NAME: - AMIT GUNJAN JHA

REGISTRATION NUMBER: - 22BSA10090

Visual Product Matcher Build

INDEX

Executive Summary

The **Visual Product Matcher Build** project was developed as part of a software engineering technical assessment, focusing on the creation of an intelligent, web-based system that enables users to find visually similar products based on an uploaded image. The primary goal of this project is to demonstrate full-stack engineering skills in designing an AI-assisted front-end experience for e-commerce-style product discovery.

The solution allows users to upload an image or provide an image URL, after which the system analyzes the input to identify visual features and retrieve matching products from a predefined catalog of 50+ items. Each product entry includes metadata such as name, category, price, and image link, ensuring a complete and realistic product browsing experience. Users can filter and view results by **similarity score**, offering a personalized and dynamic search interface.

The project is entirely **frontend-based**, developed using **HTML, CSS, and JavaScript**, with AI integration handled through **YOLO** for object detection and **CLIP embeddings** for similarity computation — executed locally in the browser using **ONNX Runtime** and **Transformers.js**. The design follows a clean, modular folder hierarchy emphasizing maintainability, responsive layout, and smooth user interactions, complete with progress indicators and error handling.

In essence, the **Visual Product Matcher Build** represents the integration of artificial intelligence with practical web engineering to solve real-world e-commerce challenges. It demonstrates strong proficiency in UI/UX design, model integration, and structured code organization, showcasing a modern approach to building AI-enabled web applications for intelligent product discovery.

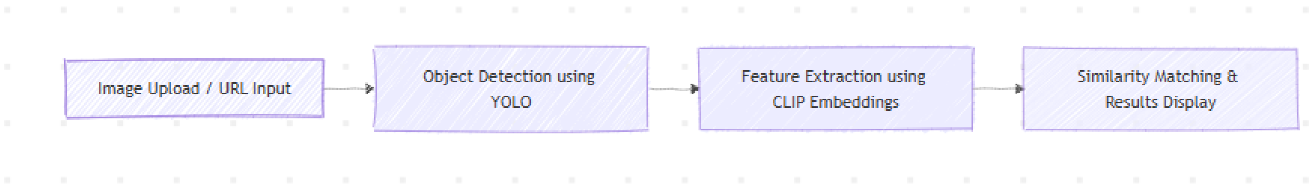


Image Upload / URL Input
Object Detection using YOLO
Feature Extraction using CLIP Embeddings
Similarity Matching & Results Display

Background & Context

In modern e-commerce, product discovery is still largely dependent on **text-based search**, which often fails to capture visual intent — for example, when a user wants to find “something that looks like this.” Traditional keyword searches struggle to interpret style, texture, or color similarity. This leads to missed opportunities, longer search times, and user frustration.

The **Visual Product Matcher Build** addresses this challenge by introducing an **AI-powered visual search system** that enables users to upload an image or paste an image URL to instantly discover visually similar products. By integrating **YOLO** for object detection and **CLIP embeddings** for similarity comparison, the system bridges the gap between human perception and machine understanding.

The web application is designed as a **frontend-only solution**, emphasizing simplicity, accessibility, and real-time interaction. It simulates how large-scale retail platforms like Amazon or Myntra use visual similarity to enhance product recommendations. Additionally, by leveraging **browser-based AI inference (ONNX Runtime + Transformers.js)**, the project demonstrates the feasibility of deploying machine learning features without backend infrastructure.

This project showcases the potential of visual intelligence in retail discovery—making searches more intuitive, personalized, and visually guided. It reflects a practical step toward next-generation **AI-driven shopping experiences**, where users engage directly through images rather than words.

System Architecture & Design

The **Visual Product Matcher Build** follows a clean, modular, and scalable **frontend-only architecture** designed for smooth integration of AI models within the browser. The system is structured around three primary layers — **User Interface**, **AI Processing Layer**, and **Data Layer** — each responsible for a distinct aspect of the application's functionality.

1. User Interface (UI Layer)

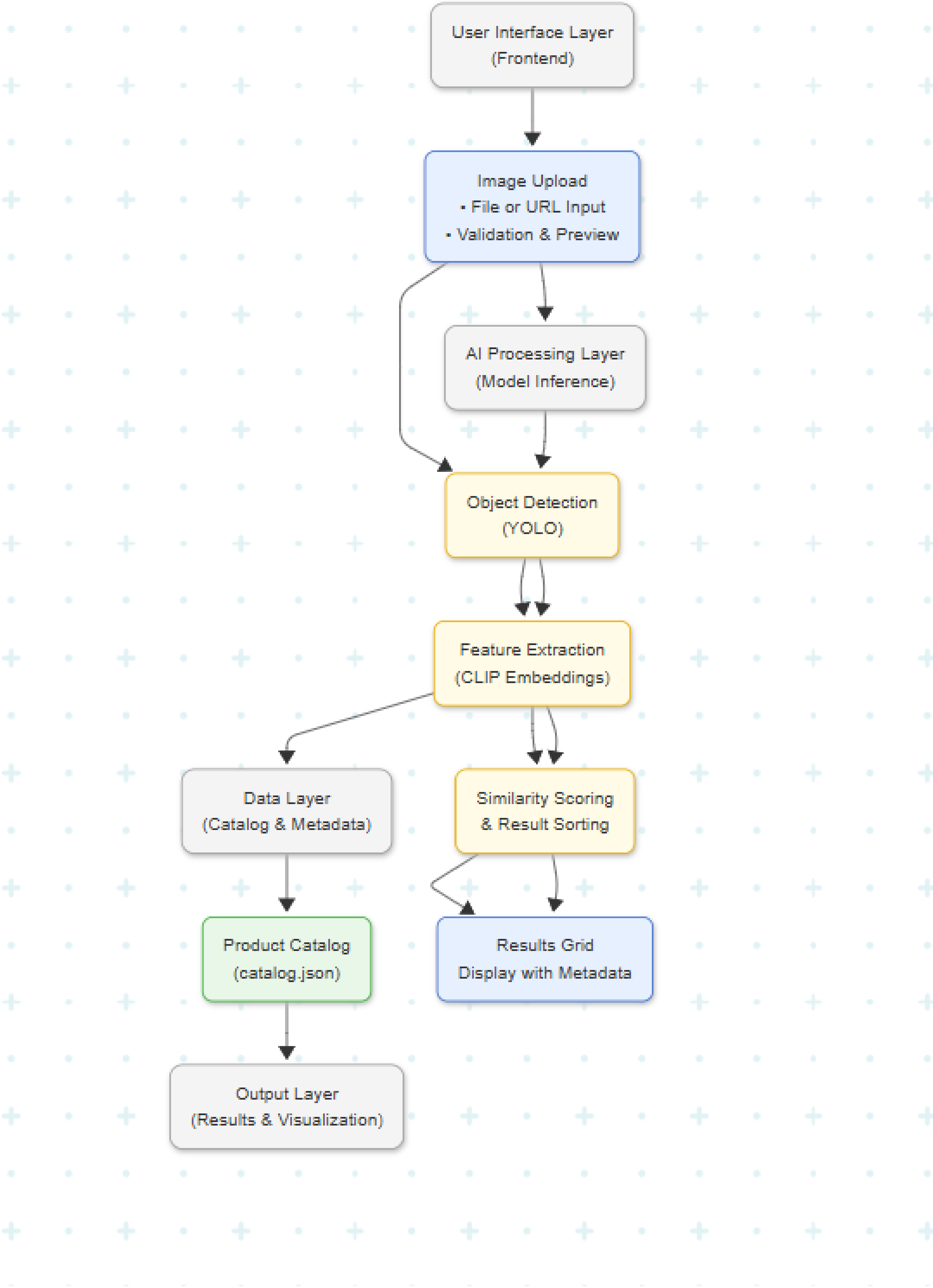
- Built with **HTML, CSS, and JavaScript**.
- Handles user interactions such as image upload, preview, URL input, and result visualization.
- Includes responsive design to ensure accessibility across devices (desktop and mobile).
- Displays similarity results using dynamic grids with product metadata and similarity scores.

2. AI Processing Layer (ML Layer)

- Powered by **YOLO (Object Detection)** to locate key objects within the uploaded image.
- Uses **CLIP Embeddings (via Transformers.js)** to generate high-dimensional vector representations of image content.
- Embeddings are compared with catalog embeddings to identify visually similar products.
- Runs entirely client-side using **ONNX Runtime Web**, ensuring no server dependency.

3. Data Layer (Product Catalog)

- Contains a **JSON-based dataset** with 50+ product entries.
- Each entry includes metadata such as ID, title, image URL, price, tags, and vendor.
- Used for similarity computation and rendering matching results dynamically on the frontend.


User Interface Layer
(Frontend)
Image Upload
• File or URL Input
• Validation & Preview
AI Processing Layer
(Model Inference)
Object Detection
(YOLO)
Feature Extraction
(CLIP Embeddings)
Data Layer
(Catalog & Metadata)
Similarity Scoring
& Result Sorting
Product Catalog
(catalog.json)
Results Grid
Display with Metadata
Output Layer
(Results & Visualization)

Feature Specification and User Experience

The **Visual Product Matcher Build** provides an intuitive, AI-assisted user experience designed for simplicity, speed, and interactivity. Each feature is crafted to simulate a modern e-commerce product discovery interface powered by image intelligence.

Core Features

1. Image Upload & Input Options

- Users can either upload an image file or paste an online image URL.
- The system validates file type and displays an instant preview.
- A drag-and-drop zone improves usability on desktop and mobile.

2. Object Detection (YOLO Integration)

- Once an image is uploaded, the YOLO model identifies and highlights objects within it.
- Detection results are cropped or extracted for feature encoding.

3. Feature Extraction (CLIP Embeddings)

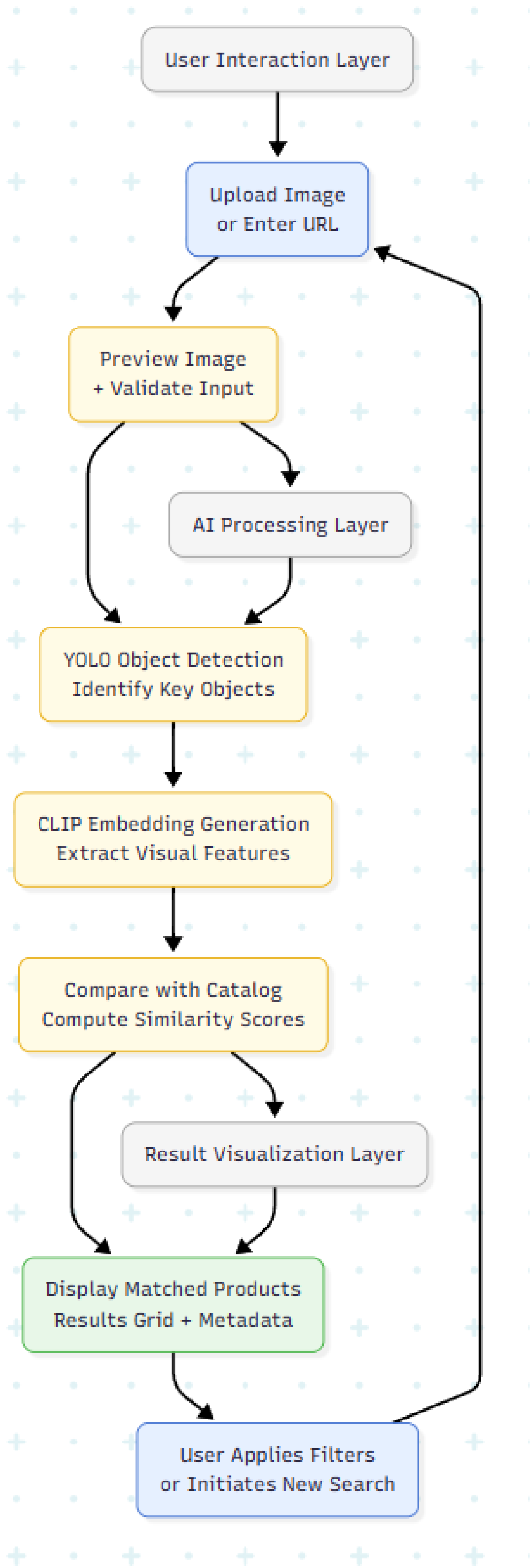
- The CLIP model transforms the detected object into an embedding vector.
- Each product in the catalog also has its precomputed embedding for comparison.
- Similarity scores are computed using cosine similarity.

4. Similarity Matching

- The system compares the query embedding with the product catalog embeddings.
- Top matches are sorted by descending similarity score.
- The user can optionally filter results based on score thresholds.

5. Results Display & Interaction

- A responsive product grid presents visually similar items with name, vendor, and price.
- Smooth hover animations and loading indicators improve experience.
- Clicking a result redirects to the corresponding product URL for exploration.


User Interaction Layer
Upload Image or Enter URL
Preview Image + Validate Input
AI Processing Layer
YOLO Object Detection Identify Key Objects
CLIP Embedding Generation Extract Visual Features
Compare with Catalog Compute Similarity Scores
Result Visualization Layer
Display Matched Products Results Grid + Metadata
User Applies Filters or Initiates New Search

Challenges & Learnings

During the development of the Visual Product Matcher Build, several challenges were encountered across model integration, frontend performance, and deployment. Each presented valuable learning experiences that strengthened the project's overall design and implementation maturity.

Scenario 1: Model Integration and Optimization
Challenge:
Integrating YOLO and CLIP models in a purely frontend setup was complex due to model size, slow initial loading, and limited browser memory.

Learning:
Using ONNX Runtime Web and lazy-loading models only when the user initiated a search minimized memory consumption and improved load time. Selecting smaller, quantized model versions made real-time inference feasible within standard browsers.

Scenario 2: Real-Time Similarity Computation
Challenge:
Comparing embeddings from the uploaded image against 50+ catalog items in real time occasionally caused UI lags.
Learning:
Pre-computing catalog embeddings and storing them in a structured **catalog.json** file significantly reduced processing delay. Leveraging asynchronous execution (async/await) ensured smooth performance without blocking the user interface.

Scenario 3: Department Switch

Challenge:
Maintaining a seamless user experience while heavy ML inference was running proved difficult.
Learning:
Implementing visual feedback elements — such as **loading indicators**, **progress states**, and **error notifications** — kept the interface interactive. A responsive grid layout and minimalistic design further enhanced usability across devices.

Future Enhancements

The Visual Product Matcher Build successfully demonstrates the potential of browser-based AI for intelligent visual product discovery. While the current system achieves the core goal of finding visually similar products from a local catalog using YOLO and CLIP models, several enhancements can further improve its scalability, accuracy, and user engagement.

1. **Backend Integration with Database Support**
   - Introduce a lightweight backend (e.g., Spring Boot or FastAPI) to store catalog items, embeddings, and user search history.
   - Enable dynamic catalog updates, pagination, and scalable query performance.
2. **Cloud-Based Model Serving**
   - Deploy YOLO and CLIP models on a cloud inference API for faster and more reliable performance.
   - Offload computation from the browser to a GPU-backed service like AWS Lambda or Hugging Face Spaces.
3. **Advanced Search & Personalization**
   - Add multimodal search support (image + text).
   - Use user interaction data to personalize recommendations and refine similarity scoring.
4. **Improved Product Metadata Visualization**
   - Enhance the result grid with hover animations, real-time filters, and product analytics (e.g., popularity tags, categories).
   - Include cross-similarity clustering to group visually related products.
5. **Pinecone or FAISS Integration for Large-Scale Retrieval**
   - Replace linear similarity search with vector databases for handling thousands of embeddings efficiently.

Conclusion

The **Visual Product Matcher Build** stands as a strong demonstration of combining **AI model intelligence** with **modern web engineering** in a purely frontend environment. It effectively bridges the gap between **deep learning concepts** and **real-world application design**, offering users a simple yet powerful way to explore products visually.

This project highlights not only technical proficiency in model deployment and frontend architecture but also design sensitivity in creating an intuitive, interactive experience. With future improvements like cloud integration and backend expansion, this prototype can evolve into a production-grade visual recommendation system suitable for real e-commerce environments.