ACIT 2515 – Object Oriented Programming - Lab 7 (Friday Set)
RESTful API with JSON

Instructor	Mike Mulder (mmulder10@bcit.ca) Also available on Slack.
Total Marks	25
Due Date	Thursday, March 7th by midnight on D2L

Goals

- Gain practice implementing RESTful web APIs using JSON in request and response messages.
- Use the Flask Python library for building RESTful APIs.

Instructions

Using versions of the Course and School classes from Lab 2 as a starting point (included in the Lab 7 zipfile), build a set of RESTful APIs that allows Course objects to be added, updated, retrieved and deleted from a School object.

The following JSON should be used to represent a Course object in the API:

```
{
     “course_id”: “ACIT2515”,
     “crn”: “12345”,
     “program”: “CIT”,
     “students”: [
          “A00000012”,
          “A00000123”,
          “A00001233”
          ]
}
```

The following JSON should be used to represent a School object in the API:

```
{
     “school_name”: “BCIT”,
     “num_courses”: 11,
     “programs”: [
          “CIT”,
          “CST”
     ]
}
```

Note that num_courses is the total number of courses in the School object and programs is the list of unique programs based on the programs associated with each of the Courses in the School.

The following are the APIs that need to be built for this lab.

Endpoint: POST /school/courses
Description: Adds a new Course to the School (if a course with the same course_id doesn't already exist)
Request: Course JSON (as above)
Response: None except for errors
Response Codes:

- 200 OK
- 400 Bad Request with response message "Course is invalid" if the JSON is invalid or missing any of the required elements (course_id, crn, program or students)
- 400 Bad Request with the response message "Course exists" if the course_id in the JSON already exists in the School

Endpoint: PUT /school/courses/<course_id>
Description: Updates an existing Course in the School (based on course_id)
Request: Course JSON (as above) but without the course_id
Response: None except for errors
Response Codes:

- 200 OK
- 400 Bad Request if the course_id is invalid with response message "Course ID is invalid"
- 400 Bad Request with response message "Course is invalid" if the JSON is invalid or missing any of the required elements (course_id, crn, program or students)
- 404 Not Found if an existing Course with the same course_id does not exist

Endpoint: DELETE /school/courses/<course_id>
Description: Deletes a Course in the School
Request: None
Response: None except for errors
Response Codes:

- 200 OK
- 400 Bad Request if the course_id is invalid with response message "Course ID is invalid"
- 404 Not Found if an existing Course with the same course_id does not exist

Endpoint: GET /school/courses/<course_id>
Description: Gets a Course in the School (based on course_id)
Request: None
Response: Course JSON (as above) for the specified course
Response Codes:

- 200 OK

400 Bad Request if the course_id is invalid with response message “Course ID is invalid”
404 Not Found if an existing Course with the same course_id does not exist

Endpoint: GET /school/courses/all
Description: Gets all Courses in the School
Request: None
Response: List of Course JSON objects OR an empty list if there are no courses
Response Codes:
200 OK

Endpoint: GET /school/courses/program/<program_name>
Description: Gets all Courses in a specific Program in the School
Request: None
Response: List of Course JSON objects in the program OR an empty list if there are no courses
Response Codes:
200 OK
404 Not Found if there are no Courses with the specified program_name

Endpoint: GET /school
Description: Gets the School details
Request: None
Response: School JSON (as above)
Response Codes:
200 OK

Key Steps:

- Make sure you add the Flask package to your PyCharm project (or using pip)
- Create your Flask based RESTful API in a file called **school_api.py**
  - Create a single instance of a School object in this module with the name of the school set to BCIT.
- Add a method “to_dict()” to the Course class (in course.py) to return a Python dictionary representation of the data held in the course. The dictionary can be converted to JSON for the API response messages.
- You will have to update the School class (in school.py) to support some of the validation and API endpoints. Make sure you update the School unit test for any new or modified methods!
- Take screenshots of each of your API endpoints being exercised in Postman (one per 200 OK response).

Make sure you test each of your API endpoints for both success and error cases. Marks will be subtracted if they do not meet the specifications above.

Grading Summary

API Implementation: • Add Course (3 marks) • Update Course (3 marks) • Delete Course (3 marks) • Get Course (3 marks) • Get All Courses (3 marks) • Get Courses in Program (3 marks) • Get School (3 marks)	21 marks
Unit Test for School Updated and Passes	2 marks
Screenshots • 1 Per API (4 marks)	2 marks
Marks will be subtracted for violations of Python and OOP best practices	-1 mark each
Total	**25 marks**

Submission

Upload the following files to the Lab 7 dropbox on D2L as a **zipfile** called **lab7.zip**:

- course.py
- school.py
- test_school.py
- school_api.py
- Screenshots of each API Endpoint exercised in Postman (success only)