University of the Philippines Los Baños
CMSC 100 - Web Programming Laboratory Exercise

# Exercise 04: JavaScript Basics part 2[1]

**Title: Simulate a password validation program in JavaScript**
**Type: Individual Assessment**
**Score: (15 points)**

**Problem:** Create 2 JavaScript files and name it as **index.js** and **test.js**. You will test your index.js by calling them in a 2nd file called test.js. Do not use Regular Expressions.

**Tips:** You may use any export style

- Use the **fs**, **uuid**, and **validator** packages.
- You do not need to run **"npm install fs**". It's built into NodeJS.
- To write files, use appendFileSync() - https://nodejs.org/api/fs.html#fsappendfilesyncpath-data-options
- UUID package: https://www.npmjs.com/package/uuid
- Validator package: https://www.npmjs.com/package/validator

**Program specifications:**

1. Create a function named **`generateUniqueID()`** in your index.js
   - The input parameters are first name (string), last name (string)
   - Create and return a unique id by concatenating the first letter of the first name (converted to lowercase), the last name (converted to lowercase), and a "unique" alphanumeric string of length 8.
   - Use the **`uuid`** package to get a unique alphanumeric string. Trim as needed.

     e.g.
     generateUniqueID("Alan", "Turing")

     returns: "`aturing5133f34e`"

2. Create a function named **`addAccount()`** in your index.js
   - The input parameter is an array with the following contents: first name (string), last name (string), email (string), age (number)

     e.g `addAccount(["Alan", "Turing", "aturing@w3c.com", 58]);`

[1] Prepared by CAG Angcana

- If the following conditions are true
    - all fields are present
    - the first name, last name, and email are non-empty strings
    - the email is in a valid format (use the validator package)
    - age is at least 18

- save the data into a new line of file called `users.txt` in the following format:
- *first name,last name,email,age,uniqueID*

    e.g.

| **users.txt** |
| --- |
| `Tim,Berners-Lee,tim@w3c.com,25,aturing5133f34e`<br>`Ted,Nelson,ted.n@w3c.com,43,tnelson0cfa6312` |

- Use the function from Item #1 to generate the uniqueID for the user.
- If the user was saved, return true. Else, return false

3. Create GITIGNORE:
    - On your terminal, type in nano .gitignore
    - Write:

```
node_modules/*
```

***Submission Guidelines:*** *Accept the invite link posted in the Google Classroom's exer 5 to create your own GitHub classroom repository. Create a file* **index.js** *and push it into your repo. Push your files in your Github repository (index.js, test.js). Make sure to include the package.json file. DO NOT INCLUDE THE node_modules/ folder. Kindly put a .gitignore file so that the node_modules will not be pushed to your remote repository. Do not forget to document your code with comments.*

*After you are done with your exercise, click the mark as done button in the Google Classroom Assignment.*

Scores breakdown:

| Criteria | Description |
| --- | --- |
| Use of GitHub<br>(5 Points) | ● Commit Messages - 3 points<br>● Documentation/ReadMe -1 point<br>● Use of GitHub classroom's repository for GitHub Pages - 1 point |
| Application of JavaScript Syntax<br>(10 Points) | ● Use of JavaScript Syntax (No Regular Expressions) / See Specs |