Product Management Service

Purpose

A Microservice that is essential for the eCommerce ecosystem to perform Create, Read, Update and Delete (CRUD) operations on Product Object/Model.
The Product Model is supposed to be a representation of an actual product sold on an eCommerce website like an iPhone, Shirt, Book, etc.

Existing setup

A docker-compose file is already created that will create a Postgres database product-management-db
Open a separate PowerShell or Command Prompt and navigate to the below location

C:\workspace\flask-microservices-training\day11\product-management-service

Execute the below command to start Postgres Database

docker-compose up

Business Requirement

- Define a requirements.txt file that contains all the required libraries
- Define a virtual environment where the dependencies will be installed and used by the flask application to run
- Define a Product Model that container below the fields

Field Name	Description	Mandatory	Constraint
id	Unique identifier auto-incremented by ORM/DB		
name	Name of the product	Yes	
description	Description of the product	No	Min length 20 Max length 300
price	Numerical price of the product. Ex - 100, 99, etc.	Yes	Min value 1 Max value 99999
currency	Currency for the above price. Ex - ₹, $, €	Yes	One of ₹, $, €
stock	Numerical representation of available stock. Ex- 5, 11, etc.	Yes	Min 1 Max 999

active	Boolean flag to determine whether a product is active or not.	Yes	

- Provide Schema validation for the above model based on the constraints specified
- Create a Flask Rest application that needs to expose 5 rest endpoints
    - Create a Product - POST /api/products
    - Fetch all the products- GET /api/products
    - Delete all the products - DELETE /api/products
    - Fetch details of a single product - GET /api/products/id
    - Update product details - PUT /api/products/id
    - Delete a product - DELETE /api/products/id
- Prepare the flask config to handle environment-specific configuration and connect to product-management-db
- Handle below custom exception
    - Product Not Found
- Application should run on port 5001 by default

Bonus Requirements

- Update the Create product API to prevent entering duplicate product if the name of the product matches with an existing product.
- Update the Fetch all products API to add an optional request param to filter products based on the active flag.
    - /api/products - will fetch all products
    - /api/products?active=true - will fetch all product for which active flag is true