Problem Statement: Developing a Retail Inventory and Order Management API using FastAPI

As part of the corporate training program, your task is to build a RESTful API for Walmart that manages retail inventory and customer orders. You will use **FastAPI** to create the API endpoints and interact with a **MySQL** or **MongoDB** database for data storage. Additionally, you can Prompt **Ollama** for natural language processing to enhance user interactions and generate codes

Data: Download: https://drive.google.com/file/d/1dVP42NMY34giT7aIV6ihtPWwY0vW232w/view?usp=drive_link

Modify the dataset to add a column order_status to record the following: pending, shipped, delivered, cancelled against various (randomly assign these values to rows)

Project Objectives:

1. **Inventory Management:**
    - **Add New Products:**
        - Create an endpoint to add new products with details such as product ID, name, category, price, stock quantity, and supplier information.
    - **View Products:**
        - Implement endpoints to retrieve product information individually or in bulk.
        - Support filtering by category, price range, or stock availability.
    - **Update Products:**
        - Allow updates to product details like price adjustments or stock replenishment.
    - **Delete Products:**
        - Enable removal of discontinued products from the database.
2. **Order Processing:**
    - **Create Orders:**
        - Develop an endpoint for customers to place orders, capturing customer information and the list of products ordered.
        - Decrease stock quantities based on ordered items.
    - **View Orders:**
        - Implement endpoints to retrieve order details by order ID or customer ID.
    - **Update Orders:**
        - Allow updates to order status (e.g., pending, shipped, delivered, canceled).
    - **Cancel Orders:**
        - Handle order cancellations and restock items accordingly.
3. **User Authentication & Authorization:**
    - **Implement Security:**

        - Use OAuth2 with JWT tokens for secure API access.
    - **User Roles:**
        - Define roles such as admin, employee, and customer with varying access levels.
    - **Access Control:**
        - Restrict certain endpoints to authorized roles only (e.g., only admins can add or delete products).
4. **Database Integration:**
    - **Choose a Database:**
        - Use **MySQL** or **MongoDB** for data storage.
    - **ORM/ODM Usage:**
        - Utilize an ORM like SQLAlchemy for MySQL or an ODM like Motor for MongoDB.
    - **Data Modelling:**
        - Define clear models for products, orders, and users.
5. **API Documentation:**
    - **Endpoint Documentation:**
        - Ensure all endpoints have clear descriptions, parameter explanations, and example responses.
        - You can use Tabnine for Documentation
6. **Testing & Validation:**
    - **Input Validation:**
        - Use Pydantic models to validate request data.
    - **Unit Tests:**
        - Write unit tests for critical components of your API.
    - **Error Handling:**
        - Implement global exception handlers to manage errors gracefully.

Deliverables:

- **Source Code:**
    - A GitHub repository containing all source code and commit history.
- **README File:**
    - Instructions on how to set up and run the application locally.
    - Explanation of the project structure and technologies used.
- **Database Schema:**
    - Scripts to create and populate the database with sample data.
- **API Documentation:**
    - Accessible documentation via Swagger UI at /docs and ReDoc at /redoc.
- **Optional Ollama Integration:**
    - Documentation on how Ollama is integrated and examples of natural language queries.