### **Project 6: RESTful API**

#### **Objective:**

Create a simple RESTful API using Flask that performs CRUD (Create, Read, Update, Delete) operations on a dataset (e.g., a book collection). Include endpoints for adding, retrieving, updating, and deleting records.

### **Instructions**

#### **Step 1: Set Up the Environment**

1. Create a new Python file called restful\_api.py.

Install the required library:  
Copy code  
pip install flask

#### **Step 2: Import Necessary Libraries**

Import Flask and other necessary modules at the top of your script:  
python  
Copy code  
from flask import Flask, jsonify, request

#### **Step 3: Create Flask App and Dummy Data**

Initialize the Flask app:  
python  
Copy code  
app = Flask(\_\_name\_\_)

Create a list of dictionaries to represent your dataset:  
python  
Copy code  
books = [

{"id": 1, "title": "1984", "author": "George Orwell"},

{"id": 2, "title": "To Kill a Mockingbird", "author": "Harper Lee"},

{"id": 3, "title": "The Great Gatsby", "author": "F. Scott Fitzgerald"}

]

#### **Step 4: Create Routes for CRUD Operations**

**GET /books**: Retrieve all books.  
python  
Copy code  
@app.route("/books", methods=["GET"])

def get\_books():

return jsonify(books)

**GET /books/<id>**: Retrieve a specific book by ID.  
python  
Copy code  
@app.route("/books/<int:id>", methods=["GET"])

def get\_book(id):

book = next((book for book in books if book["id"] == id), None)

return jsonify(book) if book else ("Book not found", 404)

**POST /books**: Add a new book.  
python  
Copy code  
@app.route("/books", methods=["POST"])

def add\_book():

new\_book = request.get\_json()

books.append(new\_book)

return jsonify(new\_book), 201

**PUT /books/<id>**: Update an existing book by ID.  
python  
Copy code  
@app.route("/books/<int:id>", methods=["PUT"])

def update\_book(id):

book = next((book for book in books if book["id"] == id), None)

if book:

updated\_data = request.get\_json()

book.update(updated\_data)

return jsonify(book)

return "Book not found", 404

**DELETE /books/<id>**: Delete a book by ID.  
python  
Copy code  
@app.route("/books/<int:id>", methods=["DELETE"])

def delete\_book(id):

global books

books = [book for book in books if book["id"] != id]

return "Book deleted", 204

#### **Step 5: Run the Application**

Add the following code at the bottom of your script to run the Flask server:  
python  
Copy code  
if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

1. Run the script and access the API using a tool like Postman or directly from the browser by navigating to http://127.0.0.1:5000/books.

#### **Step 6: Test Each Endpoint**

1. Test the GET /books and GET /books/<id> endpoints to ensure they retrieve the correct data.
2. Use POST /books to add a new book, and verify it appears in the list.
3. Use PUT /books/<id> to update an existing book and confirm the changes.
4. Use DELETE /books/<id> to remove a book and verify that it no longer appears.