***NAME: ALAGO CHIEMELA VICTOR***

***EPITA INTERNATIONAL BACHELORS***

***FIRST SEMESTER PROJECT DOCUMENTATION***

***PROBLEM TO SOLVE***:

For this project, I was tasked to create a static web application which generates web pages that contain required details from a database. This means that the information contained in each page of the web application shows only the information the user required. This is based on our database that was used in the database exam.

***MY APPROACH***:

There are different methods that can be used to solve the problem, but after much deliberation I decided to use **Flask**. Flask is a widely used micro web framework for creating APIs in Python. It is a simple yet powerful web framework that is designed to get started quickly and easily, with the ability to scale up to complex applications.

I decided to use Flask because it kept the web application more organized and easier to read. Using Flask helped me to have much fewer files, therefore I don’t have to write web pages and query files over and over. Flask also made it easier to link all the components of my web application and kept everything in one place, while automatically creating the server to run the web application.

The method of organization I used for this project begins with the parent folder called **solution files**. Inside the solutions file, there are two python files, **app.py** and **db.py**, and two folders, **templates** and **static**. It also contains a **\_pycache\_** folder which is automatically generated by flask.

The templates folder contains three web pages required for the project: **index.html, populations.html** and **grades.html**. These are the three pages we need, although the information displayed by these pages depend on the selected parameter from the database.

The static folder contains the CSS files that I used to make the web pages look more user-friendly and interesting. Nobody wants plain and ugly pages in front of their screens. These files are **style.css** and **style2.css**.

For easy comprehension, I have made a table to show the parent folder, files and subfolders as they are ordered. Well, because the order is important.

* **Solution files**
  + **app.py**
  + **db.py**
  + **templates folder**
    - **index.html**
    - **populations.html**
    - **grades.html**
  + **static folder**
    - **style.css**
    - **style2.css**
  + **\_pycache\_ folder (not important; always generated)**

***db.py***:

First, I connected the database to Flask, this is found in **db.py**. I used **jaydebeapi** to connect the database to our python workspace. All the queries used to extract the data from the database were also stored and executed in this file. In order to get the required information for each population, I used string formatting so that the required information for a particular population is generated. For the string formatting, I passed an argument for the function and this argument is **program**. Passing an argument is standard for string formatting in python. This argument, program, would also be passed in the app.py functions that generate the pages. This is necessary to allow for looping of the returned data in the html files.

***app.py***:

The app.py file holds all other files and components together. It is the file that generates the static web application. The website files and the database are connected here. In this file I imported Flask and **render\_template**. Render\_template is included in Flask but it is necessary to import it so I can return the websites I designed else I would have to write the codes for the websites inside the app.py file(not recommended).

In the flask file, that is app.py, each **@app.route** and the following **def functions** signifies a new page in the web application. Under each function, I connected the database and the web page (you will notice that I imported the db.py file). This is where the order I arranged the files gets important. For the html files, they must be stored in a folder called templates while the CSS files must be in the static folder. This is necessary because Flask recognizes the templates and static folders as where the html and CSS files are and hence easily connects to them to enhance the look of the web page it generates.

In the def, I also had to pass the argument(program) as it is in the database query so that the changes in the data generated would also determine the page and title displayed. All the database required for each page is also passed here. This means, I returned the queries that are required for the page and didn’t have to return the ones that are not needed.

***templates and static***:

The templates folder contains the three html files that contain the base layout for each page. These files contain just the structure of the page but don’t contain any data. To get the required data, I looped the data from the database which will be returned depending on the argument passed, in this case “program” which takes the value of the population code from the database. The html files also contain some Javascript which I used to generate the last update time of the pages and for the charts as seen in the index page.

The static folder contains the CSS files and the images I used for the website. This is only needed to beautify the pages and avoid writing all the styles inside the html.

**Remember**: These folders must be named templates(for html files) and static(for CSS files and images). That’s the name required by flask to allow it to access the files easily.

**TO RUN THE WEB APPLICATION**:

To start the web application, simply open the solution files folder containing all the files in your editor say VS-Code or PyCharm, then run the app.py file. After running the app.py file, the application starts and creates the server for the static web pages to run. The link to this server is shown in the terminal which you copy or click to see the web application in its glory.

I have also demonstrated how to do this in the video I attached in the site zip folder.