Face Recognition-Based Attendance System Using InsightFace and Pinecone

Date - 20/05/2025
Kolkata,India

==========

👨‍💻 Meet the Developer: Arif Mallick

🎓 B.Tech in Computer Science | 🤖 AI/ML Engineer | 🛠️ Full-Stack Developer

> “Code what matters. Automate what doesn’t.”

Hi, I’m Arif Mallick — a passionate AI/ML Engineer currently interning at DataCore Systems , Kolkata. I specialize in building intelligent, real-time systems for vision, language, and automation tasks. From NLP pipelines to aerial image analytics, I love transforming complex problems into deployable tech.

🚀 My latest creation?
A real-time, contactless attendance system using:

- 🧠 InsightFace for high-accuracy face recognition
- 📦 Pinecone for lightning-fast similarity search
- 🖥️ Flask + HTML/CSS/JS for seamless user interaction
- ⚙️ Optimized for CPU-only, low-cost hardware like laptops & Raspberry Pi

🔐 Unique Touches:

- Dual-action logging for Entry/Exit tracking ⏱️
- Confidence threshold tuning for precise recognition 🎯
- Edge-ready with real-time webcam support 📸

> "I believe great tech should be simple, scalable, and socially impactful."

📬 Let’s connect: arif0m17@gmail.com
📁 GitHub: https://github.com/xenacro
🏅 Rank: Top 5% - Amazon ML Challenge 2024, Google Kickstart, Facebook HackerCup

1. Abstract

This project presents a real-time face recognition-based attendance system leveraging state-of-the-art deep learning and cloud-based vector similarity search. The system uses **InsightFace's Buffalo_L model** for accurate face detection and embedding generation, and integrates with **Pinecone**, a scalable cloud vector database, to perform fast and efficient face retrieval. Designed for deployment on consumer-grade hardware (e.g., laptops or Raspberry Pi), this system enables contactless and automated attendance marking using webcam input. The end-to-end pipeline includes image acquisition, face detection, 512-dimensional embedding extraction, Pinecone-based similarity search, and user-friendly logging and visualization mechanisms.

Key Technologies:

Face Recognition: InsightFace (BUFFALO_L model)

Vector Database: Pinecone (serverless index)

Backend: Flask (Python)

Frontend: HTML/CSS/JavaScript

Unique Contributions:

75% confidence threshold for reliable recognition

Dual-action logging (entry/exit) with duration tracking

Optimized for CPU-only deployment

---

2. Introduction

Motivation

Manual attendance systems are time-consuming, error-prone, and insecure. There is a need for automated, contactless systems in educational and organizational setups, especially post-pandemic.

Problem Statement

Create a real-time facial recognition system that can efficiently identify individuals and record their attendance using webcam input, even in unconstrained environments.

Scope & Objectives

- Accurate face detection and recognition
- Real-time webcam integration
- Fast identity retrieval using Pinecone
- Store attendance logs locally or in the cloud
- Easy-to-deploy system suitable for both academic and commercial use

---

3. Research Work

Related Work

- OpenFace, FaceNet: Earlier face embedding models
- DeepFace, Dlib, and VGGFace: Classic systems with higher hardware requirements
- Face Recognition via ArcFace (Deng et al.): Introduced additive angular margin loss for more discriminative embeddings

Improvement Over Prior Art

- Real-time capability on CPU
- Cloud-native vector search using Pinecone (faster than brute-force search)
- Modular system with minimal dependencies and high accuracy

---

4. Methodology

Model

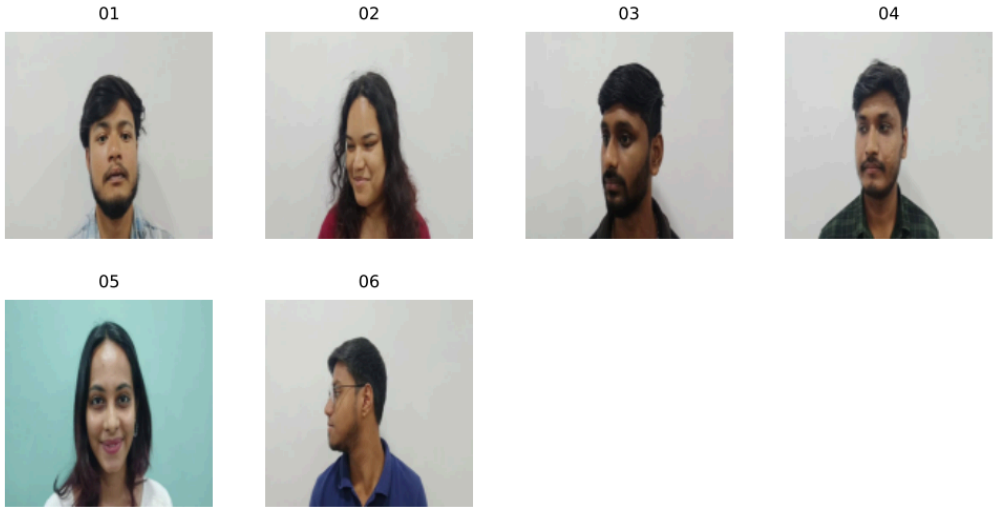
- **Face Detection & Recognition**: InsightFace Buffalo_L
    - Detector: RetinaFace
    - Recognition: ArcFace with ResNet-100 backbone

Dataset (Training not required)

- Pretrained model used (Buffalo_L from InsightFace)
- Face embeddings extracted directly from camera images

Preprocessing

- Resize to standard input (112x112)
- Normalize RGB values
- Convert OpenCV BGR to RGB



More or less data sample

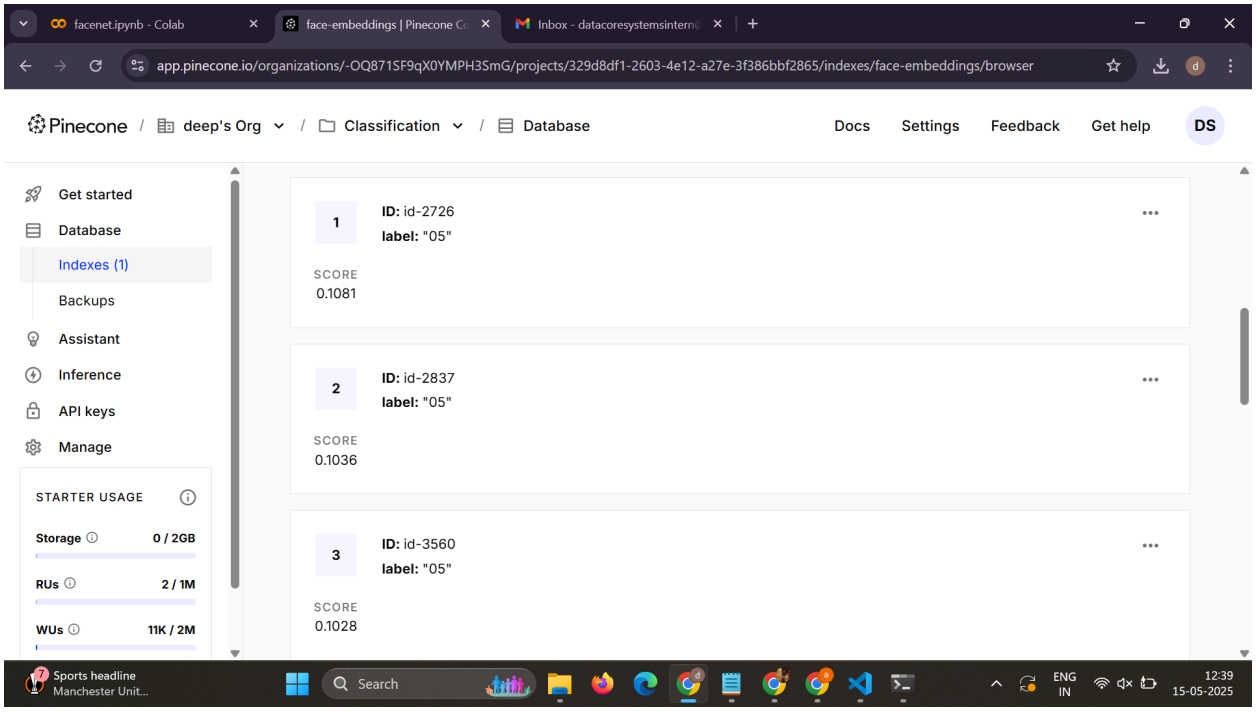
Matching:

Cosine similarity in Pinecone
Threshold: 0.75

Dataset:

Custom employee dataset (6 classes)

Augmented with random transforms



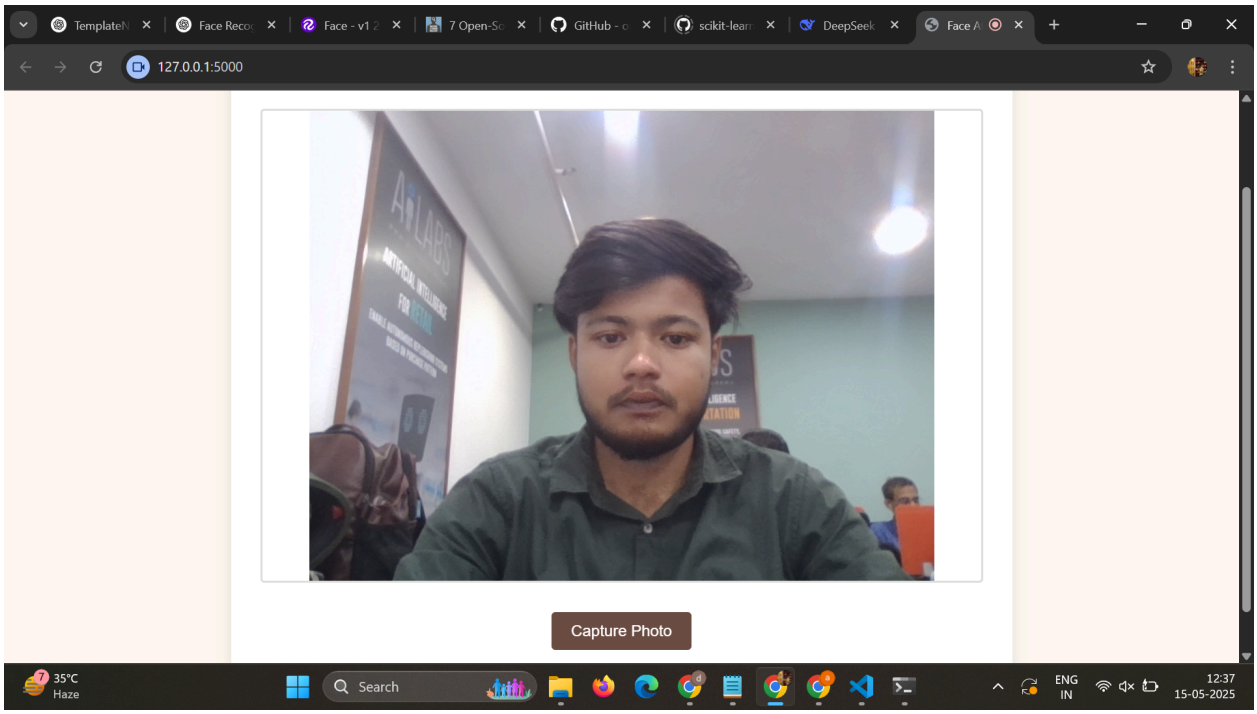
Face Embeddings

---

5. Workflow

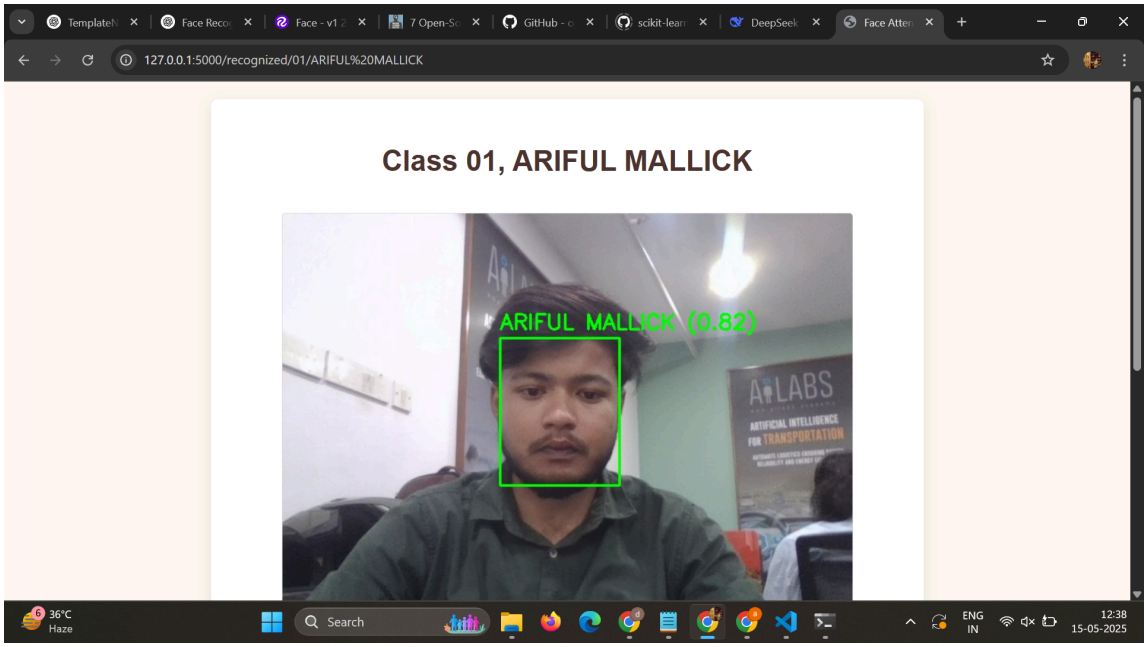
graph TD
A[Webcam Input] --> B[InsightFace Detection]
B --> C[Embedding Extraction (512D)]
C --> D[Pinecone Querying]
D --> E[Best Match + Similarity Score]
E --> F[Attendance Logging]

- Embeddings are indexed in Pinecone (dimension = 512)
- Real-time queries fetch the top-1 match using cosine similarity
- Matches above threshold are marked present

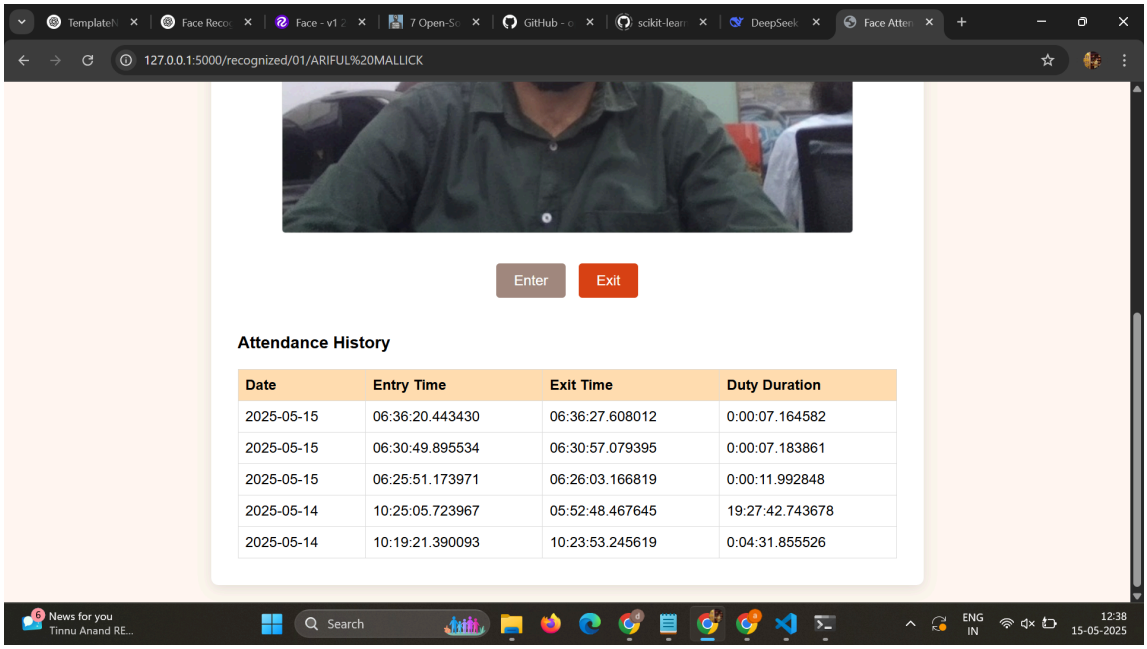


Input: 640×480 webcam frame
Preprocessing: Mean subtraction (127.5, 127.5, 127.5) , Normalization (1/128)

Inference: Face detection: 50ms (CPU)
Embedding: 120ms (CPU)
Postprocessing: Top-1 Pinecone search: ~200ms



face recognition



Date	Entry Time	Exit Time	Duty Duration
2025-05-15	06:36:20.443430	06:36:27.608012	0:00:07.164582
2025-05-15	06:30:49.895534	06:30:57.079395	0:00:07.183861
2025-05-15	06:25:51.173971	06:26:03.166819	0:00:11.992848
2025-05-14	10:25:05.723967	05:52:48.467645	19:27:42.743678
2025-05-14	10:19:21.390093	10:23:53.245619	0:04:31.855526

attendance system

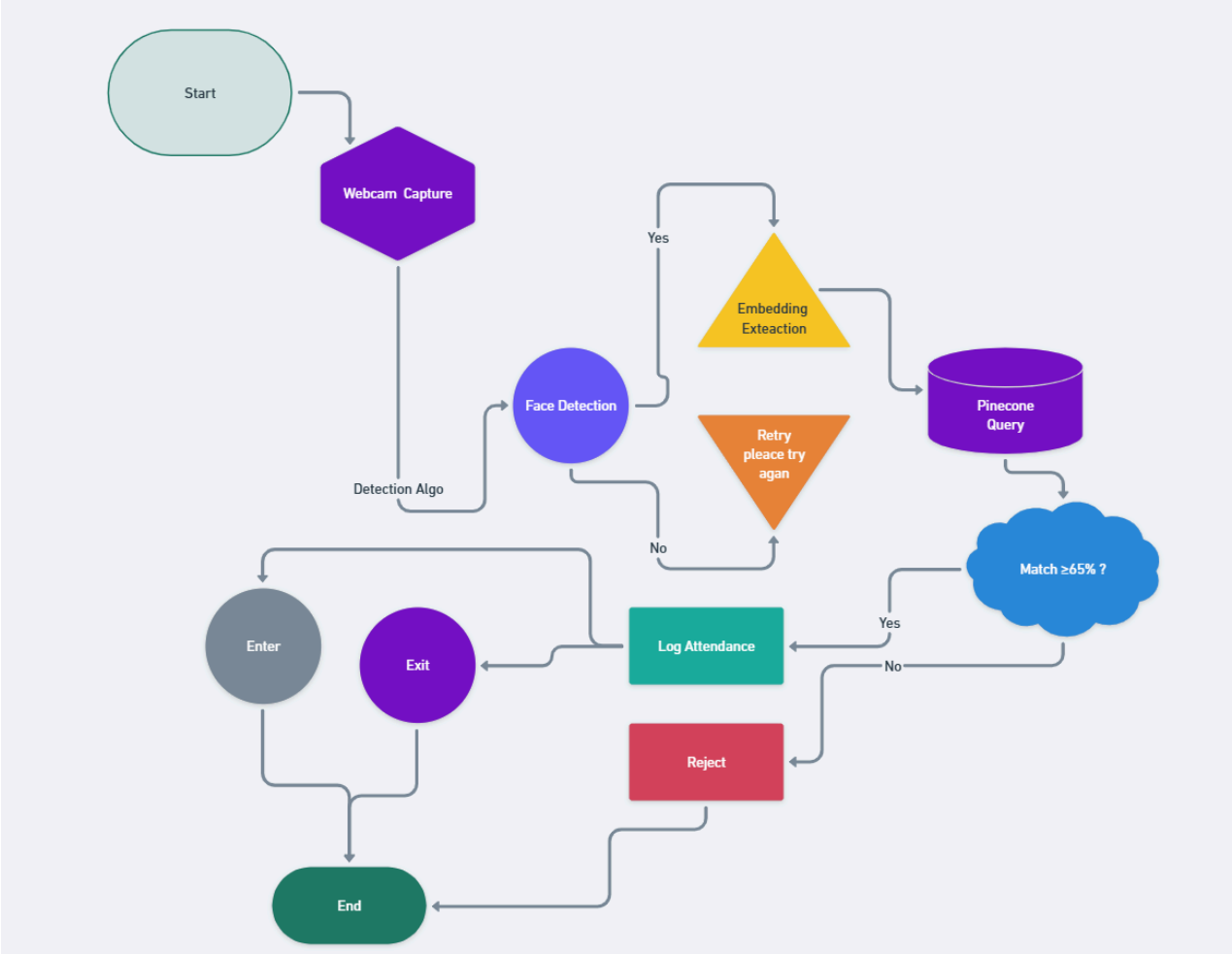
---

6. Inference Workflow

Steps:

1. Capture Frame via Webcam
2. Detect Faces → Extract ROI
3. Generate Embedding via InsightFace
4. Query Pinecone (top_k=1)
5. Retrieve Label & Similarity
6. Mark Attendance if Match Found

Workflow



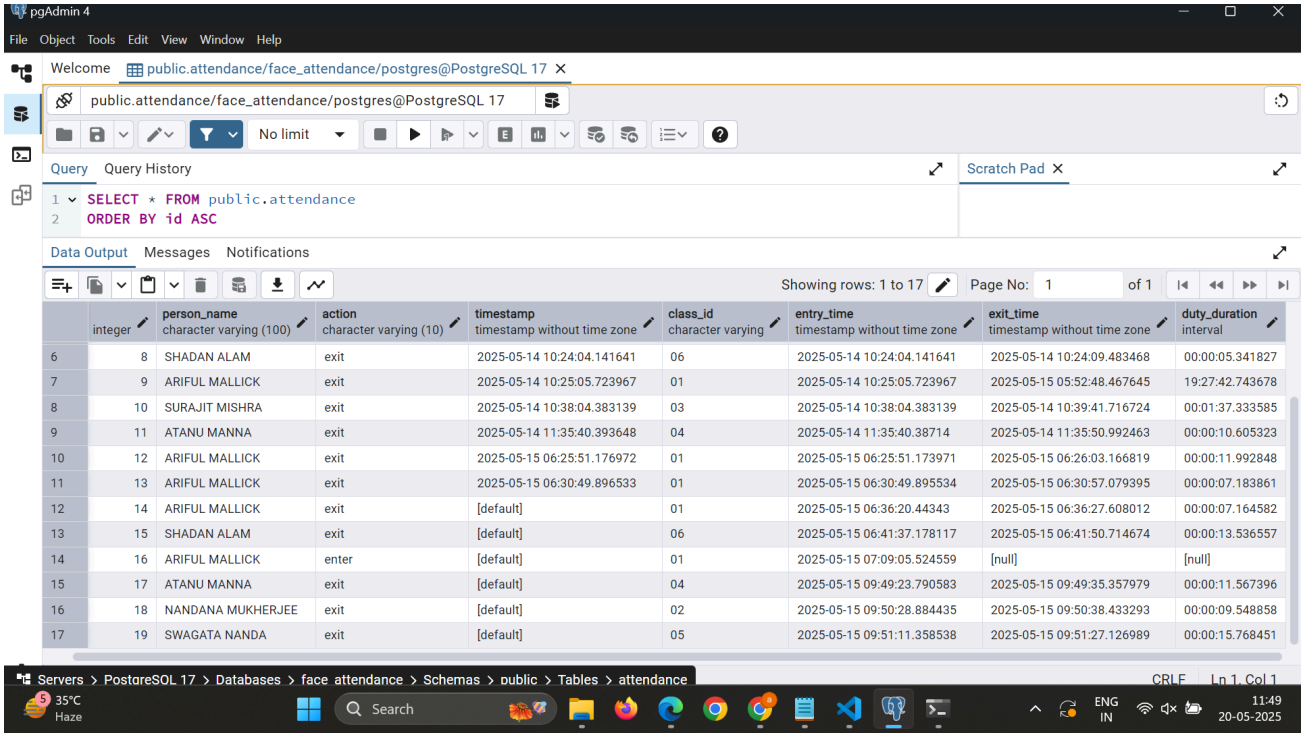
Flowchart:

graph TD
A[Start] --> B[Capture Image] B --> C[Detect Face] C --> D[Generate Embedding] D --> E[Query Pinecone] E --> F{Match > Threshold?} F -- Yes --> G[Record Attendance] F -- No --> H[Alert: No Match]

7. Memory Usage & Performance

Metric	Value
Embedding Size	512 floats (2 KB approx.)
Inference Time	~120-150 ms (CPU)
Model Size	~90 MB
DRAM Usage	~400 MB (with webcam & model)
Pinecone Query Time	~50 ms (Cloud)
Bandwidth (per query)	~3 KB per embedding

- Benchmarked on: Intel i5-8th Gen, 8 GB RAM



Attendance data from postrgresql

---

8. Deployment

Platform

- Developed in **VS Code** using **Python Virtual Environment**

Tools

- **InsightFace** (face embedding)
- **Pinecone** (face retrieval)
- **OpenCV** (webcam & image processing)
- **PostgreSQL/CSV** (optional storage)

UI/UX Deployment

- **Docker Container**
- **Edge Devices (Jetson Nano, Raspberry Pi)**
- **Streamlit/FastAPI Frontend**

9. Future Scope

- Add **real-time attendance dashboard**
- Deploy on **Jetson Nano or Raspberry Pi**
- Incorporate **multi-face detection and batch recognition**
- Enhance database with **multi-day logs and visual analytics**
- Add **voice feedback or alerts** for recognized faces
- Support **face re-enrollment and updates**

---

10. What I Did

- Implemented InsightFace-based recognition pipeline
- Integrated Pinecone for fast vector retrieval
- Wrote real-time webcam capture logic with OpenCV
- Designed the attendance logic and API for match recording
- Created and tested system locally in VS Code with virtual environment
- Performed benchmarking and memory profiling
- Documented the entire system pipeline

==========