IT assignment 2

Soumalya Kundu

Roll: 001810501033

Online Chat application

Overview

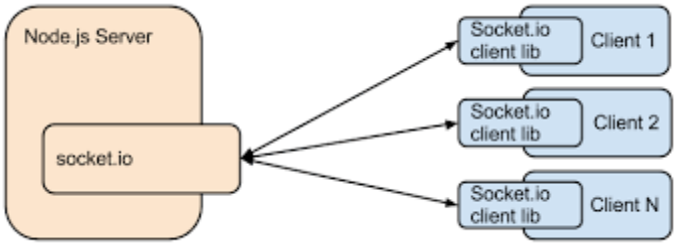
Here we need to develop an online chat application using node.js as backend React.js or similar other js framework as frontend and web sockets as a connection media.

Goals

- Having Multicast unicast and broadcast support over Websockets
- Implementing multi user format for frontend
- Support for image functionalities.

What is webSocket?

By definition WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection. Here we will be using **SOCKET.IO** as our preferred websocket api.



Features implemented

- Text message and Images are fully supported
- Broadcast unicast and multicast is totally supported
- Multi user interface
- User login logout system provided
- Timestamping on the messages are provided
- Auto scroll down when a new message is arrived

Technologies Used

Frontend: vanilla.js, react.js, ejx, html, css, react-dom

Backend: node.js, express.js, socket.io

Implementation details and Code Snippets

Login/logout:

```
router.get("/altlogin", (req, res) => {
    res.render("index");
});

router.post("/login", (req, res) => {
    // console.log(userlist.length);
    // console.log("hello")
    if (req.body.u in passwords) {
        if (req.body.p == passwords[req.body.u]) {
            //res.send("successful");
            res.render("chat_try", { messages: messages[req.body.u], username: req.body.u, num: userlist.length });
            //res.render("chat_try");
        }
        else {
            //res.send("Wrong");
            //res.sendFile(__dirname + "/public/index.html");
            res.render("index", { error: "Wrong password" });
            //res.rendor(__dirname +  "public/index.html", {error: "Wrong password"});
        }
    }
    else {
        passwords[req.body.u] = req.body.p;
        messages[req.body.u] = [];
        userlist.push(req.body.u);
        res.render("chat_try", { messages: messages[req.body.u], username: req.body.u, num: userlist.length });
        //res.render("chat_try");

    }
```

Message Functionalities:

Unicast/Multicast/broadcast @client

```
    // hooks for handling messages
    useEffect(()=> {

        socket.on('message',(message)=>{
            // admin generated messages
            setMessages([...messages, message]);
        });

    }, [messages]);

    const sendMessage = (event)=> {
        event.preventDefault();

        if(message){
            socket.emit('sendMessage', message, ()=> setMessage(''));
            // 3rd parameter is a cleanup code for textField
        }
    }


    return (
        <div className='outerContainer'>
            <div className='container'>
                <InfoBar room={ room }/>
                <Messages messages={messages} name={name}/>
                <Input message={message} setMessage={setMessage} sendMessage={sendMessage} />
            </div>
        </div>
    )
};
export default Chat;
```

Unicast/Multicast/broadcast @server

```
io.on('connect', (socket)=>{
    console.log("We have a new connection!!");

    socket.on('join',({name, room}, callback)=>{
        //callback(); // basically for error handling and needed to pass as 3rd argument in the client part
        const { error, user } = addUser({id: socket.id, name:name, room: room}); // it can return 2 things
        if(error) return callback(error);

        // for no error
        socket.join(user.room);

        // user inside the room
        socket.emit('message', {user: 'admin', text: `Welcome ${user.name}!!`});
        socket.broadcast.to(user.room).emit('message', {user: 'admin', text: `${user.name} just slid into the room!`});

        io.to(user.room).emit('roomData', {room: user.room, users: getUsersInRoom(user.room)})
        callback();
    });


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

        io.to(user.room).emit('message', {user: user.name, text: message});
        //io.to(user.room).emit('roomData', {room: user.room, users: getUsersInRoom(user.room)});

        callback();
    });
```

AutoScrollDown

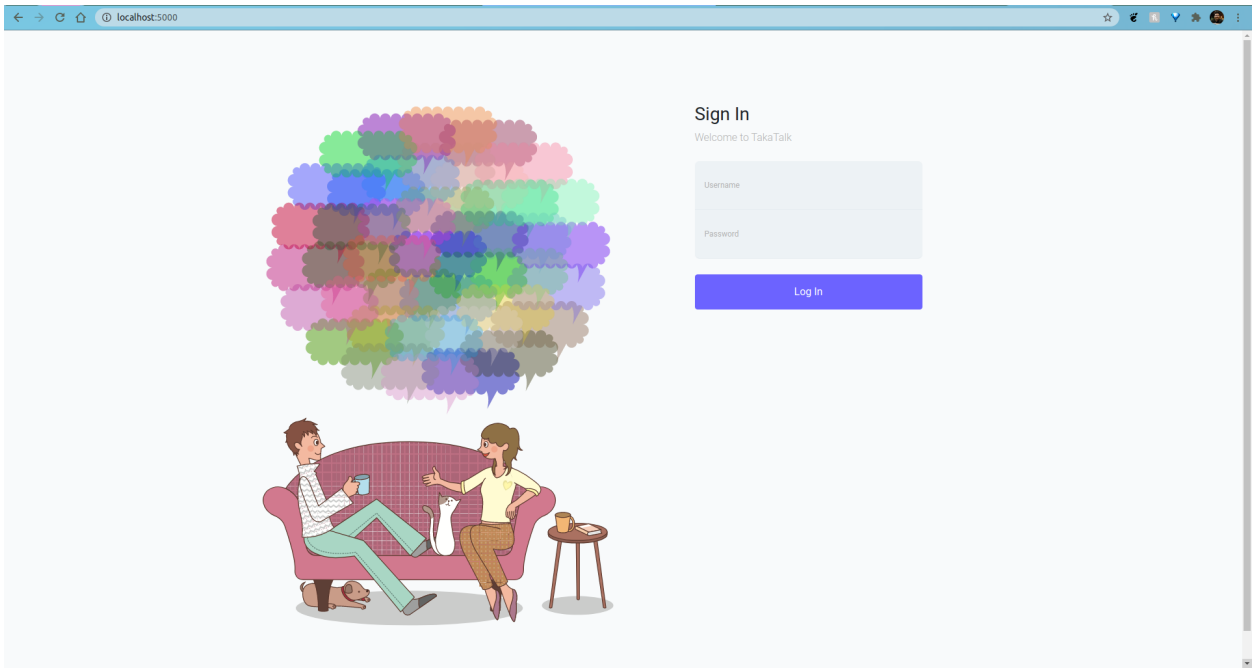
```
const Messages = ({messages, name}) => (

    <ScrollToBottom className="messages" >
        {messages.map((message, i) =>
        <div key={i}>
            <Message message={message} name={name} />
        </div>)}
    </ScrollToBottom>

);
// basically we are getting every message and wrapping them into 'Message' component
export default Messages;
```

Output:

Greeting Screen:



BroadCast :

Broadcasting test message from user 1. Total logged in user = 3

From User1:

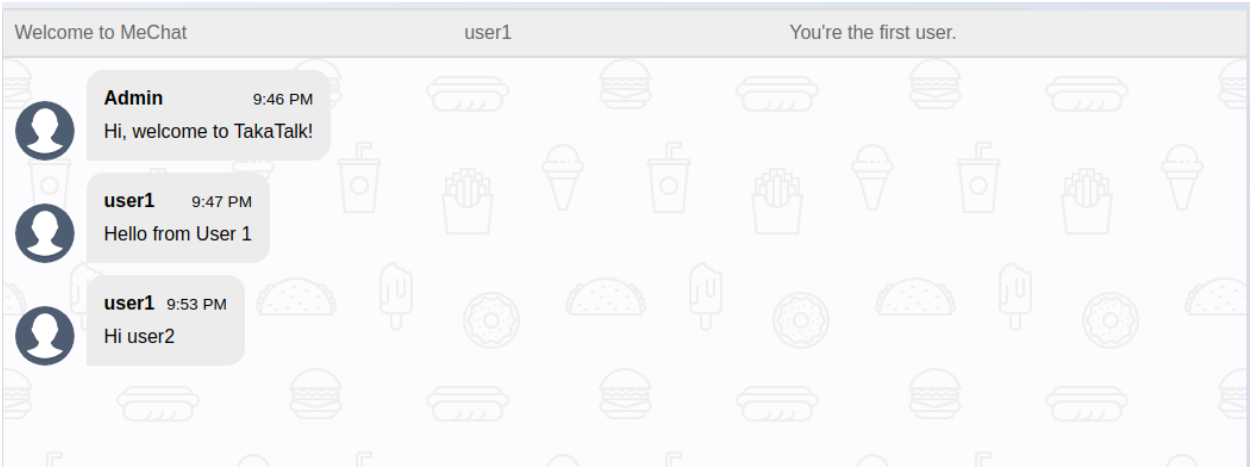
@ User2:

@ User3:

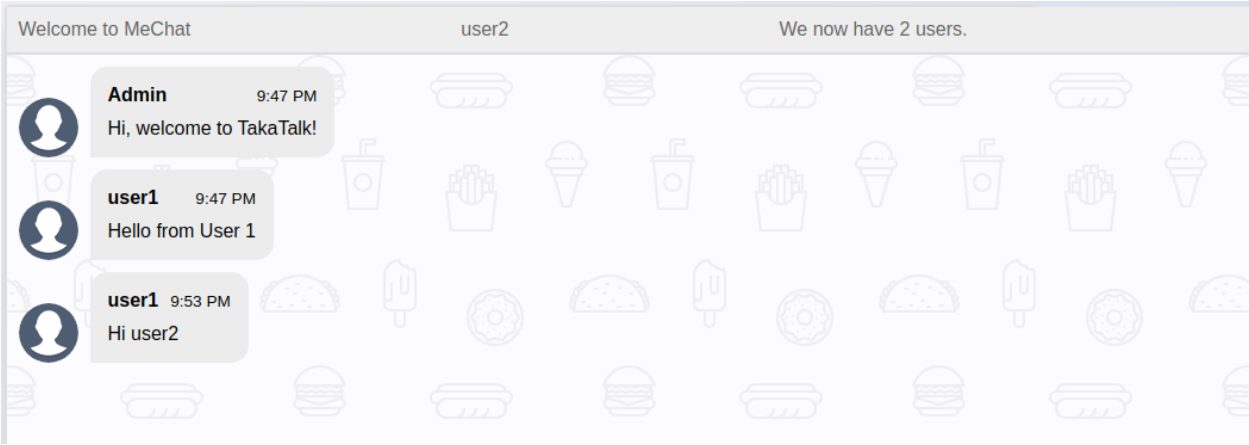
Multicast Message :

Multicasting a text message from user1 to user 2 but not to user3.

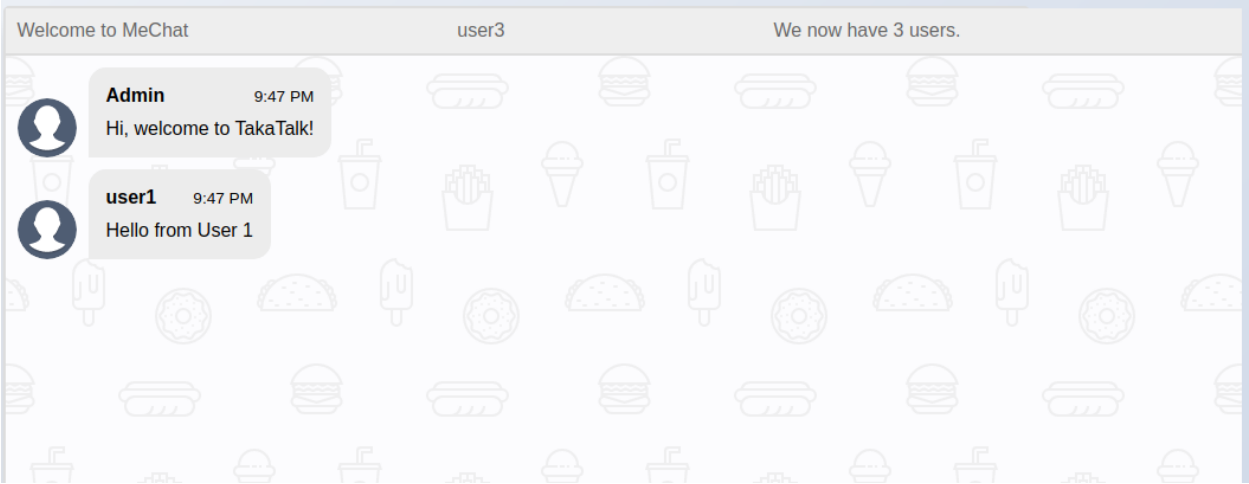
From User1:



@ User2:



@ User3:

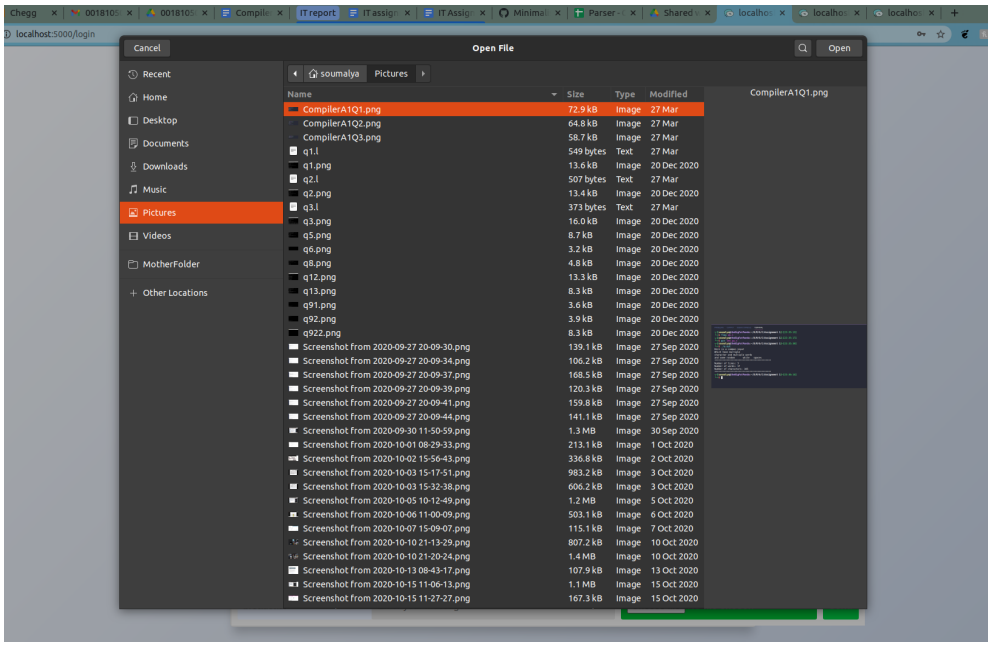


So as we can see, user3 doesn't get the message as it was not meant for him.

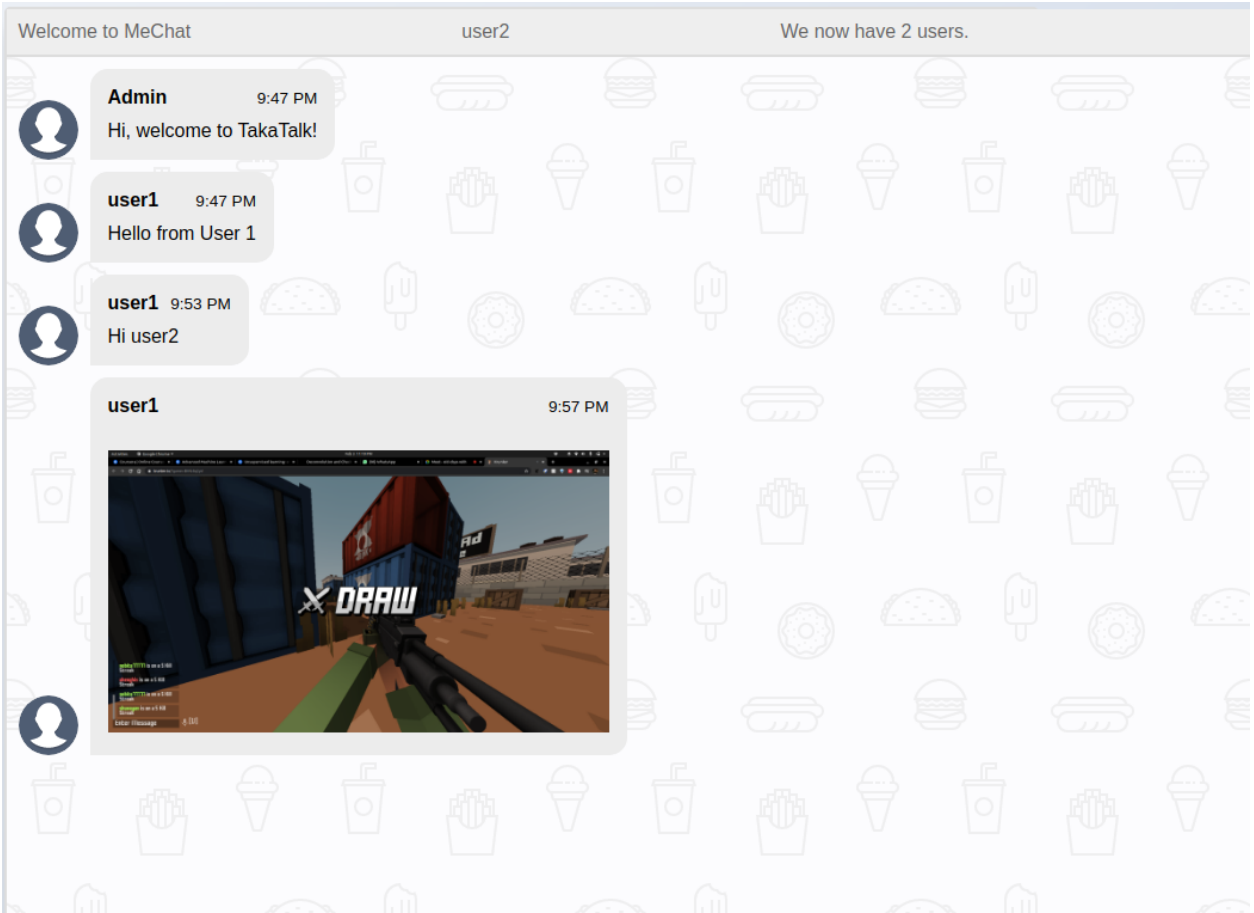
Working with ImageBased Messages:

Broadcasting a image from user1 to all the other users.

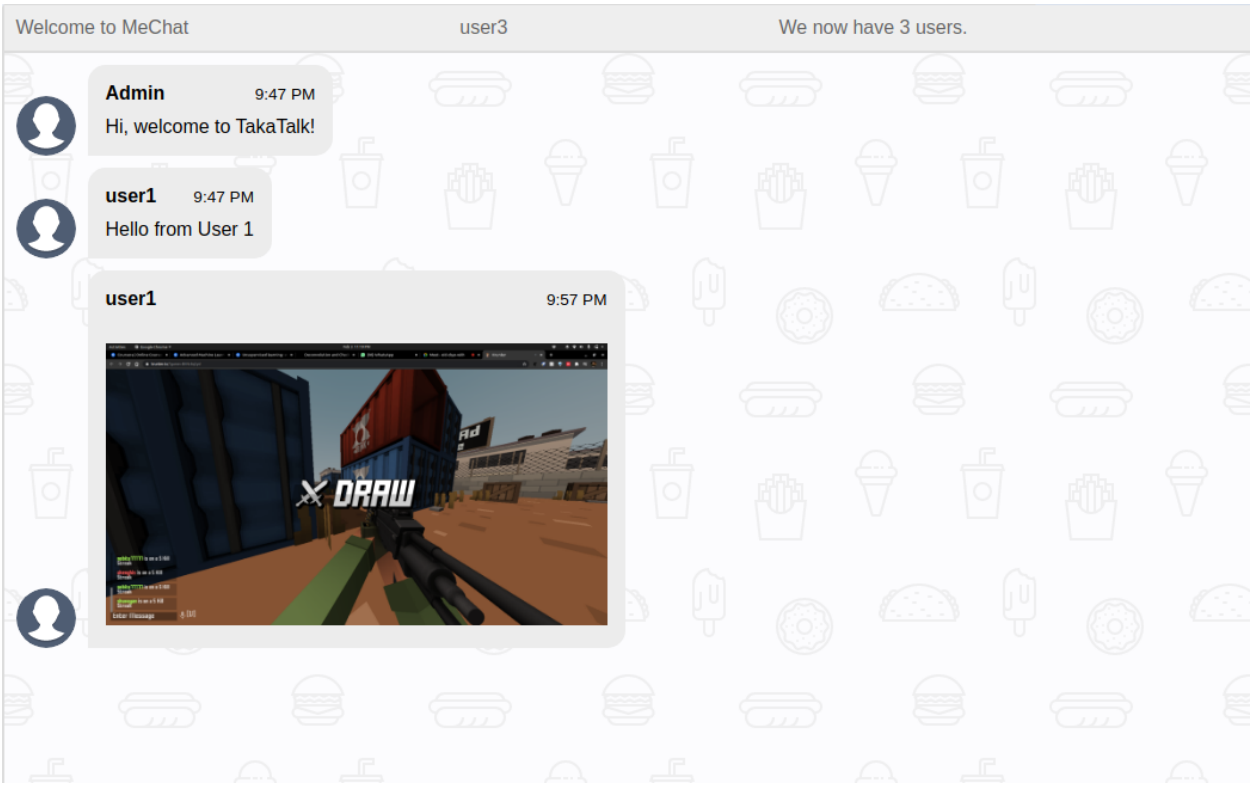
From User1:


Cancel
Open File
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
MotherFolder
Other Locations
soumalya
Pictures
Name
Size
Type
Modified
CompilerA1Q1.png
CompilerA1Q2.png
CompilerA1Q3.png
q1.l
q1.png
q2.l
q2.png
q3.l
q3.png
q5.png
q6.png
q8.png
q12.png
q13.png
q91.png
q92.png
q922.png
Screenshot from 2020-09-27 20-09-30.png
Screenshot from 2020-09-27 20-09-34.png
Screenshot from 2020-09-27 20-09-37.png
Screenshot from 2020-09-27 20-09-39.png
Screenshot from 2020-09-27 20-09-41.png
Screenshot from 2020-09-27 20-09-44.png
Screenshot from 2020-09-30 11-50-59.png
Screenshot from 2020-10-01 08-29-33.png
Screenshot from 2020-10-02 15-56-43.png
Screenshot from 2020-10-03 15-17-51.png
Screenshot from 2020-10-03 15-32-38.png
Screenshot from 2020-10-05 10-12-49.png
Screenshot from 2020-10-06 11-00-09.png
Screenshot from 2020-10-07 15-09-07.png
Screenshot from 2020-10-10 21-13-29.png
Screenshot from 2020-10-10 21-20-24.png
Screenshot from 2020-10-13 08-43-17.png
Screenshot from 2020-10-15 11-06-13.png
Screenshot from 2020-10-15 11-27-27.png

@ User2:


Welcome to MeChat
user2
We now have 2 users.
Admin 9:47 PM
Hi, welcome to TakaTalk!
user1 9:47 PM
Hello from User 1
user1 9:53 PM
Hi user2
user1 9:57 PM
DRAW

@ User3:



Final Comments

1. Synchronization is not handled so concurrency issues may occur.
2. This was a very interesting project as I learnt the basics of socket.io and node.js