UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

COS 216 Homework Assignment

- Date Issued: **6 March 2023**
- Date Due: **22 May 2023** before **08:00**
- Submission Procedure: **Upload to ClickUP. An assignment upload will be made available.**
- Submission Format: **zip or tar + gzip archive**
- This assignment consists of **3 tasks** for a total of **130 marks**.

NB: Please read through the entire specification before asking questions on Discord/Email, as there is a high likelihood you question may be answered later in the specification.

1 Introduction

During this homework assignment you will be implementing Web Sockets. You will be creating a game that can be played in real-time with another user. This game will be called **BrandRace**. The general idea of the game is to guess the logo of a car brand quicker then your opponent. More details of the game and its rules will be explained below. On completion, your assignment must contain the following

- A PHP API hosted off Wheatley that pulls from a MYSQL DB.
- A local NodeJS socket server polling your PHP API from Wheatley.
- A Web client that connects to your NodeJS socket server and will act at as the front end.

2 Constraints

1. You must complete this assignment individually.
2. You may ask the Teaching Assistants for help but they will not be able to give you the solutions.
3. You must produce all of the source files yourself; you may not use any tool to generate source files or fragments thereof automatically. **This includes ChatGPT!**
4. All written code should contain comments including your name, surname and student number at the top of each file.
5. You assignment must be programmed in NodeJS, HTML, JS, CSS and PHP. (Bootstrap is allowed).
6. The API and Database should be hosted off Wheatley, the NodeJS server should be hosted locally.

3 Submission Instructions

You are required to upload all your source files and a ReadME in an archive, either zipped or tar gzipped, to **clickUP**. No late submissions will be accepted (See due date and time at the top of the document), so make sure you upload in good time.

4 Resources

NodeJS - https://www.w3schools.com/nodejs/
https://www.tutorialspoint.com/nodejs/index.htm

ExpressJS - https://www.tutorialspoint.com/expressjs/index.htm
https://expressjs.com/
https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs

Web Sockets - https://developer.mozilla.org/en-US/docs/Web/API/WebSocket
https://en.wikipedia.org/wiki/WebSocket
https://github.com/websockets/ws
https://socket.io/
https://www.npmjs.com/package/websocket

Base64 https://www.w3docs.com/snippets/html/how-to-display-base64-images-in-html.html
https://www.base64-image.de
https://www.base64decode.org/

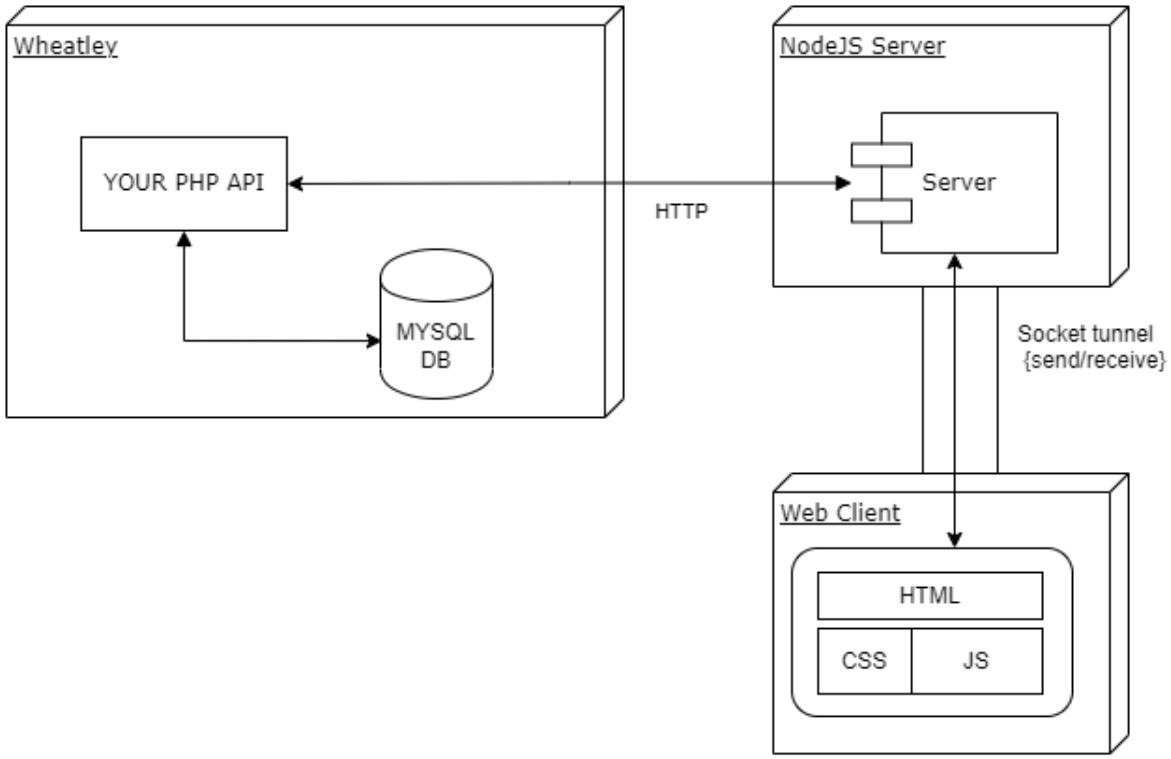
DOM Manipulation https://www.w3schools.com/js/js_htmldom.asp

5 Rubric for marking

Task 1 - PHP API & DB	**25**
MySQL DB	2
Ability to store images in the DB	5
Ability to retrieve images from an endpoint	8
The image(s) should be random every API Call	4
ReadMe explanation, 1 vs 5 images at a time	3
ReadMe explanation, Base64 vs Link	3
Task 2 - Multi-User server	**60**
Port can be chosen at run-time	3
Reserved Ports cannot be chosen	2
Server can accept multiple clients simultaneously	5
'LIST' command is implemented and fully functioning	4
'KILL' command is implemented and fully functioning	5
'QUIT' command is implemented and fully functioning	4
'GAMES' command is implemented and fully functioning	5
If connection is closed on the client the server closes the socket	5
Server keeps track of usernames	3
Server can detect When a game should start and start the game on the user side	6
Server can send through Images to the client using sockets and the API	5
Server can generate a unique GameID	3
Server implements the flow of the GameLoop correctly	10
Task 3 - Web Client	**40**
The client can connect to a server through a socket	3
The client allows the user to choose a username	3
The client allows user to enter OR generate GameID from the server	3
The client starts the Game Loop on the user side on cue from the server	5
Error Messages(Username taken, Socket disconnected, Opponent Gets a point, etc.)	8
The ability to implement the Game loop correctly	18
Security	**5**
Upload **Not uploaded to clickUP**	 **-130**
Bonus	**15**
Total	**130**

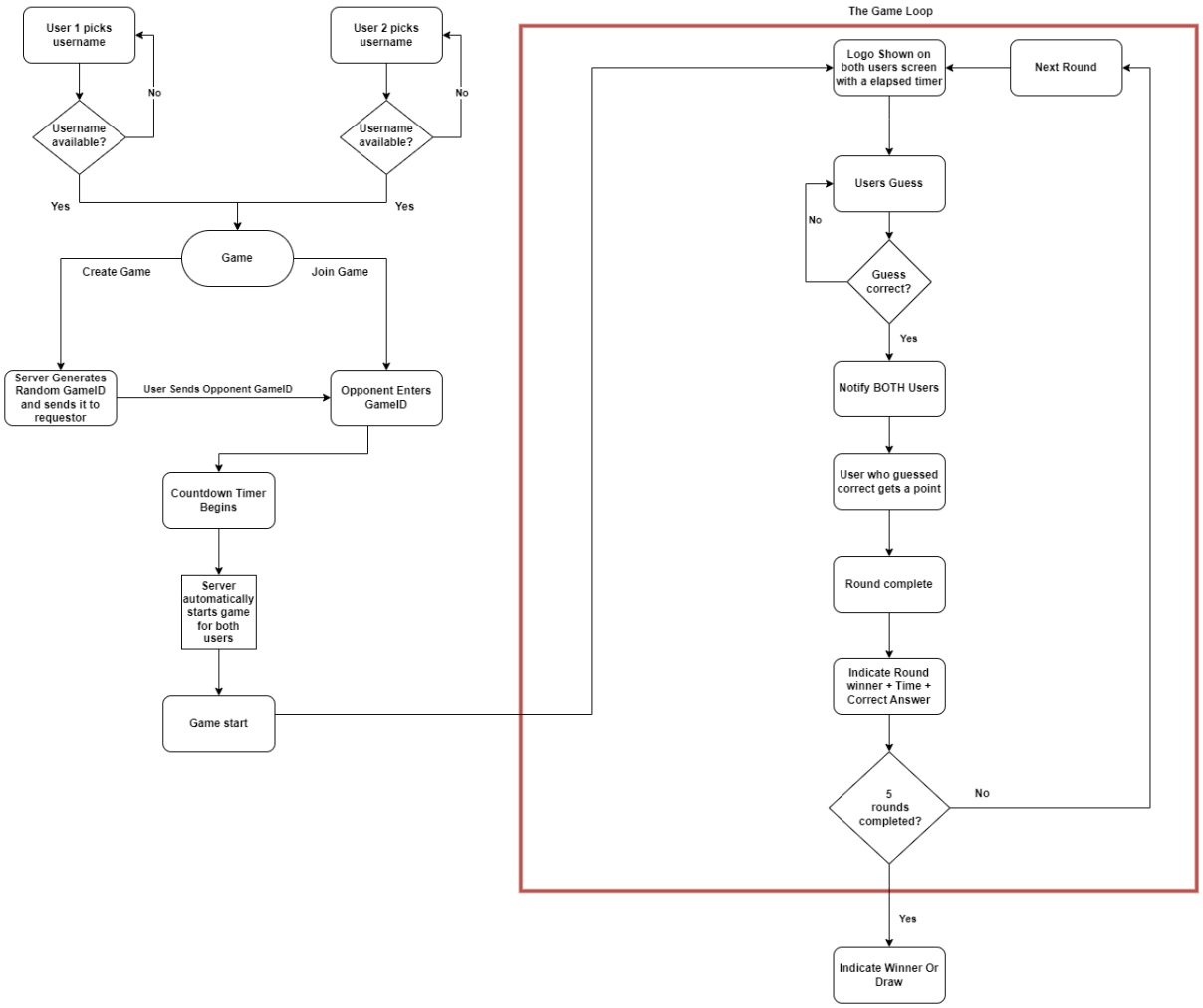
6 Overview

To understand the flow of how this all works please consult the diagram below.



7 Game details

The aim of BrandRace, is to guess a Car Brand Logo faster then your opponent. There will be a total of 5 rounds, in each round the players will see a new Logo on screen and need to enter the correct name to get the point, whoever does this the quickest wins the round. The winner is the player that has won the most rounds. However you are also racing against the clock, each player will have limited time each round and if time runs out before any player guesses the correct logo then no one gets the point. To make the game a bit easier to follow please consult the diagram below. The Area highlighted in Red will be referred to as the "Game Loop" and is referenced throughout the specification. Before the game begins a user chooses a username and has the option to either generate a GameID or enter in a GameID. When a valid GameID is entered it automatically starts the Game Loop **After a 5 second countdown on both screens**. This is a 2 player game (1 vs 1), if someone else enters in a GameID that is already in progress (game started) they should not be allowed to join in the game and an appropriate error message should be shown.



8 Assignment Instructions

Task 1: PHP API + Database ..... (25 marks)

You need to create a PHP API that is hosted off Wheatley as well as a MYSQL DB along with it. This API should have one endpoint, `GetRandomBrands`. In this endpoint you can choose between returning 5 random Car brand logos with their names OR 1 random Car brand logo with its name that should be retrieved from the MYSQL Database. There are 2 options when dealing with images. Since images themselves cannot be transferred through an API you need to store the images in Base64 format in the database and return the Base64 String. (Consult the resources about working with Base64 at the top of the doc) OR Alternatively you can send through a link to the image that is hosted somewhere on the internet which you will then display. You can populate the DB manually with PHPMyAdmin for this assignment. Below is a summary of what is mentioned above:

You have 2 options of how you can set up the **API**. *There is no incorrect answer here, but you need to motivate your choices*

- (5 At once) - This option will return an array of 5 different Car Brand Logos + Names in a single API Call
- (1 At a time) - This option will only return 1 Car Brand Logo + Name at a time

You will need to write a short explanation of your choice and why you believe its the better option. Please write this is a ReadMe.txt that you should include with your ClickUp submission.

You have 2 Options of how you can **transfer images**. *There is no incorrect answer here, but you need to back up your choice*

- Base64 - you will store images in base64 and decode them client side
- Link - you will use a link to an external image ***e.g.*** *https://images.pexels.com/photos/63294/autos-technology-vw-multi-storey-car-park-63294.jpeg?auto=compress&cs=tinysrgb&w=1260&h=750&dpr=2*

You will need to write a short explanation of your choice and why you believe its the better option. Please write this is a ReadMe.txt that you should include with your clickup submission.

Task 2: Multi-User NodeJS server ..... (60 marks)

For this task you are required to create a **Multi-User** socket server **(on localhost)**. The server must be coded in **NodeJS locally and not off wheatley**. You are allowed to use libraries for this.
Note: Wheatley is a web server that follows the LAMP stack and installing other applications on it brings security issues. You must therefore use localhost for this since the NodeJS server needs to run on an open port and it would be difficult to allow the server to be run on Wheatley as the chances of students using the same port would be high. Also, this is done to avoid some students from intentionally and unintentionally performing port blocking. This is also not possible to do due to UP's firewall.

The server must have the following functionality:

- Be able to specify a port that the server will listen on at run-time (create the socket), using input arguments or prompts. It is good practice to block the user from using reserved ports (Ports that have been reserved for other task). However there is still a chance you can choose a port in use, for the sake of this assignment if this happens, just try a different port. The ports you are allowed to use are from 1024 -¿ 49151, make sure the user can only choose a port in this range. *If you are curious about why this is the case, you can read up about it at https://www.techtarget.com/searchnetworking/definition/port-number.*
- Block the user from attempting to open a Reserved Port (Explained above).
- Accept multiple client connections and interact with the clients concurrently through sockets.
- **The server must utilize functionality of the PHP API you developed in Task 1.** That means your server should contact your PHP API to get the car brands for the game based on the options you chose.

- The server should have a `KILL` command to close a connection based on their username. This also means that you need to keep track of which socket ID corresponds to which Username since you need the SocketID to close a socket connection.
- The server needs to account for lost sockets (when a socket was created and the client [HTML web page] has closed/disconnected). These lost sockets should be closed in order to allow for new connections. If that user was in a game then the game should be stopped and the other user notified (This check can be done in the game loop).
- The server should have a `LIST` command to list all the connections it currently has, be sure to list the usernames and their respective SocketIDs.
- The server should have a `QUIT` command that sends a broadcast to all connections notifying that the server will now go off-line, thereafter closing all connections.
- The server should have a `GAMES` command that shows all games in session or created, The GameID and the players usernames.
- The server should be able to keep track of usernames, i.e a when a user requests a username the server determines if its in use, prompting for a different username.
- The server should be able to generate a Unique GameID. This should be a random 10 Character Alpha-Numeric String.
- The server should be able to start the game loop automatically if a user enters in another users GameID (remember the 5 second countdown).
- The Game loop, Once the game loop has started take note of the following from the server side
  - The server should notify the user when the game is about to start
  - The server should keep track how many rounds each user has won
  - Detect if one opponent disconnects and end the loop and notify the other
  - The server should be able to implement what is shown in the Flowchart under Game Details
- Since Wheatley is password protected, you will need to include your login details in the URL as follows:

  username:password@wheatley.cs.up.ac.za/uXXXXXXXX/path_to_api

  It is your responsibility to keep your login details as safe as possible. It is recommended that you store your username and password in a global variable and use that variable throughout. Alternatively, you can secure your details through other means. **Bonus marks may be given depending on how secure your solution is**.
  NB: Ensure that you do not submit your code with your passwords included in your clickUP submission.

To test the functionality of your server you may use https://www.piesocket.com/websocket-tester as the client before proceeding to Task 3.

Task 3: Web Client ..... (40 marks)

For this task you are required to develop a web page / web site that will interact with your server (that runs on your local machine [localhost]) you wrote for Task 2. The web client must be implemented in HTML, CSS and JS using Web Sockets. **The client should also be on localhost**.
Note: You may use any client side library of your choice (e.g. WebSockets, Socket.io, etc.). To make the changes you can/should make use of DOM Manipulation with javascript, that way you do not need to redirect to a new page everytime which might disconnect the socket

The client must have the following functionality:

- The user should be able to enter in their username. If that username is not available then an error should be shown, and be allowed to enter in a new username.
- After the user has received a username they have 2 options

  - Enter in a GameID. If the GameID exists and the game in not im progress should trigger the game loop.
  - Generate a GameID (From the server) start the game loop when another user enters in the GameID.
- The Game Loop
  - Once the server starts the game loop the screen should update to show the logo with a elapsed timer, and the ability for a user to enter in the brand name. Here you are free to chose how the user inputs the name.
  - The user should be able to guess the logo and be notified if they get it right or wrong (remember this needs to happen in real-time). When a user gets it wrong, they should be allowed to try again, provided their opponent has not got it correct, and vice versa.
  - The user should be notified if their opponent gets it right and what the correct answer was before moving onto the next logo. To aid user experience use a countdown timer before moving to the next logo.
- At the end of the game both users should be notified of the winner unless it was a draw.

Note: Bonus marks may be given to students who add extra functionality to them. A maximum of 15 bonus may be awarded (up to 10 for standard extra functionality but to achieve 15 you must add something extra-ordinary!)