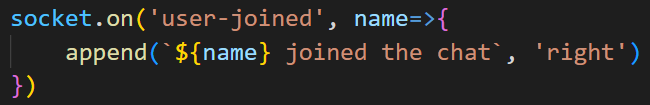


🡪And in the package.json we have added the **“type”: “module”,** and by doing this the error would be gone

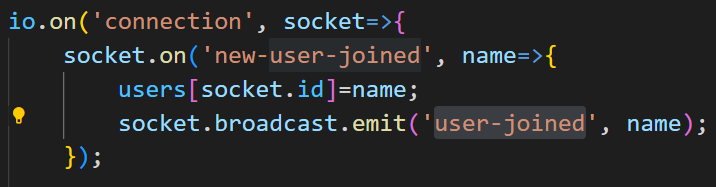
1. Now an another error comes of the CORS which is discussed in the detail in the CORS Info in the Info folder, In this project to overcome this error we use an chrome extension named the Moesif Origin & CORS Changer which is make on while using this website and than when the devops work is completed at that time this extension is made off
2. Than the website starts working
3. Now we will work on the appending the message that the new user joined the chat to all the existing memners by makking the following changes in the client.js
4. For not making the node\_modules files in the git we make an .gitignore file and in it we add the node\_modules
5. Now to append the message in the chat box we will write the following code in the client.js



🡪And the upward one is the append function which will be called when the following user-joined function is called



🡪And this user-joined function is called from the index.js which is on the nodeServer folder i.e. on the backend ApI and this will be called at the instatnt when the last line of the new-user-joined function is called

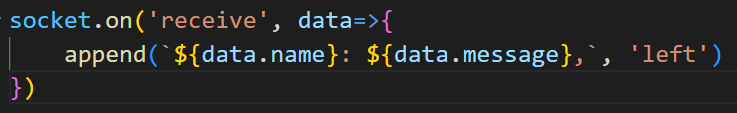


🡪And this new-user-joined function is called or emited from the client.js i.e. frontend part when the alert is submited, hence this forms an loop

1. Now we will work on the new functinality of the sending the message from one to all, for that we will first go to the index.js of the nodeServer and write the following code in it

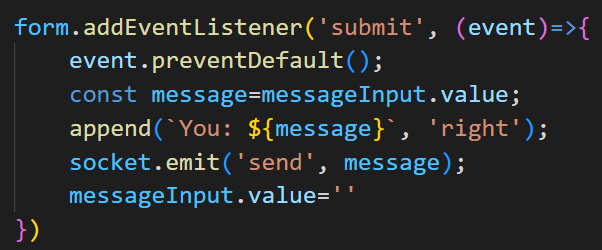


🡪Than we will make the receive function in the client.js by writing the following code in the client.js



🡪And we have to also make some changes in the form when we submit the form, we have to put the eventListener when we click on the submit

🡪And from the client.js we can add an eventListener which is be acted when the submit is been called from the form and an event will be maked to run and we will make also the use of the preventDefault to prevent the any of the change in the website

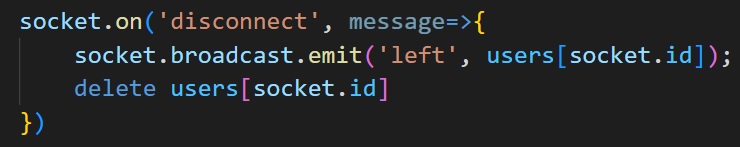


🡪And than we will save the value in the massageInput in the variable named the message

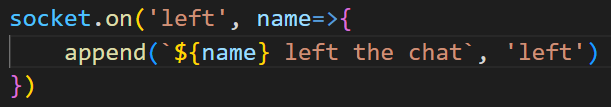
🡪Than will append the value by the help of the append fucntion

🡪Than will emit the send fucntion and make the value of the messageInput null

1. In the **socket.io** when any user leaves the socket than an event named the disconnect is been fired by the socket.io
2. For making an notifications to all that an user has leaved the socket we will make an function for it in the index.js in the nodeServer folder, Example



🡪This function will detect the disconnect automatically when the user leaves, and than will broadcast and emit the left and than the delete the user from the users array

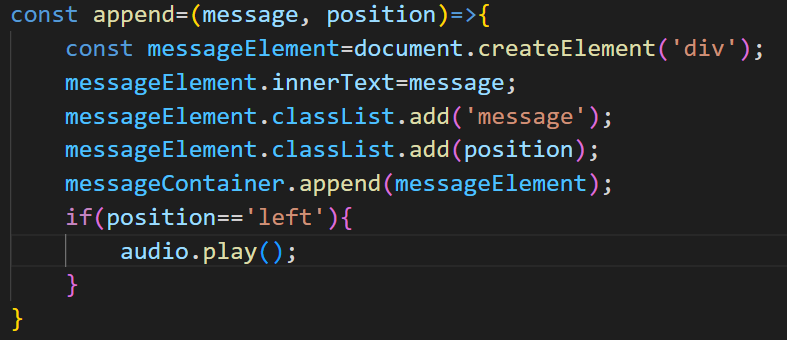


🡪And than will make the left function on the client side and than will append the left message in the text box

1. Than we will make an notification when the message will be arrived to the any of the user for it we have to download the .mp3 file of the sound
2. And than in the client.js will write the following statement on the top of the file with other import statments:

**var audio=new Audio(‘ting.mp3’);**

1. And than make the audio to play by writing the **audio.play();** where we want to play the audio, Example



🡪And we will make the condition that if the message is on the left or center than only to ring the sound otherwise not by help of the if statment