AI Outfit Styling Search – Project Documentation

Project Overview

AI Outfit Styling Search is a web application built using **Streamlit** and **Hugging Face's CLIP model** that allows users to:

1. Upload an outfit image and find visually similar styling inspirations.
2. Enter a textual description to retrieve outfits matching the text query.
3. Optionally filter results by categories (e.g., coat, sweater, denim).

It uses **image embeddings** and **FAISS similarity search** to provide fast, accurate, and style-aware results. Additionally, it includes a **CLIP-based outfit detection filter** to avoid processing irrelevant images (e.g., logos, backgrounds, non-fashion items).

Key Features

1. **Image Search**
    a. Upload a fashion image (coat, sweater, full outfit)
    b. AI finds the most visually similar outfits in the dataset
    c. Optional category refinement
2. **Text Search**
    a. Input a textual query describing style or outfit
    b. Retrieves images with visual similarity to the description
    c. Optional category refinement
3. **Outfit Detection**
    a. Uses CLIP to detect whether an uploaded image is a valid outfit
    b. Avoids showing irrelevant results for non-fashion images
4. **Caption-based Filtering**
    a. Uses BLIP-generated captions for dataset images
    b. Ensures category-specific queries (e.g., "men coat") don't return unrelated items like jackets
5. **Modern UI/UX**
    a. Gradient background
    b. Styled buttons and headers

c. Image grid with captions
d. Responsive layout using columns and tabs

Project Architecture

```
User Upload / Text Query
        |
        v
 +--------------------+
 |   Outfit Filter    |  <- CLIP-based detection against style
prompts
 +--------------------+
        |
        v
 +--------------------+
 |   CLIP Embeddings  |  <- Images or text are converted into
embeddings
 +--------------------+
        |
        v
 +--------------------+
 |      FAISS         |  <- Cosine similarity search for nearest
embeddings
 +--------------------+
        |
        v
 +--------------------+
 | Caption Filtering  |  <- Optional: filter by keywords from BLIP
captions
 +--------------------+
        |
        v
 +--------------------+
 |  Streamlit UI      |  <- Results displayed in grid with captions
 +--------------------+
```

Dependencies

- **Python 3.9+**
- **PyTorch**: For model inference (`torch`, `torchvision`)
- **Transformers**: Hugging Face CLIP and BLIP models
- **Pillow**: Image processing
- **FAISS**: Fast nearest neighbor search for embeddings
- **Streamlit**: Frontend web application

Installation:

```
pip install torch torchvision transformers pillow faiss-cpu
streamlit
```

Use `faiss-gpu` if you have a CUDA-enabled GPU for faster embedding searches.

Dataset Requirements

- **Images folder**: Contains all fashion images to search from.
- **embeddings.pkl**: Precomputed CLIP image embeddings of the dataset.
- **captions.pkl**: BLIP-generated captions describing each image.

Dataset structure example:

```
project/
│
├── images/            # Dataset images (e.g., 100 fashion outfits)
├── embeddings.pkl     # CLIP embeddings of images
├── captions.pkl       # BLIP captions for images
├── app_final_styled.py # Streamlit app
```

How It Works

1. Embeddings Preparation

- Each dataset image is encoded using the **CLIP image encoder** to generate a fixed-size vector (`embedding`).
- BLIP is used to generate **text captions** for each image.

- Embeddings are normalized and stored in **FAISS** for fast cosine similarity search.

2. Outfit Detection

- Predefined style prompts (e.g., “photo of a coat outfit”) are embedded with CLIP text encoder.
- Uploaded images are encoded and **compared against these prompts** using cosine similarity.
- Only images with similarity above a threshold (0.22) are treated as valid fashion outfits.

3. Search Pipeline

- **Image Search**:
  - Uploaded image → CLIP embedding → FAISS nearest neighbors → top 60 results
  - Optional caption-based category filtering
  - Return top 9 results
- **Text Search**:
  - Text query → CLIP text embedding → FAISS nearest neighbors → top 60 results
  - Optional caption-based category filtering
  - Return top 9 results
- **Caption Filtering**:
  - Filter search results based on keywords present in BLIP-generated captions to improve relevance.

4. Streamlit UI

- Tab-based interface: **Image Search** and **Text Search**
- Responsive columns: image preview + search options
- Grid layout: results displayed in 3×3 card style
- Styled with gradient background, button colors, and custom fonts

Usage Instructions

1. Clone the project repository and ensure dataset & embeddings are available.
2. Install dependencies:

```
pip install torch torchvision transformers pillow faiss-cpu
streamlit
```

3. Run the Streamlit app:

```
streamlit run app_final_styled.py
```

4. Use the **Image Search tab**:
   a. Upload an outfit image
   b. Optional: refine results by category
   c. Click "Find Similar Styles"
5. Use the **Text Search tab**:
   a. Enter a text query (e.g., "men coat winter")
   b. Optional: add category filters
   c. View matching outfits

Customization Options

- **Adjust Outfit Detection Threshold**:
  Modify `threshold` in `is_outfit()` to make detection stricter or more lenient.
- **Add More Style Prompts**:
  Enhance CLIP-based outfit detection by adding new prompts like "vintage outfit", "party wear", etc.
- **Change Grid Layout**:
  Modify the number of columns or top results returned in the Streamlit UI.
- **Add Infinite Scroll / Hover Effects**:
  Future UI improvements can mimic Pinterest-style browsing.

Key Benefits

- AI-powered **visual search** for fashion inspiration
- Works with **small datasets** (~100 images) or can scale to larger datasets
- **Text + image search** allows flexibility
- **Safe filtering** prevents irrelevant images from returning results
- **Beautiful, modern UI** suitable for demos and portfolios

Future Improvements

1. **Scale dataset** → integrate 1k+ images for richer recommendations.
2. **Multi-category search** → support partial outfit search (e.g., top only).
3. **Real-time fine-tuning** → improve results for niche styles like “streetwear” or “formal”.
4. **Portfolio-ready UI** → add hover captions, shadows, and infinite scroll.
5. **Mobile responsiveness** → optimize Streamlit layout for smaller screens.