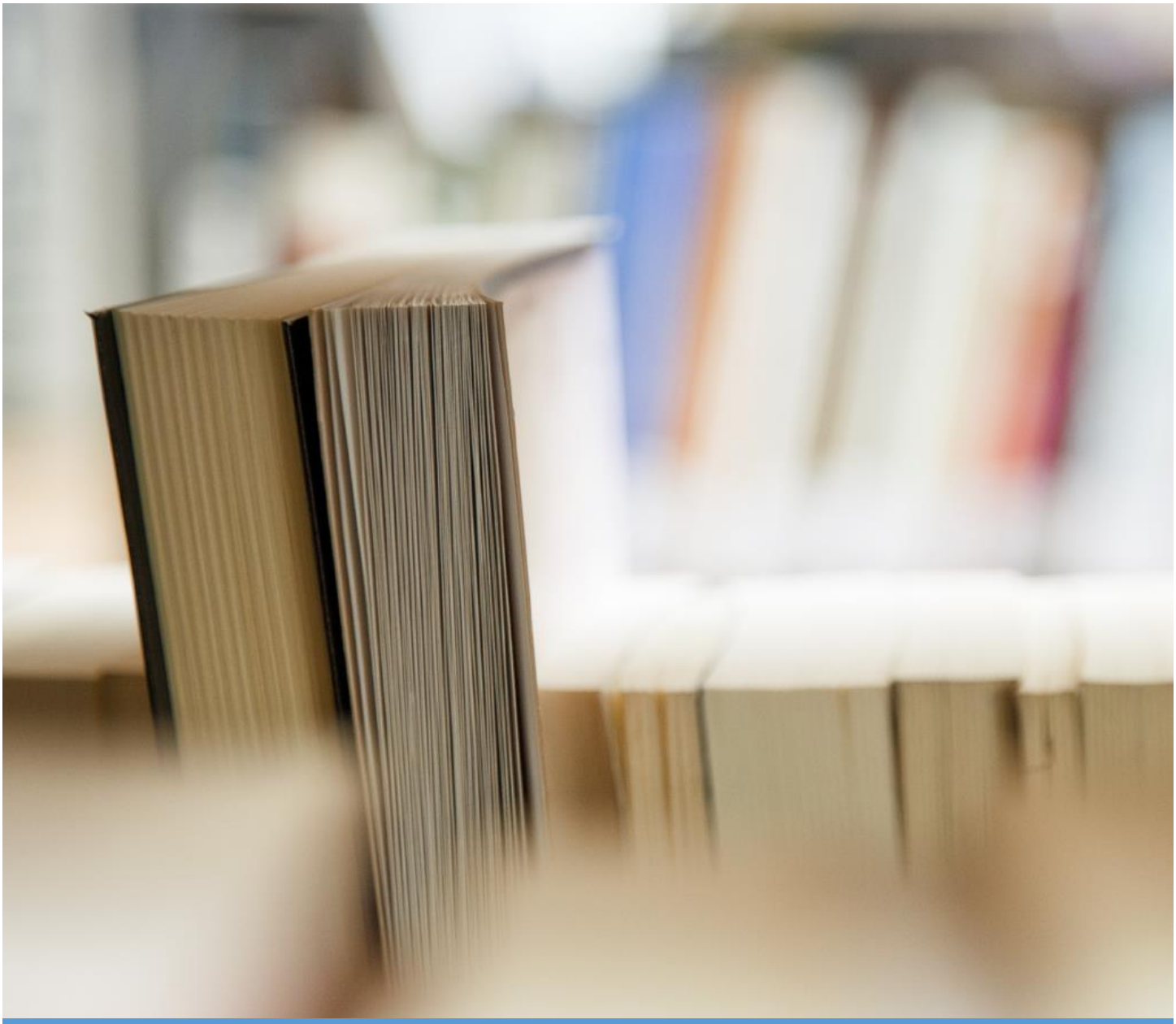

IT Lab Assignment 2

Real Time Chat Application

Description: I have developed a chat web app named "MyMessenger" which can support multiple features.

- Message unicast.
- Message multicast.
- Message broadcast.
- Image unicast.
- Image multicast.
- Image broadcast.

Technology used for this web app:

- JavaScript.
- Node JS.
- Express.
- Socket.io.

Message unicast, multicast, broadcast:

Code:

Server Side:

```
    //listen for  chat messages
    socket.on('chatMessage', ({ msg, mulusers }) => {

        if (mulusers.length != 0) {
            var multiUsers = mulusers.split(',');
            var num = multiUsers.length;

            const user = getCurrentUser(socket.id);
            io.to(user.id).emit('message', formatMessage(user.username, msg));

            for (var i = 0; i < num; i++) {
                var sock = getUserSocketID(multiUsers[i]);
                io.to(sock).emit('message', formatMessage(multiUsers[i], msg))
;
            }
        } else {
            const user = getCurrentUser(socket.id);
            io.to(user.room).emit('message', formatMessage(user.username, msg)
);
        }

    });
```

Client Side:

```
// Message from server
socket.on('message', (message) => {
    console.log(message);
    outputMessage(message);

    // Scroll down
    chatMessages.scrollTop = chatMessages.scrollHeight;
});
```

Message Formatting:

```
const moment = require('moment');

function formatMessage(username, text) {
  return {
    username,
    text,
    time: moment().format('h:mm a')
  };
}

module.exports = formatMessage;
```

Utility Functions for User:

```
const users = [];

// Join user to chat
function userJoin(id, username, room) {
    const user = { id, username, room };

    users.push(user);

    return user;
}

// Get current user
function getCurrentUser(id) {
    return users.find(user => user.id === id);
}

// User leaves chat
function userLeave(id) {
    const index = users.findIndex(user => user.id === id);

    if (index !== -1) {
```

```
        return users.splice(index, 1)[0];
    }
}

// Get room users
function getRoomUsers(room) {
    return users.filter(user => user.room === room);
}
//get user by username
function getUserSocketID(username) {
    const index = users.findIndex(user => user.username === username);

    return users[index].id;
}

module.exports = {
    userJoin,
    getCurrentUser,
    userLeave,
    getRoomUsers,
    getUserSocketID
};
```

Joining a Chat Room:

Server Side:

```
// Run when client connects
io.on('connection', socket => {
    socket.on('joinRoom', ({ username, room }) => {
        const user = userJoin(socket.id, username, room);

        //user joins
        socket.join(user.room);

        // Welcome current user
        socket.emit('message', formatMessage(botName, 'Welcome to MyMessenger!
'));

        // Broadcast when a user connects
        socket.broadcast
            .to(user.room)
            .emit(
                'message',
                formatMessage(botName, `${user.username} has joined the chat`)
            );

        // Send users and room info
```

```
        io.to(user.room).emit('roomUsers', {
            room: user.room,
            users: getRoomUsers(user.room)
        });
    });
```

Client Side:

```
// Join chatroom
socket.emit('joinRoom', { username, room });

// Get room and users
socket.on('roomUsers', ({ room, users }) => {
    outputRoomName(room);
    outputUsers(users);
});
```

Image unicast, multicast, broadcast:

Server Side:

```
//when user send the image, server then broadcasted it those who are connected
    socket.on("image", (imgData) => {
        const user = getCurrentUser(socket.id);

        socket.broadcast.emit("image", imgData);
    });
```

Client Side:

```
// For image
sendImage.addEventListener('change', () => {
    var filesSelected = document.getElementById("sendImage").files;
    if (filesSelected.length > 0) {
        var fileToLoad = filesSelected[0];

        var fileReader = new FileReader();

        fileReader.onload = function(fileLoadedEvent) {

            var srcData = fileLoadedEvent.target.result; //  data: base64

            var newImage = document.createElement('img');
            newImage.src = srcData;

            let imgData = {
                message: srcData
            }
```

```
            displayImage(srcData); //display image in the message area

            socket.emit('image', imgData); //send image to the server to broad
cast it so that users can see image
        }
        fileReader.readAsDataURL(fileToLoad);
    }
});
```

Displaying Image:

```
//display image in the message area
function displayImage(srcData) {
    const div = document.createElement('div');
    var newImage = document.createElement('img');
    newImage.src = srcData;

    // document.getElementById("historyMsg").innerHTML = user + newImage.outer
HTML;

    //messageArea.append(newImage);
    //appendMessage(`send image...`);
    div.append(newImage);
    document.querySelector('.chat-messages').append(div);
    //autoScrollDown();
    // alert("Converted Base64 version is " + document.getElementById("history
Msg").innerHTML);

};

//incoming image message
socket.on("image", (imgData) => {
    displayImage(imgData.message);
    // autoScrollDown();
});
```