# **Visual Search Engine Using CLIP & FAISS**

## **Problem Statement:**

## In traditional search engines, image retrieval is heavily dependent on textual tags and metadata, which may be inaccurate or missing. There is a growing need for a more intuitive system that allows users to search by image content or natural language descriptions - especially useful in fields like e-commerce, security, and healthcare.

## **Proposed Solution:**

## We developed a Visual Search Engine using CLIP (Contrastive Language-Image Pretraining) that can understand both image and textual semantics. The system supports:

## Text-to-Image search

## Image-to-Image search

## This is achieved through:

## Encoding text and images into a shared embedding space using CLIP

## Indexing features with FAISS for fast similarity search

## Serving the backend using Flask

## Providing a user-friendly frontend with Streamlit

## **Technical Workflow**:

## Preprocess images using open\_clip transforms

## Encode features using CLIP model (ViT-B-32)

## Normalize and index embeddings using FAISS

## Search: - Text queries encoded to embedding → compared with indexed image embeddings - Uploaded image encoded similarly → compared with index Results returned with similarity scores

## **Technologies Used:**

|  |  |
| --- | --- |
| Tool/Library | Purpose |
| Python | Backend and model development |
| Streamlit | Web-based UI |
| Flask | REST API for text/image search |
| CLIP (OpenCLIP) | Visual and textual feature extraction |
| FAISS | Fast similarity search using vector index |
| PIL, NumPy | Image processing and numerical ops |
| |  | | --- | | Watchdog | | Detects image folder changes to trigger re-indexing |

## **System Architecture diagram:**

## **Features:**

## Text-to-Image Search – Users input a text prompt to retrieve relevant images.

## Image-to-Image Search – Upload an image and get visually similar images.

## Real-time Re-indexing – Automatically updates index when new images are added.

## **Technical Workflow:**

## Preprocess images using open\_clip transforms

## Encode features using CLIP model (ViT-B-32)

## Normalize and index embeddings using FAISS

## Search: - Text queries encoded to embedding → compared with indexed image embeddings - Uploaded image encoded similarly → compared with index

## Results returned with similarity scores

## **Team Contributions:**

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Amandeep Singh | Arshdeep Kaur | Keshav |
| Project Planning & Architecture | ✅ | ✅ |  |
| CLIP Model Integration | ✅ |  | ✅ |
| FAISS Indexing | ✅ | ✅ |  |
| Flask API Development |  | ✅ | ✅ |
| Streamlit Frontend |  |  | ✅ |
| Re-indexing Automation | ✅ | ✅ |  |
| Testing & Debugging | ✅ | ✅ | ✅ |
| Documentation | ✅ |  | ✅ |

## 

## **Future Enhancements:**

## **-** Fine-tuning CLIP for domain-specific datasets (e.g., fashion, medical) - Add feedback loop to improve retrieval quality - Deploy as a cloud microservice using Docker & Kubernetes - Multi-language text query support

## **Future Enhancements:**

## Fine-tuning CLIP for domain-specific datasets (e.g., fashion, medical)

## Add feedback loop to improve retrieval quality

## Deploy as a cloud microservice using Docker & Kubernetes

## Multi-language text query support

**Conclusion:**

This project demonstrates the potential of combining Vision-Language Models with efficient search algorithms to build an intuitive and powerful Visual Search Engine. It removes the reliance on manual tags and allows seamless search via text or images, making it highly applicable to real-world domains like fashion, art, and surveillance.