Python Developer Internship

Soumen Das

November 18, 2025

Task Submission Details

- **Task Number:** 4
- **Topic:** Build a REST API with Flask
- **Objective:** Create a REST API that manages user data using Python and Flask.
- **Deliverables:** Flask app with GET, POST, PUT, and DELETE routes.

1 Project Overview

This report documents the successful completion of Task 4. A RESTful API was built using the micro-framework **Flask**. The application manages a collection of ”User” objects stored in an in-memory data structure (list of dictionaries).

The API supports the full CRUD (Create, Read, Update, Delete) lifecycle:

1. **GET /users**: Retrieve the list of all registered users.
2. **GET /users/<id>**: Retrieve details of a specific user.
3. **POST /users**: Create a new user profile.
4. **PUT /users/<id>**: Update an existing user’s details.
5. **DELETE /users/<id>**: Remove a user from the system.

2 Implementation Code

The following Python code implements the API logic.

```python
from flask import Flask, jsonify, request

app = Flask(__name__)

# In-memory database
users = [
    {"id": 1, "name": "Soumen Das", "role": "Intern"},
    {"id": 2, "name": "Admin", "role": "Supervisor"}
]

# Helper to find user
def find_user(user_id):
    return next((u for u in users if u["id"] == user_id), None)

```

```python
@app.route('/users', methods=['GET'])
def get_users():
    return jsonify({"users": users})

@app.route('/users/<int:user_id>', methods=['GET'])
def get_user(user_id):
    user = find_user(user_id)
    if user:
        return jsonify(user)
    return jsonify({"error": "Not found"}), 404

@app.route('/users', methods=['POST'])
def create_user():
    data = request.get_json()
    new_id = users[-1]['id'] + 1 if users else 1
    new_user = {
        "id": new_id,
        "name": data.get('name'),
        "role": data.get('role', 'User')
    }
    users.append(new_user)
    return jsonify(new_user), 201

@app.route('/users/<int:user_id>', methods=['PUT'])
def update_user(user_id):
    user = find_user(user_id)
    if not user:
        return jsonify({"error": "Not found"}), 404
    data = request.get_json()
    user['name'] = data.get('name', user['name'])
    user['role'] = data.get('role', user['role'])
    return jsonify(user)

@app.route('/users/<int:user_id>', methods=['DELETE'])
def delete_user(user_id):
    user = find_user(user_id)
    if not user:
        return jsonify({"error": "Not found"}), 404
    users.remove(user)
    return jsonify({"message": "Deleted successfully"})

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

Listing 1: flask_api.py

3 Testing Instructions

To verify the application functionality:

Prerequisites

Ensure Flask is installed in the python environment:

```
pip install flask
```

Execution

Run the application using the command:

```
python flask_api.py
```

The server will start on `http://127.0.0.1:5000/`.

API Testing Examples (cURL)

1. Get All Users:

```
curl http://127.0.0.1:5000/users
```

2. Create New User:

```
curl -X POST -H "Content-Type: application/json" \
     -d '{"name":"New Intern", "role":"Dev"}' \
     http://127.0.0.1:5000/users
```