Face Authentication Attendance System

AI/ML Internship Assignment Report

1. Introduction

Attendance management is an essential requirement in organizations and institutions. Traditional methods such as manual registers or card-based systems are time-consuming and prone to proxy attendance.

This project implements a **Face Authentication Attendance System** that uses computer vision and machine learning to automatically record **Punch-In** and **Punch-Out** using real-time camera input.

The system performs:

- Face registration
- Face recognition
- Automatic attendance marking
- Punch-in / Punch-out handling

2. Objectives

- Build a real-time face authentication system
- Accurately identify registered users using a webcam
- Automatically mark:
    - **Punch In** on first scan
    - **Punch Out** on next scan
- Store attendance records in a structured format
- Handle varying lighting conditions
- Implement a basic spoof-prevention mechanism

3. System Overview

High-Level Workflow

1. User registers their face using the webcam
2. Multiple face images are stored in a dataset
3. A face recognition model is trained
4. When the user scans their face:
    - If no record exists → Punch In

- If Punch In exists → Punch Out

5. Attendance is saved in a CSV file

4. Technologies Used

Component	**Technology**
Programming Language	Python
Computer Vision	OpenCV
Face Detection	Haar Cascade Classifier
Face Recognition	LBPH (Local Binary Pattern Histogram)
Data Storage	CSV (Pandas)
Camera Input	Laptop Webcam
Environment	Windows, Python 3.10

5. Model and Approach Used

5.1 Face Detection

- Haar Cascade (haarcascade_frontalface_default.xml) is used
- Detects faces in grayscale images
- Lightweight and fast for real-time applications

5.2 Face Recognition

- **LBPH Face Recognizer**
- Uses texture-based local binary patterns
- Works well with small datasets
- Robust to moderate lighting changes

Why LBPH?

- No need for GPU
- Fast training and inference
- Suitable for internship-level real-time systems

6. Training Process

1. Capture ~30 face images per user

2. Convert images to grayscale
3. Resize faces to a fixed resolution (200×200)
4. Assign numeric labels to users
5. Train LBPH model using:
    - Face images
    - Corresponding labels

Training happens locally at runtime.

7. Attendance Logic (Punch In / Punch Out)

Attendance Rules

- One row per user per day
- First scan → **Punch In**
- Second scan → **Punch Out**
- Same row is updated (no duplicate rows)

Example CSV Format

Name,Date,Punch In,Punch Out

RAGHURAM,2026-01-31,09:15:23,17:42:10

8. Spoof Prevention (Basic)

The system includes **basic anti-spoofing measures**:

- Confidence threshold check
    - Rejects faces with low confidence
- Live camera input only (no static images)
- Requires real-time face detection before recognition

This is **not a full liveness detection system**, but helps reduce basic spoof attempts like printed photos.

9. Accuracy Expectations

Condition	**Expected Accuracy**
Good lighting	90–95%
Moderate lighting	80–90%

Condition	Expected Accuracy
Poor lighting	Reduced accuracy
Same person	High
Similar-looking people	Lower

Accuracy depends on:

- Quality of registered images
- Lighting conditions
- Camera resolution

10. Known Failure Cases

- Poor or extreme lighting
- Face partially covered (mask, scarf)
- Identical twins or very similar faces
- Printed photograph spoofing (advanced spoofing not handled)
- Low-resolution webcams

11. Limitations

- Uses classical ML, not deep learning
- No advanced liveness detection (blink, depth, IR)
- CSV-based storage (not scalable for large systems)
- Single-camera, local system

12. Possible Improvements

- Use FaceNet / ArcFace embeddings
- Add blink or head-movement detection
- Integrate database (SQLite / PostgreSQL)
- Build web dashboard using Flask
- Add multi-user concurrent handling
- Deploy as a hosted service

13. Conclusion

This project demonstrates a complete **end-to-end AI/ML system** using real-time computer vision. It successfully registers users, authenticates faces, and records attendance with punch-in and punch-out logic.

The system highlights:

- Practical ML usage
- Understanding of model limitations
- Real-world deployment considerations

14. Repository Link

GitHub Repository:
https://github.com/LAKAVATHRAGHURAM/face-auth-attendance