```
# streamlit_app.py
import streamlit as st
import pandas as pd
import numpy as np
import requests
from PIL import Image
from io import BytesIO
import faiss
import torch
import torchvision.transforms as transforms
from torchvision import models
import os

# ---- CONFIG ----
st.set_page_config(page_title="Fashion Visual Search & Outfit Recommender", layout="wide")

# ---- LOAD DATA ----
dresses_df = pd.read_csv("dresses_bd_processed_data.csv")
jeans_df = pd.read_csv("jeans_bd_processed_data.csv")
data_df = pd.concat([dresses_df, jeans_df], ignore_index=True)

# ---- FEATURE IMAGE EMBEDDINGS ----
@st.cache_resource
def load_model():
    model = models.resnet50(pretrained=True)
    model.eval()
    model = torch.nn.Sequential(*(list(model.children())[:-1]))
    return model
```

```
model = load_model()

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

@st.cache_data
def get_embedding(image_url):
    try:
        response = requests.get(image_url)
        img = Image.open(BytesIO(response.content)).convert("RGB")
        img_t = transform(img).unsqueeze(0)
        with torch.no_grad():
            embedding = model(img_t).squeeze().numpy()
        return embedding / np.linalg.norm(embedding)
    except:
        return None

@st.cache_data
def generate_index(data):
    vectors = []
    valid_idx = []
    for i, row in data.iterrows():
        emb = get_embedding(row['feature_image_s3'])
        if emb is not None:
            vectors.append(emb)
```

```
            valid_idx.append(i)
    vectors = np.array(vectors).astype('float32')
    index = faiss.IndexFlatL2(vectors.shape[1])
    index.add(vectors)
    return index, data.iloc[valid_idx].reset_index(drop=True), vectors

index, filtered_df, image_vectors = generate_index(data_df)

# ---- STREAMLIT APP ----
st.title("✨ Fashion Visual Search & Outfit Recommender")

uploaded = st.file_uploader("Upload a fashion image", type=["jpg", "jpeg", "png"])

if uploaded is not None:
    input_image = Image.open(uploaded).convert("RGB")
    st.image(input_image, caption="Uploaded Image", use_column_width=True)

    img_tensor = transform(input_image).unsqueeze(0)
    with torch.no_grad():
        input_vec = model(img_tensor).squeeze().numpy()
    input_vec = input_vec / np.linalg.norm(input_vec)

    D, I = index.search(np.array([input_vec]).astype('float32'), 6)

    st.subheader("Exact & Visually Similar Matches")
    cols = st.columns(6)
    for idx, col in zip(I[0], cols):
        row = filtered_df.iloc[idx]
        try:
```

```
            response = requests.get(row['feature_image_s3'])
            col.image(Image.open(BytesIO(response.content)), use_column_width=True)
            col.caption(row['product_name'])
        except:
            continue

    st.subheader("🧥 Intelligent Outfit Suggestions")
    base_category = filtered_df.iloc[I[0][0]]['category_id']
    complement_items = filtered_df[filtered_df['category_id'] !=
base_category].sample(5)
    cols = st.columns(5)
    for _, row in complement_items.iterrows():
        try:
            response = requests.get(row['feature_image_s3'])
            cols[_ % 5].image(Image.open(BytesIO(response.content)),
use_column_width=True)
            cols[_ % 5].caption(f"{row['product_name']} | {row['brand']}")
        except:
            continue

    st.subheader("📊 Personalized Picks (based on similar patterns)")
    same_brand = filtered_df[filtered_df['brand'] ==
filtered_df.iloc[I[0][0]]['brand']].sample(3)
    cols = st.columns(3)
    for _, row in same_brand.iterrows():
        try:
            response = requests.get(row['feature_image_s3'])
            cols[_ % 3].image(Image.open(BytesIO(response.content)),
use_column_width=True)
            cols[_ % 3].caption(f"{row['product_name']} | {row['brand']}")
```

```
except:
    continue
```