Ritesh Jha | 16010423076 | SY-IT(B3)
WPL Tutorial - 8 | 24-04-25

Aim: Study of Web Services

1 : Identify Real-World Applications Using Web Services

- Payment Gateways like Razorpay and PayPal use web services to process transactions between users, merchants, and banks securely.

- Weather APIs like OpenWeatherMap provide real-time weather data by letting applications request weather conditions based on city names or coordinates.

- Authentication Services like Google OAuth or Facebook Login allow users to log in to websites using their Google or Facebook account, thanks to secure web APIs.

2 : Implementing Web Services

In the terminal write :

```
pip install flask
```

Web services Flask app code :

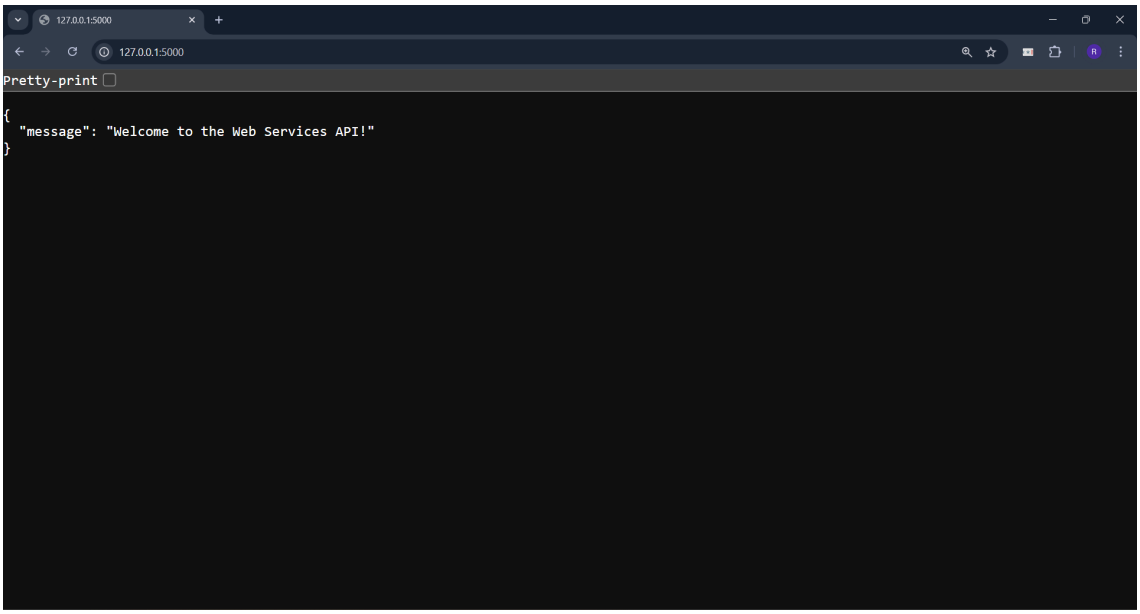
```
from flask import Flask, jsonify, request

# Initialize the Flask app
app = Flask(__name__)

# Route to check if the service is up
@app.route('/', methods=['GET'])
def home():
    return jsonify({'message': 'Welcome to the Web Services
API!'}), 200

# Basic greeting route
@app.route('/api/greet', methods=['GET'])
def greet():
    return jsonify({'message': 'Hello, Welcome to Web Services'}),
```

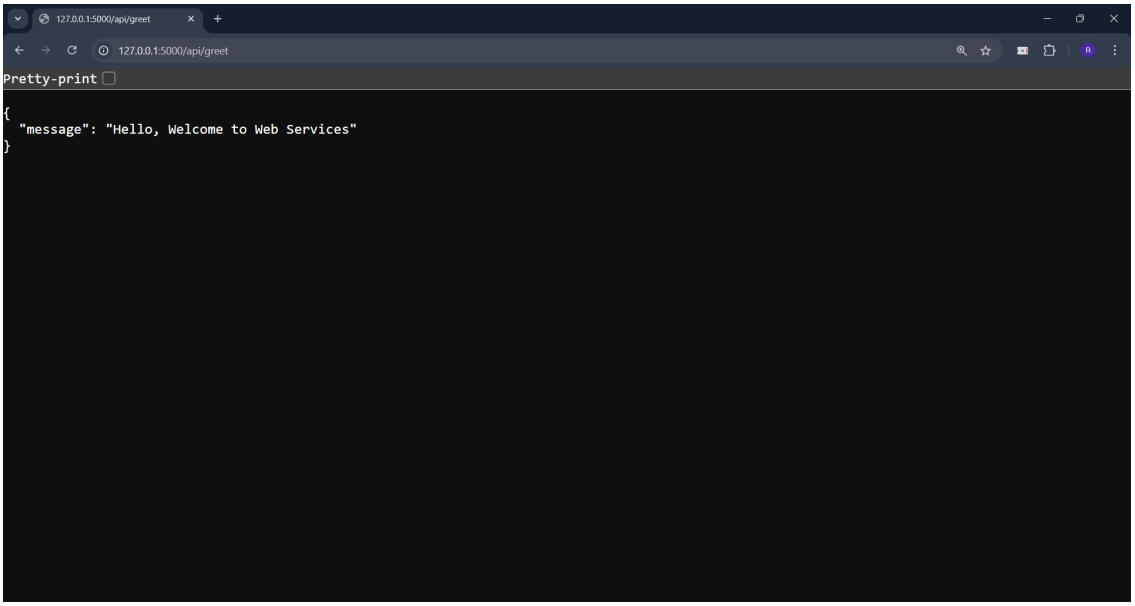
```
200

# Personalized greeting with query parameter
@app.route('/api/greet/<name>', methods=['GET'])
def greet_user(name):
    return jsonify({'message': f'Hello, {name}! Welcome to Web
Services'}), 200

# Run the app
if __name__ == '__main__':
    app.run(debug=True)
```

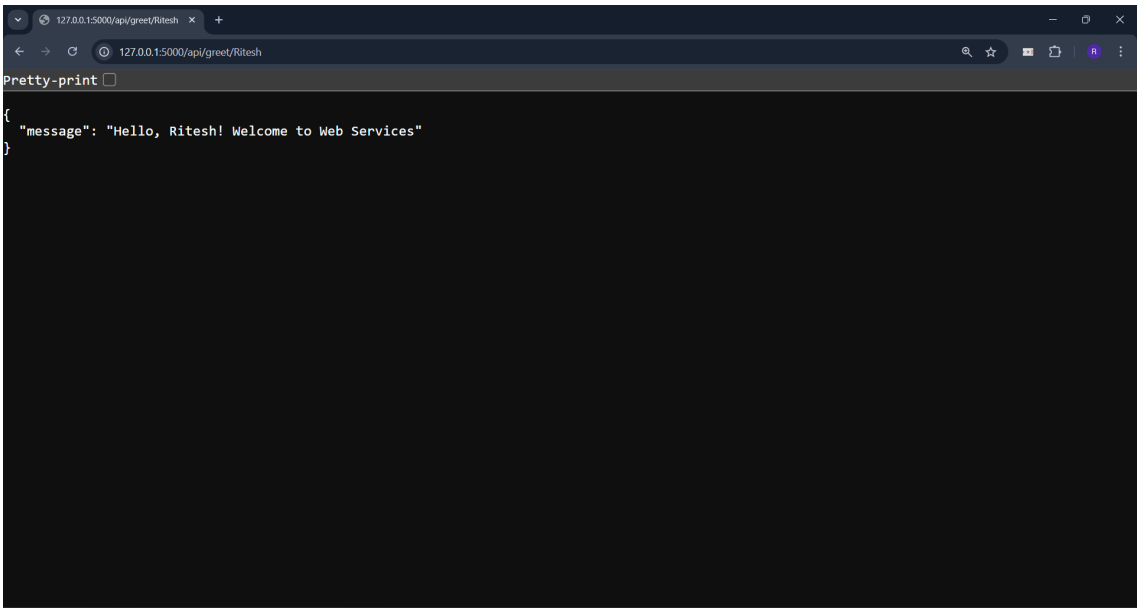
Start the flask app :

```
python app.py
```

Visit the browser localhost:







3 : Consuming a Web Service (External API)

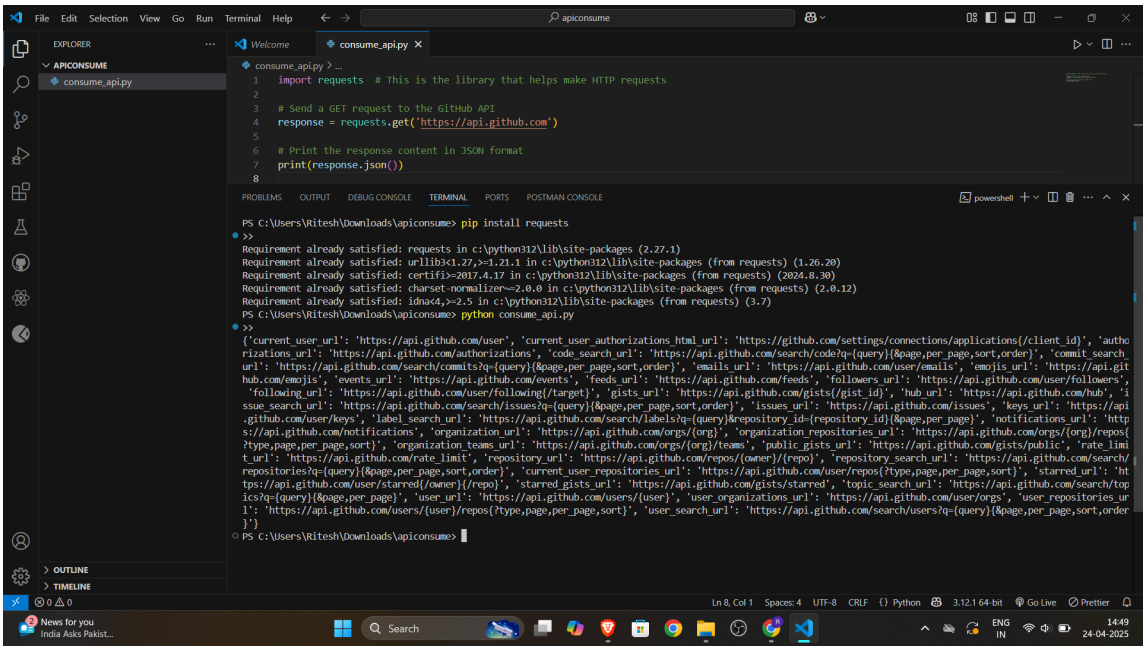
API Endpoint Used:
https://api.github.com

consume_api.py

```
import requests  # This is the library that helps make HTTP requests

# Send a GET request to the GitHub API
```

```
response = requests.get('https://api.github.com')

# Print the response content in JSON format
print(response.json())
```

Install Packages & send request:
`pip install requests`
`python consume_api.py`



Successfully connected to the GitHub API and received data.

Post Lab Questions

Q1) Compare SOAP and RESTful Web Services

SOAP (Simple Object Access Protocol) is a protocol that uses XML for every request and response. It is strict, has more rules, and is mostly used in enterprise apps like banking systems.

REST (Representational State Transfer) is an architecture style that uses normal HTTP methods like GET, POST, PUT, DELETE. It is faster, simpler, and usually returns data in JSON or XML format.

Q2) Advantages and Challenges of Using Web Services

Advantages:

- Web services allow different systems (Java, Python, .NET) to talk to each other easily.
- You can reuse old software by exposing its logic through web services.
- Helps integrate services across different departments or companies.

Challenges:

- If not secured properly, they can be vulnerable to attacks.
- Requires good knowledge of APIs, data formats (like JSON, XML), and error handling.
- Versioning and maintenance can be tough if multiple clients depend on your API.

Outcomes:

CO4: Implement Web applications and Web Services

Conclusion:

In this tutorial, I explored the concept of web services and understood how they help different applications work together. I identified real-world examples, created a basic RESTful web service using Python Flask, and also consumed an external API using GitHub's public endpoint. This showed me how web services can send and receive structured data over the internet.