***Documentation Packet [ 07 04 10 24 ] Oct 4th 2024***

|  |  |
| --- | --- |
| Student Name: |  |
| Goals:  1. Collect HTTP POST Request parameters 2. Save to a database using NodeJS | Events:  1. DocPac Due *Next Monday* 2. Pep Rally Friday 3. Homecoming Saturday |
| Included Documentation  1. High Score Database 2. Home Game Server [Bonus] | Required Documentation:  1. High Score Database 2. Reflections |
| Changes/Notes:  * None | |

# Home Game Server [Bonus]

1. Create a new NodeJS project
2. Place a HTML game you made in the ‘public’ folder, and enable reading from said public folder in your NodeJS project.
   1. You can use “Button Masher”, “Touch Controls”, “Canvas Mover”, your own game, or try to use your RPGMaker game from 9th grade.
3. Create a GET endpoint that uses **sendFile()** to send the main page of your game to the user
4. Get your parent’s permission to log into the home internet router
5. **Open the port** used by your NodeJS project
6. Find your **external IP address**
7. **Find or make a friend**
8. Get your friend to go to your external IP address at your NodeJS port and play your game
9. **[Bonus Bonus]** Buy a domain name online, and create an **A Record** to send users to your external IP address
10. **[Bonus Bonus Bonus]** At the end of the game, use client side javascript to send a POST request back to the NodeJS server, which uses the user’s IP address to update their high score in a file (see HTML Forms for most of this data).

# High Score Database

1. Create a new NodeJS project with ExpressJS and EJS
2. Import the `sqlite3` npm module
3. Create a database file using DB Browser (SQLite)
   1. Create a table called “scores” with four columns:
      1. “uid” is unique, not null, primary key, and auto incrementing
      2. “ip” is not null
      3. “name” is not null
      4. “score” is an integer/number
4. Create a ‘/’ GET endpoint that renders an ‘index.ejs’
   1. This links to ‘/hiscores’ and ‘/game’
5. Create or import files for a game you’ve made that has a scoring system
   1. Create a way (such as using prompt()) to get the player’s name
   2. When the game is over, send a POST request using Javascript with the user’s name and score
      1. XMLHttpRequest() is the most common method in browsers
6. Create a ‘/game’ GET endpoint that renders your game
7. Create a ‘/hiscores’ POST endpoint that reads the data from your game
   1. If the name and score are valid, get the requester’s IP address
      1. If it is not, render ‘error.ejs’ with the error code
   2. Use sqlite3 module’s run() method to run an SQL INSERT query to insert a row with the user’s IP, name, and score
8. Create a ‘/hiscores’ GET endpoint that renders a ‘hiscores.ejs’ template
   1. Before rendering the template:
      1. Use sqlite3 module’s all() method to run a SELECT query that gets all entries from the scores table
         1. If there is an error, render ‘error.ejs’ with the error code
      2. Sort all of the entries in descending order using sort()
      3. Use splice() to get only the top ten scores
   2. Render the template with the array of the top ten scores
   3. In the template, make a table to show each entry’s name and score
9. Create an ‘error.ejs’ to handle errors in the above steps

# Reflection

**What personal projects have you worked on recently? What do you think the impact is of doing/not doing personal projects has been?**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**What was one mistake you made in school or otherwise that you can recognize? What can you do in the future to prevent it from happening again?**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**How difficult was the assignment this week? What made it difficult/easy for you?**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**What are more ideas you can create now that you have completed the assignment for this week?**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

# A picture containing text, monitor, screen, clipart Description automatically generatedGrading

|  |  |  |
| --- | --- | --- |
| 10 | You went above and beyond expectations. You applied knowledge that was not taught in this class in addition to what was taught. Additional rewards are given | * ***If the assignment does not have its own rubric, it will default to the rubric on the left.*** * All assignments start at 10/10 possible points * 1 point is deducted per infraction   + Lateness   + Mistakes   + Unprofessionalism   + Not following instructions * Outstanding submissions, or submissions on assignments not marked in “Required Documentation” can reward pogs |
| 10 | You performed as well as can be expected for this class. You show a complete understanding and made no mistakes. You have mastered the subject. |
| 8 | Assignment is complete. You show a good understanding of the subject, but there are mistakes or minor incorrect details. You are ready to move to new subjects. |
| 7 | You show and understanding of the subject, but there are serious errors, or there are pieces you can practically use without understanding them. Remediation needed. |
| 6 | Assignment is incomplete but/or you showed that you understand at least the fundamentals of the subject. Assignment is low effort. Serious need of remediation. |
| 5 | You show minimum effort, assignment is incomplete, or have serious mistakes. You did not demonstrate that you understand the content or purpose of the submission. |
| 0 | The work was not submitted, damaged, seriously incorrect, or unprofessional. The submission is rejected. |
|  |  |  |

# High Score Database

|  |  |  |  |
| --- | --- | --- | --- |
| Pull Request correct | Works as described | Did not commit  ‘node\_modules’ folder |  |

# Reflection

|  |  |  |  |
| --- | --- | --- | --- |
| Selected an answer for each question that is unique to you and this week | Answered every question in each prompt | Answers were not repeats of previous weeks | Answers were not copies of assigned work this week |