Documentation

Git Organization

My git repository use only the master branch and I used it to make two commits. The only reason I have used it twice is because whenever I updated my files, I would forget to re-add them to the repository. I used the repository to keep track of the changes I made in my code.

Data Structures

Client side:

- Input fields: takes user inputs
- Ng Model: binds user data
- Ng for: used to loop through the messages input and displays them

Server side:

- Object: which is used for the user data
- Strings: which is used to assign things to empty
- Boolean: which was used to check if user inputs matched the stored data and to also check if user roles where equal to the stored roles in the object

Angular Architecture

Components:

- LoginComponent
- ChatComponent
- GroupadminchatComponent
- SuperuserchatComponent

Services:

- Bootstrap
- Rxjs
- Socket.io
- Body-parser
- Express
- Cors

Models:

- HTML
- CSS
- Javascript
- Angular
- Node.js

Routes:

- Login
- Chat
- Groupadminchat
- Superuserchat

Node Architecture

Modules:

- App routing module: which is used to connect the other pages.
- App module: which is used to declare and import the components

Functions:

- Listen function: which is used to host the server, also has a start time
- Connect function: which is used to connect the used with socket.io

Files:

- Api-login: which is being used to store and call user information

Global Variables:

- Express: which is used to call static values
- Path: used for routing
- Http: used for server
- Cors: which is a middleware
- Body-parser: is also middleware
- Socket.io: which is a service that manages the chat components
- Server: which is assigned to file where the server details are kept

Responsibilities

Dividing the responsibilities between the server and client was quite simple. The server takes care of loading the database and the stored user information and the client side runs the information and displays the correct forms and outputs.

List

Routes:

- Login
- Chat
- Groupadminchat
- Superuserchat
- AppModule
- App-routingmodule

Parameters:

- Express
- Path
- Http
- Cors
- Body-parser
- Socket.io
- Sockets
- Server

Return Values:

- PORT: returns the port the server will run on
- Sockets: is used to connect the socket server file to the PORT
- Server: is used to run the server
- Require: is used to include the login api to all the client pages

Interaction Details

The server.js file is used to interact with the other server files which when called run the information stored in them.

The socket.js file is used to store the functionality of when a client enters a message, this message is then taken and displayed in the client side once it passes through the server.

This listen.js file is used to host the server, when the server is ran using nodemon it will take the PORT from the server.js and run the server-side application.

The api-login.js file is used to house all the user information, this information if ran through the if statement and if the user inputs on the client side don't match and error will be displayed, but if the input values by the client are the same to one of the users then the client will be routed to there page. The only variable that changes in this process is the username and password which are stored in an object.