PROJECT

1. CORE PYTHON
2. **Project:** Personal Expense Tracker
3. **Features:**

* Add, delete, and update expenses.
* Store data in JSON/CSV.
* Handle invalid inputs gracefully.
* Why: Covers file handling, error handling, OOP basics.

1. FLASK/ WEB DEVELOPMENT
2. **Project:** Blog Website
3. **Features:**

* User authentication (register/login).
* CRUD (Create, Read, Update, Delete) for blog posts.
* HTML templates + CSS.
* Why: Covers Flask routes, Jinja templating, sessions, forms.

1. DATABASE (SQL + ORM)
2. **Project:** To-Do App with SQLite & SQLAlchemy
3. **Features:**

* Store tasks in SQLite DB.
* Add relationships (users → tasks).
* Query tasks (pending/completed).
* Why: Teaches SQL + ORM (SQLAlchemy).

1. REST APIs (FASTAPI or FLASK-RESTful)
2. **Project:** Weather API Service
3. **Features:**

* API endpoints: /weather/<city> returns JSON.
* Fetch data from OpenWeather API.
* Include error handling (invalid city).
* Why: Learn API development, JSON, status codes.

1. TESTING (PYTEST / UNIT TESTING)

a. **Project Extension**: Add Tests to Your API

- Write unit tests for your weather API.

- Test input validation & edge cases.

- Mock API calls.

- Why: Covers Pytest, mocking, test-driven mindset.

1. DOCKER & CONTAINERS

a. **Project:** Dockerize Your Blog or API

* Create a Dockerfile for your Flask/FastAPI app.
* Run app n a container.
* Push image to Docker Hub.
* Why: Learn Docker basics + containerization.

1. CI/CD (GITHUB ACTIONS)

a**. Project:** CI/CD for Your API

b. **Setup GitHub Actions workflow:**

- Run tests on every push.

- Build Docker image.

- Deploy to Heroku/AWS (optional).

- Why: Learn automation + pipelines.

1. SQL + DATA ANALYSIS

a. **Project**: Movie Database Explorer

- Load IMDb dataset into PostgreSQL/MySQL.

- Write SQL queries: top-rated movies, most popular actors, etc.

- Build a small Flask/FastAPI API to query results.

- Why: Learn SQL + integration with Python.

1. AUTHENTICATION & SECURITY

a. Project: Secure Notes App

b. Features:

- User login/signup (JWT tokens or Flask sessions).

- Store encrypted notes.

- Password hashing.

- Why: Learn auth, JWT, security best practices.

1. ADVANCE (ASYNC + MICROSERVICES)

a. **Project:** Chat App with FastAPI + WebSockets

- Real-time messaging.

- Use Redis for message storage.

- Deploy with Docker Compose.

- Why: Learn **async, websockets, microservices**.