CSC131 – Fall 2021 Kane

Due Dates: Monday Oct. 25th – Monday Nov. 1st

HW 04 REST APIs

**SETUP**

1. Read through the following Python and REST Article:  
   <https://realpython.com/api-integration-in-python/#rest-and-python-tools-of-the-trade>
2. Obtain **Python3**. Follow your OS’s specific instructions:  
   <https://www.python.org/downloads/>
   * Verify **Python** is installed correctly via the [command line](https://stackoverflow.com/questions/8917885/which-version-of-python-do-i-have-installed)
3. Obtain the Python Web Framework **Flask**:  
   <https://flask.palletsprojects.com/en/2.0.x/installation/>
   * Verify **Flask** is installed via [command line](https://stackoverflow.com/questions/5285858/determining-what-version-of-flask-is-installed/58947553)
4. Download the Sample Code: server.py
5. Confirm the skeleton works by running it
   * Test the server by running it however you installed Python3:  
     python server.py or python3 server.py or py server.py
6. Execute the test REST endpoint at /api/v1/test
   * Verify that you can execute a GET and POST using one of the below methods:
     1. GUI tool like [Postman](https://www.postman.com/) or [Insomnia](https://insomnia.rest/) (I like Insomnia)
     2. Use [curl](https://curl.se/) (If you’re on Windows, [Git Bash](https://git-scm.com/downloads) comes installed with curl)  
        **Read:**  
        curl -X GET http://localhost:8080/api/v1/test -H "Content-Type: application/json"  
        **Create:**  
        curl -X POST http://localhost:8080/api/v1/test -d “{\”name\”: \”Kane\”, \”email\”: \”gary.kane@csus.edu\”, \”address\”: \”123 Maple Street\”}” -H ”Content-Type: application/json”  
        **Update:**  
        curl -X PUT http://localhost:8080/api/v1/test/4 -d “{\”name\”: \”Kane-New\”, \”email\”: \”gary.kane@csus.edu\”, \”address\”: \”123 Maple Street\”}” -H ”Content-Type: application/json”  
        **Delete:***For your to implement!*
7. Now that you have the skeleton up and running, let’s build a REST API!

**CODE**

1. See below for the required data for this assignment. Build REST APIs to perform CRUD (create, read, update, and delete) operations on your data. Data can be implemented as a simple list of dictionaries as seen in the example.
2. A user can have 0 or more recipes
3. You should have **8 total** endpoints for CRUD operations for each of the data types.
4. Test your REST API using one of the tools mentioned in the SETUP
5. Document outputs for each endpoint. Run your REST API with some data values. Paste the REST end points **with responses** in a text file.
6. See SUBMISSION details

**DATA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Endpoint** | **Field** | | |
| user | /users/:user\_id | id | name | type |
| recipe | /users/:user\_id/recipes/recipe\_id | id | user\_id | steps |

**Example User:** {\”id\”: 1, \”name\”: \”Kane\”, \”type\”: \”admin\”}

**Example Recipes:** {\”id\”: 1, \”user\_id\”: 1, \”steps\”: \”bake cookies\”},  
{\”id\”: 2, \”user\_id\”: 1, \”steps\”: \”cook rice\”}

**GRADE**

1. 40 points for your code should have a GET, POST, PUT, and DELETE method for **each** of the 2 DATA types.
2. 30 points for correct error handling and easy-to-read code
3. 20 points for a good output.txt with data examples for each endpoint

**SUBMISSION**

1. Submit your server.py to Canvas with updated code for the new endpoints
2. Submit output.txt where you execute each of the 8 endpoints with data. Paste each of the endpoints and their responses. Make this file easy to read so it’s simple to understand your REST APIs.