WEB322 Assignment 1

# Assessment Weight:

5% of your final course Grade

# Objective:

Create a Node.js application utilizing the "fs" and "readline" modules for file and directory interaction. This program will not only read and analyze data from a specified file or directory but also perform additional tasks as outlined below.

### **Step 1:** Installing Software

* Install Visual Studio Code and Node.js.
* Create a folder for the assignment and call it “a1\_name\_studentnumber”, where ‘name’ is your first and last name and ‘studentnumber’ is your student number.
* Download the provided "data" directory and "log.txt" zip file. Unzip and place them in your assignment folder.
* Open the folder in Visual Studio Code.
* Use node’s “npm init” function to make a new package.
* Set the entry point to "a1.js”
* Create the a1.js file
* Your assignment folder should now contain:
  + (Assignment Folder)
    - data
      * bananas.txt
      * carrots.txt
      * onions.txt
      * tomatoes.txt
    - a1.js
    - log.txt

### **Step 2:** User Input

Modify a1.js to determine if the user wants to process a file or a directory. Implement user prompts using the "readline" module.

Example prompts and responses (user responses in green):

* Do you wish to process a File (f) or directory (d): f
  + File: log.txt
    - TODO: Process file log.txt
* Do you wish to process a File (f) or directory (d): d
  + Directory: data
    - TODO: Process directory data
* Do you wish to process a File (f) or directory (d): xyz
  + Invalid Selection

### **Step 3:** Processing the file

Using the “fs” module, when processing a file (e.g., "log.txt"), generate a detailed report with these additional metrics:

* Number of lines in the file.
* Number of words
* Number of characters
* Frequency of each alphabet letter (case insensitive).  
    
  **HINT**: Make these functions so you can reuse them for Step 4. You will lose marks if you have repeated code that does the same thing.

### **Step 4:** Processing a Directory

When processing an entire directory (e.g., "data"), it must ONLY process .txt files. It must also include the output from Step 3 for ALL of the files in a directory, in addition to:

* The total number of files in the directory.
* The cumulative size of all files in the directory.

**NOTE:** If the file and/or directory cannot be read, output the error to the console using "console.log(err.message);"

## Assignment Submission:

1. Add the following declaration at the top of your a1.js file.  
     
   /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
   \* WEB322 – Assignment 1  
   \* I declare that this assignment is my own work in accordance with Seneca Academic Policy.   
   \* No part of this assignment has been copied manually or electronically from any other source  
   \* (including web sites, GPT) or distributed to other students.  
   \*   
   \* Name:   
   \* Student ID:   
   \* Date:  
   \*  
   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
2. Compress (.zip) the files in your Visual Studio working directory (this is the folder that you opened in Visual Studio – it should contain a **node\_modules** folder, a **server.js** file and a **package.json** file)

## Important Note:

* **NO LATE SUBMISSIONS** for assignments. Late assignment submissions will not be accepted and will receive a **grade of zero (0)**.
* Submitted assignments **must**run locally, ie: start up errors causing the assignment/app to fail on startup will result in a **grade of zero (0)** for the assignment.
* After the end (11:59PM) of the due date, the assignment submission link on My.Seneca will no longer be available.

**Marking Scheme for WEB322 Modified Assignment 1 (Total: 30 Marks)**

#### Part 1: User Input (6 Marks)

* Correct implementation of user input using "readline": 2 Marks
* Proper handling of user responses for file or directory selection: 2 Marks
* Accurate handling and validation of invalid input: 2 Marks

#### Part 2: File Processing (12 Marks)

* Correct processing of the selected file: 3 Marks
* Accurate reporting of the number of lines in the file: 3 Marks
* Correct frequency analysis of each alphabet letter (case-insensitive): 3 Marks
* Clear and concise output formatting: 3 Marks

#### Part 3: Directory Processing (12 Marks)

* Correct listing of files in reverse alphabetical order: 3 Marks
* Accurate count of total files in the directory: 3 Marks
* Correct calculation of the cumulative file size: 3 Marks
* Implementation of word frequency analysis: 3 Marks

### Notes:

* 10% will be deducted automatically if your assignment folder is not named correctly and your entry point is incorrect.
* 10% will be deducted for any program that does not RUN and for EVERY error that needs to be corrected in order to make it run, up to a maximum of 50%.
* Partial marks can be awarded for partially correct implementations.
* Clear and efficient coding practices, along with proper code comments and formatting, are expected throughout the assignment.