System design document
Chat application for competitive community

Purpose

This is group chatting application is designed for competitive community.
This chatting feature is disabled during atcoder contest time due to prevent cheating.
This chat application should be used when contest is finished and discuss information of question.

System overview

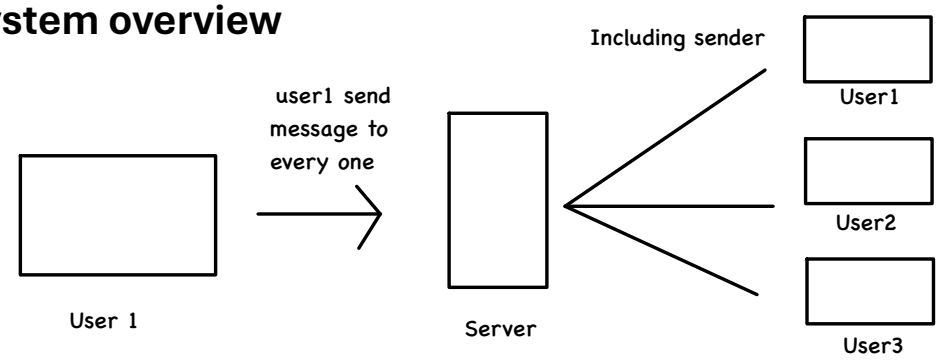

Figure 1 above representing the architectural structure I have chosen for my chat application. The backend handle the message,disable feature and generate name.

System components

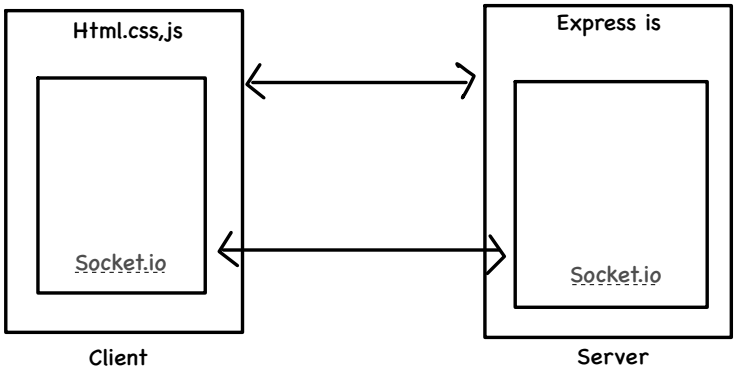

Figure 2 above showing how my chat application expected to work and how components interact.

Workflow

User choose chat room by Programming language. Then server side code check contest is held or not. If it during contest time display message and disconnect from socket.

```
io.on('connection', socket => {
    if (!isChatAllowed()) {
        socket.emit('admin-Messages', formatMessage(admin, 'Chatting is currently disabled due to ABC contest.'));
        socket.disconnect();
        return;
    }
```

If its not contest time join user with username and room name. Display admin message who is joined. This is broad cast so it will notify everyone who is in the group.

```
socket.on('joinRoom',({username,room}) => {
        // console.log('username : '+username);
        const  user = userJoin(socket.id ,username,room);
        socket.join(user.room);

        //* as soon as user connect, log on server side then emit message -> front(main.js) 1
        socket.emit('admin-Messages',formatMessage(admin,'Welcome to our community')); //* for single client


        //* Broadcast when user connect
        socket.broadcast.to(user.room).emit('admin-Messages', formatMessage(admin,`${user.username} has joined the chat`)); //* for all client except connecting

        //* Send users and info
        io.to(user.room).emit('roomUsers',{
            room : user.room,
            users: getRoomUsers(user.room)
        })
    });
```

When user disconnect, display admin message who is disconnected and delete from list by socket id.

```
socket.on('disconnect',() =>{
      const user = userLeave(socket.id);
      if(user){
          io.to(user.room).emit('admin-Messages', formatMessage(admin,`${user.username} has left the chat`)); //* all client

          //* Send users and info
          io.to(user.room).emit('roomUsers',{
          room : user.room,
          users: getRoomUsers(user.room)
          });

      }

  });
```

To handle message detect who send and who received are handling front end code.

```
function outputMessage(message,username) {
        var check = false;
        if(message.username === username) check = true;
        console.log(message.username);
        console.log(username);
        const divav = document.createElement('div');
        divav.classList.add('msg-container');
        const div = document.createElement('div');
        if(check){
            div.classList.add('msg_sent');
            console.log('sent');
        }else{
            div.classList.add('msg_received');
            console.log('received');
        }
        const p = document.createElement('p');
        if(check)p.classList.add('meta_sent');
        else p.classList.add('meta_received');
        p.innerText = message.username+" : ";
        p.innerHTML += `<span>${message.time}</span>`;
        const para = document.createElement('p');
        para.classList.add('text');
        para.innerText = message.text;
        para.appendChild(p);
        div.appendChild(para);
        divav.appendChild(div);
        console.log('div.classname :'+div.className+'\n'+'p.className '+p.className+'\n'+para.className);
        document.querySelector('.chat-messages').appendChild(divav);
}

```