RestAPI

Public API

What is it?

A **Public API** (Application Programming Interface) is an open-access interface that allows users to interact with data over the internet. Examples include APIs for weather data, cryptocurrency prices, or random user information.

Why do we use it?

- Provides **real-time data**.
- Useful for **testing, learning, and development**.
- No authentication is required in many cases.

Example Public APIs:

- Random User API: https://randomuser.me/api/
- JSONPlaceholder (Fake Data): https://jsonplaceholder.typicode.com/posts
- OpenWeatherMap (Weather Data): https://api.openweathermap.org/data/2.5/weather

What is a REST API?

REST (Representational State Transfer) is an **architecture** that allows systems to communicate over HTTP. A **REST API** provides access to resources (like data) using **HTTP methods** such as:

- **GET** → Retrieve data
- **POST** → Send data
- **PUT** → Update data
- **DELETE** → Remove data

JSON (JavaScript Object Notation)

What is it?

JSON is a lightweight data-interchange format, mainly used for **transmitting data between a server and a web application**.

Why do we use it?

- **Easy to read and write**.
- **Compatible with multiple programming languages**.
- **Lightweight and fast for data exchange**.

Pytest

What is pytest?

pytest is a Python testing framework that allows writing **simple, scalable test cases**.

Why do we use pytest?

- **Easy to use** with minimal setup.
- Supports **assertions for testing**.
- Generates **detailed test reports**.

Given assignment involves the following steps:

1. **Fetching data from a Public API** and storing the response in JSON format.
2. **Saving the response in a new file**.
3. **Writing test cases using pytest** to validate the JSON data.

Solution:

📌 API Endpoint: https://dog.ceo/api/breeds/image/random

Step 1: Install Required Python Modules

pip install requests pytest

requests: Helps us **fetch data from an API**.

pytest: Used for **testing** our results.

Step 2: Fetch Data from the Public API and Save It

📌 Create a Python file called **fetch_data.py**

python Copy Edit

```python
import requests
import json

# API URL for random dog images
API_URL = "https://dog.ceo/api/breeds/image/random"

# Fetch data from the API
response = requests.get(API_URL)

# Check if the request was successful
if response.status_code == 200:
    data = response.json()  # Convert response to JSON format

    # Save JSON data into a file
    with open("dog_image.json", "w") as file:
        json.dump(data, file, indent=4)

    print("✅ JSON data saved successfully!")
else:
    print("❌ Failed to fetch data. Status Code:", response.status_code)
```

Get data from API → requests.get(API_URL).

Check if request is successful → if response.status_code == 200.

Convert API response to JSON → response.json().

Save the JSON in a file → json.dump(data, file, indent=4).

✅ After running this script (python fetch_data.py), a new file dog_image.json is created with content like this:

```json
{
    "message": "https://images.dog.ceo/breeds/labrador/n02100735_400.jpg",
    "status": "success"
}
```

Step 3: Write Test Cases using Pytest

📌 Create a Python file called **test_api.py**

```python
import json

# Load the saved JSON file
with open("dog_image.json", "r") as file:
    data = json.load(file)

# Test if JSON contains expected keys
def test_json_keys():
    assert "message" in data  # Check if 'message' key exists
    assert "status" in data   # Check if 'status' key exists

# Test if 'message' contains a URL (dog image link)
def test_message_is_url():
    assert data["message"].startswith("https://")  # Must start with 'https://'

# Test if 'status' is 'success'
def test_status():
    assert data["status"] == "success"
```

Load the JSON file → json.load(file).

Test if the correct keys exist → "message" and "status".

Test if the dog image URL is valid → It should start with "https://" .

Test if API response is correct → "status" should be "success".

Step 4: Run the Tests

To check if our tests work correctly, run this command:

```
pytest test_api.py
```

If everything is correct, you will see this output:

```
============================= test session starts ==============================

collected 3 items

test_api.py ...                                                      [100%]

============================== 3 passed in 0.12s ===============================
```