# 🧠Missing Person Face Search System

This document explains **everything completed so far**, step by step, so you (or anyone else) can set up the system from scratch without confusion.

The system is designed using **industry-grade architecture**:

- **Qdrant** → vector database (face embeddings)
- **PostgreSQL** → metadata database (name, state, police station, etc.)
- **InsightFace** → face detection & embedding
- **Docker** → run databases cleanly

---

## 👂 Project Structure (Current)

```
final_budddy/
├── final_images/              # FinalPersonId.jpg images (1 face per person)
├── faces_found.csv            # Metadata of persons with valid faces
├── faces_not_found.csv        # Persons with no usable face
├── create_qdrant_collection.py
├── ingest_embeddings.py
├── progress_checkpoint.csv    # Resume-safe checkpoint
├── qdrant_storage/            # Qdrant persistent data (Docker volume)
├── pgdata/                    # PostgreSQL persistent data (Docker volume)
└── README.md                  # This file
```

---

## 🐱System Architecture (High Level)

```
Image (.jpg)
   ↓
InsightFace (buffalo_s)
   ↓
Face Embedding (512-D)
   ↓
Qdrant Vector DB  ── FinalPersonId ── PostgreSQL Metadata DB
```

- **Qdrant** stores only vectors + `FinalPersonId`
- **PostgreSQL** stores all human-readable metadata

---

## 🖕 Prerequisites

### 🤩Python Environment

```
python --version   # Python 3.10+ recommended
```

Create virtual environment:

```
python -m venv .venv
source .venv/bin/activate
```

Install dependencies:

```
pip install insightface qdrant-client psycopg2-binary pillow numpy tqdm
requests
```

---

## 💞Docker Setup

### 🤪Verify Docker Installation

```
docker --version
docker ps
```

If Docker is not running (macOS):

```
open -a Docker
```

Wait until Docker Desktop shows **“Docker is running”**.

---

## 🧠Start Qdrant (Vector Database)

### 🤫Run Qdrant Container

```
docker run -d
  --name qdrant
  -p 6333:6333
  -p 6334:6334
  -v $(pwd)/qdrant_storage:/qdrant/storage
  qdrant/qdrant
```

### 😈 Verify Qdrant

```
docker ps
```

Open dashboard in browser:

```
http://localhost:6333/dashboard
```

---

## 😭 Start PostgreSQL (Metadata Database)

### 🤭 Run PostgreSQL Container

```
docker run -d
  --name postgres
  -e POSTGRES_DB=faces_db
  -e POSTGRES_USER=postgres
  -e POSTGRES_PASSWORD=postgres
  -p 5432:5432
  -v $(pwd)/pgdata:/var/lib/postgresql/data
  postgres:15
```

### 🤮 Verify PostgreSQL

```
docker ps
```

---

## 😼 Connect to PostgreSQL (Recommended Way)

You **do not need** `psql` **installed locally**.

```
docker exec -it postgres psql -U postgres -d faces_db
```

Password:

```
postgres
```

Inside psql:

```
\dt
```

Exit:

```
\q
```

---

## 👎Create Qdrant Collection

### 🤯Run Collection Creation Script

```
python create_qdrant_collection.py
```

Expected output:

```
✅ Qdrant collection created with payload index
```

(or)

```
ℹ️ Qdrant collection already exists
```

Collection details: - Name: `missing_person_faces` - Vector size: `512` - Distance: `COSINE`

---

## 🧠Dataset Preparation (Already Done)

### What exists already

- Face-filtered images in `final_images/`
- One image per `FinalPersonId`
- `faces_found.csv` contains:

```
FinalPersonId, Name, Sex, BirthYear, State, District,
PoliceStation, TracingStatus, ImageFile
```

---

## 😥Embedding + Metadata Ingestion

### 🥰Run Ingestion Script

```
python ingest_embeddings.py
```

What this script does:

- Reads `faces_found.csv`
- Loads `final_images/FP_xxx.jpg`
- Generates face embedding (`buffalo_s`)
- Inserts embedding into **Qdrant**
- Inserts metadata into **PostgreSQL**
- Skips already-inserted `FinalPersonId`

This script is **resume-safe**.

---

## 👉Verify Data After Ingestion

### Qdrant

```
curl http://localhost:6333/collections
```

### PostgreSQL

```
docker exec -it postgres psql -U postgres -d faces_db
```

```
SELECT COUNT(*) FROM persons;
SELECT * FROM persons LIMIT 5;
```

---

## 👈Current Status Checklist

| Component | Status |
|---|---|
| Image dataset | ✅Ready |
| Face filtering | ✅Done |
| Qdrant running | ✅ |
| PostgreSQL running | ✅ |
| Vector ingestion | ✅ |
| Metadata ingestion | ✅ |

---

## 😕Next Steps (Future)

- ⚠️FastAPI face search API
- 🈂️Admin dashboards

- Threshold tuning
- 🚫Authentication
- 🚫GPU acceleration

---

## Summary

You now have a **real, production-style face search backend**:

- Clean separation of vector search and metadata
- Resume-safe pipelines
- Scalable architecture
- Ready for APIs and UI

This is **not a demo system** — it's how serious AI backends are built.