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

# *Assignment 2 – real-time web sockets*

Date assigned: Tuesday, October 6, 2020

Date due: **Wednesday, November 4, 2020**

**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 to six players using web sockets

To do:

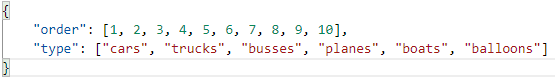
**General idea of the assignment:**

Create a chat server that also allows the people to play a simple game of "taking tricks."

**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 username (no spaces in the username)
   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 as well WITHOUT FIRST GOING TO THE SERVER.
   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. This is the colour of the box that goes around their text on the screen.
   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.
2. Users can choose to play a game together. To do this:
   1. All the users who wish to play, select the play game option (likely a button, but I'm open to other possibilities).
   2. Once users start to choose to play, a counter of the number of players waiting to play is displayed for all users and an indication whether or not the user seeing the message is waiting to play.
   3. The first user who chose to play has a button displayed that they can press to start the game after there are at least 2 people ready to play.
   4. If there are 6 users waiting to play, subsequent users receive a message that the play queue is full.
   5. A player can choose to leave the "waiting to play" queue at any time before the first user starts the game. If the first player chooses to leave, the next player in the queue becomes the first user and has the button displayed for them to start to play.
   6. Disconnecting from the server also disconnects you from the waiting to play queue.
3. The game playing area is completely separate from the chat area. That is there are two "windows" on each client, one for the chat and one for the game.
4. When the first player chooses to start the game, the server:
   1. Randomly choose a set to create from the file (more below)
   2. Reads the appropriate file.
   3. Creates a set of cards using the order and type read in from the file.
   4. Shuffles the cards (think last year).
   5. "Deals" one card at a time to each player from the deck until each player has received 9 cards. These cards are only sent to the individual player.
   6. "Turns over" the next card for everyone to see (this is the trump card)
5. After the hands are dealt, the server chooses a random player to go first. This player chooses a card to play from their hand. After the player leads, play follows in sequence up the list and back around to the initial player. The other players play one of the cards from there hand following these rules:
   1. Players must follow type if possible (if a Dog card is played, the player must play a Dog card).
   2. If the player has no cards of the type led, they can play any card including trump.
   3. Players cannot pass.
   4. When all players have played, the winner of the trick is determined as follows:
      1. The highest trump card played in the hand
      2. If no trump card has been played, the highest card played of the type that was led
   5. The server determines the winner of the trick.
6. The winner of the trick is the first to play the next trick.
7. Play continues until all 9 tricks have been played. When the game is over the winner is the player with most tricks. If two or more players have the same number of tricks the game is a draw.
8. 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 all the cards in the trick and winner of the trick
      6. At the end of the game specifies the winner of the game and the tricks for each player
   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.
9. Creating a deck is done as follows. The 'sets' folder contains a series of json files numbered sequentially 1 through 5. Each file contains a simple JSON object with 2 key value pairs. One pair has the key 'type' and the value an array of the type of cards. These can be anything such as animals, suits, colours or, as displayed below, methods of transportation. The second key is 'order' which lists the sort order of the cards in the deck from lowest to highest. This can be a number, letter or word. The 'order' determines which card wins the trick. The 'type determines which card is trump.

The 5 files provided are samples. I will use others when testing. For the basic game I will keep the sort order to be alphabetic (except for the final five below).



Once you read types and order (using an fs.readFile promise), create a deck of cards using the order and type combinations (cars 1, cars 2, cars 3, etc). In all cases you will end up with 60 cards in the deck. This is the playing deck and must be shuffled prior to play.

NOTES:

1. Players can continue to chat while playing the game.
2. Players cannot join a game already in progress.
3. When a game is complete, all players must sign up again to play again.
4. The game does NOT need to track if the player is cheating. That is done based on the honour system. The game only tracks who wins the trick and keeps track of how many tricks each player has won.

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. You MUST use classes where possible. Certainly, the deck is an obvious place for that.
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. You CANNOT use the design I provided in the examples

**The Final 5**

Doing everything about perfectly will get you a maximum 95%. To get the final 5% you need to do at least 2 of the following:

1. Add the ability for a player to have a VERY SMALL avatar with their name. How that IMAGE (yes it must be an image) is loaded, is up to you.
2. Ability for first player to choose the deck instead of choosing it randomly (and not just 1, 2, 3, 4 or 5…something more appropriate)
3. Add appropriate images to the deck to make them look like cards
4. Don't rely on the decks being named 1, 2, 3, 4 and 5. Read the folder (there is an fs method for this) and then read any of the files that is there.
5. Use the sort order as specified in the file, so the order could be random (2, 5, 8, 12, 6, 7, 1, etc) or (2, 3, 4, 5, 6, 7, 8, 9, 10, "Jack", "Queen", "King", "Ace", "Joker")

**To submit**

When you have completed the assignment zip the folder containing all the files for the assignment and copy it to the course page.