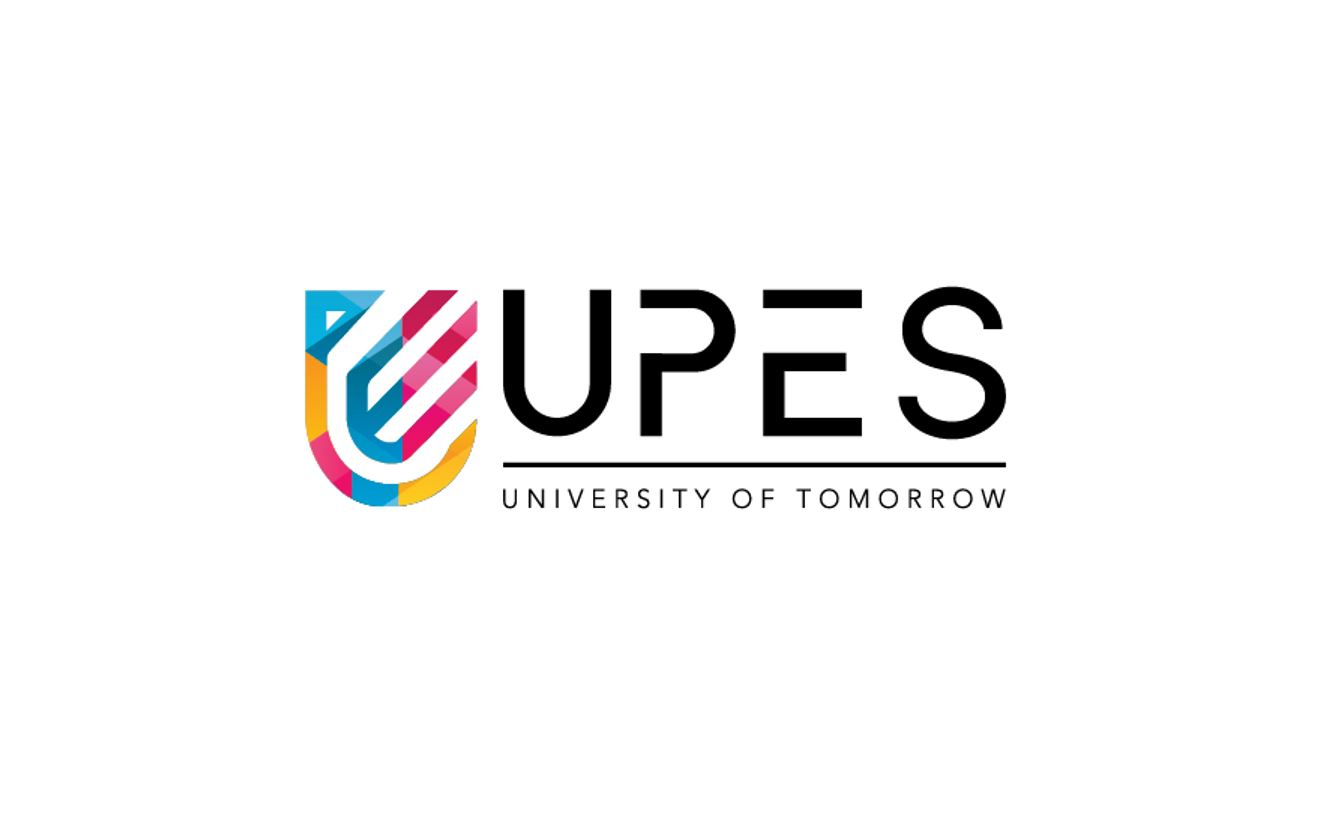
****

**DEVOPS**

**Automated Testing Assignment**

**Submitted By-                                          Submitted To -**

Priyanshi                                                  Dr. Mitali Chugh

SAP ID – 500107299

Roll no - R2142220603

Batch – B7 (CCVT NH)

**Step 1: Install Python and Pip**

First, ensure that **Python** and **pip** are installed.

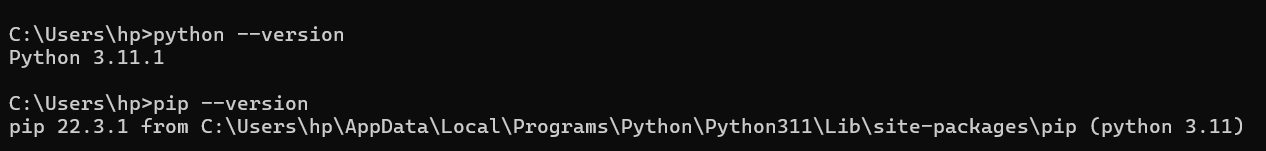
1. Open **Command Prompt (CMD)** and check if Python is installed:

python –version

If Python is not installed, download it from [Python's official website](https://www.python.org/downloads/) and install it.  
✅ Make sure to **check the "Add Python to PATH" option** during installation.

1. Verify if **pip** (Python's package manager) is installed:

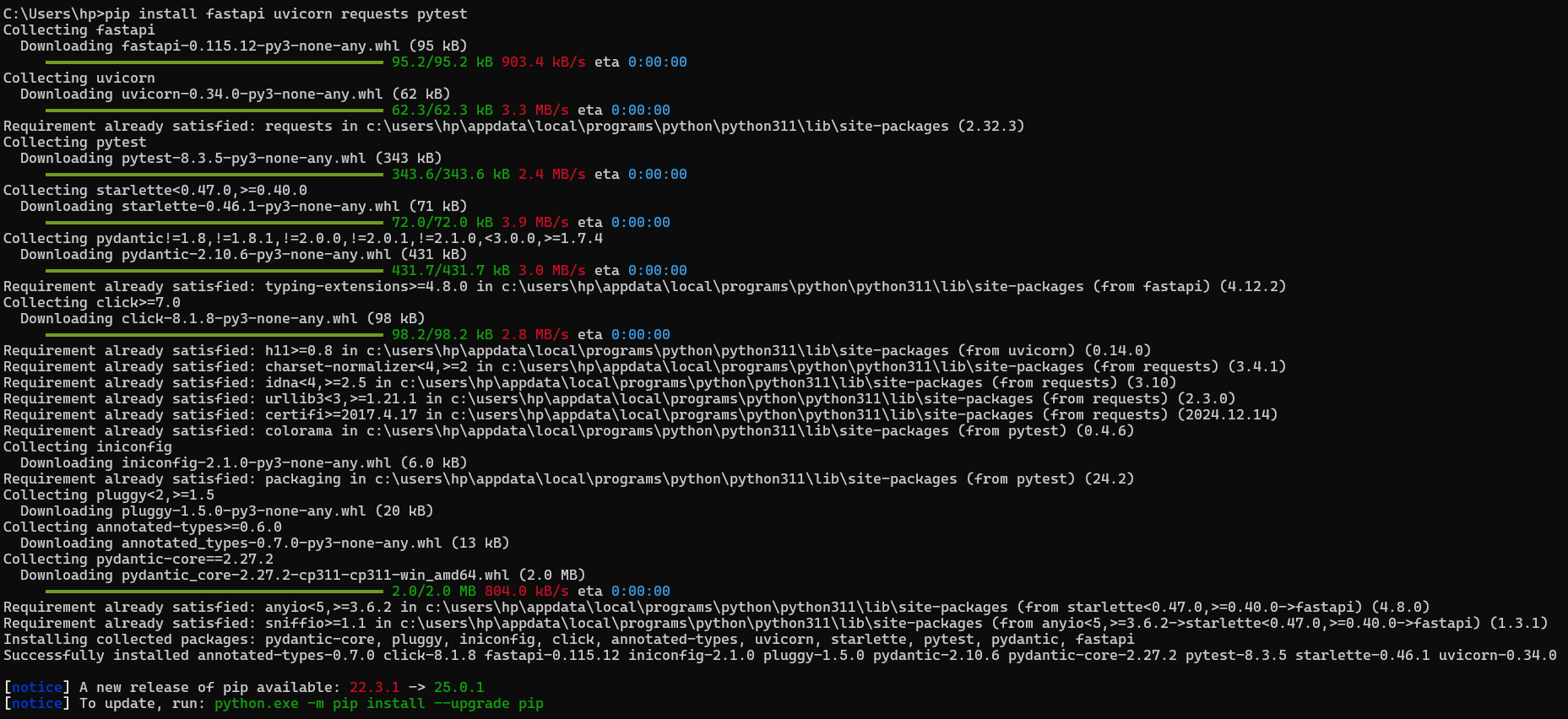
pip –version



**✅ Step 2: Install FastAPI and Required Libraries**

Run the following command in **CMD or PowerShell** to install FastAPI, Uvicorn (server), and Requests (for API testing):

pip install fastapi uvicorn requests pytest



**✅ Step 3: Create the API Server (FastAPI)**

1. **Open Notepad, VS Code, or any text editor** and create a new file:  
   **Save it as** apiserver.py
2. **Copy and paste the following FastAPI code:**

from fastapi import FastAPI

app = FastAPI()

@app.get("/")

def home():

return {"message": "Welcome to FastAPI Calculator API!"}

@app.get("/add/{num1}/{num2}")

def add(num1: int, num2: int):

return {"result": num1 + num2}

@app.get("/subtract/{num1}/{num2}")

def subtract(num1: int, num2: int):

return {"result": num1 - num2}

@app.get("/multiply/{num1}/{num2}")

def multiply(num1: int, num2: int):

return {"result": num1 \* num2}

if \_\_name\_\_ == "\_\_main\_\_":

import uvicorn

uvicorn.run("apiserver:app", host="127.0.0.1", port=8000, reload=True)

**✅ Step 4: Run the API Server**

1. Open **CMD** and navigate to the folder where apiserver.py is saved:

cd path\to\your\file

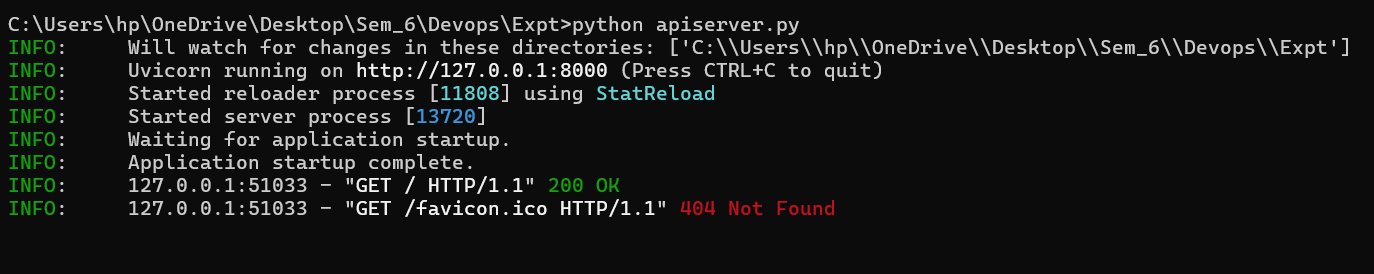
*(Replace path\to\your\file with the actual folder path where your Python file is stored.)*

1. Start the API server using:

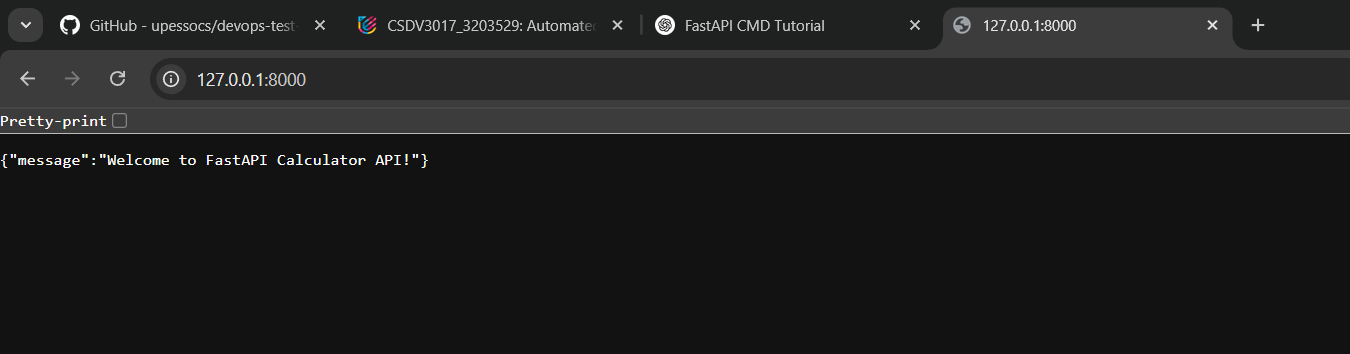
python apiserver.py

OR

uvicorn apiserver:app --host 127.0.0.1 --port 8000 –reload



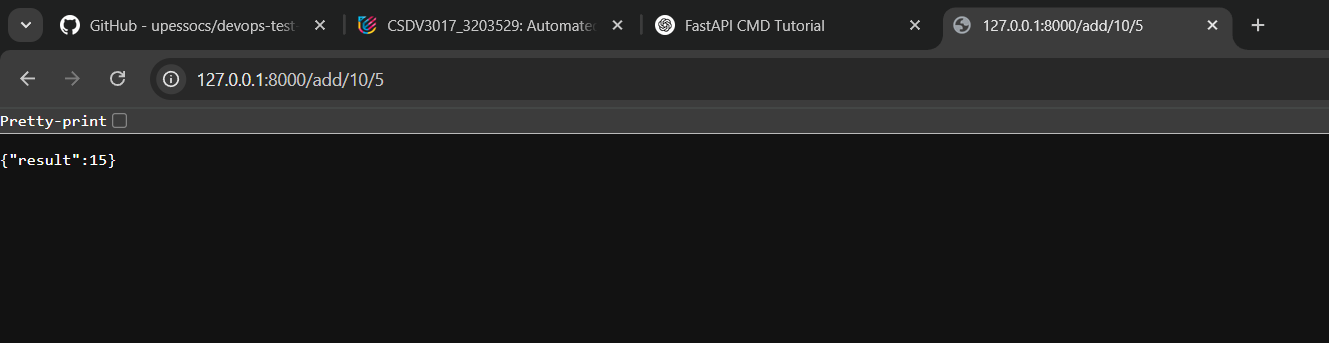
✅ **Now your API is running on** <http://127.0.0.1:8000>



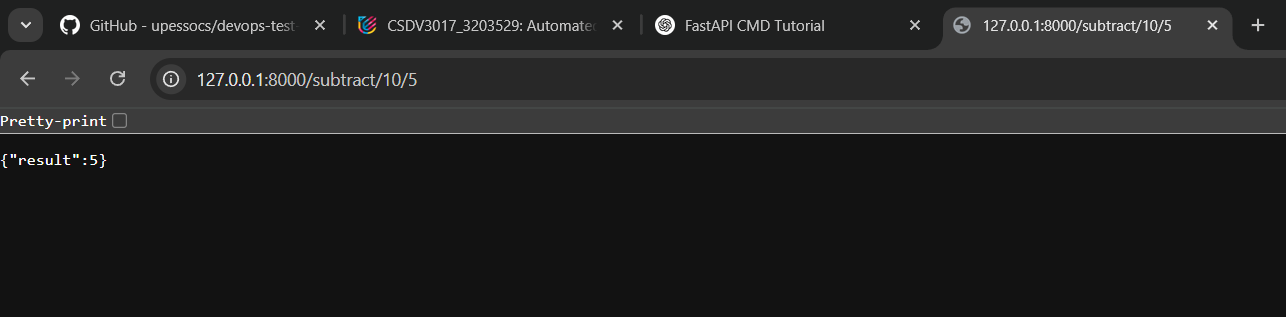
**✅ Step 5: Test the API in a Web Browser**

Open your browser and enter the following URLs to test:

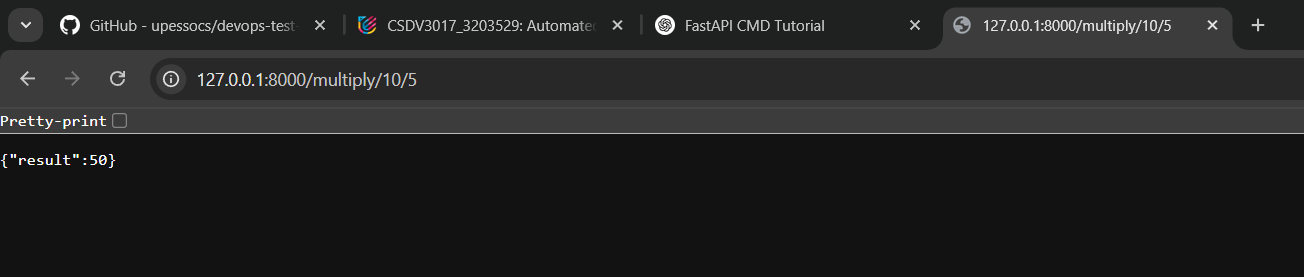
* **Addition**: http://127.0.0.1:8000/add/10/5 → Should return {"result": 15}



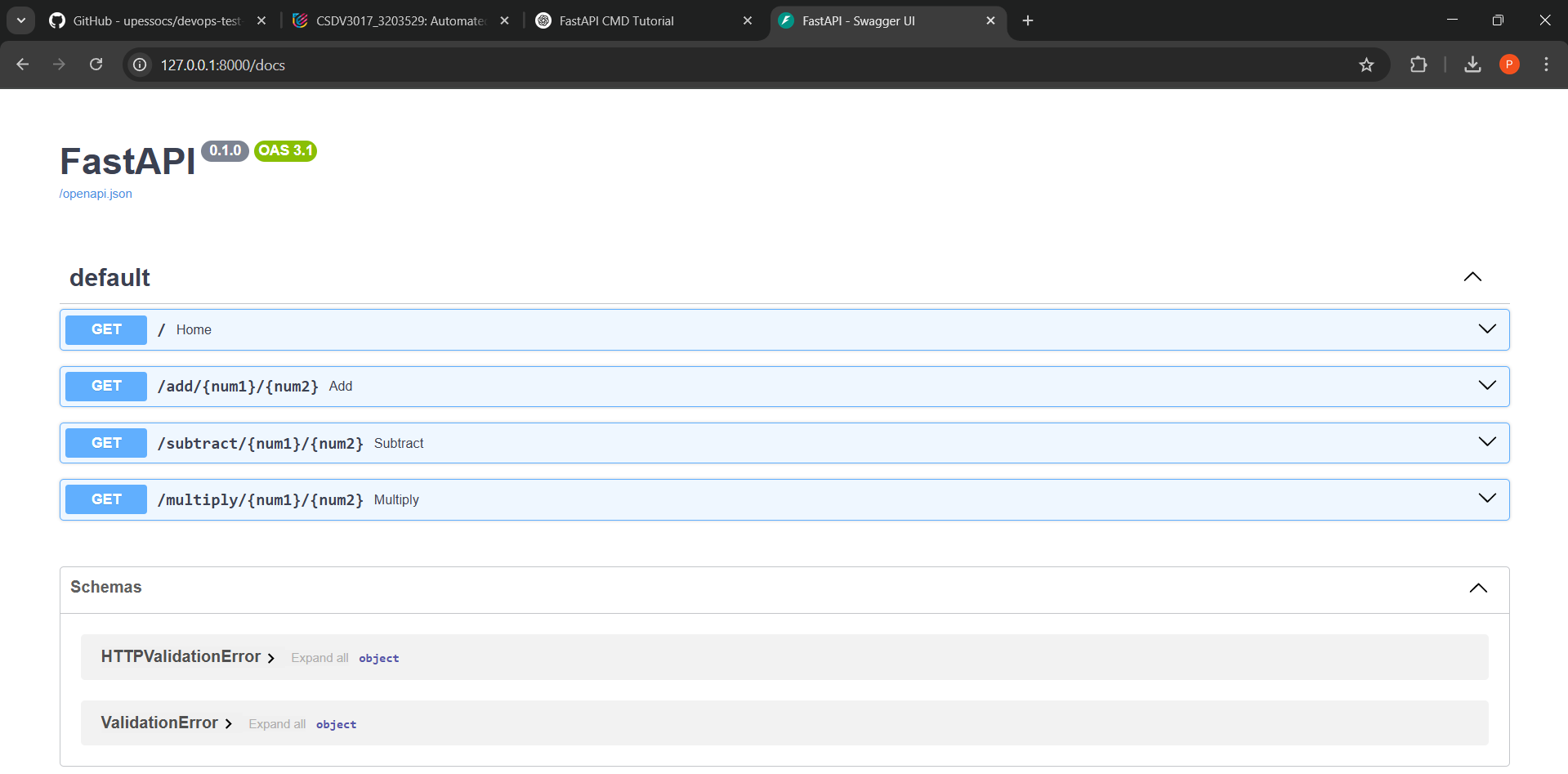
* **Subtraction**: http://127.0.0.1:8000/subtract/10/5 → Should return {"result": 5}



* **Multiplication**: http://127.0.0.1:8000/multiply/10/5 → Should return {"result": 50}



📌 **FastAPI also provides interactive documentation:**  
Open **http://127.0.0.1:8000/docs** in your browser to test the API using Swagger UI.



**✅ Step 6: Automate API Testing**

1. **Create another file** and save it as testAutomation.py.
2. **Copy and paste the following test script:**

**✅ Step 6: Automate API Testing**

Now that our FastAPI server is running, let's write a test script to **automate API testing**.

**📌 Steps to Follow**

1. Create a new Python file and name it **testAutomation.py**.
2. Copy and paste the following test script:

python

CopyEdit

import requests

# Define test cases

testcases = [

("http://127.0.0.1:8000/add/10/5", 15, "Test addition of 10 and 5"),

("http://127.0.0.1:8000/subtract/10/5", 5, "Test subtraction of 10 and 5"),

("http://127.0.0.1:8000/multiply/10/5", 50, "Test multiplication of 10 and 5"),

("http://127.0.0.1:8000/add/-3/3", 0, "Test addition of -3 and 3"),

("http://127.0.0.1:8000/multiply/0/5", 0, "Test multiplication by zero"),

]

def test\_api():

"""

Runs automated tests on API endpoints.

Asserts that the API response matches the expected result.

"""

for url, expected, description in testcases:

response = requests.get(url)

result = response.json()["result"]

assert result == expected, f"{description} FAILED! Expected {expected}, got {result}"

print(f"{description} PASSED ✅")

print("\n✅ All tests passed successfully!")

# Run the test function

if \_\_name\_\_ == "\_\_main\_\_":

test\_api()

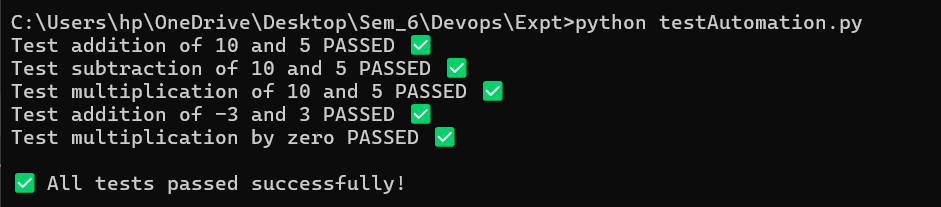
**✅ Step 7: Run the Automated Tests**

1. **Make sure your FastAPI server is running** (python apiserver.py).
2. Open a new **Command Prompt (CMD)**.
3. Navigate to the folder where testAutomation.py is saved:

cd path\to\your\file

1. Run the test script:

python testAutomation.py



✅ **Expected Output (if all tests pass):**

Test addition of 10 and 5 PASSED ✅

Test subtraction of 10 and 5 PASSED ✅

Test multiplication of 10 and 5 PASSED ✅

Test addition of -3 and 3 PASSED ✅

Test multiplication by zero PASSED ✅

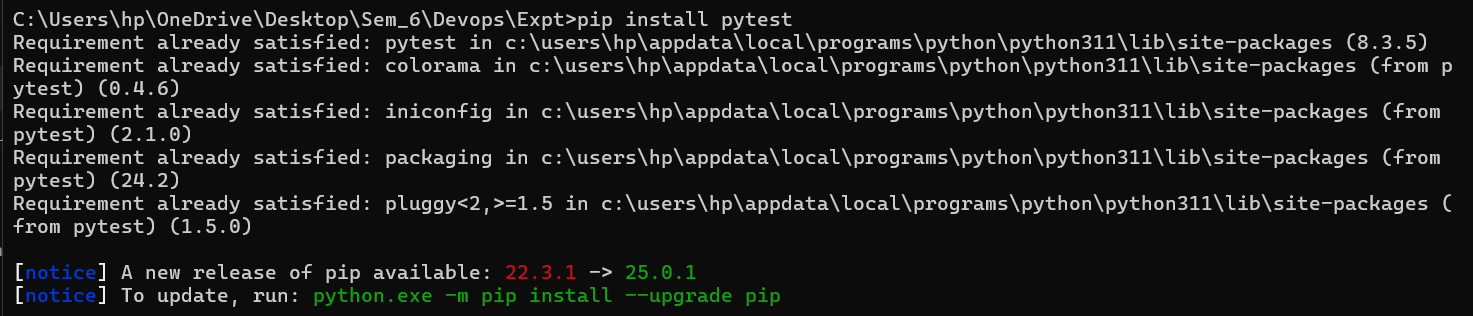
✅ All tests passed successfully!

**✅ Step 8: Enhance Testing with Pytest**

Instead of manually running the script, we can use **pytest** for better test automation.

**Install pytest:**

pip install pytest



**Modify testAutomation.py for pytest:**

Replace the previous code with this **pytest version**:

import pytest

import requests

# Define test cases for parameterized testing

testcases = [

("http://127.0.0.1:8000/add/10/5", 15, "Addition of 10 and 5"),

("http://127.0.0.1:8000/subtract/10/5", 5, "Subtraction of 10 and 5"),

("http://127.0.0.1:8000/multiply/10/5", 50, "Multiplication of 10 and 5"),

("http://127.0.0.1:8000/add/-3/3", 0, "Addition of -3 and 3"),

("http://127.0.0.1:8000/multiply/0/5", 0, "Multiplication by zero"),

]

@pytest.mark.parametrize("url, expected, description", testcases)

def test\_api(url, expected, description):

"""

Parameterized test for API endpoints.

"""

response = requests.get(url)

result = response.json()["result"]

assert result == expected, f"{description} FAILED! Expected {expected}, got {result}"

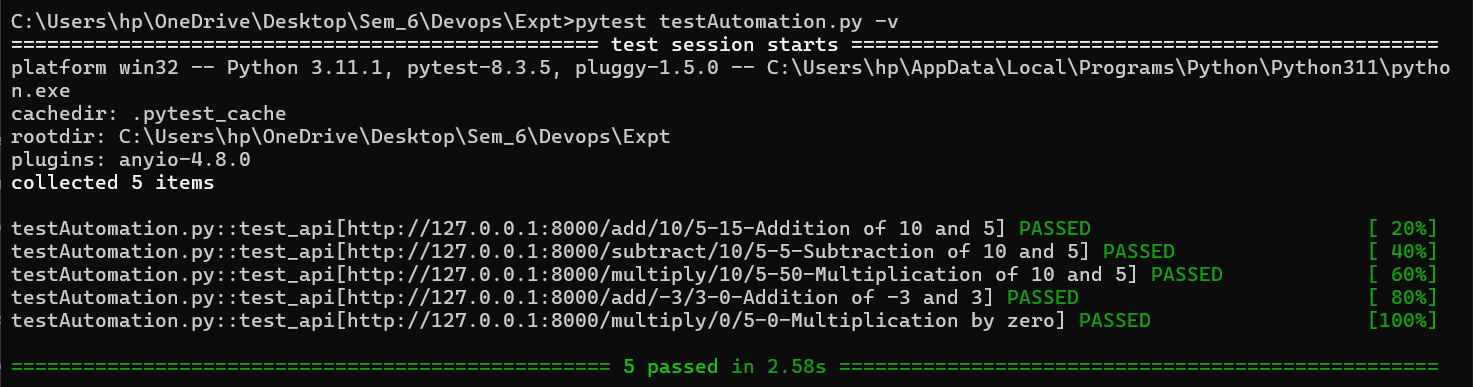
print(f"{description} PASSED ✅")

if \_\_name\_\_ == "\_\_main\_\_":

pytest.main()

**Run pytest:**

pytest testAutomation.py -v



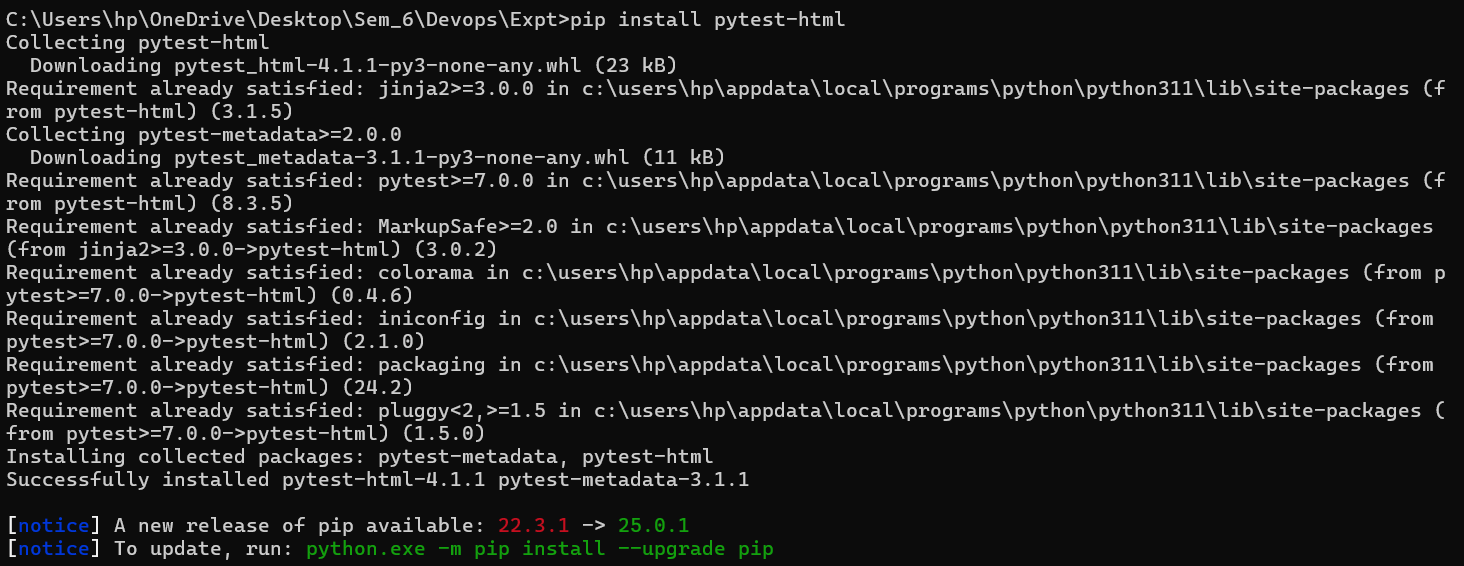
✅ **This makes testing easier and provides detailed reports.**

**✅ Step 9: Expanding for Real-World Use**

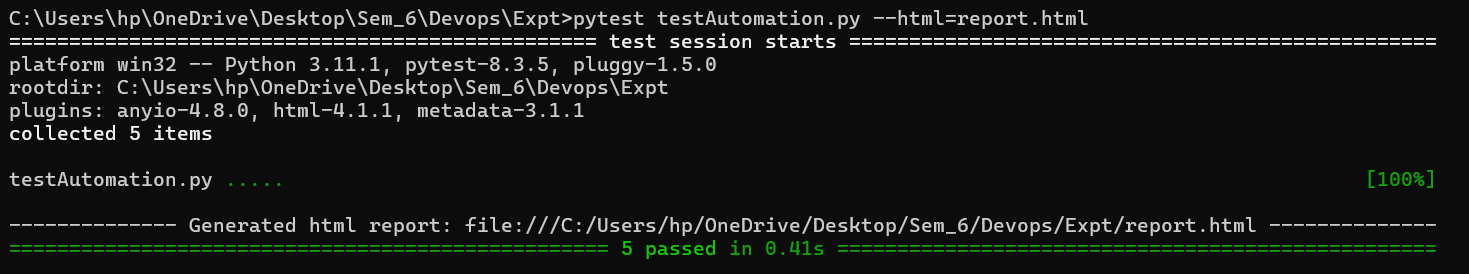
**How to Improve This Further?**

* **Add Edge Cases**: Test negative numbers, non-integer inputs, and large values.
* **Use CI/CD**: Integrate with **GitHub Actions** or **Jenkins** for automated testing.
* **Mock Dependencies**: Use **responses** or **unittest.mock** to simulate external API calls.
* **Generate Reports**: Use **pytest-html** for detailed reports:

pip install pytest-html



pytest testAutomation.py --html=report.html



**🎯 Final Summary**

**What We Achieved**

✔ **Set up a FastAPI server** with endpoints (add, subtract, multiply).  
✔ **Created an automated test script** using requests.  
✔ **Enhanced testing** with pytest for better automation.  
✔ **Expanded the project for real-world usage** with CI/CD, reporting, and mocking.

**🚀Next Steps**

Now that you have **Backend API Test Automation** working on **Windows**, try:

* **Adding more endpoints** (e.g., division, square root).
* **Implementing input validation** (handle non-integer values).
* **Scaling the project** to integrate with databases.