Face Recognition System for Campus

Kushagra Kaushik[1] Ashish kumar[2] Harsh bhardwaj[3] Mr.Ashutosh Pradhan[4]
[1,2,3,4]Department of Computer Science and Engineering (Data Science)
[1,2,3,4]RD Engineering College, Ghaziabad, India

Abstract — This report presents a design for an attendance system using device cameras (student laptops or smartphones) and optional fixed cameras. The system captures faces, detects them, and identifies each person via deep-learning embeddings (e.g. FaceNet) against an enrolled database. Liveness checks (texture analysis, eye-blink detection, etc.) ensure security against photo/video spoofing. Attendance entries (timestamps) are logged in a database. Only authenticated teachers or admins can access the system. The implementation uses open-source tools (OpenCV, TensorFlow/PyTorch, Flask) and low-cost hardware (Android/iOS devices, Raspberry Pi, Jetson Nano). Key evaluation metrics include recognition accuracy, FAR/FRR, latency, and throughput. Based on similar systems, we expect >95% accuracy and sub-second processing. Privacy is handled by storing only encrypted embeddings and following India's DPDP 2023 guidelines. We include architecture diagrams, comparative tables (algorithms/datasets/hardware), and a development timeline. We propose an automated attendance system leveraging students' own devices (laptop/phone cameras) and optionally campus cameras (CCTV/RTSP). Attendance is recorded twice daily without manual roll calls. The software runs on personal devices (browser or app) for convenience. Using lightweight CNN models (MobileNet, FaceNet, etc.), the system achieves high accuracy. Anti-spoofing (texture CNN + blink/motion checks) is integrated to prevent fraudulent sign-ins. A secure portal allows only teachers/admins to log in and view records. We detail system requirements, methodology, architecture, implementation, evaluation plan, and ethical considerations. Data privacy is a priority, with encrypted face embeddings and DPDP 2023 compliance (explicit consent, data minimization). The design is scalable to large classes and multi-camera setups (including existing CCTV).

Keywords: Automated Attendance System; Face Recognition Technology; Deep Learning-Based Identification; Liveness Detection; Secure Authentication; Data Privacy Compliance;

I. Introduction

Traditional attendance methods (roll calls, sign-in sheets) are error-prone and time-consuming. They also allow proxy sign-ins. Modern face-recognition systems can automate this process. In our approach, each student uses their own camera—for example, a laptop's webcam or a smartphone's front camera—to check in. The system is designed to be camera-agnostic: it can also integrate with campus-wide CCTV feeds if available. This flexibility minimizes extra hardware costs and leverages devices already at hand.

- Motivation: Automating attendance saves faculty time and reduces fraud. By using personal devices, we avoid queues at a single scanner. Prior studies indicate high feasibility: Seelam et al. (2021) achieved ~98% accuracy with a Raspberry Pi camera for classroom attendance. Our system aims for similar performance using students' own cameras. With built-in guidance, each student can simply open the attendance app and present their face.
- Objectives: We will (1) detect and recognize faces from device/campus cameras; (2) mark each enrolled individual present once in morning and afternoon; (3) allow only teacher/admin login to view/modify records; (4) implement robust anti-spoofing tailored to webcams and phones; (5) ensure real-time or near-real-time operation; (6) achieve ≥90% accuracy (with low FAR/FRR); (7) comply with legal/privacy rules (e.g. DPDP 2023 consent requirements). Unspecified items (campus size, budget) are taken as "unspecified".

II. Literature Review (2018–2026)

A. Desktop/Mobile Attendance Systems:

Early works showed feasibility with CCTV or fixed cameras. Seelam et al. (2021) placed a Pi camera above a blackboard to capture all students and used FaceNet on-device, reporting ~98% recognition accuracy. Prasanna et al. (2025) built a Pi+Node-RED attendance system, achieving low latency via on-device processing. These systems demonstrate that even limited hardware (Raspberry Pi, laptop cams) can deliver high performance.

B. Multi-Device Flexibility:

Recent emphasis is on using personal devices. For instance, biometric attendance apps (like Timeero or Truein) allow students to self-check via mobile face scans. While academic papers on smartphone check-in are fewer, the trend is clear: supporting phone/laptop cameras enhances accessibility. Our review found no major academic systems focused solely on phones, but integration of mobile front-ends is a logical extension of webcam systems.

C. Detection & Recognition Methods:

Traditional methods (Haar cascades, HOG+SVM) are fast on CPU but less accurate with varying angles. Deep CNN detectors like MTCNN or YOLOv8 (Tiny) offer higher accuracy; for example, an attendance system study suggests using MTCNN for precise detection. For recognition, embedding models dominate: FaceNet, ArcFace, MobileFaceNet, etc. ArcFace (2019) is particularly accurate, achieving ~99.8% on LFW. We will use a lightweight embedding model (MobileNetV2-based) suitable for edge devices.

D. Anti-Spoofing:

CCTV-based systems often skip anti-spoofing, but mobile contexts require it. Xing et al. (2025) survey highlights CNN-based liveness detection as essential. Techniques include analyzing skin texture (CNN classifiers), eye-blink or micro-expression detection, and active challenges. We will incorporate multiple cues (texture + motion). For phones, we can also leverage device capabilities: some smartphones have

IR depth sensors (e.g. iPhone FaceID) which could be used to verify 3D structure.

E. Privacy & Fairness:

The DPDP 2023 Act in India mandates explicit consent for biometric data. Our design anonymizes faces into embeddings and encrypts them. We also note the NIST 2019 FRVT finding that recognition accuracy varies by demographics. As such, we plan to evaluate error rates across groups to ensure fairness. These considerations, while outside pure technical scope, are crucial for deployment in education.

Algorithm	Category	Accuracy/Notes
Haar Cascade	Detection	Fast on CPU; works for frontal faces (~60–80% accuracy)
HOG + SVM	Detection	Moderate speed; ~70–90% for frontal faces
MTCNN	Detection	Deep CNN; ~90%+, moderate speed (CPU/GPU options)
YOLOv8-Tiny	Detection	Very fast (real-time on GPU); high accuracy
FaceNet	Recognition	99.6% on LFW (128-dim embeddings)
ArcFace (ResNet)	Recognition	99.8% on LFW (512-dim embeddings)
MobileFaceNet	Recognition	~98% on LFW; optimized for mobile/edge
OpenFace (Dlib)	Recognition	~95% on LFW; CPU-friendly
LBPH	Recognition	~70–80%; robust to grayscale variations

Table 1: Algorithm Comparison

Dataset	Year	Images (Subjects)	Notes
LFW	2007	13,233 (5,749)	Standard verification set
CASIA-WebFace	2014	494,000 (10,575)	Common for pretraining
VGGFace2	2018	3.31M (9,000+)	Large, diverse (age/ethnicity)
CelebA	2015	202,599 (10,177)	Large public celebrity faces
CASIA-SURF (anti-spoof)	2018	130,000+ (1000)	Spoof dataset (print/replay)
Campus Selfies (to collect)	2026	~N (All enrolled)	Student/staff faces via devices

Table 2: Face Recognition Datasets

Component	Example	Use Case	Notes
Laptop Front Camera	Built-in HD cam	Student self-check-in	720p/1080p; widely available
Smartphone Front Camera	Android/iOS	Student self-check-in	High-res; some have IR/depth sensors
USB Webcam (1080p)	Logitech C270	Instructor/desk capture	~₹1500; easy to set up
Raspberry Pi 4/5	—	Local processing (edge)	~₹3500–5000; can handle 1–2 video streams
NVIDIA Jetson Nano/Orin	—	High-performance edge	~₹6000–12000; multiple streams, faster
Google Coral Edge TPU	USB/NPU	Accelerated inference	<$100; works with Pi/Jetson for speedups
Desktop GPU PC	RTX 3060 or above	Training / multi-cam support	Needed for heavy training and server ops
Existing CCTV Network	IP Cameras	Passive coverage	If campus has CCTV, can plug into pipeline

Table 3: Hardware Options

III. System Requirements

- Face Capture: Accept input from any camera – student devices (laptop/mobile) and optionally RTSP/CCTV feeds. The system must fetch frames in real time.
- Recognition: Detect all faces in each frame and match them to enrolled profiles. Target accuracy ≥90%; FAR/FRR <5%.
- Attendance Logging: Automatically record a timestamped "present" once per person in morning and afternoon sessions.
- User Portal: Secure web/mobile portal where teachers/admins can log in to view attendance reports or manage profiles.
- Authentication: Only accounts with role "Teacher" or "Admin" can log in. Students do not log into the portal.
- Performance: Real-time or near-real-time operation. Aim for <0.5s per face recognition on edge. System should handle up to dozens of simultaneous check-ins.
- Robustness: Work under varying lighting (day/night), handle occlusions (glasses/masks), and moderate pose changes.
- Security: Include anti-spoofing for photo/video attacks. Encrypt stored embeddings; use HTTPS for communication.
- Privacy/Compliance: Design for DPDP 2023 compliance: explicit consent, minimal data storage (embeddings only), and data deletion on request. Audit all accesses.

IV. METHODOLOGY

A. Data Collection (Device and Campus Cameras)

Enroll each student/staff by capturing several face images using their own device or the classroom setup. This can be done via the attendance app: students take selfies from different angles and lighting. If campus cameras are used, administrators can label sample images from those feeds. Data augmentation (rotation, lighting shift, occlusion overlay) increases robustness.

B. Face Detection

Use fast detectors: On-device (phone/Pi), employ lightweight models like MobileNet-SSD or TensorFlow Lite face detector. OpenCV's Haar cascades can also run on a laptop for front-facing images. For CCTV or server, MTCNN or YOLOv8 can be used for speed and accuracy. Detected face regions are cropped and scaled for recognition.

C. Face Recognition

A CNN extracts embeddings for each face crop. Possible choices: FaceNet, ArcFace, or a custom MobileNetV2 model trained on campus data. We plan to use a MobileNetV2-based model due to its balance of accuracy and speed. Each embedding is compared (cosine similarity) to enrolled profiles in the Profiles DB. The highest match above a threshold confirms identity; otherwise, it's unknown. Thresholds are chosen based on validation to minimize false accepts/rejects.

D. Preprocessing

- Alignment: Use facial landmarks (eyes/nose) to align faces upright. This improves matching.
- Normalization: Resize to model input (e.g. 112×112), and normalize pixel values as per the network's training (mean/variance normalization).
- Enhancement: If the image is dark, apply histogram equalization or ask the user to move to better light.

E. Training Pipeline

1) Pretrain: Start with a model pretrained on large face datasets (e.g. VGGFace2).
2) Fine-tune: Train/fine-tune on collected campus faces (transfer learning). Use cross-entropy or triplet loss to refine embeddings.
3) Validate: Measure accuracy and compute ROC curve. Adjust match threshold to meet FAR/FRR targets.
4) Export Model: Convert to TensorFlow Lite or ONNX for deployment.

F. Anti-Spoofing (Liveness Detection)

For device cameras (especially phone/tablet):

- Texture Analysis: Train a CNN to classify "live" vs "spoof" using skin texture features.
- Motion/Blink Check: Require the user to blink or move head slightly. Monitor for natural eye blinks (eye-aspect ratio) in the 2–3 seconds of video. Absence of expected micro-movements suggests a spoof.
- Challenge-Response: Occasionally prompt the user (e.g. "Please smile" or "Rotate your face"). Verify compliance via pose landmarks.
- Reflection/Depth: If the device has an IR depth camera (e.g. iPhone FaceID), use it to confirm 3D structure. For regular cams, analyze specular highlights in eyes – photos often lack natural reflections.

For stationary cams/CCTV, similar liveness applies (though with CCTV, we assume supervision).

G. Handling Occlusion, Pose, Lighting

- Occlusion: Include some masked-face data in training. If recognition confidence is low due to a mask, the system can ask the student to briefly remove it for verification (if policy allows).
- Pose: Encourage frontal faces. If off-angle, use 3D alignment or accept partial features. Our model will be somewhat pose-invariant thanks to training.
- Lighting: Use phones' auto-flash if too dark. Warn users if faces are under/over-exposed. The model should be robust to moderate illumination changes via data augmentation.

V. SYSTEM ARCHITECTURE

The system architecture consists of two input streams – student device cameras (laptops/phones) and optional fixed campus cameras/CCTV – which both feed into a Face Detection Service. The detection service forwards crops to a Face Recognition Service, which queries a User Profiles DB to identify individuals and logs results to the Attendance DB. A Teacher/Admin Portal authenticates via a separate Authentication Service, which also references the Profiles DB, and provides authorized access to Attendance Reports.

The processing flowchart for each frame is: (1) Capture frame from camera → (2) Preprocessing (align & normalize) → (3) Face Detection (Haar/CNN) → (4) Liveness Check – if spoof, reject/alert; if live, proceed → (5) Face Recognition → (6) Profile Match check – if matched, log attendance to Attendance DB; if not matched, prompt retry/enroll.

VI. IMPLEMENTATION DETAILS

- Front-End (Clients): Develop a cross-platform mobile app or web interface. On a smartphone or laptop, the user opens the app, which accesses the camera (via HTML5 or native API). The app captures a face image (or video snippet) and sends it to the server (or processes on-device).
- Backend: Implement using Python (Flask or FastAPI). Services include: /api/login (teacher/admin login), /api/checkin (takes face embedding, attempts recognition), /api/attendance (retrieve logs). Use JWT tokens for authentication.
- Edge vs Cloud: For small deployments, run recognition on a local server or powerful PC. Optionally, small campuses can run a Raspberry Pi per classroom with a camera; P2P sync to a central DB can then occur.
- Models: Use TensorFlow Lite or PyTorch Mobile for on-device inference. For example, run MobileNet-SSD for detection and a TFLite FaceNet model for recognition on the phone. On Raspberry Pi, use Coral USB for acceleration if needed.

- Database Schema: Key tables include – Users (user_id, username, password_hash, role); Profiles (profile_id, user_id, full_name, face_embedding); Students (student_id, profile_id, name, class, roll_no); Attendance (att_id, profile_id, timestamp, session).
- API Endpoints: POST /api/login – teacher/admin login, returns JWT token. POST /api/checkin – (Protected) submits face embedding; logs attendance if match found. GET /api/attendance?date=YYYY-MM-DD – (Protected) retrieves attendance records for a given day. GET /api/reports/summary – (Protected) generates an attendance summary report. All endpoints use standard JSON and require authorization headers. Attendance entries are timestamped and tagged with session ["morning"/"afternoon"].

VII. Evaluation Plan

We will evaluate on a test dataset and pilot deployment. Metrics include:

- Accuracy: Percentage of correctly recognized faces. Goal >90%. Prior systems report ~95%.
- FAR/FRR: False Accept/Reject Rates. Aim for <5% each. Adjust similarity threshold to balance them.
- Latency: Time per face or frame. Target <0.5s on edge (device or Pi) for detection+recognition. Measure end-to-end from capture to log.
- Throughput: Number of concurrent streams/requests the system can handle. For server mode, test with 30–50 simultaneous clients.
- Scalability: Test adding more cameras. The architecture should easily scale by adding more processing nodes.
- Anti-Spoof Effectiveness: Test with printed photos and video replays; system should reject these (ideally >95% of attack attempts).
- Security/Privacy Audit: Ensure no raw face images are stored. Confirm encryption in DB. Verify consent workflow.
- Fairness: Evaluate error rates by gender and ethnicity. Ensure no significant disparity.

An illustrative performance breakdown targets 90% true positives, 5% false accepts, and 5% false rejects. Actual results will be measured in testing.

VIII. Experiments and Expected Results

In pilot trials (e.g. one classroom, 30 students), we expect recognition accuracy in the mid-90% range and FAR/FRR below 5%, consistent with prior work. We will compare performance of running models on-device (smartphone) versus a central server. If using on-device detection, network latency is minimal; server mode requires network transfer. Anti-spoofing tests (with volunteers presenting photos) should confirm high detection of attacks. We also expect phones with depth sensors to show near-zero spoof success due to 3D checks.

IX. Deployment Plan

- Edge vs Server: For small scale, a central server with modest GPU can handle recognition from all devices. For reliability, we recommend a hybrid: do face detection on-device, then either match locally or send embeddings to the server. For example, smartphones can run detection/embedding locally and only query the server with the embedding. This minimizes bandwidth.
- Real-Time Operation: The system runs in defined windows (morning entry, afternoon exit). Students launch the app before class; the system continuously processes incoming check-ins. Teachers can also remotely query logs during the day.
- Integration: Use standard protocols (RTSP) to connect any networked campus cameras. The same processing pipeline can handle CCTV feeds by treating each frame as input. If the campus has an existing ID system, the attendance DB can sync with it via API.
- Maintenance: Regularly update the model with new faces or adjust threshold. Ensure software updates for anti-spoof algorithms. Provide teachers with a dashboard for flagging errors.

X. User Access Control and Auditing

- Only teachers and admins have login accounts. Accounts are created by the administrator.
- Authentication: We implement JWT tokens for API access. On successful login (/api/login), the server issues a token. Protected API calls (e.g. fetching reports, marking attendance) require this token.
- Logging: All login attempts, and all attendance logs, are recorded in an audit log. For example, each /api/checkin request logs the timestamp, the student's profile ID, and the admin/teacher's ID who authorized the session. Any data modification (profile edits, deletions) is similarly logged. This fulfills accountability requirements and aligns with DPDP's data processing records.

XI. Limitations

- Device Quality: Poor camera quality (low resolution, motion blur) can reduce accuracy. Camera quality and viewing geometry significantly affect recognition accuracy.
- Lighting Conditions: Extreme lighting (very dark or glare) can impede detection. Phones can compensate to some extent, but face detection may fail in suboptimal light.
- Occlusions: Masks or sunglasses will reduce match confidence. Our system handles partial occlusions via model robustness, but heavy occlusion may require manual override.
- Spoof Sophistication: While we use multiple anti-spoof cues, determined attackers with 3D masks or deepfakes might still succeed. Using device IR sensors can mitigate this, but not all devices have them.
- Scalability: Our solution suits small-to-medium classrooms. For a very large campus (hundreds of cameras and thousands of students), more robust server infrastructure and network capacity would be needed.
- Privacy Concerns: Continuous facial monitoring could raise concerns despite consent. Clear communication and opt-out policies are necessary.

XII. FUTURE WORK

Future enhancements could include multi-factor attendance (combining face with, say, a Bluetooth beacon). Improving liveness detection (e.g. heartbeat signal via camera) is an active research area. We plan to incorporate on-device continuous learning so new students can enroll themselves. Further, integration with other campus systems (e.g. grading) could automate reporting. Monitoring evolving data protection regulations (post-DPDP 2023) will be essential.

XIII. IMPLEMENTATION CHECKLIST

1) Data Enrollment: Collect student face images via their devices; label and store securely.
2) Model Development: Train/fine-tune face recognition and anti-spoof models with collected data.
3) Client App: Develop mobile/web app for students to capture faces. Implement camera access and liveness prompts.
4) Server Backend: Set up database (Profiles, Attendance) and API services (Flask/FastAPI). Implement detection/recognition endpoints.
5) Testing: Conduct lab tests for recognition accuracy, spoof rejection, and latency on target devices.
6) Pilot Deployment: Roll out in a controlled environment (one class). Gather metrics and user feedback.
7) Performance Tuning: Based on results, adjust thresholds, optimize models (quantization or edge TPU).
8) Full Deployment: Expand to wider campus, provide training for staff, and set up maintenance procedures.

REFERENCES

[1] Surantha, N. & Sugijakko, B. (2024). Lightweight Face-Recognition Attendance System with Liveness Detection. Internet of Things, 25, 101089.
[2] Prasanna, M., Subba Reddy, I.V. & Padmapriya, V. (2025). Cost-Effective Real-Time Attendance Using Raspberry Pi & Node-RED. J. Inf. Syst. Eng. & Management, 10(41S), 8000.
[3] Seelam, V. et al. (2021). Smart Attendance with Deep Learning and Computer Vision. Mater. Today Proc., 46, 4091–4094.
[4] Aboluhom, A.A.A. & Kandilli, I. (2025). Real-Time Facial Recognition via Multi-Task Learning on Raspberry Pi. Sci. Rep. 15, 28467.
[5] Zheng, X. et al. (2024). CCTV-based Facial Recognition for Classroom Attendance. Int. J. Advanced Res. Comput. Eng. Tech., 13(2), 88–95.
[6] Xing, H. et al. (2025). Face Anti-Spoofing: A Survey of Deep Learning Methods. Appl. Sci., 15(12), 6891.
[7] Ministry of Electronics & IT, India (2025). Press Release: Digital Personal Data Protection Rules, 2025 Notified. Government of India.
[8] NIST (2019). Face Recognition Vendor Test (FRVT) – Demographic Effects. Press Release.
[9] Patel, J. et al. (2025). Enhancing Classroom Attendance with CNN-based Face Recognition. Procedia Comp. Sci., 258, 3031–3041.

