# 🐾 Reverse Image Search

## Architecture Overview

This project allows users to search for visually similar images by uploading an image. It uses deep learning to extract features from images and stores these features in a vector database. When a user uploads a new image, its features are extracted and compared with those in the database to retrieve the most similar matches. The architecture consists of ***four core components*** working together:

1.  ## Feature Extraction Layer (Google Colab)

    This is where the image dataset is pre-processed and encoded into numerical feature vectors.

    - Uses a pre-trained *ResNet50 CNN* to generate a 2048-dimensional embedding for each image.
    - The vectors are normalized for efficient similarity search using cosine distance.
    - Feature vectors are batched and uploaded to Pinecone along with image metadata (e.g., filename, Cloudinary URL).

    *This layer is run once initially (or periodically) to prepare the dataset.*

2.  ## Vector Database (Pinecone)

    Pinecone stores all feature vectors and enables fast similarity searches.

    - The index is set with 2048 dimensions and cosine similarity metric.
    - Each vector is accompanied by metadata, such as image name and URL, making the results human-readable.
    - Pinecone returns the top-k most similar vectors when queried with a new image's features.

    *This acts as the brain of the search system — optimized for high-speed, approximate nearest neighbor lookups.*

3.  ## Flask API

    This web service handles search requests from the frontend.

    - Accepts base64-encoded images from the frontend and processes them in-memory using Pillow.
    - Applies the same ResNet50 feature extraction pipeline as used in Colab.
    - Queries Pinecone with the extracted vector.
    - Parses the result and returns top matches (URLs, filenames, similarity scores) as JSON.

*This is the glue between your client-side UI and the vector database.*

4. Frontend (HTML + JavaScript)

This is the user-facing interface where visual search is initiated.

- Users can upload or drag an image to start a search.
- Shows a preview of the uploaded image and enables the search button.
- Sends the image to the backend via a POST request.
- Renders similar image thumbnails with similarity percentages.

## Components

| Component | Technology | Purpose |
|---|---|---|
| Feature Extraction | PyTorch (ResNet50) | Converts image into a vector |
| Vector DB | Pinecone (Free Tier) | Stores vectors and supports similarity search |
| API Server | Flask (Python) | Serves search endpoint |
| Frontend | HTML, JavaScript | Image upload and results display |
| Hosting | Railway hosting | Deploys backend and frontend |
| Storage | Cloudinary (Free Tier) | Stores original image files for preview |

## Codebase

To keep this guide concise, the full source code is broken down below:

- 🧠 **Feature Extraction/Flask API** → [Visual Search - Github](#)
- 💻 **WebApp** → [Visual Search App](#)