Lab: Flask and APIs

In this assignment, we are going to expand on the previous lab and build an API to allow us to interact with the school and students.

Improve the models for web usage

Create the `add_grade` method on the `Student` class

The `add_grade` method:

- takes a grade as argument
- the grade can be an integer or a "string similar to an integer"
- the grade must be between 0 and 100
- otherwise, raise a `ValueError`

If the grade provided is valid, `add_grade` adds it to the list of grades for that student.

Create the `add_student` method on the `School` class

The `add_student` method:

- takes two arguments: the name and student ID of the student
- the arguments must be non-empty strings - otherwise, raise a `ValueError`
- the method creates the student and adds it to the collection of students in the school
- if a student with that student ID already exists, return `False`
- otherwise, return `True`

You can use the updated test files to check your models.

Create the API endpoints

Your application will now have additional HTTP endpoints:

- one to create a new student
- one to add a grade to a student

Endpoint "create a student"

Create a new Flask view for the route `/student/add`. Make sure this view only accepts `POST` requests.

The view expects data passed in as JSON. The JSON must have the following keys:

- `name`: name of the student
- `student_id`: name of the student

If the data provided is valid, the endpoint will create the student, add it to the school and return HTTP status code 200. If the data provided is not valid, return a 400 HTTP error page. If a student with that student ID already exists, return a 409 HTTP error page.

Endpoint "add a grade"

Create a new Flask view for the route `/student/<STUDENT_ID>/grades/add`. This view should only accept `POST` requests. The student ID is available in the URL, and the view expects to receive a JSON payload similar to: `{"grade": 67}`. The grade provided must be between 0 and 100. Otherwise, return a 400 HTTP error page. If the student ID provided in the URL does not exist, return a 404 HTTP error page.

> Reuse the methods you created in the first part of the lab!

Submit all your files to D2L in a ZIP archive:

- Flask application
- `models` folder with the `school.py` and `student.py` files
- `templates` folder with all the required HTML files

> You can use the `check_app.py` script provided to check your Flask setup.