Assignment III

General

Oh my god! **Code Red! I repeat! Code Red!**

A terrible thing has happened! The mIRC has been wiped from the planet and nobody has a copy of it's installation file. The CWS (*Chat World Service*) wants you to create a new mIRC so the world can chat again. Will you help us?

Description

As mentioned above this assignment is about creating a chat application. In this assignment, you should write a chat program using React and WebSockets. A server is provided and instructions on how to run the server are in the **Server** section below. The following list enumerates the functional requirements for this assignment:

- **(10%)** The user should upon arrival specify his/her nickname. If the nickname is free, i.e. no other user is currently active with the same nickname, he/she can proceed, otherwise a new nickname must be provided. *Note: no password is required, only nickname.*

- **(10%)** After the user has been identified, he/she should see a list of chat rooms already active.

- **(5%)** The user should be able to join a chat room, and leave a room as well. It should of course also be possible to create a new room.

- **(15%)** Inside a given room, the user should be able to send messages to the room, see previous messages, and see new messages appear in real time (without having to refresh the page manually).

- **(10%)** It should be possible to send a private message to another user.

- **(15%)** The creator of a room should be able to kick a user from a room. A user which has been kicked from a room can re-enter. The creator can also ban a user, which means he/she won't be able to join the room again

In addition, the following technical requirements are given:

- **(10%)** Each component should reside in a single folder, where the implementation of the component is and tests for the particular component
  - Each component should be tested using Jest
  - Each component should make use of **PropTypes**
- **(5%)** All external dependencies should be installed with npm/yarn
- **(5%)** The code should go through ESLint without warnings
- **(15%)** Webpack should be setup and do the following:
  - Bundle all components in a single JS file
  - Run tests
  - Run ESLint
  - Minifies the JS code
  - Offers an option to enable watch
  - If ES6 code is being used, integrate Babel in to Webpack

Server

The server can be downloaded in the assignment section in Canvas. It is a NodeJS server and information on how to use the server is in the **README.md** file in the .zip file for the server.

ESLint

Configuration file for ESLint can be downloaded in the description section for this assignment in Canvas. The file is **.eslintrc** and should be in the root directory of the application.

Resources

Libraries such as **jQuery, Bootstrap** and **etc…** are allowed.

Submission

Hand in a single archive file, containing all the source files for the application, including instructions on how to install and run the application in a README.md file.

Note: unless you make changes to the supplied server, then you DON'T need to hand in that code! As a matter of fact, your code should absolutely not be integrated into that folder, the server and the client should be stored in two separate folders!

Do note that your solution should NOT include **node_modules** folder!