CSC 103 - HW 8

Complete each of the following exercises. To receive full credit please ensure the following:

- Submission structure follows the following:

```
HW8
    -> data
           -> poke.csv
           . . . .
    -> school.py
```

- Exercise artifacts are uploaded to your GitHub repository under a folder called `HW8` (i.e. https://github.com/<username>/carlow-me)
- Exercise artifacts are uploaded to Brightspace as a compressed `.zip` file called `<LastName>_HW8.zip`.

It is crucial that you are naming artifacts and storing files in the format mentioned above. All files that need to reference datasets should use a relative reference that refers to it's location in the `data` directory and not an absolute reference that would only work on your machine. Failure to meet any of these requirements will result in loss of points for that and any following problems.

Exercises

This week's exercise is a bit different. This week we are going to be building a collective application (API) that needs to meet a series of criteria. So what you'll need to do is provide the appropriate code in `school.py` that accomplishes those tasks and stores or reads data for any of those tasks in the `data` directory.

1. Create an endpoint called `student` that accepts GET, POST, and DELETE actions
2. Create an endpoint called `teacher` that accepts GET, POST, and DELETE actions
3. Create an endpoint called `class` that accepts GET, POST, and DELETE actions

For any future actions that ask to store data, we should create CSV datasets that store the results of these actions. The student dataset should be stored in `student.csv`, the teacher dataset in `teacher.csv` and class dataset in `class.csv`.

Teachers should have the following attributes: ID, Name, Email, Phone
Students should have the following attributes: ID, Name, Email, Phone, Year, Status
Class should have the following attributes: ID, Name, Department, TeacherID

4. Provide the ability to create a teacher via POST with data for the above fields.
5. Provide the ability to create a student via POST with data for the above fields.
6. Provide the ability to create a class via POST with data for the above fields. All fields except for the Department are required. If Department is not provided, it should default to the value of Misc.
7. For all required fields, if a field is not provided the response should return a `HTTP 500` error with the message `{'error': '<field name> is required'}' where `<field name>` is replaced with the field that is missing
8. When any of the above POST actions are taken, the appropriate values should be written to the appropriate file.

When deleting any of the above entries, we should allow this to be done by passing the ID to the appropriate endpoint (i.e. `{'id':1234}`).

9. Provide the ability to delete a teacher via DELETE by passing the ID via data
10. Provide the ability to delete a student via DELETE by passing the ID via data
11. Provide the ability to delete a class via DELETE by passing the ID via query param.

When attempting to get information from any of the endpoints, we should be able to retrieve this data by providing the ID for that component via a url parameter. Responses for each respective object should return all available fields for that object.

12. Provide the ability to get a teacher information via GET via ID
13. Provide the ability to get a student information via GET via ID
14. Provide the ability to get class information via GET via ID

15. Provide the ability to make a GET request to the class endpoint with a query parameter of `count=true` . If this parameter is present, it should return data that looks like the following: `{'count': <number>}` where number represents the number of classes that are present.
16. Provide the ability to make a GET request to the teacher endpoint with that teachers ID as a url parameter. If the query parameter `count=true` is present, it should return the number of classes that teacher teaches as part of the response. That is to say, the usual data should be returned with `count` as an additional field represented in a way similar to Q15.