# *Web Programming III (420-H30-HR)*

# *Assignment 2 – Real-Time Web Sockets*

Date assigned: October 29, 2022

Date due: **Friday, November 18, 2022**

**Learning Objectives**

Upon successful completion of this assignment, the student will be able to:

* Create a working server in node using web sockets
* Complete a chat function between multiple people in web sockets
* Complete a simple game for two players using web sockets

To do:

**General idea of the assignment:**

Create a chat server that also allows the people to play a simple game of Rock, Paper, Scissors

**Details**

1. Create a web socket server using socket.io and express that will track connections and disconnections to the server. Set up chat functionality such that:
   1. People in the chat **must** specify a unique username (no spaces in the username) when they first connect. Usernames must NOT be case sensitive and must be unique (Elephant is the same as elephant)
   2. People in the chat can talk to everyone else
      1. When a message is displayed to everyone else over web sockets it must be displayed on the initiator's screen.
      2. A person’s own messages should be different than the messages received from other people in that they should be right justified in the window and, if desired, a different colour.
      3. When a message is displayed from someone, the person’s username must be displayed before the message; for example,   
         (user1) This is the message -OR-   
         <<user1>> This is the message
   3. People in the chat are assigned a random colour (think of the canvas animation lab in Web II) and no two colours are the same. Whenever anything is posted by the person their colour must be used to differentiate their messages (either as a border, the background colour, the text colour or the colour of their displayed name)
   4. People in the chat can send a private message to just one other person in the chat. This is done by prefixing their message with !*username* where *username* is the name of the person the message goes to.
   5. People in the chat can display a list of the usernames of all the users on the server by issuing the command !!users. This is displayed sorted by the username.
   6. People in the chat can invite other people in the chat to join a private room. If a person is invited, they must respond and then both are placed in a private room
      1. If another person is invited to the room, they are added to the existing room (the room does not need to be created a second time).
      2. A person can leave the room at any time by issuing the #leave command. When there is only one person in the room, the room is removed, and the person is put back in the main chat.
      3. Messages in a private room must be differentiated from messages in the normal chat (preferably by putting them on another part of the screen, but I am open to your design).
      4. There can be multiple private rooms on a server.
2. Two people in the chat can choose to play a Rock/Paper/Scissors (RPS) game together. To do this:
   1. A person either invites a specific person or asks if anyone wants to play RPS.
   2. When a person accepts, the two players are brought to a separate window (think a new div) within the chat.
   3. A countdown timer from 3 is displayed.
   4. When the timer is done, both people select whether they choose Rock, Paper or Scissors (use a button, an image, or something of your own choosing).
   5. The server determines who the winner is (Rock beats Scissors beats Paper beats Rock) and tells both people who won (or if it was a tie).
   6. The people can choose to play again.
   7. If either person chooses to leave or not continue, the game is over and the window is closed.
   8. Disconnecting from the server also disconnects a person from a game and ends the game.
   9. There can be multiple games being played at the same time.
   10. Players can continue to chat while playing a game.
3. The game playing area is separate from the chat area. That is there are at least two "windows" on each client, one for the chat and one for the game.
4. Events must be logged to the log file *date*events.log in the logs folder. Where date is the current date as yyyymmdd.
   1. Each entry is a single line that:
      1. Begins with the date and time in the format yyyy/mm/dd tt:mm:ss
      2. Next specifies the username of the person
      3. For a chat message, specifies the length of the message
      4. For a person-to-person chat message specifies the person receiving the message and the length of the message
      5. For a game message specifies the winner.
   2. If the log file MATCHING THE CURRENT DATE is not found, a new one is created. If it is found, it is appended to.

TECHNICAL NOTES:

1. You MUST set up a package.json file for the project and **CANNOT** include the node\_modules folder in what is handed in. I will run an npm install to set up the project.
2. Use classes where possible.
3. All reads and writes (appends) must be done using promises within the fs module.
4. You MUST handle the favicon.ico request. If there is no favicon, return nothing (empty response). If there is a favicon, read the icon and return it.
5. There are marks for interface design.

**To Do:**

1. Breakdown the assignment into smaller pieces and determine the order to implement. This document (called *initials*Breakdown.docx) MUST be handed in with the assignment.
2. Implement and test each piece. As a piece is integrated into the larger application make sure to (re)test everything.

**The Final 5:**

Doing everything about perfectly will get you a maximum 95%. To get the final 5% you need to do at least 1 of the following:

1. Add an avatar (selectable by the person but with a default) to identify the person.
2. Have the list of people (sorted by username) always display (and update) in a separate window on the chat screen.
3. Allow a person to be in multiple private rooms simultaneously.
4. Keep track of the wins, losses and draws in a game and display the current statistics when each round of the game is played.
5. Automatically disconnect a person who has not done anything on the server for 10 minutes.

**To submit**

When you have completed the assignment REMOVE THE node\_modules FOLDER, zip the folder containing all the files for the assignment and copy it to the course page.

**ADDENDUM**

* ERROR HANDLING!
  + Bad command
  + Multiple acceptance
  + Accepting when room is closed
  + Invalid username
  + Accepting game when invitation ended
  + *Many others I am sure*
* HOW DO YOU INVITE OTHERS TO JOIN A PRIVATE ROOM?
  + !invite username sends a private message (with room number) and requestors username
  + !join to join (this can be a button if you want)
* All commands that can be issued in the main room can be issued in a private room and can be seen “outside” the room
* HOW TO CREATE PRIVATE ROOM?
  + After invite is accepted, both people are moved into the room
* HOW TO JOIN EXISTING PRIVATE ROOM
  + Same way. Respond to invitation. Needs error checking in case someone has left and room has disappeared
* HOW CAN WE TELL IF TWO COLOURS CAN BE DIFFERENTIATED BY THE HUMAN EYE
  + Two methods have been provided.
    - randomColour returns a random colour in RGB format
    - areDifferent(colour1, colour2) returns true if colours passed are perceptibly different
* HOW ARE PEOPLE INVITES TO A GAME?
  + Use the command !play username to invite a specific user
  + Use the command !!play to put a message to public chat for anyone
  + Accept playing by entering command !accept
* LOG PRIVATE ROOM MESSAGES – YES
* HOW IS A REQUEST FOR GAME OR PRIVATE ROOM CANCELLED?
  + !cancel command cancels all outstanding requests