

Gabriel Q. Escobido

BSCpE – 2A

SOFTWARE DESIGN

Laboratory Exercise No. 7

Title: Designing and Consuming APIs

Brief Introduction

This exercise explores the creation and consumption of APIs in software development. Students will design a simple API and consume an external REST API.

Objectives

- Understand the principles of API design.
- Build a simple REST API using Flask.
- Learn how to consume an API using Python's requests library.

Detailed Discussion

API Design Principles:

- **Consistency:** Ensure uniform structure and naming conventions.
- **Security:** Use authentication methods like API keys.
- **Documentation:** Provide clear instructions on how to use the API.

Materials

API Component	Description	Example
Endpoint	URL to access a resource.	<code>/api/users</code>
HTTP Methods	Actions performed on data (GET, POST, etc.).	GET for retrieval, POST for creation
Response Format	Data format for communication (JSON, XML).	<code>{ "id": 1, "name": "Alice" }</code>

- Python environment
- Flask library
- Access to an open REST API (e.g., JSONPlaceholder)

Procedure

1. Create a simple REST API using Flask:

```
from flask import Flask, jsonify
```

```
app = Flask(__name__)
```

```
@app.route('/api/greet/<name>', methods=['GET'])
```

```
def greet(name):
```

```
    return jsonify({"message": f"Hello, {name}!"})
```

```
if __name__ == '__main__':
```

```
    app.run(debug=True)
```

1. Use Python's requests library to consume an external API:

```
import requests
```

```
response = requests.get("https://jsonplaceholder.typicode.com/posts/1")
```

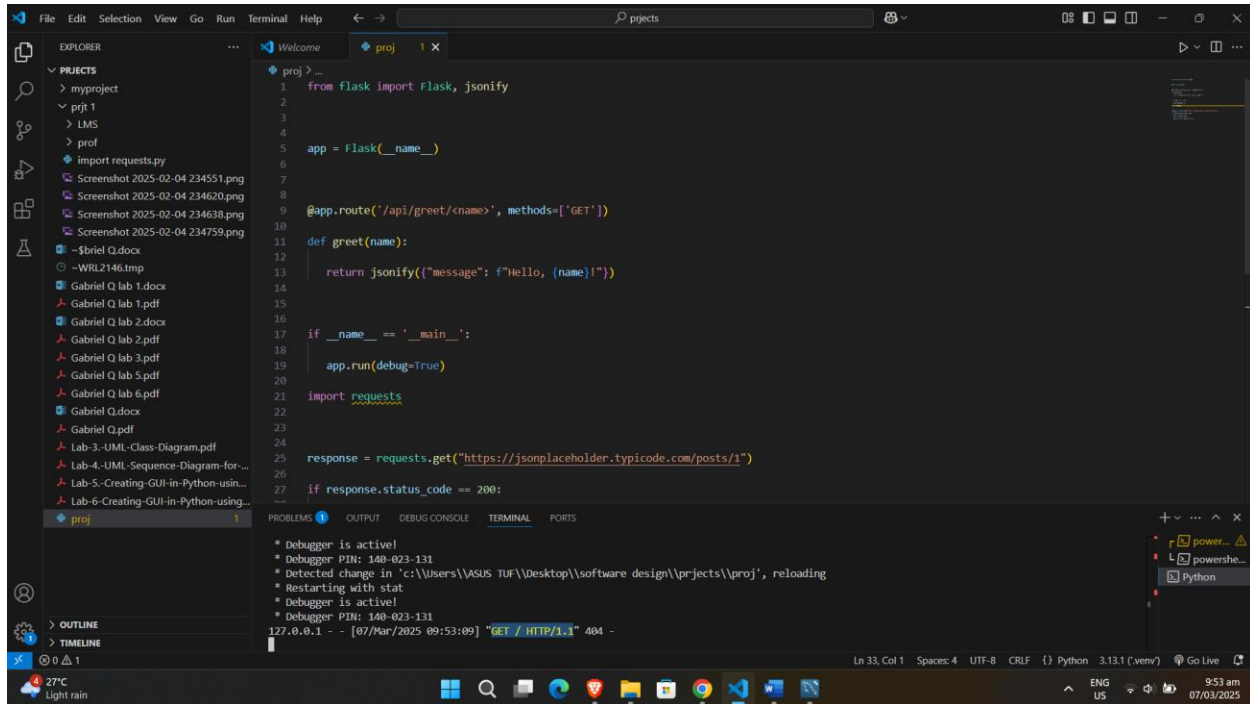
```
if response.status_code == 200:
```

```
    data = response.json()
```

```
    print(f"Title: {data['title']}")
```

Results

Submit the Flask API code and the output of consuming the external API.



The screenshot shows a Visual Studio Code editor with a project named 'proj'. The Explorer sidebar on the left lists various files, including 'import requests.py' and several screenshots. The main editor window displays a Python file with the following code:

```
1 from flask import Flask, jsonify
2
3
4
5 app = Flask(__name__)
6
7
8
9 @app.route('/api/greet/<name>', methods=['GET'])
10
11 def greet(name):
12     return jsonify({"message": f"Hello, {name}!"})
13
14
15
16
17 if __name__ == '__main__':
18     app.run(debug=True)
19
20
21 import requests
22
23
24
25 response = requests.get("https://jsonplaceholder.typicode.com/posts/1")
26
27 if response.status_code == 200:
```

The bottom panel shows the TERMINAL output, which includes messages about the debugger being active and a successful GET request to the external API:

```
* Debugger is active!
* Debugger PIN: 140-023-131
* Detected change in 'c:\Users\ASUS TUF\Desktop\software design\projects\proj', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 140-023-131
127.0.0.1 - - [07/Mar/2025 09:53:09] "GET / HTTP/1.1" 404 -
```

Follow-Up Questions

1. What are the key considerations when designing an API?

ANSWER: Key considerations when designing an API include clarity, consistency, scalability, versioning, security, and error handling. It should be simple to use, flexible to scale, and secure, with clear documentation and meaningful error responses.

2. How can you secure your API?

ANSWER: Key considerations when designing an API include clarity, consistency, scalability, versioning, security, and error handling. It should be simple to use, flexible to scale, and secure, with clear documentation and meaningful error responses.

3. Describe the importance of documentation in API development.

ANSWER: Documentation is crucial in API development because it helps developers understand how to use the API effectively. Well-written documentation ensures ease of integration, provides clarity on available endpoints, and accelerates the adoption of the API by others.

