REST API with Flask — Project Description

A concise, portfolio-ready description for the repository `vijaytakbhate2002/rest_api_with_flask_learning_repository`.

Project Overview

REST API with Flask is a learning project where I built a simple RESTful API using Flask (Python) and tested the endpoints using Postman. The repository demonstrates core API concepts — routing, request/response handling, JSON input/output, basic validation, and simple in-memory or file-based data persistence — and serves as an example of backend fundamentals suitable for inclusion in a developer portfolio.

Key Features

- Lightweight Flask backend implementing RESTful endpoints
- JSON request and response handling
- CRUD operations for a sample resource (e.g., users, tasks, or notes)
- Input validation and basic error handling
- Local testing via Postman collection (included if available)
- Clear instructions to run locally

Tech Stack

- Python
- Flask
- (Optional) Flask-RESTful / Flask-SQLAlchemy (if used)
- Postman for API testing
- Dependencies listed in requirements.txt

Example Endpoints

(Replace with actual endpoints from your project) - GET /items — list all items - GET /items/ — get a single item by id - POST /items — create a new item - PUT /items/ — update an item - DELETE /items/ — delete an item

Example request (POST /items)

```
{
 "title": "Buy groceries",
 "completed": false
}
```

Response:

```
{
 "id": 1,
 "title": "Buy groceries",
 "completed": false
}
```

How to Run Locally

1. Clone the repo: git clone https://github.com/vijaytakbhate2002/rest_api_with_flask_learning_repository.git
2. Create and activate a virtual environment: python -m venv venv source venv/bin/activate # macOS/Linux venv\Scripts\activate # Windows
3. Install dependencies: pip install -r requirements.txt
4. Start the server (adjust entrypoint if different): export FLASK_APP=app.py # macOS/Linux export FLASK_ENV=development flask run -or- python app.py
5. Use the included Postman collection (or import endpoints manually) to send requests and test the API.

Testing with Postman

- Import the provided Postman collection (if included) to quickly run and verify all endpoints.
- Example tests: create resource, list resources, update resource, delete resource, and verify error handling.

What I Learned

- Designing REST endpoints and HTTP method usage
- Handling JSON payloads and response codes
- Basic request validation and error responses
- Using Postman for manual API testing
- Packaging a small Flask app for local development

Suggestions to Improve Portfolio Presentation

To make this project stand out on your portfolio, consider adding: - README enhancements: diagrams, endpoint documentation, example requests/responses - Postman collection link and example environment - Automated tests (unit tests / pytest) and CI (GitHub Actions) - Dockerfile + docker-compose for easy setup - Swagger/OpenAPI documentation for discoverability - Persist data with a lightweight DB (SQLite + SQLAlchemy) to demonstrate persistence - Sample client (simple HTML/JS or a small React app) that consumes the API - Screenshots or a short demo GIF showing Postman requests and responses

Short Blurb for Portfolio

"REST API with Flask — A compact learning project demonstrating RESTful API design using Flask. Implemented CRUD operations, JSON handling, and tested endpoints in Postman. Ideal demonstration of backend fundamentals for portfolio."

If you'd like, I can: - Draft a full README.md file using the exact endpoints and commands from your repo, - Generate a Postman collection or a simple OpenAPI spec, - Or create a Dockerfile and a GitHub Actions workflow to add CI and make the project more portfolio-ready.

(Committed by GitHub Copilot assistant.)