SV Database Assessment

As the next step in the application process, we request that you build a small web application to show off your technical knowledge, skills, and performance in a simulated work environment.

This project is expected to take anywhere from a half a day or full day depending on your coding proficiency and familiarity with building similar applications. You will have **48 hours** to submit your finished application, starting from when you receive this project description.

Project Description : For this assessment, you will be implementing a small web application for looking up and submitting zipcode information using CSV files. This zip-code information will be stored in two separate databases that your application will need to interact with.

Database 1: Zip Code Products

The first data set contains information on which internet products are used in each zip-code. Associated with this information is the date that it was recorded, the original user who entered the data, and the user who most recently modified the data.

This database will be initially populated with the contents of the **product_master.csv** file.

Database 2: Zip Code Populations

The second data set contains information on the 5-mile population of each zip code. Again, this information is associated with the date it was recorded, the original user who entered the data, and the user who most recently modified the data.

This database will be initially populated with the contents of the **population_master.csv** file.

Application Requirements

For each database, your application must be capable of the following:

Zip-code lookup via CSV.

1. The user must be able to upload a CSV consisting of a single column of 5-digit zip codes (see **lookup.csv** for an example).
2. The user can then download a CSV of containing the corresponding database information for each zip-code in their uploaded CSV (see **product_lookup_result.csv** and **population_lookup_result.csv** for an example).
   a. The result file entries must be in the same order as the uploaded file entries.
   b. When a zip-code does not have any existing data in the database, all data fields will be listed as ‘N/A’ in the results file.

Zip-code update via CSV

1. The user can upload a CSV file in the same format as the lookup results file (see **product_update.csv** and **population_update.csv** for an example).
2. The database is then updated using the contents of the file.
    a. If the modified user in a row has **not** been specified, then the row is **ignored**.
    b. If the modified user has been specified and the zip-code does not yet exist in the database, then the data for that zip code should be **added** to the database.
    c. If the modified user has been specified and the zip-code does exist, then the data for that zip-code should be **updated** in the database.
3. If any of the rows to be inserted or updated contains an 'N/A' value, then the data is not uploaded, and an error is displayed to the user.
    a. If a row's modified user is not specified, the contents of the rest of the row do not matter. It should be skipped regardless.
4. Once the data has finished uploading, display a success message to the user.

This functionality needs to be available for each database **individually**. In other words, do not combine the data form both databases into one output file. There needs to a be a lookup/update functionality for the products database, and a separate lookup/update functionality for the populations database. These two feature sets will be identical except for which database they are pulling the data from and how the data is stored in the CSVs.

Technical Specifications

The application must be implemented as a website and hosted in the cloud. Both databases must also be hosted in the cloud, separately from the main application.

When setting up the database and application hosting, please use free trials or small instances, as we are not responsible for any hosting costs you incur. If you need assistance setting up a new account or have concerns about hosting costs, please let us know.

Recommended Languages, Frameworks, and Services

The following are the primary frameworks in use by our active projects, so to evaluate your experience with our tech stack, we recommend using technologies from this list. You may also add any additional technologies from your toolbelt to demonstrate your skills and knowledge.

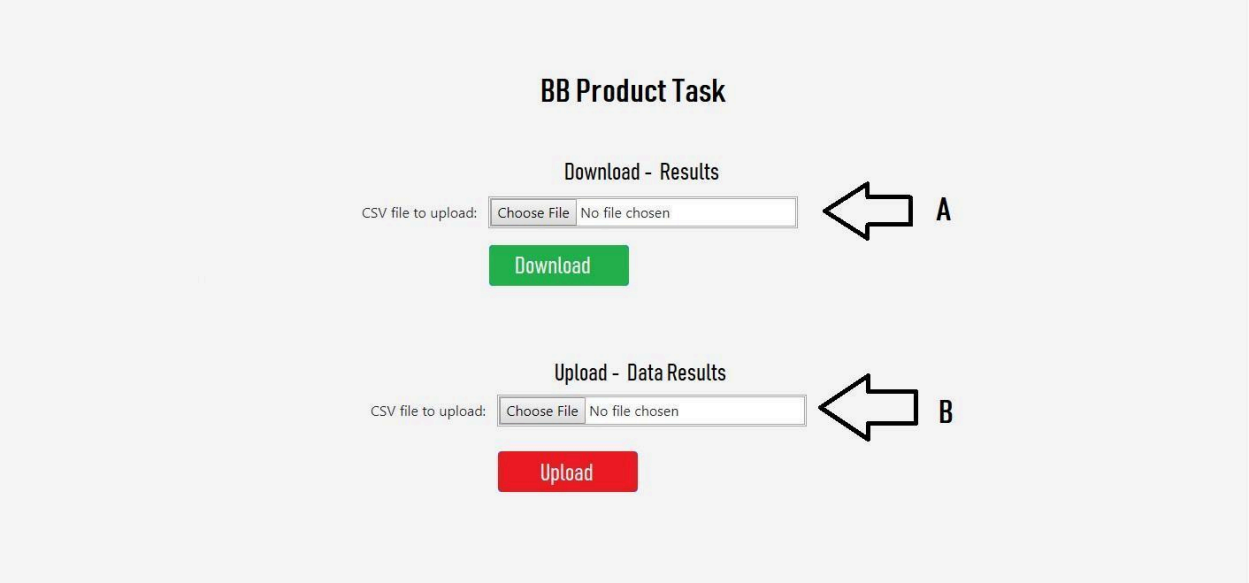
- Application
  - Python API
    - AWS Lambda
  - JavaScript Frontend
    - Angular
    - React
- Databases
  - MongoDB
- Hosting
  - Amazon Web Services (AWS)

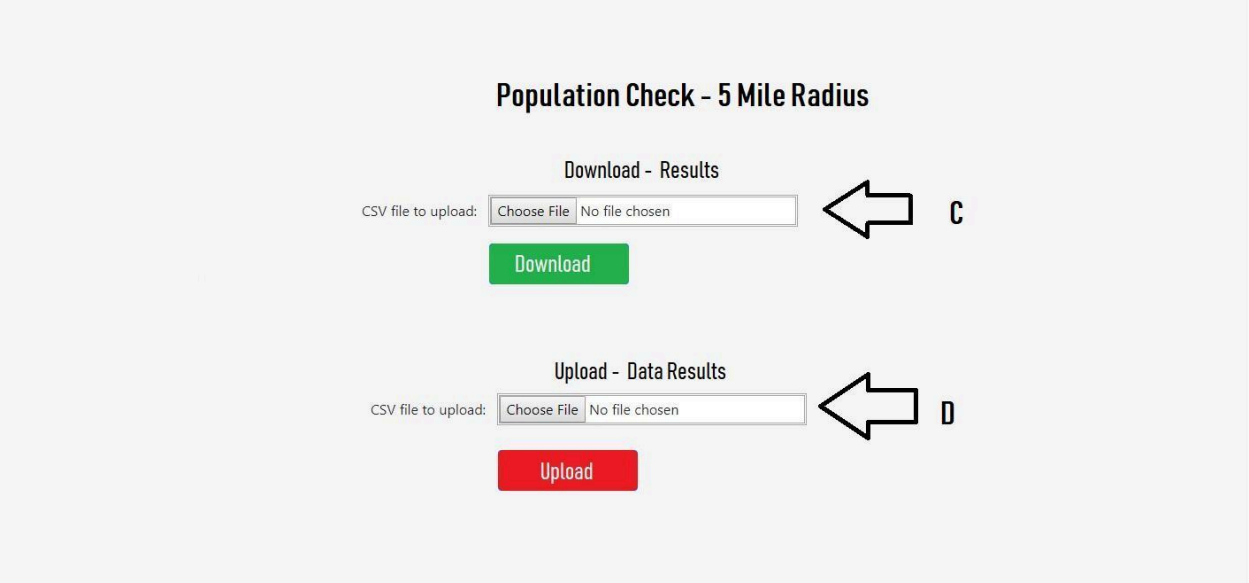
UI Design

With this project, we also wish to get sense for your visual design style. As such, you are encouraged to **get creative with the UI design** of the application (ready for an end-user client). You are welcomed to display error message on the front-end with more instructions to the end-user if you feel appropriate. Though keep in mind that all major functionalities from the *Application Requirements* section must still be present.

An example of the most basic UI design is shown below. Major functionalities are listed as A, B, C, and D, defined as follows:

- **A -** Zip-code **lookup** via CSV on the **Product** database
- **B -** Zip-code **update** via CSV on the **Product** database
- **C -** Zip-code **lookup** via CSV on the **Population** database
- **D -** Zip-code **update** via CSV on the **Population** database





Testing

Please test your application thoroughly before submission to avoid any issues that might hurt from your evaluation.

While testing, you can assume that all files will be in .csv format and be consistent with the structure of the sample files. In other words, there is no need to validate the CSV to check for incorrect values.

Evaluation Criteria

As stated before, we are asking you to complete this project to test your technical knowledge, skills, and performance in a simulated work environment. As such, we are looking for:

- A robust application free of bugs.
- Adherence to the application and technical requirements.
- Understandable, well-structured code.
- A clear understanding of database setup and use.
- An appealing and professional UI.
- On-time delivery.