Face Search AI Project Document

# **Theme Park Photo Search & Management System**

**Face Recognition Powered Guest-Photo Experience**

## **Overview**

This system lets guests **search, download, and email event photos** with face recognition. Admins manage image collections, users, logs, and configuration via a unified FastAPI backend.

* Face detection/embedding via **InsightFace**
* Fast similarity search on **Milvus** vector database
* Meta & audit data with **SQLAlchemy**

## **Key Libraries & Frameworks**

* **FastAPI:** API backend, web endpoints
* **SQLAlchemy:** Database models, audit logging
* **InsightFace:** Deep learning face detection & embedding
* **OpenCV (cv2):** Image reading, preview, watermark
* **NumPy:** Efficient image, array conversion
* **pymilvus / Milvus:** Embedding indexing, nearest-neighbor search
* **dotenv:** Environment variable management

## **User (Guest) Workflow**

## **Login & Access**

* Guests log in with name & mobile.
* New guests automatically registered; repeat users get session cookies.

## **Photo Search**

* Upload a selfie to trigger search.
* System detects faces, encodes features (512-d), searches collection for matches (threshold = 1.0, L2 distance).
* Matched images shown as previews.

## **Downloading Photos**

* Choose desired matches, download originals as ZIP , send to email and prints .

## **Emailing Photos**

* Selected photos can be emailed; handled as a background task.

## **Logout**

* Session cookies deleted on logout.

## 

## 

## **Admin Workflow**

## **1. Admin Authentication**

* Dedicated login page and session-protected dashboard

## **2. Collection Management**

* View stats (number of images/embeddings), upload new image folders, update/sync collections, or bulk delete as needed

## **3. User & Activity Management**

* Monitor, list, or bulk delete guest accounts
* Full audit trails: view or bulk delete activity logs (searches, downloads, logins)

## **4. Admin User Management**

* Create new admins, view admin list, and bulk delete (self-deletion protection)

## **5. Guest Impersonation**

* Temporarily log in as any guest—great for support or issue diagnosis

## **API Endpoint Reference**

## **Authentication Endpoints**

| **Endpoint** | **Method** | **Description** | **Auth Type** |
| --- | --- | --- | --- |
| /login | POST | Guest login | Public |
| /guest/logout | POST | Logout guest | Guest |
| /admin/login | POST | Admin login | Public |
| /admin/logout | POST | Logout admin | Admin |

## **Guest API Endpoints**

| **Endpoint** | **Method** | **Description** | **Auth Type** |
| --- | --- | --- | --- |
| /api/collections | GET | List photo collections | Guest |
| /api/search/{collection} | POST | Face search in collection | Guest |
| /api/download-selected/ | POST | Download selected photos as ZIP | Guest |
| /api/send-email | POST | Email selected photos | Guest |

## **Admin API Endpoints**

| **Endpoint** | **Method** | **Description** | **Auth Type** |
| --- | --- | --- | --- |
| /api/admin/collections | GET | Collection stats/details | Admin |
| /api/admin/guests | GET | List guest users | Admin |
| /api/admin/activities | GET | List recent activity logs | Admin |
| /api/admin/admins | GET | List all admin users | Admin |
| /api/admin/available-folders | GET | List folders for new uploads | Admin |
| /api/admin/update-collection/{name} | POST | Add/update images in a collection | Admin |
| /api/admin/sync-collection/{name} | POST | Sync/refresh a collection | Admin |
| /api/admin/collections/bulk | DELETE | Bulk delete collections | Admin |
| /api/admin/guests/bulk | DELETE | Bulk delete guest accounts | Admin |
| /api/admin/activities/bulk | DELETE | Bulk delete activity logs | Admin |
| /api/admin/create-admin | POST | Create admin user | Admin |
| /api/admin/admins/bulk | DELETE | Bulk delete admins (no self-delete) | Admin |
| /api/admin/login-as-guest/{guest\_id} | POST | Admin impersonates specified guest | Admin |

## **Page Serving Endpoints**

| **Endpoint** | **Method** | **Description** | **Auth Type** |
| --- | --- | --- | --- |
| / | GET | Guest login page | Public |
| /app | GET | Main app after guest login | Guest |
| /admin/login | GET | Admin login page | Public |
| /admin | GET | Admin dashboard | Admin |

## **Static & Asset Directories**

* /static — Guest site assets
* /admin\_static — Admin dashboard assets
* /images, /images\_preview — Served images/previews

## 

## 

## 

## 

## 

## 

## 

## **Image Capture & Matching: Query Workflow**

## **Backend Flow**

1. **Image Upload:**
   * Guest uploads a photo; read using **OpenCV**, converted to a **NumPy array**.
2. **Face Detection & Embedding:**
   * Faces detected via **InsightFace (buffalo\_l)** model.
   * For each face, a unique **512-d embedding** is extracted.
   * Embedding, image path, and primary key saved in **Milvus** DB.
3. **Vector Search in Milvus:**
   * New face embedding is queried against the existing collection with **L2 (Euclidean) distance**.
   * Only results with distance **< threshold (default 1.0)** are returned as matches.
4. **Post-Processing Results:**
   * Filter for valid matches (distance below threshold).
   * Remove duplicates, keeping only the best hit per image (smallest distance).
   * Gather matched image metadata and preview paths.
5. **Frontend Display:**
   * Return paths to preview images, ready for user display.

## 

## 

## 

## 

## 

## 

## 

## **Technical Flow Diagram**

text

[Image Folder]

|

v

[OpenCV + InsightFace] <- face detection/embedding

|

v

[Face Embeddings (NumPy, 512-d)]

|

v

[Milvus Vector DB]

^

|

[New Query Embedding] |

[Milvus Vector Search (L2, threshold)]

|

[Fetch + Filter Matches, Display Previews]

## **End-to-End Process: Quick Table**

| **Step** | **Library/Model** | **Metric/Parameter** | **Output** |
| --- | --- | --- | --- |
| Image reading | OpenCV / NumPy | - | NumPy array |
| Face detection/embedding | InsightFace (buffalo\_l) | 512-d embedding | Embedding array |
| Storage (add) | Milvus (pymilvus) | IVF\_FLAT, L2, nprobe=20 | DB record, indexed vector |
| Matching (search) | Milvus (pymilvus) | L2, threshold=1.0 | Ranked match list |
| Preview/watermark | OpenCV | JPG quality | Preview image for UI |

## 

## **Security & Sessions**

* All endpoints (except login) require authentication
* Sessions via HTTP-only cookies; all actions are logged
* Admin passwords are securely hashed

## **Internal Modules**

* **database:** Models, session utils, logging, password helpers
* **dependencies:** Authentication for FastAPI routes
* **face\_search\_logic\_milvus:** FaceSearchEngine; core embedding/search logic
* **payment:** Payment, email, download endpoints/schemas
* **email\_utils:** Email sending logic

## **Special Services**

* **Reverse Geocoding:** For admin-side collection tagging (location)
* **BackgroundTasks:** Asynchronous email/ZIP download

This document covers **system architecture, all user/admin workflows, API endpoints, internal data flows**, and **key libraries**—making it easy for any new developer or admin to get started and understand the backend query and image processing logic.