INT219 (FRONT END WEB DEVELOPER)

CHAT APPLICATION

CA3 - Project

Name – Priya Jain

Reg. No. - 11918757

Roll No. – RKM015A39

Github URL – https://github.com/PriyaJain18/chatappca3.git

Course Teacher – Dr. Shivi Sharma

Technologies used

- HTML
- CSS
- JAVASCRIPT
- REACT JS
- NODE JS
- SOCKET IO

HTML

HTML (**H**yper**t**ext **M**arkup **L**anguage) is the code that is used to structure a web page and its content. For example, content could be structured within a set of paragraphs, a list of bulleted points, or using images and data tables. As the title suggests, this article will give you a basic understanding of HTML and its functions.

Basic HTML Document Looks like :

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS describes how HTML elements are to be displayed on screen, paper, or in other media
CSS saves a lot of work. It can control the layout of multiple web pages all at once . External stylesheets are stored in CSS file.

JAVASCRIPT

JavaScript is the world's most popular programming language .JavaScript is the programming language of the Web. JavaScript is easy to learn.

JavaScript is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive. Where HTML and CSS are languages that give structure and style to web pages, JavaScript gives web pages interactive elements that engage a user. Common examples of JavaScript that you might use every day include the search box on Amazon, a news recap video embedded on The New York Times, or refreshing your Twitter feed.

Client-side: It supplies objects to control a browser and its Document Object Model (DOM). Like if client-side extensions allow an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation. Useful libraries for the client-side are AngularJS, ReactJS, VueJS and so many others

Server-side: It supplies objects relevant to running JavaScript on a server. Like if the server- side extensions allow an application to communicate with a database, and provide continuity of information from one invocation to another of the application, or perform file manipulations on a server. The useful framework which is the most famous these days is node.js.

JavaScript can be added to your HTML file in two ways:

Internal JS

External JS

REACT JS

React JS is a **declarative**, **efficient**, and flexible **JavaScript library** for building reusable UI components. It is an open-source, component-based front end library which is responsible only for the view layer of the application. It was initially developed and maintained by Facebook and later used in its products like WhatsApp & Instagram.

React code is made of entities called components. Components can be rendered to a particular element in the DOM using the React DOM library. When rendering a component, one can pass in values that are known as "props".
The two primary ways of declaring components in React is via function components and class-based components.

- React is Flexible
- React Has a Great Developer Experience
- React Has Facebook's Support/Resources
- React Also Has Broader Community Support
- React Has Great Performance
- React is Easy to Test

SOCKET-IO

Socket.IO is a JavaScript library for real-time web applications. It enables real-time, bi-directional communication between web clients and servers. It has two parts – a client-side library that runs in the browser, and a server-side library for node.js. Both components have an identical API.

NODE JS

Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009 .

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

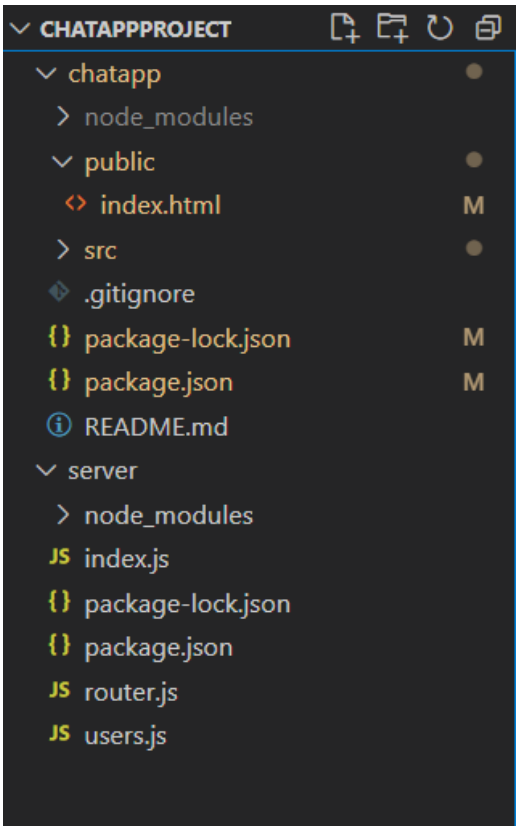
Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

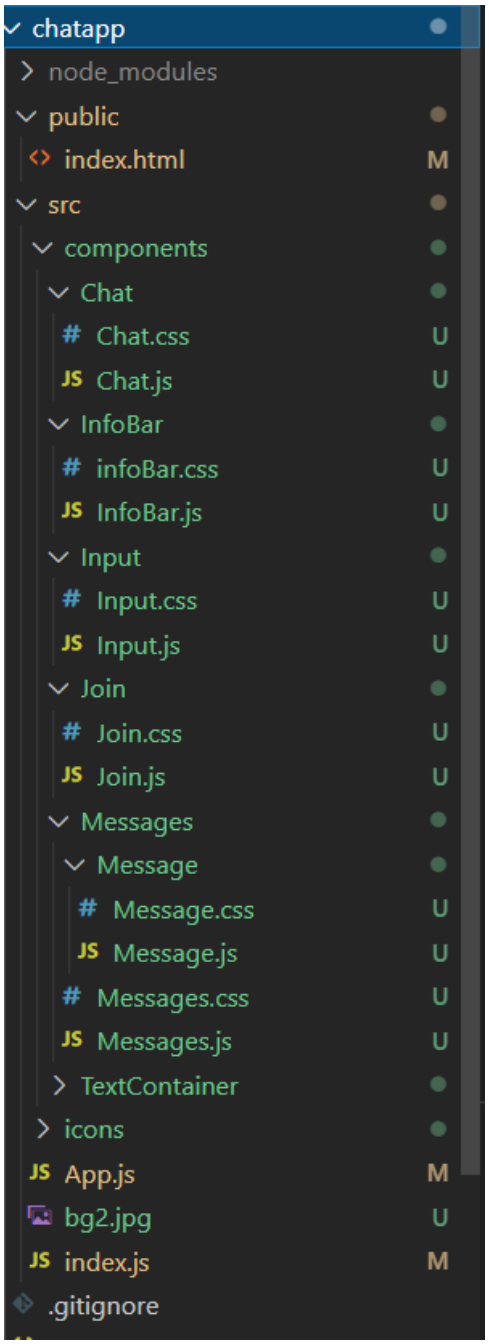
Features of Node.js.

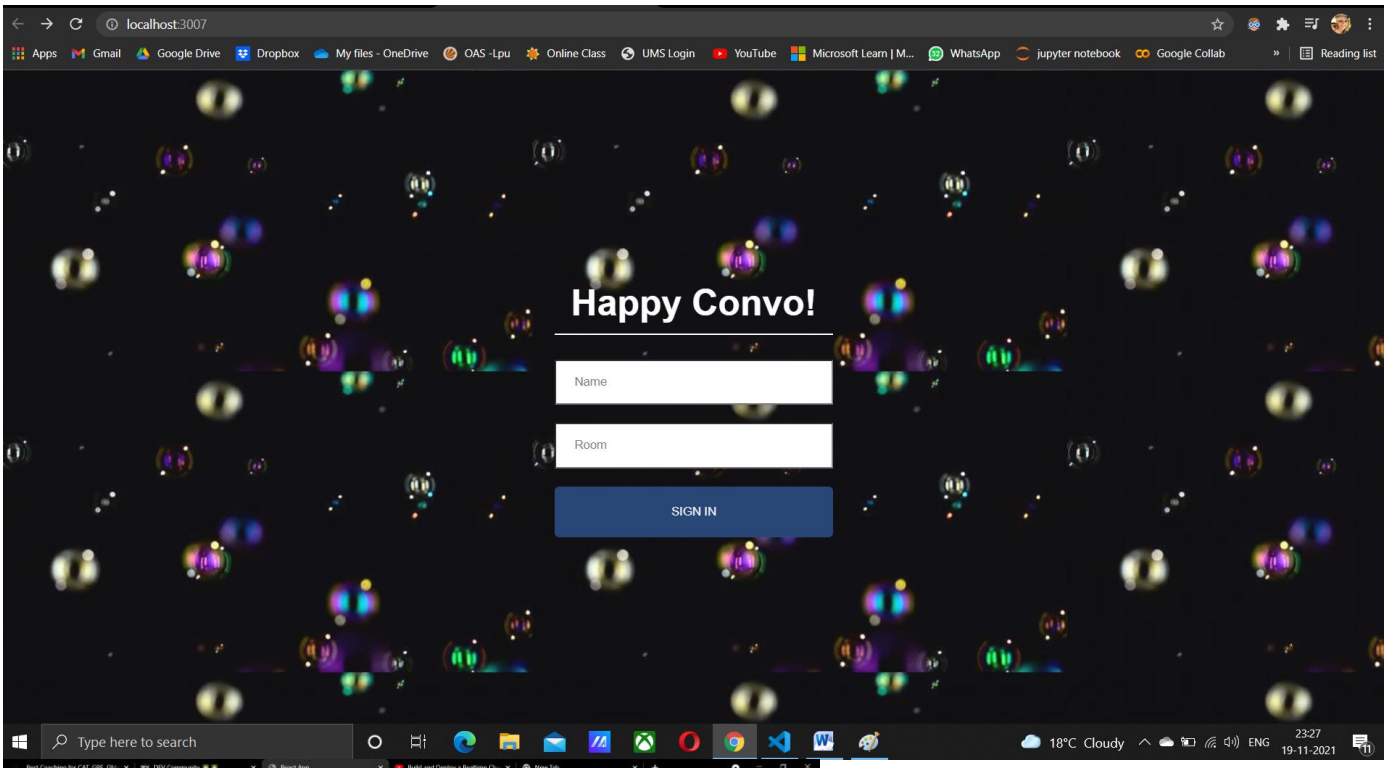
- Asynchronous and Event Driven − All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
- Very Fast − Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- Single Threaded but Highly Scalable − Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
- No Buffering − Node.js applications never buffer any data. These applications simply output the data in chunks.
- License − Node.js is released under the MIT license

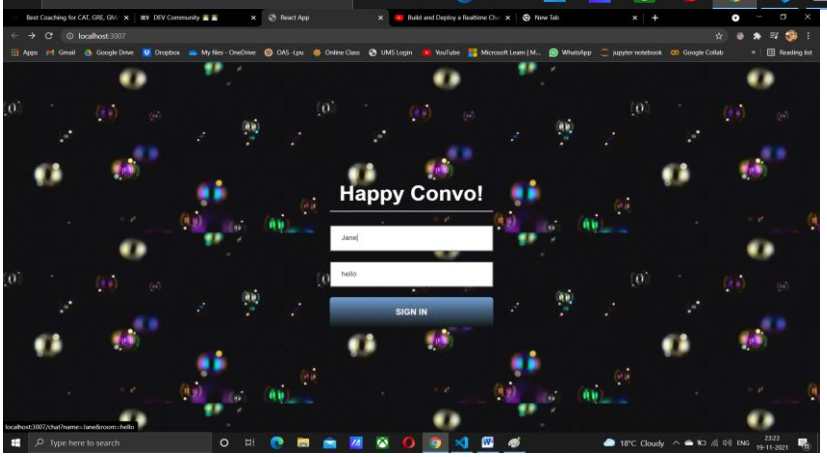
Project (ChatApp Website)

FILE-STRUCTURE :

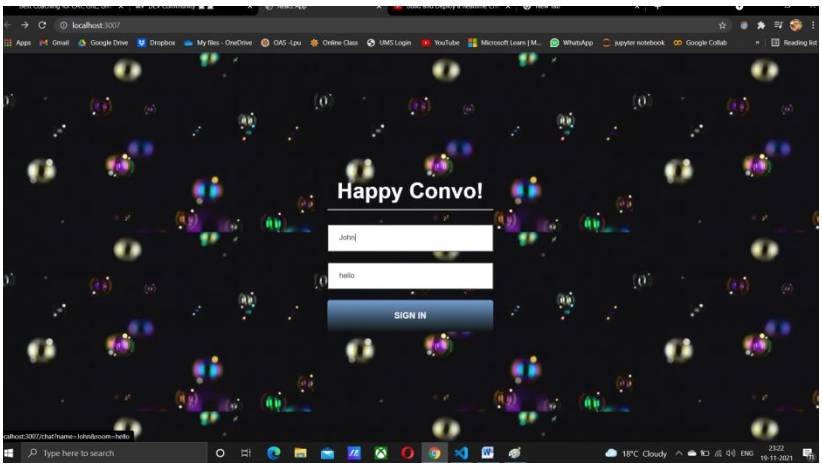






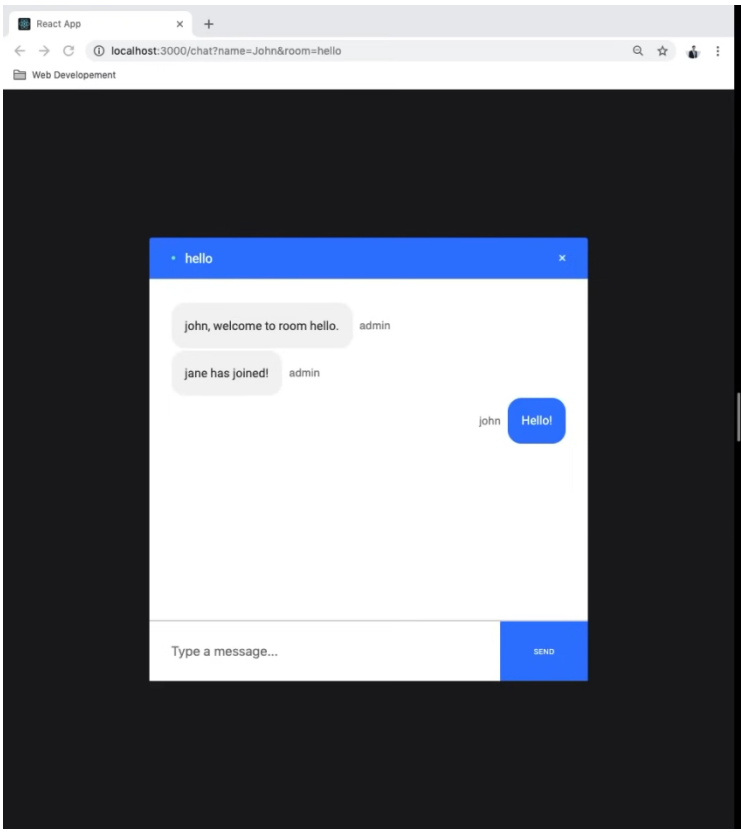


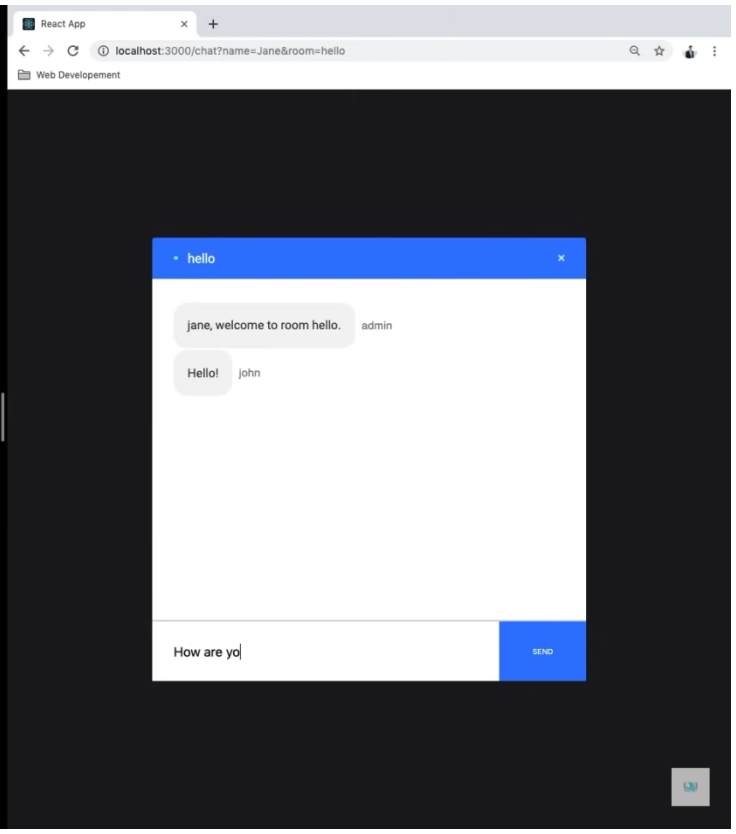
. . . LOGGING IN AS JANE TO ROOM “hello”

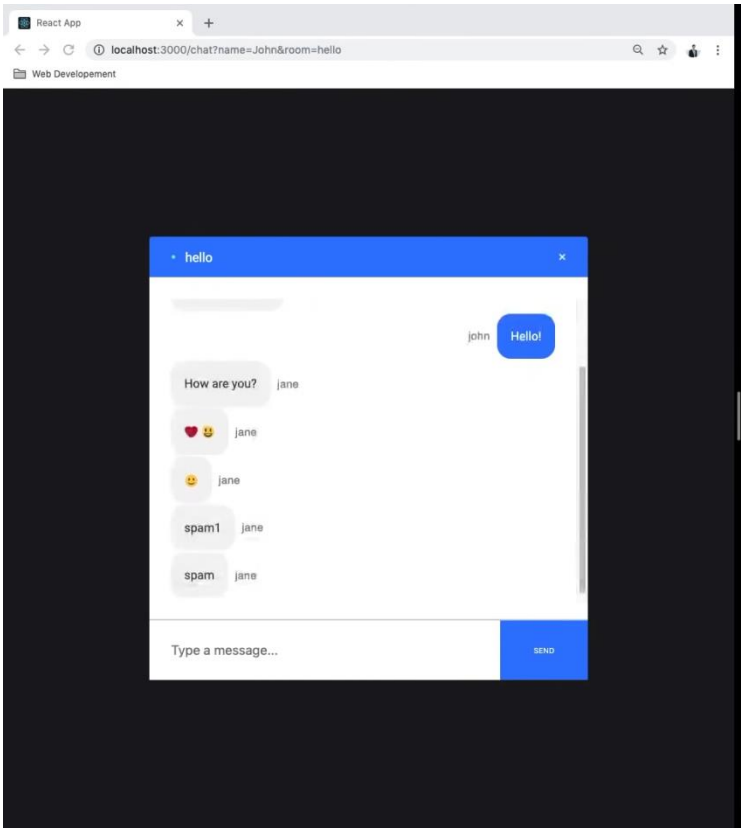


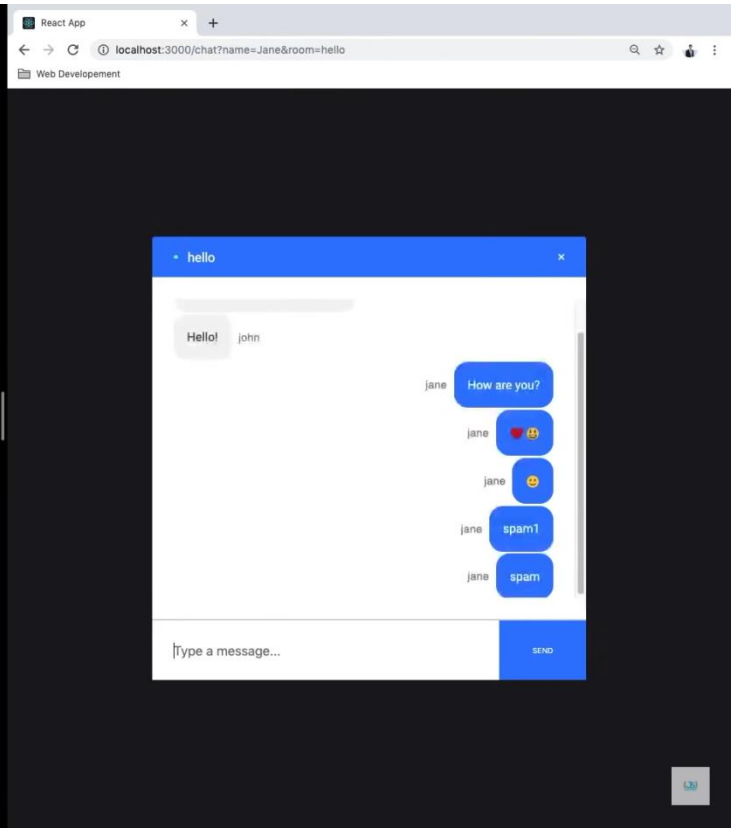
. . . LOGGING IN AS JOHN TO ROOM “hello”

Page After Logging in …. Chat Page :









CLIENT SIDE CODE FILES :

Chat.js and Join.js (Above pages are mainly coded here) :

Join.js

```
// FRONT JOIN-ROOM PAGE

import React, { useState } from 'react';  7.2K (gzipped: 3K)
import { Link } from "react-router-dom";  9.6K (gzipped: 3.8K)

import './Join.css';

Complexity is 16 You must be kidding
export default function SignIn() {
  const [name, setName] = useState('');
  const [room, setRoom] = useState('');

  return (
    <div className="joinOuterContainer">
      <div className="joinInnerContainer">
        <h1 className="heading">Happy Convo!</h1>
        <div>
          <input placeholder="Name" className="joinInput" type="text" onChange=
        </div>
        <div>
          <input placeholder="Room" className="joinInput mt-20" type="text" onC
        </div>
        Complexity is 3 Everything is cool!
        <Link onClick={e => (!name || !room) ? e.preventDefault() : null} to={`
          {/* event.preventDefault() will prevent the default functionality w
    thus not sending user to anywhere*/}

          <button className={'button mt-20'} type="submit">Sign In</button>
        </Link>
      </div>
    </div>
  );
}
```

Chat.js

```
import React,{ useState , useEffect } from "react";  7.2K (gzipped: 3K)
import queryString from "query-string";      //Parse and stringify URL quer
import io from 'sock module "c:/Users/ASUS/Desktop/Priya/CODING (practicals)/Web
import "./Chat.css"; Dev
import InfoBar from  Practicals/chatappproject/chatapp/src/components/TextContainer/T
import TextContainer from '../TextContainer/TextContainer';
import Messages from '../Messages/Messages';
import Input from '../Input/Input';


let socket;
Complexity is 18 You must be kidding
const Chat = ({location}) => {
    // Following useState hooks as same in 'Join.js' :
    let [name ,setName] = useState('');
    let [room ,setRoom] = useState('');
    let [message ,setMessage] = useState('');            //message = curre
    let [messages ,setMessages] = useState([]);        //messages = array
    const [users, setUsers] = useState('');
    const ENDPOINT = 'localhost:5000';
    Complexity is 3 Everything is cool!
    useEffect(() => {
        const { name, room } = queryString.parse(location.search);

        socket = io(ENDPOINT);

        setRoom(room);
        setName(name)

        socket.emit('join', { name, room }, (error) => {
          if(error) {
            alert(error);
          }
        });
      }, [ENDPOINT, location.search]);

      Complexity is 4 Everything is cool!
      useEffect(() => {
```

Other javascript pages :

Input.js

```
import './Input.css';

Complexity is 8 It's time to do something...
const Input = ({ setMessage, sendMessage, message }) => (
  <form className="form">
    <input
      className="input"
      type="text"
      placeholder="Type a message..."
      value={message}
      onChange={({ target: { value } }) => setMessage(value)}
      onKeyPress={event => event.key === 'Enter' ? sendMessage(event) : null}
    />
    <button className="sendButton" onClick={e => sendMessage(e)}>Send</button>
  </form>
)

export default Input;
```

InfoBar.js

```
import React from 'react';  7.2K (gzipped: 2.9K)

import onlineIcon from '../../icons/onlineIcon.png';
import closeIcon from '../../icons/closeIcon.png';

import './infoBar.css';
Complexity is 8 It's time to do something...
const InfoBar = ({ room }) => (
  <div className="infoBar">
    <div className="leftInnerContainer">
      <img className="onlineIcon" src={onlineIcon} alt="online icon" />
      <h3>{room}</h3>
    </div>
    <div className="rightInnerContainer">
      <a href="/"><img src={closeIcon} alt="close icon" /></a>
    </div>
  </div>
);

export default InfoBar;
```

Messages.js U

chatapp > src > components > Messages > Messages.js > ...

```
import React from 'react';  7.2K (gzipped: 2.9K)

import ScrollToBottom from 'react-scroll-to-bottom';  85.5K (gzipped: 27.2K)

import Message from './Message/Message';

import './Messages.css';

Complexity is 5 Everything is cool!
const Messages = ({ messages, name }) => (
  <ScrollToBottom className="messages">
    Complexity is 3 Everything is cool!
    {messages.map((message, i) => <div key={i}><Message message={message} name={name}/></div>)}
  </ScrollToBottom>
);

export default Messages;
```

Message.js - chatappproject - Visual Studio Code

TextContainer.js U

chatapp > src > components > TextContainer > TextContainer.js > ...

```
import React from 'react';

import onlineIcon from '../../icons/onlineIcon.png';

import './TextContainer.css';

Complexity is 17 You must be kidding
const TextContainer = ({ users }) => (
  <div className="textContainer">
    <div>
      <h1>Realtime Chat Application <span role="img" aria-label="emoji">
      <h2>Created with React, Express, Node and Socket.IO <span role="im
      <h2>Try it out right now! <span role="img" aria-label="emoji">
    </div>
    {
      users
        ? (
          <div>
            <h1>People currently chatting:</h1>
            <div className="activeContainer">
              <h2>
                Complexity is 3 Everything is cool!
                {users.map(({name}) => (
                  <div key={name} className="activeItem">
                    {name}
                    <img alt="Online Icon" src={onlineIcon}/>
                  </div>
                ))}
              </h2>
            </div>
          </div>
          )
          : null
    }
  </div>
);

export default TextContainer;
```

Messages.js U | Message.js U

chatapp > src > components > Messages > Message > Message.js > ...

```
import React from 'react';  7.2K (gzipped: 2.9K)

import './Message.css';

import ReactEmoji from 'react-emoji';  44.5K (gzipped: 14.3K)

Complexity is 12 You must be kidding
const Message = ({ message: { text, user }, name }) => {
  let isSentByCurrentUser = false;

  const trimmedName = name.trim().toLowerCase();

  if(user === trimmedName) {
    isSentByCurrentUser = true;
  }

  return (
    isSentByCurrentUser
      ? (
        <div className="messageContainer justifyEnd">
          <p className="sentText pr-10">{trimmedName}</p>
          <div className="messageBox backgroundBlue">
            <p className="messageText colorWhite">{ReactEmoji.emojify(text)}</p>
          </div>
        </div>
        )
        : (
          <div className="messageContainer justifyStart">
            <div className="messageBox backgroundLight">
              <p className="messageText colorDark">{ReactEmoji.emojify(text)}</p>
            </div>
            <p className="sentText pl-10 ">{user}</p>
          </div>
        )
  );
}

export default Message;
```

CSS PAGES : The main css files (Join.css and Chat.css) are below ,while there are CSS Files for every Component of the Client 's :

Join.css

```
html, body {
    font-family: 'Roboto', sans-serif;
    padding: 0;
    margin: 0;
  }

  #root {
    height: 100vh;
  }

  * {
    box-sizing: border-box;
  }

  .joinOuterContainer {
    display: flex;
    justify-content: center;
    text-align: center;
    height: 100vh;
    align-items: center;
    background-color: #1A1A1D;
    background: url("../../bg2.jpg");
    /* background-repeat: no-repeat; */
    opacity: 94%;
  }

  .joinInnerContainer {
    width: 20%;
  }

  .joinInput {
    border-radius: 0;
    padding: 15px 20px;
    width: 100%;
    -webkit-box-shadow: 0 2px 10px 1px rgba(0,0,0,0.5);
  box-shadow: 0 2px 10px 1px rgba(0,0,0,0.5);
  }
```

Chat.css

```
/* html{
    background-color: black;
    font-size: 62.5%;  /* 1 rem = 10 px
} */
.outerContainer {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #1A1A1D;
  }

  .container {
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    background: #FFFFFF;
    border-radius: 8px;
    height: 60%;
    wi
  }

  @med
    .o
      height: 100%;
    }

    .container {
      width: 100%;
      height: 100%;
    }
  }

  @media (min-width: 480px) and (max-width: 1200px) {
    .container {
      width: 60%;
    }
  }
```

Specifies the height of the content area, padding area or border area (depending on 'box-sizing') of certain boxes.

Syntax: <viewport-length>{1,2}

MDN Reference

Input.css

```
.form {
    display: flex;
    border-top: 2px solid #D3D3D3;
  }

  .input {
    border: none;
    border-radius: 0;
    padding: 5%;
    width: 80%;
    font-size: 1.2em;
  }

  input:focus, textarea:focus, select:focus{
    outline: none;
  }

  .sendButton {
    color: #fff !important;
    text-transform: uppercase;
    text-decoration: none;
    background: #2979FF;
    padding: 20px;
    display: inline-block;
    border: none;
    width: 20%;
  }
```

Message.css

```
.messageBox {
  background: #F3F3F3;
  border-radius: 20px;
  padding: 5px 20px;
  color: white;
  display: inline-block;
  max-width: 80%;
}

.messageText {
  width: 100%;
  letter-spacing: 0;
  float: left;
  font-size: 1.1em;
  word-wrap: break-word;
}

.messageText img {
  vertical-align: middle;
}

.messageContainer {
  display: flex;
  justify-content: flex-end;
  padding: 0 5%;
  margin-top: 3px;
}

.sentText {
  display: flex;
  align-items: center;
  font-family: Helvetica;
  color: #828282;
  letter-spacing: 0.3px;
}

.pl-10 {
  padding-left: 10px;
```

Index.js and App.js for the client side :

```
// CLIENT SIDE
import React from "react";  7.2K (gzipped: 2.9K)
import ReactDOM from "react-dom";  121.2K (gzipped: 39.4K)
import App from './App';
import { BrowserRouter} from "react-router-dom";  10K (gzipped: 4K)

ReactDOM.render(
    <BrowserRouter>
    <App/>
    </BrowserRouter>
,document.getElementById('root'));
```

```
import React from "react";
// import { BrowserRouter as Router, Route } from "react-router-dom";
import { Route, Switch } from "react-router-dom";
import Chat from './components/Chat/Chat';
import Join from './components/Join/Join';

Complexity is 5 Everything is cool!
const App = () => {
    return(<>
    <Switch>
    <Route path="/" exact component={Join}/>
    <Route path="/chat" exact component={Chat}/>
    </Switch>
    </>)
}
export default App;
```

SERVER SIDE FILES :

```
const users = [];

Complexity is 6 It's time to do something...
const addUser = ({id,name,room}) => {
    name = name.trim().toLowerCase();
    room = room.trim().toLowerCase();

    const existingUser = users.find((user)=> user.room === room && user.name
    if (existingUser){
        return {error : 'Username is taken'}
    }
    const user = { id, name, room };
    users.push(user);
    return {user};
}

Complexity is 4 Everything is cool!
const removeUser = (id) => {
    const index = users.findIndex((user) => user.id === id );
    if(index !== -1){
        return users.splice(index,1)[0];
    }
 }
const getUser = (id) => { users.find((user)=> user.id === id)}

const getUsersInRoom = (room) => { users.filter((user)=> user.room === room )

module.exports = { addUser, removeUser, getUser, getUsersInRoom }
```

```
const http = require('http');
const express = require('express');
const socketio = require('socket.io');
const cors = require('cors');  4.4K (gzipped: 1.9K)

const { addUser, removeUser, getUser, getUsersInRoom } = require('./users');

const router = require('./router');

const app = express();
const server = http.createServer(app);
const io = socketio(server);

app.use(cors());
app.use(router);

Complexity is 7 It's time to do something...
io.on('connect', (socket) => {
  Complexity is 3 Everything is cool!
  socket.on('join', ({ name, room }, callback) => {
    const { error, user } = addUser({ id: socket.id, name, room });

    if(error) return callback(error);

    socket.join(user.room);

    socket.emit('message', { user: 'admin', text: `${user.name}, welcome to r
    socket.broadcast.to(user.room).emit('message', { user: 'admin', text: `${

    io.to(user.room).emit('roomData', { room: user.room, users: getUsersInRoo

    callback();
  });

  socket.on('sendMessage', (message, callback) => {
    const user = getUser(socket.id);

    io.to(user.room).emit('message', { user: user.name, text: message });
```

router.js

server > router.js > ...

```javascript
const express = require('express');
const router = express.Router();
router.get('/',(req,res) => {
    res.send('Server is up and running');
});

module.exports = router;
```

THANK YOU