***Documentation Packet [ 02 29 08 25 ] Aug 29th 2025***

|  |  |
| --- | --- |
| Student Name: |  |
| Goals:  1. Create a basic NodeJS HTTP server with ExpressJS 2. Use Gamepad controls in a webpage 3. Get ready to begin CIW training | Events:  1. Picture Day Tuesday 2. No School Next Monday |
| Included Documentation  1. Button Masher 2. NEE Stack Intro 3. CIW Setup 4. NTHS Application 5. Syllabus | Required Documentation:  1. Button Masher 2. NEE Stack Intro 3. Signed Syllabus (printed) 4. Reflection |
| Changes/Notes: | |

# Syllabus

1. Review the attached Syllabus with your parents
2. Have your parents write their name, sign, and date the Syllabus to agree
3. Write your name, sign, and date the Syllabus to agree
4. Remove the back page of the Syllabus and submit it inside of this DocPac

# CIW Setup

1. Log in to your school email.
2. Look for an email from uCertify or CIW
3. Use your login information in that email to log in to [www.ucertify.com/](http://www.ucertify.com/)
   1. Contact the teacher if you need your assigned password
4. Change your password
5. Make sure you are in the 2526 section with the course “JavaScript Specialist (1D0-835) v3”

# NTHS Application

1. Review the Nation Technical Honor Society document
2. If interested, return the completed application to the instructor ASAP.

# NEE Stack Intro

Create a new NodeJS project to practice NodeJS, ExpressJS, and EJS (N.E.E.). Use your senior mentor and online resources to complete the following tasks. As a matter of self-educating, practice reading online tutorials and documentation instead of asking ChatGPT.

1. Create and initialize a NodeJS project
   1. npm init
2. Install ExpressJS into your project
   1. npm i express
3. Import ExpressJS module in your application’s main script and start an Express app object
   1. const express = require('express');
   2. const app = express();
4. Configure ExpressJS to use the EJS templating engine
   1. app.set('view engine', 'ejs');
5. Create a HTTP Listen Server with Express
   1. app.listen(3000, () => { /\* log success here \*/ });
6. Create a root (‘/’) GET endpoint that responds by **sending** an HTML string
   1. res.send('Welcome to My App!!');
7. Create another GET endpoint that handles the request in the following way:
   1. Create a “name” variable in the endpoint’s callback function whose value is “Guest”
      1. let username = 'Guest';
   2. If the request had a “name” **query parameter**, change “name” to the value of that parameter
      1. if (req.query.name) { username = req.query.name; }
   3. Respond by rendering an EJS template of a basic web page
      1. res.render(templatefile, { name: username });
8. Create the .ejs template and put it in a “views” folder
   * 1. templatefile.ejs
   1. The template must have a header that says “Hello, <name>”, where <name> is the value of the “name” variable.
9. Test to make sure your name appears in the template if you give the endpoint a “name” query parameter ( *localhost:3000/endpoint?name=yourname* )

When you are finished:

1. Test every feature thouroughly!
2. Attempt to break your program every way you can
3. Double check for adherence to instructions.
4. Delete the ‘node\_modules’ folder
5. Follow the Github Submission Instructions to submit

# Reflection

**What did you do to understand the instructions in your assignments this week? What were the advantages/disadvantages of doing it this way?**

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

**How are you doing on managing and completing all of your class’s work this year? What are the advantages/disadvantages of your current strategy to complete the work?**

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

**What is one project you are actively working on in programming outside of the classwork? What strategy do you have to complete it? If no project, why not?**

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

**How is the Junior/Senior pairing working for you?**

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

# Grading

# Button Masher

|  |  |  |
| --- | --- | --- |
| **1** | Followed Submission Instructions |  |
| **2** | Pull Request was valid and accepted |  |
| **3** | Can read direction/button input and increase score |  |
| **4** | Timer ending prevents score from increasing |  |

# NEE Stack Intro

|  |  |  |
| --- | --- | --- |
| **1** | Followed Submission Instructions |  |
| **2** | Pull Request was valid and accepted |  |
| **3** | Server can run |  |
| **4** | Sends an html string when you go to “/” |  |
| **5** | Second endpoint can read the user’s name with a query parameter |  |

# DocPac and Reflection *DocPac Submission Rules, DP09*

|  |  |  |
| --- | --- | --- |
| **1** | DocPac is turned in on time |  |
| **3** | a. DocPac is neatly folded |  |
| **3** | b. DocPac is not stained or damaged |  |
| **3** | c. No doodles, scribbles, or unnecessary writing |  |
| **4** | d. Answered each question in each prompt fully (no short answers) |  |
| **4** | e. Spelling and handwriting |  |
| **4** | f. No repeated answers from other DocPacs |  |
| **4** | g. Did not paraphrase assigned work |  |
| **6** | You are prepared to justify the use of any AI (you know what it does and why) |  |