In His Name

E-Commerce platform by FastAPI
as a Task for Lagrange Co.

Objective:
Develop a simple FastAPI application that serves as a backend for a basic e-commerce platform. The application should be able to handle products, users, and orders. Ensure that the application is containerized using Docker and can be easily deployed using Docker Swarm.

Requirements:
Programming Language: Python
Web Framework: FastAPI
Database: Postgres
ORM: SQL Alchemy
Data Validation: Pydantic
Containerization: Docker/Docker Swarm

Task Steps:
Setting Up the Environment:
Initialize a new Python project with a virtual environment.
Set up a new FastAPI project.
Initialize a Dockerfile for the application.

Database Setup:
Set up a Postgres database using Docker.
Use SQL Alchemy to connect your FastAPI application to the Postgres database.

Model Creation:
Design three main models using SQL Alchemy and Pydantic:
User: with fields such as id, username, email, and hashed_password.
Product: with fields like id, name, description, price, and quantity.
Order: linking User and Product, with fields such as id, user_id, product_id, and order_date.

API Endpoints:
Create CRUD (Create, Read, Update, Delete) endpoints for the Product model.
Create endpoints to register a User and to place an Order.
Implement token-based authentication for users. Only authenticated users should be able to place

orders.

Data Validation:
Use Pydantic to validate data coming into your API endpoints. Ensure that the data adheres to the structure of your models.

Containerization and Deployment:
Containerize your FastAPI application using Docker. Make sure the Postgres database is also running in a separate container.
Write a docker-compose.yml file to manage multi-container Docker applications.
Set up a basic Docker Swarm deployment for your application.

Documentation:
Document all API endpoints using FastAPI's built-in documentation feature.
Provide a README file detailing how to set up, run, and deploy the application using Docker and Docker Swarm.

Unit Test and Error Handling:
Implement basic unit tests for your API endpoints.
Add error handling and logging to your application.

Submission:
Please submit the following:
Source code of your FastAPI application.
Dockerfile and docker-compose.yml files.
README file with setup and deployment instructions.
(Optional) Any additional documentation or notes you'd like to share.

Evaluation Criteria:
Functionality and correctness of the code.
Code organization and readability.
Proper use of FastAPI, SQL Alchemy, Pydantic, and Docker/Docker Swarm.
Quality of documentation.