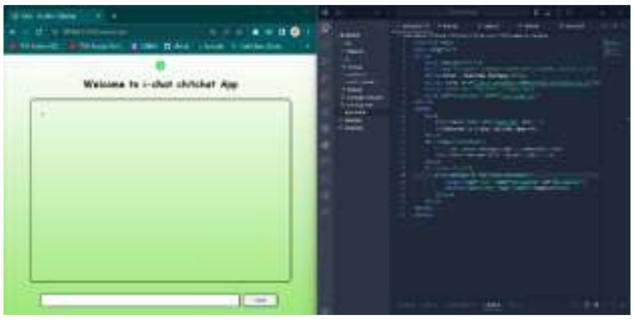
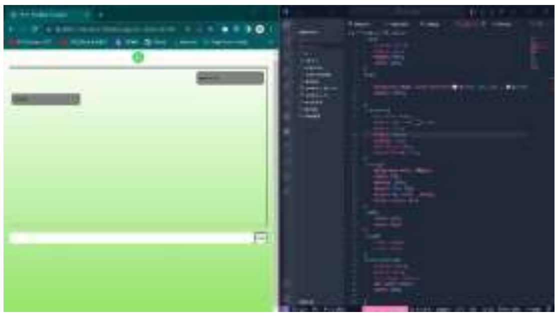
HOW I CREATED A CHAT APP WITH NODEJS AND SOCKET IO?





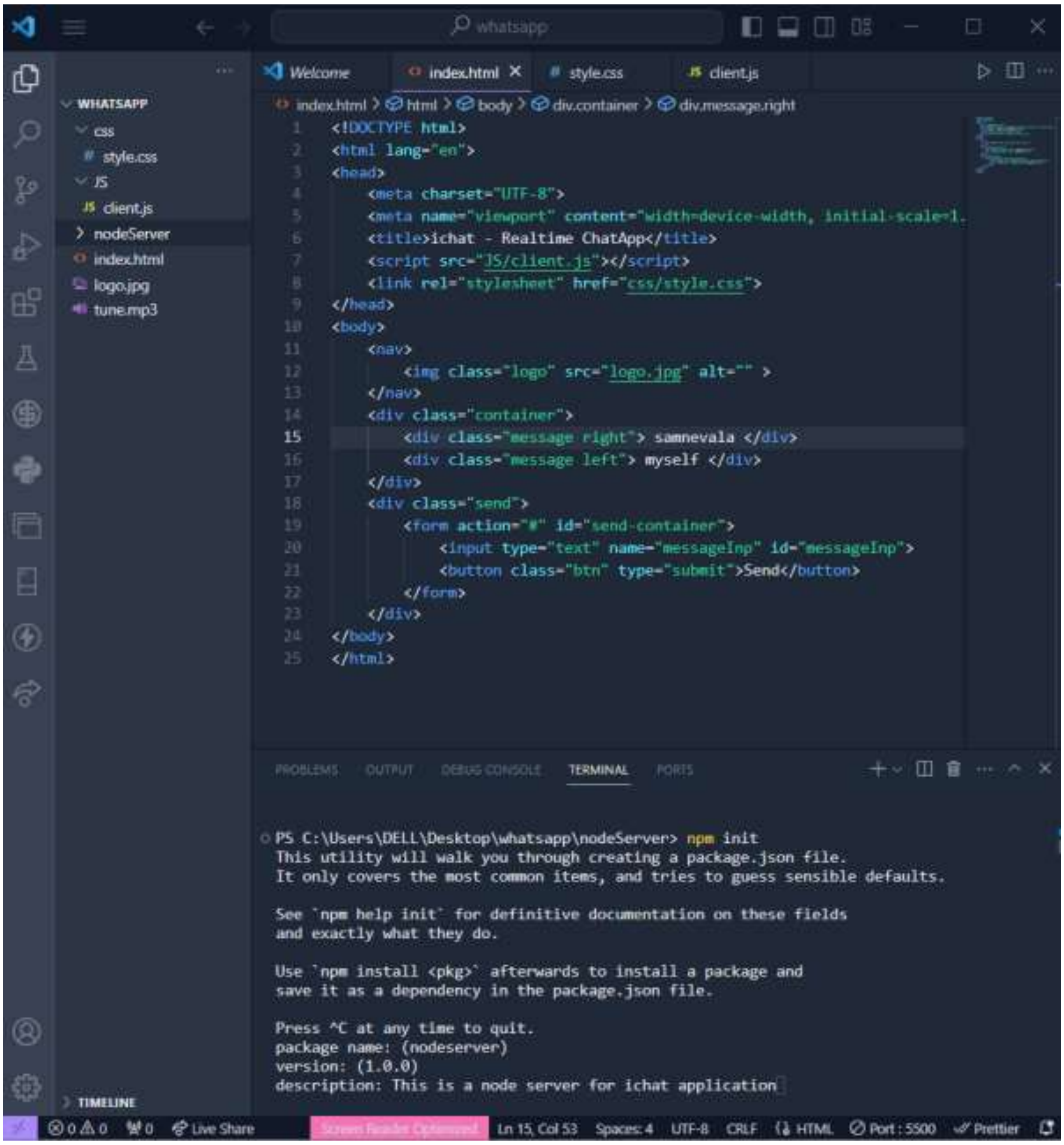
The front end is completed first, which is how I want my chat application to use HTML for the information you want to make visible on the website and CSS to style the information, such as where you want to view what, as shown in the following steps.

- The first step would be to display all of the items in the body by setting the height of the body to 100vh, which will cover the entire page. I've also created a backdrop of green and white using CSS styling called linear-gradient.

- Second, add a logo and centre it with CSS styling by using display: flex and margin auto to move the logo to the centre of the page. Determine the height and width of your logo.

- Then, create a container in which you'll be building left and right message boxes, so that when a user sends you a message, it appears on the right side and when you send a message, it appears on the left side.

- After that, the send box is formed, which will direct you to send the message you typed in the input bar. This send box is made up of two div : messageinp , where you type the message you want to send, and button, which is a send button click, where the message you wrote is sent to the user and appears on the left side of the container.

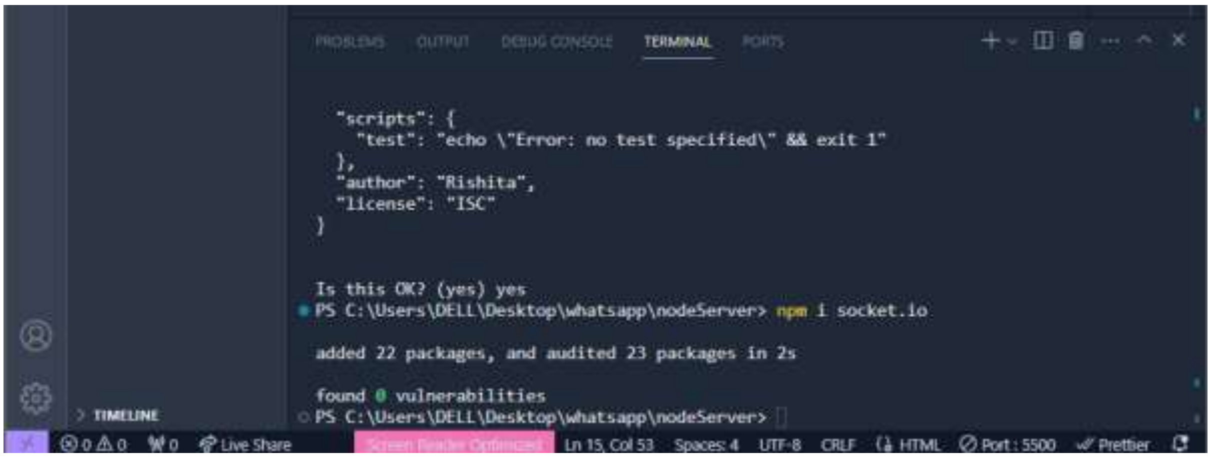


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <title>ichat - Realtime ChatApp</title>
    <script src="JS/client.js"></script>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <nav>
        <img class="logo" src="logo.jpg" alt="" >
    </nav>
    <div class="container">
        <div class="message right"> samnevala </div>
        <div class="message left"> myself </div>
    </div>
    <div class="send">
        <form action="#" id="send-container">
            <input type="text" name="messageInp" id="messageInp">
            <button class="btn" type="submit">Send</button>
        </form>
    </div>
</body>
</html>
```

```
PS C:\Users\DELL\Desktop\whatsapp\nodeServer> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (nodeserver)
version: (1.0.0)
description: This is a node server for ichat application
```

- Now, create a folder called nodeServer and open a terminal window by typing cd within the nodeServer folder. Write the command, for example, npm init (node package manager); this npm init command will guide you through the process of establishing a package.json file. This package.json file is significant because it stores important metadata about a project that is required before publishing to NPM. It also defines functional attributes of a project that npm uses to install dependencies, run scripts, and identify the package's entry point.

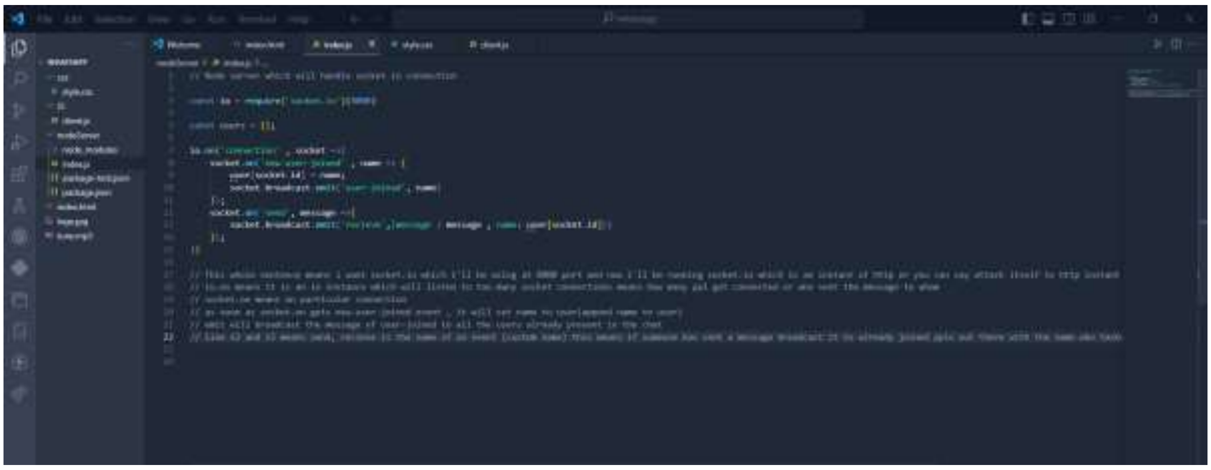
- Some of the question asked will be description that you wanna give to your website . About the author i.e who is the author / creater and many more .



```
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Rishita",
  "license": "ISC"
}

Is this OK? (yes) yes
PS C:\Users\DELL\Desktop\whatsapp\nodeServer> npm i socket.io

added 22 packages, and audited 23 packages in 2s

found 0 vulnerabilities
PS C:\Users\DELL\Desktop\whatsapp\nodeServer>
```

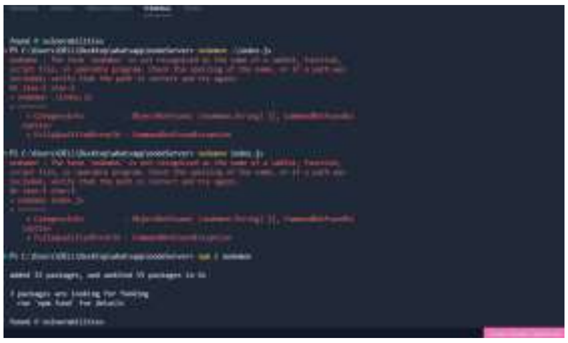
- npm i socket.io helps install the socket.io to be used in NodeJs.

THE JS COMMAND FOR SERVER SIDE WITH EXPLAINATION

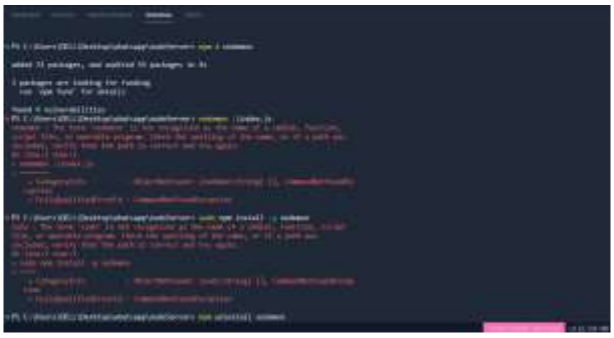


```
// Node server which will handle socket io connection

const io = require('socket.io')(8000)

const users = {};

io.on('connection', socket =>{
    socket.on('new-user-joined', name =>{
        users[socket.id] = name;
        socket.broadcast.emit('user-joined', name)
    });
    socket.on('send', message =>{
        socket.broadcast.emit('receive',{message : message , name: users[socket.id]})
    });
})

// This whole sentence means i used socket.io which i'll be using at 8000 port and now i'll be running socket.io which is an instance of http [illegible]
// io.on means it is an io instance which will listen to too many socket connections [illegible]
// socket.on means on particular connection
// as soon as socket.on gets new-user-joined event , it will set name to users[append name to user]
// emit will broadcast the message of user-joined to all the users already present in the chat
// line 12 and 13 means send, receive is the name of an event [custom name] this means if someone has sent a message broadcast [illegible]
```

DIFFICULTIES FACED DURING RUNNING THE SERVER SIDE

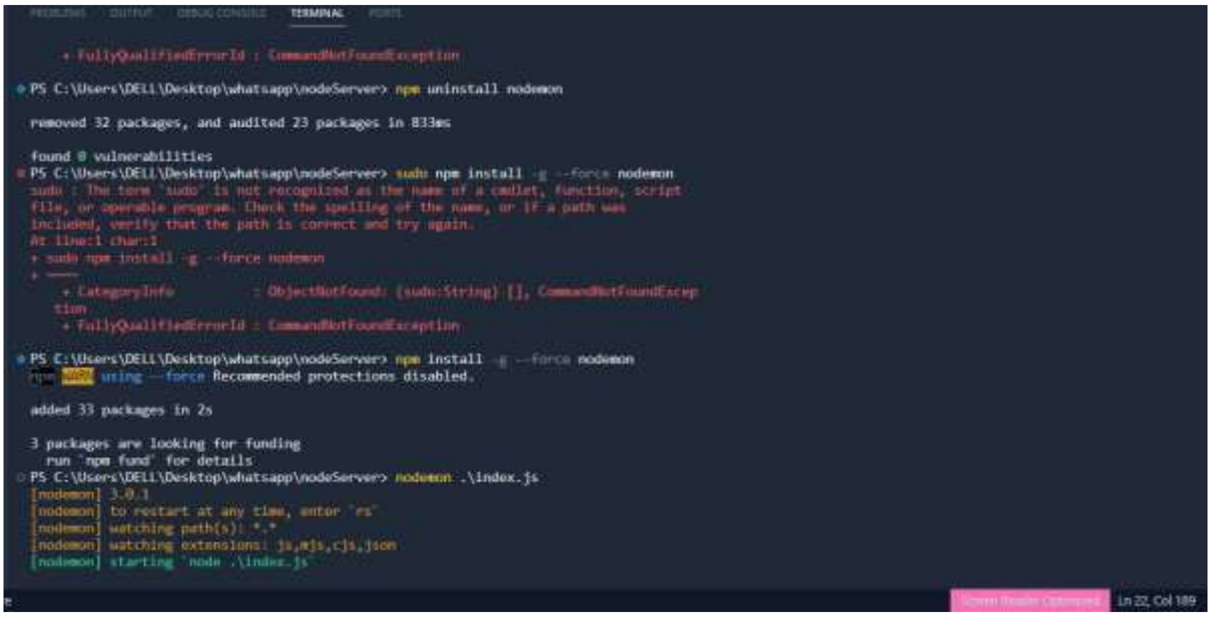


```
found 0 vulnerabilities
PS C:\Users\DELL\Desktop\whatsapp\nodeServer> nodemon .\index.js
nodemon : The term 'nodemon' is not recognized as the name of a cmdlet, function,
script file, or operable program. Check the spelling of the name, or if a path was
included, verify that the path is correct and try again.
[illegible]
PS C:\Users\DELL\Desktop\whatsapp\nodeServer> nodemon index.js
nodemon : The term 'nodemon' is not recognized as the name of a cmdlet, function,
script file, or operable program. Check the spelling of the name, or if a path was
included, verify that the path is correct and try again.
[illegible]
PS C:\Users\DELL\Desktop\whatsapp\nodeServer> npm i nodemon

added [illegible] packages, and audited [illegible] packages in [illegible]

[illegible] packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```



```
PS C:\Users\DELL\Desktop\whatsapp\nodeServer> npm i nodemon

added [illegible] packages, and audited [illegible] packages in [illegible]

[illegible] packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\DELL\Desktop\whatsapp\nodeServer> nodemon .\index.js
nodemon : The term 'nodemon' is not recognized as the name of a cmdlet, function,
script file, or operable program. Check the spelling of the name, or if a path was
included, verify that the path is correct and try again.
[illegible]
PS C:\Users\DELL\Desktop\whatsapp\nodeServer> sudo npm install -g nodemon
sudo : The term 'sudo' is not recognized as the name of a cmdlet, function, script
file, or operable program. Check the spelling of the name, or if a path was
included, verify that the path is correct and try again.
[illegible]
PS C:\Users\DELL\Desktop\whatsapp\nodeServer> npm uninstall nodemon
```

HOW TO RESOLVE THE EROOR

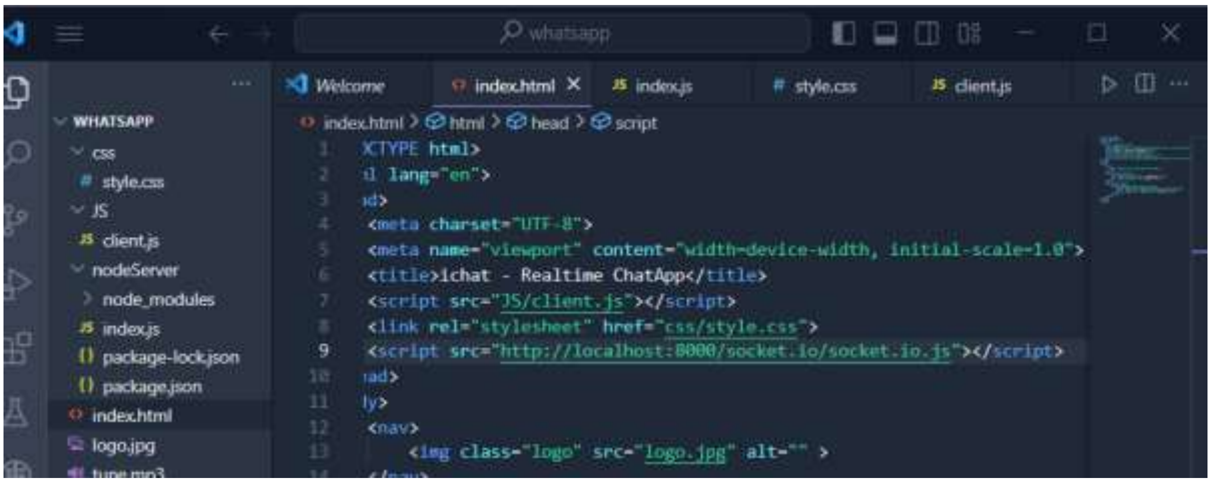


```
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\DELL\Desktop\whatsapp\nodeServer> npm uninstall nodemon

removed 32 packages, and audited 23 packages in 833ms

found 0 vulnerabilities
PS C:\Users\DELL\Desktop\whatsapp\nodeServer> sudo npm install -g --force nodemon
sudo : The term 'sudo' is not recognized as the name of a cmdlet, function, script
file, or operable program. Check the spelling of the name, or if a path was
included, verify that the path is correct and try again.
At line:1 char:1
+ sudo npm install -g --force nodemon
+ ~~~~
    + CategoryInfo          : ObjectNotFound: (sudo:String) [], CommandNotFoundExcep
   tion
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\DELL\Desktop\whatsapp\nodeServer> npm install -g --force nodemon
npm WARN using --force Recommended protections disabled.

added 33 packages in 2s

3 packages are looking for funding
  run `npm fund` for details
PS C:\Users\DELL\Desktop\whatsapp\nodeServer> nodemon .\index.js
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node .\index.js`
```

You'll have to first install the nodemon tool that will develop Node.js based application by automatically restarting the application whenever there is a change in the file or directory to do this follow the following steps

- npm install -g –force nodemon
- nodemon filename as abode

BUILD CLIENT – SERVER CONNECTION

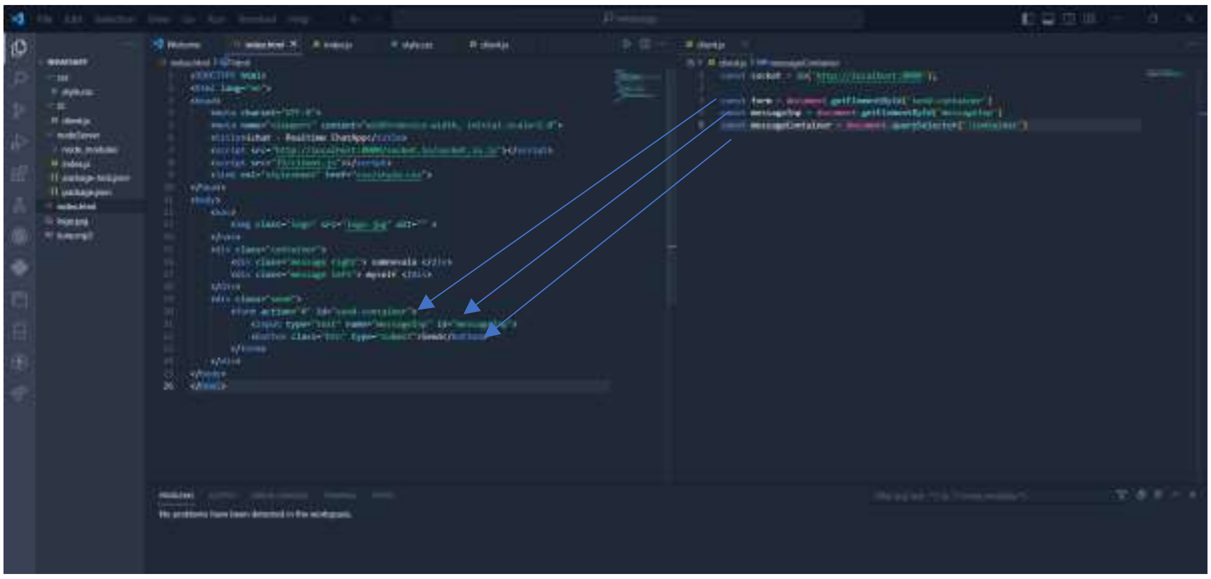


```
XCTYPE html>
:l lang="en">
ad>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>iChat - Realtime ChatApp</title>
<script src="JS/client.js"></script>
<link rel="stylesheet" href="css/style.css">
<script src="http://localhost:8000/socket.io/socket.io.js"></script>
ad>
ly>
<nav>
    <img class="logo" src="logo.jpg" alt="" >
```

<script src = https://localhost:8000/socket.io/socket.io.js></script>

- This above Script helps building the connection between client side and server side that is who sends the message and who receives the message.
- Make sure you wrote the above command before src = "JS/client.js" part

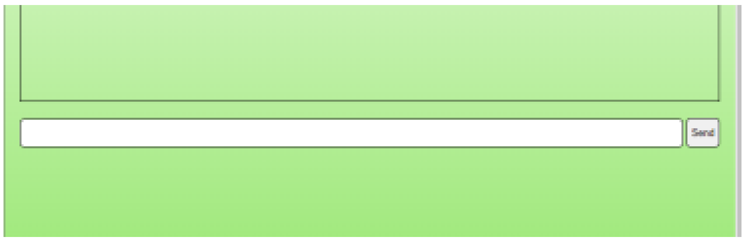
MESSAGE SENDING PART



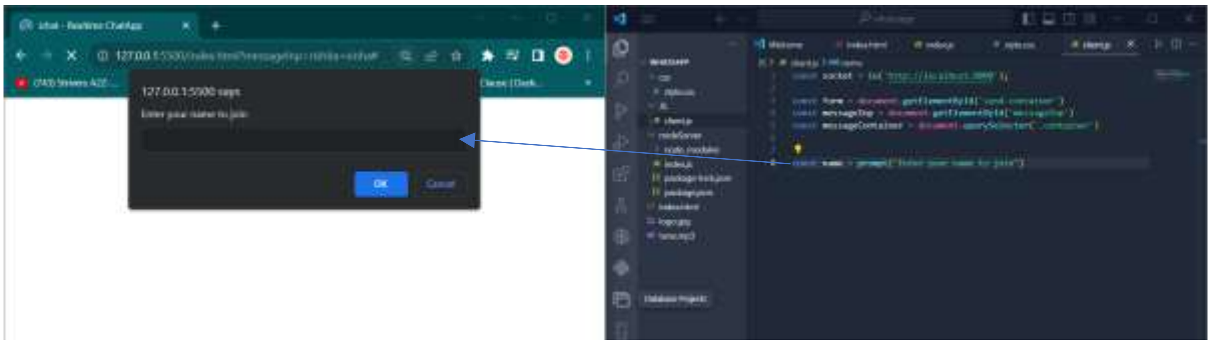
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>iChat - Realtime ChatApp</title>
    <script src="http://localhost:8000/socket.io/socket.io.js"></script>
    <script src="js/client.js" defer></script>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <nav>
        <img class="logo" src="logo.jpg" alt="" >
    </nav>
    <div class="container">
        <div class="message right"> [illegible] </div>
        <div class="message left"> [illegible] </div>
    </div>
    <div class="send">
        <form action="#" id="send-container">
            <input type="text" name="messageInp" id="messageInp">
            <button class="btn" type="submit">Send</button>
        </form>
    </div>
</body>
</html>
```

```javascript
const socket = io('http://localhost:8000');

const form = document.getElementById('send-container');
const messageInp = document.getElementById('messageInp');
const messageContainer = document.querySelector('.container');
```

- These arrow shows how we connected client side to html side i.e.whenever form, will go to send-container at html side
- Again, messageInp will go to messageinput side which is a box we made to input message that needs to be sent



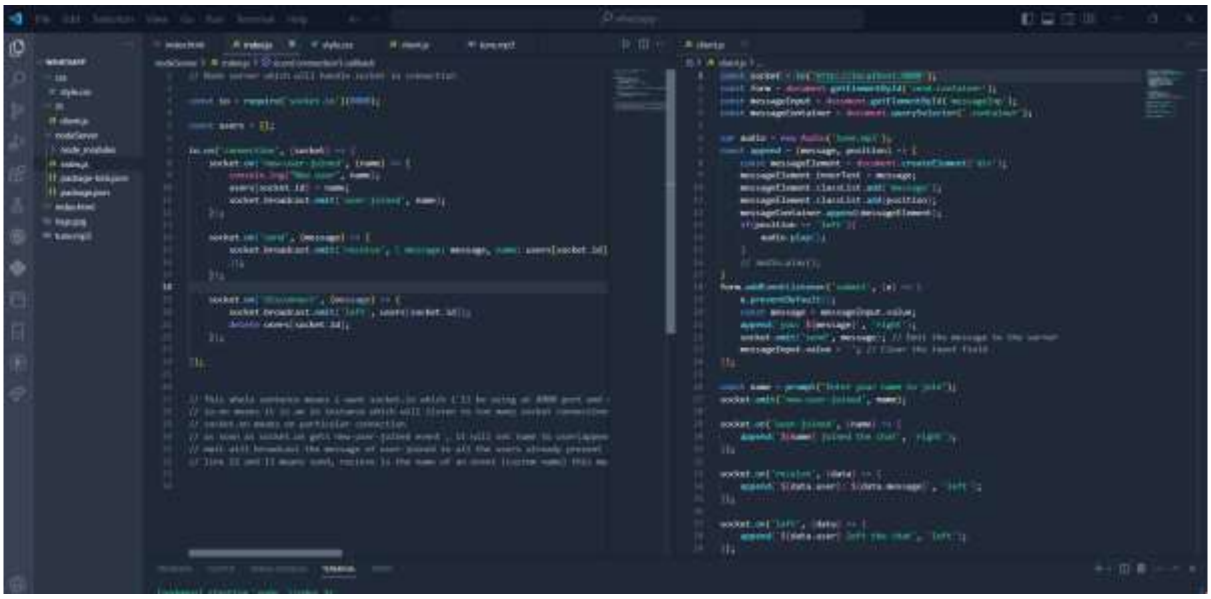
- This send button will send the message or broadcast the input message to the users joined out there.



```javascript
const socket = io('http://localhost:8000');

const form = document.getElementById('send-container');
const messageInp = document.getElementById('messageInp');
const messageContainer = document.querySelector('.container');

const name = prompt("Enter your name to join");
```

This command will help you join by your name taking it as an input unless you write your name and click ok you won't be able to join the chat.

HOW TO ADD THE JS PART



```
// Node server which will handle socket io connections

const io = require('socket.io')(8000);

const users = {};

io.on('connection', (socket) => {
    socket.on('new-user-joined', (name) => {
        console.log("New user", name);
        users[socket.id] = name;
        socket.broadcast.emit('user-joined', name);
    });

    socket.on('send', (message) => {
        socket.broadcast.emit('receive', { message: message, name: users[socket.id]
        });
    });

    socket.on('disconnect', (message) => {
        socket.broadcast.emit('left', users[socket.id]);
        delete users[socket.id];
    });

});

// This whole sentence means I want socket.io which I'll be using at 8000 port and
// io.on means it is an io instance which will listen to the many socket connections
// socket.on means on particular connection
// as soon as socket.io gets new-user-joined event , it will set name to users[appen
// emit will broadcast the message of user joined to all the users already present
// [illegible]
```

```
const socket = io('http://localhost:8000');
const form = document.getElementById('send-container');
const messageInput = document.getElementById('messageInp');
const messageContainer = document.querySelector(".container");

var audio = new Audio('tune.mp3');
const append = (message, position) => {
    const messageElement = document.createElement('div');
    messageElement.innerText = message;
    messageElement.classList.add('message');
    messageElement.classList.add(position);
    messageContainer.append(messageElement);
    if(position == 'left'){
        audio.play();
    }
    // [illegible]
}
form.addEventListener('submit', (e) => {
    e.preventDefault();
    const message = messageInput.value;
    append(`You: ${message}`, 'right');
    socket.emit('send', message); // Sent the message to the server
    messageInput.value = ''; // Clear the input field
});

const name = prompt("Enter your name to join");
socket.emit('new-user-joined', name);

socket.on('user-joined', (name) => {
    append(`${name} joined the chat`, 'right');
});

socket.on('receive', (data) => {
    append(`${data.user}: ${data.message}`, 'left');
});

socket.on('left', (data) => {
    append(`${data.user} left the chat`, 'left');
});
```

- New-user-joined – whenever a user has joined , name of the user joined the chat text is up on the chat box.
- Broadcast means all the joined users would know if any change has occurred if the user has joined / left / sent a message in the chat box.
- Delete user helps deleting the user from DB as he/she has left the chat.
- Constant (const) name gives a top-up message to write the name of the user whosoever want to join the chat box.
- Append helps to add the changes that has happened during the chat when received a message / someone left the chat box / someone has joined the box.
- Message = message-Input value means write the message whatever you want to write / send to the chat-box.
- Position is for where and which message will appear i.e. if I'm sending someone a message it should appear on the left side whereas if someone sends me a message it should appear at the left side of the chat-box .
- Audio play plays the tune of receiving the message whenever the message has been received i.e. if the message appears at the right side of the chat-box it should play the sound.
- Event (e) – submit. Whenever you send a message in the chat-box and click to the submit button / press enter , the message would be sent and the message input box after submitting / sending the message will go back to it's original / previous state when the box was empty.
- The emit method allows you to emit / call an event , which caused all the call-backs that are registered to the event to 'fire' / 'set' and is get called to perform the task allocated.