PROFICIENCY TEST

Thank you for taking the time to attempt this test.

As a business, we encourage **innovative thinking** and **initiative**. In these tests, we are looking for those elements, but we strongly urge you to consider what has been asked for. Anything above what is asked is a bonus but, due to time constraints, may hinder your progress.

The test should be conducted in the **language of your choice** and submitted to a **GitHub repository**. Make sure you can run your solution on your own computer during the review process.

Please do not make use of a framework. We want to understand your competency in the language of your choice.

Make sure to follow all the instructions carefully.

TEST 1

Create an HTML form with the following input fields to allow for the capturing of data into a Mongo database:

- **Name**
- **Surname**
- **ID Number**
- **Date of Birth**
- **POST button**
- **CANCEL button**

Create a Mongo database with a relevant schema to store the input fields in.

REQUIREMENTS:

- Save 3 records into the database without duplicating the ID Number. The ability to capture a duplicate ID Number in the database table is an immediate fail.
- If a duplicate ID Number is found upon capturing, the user must be informed about this and the form repopulated. People do not like to input their information twice.
- Validate the ID Number field to ensure it is a number and exactly 13 characters long.
- Validate the Date of Birth field to ensure the input date is in the format dd/mm/YYYY.
- There must be valid data in the Name and Surname fields, with no characters that could prevent the record from being inputted into the database.

PASS:

- Validation works as per requirements.
- The Date of Birth field is captured correctly and is stored properly in the database.
- The ID Number field is no more than 13 characters long. (Bonus: Check match with Date of Birth.)
- No duplicate ID Numbers are in the database, and the user is made aware of this.

TEST 2

This task is to test your skills in manipulating arrays and file handling.

This task is to test your skills in manipulating arrays and file handling.

In this test, you will create a CSV file of variable length. A form will ask for the amount of data to generate. Check the requirements for generating the file.

The file will have the following header fields:
Id, Name, Surname, Initials, Age, DateOfBirth.

The data will look like this:

```
"1", "Sean", "Pompeii", "S", "33", "13/02/1979"
"2", "Sasha Cohen", "Hall", "SC", "32", "03/06/1980";
```

After this, you will import the file into a SQLite database and output a count of all the records imported.

REQUIREMENTS:

1. Create two arrays, one for names and one for surnames. There should be 20 names and 20 surnames in each array. Use these arrays to generate random names, ages, and birth dates to populate a CSV file. The initials are always the first character of the name. Write a function to perform the task of creating the CSV file; you should pass it the number of variations you need.
2. The CSV file should be outputted to an output folder, and the name of the file must be `"output.csv"`.
3. An input field will specify the number of records to be generated.
4. There should be **NO DUPLICATE ROWS IN THE CSV**. The name, surname, age, and date of birth must be unique.
5. Output a CSV file with 1,000,000 records.
6. Import the file using a form variable of file type. Users should be able to browse for the file and upload it to the website.
7. Create a table called `"csv_import"` with the relevant fields and types to hold the data from the CSV file. Use code to create the table.

PASS:

- The file is outputted to the correct folder with the correct name and the correct number of records. (E.g., a 100-record file will have 101 lines, including the headers.)
- The code includes the correct arrays of names and surnames.
- The file is generated according to the specification of no duplicate rows.
- All instructions are completed (1-7).

FAIL:

- The data imported into the database is not correct (check dates).
- There are duplicate rows in the CSV file.
- The test file to generate 1,000,000 records fails for any reason.
- The import of this large file fails for any reason.

In case you were wondering, SQLite can handle up to 14 terabytes of row data, so 1,000,000 records should be manageable, and there should be enough disk space for 1,000,000 records.