from fastapi import FastAPI

from pydantic import BaseModel

from typing import List, Optional

from sentence\_transformers import SentenceTransformer, util

import torch

# Initialize FastAPI app

app = FastAPI()

# Load sentence transformer model

model = SentenceTransformer('all-MiniLM-L6-v2')

# Default product data

default\_products = [

{

"id": 1,

"name": "Nike Air Max 90",

"description": "Classic black sneakers with air cushioning",

"category": "Footwear",

"tags": ["sneakers", "black", "nike", "shoes"]

},

{

"id": 2,

"name": "Adidas Ultraboost White",

"description": "Comfortable white running shoes",

"category": "Footwear",

"tags": ["sneakers", "white", "adidas", "running"]

},

{

"id": 3,

"name": "Puma RS-X Black",

"description": "Stylish black running sneakers with modern design",

"category": "Footwear",

"tags": ["puma", "black", "sneakers", "running"]

},

{

"id": 4,

"name": "Converse Chuck Taylor",

"description": "Classic white high-top sneakers",

"category": "Footwear",

"tags": ["white", "sneakers", "converse", "casual"]

},

{

"id": 5,

"name": "New Balance 574",

"description": "Gray suede sneakers with retro design",

"category": "Footwear",

"tags": ["gray", "sneakers", "new balance", "retro"]

}

]

# Pydantic model for a product

class Product(BaseModel):

id: int

name: str

description: str

category: str

tags: List[str]

# Request schema

class QueryRequest(BaseModel):

query: str

top\_k: Optional[int] = 3

products: Optional[List[Product]] = None

# Recommendation function

def recommend\_products(query: str, products: List[dict], top\_k: int = 3):

product\_texts = [f"{p['name']} {p['description']}" for p in products]

product\_embeddings = model.encode(product\_texts, convert\_to\_tensor=True)

query\_embedding = model.encode(query, convert\_to\_tensor=True)

scores = util.cos\_sim(query\_embedding, product\_embeddings)[0]

top\_results = torch.topk(scores, k=top\_k)

recommended = [products[idx] for idx in top\_results.indices]

return recommended

# Endpoint

@app.post("/recommend")

def get\_recommendations(req: QueryRequest):

products\_to\_use = req.products if req.products is not None else default\_products

results = recommend\_products(req.query, products\_to\_use, req.top\_k)

return {"recommendations": results}