

White Paper

Face Identification System Using AuraFace and FAISS

A Scalable AI-Based Solution for Efficient Image Retrieval

Date: 31/07/2026

Abdul Basith P M

AI / ML

N-KORR Technologies Pvt Ltd.

1 Executive Summary

As part of a customer-driven requirement to quickly identify and retrieve photographs of a specific person from large collections of event and workplace images, NKORR developed a solution to address the challenges associated with manual image search. The existing manual search process was time-consuming and inefficient, especially as the volume of photographs increased.

To address this requirement, NKORR developed a Face Identification System that allows users to upload a reference image and find matching photographs without training a separate model for each person.

The solution uses AuraFace to generate facial feature embeddings and FAISS for fast similarity-based search across the indexed image collection. FastAPI powers the backend services, while Angular provides the user interface.

The solution enables efficient, scalable, and automated person-based image retrieval while significantly reducing the manual effort involved in searching large image collections.

2 Problem in Context

Organizations routinely capture thousands of photographs during a person's onboarding, corporate events, training sessions, conferences, and team-building activities. Over time, these images accumulate into large repositories that are difficult to organize and search efficiently. Retrieving images of a specific individual manually is often time-consuming and labor-intensive.

Conventional approaches, such as organizing images by filenames, folders, or manually assigned tags, are difficult to maintain and do not scale as image collections grow. Manual tagging is both time-consuming and prone to human error, while many existing consumer-oriented facial recognition solutions depend on cloud-based services or are not designed to meet enterprise requirements for privacy, customization, and integration.

Organizations therefore require an intelligent, scalable, and secure solution that can automatically identify individuals across large image collections without relying on manual metadata. By leveraging advances in artificial intelligence, facial embeddings, and vector similarity search, all the photographs can be indexed and searched based on facial features rather than filenames or tags. This enables fast, accurate, and automated image retrieval while significantly reducing manual effort and improving the overall accessibility of organizational photo repositories.

For production deployment, the solution would require appropriate privacy and compliance controls, including consent where required, data retention and deletion policies, secure storage and access controls, and assessment of applicable regulatory and organizational requirements. These considerations are outside the scope of the current POC but must be addressed for production readiness.

3 Technical Deep Dive

The Face Identification System is built on a modular architecture that combines modern web technologies with deep learning-based face recognition and high-performance vector search. The system consists of an Angular frontend, a FastAPI backend, AuraFace for facial feature extraction, FAISS for similarity search, and a lightweight SQLite database for metadata management.

3.1 System Architecture

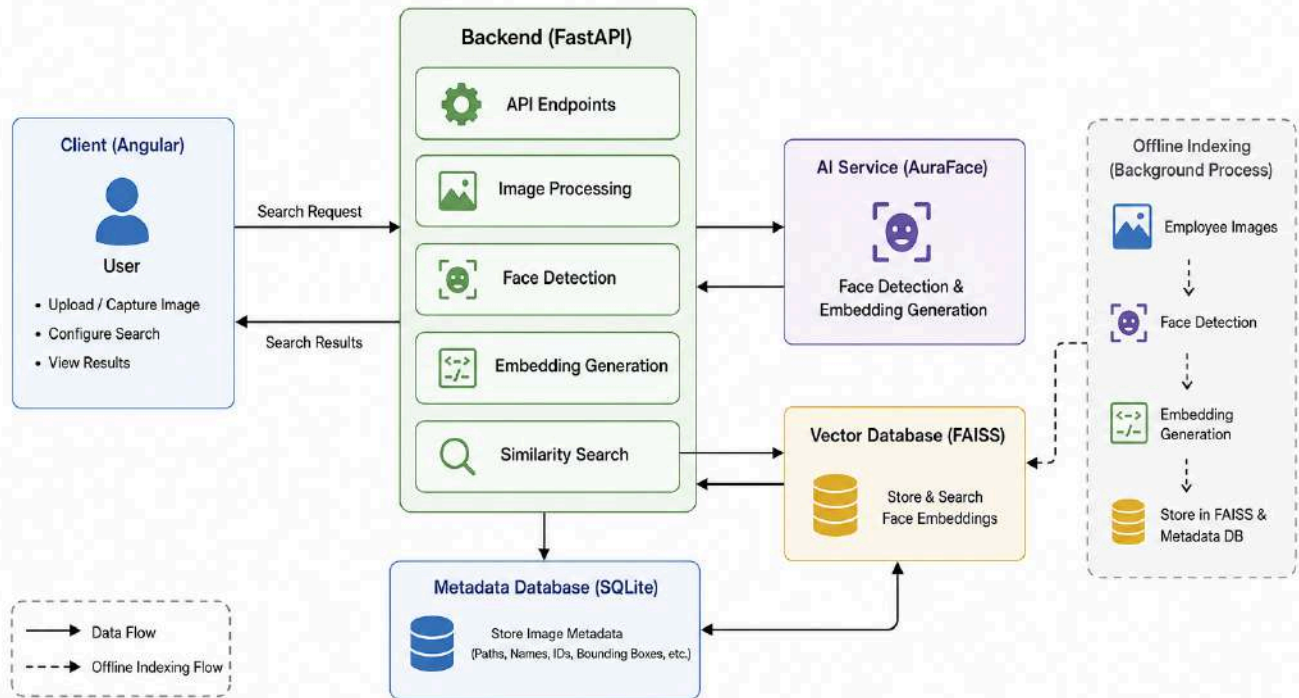


fig 3.1 Architecture diagram of Face Search application

The solution follows a client-server architecture. The Angular frontend provides an intuitive interface for uploading images, capturing photos through a webcam, configuring search parameters, and displaying search results. The FastAPI backend exposes RESTful APIs that process search requests, manage image indexing, and return matching results. The backend communicates with the AI services responsible for face detection, embedding generation, and vector similarity search.

3.2 Face Detection and Embedding Generation

All the photographs are processed using AuraFace (InsightFace), which performs face detection and extracts a fixed-length numerical representation (embedding) for every detected face. These embeddings capture the unique facial characteristics of an individual while remaining robust to variations in pose, lighting, and facial expressions. Since the embeddings represent facial identity rather than image pixels, they enable accurate comparison between different photographs of the same person.

3.3 Offline Indexing

To ensure efficient searching, all the photographs undergo an offline indexing process. The system scans the image repository, detects all visible faces, generates facial embeddings, and stores them in a FAISS index. Metadata such as image filename, image path, face identifier, and bounding box coordinates are stored separately in a SQLite database. Performing these computationally intensive tasks offline eliminates the need to repeatedly process the database during every search request.

3.4 Similarity Search

When a user uploads a query image, the system detects the face, generates its embedding, and performs a similarity search against the pre-built FAISS index. The search identifies the closest matching embeddings using vector similarity and returns the most relevant photographs. A configurable similarity threshold filters low-confidence matches, while the maximum number of returned results can also be customized according to user requirements.

3.5 Backend Services

The backend follows a service-oriented architecture in which individual services are responsible for specific tasks such as image processing, face detection, indexing, and similarity search. This modular design improves maintainability, simplifies testing, and allows components to be updated independently without affecting the overall system.

3.6 Frontend Interface

The Angular frontend provides a responsive user experience with support for image uploads, webcam capture, drag-and-drop functionality, configurable search parameters, and dynamic result visualization. Search results display the matched images together with similarity scores, detected face locations, and links to the original photographs, enabling users to quickly identify relevant images.

3.7 System Workflow

The system separates offline indexing from online searching to optimize performance. All photographs are indexed once to generate facial embeddings and construct the FAISS index. During a search, only the uploaded query image is processed, significantly reducing response time and enabling efficient retrieval even for large image collections.

This architecture provides a scalable and efficient facial image retrieval platform that combines deep learning-based face representation with high-performance vector similarity search, making it suitable for enterprise-scale image management.

4 Results

Employee Face Search
Backend Connected

Upload File

Live Camera

Similarity Threshold

0.40

Higher thresholds decrease false matches.

Max Matches

5 Matches

Search Employee Photos

Search Results

No matches to display

Upload a portrait face photo on the left, adjust your threshold, and click search to locate matching employee group photos.

Search Results

5 Matches Found

IMG_20250823_165629.jpg

88%

Indexed Face ID: 346

MATCH SCORE

Open Original Photo

IMG_20250823_165625.jpg

87%

Indexed Face ID: 340

MATCH SCORE

Open Original Photo

IMG_20250823_165502.jpg

85.7%

Indexed Face ID: 334

MATCH SCORE

Open Original Photo

IMG_E3957.JPG

79.7%

Indexed Face ID: 393

MATCH SCORE

Open Original Photo

IMG_20250827_104751.jpg

77.9%

Indexed Face ID: 365

MATCH SCORE

Open Original Photo

5 Limitations

While the system is designed to provide efficient face-based image retrieval, identification performance can vary depending on image quality and capture conditions. Challenging scenarios such as partial face occlusion, masks, extreme lighting conditions, significant pose variations, low-resolution images, and blurred or partially visible faces may reduce matching reliability. Visually similar individuals, including twins or people with similar facial characteristics, may also result in incorrect matches.

The system relies on similarity thresholds to determine whether a detected face should be considered a match. An inappropriate threshold may increase false-positive matches or exclude valid matches. Therefore, threshold values should be calibrated and validated using representative images from the intended deployment environment.

The current implementation is a proof of concept and does not establish performance guarantees across all possible operating conditions. Further evaluation using representative datasets and edge-case scenarios would be required before production deployment, particularly where incorrect identification could have significant consequences.

6 Recommendations

The system can be further improved through:

- **Incremental indexing** to add new photographs to the FAISS index without rebuilding the complete dataset.
- **Similarity threshold calibration** using representative images to balance false matches and missed matches.
- **FAISS index optimization** as the number of stored face embeddings increases, balancing search speed and accuracy.
- **Model and index versioning** to ensure embeddings remain consistent when the AuraFace model or configuration changes.
- **FAISS-SQLite synchronization** to maintain reliable mapping between face embeddings, source images, face IDs, and bounding boxes.

7 References

#	Source	Notes
1	AuraFace-v1 Model - https://huggingface.co/fal/AuraFace-v1	The model we used to create Facial Embeddings
2	FAISS - https://faiss.ai/index.html	Introduction to FAISS