Networks and Distributed Systems

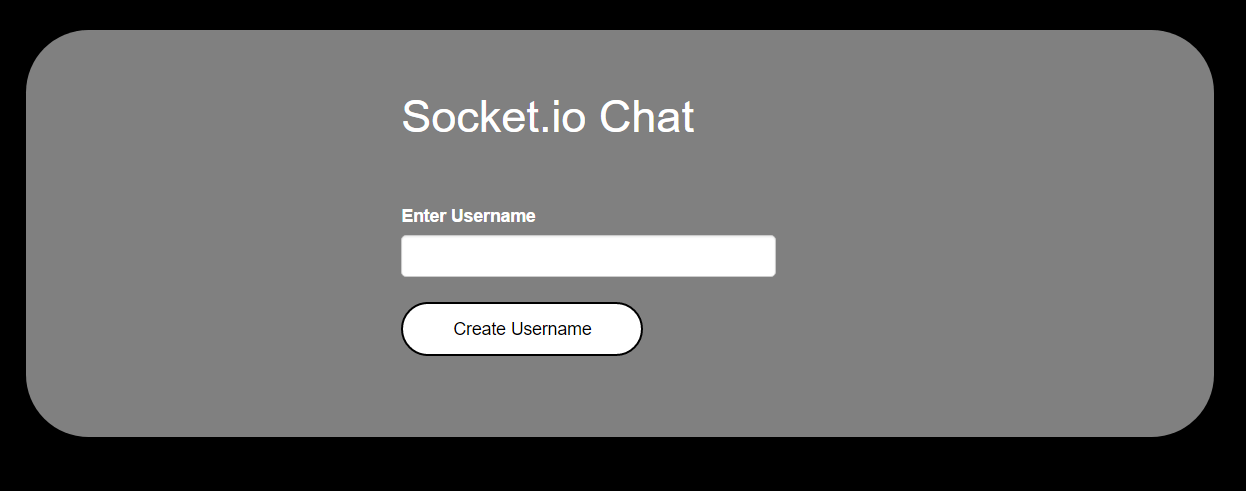
CA 3 Report

Daniel O’Toole N00145852

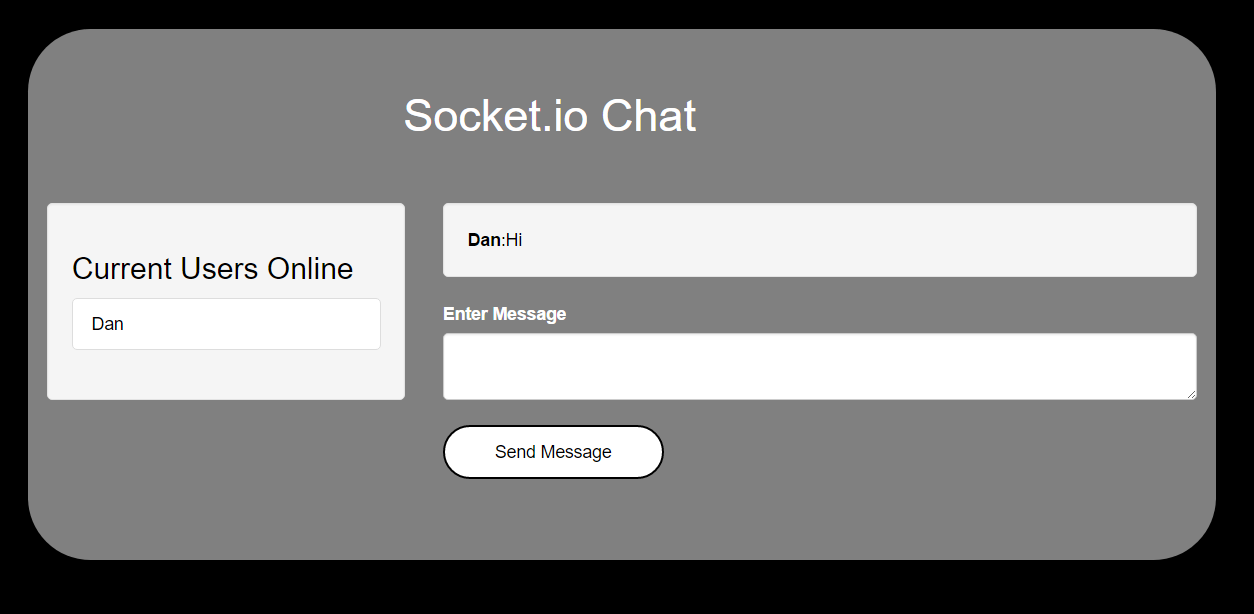
Introduction

For this ca we were tasked with making a web application with client and server sides in it. The ca was implemented with node.js, programmed in notepad++ and the server was runs through the command prompt program in any windows device. I used the Socket.io library which uses WebSockets to make the application work. For my ca I decided to make a chat room web application to keep it relatively simple as I am not extremely well versed in the Socket.io library. Originally I followed the tutorial on the socket.io Chat website and finished it but it didn’t have the username feature in it, so after looking through numerous website on the internet I was finally able to figure out how to add the username feature. The user can create a username then can chat with anyone else connected to the server.

Below you can see a screenshot of the create username web page. The user makes their own name and brings them to the chat room web page.

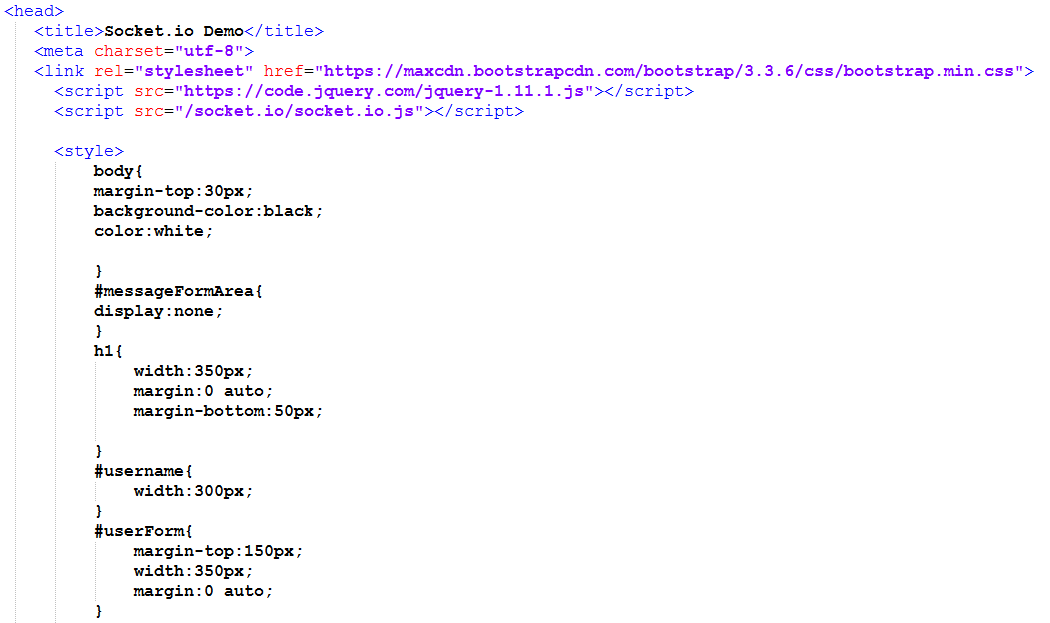


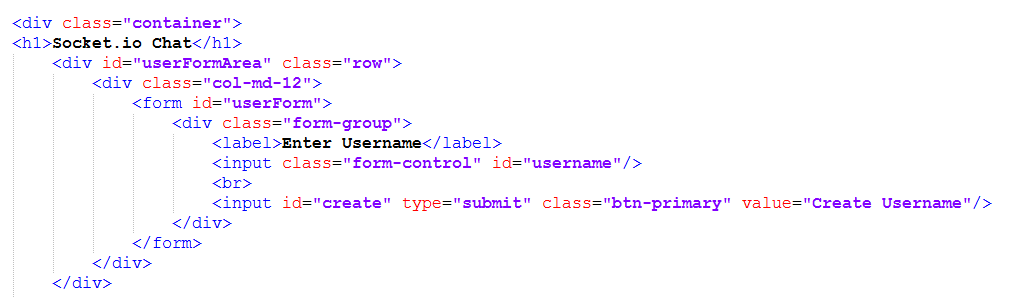
Below is another screenshot of the chat room where users can chat to each other. It also displays who is online to chat to. And displays who sends the message.



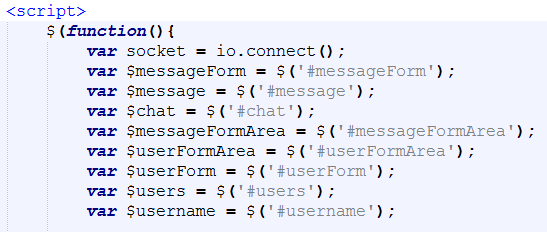
Index.html

In the index.html file I have the script type references, the css, the html and some jquery. This is where all the layout and styling happens for the web application.

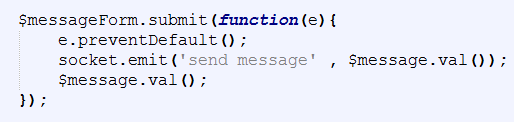
Below is a snippet of the script references and part of my css. I am using bootstrap in my web application.

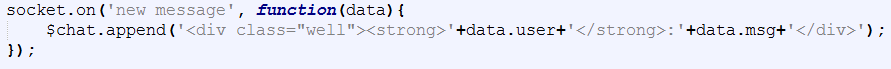
Below is screenshot of the create username form when you first start up the web chat application. I have it where when the user creates a username it hides the username form and shows the user message chat form. The user message chat form is the same just with different id names.

Below is screenshot of the start of the jquery code. I have defined the socket connection and the other various message and user variables as jquery type. An example of why I do this is so whenever I do something to $messageForm in the jquery code it will do something to the corresponding #messageForm id in the html.

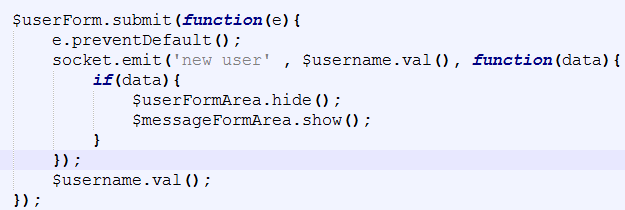


Below is a screenshot of the send message function in jquery. Socket.emit is a built-in function in socket.io library that can be used to display information. The ‘send message’ is a reference to the same named function in the server.js file. It then just takes the value of the message, the text, and sends it to the server where it sends it back in a new function.

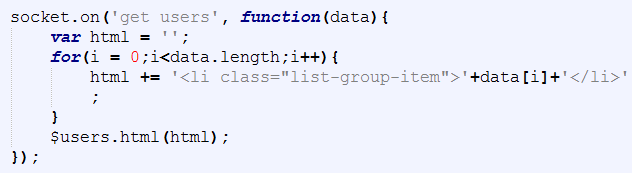


Below is a screenshot of the new or display message function in jquery. when the send message above is called it sends it to the server side then sends it back as ‘new message’ with the users name and their message.

Below is a screenshot of the username submit jquery code. Once the user presses the submit button it will hide the userFormArea and show the messageFormArea. And then submit the value of the username. Now a missing feature or a glitch of sorts is that the user doesn’t actually have to enter anything in the username submit form to be able to chat with other users, they will simply show up nameless, but with the time constraints of other cas I could not look into this in further detail.



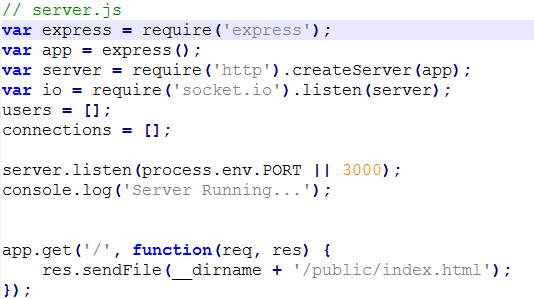
Below is a screenshot of the get users function or simply it display the current online users to the webpage. I’ve created a html variable that is equal to nothing but when a new user is added it adds one increment to the list group item

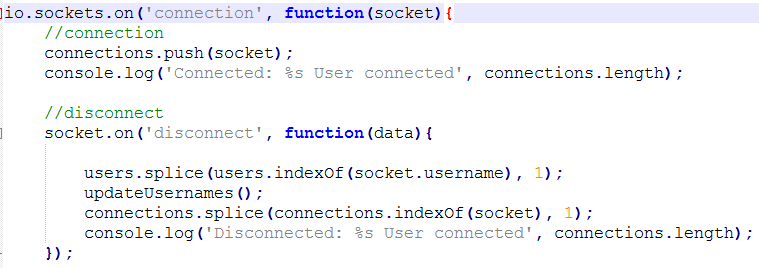


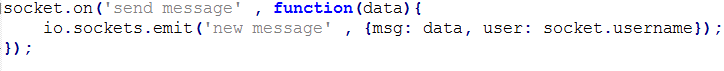
Server.js

This is the server side or backend of the web application. This is where the important code is. Where the html file sends data to and this file sends it back using the socket.io built-in features.

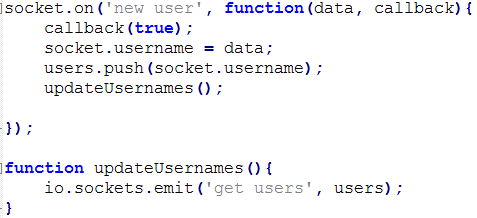
Below is a screenshot of the initial variables and arraylists for the server side file. This code sets up an express webserver and starts listening on port 3000 to start the webserver to run. Now you simply type into a web browser ‘localhost:3000’ and you get the html file connected with this server file on the screen. Below the port code is the connection between the server.js file and the index.html file.



Below is a screenshot of the start of the connection. First it push a socket connection into the connections arraylist and then posts to the cmd log that a user has connected, this is done to give someone looking at the server some feedback about how many users are connected. Next the disconnection part, when a user disconnects the server cuts them out of the username list and updates the current list. Next it cuts their connect from the index and then posts it to the cmd log.

Below is a screenshot of the continued connection code in server.js. this is the send message function that will be used here and in the html file. When the socket is on or open , the send message function is available. When the send message comes from the html file with message data the socket emits it back as ‘new message’ with the message data and the username of the user.

Below is a screenshot of new user and updateusernames, they work together. Basically when a new user is added it push the user into the users arraylist and updates the current online users list back in the jquery code in the html file.



Conclusion

I would have to say this ca was more difficult for me than I thought it was going to be as it was harder for me to grasp an understanding of the socket.io library. After looking through numerous web tutorials I finally managed to get a decent understanding of how socket.io works. I plan to develop my knowledge of socket.io in the future.