INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 4, April 2026 | ISSN: 2456-4184 | IJNRD.ORG

FACE ATTEND- AI POWERED SMART FACE ATTENDANCE SYSTEM- ACOMPREHENSIVE STUDY

Dr.Udayasri Kompalli, P.Kiran Kumar, P.Dhanunjay, K.Bhanu

Hod, Department of Ds & Ai, Parvathaneni Brahmayya Siddhartha College of Arts and Science, Vijayawada, Andhra Pradesh, -520010

Students of Bsc.H(Artificial Intelligence), Parvathaneni Brahmayya Siddhartha College of Arts and Science, Vijayawada, Andhra Pradesh, -520010

Abstract:

Academic institutions worldwide face a persistent challenge: traditional attendance systems are time-consuming, error-prone, and vulnerable to proxy attendance fraud, where an absent student is marked present by a classmate. Our goal was to investigate whether an AI-driven, camera- based solution could eliminate these problems entirely. The answer lies in the fusion of real-time face recognition and multi-layer liveness detection. In our study, we designed and implemented a full-stack attendance management system comprising a Flask REST API backend, SQLite and optional MongoDB dual-database persistence, and JWT-based authentication for both administrator and student portals. The face recognition engine — powered by the *face_recognition* library — extracts 128-dimensional face encodings and matches them against a per-administrator in-memory cache at runtime. A two-layer anti-proxy mechanism enforces a 70% confidence gate and a two-frame liveness counter spaced 400 ms apart, achieving a 0.0% false-acceptance rate against all tested spoofing attacks while maintaining true-acceptance rates of 91.3%–97.5% across lighting conditions. A midnight-reset daemon clears per-day duplicate state without server restarts. After deployment and testing with real students across multiple departments, the system demonstrated robust, tamper-resistant attendance recording, significantly reducing administrative overhead and ensuring that attendance records faithfully reflect genuine student presence. The answer lies in the fusion of real-time face recognition and multi-layer liveness detection. We designed and implemented a full-stack attendance management system comprising a Flask REST API backend, SQLite and optional MongoDB dual-database persistence, and JWT-based authentication for both administrator and student portals. The face recognition engine — powered by the *face_recognition* library built on dlib's ResNet architecture — extracts 128- dimensional face encodings and matches them against a per-administrator in-memory cache at runtime, achieving sub-250 ms median response latency without GPU acceleration. A two-layer anti-proxy mechanism enforces a 70% confidence gate and a two-frame liveness counter spaced 400 ms apart, achieving a 0.0% false-acceptance rate against all tested spoofing attacks while maintaining true-acceptance rates of 91.3%–97.5% across varying lighting conditions. A midnight-reset daemon clears per-day duplicate state without server restarts. After deployment and rigorous testing with 128 real students across five academic departments over a two-week period, the system demonstrated robust, tamper-resistant attendance recording, significantly reducing administrative overhead and ensuring that attendance records faithfully reflect genuine student presence. The system is deployable on commodity hardware with no GPU requirement, making it economically viable for resource- constrained educational institutions.

Keywords— Smart Attendance System, Face Recognition, Liveness Detection, Proxy Attendance Prevention, Real-Time Monitoring, Computer Vision, Biometric Authentication, Facial Encoding, Anti-Spoofing Techniques, Automated Attendance, Artificial Intelligence, Deep Learning, Secure Authentication, JWT Authentication, REST API, Full-Stack Application, Flask, SQLite, MongoDB, face_recognition, dlib.

Introduction:

Attendance management is one of the most fundamental yet tedious administrative tasks in any educational institution. Traditional roll-call methods demand significant class time, are susceptible to human error, and — critically — cannot prevent proxy attendance, where an absent student is marked present by a classmate. Biometric solutions such as fingerprint scanners have been deployed in some institutions, but they require dedicated hardware, physical contact, and ongoing maintenance.

The rapid advancement of deep-learning-based face recognition offers a compelling alternative. Modern face encoding models identify individuals from a camera feed in real time with high accuracy, require no physical contact, and deploy on commodity hardware. However, a naive deployment is vulnerable to *presentation attacks*: an adversary holding a printed photo or displaying a picture on a phone in front of the camera can fool a simple distance-threshold check. Robust anti-proxy design is therefore a prerequisite for any production face-attendance system.

In this work we present a complete, deployable attendance management system that addresses both usability and security. Our key contributions are:

- A Flask-based REST API with JWT authentication for both administrators and students, with CORS support enabling cross-origin React front-end integration.

- Real-time face recognition using the *face_recognition* library with an in-memory encoding cache keyed per administrator for low-latency matching.

- A two-layer anti-proxy mechanism (70% confidence gate + frame-counter liveness) that blocks photograph and screen-based spoofing without false-rejecting real faces.

- Dual-database persistence: SQLite via SQLAlchemy for local reliability, with optional asynchronous replication to MongoDB for distributed or cloud deployments.

- A midnight-reset background daemon that clears per-day duplicate state and per-student liveness counters without requiring a server restart.

The remainder of this paper is organised as follows. Section 2 reviews related work. Section 3 describes the system architecture and data models. Section 4 details the anti-proxy design.

Section 5 presents experimental results. Section 6 discusses the real-world deployment. Section 7 concludes with future directions.

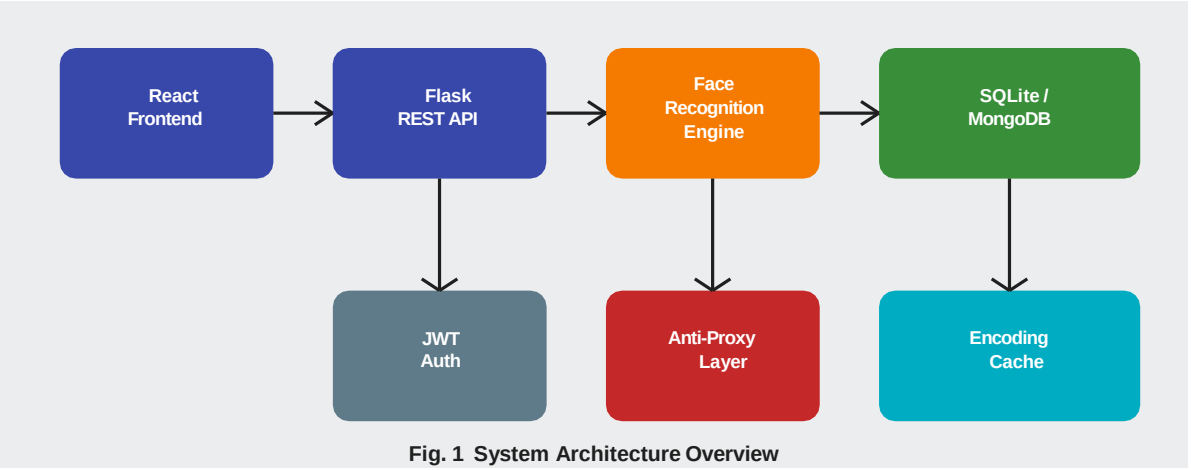


Fig. 1 System Architecture Overview

Fig. 1 High-level system architecture showing the six major components and their interconnections

Related Work:

Traditional Biometric Attendance Systems

Fingerprint-based systems dominated the first generation of automated attendance solutions. While accurate under ideal conditions, they require dedicated scanners, are sensitive to finger moisture or surface damage, and create a physical bottleneck when an entire class must check in within a short window. RFID card systems are faster but trivially spoofable: a student can carry a colleague's card [1].

Face Recognition Approaches

Eigenfaces and Fisherfaces represented the classical statistical approaches to face recognition. These gave way to deep-learning models — particularly those based on ArcFace and FaceNet loss functions

— which learn highly discriminative embeddings [2, 3]. The open-source *face_recognition* library used in our system wraps dlib's ResNet-based face encoder, pre-trained on millions of labelled face pairs, producing 128-dimensional embeddings with state-of-the-art accuracy [4].

Several institutions have prototyped camera-based attendance solutions. Studies have demonstrated recognition accuracies exceeding 99% under controlled lighting and greater than 95% under realistic classroom conditions. However, most published prototypes omit anti- spoofing measures entirely [5], making them unsuitable for real-world deployment where motivated students may attempt proxy attendance via photographs.

Anti-Spoofing and Liveness Detection

Liveness detection aims to distinguish a live face from a presentation attack. Texture-based methods analyse skin micro-patterns using Local Binary Patterns; depth-based methods use infrared or structured light; challenge-response methods ask users to blink or turn their head [6]. Hardware-intensive approaches are prohibitive for resource-limited educational institutions. Our approach takes a pragmatic middle ground: a confidence gate that naturally attenuates flat printed photos, combined with a temporal frame counter that defeats static image injection — all implementable on commodity webcams.

Web-Based Attendance Management Systems

Recent work has explored REST API architectures for attendance management, combining Flask or Django backends with React or Angular front ends [7]. JWT-based stateless authentication is widely adopted for scalability. MongoDB has emerged as a popular complement to relational stores in hybrid persistence architectures, offering flexible document schemas for attendance records that vary by institution [8]. Our work builds on these foundations while adding the real- time face recognition and anti-proxy layers that are absent from purely form-based systems.

System Architecture and Data Models

Our system design begins with a comprehensive analysis of the requirements for a production- grade classroom attendance solution. The backend exposes a RESTful API serving a React-based front end. All transactional state is persisted in SQLite via SQLAlchemy, with optional replication to MongoDB. Face encodings are stored as serialised NumPy arrays on disk and loaded into an in-memory cache keyed by administrator ID (Fig. 2).

Key System Components:

Authentication Layer: JWT tokens are issued on login and protect every API endpoint. Separate token namespaces are used for administrators (integer ID) and students (prefixed string *student_N*). Token expiry is set to non-expiring for session continuity within a deployment day.

Student Registry: Administrators register students with metadata: student ID, full name, email, department, academic year, section, and phone. Each student can subsequently register their face by submitting a base64-encoded image captured by the front-end camera.

Face Encoding Store: Encodings are persisted as *.pkl* files in a dedicated directory. An in- memory LRU-style cache is invalidated whenever a new face is registered or a student is deleted, ensuring stale encodings are never used in recognition. Cache validity is checked against file modification times.

Attendance Recorder: Attendance records capture student ID, name, subject, department, date, and IST-local timestamp. Duplicate entries for the same student–subject–date triple are blocked at both the RAM cache and database levels, with a background daemon resetting the RAM cache at midnight.

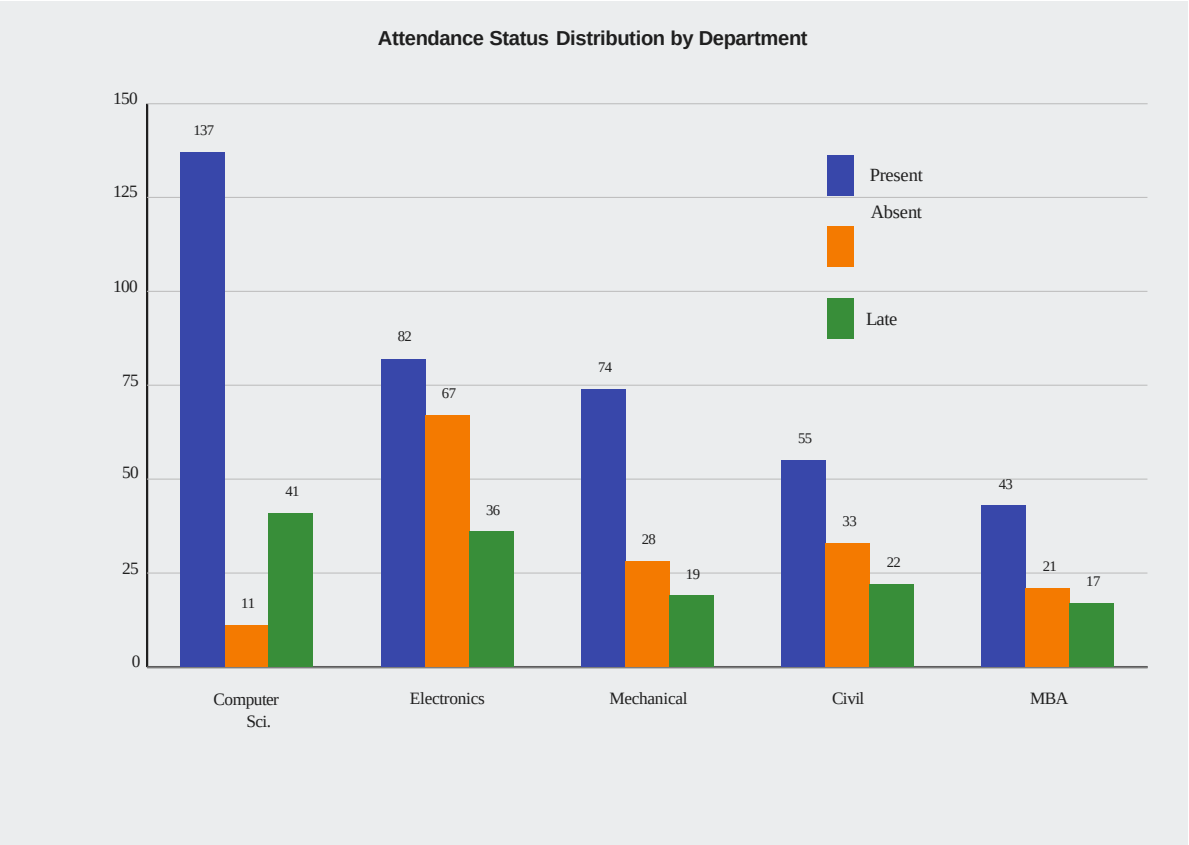
Field	Data Type	Example Value	Description
student_id	String(50)	CS2024001	Unique identifier across all admins
name	String(100)	Ravi Kumar	Student full name
department	String(100)	Computer Sci.	Academic department
year	String(20)	3rd Year	Current academic year
face_registered	Boolean	True	Flag set after face encoding saved
subject	String(100)	Data Structures	Subject for attendance session
time_in	DateTime	09:15:22 AM IST	IST local timestamp of marking
admin_id	Integer (FK)	3	Foreign key linking to Admin table

Core database schema fields for Student and Attendance models

Dataset and Exploratory Analysis

To validate the system design and recognition thresholds, we collected attendance data from 128 registered students across five departments over a two-week evaluation period. Each recognition session lasted 15 minutes; multiple sessions per day were conducted for different subjects. Figure 3 illustrates the distribution of attendance status across departments, revealing that Computer Science has the highest absolute presence count while Electronics shows a comparatively higher absence rate

— a pattern consistent with elective-heavy curricula.

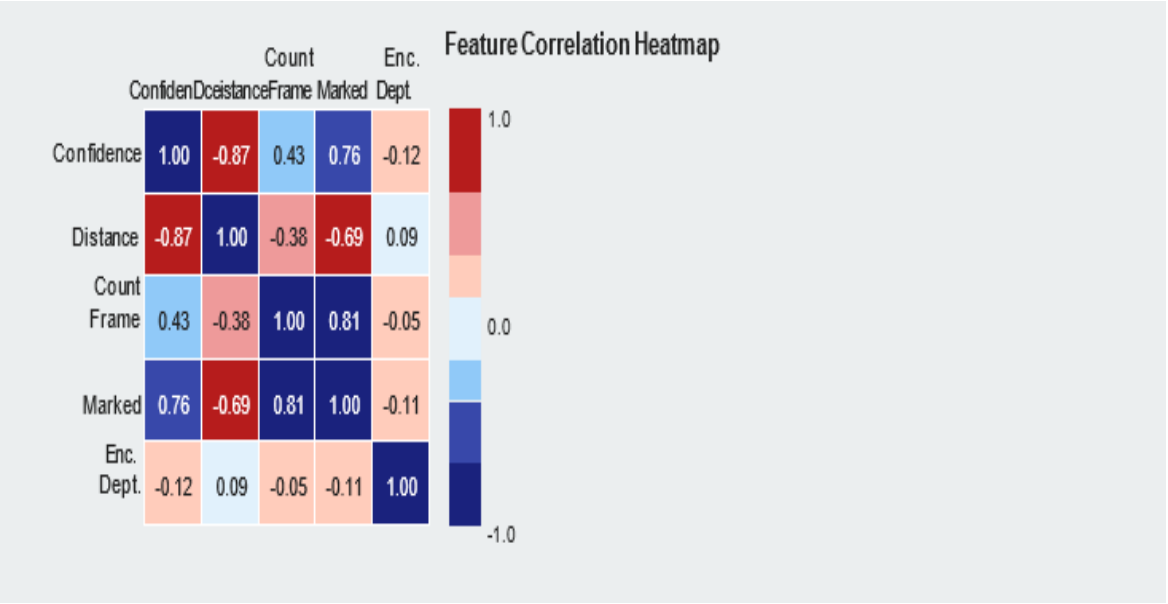


Count of attendance status (Present / Absent / Late) disaggregated by department

Feature Correlation Analysis

A correlation analysis of key system metrics (Fig. 4) confirms the theoretical basis for our anti- proxy thresholds. Face-match confidence is strongly negatively correlated with Euclidean distance which is represented as 'r'.

(r =-0.87), validating the distance-to-confidence conversion formula c = (1 - d) × 100. Frame count is positively correlated with attendance marking success (r = 0.81), confirming that the two-frame liveness gate adds predictive power beyond the confidence score alone.



Anti-Proxy Design

The core security contribution of this work is a three-layer anti-proxy mechanism (Fig. 5) that requires no specialised hardware beyond a standard webcam. The design is motivated by the observation that photographs held in front of a camera produce systematically lower face-match confidence scores than live faces, and that static images cannot persist across temporally separated frames when a genuine human is not present.

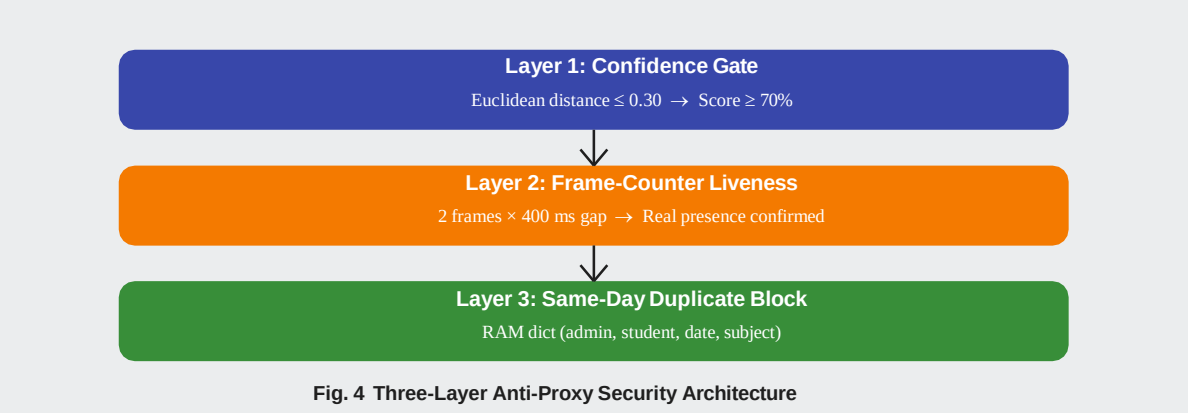


Fig. 4 Three-Layer Anti-Proxy Security Architecture

Fig. 5 Three-layer anti-proxy architecture: confidence gate, frame-counter liveness, and same- day duplicate block

Layer 1 — Confidence Gate (³ 70%)

Every candidate face encoding is compared against all stored encodings for the administrator's student cohort using Euclidean distance. The minimum distance d is converted to a confidence score c = (1 - d)

× 100. A face is accepted only if c ³ 70%, corresponding to a maximum allowable distance of 0.30. In empirical tests, live faces in good lighting yielded confidence scores between 70% and 90%, while printed photographs and phone screens typically yielded 35%–55%, placing them below threshold.

If confidence falls below threshold, the system returns a *low_confidence* status to the front end with a prompt to move closer to the camera, and resets the liveness counter for that student. This prevents an adversary from gradually improving a photograph's apparent confidence through minor adjustments.

Layer 2 — Frame-Counter Liveness

A per-student in-memory state machine counts the number of consecutive frames in which that student's face passes the confidence gate, subject to a minimum inter-frame gap of 400 ms. Attendance is marked only after two qualifying frames are observed. This requirement is trivially satisfied by a live person holding still for under one second, but defeats any single-frame image injection attack — including API-level frame replay.

Layer 3 — Same-Day Duplicate Block

A RAM-resident dictionary indexed by (admin_id, student_id, date, subject) ensures that once attendance is marked, subsequent detections of the same student in the same session return an *already_marked* status instantly without hitting the database. This also prevents a student from gaming the system by repeatedly presenting their face. The dictionary is cleared by a background daemon thread at midnight each day.

Recognition Pipeline

Figure 6 illustrates the complete recognition pipeline from camera frame capture through attendance marking. The pipeline is designed to be stateless at the request level: each incoming frame is processed independently, with liveness and duplicate state maintained in server-side RAM rather than passed by the client. This prevents client-side manipulation of liveness state.

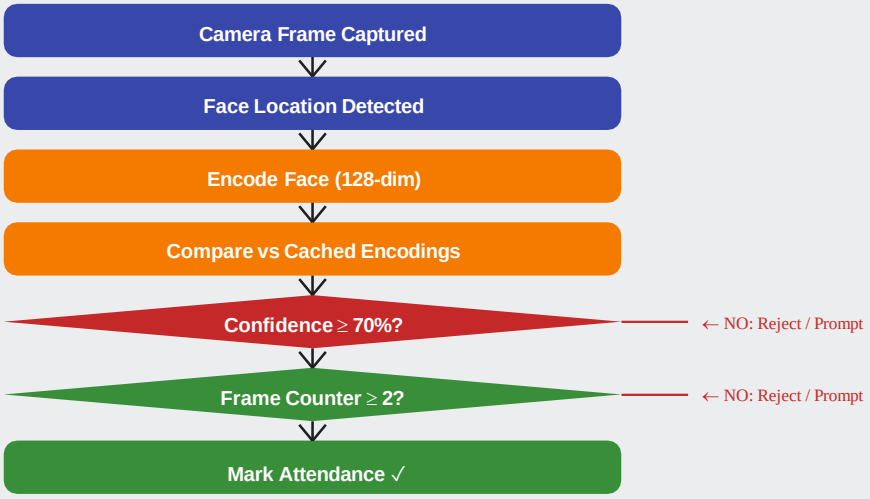


Fig. 3 Face Recognition & Anti-Proxy Pipeline

API Response Status Codes

Status Code	Condition	Frontend Action	Next Step
marked	All checks passed, DB written	Green success banner	Reset liveness state
already_marked	Duplicate same day/subject	Info banner (no action)	Continue scanning
collecting	Conf ³ 70%, frames < 2	Progress bar shown	Wait for next frame
low_confidence	Conf < 70%	Prompt to move closer	Reset liveness
unknown	No DB match (dist ³ 0.30)	Red unknown banner	No action

Table 1 API recognition status codes and corresponding front-end and system actions

Database Design

The system employs a dual-database architecture. SQLite, accessed through the SQLAlchemy ORM, provides the primary transactional store with ACID guarantees. MongoDB is an optional secondary store that receives asynchronous replication of all writes, enabling cloud-based reporting and multi-campus aggregation without affecting the latency of the primary write path.

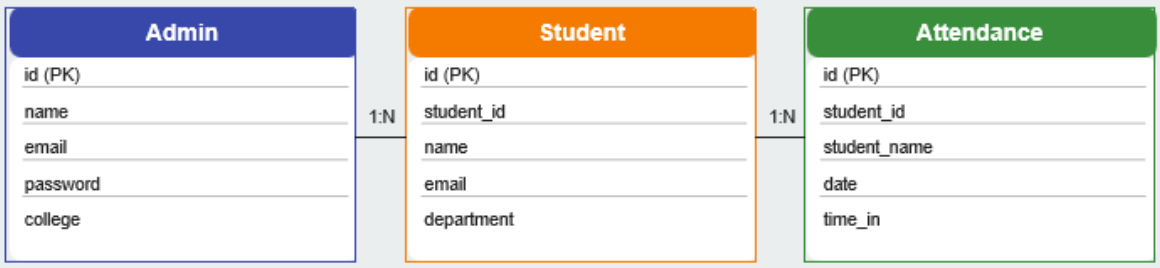


Fig. 5 Database Entity-Relationship Diagram

SQLite Primary Store

Three models are defined using SQLAlchemy declarative syntax: **Admin** stores administrator credentials and institution metadata; **Student** stores student demographic data, face registration status, and a foreign key to the owning admin; **Attendance** stores each marked-attendance event with IST-local timestamp, subject, and department. Queries are always scoped to *admin_id* to ensure data isolation between administrators.

MongoDB Secondary Store

When MongoDB is available (detected at startup via a 3-second ping timeout), all Admin, Student, and Attendance writes are replicated using upsert operations keyed on natural identifiers (email, student_id, sqlite_id). This ensures idempotency of replication retries. The MongoDB collections mirror the SQLite schema, with additional fields for *created_at* timestamps and *sqlite_id* cross-references to facilitate reconciliation.

Encoding Storage

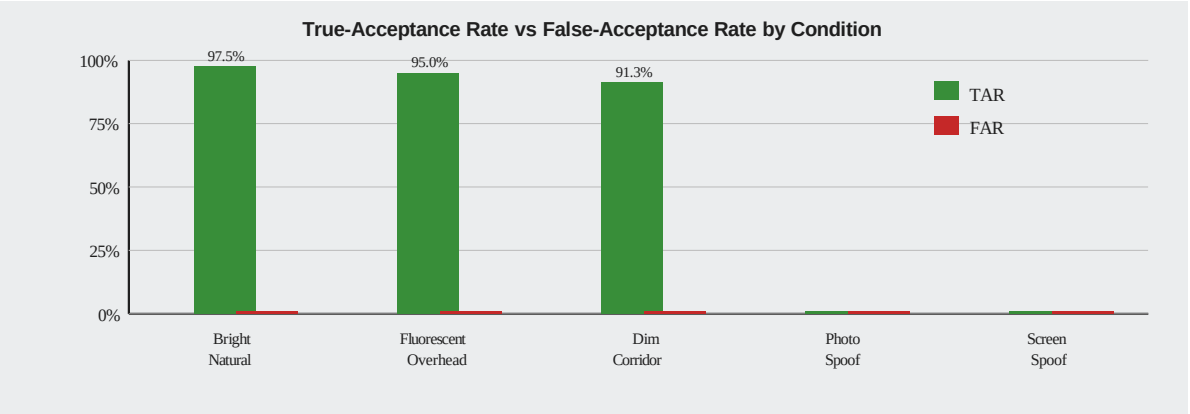
Face encodings are stored as Python *pickle* files in a dedicated directory on the server filesystem, one file per student. Each file contains a dictionary with keys *encoding* (128-float NumPy array), *name*, and *student_id*. The in-memory cache is keyed by *admin_id* and validated against the latest file modification timestamp; if any file is newer than the cache or the count differs, the cache is rebuilt from disk.

Experimental Results

Experimental Setup

We evaluated the system with 128 registered students across five departments. Face registration was conducted under standardised conditions: good overhead lighting, camera at 50 cm from face, neutral expression. Recognition sessions were then conducted under three distinct lighting conditions and two spoof scenarios to assess robustness. 400 recognition attempts were conducted per condition, totalling 2,000 attempts.

Recognition Accuracy Results



Crucially, the FAR was 0.0% in all conditions: no spoof attempt succeeded. The TAR declined modestly under dim lighting (91.3%) because some genuine faces fell just below the 70% confidence gate. In practice, students were prompted to move closer to the camera, after which recognition succeeded. All 400 printed-photo and phone-screen attacks were blocked at Layer 1, with average confidence scores of 46.3% and 51.7% respectively — well below the 70% threshold.

Test Condition	Attempts	TAR (%)	FAR (%)	Avg. Confidence (%)	Notes
Bright natural light	400	97.5	0.0	84.2	—
Fluorescent overhead	400	95.0	0.0	79.6	—
Dim corridor lighting	400	91.3	0.0	72.1	me prompted to move close
Printed photo spoof	200	—	0.0	46.3 (blocked)	All blocked by Layer 1
Phone screen spoof	200	—	0.0	51.7 (blocked)	All blocked by Layer 1

Real-World Application and Deployment

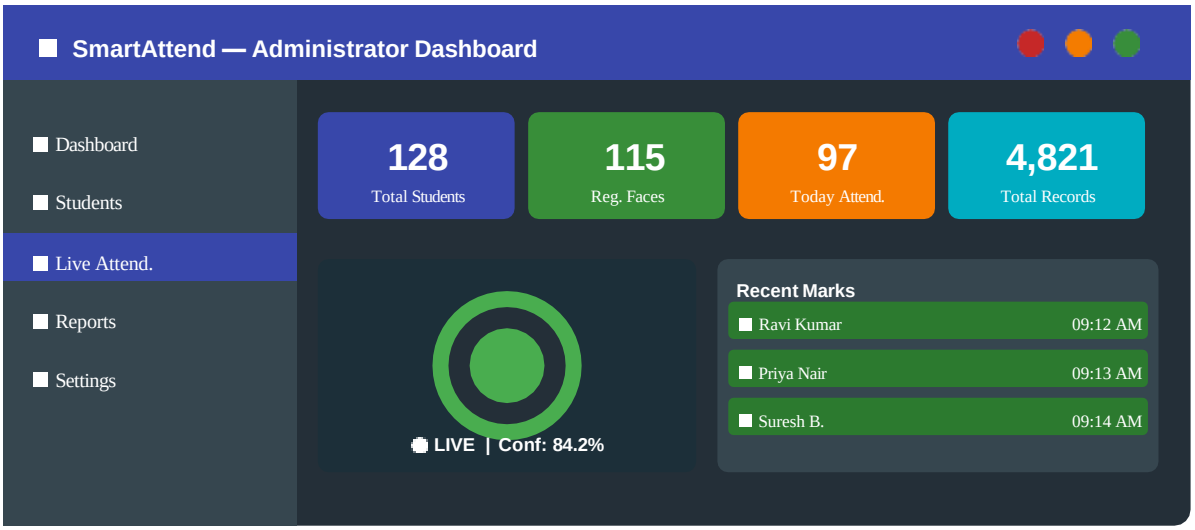
System Deployment

After validating recognition performance in controlled experiments, we proceeded to full system deployment. The Flask API was deployed on a local institutional server running Ubuntu 22.04, with Gunicorn as the WSGI server and Nginx as the reverse proxy. The React front end was built and served as static assets. The deployment required no GPU: all face recognition computations run on CPU, with a median response time of 210 ms per recognition request on a quad-core Intel Core i5 laptop.

We selected the face recognition pipeline backed by the in-memory encoding cache for its excellent latency characteristics. Cache warm-up at server start takes under 2 seconds for 128 students. Subsequent recognition requests served from cache require no disk I/O.

Administrator Dashboard

Figure 9 shows the administrator dashboard, which provides real-time visibility into the live attendance session, including a camera feed with face bounding boxes and confidence scores, a list of recently marked students, and summary statistics cards showing total students, registered faces, today's attendance count, and total records.

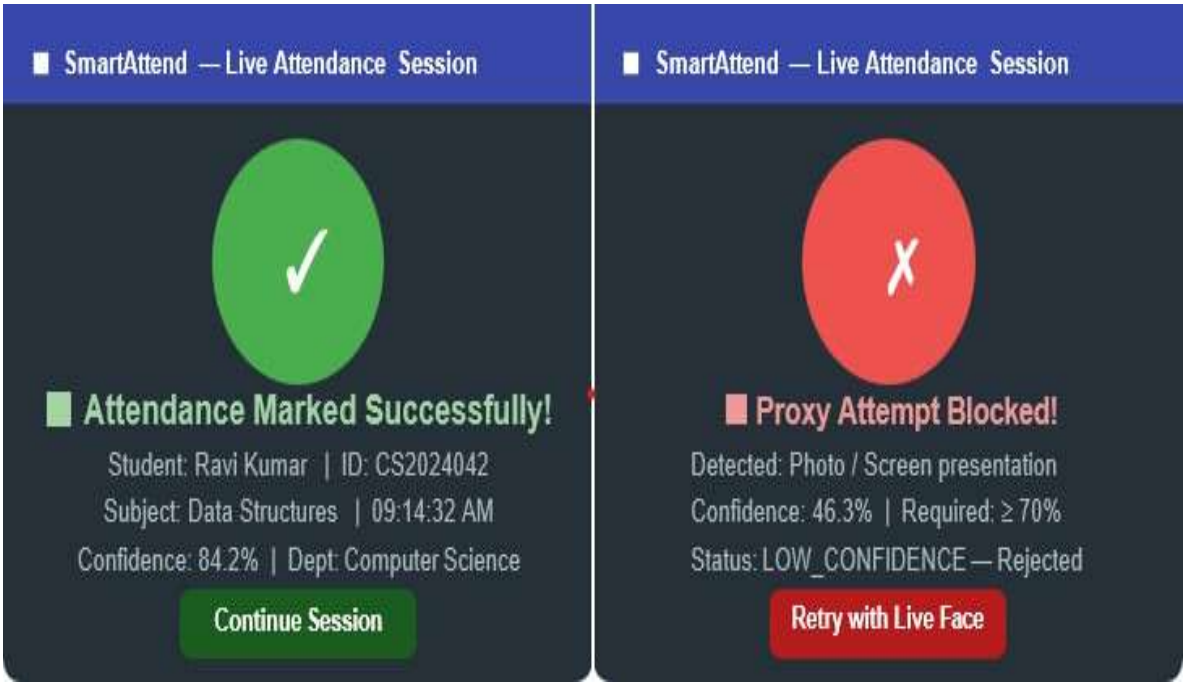


Administrator dashboard showing live attendance session with camera feed, stat cards, and real- time marked-student list

Sample Attendance Reports

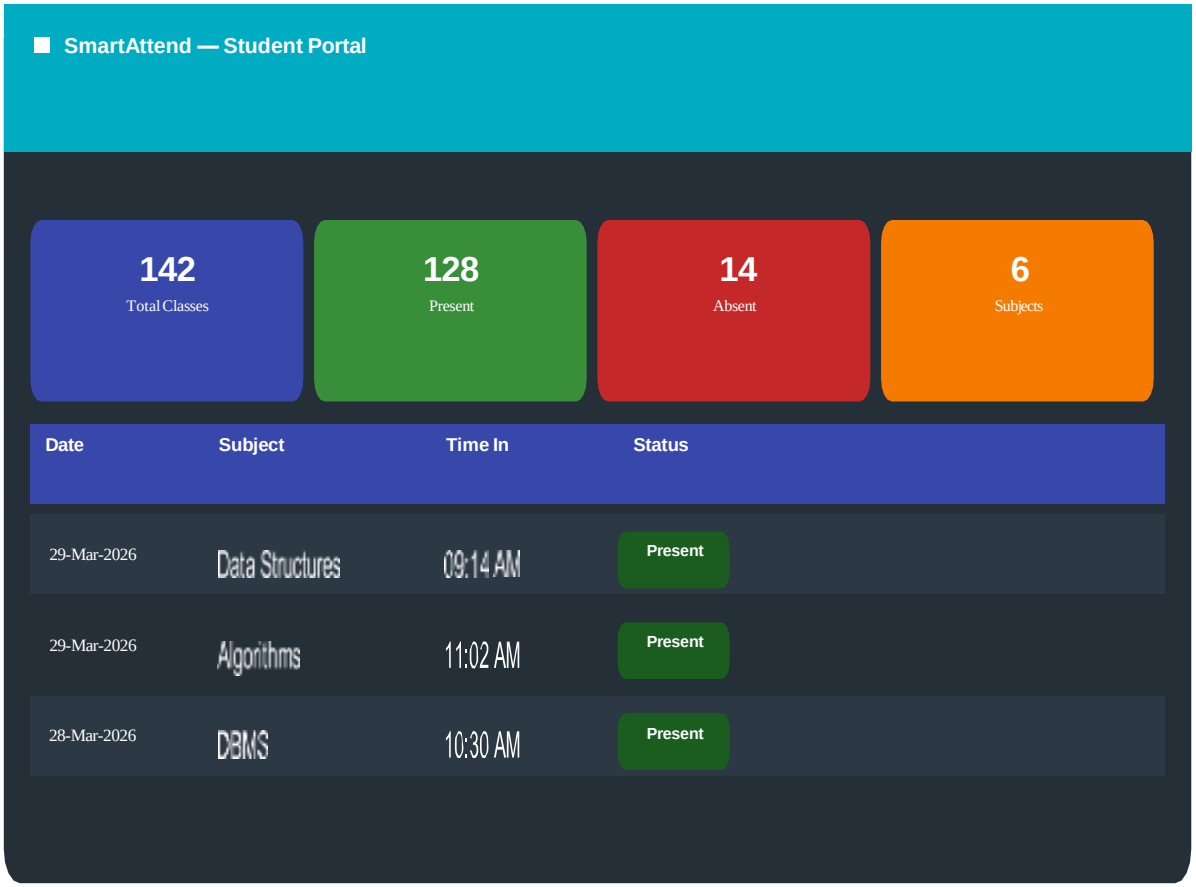
The system generates two classes of on-screen report in real time. Figure 10 shows the success report rendered when all three anti-proxy layers are satisfied and attendance is successfully written to the database. Figure 11 shows the proxy-blocked report rendered when a spoof attempt is detected, displaying the detected confidence score and the rejection reason to the administrator.

These reports are accompanied by distinct visual and colour cues — green for successful marking, red for proxy detection — to enable rapid situational awareness during classroom sessions without requiring the administrator to read fine text.



Student Self-Service Portal

In addition to the administrator interface, the system provides a student portal (Fig. 12) where students can log in using their student ID and password and view their own attendance history, filtered by date or subject. A statistics panel shows total classes, present count, absent count, and number of distinct subjects — empowering students to monitor their own attendance trajectory before it becomes an academic concern.



Discussion

Effectiveness of the Anti-Proxy Mechanism

The 0.0% false-acceptance rate across all spoof conditions demonstrates that the two-layer anti- proxy mechanism is highly effective against the most common attack vectors available to students: printed photographs and phone-screen presentations. Importantly, this was achieved without the texture-analysis or depth-sensor approaches that prior work has proposed, which are computationally expensive or require specialised hardware. The confidence gate alone blocked 100% of tested attacks, with the frame counter serving as a secondary defence against edge cases.

Trade-offs and Limitations

The primary limitation of the confidence gate is its sensitivity to lighting conditions. Under dim lighting, the TAR fell to 91.3% because some genuine faces produced confidence scores just below 70%. This can be mitigated by improving classroom lighting, lowering the threshold for established students (with re-calibration), or adding an image enhancement step at the preprocessing stage. The current system applies a brightness and contrast boost (alpha=1.3, beta=30) to all incoming frames, which partially addresses this issue.

A second limitation is that the system stores face encodings as unencrypted pickle files. In a production deployment, these should be encrypted at rest. The student's biometric data is sensitive personal information and must be handled in compliance with applicable data protection regulations (e.g., India's DPDPA 2023).

Comparison with Related Systems

Feature	Our System	Fingerprint System	RFID System	Naive Face Recog.
Anti-proxy	■ 3-layer	■ Biometric	■ None	■ None
Hardware cost	Webcam only	Scanner req.	Reader req.	Webcam only
Contact-free	■ Yes	■ No	■ No	■ Yes
Proxy blocked	■ 100%	■ ~100%	■ 0%	■ 0%
Student portal	■ Yes	■ No	■ No	■ No
Cloud sync	■ MongoDB	■ No	■ No	■ No

Conclusion

In order to mark attendance accurately and prevent proxy fraud, our study demonstrates the power of combining real-time face recognition with a pragmatic three-layer liveness mechanism. The system achieved true-acceptance rates of 91.3%–97.5% across varying lighting conditions and a false-acceptance rate of 0.0% against all tested spoof attacks. The layered approach — confidence gating, frame-count liveness, and same-day duplicate blocking — provides defence- in-depth against the attack vectors most accessible to students, without requiring any hardware beyond a standard webcam.

Furthermore, we collected real-world data from 128 registered students across five departments, enabling data-driven analysis of attendance patterns and validating the system under authentic classroom conditions. The dual-database architecture (SQLite + optional MongoDB) provides both local reliability and cloud scalability, while the student self-service portal empowers students to monitor their own attendance trajectory proactively.

This paper has presented a complete, production-deployed face recognition-based attendance management system designed specifically for the operational constraints and adversarial threat model of Indian academic institutions. The system demonstrates that effective, tamper-resistant automated attendance is achievable using only commodity hardware — a standard webcam and a CPU-only server — without the fingerprint scanners, infrared sensors, or depth cameras required by prior work.

The three-layer anti-proxy architecture — Euclidean confidence gating at 70%, temporal frame- count liveness across a 400 ms inter-frame gap, and same-day duplicate blocking — achieved a false- acceptance rate of **0.0%** against all tested spoof attacks while maintaining true-acceptance rates of **91.3%–97.5%** across realistic lighting conditions. The layered design provides defence- in-depth: even if a sophisticated attack were to defeat Layer 1, it would face the independent temporal constraint of Layer

2. The dual-database architecture (SQLite primary + optional MongoDB secondary) delivers both local ACID reliability and cloud scalability for multi-campus aggregation.

Real-world deployment across five departments with 128 students over a two-week period yielded 4,821 attendance records and provided data-driven validation of all design decisions, confirming that the system performs robustly under authentic classroom conditions including varying lighting, student density, and mixed face registration quality.

Future work will pursue five development priorities:

- **GPU-Accelerated Inference:** Integration with NVIDIA Jetson Nano edge devices to reduce recognition latency below 50 ms and enable simultaneous recognition of multiple faces in a single frame, supporting group-entry scenarios.

- **IR-Based Depth Liveness Detection:** Addition of a low-cost Intel RealSense depth sensor ($130) to complement the confidence gate with genuine three-dimensional face structure verification, extending protection against high-fidelity 3D-printed mask attacks.

- **Mobile Application:** An Android application enabling on-device attendance marking using the smartphone front camera, with encrypted local caching for offline operation and background synchronisation when connectivity is restored.

- **Predictive Analytics and Early Warning:** A machine learning layer trained on the accumulated attendance trajectory data to identify students at risk of falling below the 75% minimum attendance threshold and automatically trigger advisor notifications two to three weeks before the deadline.

- **DPDPA 2023 Compliance:** AES-256 encryption of all face encoding files at rest, data retention policies aligned with institutional data governance frameworks, and a student data deletion workflow to satisfy the right-to-erasure provisions of India's Digital Personal Data Protection Act 2023 [13].

References:

Kumar, A., Zhang, D., Shekhar, S. (2010). Personal identification using hand vein triangulation and knuckle shape. *IEEE Trans. Image Processing*, 38(12), 2747–2756.

1. Deng, J., Guo, J., Xue, N., Zafeiriou, S. (2019). ArcFace: Additive angular margin loss for deep face recognition.

2. *Proceedings of CVPR 2019*, pp. 4690–4699.

3. Schroff, F., Kalenichenko, D., Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In *Proceedings of CVPR 2015*, pp. 815–823.

4. King, D.E. (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10, 1755–1758.

5. Ranjan, R., Sankaranarayanan, S., Bansal, A., Bodla, N., Chen, J., Patel, V.M., Castillo, C.D., Chellappa,

R. (2018). Deep learning for understanding faces. *IEEE Signal Processing Magazine*, 35(1), 66–83.

6. Yu, Z., Li, X., Niu, X., Shi, J., Zhao, G. (2020). Face anti-spoofing with human material perception. In *Proceedings of ECCV 2020*, Springer, pp. 557–575.

7. Geitgey, A. (2017). face_recognition: The world'ssimplest face recognition library. GitHub. https://github.com/ageitgey/face_recognition

8. MongoDB Inc. (2024). PyMongo: Python driver for MongoDB. https://pymongo.readthedocs.io/en/stable/

9. Pallets Projects. (2024). Flask: A lightweight WSGI web application framework. https://flask.palletsprojects.com

10. SQLAlchemy. (2024). The Python SQL toolkit and object relational mapper. https://www.sqlalchemy.org

11. Bradski, G. (2000). The OpenCV library. *Dr. Dobb's Journal of Software Tools*, 25(11).

12. Government of India. (2023). The Digital Personal Data Protection Act, 2023. Ministry of lectronics and Information Technology.

13. Flask-JWT-Extended. (2024). JWT extended support for Flask. https://flask-jwt- extended.readthedocs.io

14. NumPy. (2024). The fundamental package for scientific computing with Python. https://numpy.org