CS494 Tyler Alway
Internet Draft Portland State University
Intended status: IRC Class Project Specification November 29, 2016

Internet Relay Chat Class Project
irc-pdx-cs494.txt

Table of Contents

1. Introduction

This is a specification for an implementation of an Internet Relay Chat (IRC) protocol. This system includes a server where messages are sent and routed to the clients. The clients will be connected to a default room (global) and will be able to join and leave additional rooms. The client will also be able to send messages to these rooms and list all rooms and user in given rooms.

2. Basic Information

All communication will take place over TCP/IP meaning there will be persistent connection. The program will be written in JavaScript with Node.js and express for the backend with all communication being handled by Socket.io interface done in html/CSS and jQuery.

3. Message Infrastructure

3.1. Message Format

Messages sent between the server and client will be sent by emitting a Socket.io event over a connection which will send an event type and a JSON object which should have the fields seen below.

```
{
        'id': string,
        'user': string,
        'rooms': string,
        'message':string,
        'error': boolean,
        'errorMessage': string,
}
```

Each messages will also have an type or an event type. Example usage:

```
socketHandle.emit('SOMETYPE', {
        'id': string,
        'user': string,
        'rooms': string,
        'message':string,
        'error': boolean,
        'errorMessage': string,
}
```

3.1.1. Field definitions

id is the unique user id assigned at start up from the server this is required to be sent with all server messages.

user: username field for a list of usernames when listing a room or sending the clients username. This field may be a single string or an array of strings.
rooms: specifies the room/rooms to send the message to, join, list, or create. This field may be a single string or an array of strings.

message: is the actual message that was sent.

error: error specifies if there has been an error or not. This field is always required to come from the server. Also if this field is true you must ignore all other fields except on JOIN events (this will be explained in the JOIN section).

errorMessage: hold the error message this is only required if the error field is marked true.

3.1.2. event types

NAME
NAME_RESP
HELLO
LIST_ROOMS
LIST_ROOMS_RESP
LIST_USERS
LIST_USERS_RESP
JOIN_ROOM
JOIN_ROOM_RESP

LEAVE_ROOM
LEAVE_ROOM_RESP
CREATE_ROOM
CREATE_ROOM_RESP
MESSAGE
connection
disconnect

4. Name Semantics

Rooms and users both have names.
Names are unique to the user or room.
Names must be ASCII characters.
If any of these things is not true, then send an error.

5. Client Messages

5.1. First message sent to the server

When you set up the connection to the server on the client side a connection event will be sent to the server and the server will pick a random unique user id and send a NAME event to get a username form the user this will be in the form:

```
socketHandle.emit('NAME', {
            'id': id,
            'message': "Please enter a userneame",
            'error': false,
 });
```

This will be caught by the client and the user will input a username and send a NAME_RESP event:

```
socketHandle.emit('NAME_RESP', {
            'id': strId,
            'user': jQuery('#uName').val(),
            'error': false,
});
```

This will be met by the server sending a HELLO event to the client to signal that they are connected.

5.2. Listing Rooms

When a click event is detected on the listrooms button emit a LIST_ROOMS event:

```
socketHandle.emit('LIST_ROOMS', {'id': strId,});
```

5.3. Creating Rooms

When a click event is detected on the submitroom button emit a CREATE_ROOM event:

```
socketHandle.emit('CREATE_ROOM', {
```

```
        'id': strId,
        'rooms': room,
        'error': false,
});
```

5.4. Joining a Room

When a click event is detected on the joinroom button emit a JOIN_ROOM event:

```
socketHandle.emit('JOIN_ROOM', {
        'id': strId,
        'user': username,
        'rooms': room,
});
```

The JOIN_ROOMS operation will still add the user to all available rooms even if an error was found so the JOIN_ROOM_RESPONSE is the only response that you can't ignore all fields if you get an error message.

5.5. Leaving a Room

When a click event is detected on the leaveroom button emit a LEAVE_ROOM event:

```
socketHandle.emit('LEAVE_ROOM', {
        'id': strId,
        'user': username,
        'rooms': room,
});
```

5.6. Sending Messages

When a click event is detected on the submit button emit a MESSAGE event:

```
socketHandle.emit('MESSAGE', {
        'user': username,
        'id': strId,
        'rooms': jQuery('#joined').val(),
        'message': jQuery('#message').val()
});
```

6. Server Messages

6.1. Listing Room Response

When a LIST_ROOMS event is detected get all of the rooms into an array and send them in the rooms field in a 'LIST_ROOMS_RESP event:

```
socketHandle.emit('LIST_ROOMS_RESP', {
        'rooms': send,
        'error': false
});
```

6.2. Forwarding Messages to Clients

When a MESSAGE event is detected check to see if the room exists if it does not emit an error. If the room exists emit the message to everyone in the room with a ' MESSAGE event:

```
socketHandle.emit('MESSAGE', {
        'user': data.user,
        'rooms': data.rooms,
        'message': data.message,
        'error': false,
        'errorMessage': '',
});
```

6.3. Join Response

When a JOIN_ROOM event is detected make sure the room exists and if it does not emit an error. Else emit a ' JOIN_ROOM_RESP event:

```
socketHandle.emit('JOIN_ROOM_RESP', {
        'error': error,
        'rooms': roomsJoined,
        'message': messageS,
});
```

The JOIN_ROOMS operation will still add the user to all available rooms even if an error was found so the JOIN_ROOM_RESPONSE is the only response that you can't ignore all fields if you get an error message.

6.4. Create Room Response

When a CREATE_ROOM event is detected check to see if the room already exists and if it does emit a error. If it does not exist emit a ' CREATE_ROOM_RESP event:

```
socketHandle.emit('CREATE_ROOM_RESP', {
        'error': error,
        'rooms': data.rooms,
        'message': messageS,
});
```

6.5. Listing Users Response

When a LIST_USERS event is detected get all of the users in a room into an array and send them in the users field in a 'LIST_USERS_RESP event:

```
socketHandle.emit('LIST_USERS_RESP', {
        'user': send,
        'error': false
});
```

6.6. Leaving Response

When a LEAVE_ROOM event is detected find the correct room and verify it exists and the remove the user from this room. If there was an error emit a ' LEAVE_ROOM_RESP event like:

```
socketHandle.emit('LEAVE_ROOM_RESP', {
        'error': true,
        'errorMessage': 'Error you are not in ' + data.rooms,
        'rooms': data.rooms,
});
```

If there was no error emit a ' LEAVE_ROOM_RESP event like:

```
socketHandle.emit('LEAVE_ROOM_RESP', {
        'error': false,
        'message': 'You have left ' + data.rooms,
        'rooms': data.rooms,
});
```

7. Error Handling

The server and client must both be able to detect if they have disconnected from one another. If this should occur the server will disconnect the user from all rooms that they were in. The client will get a message saying that they have been disconnected from the server.

8. 'Extra' Features Supported

None.

9. Security Considerations

Messages sent while using this app will not be secure.

10. Acknowledgments

This document was prepared using the example RFC by Byron Marohn.