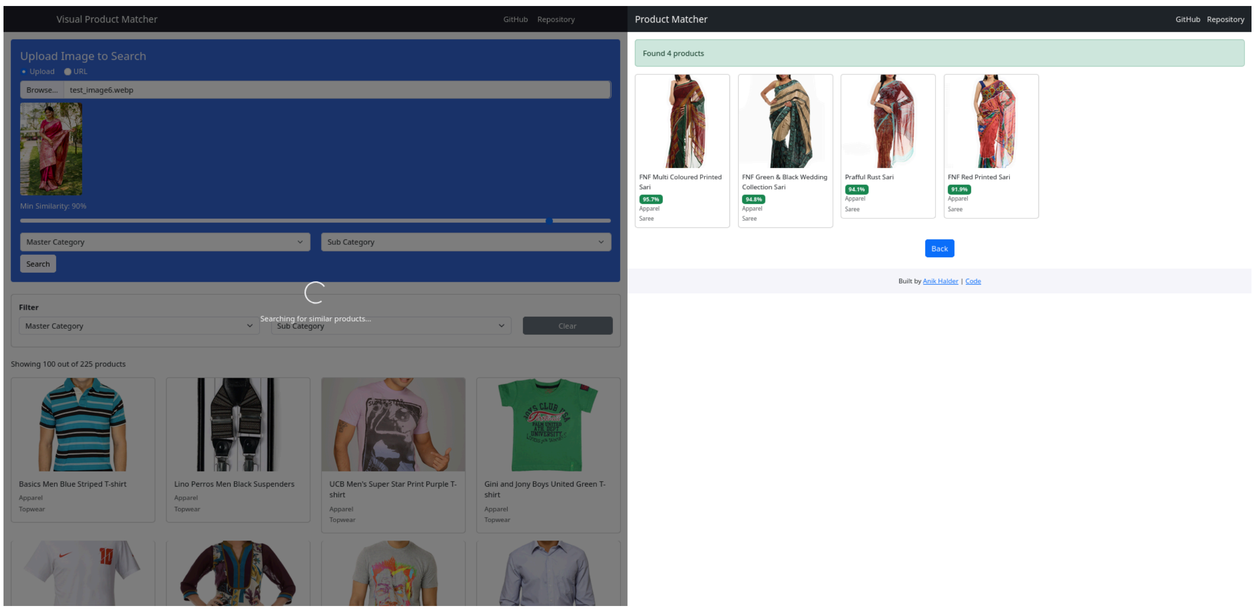
PROJECT REPORT
Submission by: ANIK HALDER 24MCA0251

Visual Product Matcher Build is a web-based fashion product catalogue and search application built with Flask, FastAPI, Bootstrap 5 and MongoDB Atlas. The application employs a custom built, fine-tuned and hosted hybrid machine learning architecture combining ResNet50 and OpenAI-CLIP models, outperforming traditional standalone CLIP implementations by more than 10% with 90% visual similarity and 95% semantic accuracy.

Link to deployed Application : https://visual-product-matcher-1hkn.onrender.com
Link to Github Repository: https://github.com/Anikrage/Visual_Product_Matcher

Sample Screenshot:



Model Architecture:
Unlike conventional single-model approaches, the custom built model uses:

1. Three-stage ranking algorithm that first identifies the visual matches using ResNet50 and CLIP image embeddings using cosine similarity.
2. Followed by a semantic filtering using CLIP text embeddings to eliminate gender/category confusions.
3. and then finally combining the scores to create a hybrid score and ranking the results.

This approach addresses the common failures in simpler systems that often mismatch men's and women's products.

Full Stack Implementation:
The frontend uses Flask templating with Bootstrap 5 and vanilla javascript for responsive UI, while the backend FastAPI service handles the ML inference. The database is hosted online with MongoDB Atlas that stores product metadata, pre-computed embeddings, links to product image hosted via fast CDN using Cloudinary, with indexed queries for fast filtering.

Database:
The application uses a cloud deployed instance of MongoDB Atlas that stores the database as a non relational document format.
Sample Document Structure:

```json
{
   "_id": "68e6ae54a9146459b0dece59",
   "product_id": 12947,
   "name": "Basics Men Blue Striped T-shirt",
   "master_category": "Apparel",
   "sub_category": "Topwear",
   "article_type": "Tshirts",
   "gender": "Men",
   "base_colour": "Blue",
   "season": "Fall",
   "year": 2011,
   "usage": "Casual",
   "image_url":
"https://res.cloudinary.com/driwu0lv4/image/upload/v1760032271/product_matcher/products/12947.jpg",
   "cloudinary_public_id": "product_matcher/products/12947",
   "embedding": [
       0.001910855178721249,
       0.008678504265844822,
       0.007128174416720867,
       0.0014160072896629572,
       0.002942504594102502,
       ...
   ],
   "clip_embedding": [
       -0.006649017333984375,
       0.03448486328125,
       -0.0270233154296875,
       0.022735595703125,
       -0.038787841796875,
       -0.01126861572265625,
       -0.0301971435546875,
       ...
   ],
   "text_embedding": [
       0.00408172607421875,
       -0.002872467041015625,
       -0.047393798828125,
       0.0382080078125,
       -0.06884765625,
       0.01515960693359375,
       -0.0182037353515625,
       ...
   ]
}
```

Field Descriptions

Field	Type	Description
_id	ObjectId	MongoDB unique identifier
product_id	Integer	Product identifier for API matching
name	String	Product display name
master_category	String	Top-level category (e.g., "Apparel", "Accessories")
sub_category	String	Second-level category (e.g., "Topwear", "Bottomwear")
article_type	String	Specific product type (e.g., "Tshirts", "Jeans", "Shoes")
gender	String	Target gender (e.g., "Men", "Women", "Unisex")
base_colour	String	Primary color of the product
season	String	Seasonal category (e.g., "Fall", "Summer", "Winter")
year	Integer	Product year or collection year
usage	String	Use case (e.g., "Casual", "Formal", "Sports")
image_url	String	Cloudinary hosted image URL
cloudinary_public_id	String	Cloudinary asset identifier
embedding	Array[Float]	Image embedding vector for similarity matching
clip_embedding	Array[Float]	CLIP model embedding for multimodal search
text_embedding	Array[Float]	Text-based embedding for semantic search

Embedding Vectors

- embedding: Visual features extracted from product images for image-to-image similarity
- clip_embedding: CLIP (Contrastive Language-Image Pre-training) vectors for cross-modal matching
- text_embedding: Text-based features for semantic text search

All embedding arrays are truncated with "..." for documentation purposes. Actual vectors contain 512+ dimensions.

Optimizations:

The application follows a separation of concern approach, Flask for presentation, FastAPI for ML inference and MongoDB for persistence, enabling independent scaling. Further optimizations are done via :

1. pre-computed embeddings eliminating real-time inference
2. Numpy vectorized similarity calculations
3. heap-based selection for O(n log k) complexity
4. lazy loading with async pagination
5. CDN-cached resources

Cloud Deployment:

1. The custom built model is hosted via HuggingFace Spaces that provides an API for connectivity and inference
2. The frontend and backend is hosted via Render with automatic configuration for continuous integration and deployment.

This approach results in a production system that demonstrates complete software engineering from data preprocessing and model training to API deployment,frontend integration and cloud deployment, achieving 1-2 second response times.