👗 Find It or Create It ✂️ — AI-Powered Fashion Search & Custom Design Platform

Combining Vision Transformers, LLM Workflows, and Intelligent Data Structuring

🌟 Project Overview

Find It or Create It is an AI-driven fashion web platform designed to help users **find or create Indian traditional outfits** using **text or image-based search**.
If a user doesn't find the perfect outfit, they can directly **book a designer** to stitch a custom design — all within one integrated system.

This project showcases technical and creative expertise across:

- Vision Transformer–based AI search
- LLM-style dataset processing and mapping
- Backend automation using Flask
- FAISS similarity search
- Containerized deployment on Docker and Hugging Face Spaces

This project is part of my **portfolio showcase for AI, LLM, and Vision roles**.

🎯 Problem & Motivation

In India, traditional outfits such as **sarees, kurtas, and lehengas** vary by region and design.
People often want to find similar outfits from online stores but struggle because **image-based searches fail** for Indian-specific styles.

To solve this:

- The system allows **text + image-based product search**.
- When results don't satisfy, users can **book a designer** for personalized stitching.
- It uses **AI embeddings, dataset mapping, and automation** for a realistic workflow simulation.

🧩 Data Structuring and Mapping

A structured dataset was created from scratch to help AI understand and retrieve results efficiently.
Each product record includes:

- Unique ID (auto-generated, e.g., `LEH0001`, `KUR0002`)
- Product name and description
- Image path
- Category (Saree, Kurta, Lehenga, etc.)
- Source links (Amazon, Flipkart, Google Images)
- Optional stitching/designer info

All data was stored in a single **JSON manifest file** — acting as the main dataset and connection between AI embeddings and the search interface.

🛡️ Data Security and Privacy

We ensured all data and embeddings remain protected. Secret variables were created in Hugging Face to securely store the dataset and embedding keys, preventing exposure or copying in the public domain.

🔢 Unique ID Mapping

Each record's unique ID acts as the **link between multiple layers**:

- JSON dataset entry
- Text embedding generated
- Image embedding generated
- FAISS index position

This ID-based mapping ensures traceability — for example, if FAISS returns index `108`, the corresponding product in JSON can be fetched directly using its mapped ID.

🧠 AI Model: Vision Transformer (SigLIP)

The core AI engine is **SigLIP**, a **Vision Transformer** model from Google.
It aligns both images and text in the same vector (embedding) space, enabling cross-modal similarity matching.

Workflow Overview:

- Images → passed through SigLIP Vision Encoder → get 512-dim image embeddings
- Text (Search queries) → passed through SigLIP Text Encoder → get 512-dim text embeddings
- Embeddings are normalized and saved as `.npy` files for fast FAISS indexing

This forms a **multimodal retrieval system** similar to the architecture used in **Large Multimodal Models (LMMs)** like CLIP and Gemini.

⚙️ FAISS Indexing & Retrieval

All embeddings are indexed using **FAISS (Facebook AI Similarity Search)** for high-speed nearest-neighbour lookup.

Search Flow:

When user uploads an image:

1. The image is embedded using SigLIP.
2. FAISS compares it with stored image embeddings.
3. Top N similar embeddings are retrieved.
4. Matching product info is fetched from the JSON dataset.

When user enters a text query:

1. Text input is embedded using SigLIP Text Encoder.
2. FAISS searches across text embeddings.
3. Results are mapped via unique IDs to the dataset.

This structure behaves like **Retrieval-Augmented Generation (RAG)** — but returns **fashion items instead of text outputs**.

🧮 Data Flow Summary

Here's how each layer connects (add image of system diagram if needed):

```
User Input (Text/Image)
        ↓
Embedding Generation (SigLIP Vision Transformer)
        ↓
FAISS Similarity Search
        ↓
Mapping via Unique IDs
        ↓
Retrieve Data from JSON
        ↓
Display Results / Book Designer
```

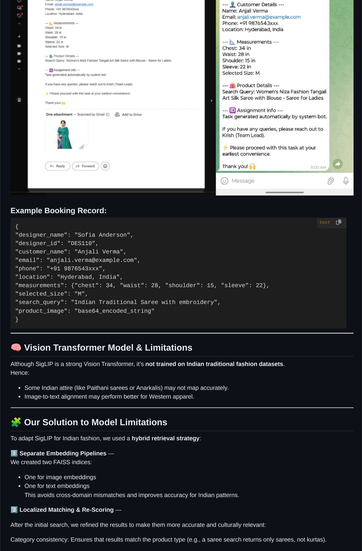
🎨 Designer Booking & Automation Flow

If a user doesn't find an ideal match, they can switch to **Create It mode** for custom stitching.

Process:

1. User submits their preferences and body measurements.
2. The backend automatically assigns an available designer from a JSON dataset.
3. A booking record is generated containing both **product ID** and **designer ID**.
4. Automated notifications are sent via **Gmail** and **Telegram**.

Gmail confirmation and **Telegram alert screenshot**



Example Booking Record:

```
{
"designer_name": "Sofia Anderson",
"designer_id": "DES110",
"customer_name": "Anjali Verma",
"email": "anjali.verma@example.com",
"phone": "+91 9876543xxx",
"location": "Hyderabad, India",
"measurements": {"chest": 34, "waist": 28, "shoulder": 15, "sleeve": 22},
"selected_size": "M",
"search_query": "Indian Traditional Saree with embroidery",
"product_image": "base64_encoded_string"
}
```

🧠 Vision Transformer Model & Limitations

Although SigLIP is a strong Vision Transformer, it's **not trained on Indian traditional fashion datasets**.
Hence:

- Some Indian attire (like Paithani sarees or Anarkalis) may not map accurately.
- Image-to-text alignment may perform better for Western apparel.

🌱 Our Solution to Model Limitations

To adapt SigLIP for Indian fashion, we used a **hybrid retrieval strategy**:

1️⃣ Separate Embedding Pipelines —
We created two FAISS indices:

- One for image embeddings
- One for text embeddings
  This avoids cross-domain mismatches and improves accuracy for Indian patterns.

2️⃣ Localized Matching & Re-Scoring —

After the initial search, we refined the results to make them more accurate and culturally relevant:

Category consistency: Ensures that results match the product type (e.g., a saree search returns only sarees, not kurtas).

Color & keyword matching: Prioritizes results that closely match the color or key terms in the query.

Trusted source preference: Gives higher ranking to products from reliable platforms like Amazon or Flipkart.

This step ensures that the top results are meaningful, accurate, and suitable for Indian traditional fashion.

🔍 Generating Search Queries Using Ollama (Mistral Model)

To improve SEO and result accuracy, we used **Ollama (local Mistral LLM)** to regenerate product titles into **natural, search-optimized queries**.
These rewritten queries make it easier to match similar items on Google, Amazon, and Flipkart.

Example:
Input Title: *"Women's Silk Saree with Blouse Piece"*
Output Query: *"Elegant Silk Saree for Women with Blouse – Traditional Saree"*

This step ensures our AI search and online search alignment remain consistent and human-readable.

🐳 Deployment on Docker + Hugging Face Spaces

To make the application publicly accessible and easy to test, the entire system (Flask backend, FAISS index, static UI, and model) was deployed in a **Docker container**.

Why Docker?

- Ensures environment consistency
- Prevents dependency conflicts
- Makes it easy to run anywhere

The **final Docker image** was pushed and deployed on **Hugging Face Spaces** using custom Docker runtime.

Why Hugging Face Spaces?

- Simple model hosting with CPU/GPU options
- Great for recruiters to test the live app directly
- Secure and scalable deployment

🧰 Tools and Technologies

Category	Tools Used	Description
AI & Embeddings	SigLIP (Vision Transformer)	Image-text alignment for multimodal retrieval
Vector Search	FAISS	Fast similarity search across embeddings
Backend	Flask	RESTful API and logic layer
Data Storage	JSON, NumPy	Lightweight structured mapping
Frontend	HTML, CSS, JS	Simple and responsive web interface
Optimization	Pillow, Regex	Image resizing & SVG cleanup
Deployment	Docker, Hugging Face	End-to-end containerized hosting
LLM Integration	Ollama (Mistral)	Query rewriting for better text search

---

💡 Future Improvements and Scalability

The current prototype uses around **10,000 image-text pairs** — enough for testing, but performance and precision can improve by:

- Adding larger, diverse datasets
- Expanding Indian fashion-specific categories
- Regenerating embeddings with enriched metadata

Monetization Potential

In the future, each product result (Amazon/Flipkart link) can include **affiliate tracking**.
If a user clicks and purchases, the platform earns a **commission-based reward** — creating a sustainable revenue model without altering UX.

👨‍💼 For Recruiters

This project is a **portfolio showcase**, not a commercial product.
It highlights:

- Hands-on experience in **Vision Transformers and LLM pipelines**
- Building multimodal AI workflows (image + text)
- Handling real-world data structuring and deployment
- Integrating business logic into AI systems

📧 **Email:** mr.bhanuprakash@gmail.com
🌐 **Project Link / Portfolio Showcase:** Find It or Create It - AI Fashion Search

---