Intermediate Programming with Python

Final project

Due date is December 11 before 11:59 PM

Fall 2023

The purpose of this project is to provide hands-on experience in the design and development of a Flask application and to assess your understanding of class material including Python concepts, object-oriented programming, data serialization, RESTful APIs, Flask, Linting, Unit testing.

You can select your own topic for the final project - imagine that you are working for a client or for your own business to implement an API. Below are two example topics that can be picked for the final project. You can choose from these examples or come with your own topic. Carefully read the list of features that each project support. Your project will be graded based on your coverage of these features.

Project Topic Idea #1: Implement a SmartHome API that allows owners to run particular operations. The smart home can have various smart home devices such as thermostat, light bulbs, smart vacuum, etc. (all extend from the same parent class). The API should allow users to add/remove smart home devices, list these devices (as an HTML page), alter their information, communicate with them, etc. For example, the API should allow its users to ask: 'the temperature difference between the home and outside'. To satisfy this functionality, the API should check home's temperature from Thermostat and consume a public API that provides current temperature outside the home (i.e., providing the home address or allowing the user to provide a new address/zip code). The API saves the home devices to a file. Thus, if the server restarted, the home devices should be created by reading this file. Moreover, the API provides an endpoint for users to view their past activities such that this history should be able to share with the user whenever they want to.

Project Topic Idea #2: Implement a language learning flashcards that allows users to study and practice vocabulary flashcards. The API should allow users to create new flashcards, list all available flashcards, remove a flashcard, update a flashcard, test their knowledge by presenting a random flashcard when requested, saves a user's progress, etc. While creating a flashcard, a user can provide an English word (front side of the flashcard) and a target language as a parameter for learning (i.e., Spanish) or meaning of that word. The API should consume a public API to find the corresponding word in Spanish and automatically create the flashcard (e.g., the user does not provide the Spanish word). The user can ask API to its progress. The API should support different types of flashcards such as vocabulary flashcards, grammar flashcards, translation flashcards, pronunciation flashcards (all extend from the same parent class). The API should save all flashcards a file and when the server restarts, the API should read these flashcards back.

General Project Requirements and Grading Prolicy

Here you can find a list of features that each project should provide. You will need to provide evidence in your submission document which clearly demonstrate how your application covers these features.

1. **Flask Application:** You must implement your application using the **Flask** framework. Usage of any other framework will not be accepted, and you will get **0** from this project. [this is a prerequisite]
2. **Driver code for demonstration:** You do not need to implement a graphical user interface for this project. For satisfying the project requirements, you can demonstrate the capabilities of your API through a Jupyter Notebook. For example, your notebook should demonstrate how someone can consume your API and what it can do. Think it as if you are marketing your API by showcasing your API features. [15 points]
3. **Object-Oriented Model:** Design your application such that you leverage object-oriented development paradigms. You can for example leverage encapsulation, inheritance, or abstract classes. For example, in Project Idea #1, you can have multiple smart home devices that extend from a parent SmartDevice class instead of developing each class independently. [10 points]
4. **Data Persistence:** Your application should implement simple data serialization methods for data storage. During this course, we did not cover

any database implementation. However, we can still use other serialization techniques we have covered so far for data persistence, such as JSON or Pickle based data serialization. [15 points]

5. **RESTful API Endpoints:** Your application should support multiple API endpoints for users including at least two GET routes, one POST route, one DELETE route, and one PUT route. DELETE and PUT routes should be implemented using variable routes (e.g., passing an ID in URL). Do not implement a POST endpoint if it more suitable to implement as a GET endpoint. Your endpoints must use JSON data format for data transfer. One of the endpoints (one of the GET routes) should render an HTML page using Flask built-in templates. [40 points]
6. **Consuming publicly available APIs:** Your API should consume at least one publicly available API to support its functionality. For example, you can consume a weather API or Google Translate API. You can check this list of APIs to get inspired: https://github.com/public-apis/public-apis. [20 points]

Bonus Requirements

You can earn extra points towards your project grade by implementing these bonus requirements. The points you will earn beyond the full grade (100) will be added to your course grade. E.g., if you get 110 points from this project, 10 points will be added to your overall course grade after multiplied by project's grade percentage (e.g., 10x0.25 => +2.5% on your final grade).

7. **Static Code Analysis:** The final project submission should include the outputs of linting analysis for every Python file. You must use SonarLint for this purpose and submit your project without any warnings. [10 points]
8. **Unit Testing:** In one of your Python classes, you must demonstrate the use of Python's built-in unit testing package. Design your project such that you have a method that you can write a unit test for. Indicating that your project does not need any unit testing will not be accepted as a valid reason for satisfying this requirement [10 points]

Submission Guidelines

Like assignment submissions, you must submit your project to your GitHub repository and share the repository with your instructor. Make sure that you have created a private repository and included your instructor as a collaborator. Make double sure that you have committed and pushed your project code to your repository before deadline. No missing/late submissions will be accepted (I will not remind you to push your code if I see an empty project folder, you will get 0).

Your project code should satisfy all the project requirements highlighted above to get full grade.

In addition to the project code submission, you must submit a short project report in PDF format that includes the following items:

- The project's GitHub repository information.
- Up to 250 words abstract that summarizes the API features.
- If you chose to implement a different project idea, you must explain your idea in detail that highlights what should users expect from your API.
- List of API endpoints and very brief description of their functionality.
- Description of your data persistence decision – JSON or Pickle or something else.
- A simple UML diagram that demonstrates how you used object-oriented paradigms.
- Selected public API and how and why you used in your project.
- The output/screenshots of linting process [BONUS].
- The output/screenshots of unit test results [BONUS].

Submission Due Date – No Late Submission

Due date is December 11 before 11:59 PM. No late submissions will be accepted. Make sure you have a valid submission before the deadline. You can keep working until the end and make another submission/update your submission before the deadline.