International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Jan-Feb-2026, ISSN (Online): 2395-566X

# AI-Powered Smart Attendance Management System Using Facial Recognition

**Vikasini E, Daniya U, Guide Dr.P.Jayasheelan, Assistant Professor**
Department of Data Science, Sri Krishna Adithya College of Arts and Science

**Abstract- — Paper registers and card systems for taking student and employee attendance are slow and full of mistakes. People can fake entries. Proxy marking is easy. Schools and workplaces need something more reliable and automatic to track who shows up. So we built an AI-powered smart attendance management system that uses facial recognition to record attendance in real time. The system is written in Python and uses OpenCV and the face recognition library. A SQLite database stores the structured data. A camera-enabled desktop app captures facial images. It matches people against a pre-registered face database and logs attendance with timestamps. No manual data entry. No easy way to mark attendance for someone else. The graphical interface uses Tkinter. Admins can manage records and run reports. They can also view attendance history. Tests show the system reaches high recognition accuracy under controlled lighting. It also cuts down administrative work a lot. This research shows how artificial intelligence and computer vision can be applied to institutional management systems to improve efficiency, reliability and accountability.**

**Keywords: Facial Recognition, Attendance Management, OpenCV, Machine Learning, Python Desktop Application, Computer Vision, SQLite, Tkinter, Biometric System.**

## I. INTRODUCTION

Attendance monitoring is a basic admin task in schools, offices and public organizations. You need accurate records to track student participation, measure employee output and meet rules. Yet many places still use old manual methods. Paper-based registers. Roll calls. Magnetic ID cards. Slow and messy.

These methods have clear problems. Manual registers allow proxy attendance where someone signs for another person. Card systems get shared. Manual entry takes time and introduces errors. Compiling and analysing records eats up staff hours.

AI and computer vision have opened new options to automate and secure attendance. Facial recognition is promising. It uses math to find and identify faces in images or video. Contactless. Harder to fake.

This project proposes an AI-powered smart attendance system that records attendance with real- time facial recognition. A desktop camera captures images of people entering a room or workspace. Detected faces are checked against a registered face database. Matches are logged automatically with accurate timestamps. No manual work. Fake entries are much harder.

The app is built in Python. It uses OpenCV for image work and the face recognition library for biometric ID. A SQLite database holds student or employee records and attendance data. The Tkinter interface lets admins register users, see reports and change settings.

The main goal is a reliable, accurate and low-cost attendance solution that can be deployed in institutions without special hardware beyond a standard webcam.

## II. LITERATURE REVIEW

Biometric tech for identity and access control has been studied for years. Fingerprint scanners, iris systems and face detection have been used in border control, banking, healthcare and education.

Facial recognition drew a lot of attention because it is contactless and easy to deploy. Early systems used geometric features and looked at positions of eyes, nose and mouth. They worked in controlled settings but struggled with different lighting, pose and expression.

Deep learning changed the field. Convolutional neural networks trained on big datasets extract high-dimensional facial embeddings that cope with many real-world changes. Models like Face Net and Deep Face reached near-human results on benchmarks and made practical face recognition possible.

Several studies looked at using facial recognition for attendance in schools. Those systems usually combine a camera, a face detector and a database backend to log people. Research shows automated facial attendance cuts admin work and improves record accuracy.

Open-source tools such as OpenCV and dlib made face recognition reachable for developers who are not ML experts. The face recognition library built on dlib offers a simple API for detection, encoding and comparison and fits well for attendance apps.

Many existing systems miss full admin interfaces and reporting. This system tries to fix that by pairing a solid recognition engine with a desktop app that supports user registration, logging and report generation.

## III. SYSTEM ARCHITECTURE

The system uses a modular design to separate key functions so it stays maintainable and scalable. It runs as a standalone desktop application. No internet required for the core features. There are five main modules that work together to provide the attendance functionality.

**Camera and Image Capture Module**
This module talks to the webcam through OpenCV and grabs real-time video frames. It watches the camera feed and detects human faces in each frame. Detected face regions are extracted and sent to the recognition engine.

**Facial Recognition Engine**
The engine uses the face_recognition library to compute 128-dimensional face encodings from detected faces. Those encodings are compared to a pre-registered database of known faces using Euclidean distance. A configurable tolerance threshold decides what counts as a match so the system can trade off accuracy and false positives.

**Database Management Module**
All structured data, including user profiles, face encodings and attendance logs, are stored in a SQLite database. The schema supports queries for daily, weekly and monthly reports. SQLite was chosen because it is lightweight and works well for desktop deployments.

**Graphical User Interface Module**
The GUI is built with Tkinter and gives admins an interactive way to use the system. Screens include user registration, face sample capture, live attendance logs and summary reports. The design focuses on simplicity and ease of use.

**Reporting and Notification Module**
This module reads attendance records from the database and creates formatted reports that can be exported as CSV. Admins can filter results by date range, department or individual to look at attendance patterns and spot irregularities.

## IV. SYSTEM IMPLEMENTATION

The whole system was implemented in Python. Python was chosen for its library ecosystem, clear syntax and strong community support for computer vision and ML work.

Face registration happens in an enrolment phase where photos are taken from multiple angles using the webcam. The face recognition library turns those images into 128-dimensional face encodings. Encodings are saved in the database with the user's name and ID number. Enrolment takes less than two minutes per person and needs no special gear.

During attendance sessions the system keeps grabbing video frames. Each frame is processed to find face regions using histograms of oriented gradients detection. For each found face the system computes an encoding and compares it with all registered encodings. If a match is within the tolerance threshold the system records an attendance entry with the user details and current timestamp.

To avoid duplicate entries the system checks whether a record already exists for that person in the current session. Only the first successful recognition is logged so people are not recorded multiple times during one session.

The Tkinter interface shows the camera feed with bounding boxes around detected faces and labels with recognized names. Admins can watch the process and spot any issues right away. The backend follows object-oriented principles with separate classes for camera control, recognition tasks, database work and report generation. This keeps the codebase maintainable and lets modules be updated or swapped independently.

## V. RESULTS AND DISCUSSION

We evaluated the system in controlled tests with twenty registered participants. Tests covered different lighting and distances to see how robust the recognition engine is.

In normal indoor lighting the system reached about 96 percent recognition accuracy when people stood one to two meters from the camera. The recognition for each person finished in

under one second. So, the system can process multiple people quickly.

Performance fell a bit in low light. Accuracy dropped to around 88 percent in dim conditions. That shows adequate lighting matters. Adding infrared lighting or image enhancement before recognition could help in future versions.

Automated logging removed all proxy marking during tests. Participants said registration and daily attendance were straightforward and low effort. Admin users reported much less time spent on attendance tasks versus manual methods.

The SQLite database handled concurrent reads and writes well for the group size we tested. Generating a one-month report took about three seconds, which shows the database design is suitable for institutional use.

## VI. CONCLUSION

The AI-Powered Smart Attendance Management System offers a practical solution to the problems of manual attendance tracking in schools and organizations. By pairing facial recognition with a desktop application, the system automates attendance, prevents proxy marking and cuts admin time.

Using open-source tools like OpenCV and the face recognition library keeps the system affordable and lets it run without special hardware. The modular design means new features can be added later, such as mobile app support, cloud storage or multiple cameras.

Future work could add liveness detection to stop photo spoofing, emotion or mask detection, and web dashboards for remote monitoring. Stronger ML models could improve accuracy in difficult environments.

The successful build and testing of this system shows that AI and computer vision can help solve institutional management problems and bring real gains in efficiency, accuracy and accountability.

## VII. REFERENCE

1. P. Viola and M. Jones, "Robust Real-Time Face Detection," International Journal of Computer Vision, vol. 57, no. 2, pp. 137–154, 2004.
2. G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments," University of Massachusetts, Technical Report, 2007.
3. F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," IEEE Conference on Computer Vision and Pattern Recognition, pp. 815–823, 2015.
4. Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," IEEE Conference on Computer Vision and Pattern Recognition, pp. 1701–1708, 2014.
5. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Advances in Neural Information Processing Systems, vol. 25, pp. 1097–1105, 2012.
6. Python Software Foundation. Python Documentation. Available https://docs.python.org
7. OpenCV Development Team. OpenCV Documentation. Available https://docs.opencv.org
8. A. Geitgey. face recognition Library. Available https://github.com/ageitgey/face_recognition
9. SQLite Consortium. SQLite Documentation. Available https://www.sqlite.org/docs.html
10. Python Software Foundation. Tkinter GUI Programming Reference. Available https://docs.python.org/3/library/tkinter.html