CSCI 2081 Introduction to Software Development - Fall 2024
Homework 04 - Client / Server

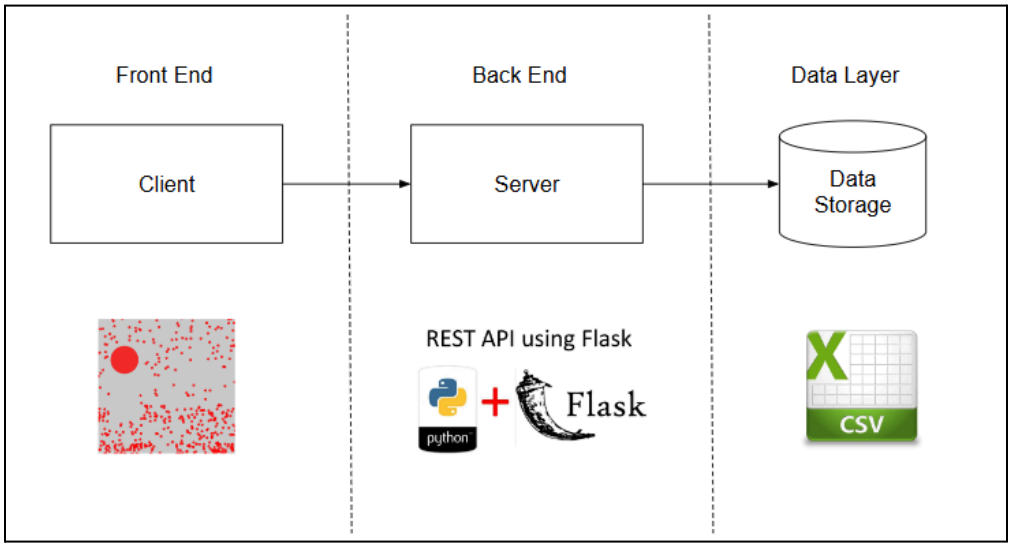

Figure 1 - Add a backend and data layer to your particle simulation.

Due Date: Friday, December 6, 2024 @ 11:59pm

Instructions: This homework assignment involves maintaining a client / server architecture. The particle simulation will act as a client for a server that stores obstacle information.

Getting Started: Before you begin this assignment, be sure to become familiar with the following course content:

- **Video Tutorial:** Lecture 28 - Web Services
- Writing CSV files in Python - GeeksforGeeks

Submission: Your submission is entirely on github (code, documentation, branches, pull requests). You should make sure all tasks in this assignment are completed.

- Tag as **hw4** to create a **Homework 4** release.

Task 1: Create a Python Flask REST Web Service (60 points)

Task: Create a flask application that runs a web service for saving obstacle information.

1. In your <x500>_hw repository add a **`server`** folder.

2. In the **`server`** folder, create a python flask application (e.g. review Lecture 28 - Web Services and see example api code for starter code.) You will create an application that modifies the following functions:

```
# load from CSV
obstacles = []

@app.route('/obstacle', methods=['POST'])
def add_obstacle():
    # add to the obstacles (request.json)
    # save obstacles to csv
    return "Success"

@app.route('/obstacle', methods=['POST'])
def delete_obstacle():
    # delete obstacle from obstacles (perhaps use an id)
    # save obstacles to csv
    return "Success"

@app.route('/obstacles', methods=['GET'])
def get_obstacles():
    return jsonify(obstacles)
```

   - **obstacles list** - Load the list from the CSV file when the server starts.
   - **add_obstacle()** stores an obstacle in obstacles list and saves in the CSV file.
   - **delete_obstacle()** deletes an obstacle from the obstacles list and saves in the CSV file.
   - **get_opstacles()** returns obstacles from obstacles list.

3. Review the following documentation on how to save and open CSV files using python:
   - Writing CSV files in Python - GeeksforGeeks

Rubric:

- 10 points - API Exists.
- 20 points - The two functions, add_obstacle() and delete_obstacle(), modify the obstacle list correctly.
- 20 points - The obstacles are saved to a CSV whenever the obstacle list is modified.
- 5 points - The function, get_obstacle(), returns the obstacle list correctly.
- 5 points - The obstacles are loaded from the CSV when the API starts.

Task 2: Java Client (40 points)

Task: Modify your particle simulation to get, add, and delete obstacles using a web service backend.

1. Create a class called ObstacleAPI that handles all the calls to the web service (add, remove, and get). Use Postman to generate the Java code and copy jars discussed in Lecture 28 - Web Services. See example and lecture 28 for how to do this.

2. When your particle simulation starts, load in the obstacles from the ObstacleAPI.getObstacles() and store them in the linked list.

3. Whenever you create a new obstacle, call the ObstacleAPI.add(Obstacle obs) function, which calls the python backend.

4. Whenever you delete an obstacle, call the ObstacleAPI.delete(Obstacle obs) function, which calls the python backend. (You may want to add an id to your obstacle to enable this).

Rubric:

- 20 points - ObstacleAPI calls the three functions described in Task 1 in the web service backend.
- 10 points - When the program starts, obstacles are loaded from the backend.
- 5 points - When you add an obstacle, it is added to the CSV using the web service.
- 5 points - When you delete an obstacle, it is removed from the CSV using the web service.

Submission: Your submission is entirely on github (code, documentation, branches, pull requests). You should make sure all tasks in this assignment are completed.

- Tag as **hw4** to create a **Homework 4** release.