1 A Chatroom Web Service

In this assignment you are asked to implement a multi-chatroom web service, based on WebSockets. Your chatroom server should be implemented in Node.js. The browser’s interface can be as minimalistic as you wish.

The overall idea is summarized in the following functionality:

• anyone can register as a user with a unique username
• any registered user can create a new chatroom
• any registered user can join any existing chatroom, and then receive all messages posted from that moment on in that chatroom and post messages to all other users in that chatroom.

The entry point to your Node.js web application should be index.js, and it should accept exactly two arguments: the port where your web server listens, and the port where your WebSocket listens. In order to define dependencies on external libraries, please make sure your dependencies are listed in the provided package.json file, and this file is in the root of your project. We are going to start your web application by npm start <http_port> <WebSocket_port>. For instance, to start the chat server on port 5555 with the WebSocket listening on port 7000 you would write npm start 5555 7000.

2 Protocol Definition

For the communication with the browsers, your web service should support the following operations: register user, create chatroom, and join chatroom. For simplicity users cannot deregister and chatrooms cannot be destroyed. All parameter passing for your web API should be encoded in JSON. This does not apply to the communication over WebSockets for the actual participation in a chatroom.

2.1 User Registration

This should be implemented as a POST request to the /users URI. It should contain one argument, username, which will be the user’s chosen name. Usernames are strictly alphanumeric and cannot contain spaces. For example, a valid body of the request would be: {"username":"Tom"}.
If the username is not taken (and is not the reserved username admin) the response status code should be 201 (Created), and the response body should include a unique ID to be used by this user for identification. This ID should be the MD5 hash of the username. For example, a valid response to the aforementioned request would be: "ID": "57a67cf04d232754672443c12874ca76".

If that name is already taken, the response status code should be 409 (Conflict). The response body is not specified.

2.2 Chatroom Creation

This API is a POST request to the /rooms URI. It should contain the username, the assigned ID, and the name of the new chatroom (the variable names should be precisely the highlighted ones).

If the creation was successful (i.e., the chatroom name was available), the response status code should be 201 (Created). If the name was already taken, the response status code should be 200 (OK), and the chatroom is anyway available for use. In both cases (i.e., 200 or 201), the response body should contain the WebSocket URL in a parameter called url. For example, "{url":"ws://localhost:7000/rooms/IntProgHelp"}.

2.3 Connection and Participation in a Chatroom

A registered user can connect to a chatroom by opening a websocket to the URI /rooms/<chatroom>. All subsequent communication should take place over that websocket.

The client should first send a string containing its username. Then it should send a string containing its ID. At this point, and assuming the username corresponds to a registered user and the ID matches the ID assigned to that username, the chat server assumes the user has entered this chatroom. This means that any messages submitted over this websocket should be propagated to all other (if any) users currently connected to this chatroom, and all other users' messages should also be propagated to this user.

To submit a new message, a user simply sends it in a string, in one go. There is no need to retransmit its username or ID in each message, as the chat server already knows whom this websocket belongs to.

The chat server propagates each message to all other users in this chatroom, by sending them a string of the form "<username> <message>". For example such a message could be "Tom Hi everyone!". It is up to the discretion of the students' client-side code how to display that, e.g., whether the username will be displayed in a different color, etc.

When a user connects, the chat server should send that user a welcoming message as if sent by special user admin: "admin_Welcome_to_chatroom_<chatroom>!". When a user connects (or disconnects), the chat server should send all other users of the chatroom the message: "admin_User_<username>_joined" (or left).

User disconnection is simply done by closing one's browser.
3 Submission

Please make a tar file of your submission named `hw3.tar`, and email it to `spyros-intprog@cs.vu.nl`. The subject should follow the usual format, e.g., `abc123_def456_3`. You will receive instant confirmation of the submission, but your code will be manually tested after the end of the deadline. As usually, you may submit any number of times, and only your latest submission will count as official.