IT Lab Report – Assignment 2

TITLE

Name – Biswarup Ray

Roll – 001810501082

Class – BCSE 3rd year

Group – A3

Assignment Number – 2

Write a multi-client chat application consisting of both client and server programs. In this chat application simultaneously several clients can communicate with each other. For this you need a single server program that clients connect to. The client programs send the chat text or image (input) to the server and then the server distributes that message (text or image) to all the other clients. Each client then displays the message sent to it by the server. The server should be able to handle several clients concurrently. It should work fine as clients come and go. Clients should be able to send messages in any two modes out of the following.

(i)Unicast; (ii) multicast; and (iii) broadcast

Develop the application using a framework based on Node.JS.

How are messages handled concurrently? Which web application framework(s) did you follow? Prepare a detailed report of the experiments you have done, and your observations on the protocol layers thus created.

<u>Modes of messaging</u>.

Multicast:

A multicast is a transmission of data from a single source to multiple recipients. Multicasting is similar to broadcasting, but only transmits information to specific users. It is used to efficiently transmit streaming media and other types of data to multiple users at one time.

Multicasting have been achieved in my implementation using **room/channel system** where all the users/participants present in a channel could receive the messages sent in the channel whereas the users present in another channel at the same time would not receive the other channel messages.

Implementation/Code:

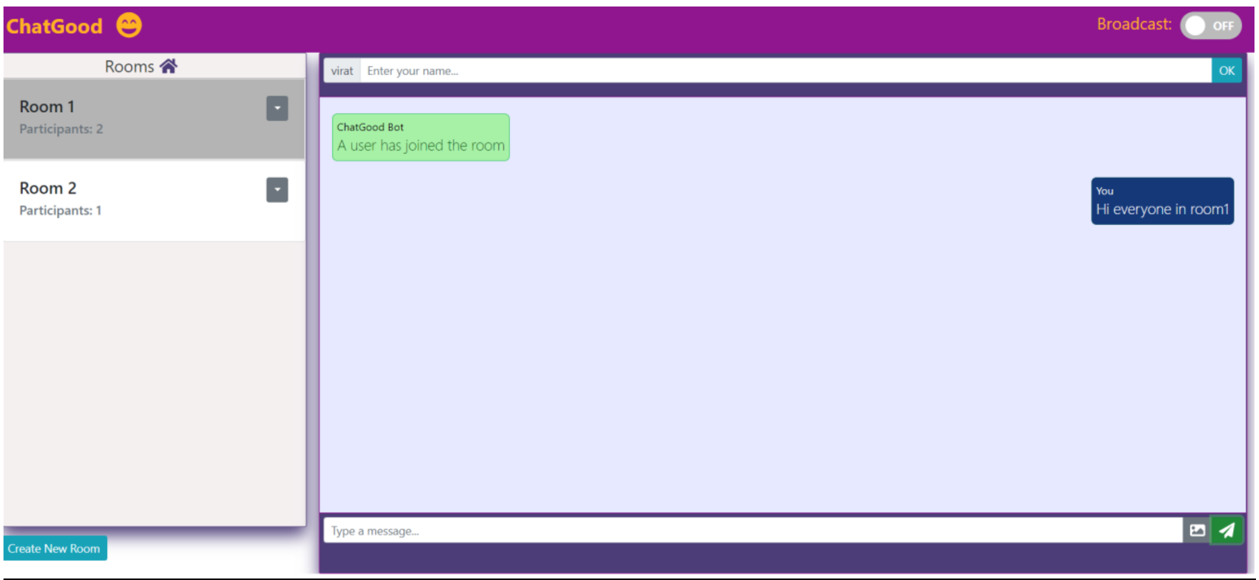
<u>In server.js</u>

```
channels.find((c, index)=> {
            if(c.channelName === data.channel.channelName) {
                c.messages.push(toSend)
                channels[index] = c
                for(let i=0;i<c.participants.length; i++) {
                    io.to(c.participants[i].socketid).emit('message', toSend);
                }
            }
        }
```

<u>In client.js</u>

```
this.state.socket.emit('sendmessage', { channel: this.state.channel, message: message, senderName: this.state.username, isFileAttached: this.state.isFileAttached, file: this.state.file })
```

Demo Output:



Room 1
Participants: 2

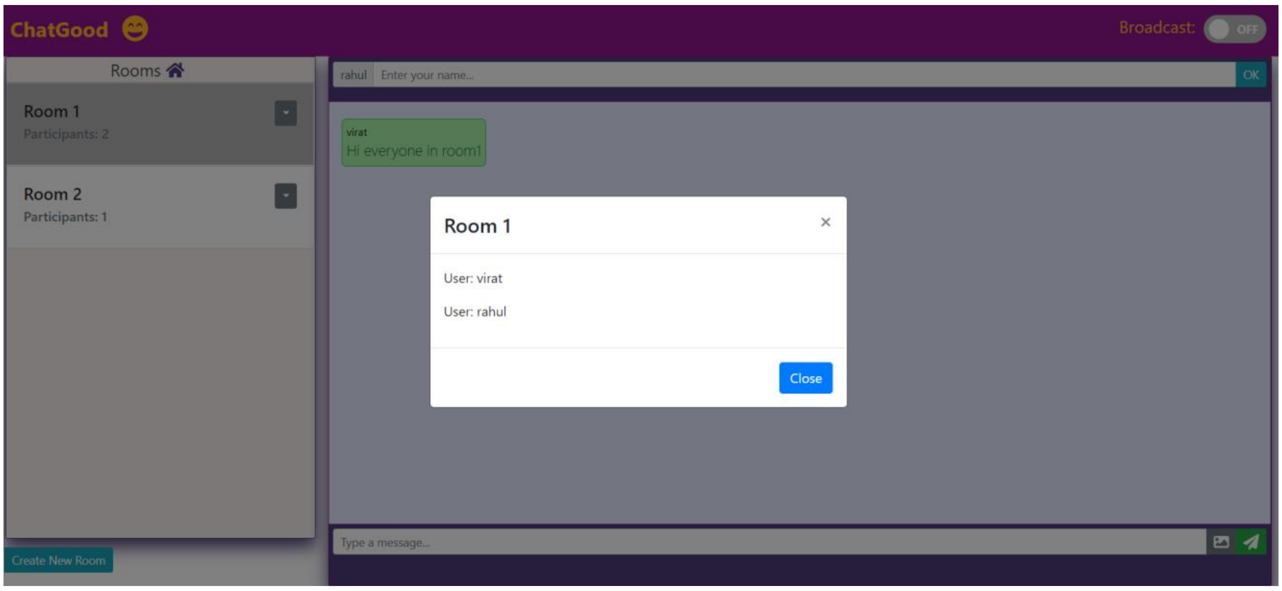
Room 2
Participants: 1

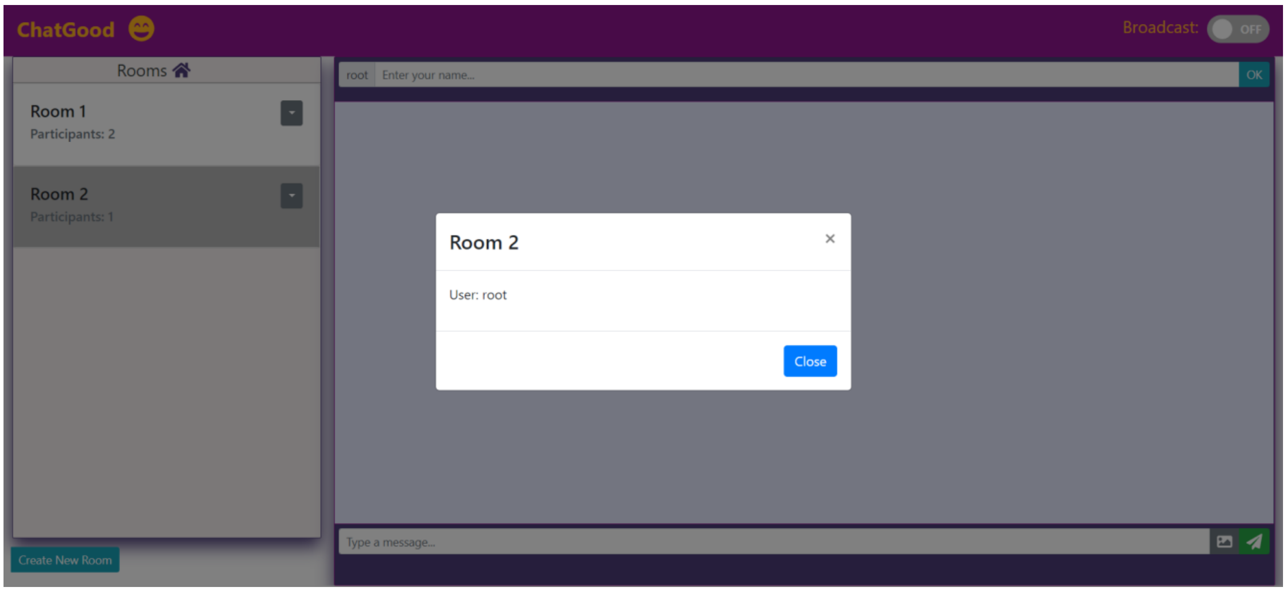
virat | Enter your name...

ChatGood Bot
A user has joined the room

You
Hi everyone in room1

Type a message...





Here Rahul and virat both are users of the Room 1 hence rahul can see the message "*Hi everyone in room1*" sent by virat. But since root is a user of the Room 2 he can't see the message sent by virat being sent to a different room/channel.

Broadcast:

Broadcasts are messages that one can send to all of the subscribers, or a group of subscribers, at once. They operate independently of any autoresponder and follow up messages that one may be sending. Sending broadcast messages does not affect your autoresponder or follow up messages.

Broadcasting have been achieved in my implementation using a **toggle button** which once clicked emits the message sent by that specific user/participant to emit it to all the rooms regardless of the rooms the other users are present in.

Implementation/Code:

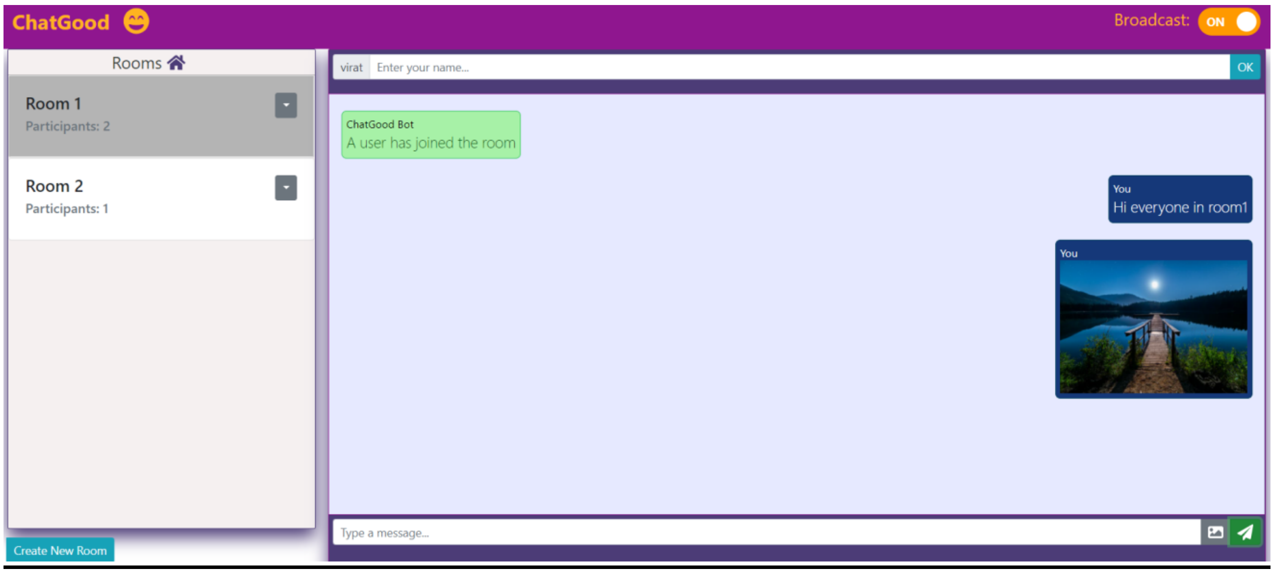
In server.js

```
if(data.channel.channelName === "__broadcast") {
            for(let i=0;i<channels.length; i++) {
                channels[i].messages.push(toSend)
            }
            io.emit('message', toSend)
        }
```

In client.js

```
this.state.socket.emit('send-
message', { channel: {channelName: "__broadcast"}, message: message, senderNam
e: this.state.username, isFileAttached: this.state.isFileAttached, file: this.
state.file })
    }
```

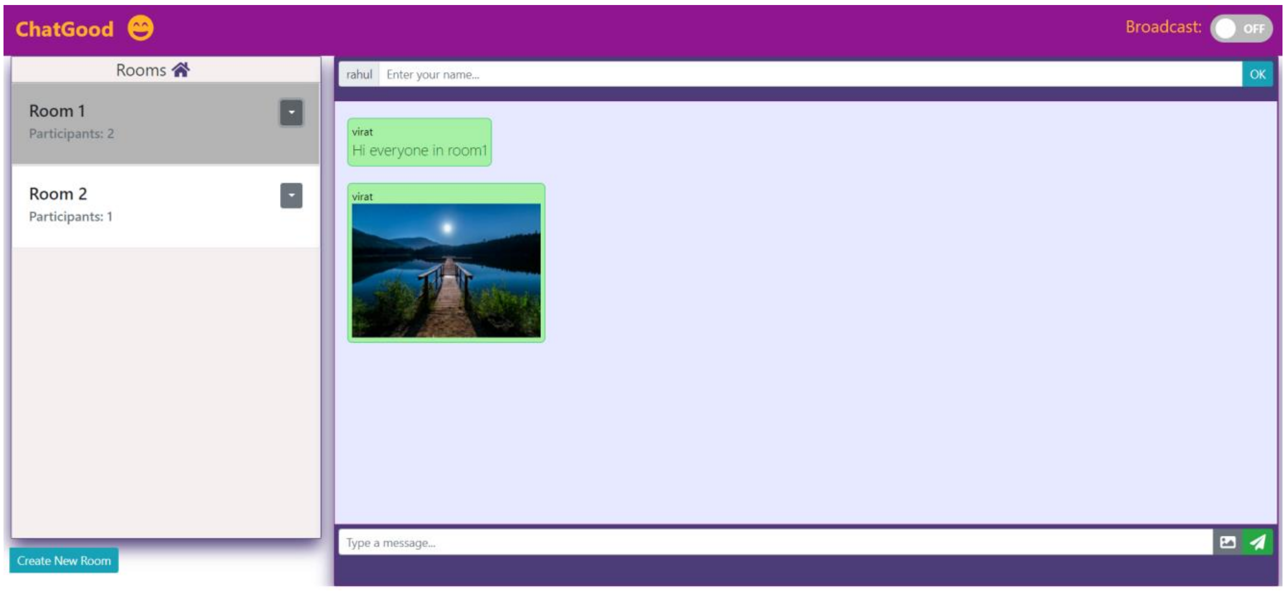
Demo Output:



ChatGood Bot
A user has joined the room

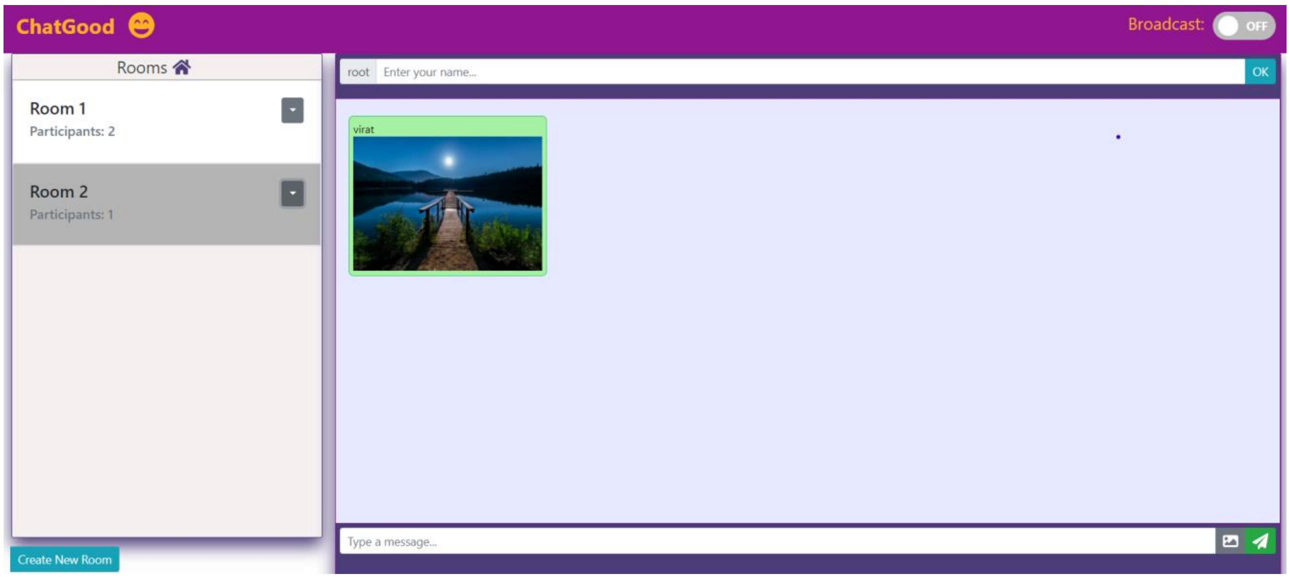
You
Hi everyone in room1

You



virat
Hi everyone in room1

virat



virat

Here Rahul and virat both are users of the Room 1 and root is in Room 2 . But since virat has sent the image after keeping the **broadcast toggle button** on hence every user regardless of the room they are present in could see the image.

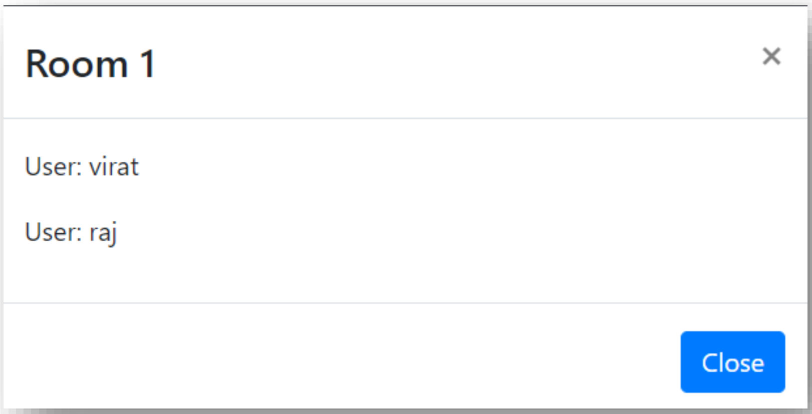
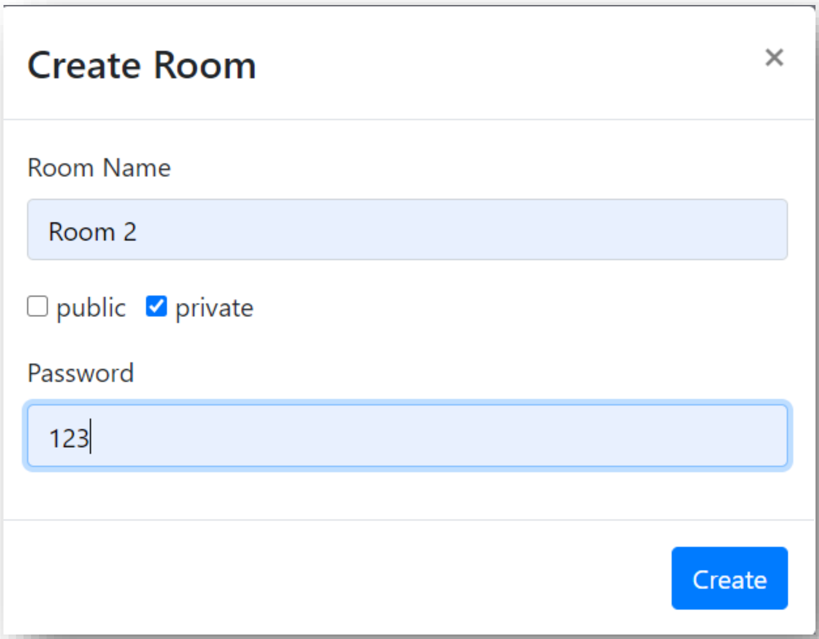
Table for Features present in the implementation

S.No	Feature	Implementation and Description

1	Image Sending	`    this.toBase64 =this.toBase64.bind(this)`
		`  }`
		
		`  toBase64(arr) {`
		`    if(arr) {`
		`      return btoa(`
		`        arr.data.reduce((data, byte) => data +`
		`String.fromCharCode(byte), '')`
		`      )`
		`    } else {`
		`      console.log('Error')`
		`    }`
		`  }`
		Using base 64 encoding of the image first.

2	Front End in React	The Front-End has been coded in Reactjs

3.	Room/ Channel system	`//layout of the channels variable`
		`channels = [{`
		`    channelName: "",`
		`    number_of_users: 0,`
		`    participants: [{`
		`        socketid: "",`
		`        username: ""`
		`    }],`
		`    messages: [{`
		`        socketid: "",`
		`        username: "",`
		`        message: ""`
		`    }]`
		`}]`

4.	Ui to check the list of users present in the room	 Room 1 User: virat User: raj
5.	Private and public channels/rooms With password protected private rooms.	 Create Room Room Name: Room 2 [ ] public [✓] private Password: 123