FastAPI Assignment: E-commerce Inventory Management System

Objective

Develop a RESTful API for managing an e-commerce inventory system. This system should handle product listings, inventory management, and user roles such as inventory managers and administrators.

Assignment Overview

You are tasked with creating an Inventory Management API that allows users to manage product listings, track inventory levels, and perform role-specific operations. The API will feature authentication and role-based access control.

Requirements

1. **Project Setup**
    a. Use FastAPI to build the application.
    b. PostgreSQL will serve as the database.
    c. Setup PGVector -

    ```
    docker pull pgvector/pgvector:0.7.0-pg16

    docker run --name postgres -e POSTGRES_PASSWORD=postgres -d -p 5432:5432
    pgvector/pgvector:0.7.0-pg16
    ```

    d. Use SQLAlchemy for the ORM.
    e. Ensure environment variables are used for configuration.
2. **Database Connection and Migrations**
    a. Connect the application to a PostgreSQL database.
    b. Utilize Alembic to manage database migrations.
    c. Define an initial migration to establish the database schema.
3. **Database Schema**
    a. Create users table with fields: id, username, password_hash, role (e.g., admin, manager).
    b. Create products table with fields: id, name, description, price, stock_quantity.
    c. Create categories table with fields: id, name, description.
    d. Establish relationships, such as a product belonging to multiple categories.
4. **CRUD Operations**
    a. Implement endpoints for the following operations:
        i. **Products**
            1. Create a new product.

        2. Retrieve all products or a specific product by ID.
        3. Update a product's details and stock quantity.
        4. Delete a product.
    ii. **Categories**
        1. Add new categories.
        2. Retrieve a list of categories or a specific category.
        3. Update and delete categories.

5. **Authentication and Authorization**
    a. Implement user registration and login using JWT for authentication.
    b. Ensure user passwords are securely hashed.
    c. Protect endpoints to ensure only authenticated users can access them.
    d. Role-based access:
        i. "admin" users can perform all operations.
        ii. "manager" users can manage product and category listings but cannot delete them.
6. **Testing**
    a. Develop unit tests for all API endpoints.
    b. Include tests for authentication, authorization, and business logic scenarios, such as low stock alerts.
7. **Documentation**
    a. Utilize FastAPI's auto-generated API documentation features.
    b. Ensure the API documentation includes detailed descriptions of each endpoint, including request and response structures with examples.

BONUS

8. **Inventory Management Logic**
    a. Ensure stock levels are accurately updated during product updates.
    b. Implement a check for low inventory levels and trigger alerts/logging when below a specified threshold.

Technical Constraints

- Use Python 3.8 or later.
- Adhere to PEP 8 standards for code quality.
- Implement proper error handling and return appropriate HTTP status codes.

Deliverables

- A GitHub repository containing the complete project.
- A README file with instructions for setting up and running the application.
- Instructions for performing database migrations.
- Documentation on API usage with sample requests.

Expectations

- Ensure scalability for handling multiple requests simultaneously.
- Write clean, maintainable, and modular code.
- Implement comprehensive logging and error handling.