CODTECH INTERNSHIP TASK-03

BLOG BACKEND SYSTEM

NAME : VANTHANA.V

INTERN ID : CTO4DY468

Email : [poojavanthana5@gmail.com](mailto:poojavanthana5@gmail.com)

Title: E-Commerce Backend

**Instructions:**

You are required to **develop a backend system** for an e-commerce platform. The system must include the following core functionalities:

* **User Authentication**  
  Implement secure registration, login, and session management.
* **Product Management**  
  Allow CRUD (Create, Read, Update, Delete) operations for products, including images, descriptions, pricing, and stock management.
* **Order Processing**  
  Handle order creation, order tracking, and status updates.

### **Deliverable:**

A **fully functional RESTful API** using a modern backend framework such as:

* **Node.js** with Express
* **Python** with Django / Flask
* **Java** with Spring Boot
* **Any other modern backend framework**

Your API should:

* Connect with a **relational (e.g., MySQL, PostgreSQL)** or **NoSQL (e.g., MongoDB)** database.
* Follow standard practices for scalability and security.
* Be documented (e.g., using Swagger or Postman).

**Program :**

from flask import Flask, request, jsonify

from flask\_sqlalchemy import SQLAlchemy

from flask\_bcrypt import Bcrypt

from flask\_jwt\_extended import JWTManager, create\_access\_token, jwt\_required, get\_jwt\_identity

from datetime import datetime

app = Flask(\_\_name\_\_)

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///ecommerce.db'

app.config['SECRET\_KEY'] = 'your\_secret\_key'

app.config['JWT\_SECRET\_KEY'] = 'jwt\_secret\_key'

db = SQLAlchemy(app)

bcrypt = Bcrypt(app)

jwt = JWTManager(app)

# Models

class User(db.Model):

id = db.Column(db.Integer, primary\_key=True)

username = db.Column(db.String(80), unique=True, nullable=False)

password = db.Column(db.String(200), nullable=False)

class Product(db.Model):

id = db.Column(db.Integer, primary\_key=True)

name = db.Column(db.String(100), nullable=False)

price = db.Column(db.Float, nullable=False)

stock = db.Column(db.Integer, nullable=False)

class Order(db.Model):

id = db.Column(db.Integer, primary\_key=True)

user\_id = db.Column(db.Integer, nullable=False)

product\_id = db.Column(db.Integer, nullable=False)

quantity = db.Column(db.Integer, nullable=False)

timestamp = db.Column(db.DateTime, default=datetime.utcnow)

# Routes

@app.route('/register', methods=['POST'])

def register():

data = request.get\_json()

hashed\_pw = bcrypt.generate\_password\_hash(data['password']).decode('utf-8')

user = User(username=data['username'], password=hashed\_pw)

db.session.add(user)

db.session.commit()

return jsonify({'message': 'User registered successfully!'})

@app.route('/login', methods=['POST'])

def login():

data = request.get\_json()

user = User.query.filter\_by(username=data['username']).first()

if user and bcrypt.check\_password\_hash(user.password, data['password']):

token = create\_access\_token(identity=user.id)

return jsonify({'token': token})

return jsonify({'error': 'Invalid credentials'}), 401

@app.route('/products', methods=['POST'])

@jwt\_required()

def add\_product():

data = request.get\_json()

product = Product(name=data['name'], price=data['price'], stock=data['stock'])

db.session.add(product)

db.session.commit()

return jsonify({'message': 'Product added'})

@app.route('/products', methods=['GET'])

def get\_products():

products = Product.query.all()

result = []

for p in products:

result.append({'id': p.id, 'name': p.name, 'price': p.price, 'stock': p.stock})

return jsonify(result)

@app.route('/order', methods=['POST'])

@jwt\_required()

def create\_order():

data = request.get\_json()

user\_id = get\_jwt\_identity()

order = Order(user\_id=user\_id, product\_id=data['product\_id'], quantity=data['quantity'])

db.session.add(order)

db.session.commit()

return jsonify({'message': 'Order placed successfully'})

if \_\_name\_\_ == '\_\_main\_\_':

with app.app\_context():

db.create\_all()

app.run(debug=True)

**Sample output:**

Request

POST /products

Authorization: Bearer <JWT\_TOKEN>

Content-Type: application/json

{

"name": "Laptop",

"price": 1200.0,

"stock": 10

}

Reponse:

HTTP 200 OK

{

"message": "Product added"

}

Placed Order :

POST /order

Authorization: Bearer <JWT\_TOKEN>

Content-Type: application/json

{

"product\_id": 1,

"quantity": 2

}

Response:

HTTP 200 OK

{

"message": "Order placed successfully"

}