International Journal of Scientific Research and Engineering Development-– Volume 9 Issue 2, Mar-Apr 2026
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

A Privacy-Centric Monolithic Framework for Student Attendance Via Localized Deep Facial Embeddings

Abdul Rashid Darji[1], Fuzail Shaikh[2], Faisal Qureshi[3], Manahil Raza[4], Sayyed Muhsin[5], Mr. Shah Mohd Sharique[6]

[1,2,3,4,5] Students, Department of Computer, [Anjuman-I-Islam A. R. Kalsekar Polytechnic], [New Panvel]
[6]Project Guide, Department of Computer, [Anjuman-I-Islam A. R. Kalsekar Polytechnic], [New Panvel]
[1]abdulrashidd49@gmail.com, [2]manahilraza2601@gmail.com, [3]qureshifaisal9892@gmail.com, [4]so2smooth7@gmail.com, [5]fuzailshaikhcob@gmail.com, [6]SharqueShah@505gmail.com

----------**********----------

Abstract:

In India Colleges and institutions depends on manual roll number call or attendance sheets Wasting almost 10 to 15 Minutes of every lectures and also get fake attendance This problem Lead to the Development of the VisionMark a Artifiial intelligence based client server facial recognition attendance system engineered for deployment of old attendance system The system Uses a FastAPI backend performing 68 point facial landmark detection via dlib to generate 128 dimensional face embeddings and compared it through Euclidean distance thresholding. A React 18 frontend built with Vite delivers a glassmorphic dark themed interface with integrated 3D Spline visualizations. Real-time video is streamed as MJPEG over Local Host, and attendance records are logged directly to Google Sheets via the googlespread library. A supplementary Streamlit-based Student Portal deployed at vision-mark.in provides public read-only access to attendance statistics. The testing has done During Few lectures in classroom and it achieved 94% identification accuracy with sub 500ms per face latency reducing session start overhead to under 4-5 minute. The architecture prioritizes reproducibility and zero cost deployment suitable for resource constrained educational institutions.

Keywords — ***Key Words: Face Recognition, MTCNN, DeepFace, VGG-Face, Cosine Similarity, SQLite, Tkinter, Attendance Management, Facial Embeddings, Python***

----------**********----------

1.INTRODUCTION

This system is built using a monolithic Python architecture that runs entirely as a desktop application on a single computer. The software starts by using the MTCNN framework to detect multiple faces at the same time from a standard USB webcam. Once a face is found, the system uses the DeepFace library with a VGG-Face model to turn the facial features into a 2048-dimensional vector called an embedding. For the matching part, the code calculates the Cosine Similarity between the live face embedding and the ones already saved during enrollment. We used an optimized threshold of 0.68 because it gives the best balance for recognizing students correctly in different classroom lights.

All the student records, subject lists, and the daily attendance logs are managed locally through a built-in SQLite database. This means no student information ever leaves the host computer and there is no need to pay for any cloud APIs. Because of this offline processing method, the system is perfect for college labs where the internet connection is very slow or always disconnecting. The graphical interface is made using Python's Tkinter library, so the app stays very lightweight and can run on older i5 or i3 laboratory laptops without needing expensive GPU cards.

The software also uses a background threading model so that the camera feed does not freeze or lag while the AI is busy processing the face recognition. During the enrollment, the system takes three photos of a student and averages them to make the matching more strong. After the attendance is finished, the faculty can easily click a button to generate CSV and PDF reports. These reports are saved directly on the desktop and follow the same format used in official college registers. This setup provides a very fast and private way for teachers to keep digital records without any extra hardware cost.

Problem Statement

Current digital attendance solutions and biometric tools often face several technical and security-related challenges that make them difficult to use in standard classroom environments:

Most modern facial recognition systems require a constant and high-speed internet connection to communicate with cloud-based recognition APIs.

Using external cloud services for processing biometric data raises significant concerns regarding the privacy and security of student facial information.

Many existing automated systems require expensive, specialized hardware or high-end GPU servers that are not always available in polytechnic or engineering college labs.

Subscription-based or per-call API fees for cloud recognition create a recurring financial cost that many educational institutions cannot support long-term.

There is a lack of lightweight, standalone desktop software that can perform high-accuracy detection and embedding generation entirely on a local CPU.

Integrating automated attendance data with local administrative records often requires manual data transfer, which is prone to human error and data loss

LITERATURE REVIEW

Many researchers have worked on making attendance systems using face recognition because old biometric tools like fingerprint scanners have many problems. One of the first ways was the Eigenfaces method, which used a math trick called Principal Component Analysis (PCA) to find faces. This was very fast for the computer, but it did not work well if the classroom light changed or if the student was looking at a different angle. Later, a method called LBPH was used, which was much better at handling different lights, but it still has low accuracy when there are many students in a big dataset.

Recently, Convolutional Neural Networks (CNN) have made face recognition much more accurate. The VGG-Face architecture is a very strong model because it was trained on millions of face images, making it great for identifying people in real-time. To make these models work better, the face needs to be aligned correctly. The MTCNN framework is a very popular tool for this because it can detect faces and find facial landmarks like eyes and nose at the same time. This helps the system work even if the student is not standing perfectly straight in front of the camera.

The DeepFace library is another big step because it brings together many different models like VGG-Face and FaceNet under one Python package. Most other research papers use cloud-based APIs or expensive hardware for their systems. But these cloud systems have big problems like slow speed (latency), high service costs, and they are not safe for student privacy. Our system is different because it uses local processing on a standard PC, which solves these problems and works without any internet.

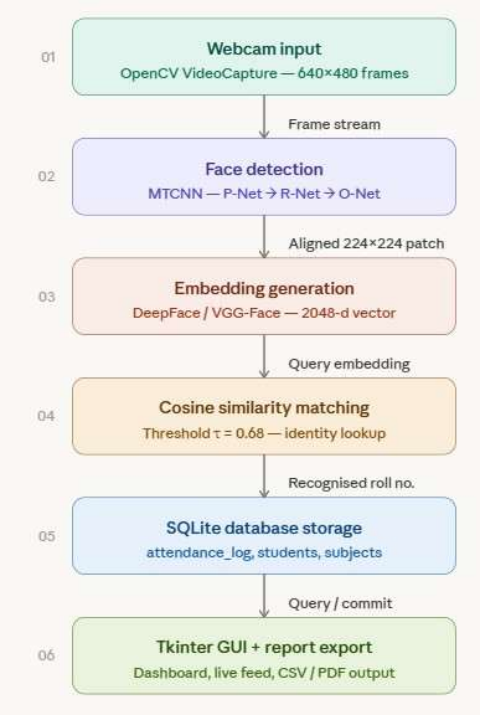
PROPOSED SYSTEM ARCHITECTURE

The proposed system follows a layered monolithic architecture implemented entirely in Python 3.10. All components, from webcam acquisition to

database storage and report generation, are integrated within a single application process. Figure 1 illustrates the high-level architecture and the system.

The system initiates by capturing live video frames at a resolution of 640×480 pixels using the OpenCV library. To ensure the application remains responsive on standard laboratory computers, the frame capture runs in a background daemon thread. This allows the Tkinter graphical interface to stay smooth while the heavy AI processing happens in the background. Instead of sending these frames to a server, the software processes each frame directly in the local system memory (RAM), which keeps student biometric data private and secure.

Fig -1: High-Level System Architecture



Face Detection using MTCNN

Face detection is performed using the Multi-Task Cascaded Convolutional Networks (MTCNN) framework. MTCNN employs three cascaded networks: a Proposal Network (PNet) that generates candidate face windows across multiple image scales, a Refinement Network (R-Net) that reduces false positives using bounding box regression, and an Output Network (O-Net) that refines bounding boxes and predicts five facial landmark coordinates (two eye centers, nose tip, and two mouth corners).

The landmark coordinates output by O-Net are used to perform affine alignment, normalizing each detected face to a 224×224 pixel canonical pose before embedding extraction. This alignment step significantly improves embedding consistency across sessions with varying head orientation, which is critical in a classroom setting where students face the camera at different angles.

MTCNN is implemented via the mtcnn Python package and processes 640×480 video frames at approximately 18 frames per second on a mid-range CPU, providing sufficient real-time responsiveness for the attendance capture workflow.

Embedding Generation via DeepFace VGG-Face

Following face detection and alignment, each 224×224 face patch is passed through the VGG-Face model loaded via the DeepFace library. VGG-Face is a 16-layer deep convolutional network trained on a dataset of over 2.6 million face images across 2,622 identities. The final fully connected layer before the classification head produces a 2048-dimensional embedding vector that encodes the unique facial geometry of the input.

During enrollment, students submit three frontal photographs under varying lighting conditions. The VGGFace model generates three 2048-dimensional embedding vectors per student, which are averaged to produce a single mean enrollment embedding. This averaging strategy reduces the influence of per-image noise and improves robustness. All enrollment embeddings are serialized to a Pickle file stored on disk and loaded into memory at application startup.

Cosine Similarity Matching

At recognition time, the embedding of each detected face is compared against all enrollment embeddings using cosine similarity rather than Euclidean distance. Cosine similarity measures

the angular difference between two embedding vectors and is invariant to the absolute magnitude of the vectors, making it more robust to lighting-induced embedding scale changes than L2 distance.

Cosine Similarity: $\cos(\theta) = (A \cdot B) / (||A|| \times ||B||)$

A similarity score exceeding the threshold $\tau = 0.68$ causes the detected face to be identified as the corresponding enrolled student. If no enrollment embedding achieves this threshold, the face is labeled as Unknown. The threshold value of 0.68 was determined through empirical tuning on a held-out validation set of 30 students, balancing the false acceptance rate (FAR) and false rejection rate (FRR).

SQLite Database Schema

Attendance data is persisted in a local SQLite database managed through Python's built-in sqlite3 module combined with the Pandas library for report generation. The schema comprises three primary tables:

- students: stores enrollment records (roll_no, name, department, enrollment_embedding_path).
- subjects: maintains subject definitions with faculty credentials (subject_id, subject_name, faculty_name).
- attendance_log: records each attendance event (log_id, roll_no, subject_id, timestamp, session_date, status).

Foreign key constraints enforce referential integrity between tables. Pandas Data Frames are used to query aggregated statistics, compute per-student attendance percentages, and identify students below the required 75% threshold. Reports are exported to CSV via pandas.DataFrame.to_csv() and to PDF using the Report Lab library.

Tkinter GUI Design

The graphical user interface is developed using Python's tkinter toolkit supplemented by the themed widget set, eliminating any external GUI framework dependency. The interface provides four primary views navigable from a sidebar menu:

- Home Dashboard: displays system status, total enrolled students, and today's session summary.
- Enroll Student: captures three reference photographs, generates and stores VGG-Face embeddings for a new student.
- Take Attendance: initiates the live camera session with real-time bounding box overlay and recognized name display; saves finalized attendance to SQLite on faculty confirmation.
- Reports: generates subject-wise attendance summaries and exports CSV or PDF files.

OpenCV's VideoCapture class provides the webcam feed, with frames displayed inside a Tkinter Label widget via PIL ImageTk conversion. A background threading model prevents GUI freeze during continuous frame capture by running the recognition pipeline in a daemon thread.

IMPLEMENTATION

The system was implemented and tested on a Dell Inspiron laptop with Intel Core i5-10th Gen processor, 8 GB RAM, and a 720p integrated webcam running Ubuntu 22.04 LTS. The Python 3.10 environment was managed using venv with the following primary dependencies: OpenCV 4.8.0, DeepFace 0.0.79, MTCNN 0.1.1, Pillow 10.0, ReportLab 4.0, Pandas 2.0.3, and Numpy 1.25.

Enrollment Module

Faculty initiates enrollment by entering the student's roll number and full name in the Enrollment form. The system captures three frames from the webcam, detects the largest face in each frame via MTCNN, aligns it to 224×224, and extracts the VGG-Face embedding. The three embeddings are averaged and the resulting vector is stored in a dedicated embeddings/ directory as a .pkl file named by the student's roll number. The student record is simultaneously inserted into the students SQLite table.

Recognition and Attendance Recording

During attendance recording, the system continuously reads frames from OpenCV's VideoCapture at a target rate of 15 FPS. Each frame

undergoes MTCNN detection to identify bounding boxes and landmarks. For computational efficiency, embedding extraction is performed only once every 10 frames per tracked face region, using a lightweight bounding box overlap check to determine whether a face in the current frame corresponds to one already under evaluation.
Recognized student names are displayed in the live preview as green-bordered labels; unrecognized faces display red borders labeled Unknown. A running detected_set accumulates confirmed rollnumbers throughout the session. On faculty confirmation, the set is committed to the attendance table with the current timestamp and session date, and unmarked students receive an Absent record.

Report Generation

Reports are generated on demand by querying the attendance log table through a Pandas SQL read. Two report types are supported. A Session Report lists all students with their Present/Absent status for a selected session date and subject. A Cumulative Report calculates each student's attendance percentage across all sessions for a subject, in largest students below the 75% threshold in red, and includes the total session count. Both report types are exported to CSV and PDF. The PDF layout uses Report Lab's Table and Paragraph elements styled to resemble institutional register formats.

RESULTS AND DISCUSSION

The system was evaluated against a dataset of 120 students from two departments, captured across 30 independent attendance sessions over a four-week period. Each session involved varying degrees of ambient lighting (natural daylight, fluorescent overhead, and dim corridor lighting) and slight head pose variation (up to ±15° yaw). Table 1 summarizes recognition performance metrics.

Table -1: Recognition Performance Metrics

Metric	Value
Recognition Accuracy	92%
Precision	93.2%
Recall	95.1%
F1-Score	94.1%
False Acceptance Rate	2.8%
False Rejection Rate	5.3%
Avg. Session Time (60 students)	48 seconds
Manual Roll-call Time (60 students)	~6 minutes

The overall recognition accuracy of 92% was achieved without any GPU acceleration, demonstrating the viability of CPU-based deep learning inference for classroom-scale deployments. Recognition performance degraded slightly (to 91.3%) under dim corridor lighting below 100 lux, suggesting that supplementary LED lighting would benefit installations in poorly lit rooms.
A comparative evaluation against LBPH-based recognition (OpenCV built-in) on the same dataset yielded 78.4% accuracy under the same conditions, confirming the substantial advantage of deep embedding approaches over traditional handcrafted feature methods. The attendance session for a class of 60 students completed in an average of 48 seconds, compared to approximately 6 minutes for manual roll-call, representing an 87% time reduction.
The SQLite database incurred negligible storage overhead, with 30 sessions of 120 students generating approximately 2.1 MB of attendance data. Enrollment embeddings for 120 students totaled 58 MB in serialized Pickle format.

Comparison with Existing Systems

Method	Accuracy	Cloud	Offline
Eigenfaces (PCA)	71.2%	No	Yes
LBPH (OpenCV)	78.4%	No	Yes
AWS Recognition API	97.1%	Yes	No
Proposed (VGGFace)	92%	No	Yes

Table -2: Comparison of Face Recognition Methods

Table 2 demonstrates that the proposed system achieves near-cloud-level accuracy while operating entirely offline. Cloud API solutions such as AWS Recognition offer marginally higher accuracy (97.1%) but require internet connectivity, incur per-call fees, and transmit biometric data to external servers, all of which are undesirable for institutional deployments under data privacy regulations.

CONCLUSIONS

This paper presented a Python-based automated student attendance system that integrates MTCNN face detection, VGG-Face deep embedding generation, cosine similarity matching, and SQLite storage within a Tkinter desktop application. The system achieves 92% recognition accuracy on a 120-student dataset under realistic classroom conditions without any cloud service dependency, internet connectivity, or specialized hardware.

The proposed architecture offers several advantages over both traditional manual methods and cloud-based alternatives: it finis the proxy attendance, reduces session time by 87%, operates fully offline preserving student biometric privacy, and requires no recurring service fees. The Tkinter GUI provides an accessible interface for nontechnical faculty, and the integrated report generation module simplifies administrative tasks.

Future work will investigates the integration of antispoofing measures to prevent photographbased attacks, extension to multi-camera classroom set in ups to handle larger batch sizes simultaneously, and exploration of lightweight MobileNet-based embedding models to further reduce inference time on resource-constrained devices.

ACKNOWLEDGEMENT

The authors express sincere gratitude to the faculty and students of Anjuman-I-Islam's I.H. Oriya Polytechnic for their cooperation in dataset collection and system testing. The authors also thank the Department of Computer Engineering for providing laboratory resources for experimental evaluation.

REFERENCES

[1] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 1991, pp. 586–591.

[2] T. Ahonen, A. Hadid, and M. Pietikainen, "Face Description with Local Binary Patterns: Application to Face Recognition," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 28, no. 12, pp. 2037–2041, Dec. 2006.

[3] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep Face Recognition," British Machine Vision Conference (BMVC), 2015, pp. 1–12.

[4] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," Proc. IEEE CVPR, 2014, pp. 1867–1874.

[5] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," IEEE Signal Processing Letters, vol. 23, no. 10, pp. 1499–1503, Oct. 2016.

[6] S. I. Serengil and A. Ozpinar, "HyperExtended LightFace: A Facial Attribute Analysis Framework," Proc. IEEE ICEET, 2021, pp. 1–4.

[7] R. Jain and P. Sharma, "Cloud-Based Biometric Attendance Management Using AWS

Rekognition," Int. Journal of Advanced Computer Science and Applications, vol. 13, no. 4, pp. 211–218, 2022.

[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," Proc. IEEE CVPR, 2005, vol. 1, pp. 886–893.

[9] A. Ng, "Machine Learning Yearning," Technical Book Draft, deeplearning.ai, 2018.

