# RESTful APIs with Flask – II

✓ JSON basics
✓ JSON in RESTful APIs
✓ JSON handling in Flask
✓ Serialization / deserialization
✓ Testing APIs
✓ Step-by-step program with explanations

**1. What is JSON? (Basics)**
**JSON** = *JavaScript Object Notation*
It is a lightweight format used to **store and exchange data** between applications, especially web and mobile apps.
✓ **Why JSON is Popular?**

- Human-readable
- Language-independent
- Supported everywhere (web, mobile, APIs)
- Fast and lightweight

**2. JSON Structure**
JSON contains:
**A. Objects (key–value pairs)**

```
{
  "name": "John",
  "age": 25
}
```

**B. Arrays**

```
[
  {"id": 1, "name": "Book"},
  {"id": 2, "name": "Pen"}
]
```

**C. Values (allowed types)**

- String "text"
- Number 10
- Boolean true/false
- Null null
- Object {...}
- Array [...]

**3. JSON in RESTful APIs**
REST APIs exchange data using:
✓ **JSON Requests** → Client → Server
✓ **JSON Responses** → Server → Client
**Example Flow:**

1. Android / Web app sends data → JSON body
2. Flask API receives JSON → processes it

3. Flask API returns result → JSON response

**4. JSON Serialization & Deserialization**
**Serialization**
→ Converting Python objects → JSON
Example:
jsonify({"name": "Alice"})
**Deserialization**
→ JSON input → Python dictionary
Example:
data = request.get_json()

**5. Handling JSON in Flask – REST API Concepts**
Flask provides:
**A. Getting JSON from a request**
request.get_json()
**B. Sending JSON as a response**
jsonify(data)
**C. Sending status codes**
return jsonify({"error": "Not found"}), 404

**6. Testing APIs**
**Methods to test a Flask API:**
✓ Browser (for GET only)
✓ Postman
✓ cURL
✓ Android app / JS app
✓ Python script using requests library
For JSON handling, Postman is best.

**7. Step-By-Step Flask Program (REST API + JSON Handling)**

This example demonstrates:
✓ JSON POST
✓ JSON GET
✓ Serialization
✓ Deserialization
✓ Testing

## REST API With JSON (Flask)

**Step 1 — Install Flask**

```
pip install flask
```

**Step 2 — Create app.py**

```
from flask import Flask, request, jsonify

app = Flask(__name__)

# Temporary in-memory database
students = []
```

**Step 3 — Create API: Add a Student (POST Request)**
**→ This API receives JSON from client**

```
@app.route('/add_student', methods=['POST'])
def add_student():
    data = request.get_json()      # Deserialize JSON → Python dictionary

    # Extract fields
    name = data.get("name")
    age = data.get("age")
    course = data.get("course")

    student = {
        "id": len(students) + 1,
        "name": name,
        "age": age,
        "course": course
    }

    students.append(student)

    return jsonify({"message": "Student added successfully", "student": student}), 201
```

**Step 4 — Create API: Get All Students (GET Request)**
**→ This API returns JSON data**

```
@app.route('/students', methods=['GET'])
```

```
def get_students():
    return jsonify(students), 200   # Serialize Python → JSON
```

**Step 5 — Run Flask Server**

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

**8. Testing the API in Postman**

**A. Test POST /add_student**

- Method: **POST**
- URL: http://127.0.0.1:5000/add_student
- Body → Raw → JSON:

```
{
  "name": "John",
  "age": 22,
  "course": "Computer Science"
}
```

**Expected Output:**

```
{
  "message": "Student added successfully",
  "student": {
    "id": 1,
    "name": "John",
    "age": 22,
    "course": "Computer Science"
  }
}
```

**B. Test GET /students**

- Method: **GET**
- URL: http://127.0.0.1:5000/students

**Expected Output:**

```
[
  {
    "id": 1,
    "name": "John",
    "age": 22,
    "course": "Computer Science"
  }
]
```

**9. Key Concepts**

**A. request.get_json()**

- Reads JSON sent by client
- Converts JSON → Python dict

**B. jsonify()**

- Converts Python dict/list → JSON

- Sets correct HTTP headers

**C. HTTP Status Codes**

- 200 → Success
- 201 → Created
- 400 → Bad request
- 404 → Not found

**D. In-memory database**

- Temporary list used here
- In real apps → use MySQL, MongoDB, Firebase, etc.

**10. Summary**

| Topic | Explanation |
|---|---|
| JSON | Data exchange format (lightweight) |
| Serialization | Python → JSON |
| Deserialization | JSON → Python |
| Flask REST API | Build endpoints for GET/POST |
| Testing | Postman is used to test APIs |
| Example Program | Student API with JSON handling |