Current version: **02/02/25**

Lab 03: Web Services & Restful APIs

Release Date: 04th February, 2025(Tuesday Section) & 6th February, 2025(Thursday Section)

Demo and Submission Date: 11th February, 2025(Tuesday Section) & 13th February, 2025(Thursday Section)

You can work in teams of 2 or choose to work solo.

Files to submit via Vocareum are below. **Students in teams must submit on Vocareum once**, together as a team:

- mail_server.py
- mail_client.py
- weather.py
- README.md (or .txt)

1. Introduction

In this lab, you'll practice creating and using RESTful API services. This is a nice article by AWS on RESTful APIs. You will need to read through it to answer the lab questions below.

Question 1: Why are RESTful APIs scalable?

Code

Download codes to your VM, first part until REST should be done on the VM.

Creating a Client/Server

First, you'll need some python packages that don't come with the standard library. These requirements are listed in the requirements.txt file in the repository. To install them, you can run the commands in Section 3, Section 7.

Check out this article for more detailed information on pip and the requirements.txt file!

In the first part of the lab, you'll create a basic email server! Open mail_server.py and read through the comments/documentation carefully to understand what the application is

doing. Then, implement the TODOs.

Many of the TODOs ask you to fill out the “docstrings” of the methods. These strings below the method definition describe what the method does, its arguments, what it returns, and sometimes more! Please be descriptive! See these examples of good docstrings for reference.

Question 2: According to the definition of “resources” provided in the AWS article above, What are the resources the mail server is providing to clients?

Question 3: What is one common REST Method *not* used in our mail server? How could we extend our mail server to use this method?

Question 4: Why are API keys used for many RESTful APIs? What purpose do they serve? Make sure to cite any online resources you use to answer this question!.

2. Part 1: Flask

Flask is a compelling web application framework for Python. It's impossible to cover the entire framework in just one lab session, so we will expose you to some of its features through a learn-by-example approach. In particular, you will complete some TODOs in the starter code to complete the mail service application.

Here are some excellent references to learn more about Flask:
https://pythonspot.com/flask-web-app-with-python/

https://www.tutorialspoint.com/flask

The official tutorial is more advanced, but you can find it here if you dare:
http://flask.pocoo.org/docs/1.0/tutorial/

3. Setup

To install the necessary packages on VM, run the following command:

```
pip3 install flask [IF THIS DOESN’T WORK DUE TO VM REASONS, try the second one]

pip3 install flask --break-system-packages
```

You'll need Flask on your VM to complete this lab

4. Test on Raspberry Pi (1 point bonus)

Figure out how to run the server side of your HTTP application on the Raspberry Pi, with the client side running on your VM. Show this during your demo for extra credit.

PART 2

5. Introduction

The purpose of this lab is to expose you to using RESTful APIs to access the wealth of information available on the Internet. The possibilities are endless!

6. What is REST?

REST stands for **RE**presentational **S**tate **T**ransfer. It's all about getting or updating resources, and transferring that information between a client and server. An example of a RESTful API is a phone book service. The resource in this case would be a person and the data could include things like name, address, and phone number.

The REST architectural style involves six constraints:

- uniform interface - standardized protocol for communication (i.e. HTTP)
- stateless - each request is self-contained (no client state maintained at server)
- client-server communication model
- cacheable
- layered

RESTful APIs use the HTTP protocol and resources are identified by URIs or uniform resource identifiers. Information is often sent using common formats such as JSON or XML. For this lab, we'll be using JSON.

7. Setup

To install the necessary packages on Raspbian and Ubuntu, run the following command:

```
pip3 install requests --upgrade
```

While we really only need it on the RPi for this lab, you should install it on your VM as well for testing purposes and for use in future labs. It will be easier to test your "apps" on your VM before deploying them on your RPi.`

8. Using APIs

As mentioned earlier, HTTP is the common protocol used for RESTful APIs. HTTP supports a variety of methods, e.g. GET, PUT, POST, DELETE. However, in this lab we'll be focusing on the GET method. Python provides the requests library which handles a lot of the work for us. In the starter code, we've provided you with a file, namely *my_weather.py*. You'll need to modify this file to make it fully functional.

Requests Documentation: https://requests.readthedocs.io/en/stable/
Response status codes are included as part of the HTTP standard. Here are some good ones to know:

Status Code	String	Description
200	OK	the request has succeeded
400	Bad Request	the server cannot or will not process the request due to something that is perceived to be a client error
401	Unauthorized	the request has not been applied because it lacks valid authentication credentials for the target resource
403	Forbidden	indicates that the server understood the request but refuses to authorize it
404	Not Found	indicates that the server can't find the requested resource

Table 1. HTTP Status Codes

Creativity is allowing yourself to make mistakes. Art is knowing which ones to keep. Scott Adams

9. Using a "Real" API

STEP 1

- **Get a WeatherAPI Account: Sign up at WeatherAPI and get your API key.**
- **After signing up, you will be provided with an API key. This key is essential for making API requests to fetch weather data.**

STEP 2

Setting Up Your Python Script

1. **Create a Python file, for example, weather.py, this file is also present in your drive.**
2. **Import the necessary modules (requests and json) and define the base API URL.**

STEP 3: Understanding the Code

1. **API Key: Store your API key in the WEATHER_API_KEY variable. Make sure you replace 'your_api_key_here' with the actual key.**
2. **API Request: We use the requests.get() function to make a GET request to WeatherAPI's endpoint for current weather data.**
3. **Response Handling: The response is checked for a 200 OK status code, which indicates a successful request. If successful, the script parses the JSON response and prints out the temperature, weather condition, and humidity.**
4. **User Input: The city name is taken as input from the user to fetch weather information for a specific city.**

STEP 4: RUN THE SCRIPT

python weather.py

When prompted, enter a city name (e.g., Los Angeles):

Output: Temperature in Los Angeles: 72.5°F, Condition: Clear, Humidity: 55%

```
(IOBT) tamoghna@tsarkar:~/Desktop/apitest$ python app2.py
Enter city name: kolkata
Status 200: OK
Weather in kolkata:
Temperature: 84.6°F (Feels like: 95.5°F)
Condition: Mist
Humidity: 89%
Wind: 4.9 mph, Direction: SE
Pressure: 1006.0 mb
UV Index: 7.0
Cloud Cover: 50%
Visibility: 1.0 miles
```

Bonus: for one extra point, implement your own code connecting to another production API! Some ideas:

- https://official-joke-api.appspot.com/random_joke: A random joke API
- http://apiv3.iucnredlist.org/api/v3/docs: IUCN Red List of Threatened Species
- https://wordcloudapi.com/: Easily create word clouds
- https://www.blockchain.com/explorer/api: Bitcoin Payment, Wallet & Transaction Data

Find more public APIs here: https://github.com/public-apis/public-apis

10. References & Tutorials

https://www.youtube.com/watch?v=MwZwr5Tvyxo

https://pythonspot.com/flask-web-app-with-python/

https://www.tutorialspoint.com/flask

https://docs.python.org/3.5/library/pickle.html

http://docs.python-requests.org/en/master/

Demo and Code Grading Rubric

Points	**Description**
Demo	
2	Client can send mail
2	Client can get received mail
2	Client can list all mail for sender/receiver
2	Client can delete mail
Code	Each Team must submit files to Vocareum
2	Team member names listed in Readme.
6	Code correctness (no python syntax errors/code compiles, sufficiently bug-free for the assignment, etc.)
1+1 (bonus)	**Bonus only applicable when both requirements are met -** Run VM as client and RPi as server + (bonus) Showcase an additional

	call to another production API.
Reflection	Answer Lab Questions in Readme.md
1	Question 1
1	Question 2
1	Question 3
1	Question 4
	Total: 20