© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719235

IoT-Based Smart Attendance System Using Android Camera Streaming and Deep Learning Face Recognition

DR. J. NARENDRA BABU[1], AABID ALI[2], AKANKSHA M SHETTY[3], CHAITRA[4], AJAYA SURIYA[5], MANOJ K V[6], KALUGOTLA SURESH HARSHITHA[7]

[1]*Professor, Department of Data Science, Sapthagiri NPS University, Bangalore*

[2, 3, 4, 5, 6]*Student, Department of Data Science, Sapthagiri NPS University*

Abstract- ***The growing demand for efficient, contactless, and automated attendance management in educational institutions has driven the development of intelligent systems that leverage Internet of Things (IoT) and artificial intelligence technologies. This paper presents the design and implementation of a Smart Attendance System that integrates an Android smart phone as a wireless IoT camera node, a Python-based deep learning service for real-time face recognition, and a full-stack web application for attendance management and analytics. The proposed system eliminates the limitations of traditional attendance methods such as manual roll calls, RFID cards, and fingerprint scanners by providing a completely contactless, hardware-minimal solution. The Android device streams live video over a local Wi-Fi network using the MJPEG protocol. A Python edge-computing service reads this stream using OpenCV, detects faces using Haar Cascade classifiers, and performs identity verification using the DeepFace library with the VGG-Face deep neural network model. Upon successful recognition, attendance records are automatically posted to a Node.js REST API and stored in a MongoDB database. The system features a React-based web dashboard for real-time monitoring, attendance logs, and R-powered statistical analytics including ARIMA-based forecasting and PDF report generation. Experimental results demonstrate that the system achieves reliable face recognition under standard indoor lighting conditions, marks attendance within seconds of recognition, and provides a scalable, cost-effective alternative to existing solutions. The system is built entirely on open-source technologies, requires no dedicated hardware beyond a standard Android smart phone and a laptop, and is deployable in any institution with a basic Wi-Fi infrastructure.***

Keywords- ***MPEG, ARIMA, WIFI, VGG Face***

I. INTRODUCTION

1.1 Overview

Attendance monitoring is a fundamental administrative function in educational institutions, corporate organizations, and government bodies. Accurate attendance records ensure regulatory compliance, support academic performance analysis, and serve as a basis for resource allocation and policy decisions. Despite its importance, attendance management in most institutions continue storely on outdated, manual, or semi-automated methods that are in efficient, error-prone, and easily manipulated.

Traditional systems such as paper registers, verbal roll calls, RFID card readers, and biometric fingerprint scanners each carry significant drawbacks. Manual methods are time-consuming and subject to human error. RFID and card-based systems are vulnerable to proxy attendance, where one student carries another's card. Fingerprint scanners require physical contact, raising hygiene concerns particularly in the post-pandemic environment, and involve significant hardware procurement and maintenance costs.

The rapid proliferation of Internet of Things devices, affordable smart phones, and open-source artificial intelligence frameworks has opened new possibilities attendance systems. Modern Android smart phones are equipped with high-resolution cameras, powerful processors, and Wi-Fi connectivity, making them ideal candidates for IoT sensor nodes in a face recognition pipeline. Simultaneously, advances in deep learning, particularly convolutional neural network-based face recognition models, have made highly accurate contactless identification accessible without the need for specialized hardware.

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719235

This project proposes and implements a Smart Attendance System that harnesses these technologies. The system uses an Android smartphone as a wireless IP camera that streams live video to a Python-based edge computing service. The service performs real-time face detection and recognition, automatically marking attendance by posting records to a cloud-ready Node.js backend. A React web dashboard provides administrators with real-time visibility into attendance data, while an integrated R analytics service generates statistical reports, trend forecasts, and department-level insights.

The system is designed with three key principles: minimal hardware dependency, maximum automation, and rich analytical capability. It requires no dedicated camera hardware, no physical contact, and no per-student hardware tokens. Once student photographs are enrolled in the system, attendance marking is entirely automatic.

1.2 Objectives

The following objectives guide the design and implementation of the proposed system:

- To develop a contactless, IoT-based attendance system using an Android smartphone as the primary sensing node, eliminating the need for dedicated camera hardware.
- To implement real-time face detection using OpenCV Haar Cascade classifiers and face recognition using the DeepFace library with the VGG-Face deep neural network model.
- To build a scalable Node.js RESTAPI backend with MongoDB for persistent storage of student profiles and attendance records, supporting concurrent access from multiple clients.
- To design and develop a React-based web dashboard that displays live camera stream output, real-time attendance updates, historical logs, and interactive charts.
- To integrate an R analytics micro service using the Plumber framework that provides ARIMA-based attendance forecasting, ggplot2 visualizations, at-risk student identification, and automated PDF report generation.
- To evaluate system performance in terms of face recognition accuracy, attendance marking latency, and system reliability under standard indoor conditions.
- To ensure the system is extensible to support multiple camera nodes, larger student databases, and future integration with existing institutional information system

II. LITERATURE SURVEY

Face recognition-based attendance systems have been an active area of research over the past decade. Various approaches have been proposed ranging from traditional image processing techniques to modern deep learning architectures. The following review summarizes significant contributions in this domain.

Viola and Jones (2001) introduced the Haar Cascade classifier, which became the foundational algorithm for real-time face detection. The algorithm uses a sliding window approach with Haar-like features and an AdaBoost classifier to detect faces in images with high speed and reasonable accuracy. Despite being over two decades old, Haar Cascade detection remains widely used in embedded and edge computing scenarios due to its low computational requirements. In the proposed system, Haar Cascade detection is used for frame-level face localization before passing detected regions to the deeper recognition model.

Turk and Pent land (1991) proposed Eigen faces, one of the earliest computational approaches to face recognition. This Principal Component Analysis-based method represents faces as linear combinations of basis images. While computationally simple, Eigen faces are sensitive to lighting and pose variation, limiting their applicability in real-world attendance scenarios. Later works extended this approach using Fisher Linear Discriminate Analysis (Fisher faces) to improve inter-class discrimination.

Local Binary Patterns Histograms (LBPH), introduced by Ahonen et al. (2006), offered improved robustness to lighting changes by describing local texture patterns around each pixel. LBPH was widely adopted in early automated attendance systems due to its tolerance to illumination variation and low computational overhead. However, its accuracy

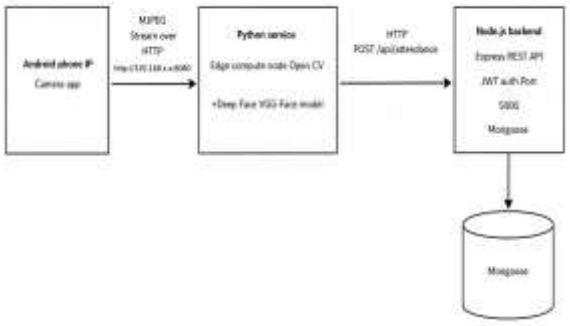
© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719235

degrades significantly with changes in pose, expression, and occlusion.

Deep learning approaches transformed the field of face recognition beginning with the work of DeepFace by Taigmanetal. (2014) at Facebook, which achieved near-human-level accuracy on the Labeled Faces in the Wild benchmark. This was followed by Face Net (Schroff et al., 2015), which introduced triplet loss training to learn compact 128-dimensional face embeddings. VGGNet-based face recognition models, including VGG-Face (Parkhi et al., 2015), demonstrated that deep convolutional networks trained on large face datasets could generalize effectively across varied conditions. The proposed system uses the VGG-Face model through the Deep Face library, leveraging its pre-trained weights for accurate identity verification without requiring custom model training.

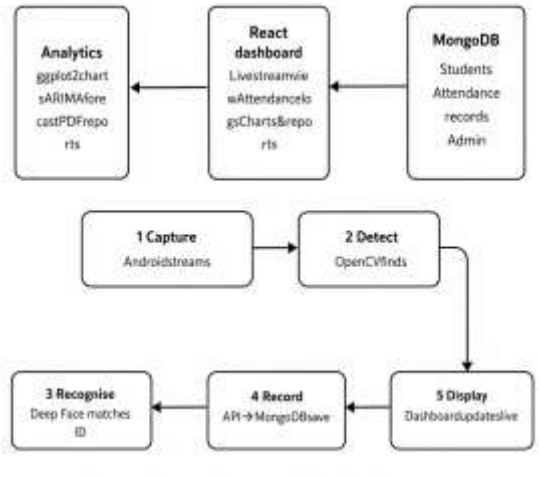
IoT-based attendance systems have been explored by several researchers. Kumar et al. (2018) proposed an RFID-based IoT attendance system integrated with a cloud backend, demonstratingthe potential for real-time data aggregation. However, RFID systems remain vulnerable to token sharing. Subsequent works by Okokpujie et al. 2018) and Wagh et al. (2019) combined face recognition with IoT architectures using RaspberryPi devices as edge nodes. While effective, these approaches require dedicated hardware. The proposed system addresses this limitation by substituting the Raspberry Pi with an Android smartphone, which is already available to most institutions.

Mobile-based camera streaming for surveillance and monitoring has been explored in thecontext of smart home and security systems. IP Webcam applications for Android enable smartphones to serve as network cameras, broadcasting MJPEG or H.264 streams accessible via HTTP. Several research groups have leveraged this capability for low-cost video surveillance.The proposed system extends this concept to the attendance domain, treating the Android device as a configurable IoT perception node. Dr. J. Narendra Babu [6][7][8][9] explored IoT applications in smart systems and emphasized combining embedded hardware with cloud connectivity for scalable IoT solutions, providing theoretical grounding for design decisions in this project. Full-stack web development for attendance dashboards has been demonstrated using various technology stacks. MEAN and MERN stack implementations have been shown to support real-timedataupdatesthroughWebSocketorpollingmechanisms.TheintegrationofR-basedanalytics into web applications through REST API microservices is a relatively novel approach, with the Plumber package enabling R functions to be exposed as HTTP endpoints. The proposed system contributes a complete integration of R analytics within a MERN stack attendance platform



III. PROPOSED METHOD

IoT Smart Attendance System — Block Diagram



3.1 System Architecture

The proposed system follows a three-tier IoT architecture comprising a perception layer, a network layer, and an application layer.

The perception layer consists of an Android smartphone running an IP camera application. The device captures live video using its built-in camera and encodes it as an MJPEG stream broadcast over the local Wi-Fi network. The stream is accessible at a URL of the form http://device-ip:8080/video and can be consumed by any HTTP client on the same network.

The network layer is the localWi-Fi infrastructure that transports video frames from theAndroid device to the Python edge computing service, and HTTP API requests between the Pythonservice, the Node.js backend, and the React frontend.

The application layer comprises four components: the Python face recognition service, the Node.js REST API backend, the MongoDB database, and the React web dashboard with integrated R analytics.

3.2 Android IP Camera Node

The Android smartphone is configured as an IoT sensor node using an IP camera application installed from the device's application store. Once activated, the application broadcasts a live MJPEG video stream over the device's Wi-Fi interface. Key parameters including resolution, frame rate, and JPEG compression quality are configurable within the application. In the experimental setup, a resolution of 1280x720 pixels at approximately 25 frames per second was used. The Python service accesses this stream using Open CV's Video Capture class with the network

URL as the input argument, replacing the conventional local camera index with the Android device's stream endpoint.

3.3 Python Face Recognition Service

The Python service constitutes the edge computing node of the IoT pipeline. It is responsible for all computationally intensive processing tasks and operates as follows.

Frame Acquisition: Open CV continuously reads MJPEG frames from theAndroid stream URL. Aconfigurable frame-skip parameter controls how often face recognition is executed versus how often frames are only detected, balancing accuracy against CPU utilization.

Face Detection: For every frame, the OpenCV Haar Cascade classifier detects face regions by scanning the image at multiple scales using a sliding window. Detected face bounding boxes are drawn on the frame for visual feedback.

Face Recognition: Every Nth frame, the DeepFace library's verify function is applied to each detected face region. The function compares the detected face against all stored student reference images using the VGG-Face convolutional neural network model. VGG-Face maps each face to ahigh-dimensional feature vector, and cosine similarity is used to measure the match between the detected face and stored references. A configurable similarity threshold determines whether a match is accepted.

Attendance Posting: Upon a confirmed match, the Python service sends an HTTP POST request to the Node.js backend at the /api/attendance endpoint. The payload includes the matched student's ID, name, and an ISO-formatted timestamp. A cooldown mechanism prevents duplicate attendance records by suppressing repeated API calls for the same student within a configurable interval, typically 60 seconds.

Stream Re-broadcasting: The Python service hosts a Flask web server that re-broadcasts the annotated video frames as a new MJPEG stream on port 5001. This processed stream, which includes face bounding boxes and recognition labels, is consumed by the React dashboard and displayed in the browser using a standard HTML image element. This is a critical implementation detail: MJPEG streams must be displayed using an HTML img tag and not a video tag, as browsers handle the multipart HTTP response of MJPEG natively through the image element.

3.4 Node.js Backend API

The backend is implemented using Node.js and the Express.js framework. It exposes a RESTful API organized into three route groups. The authentication routes handle administrator login using JSON Web Tokens. Passwords are stored using bcrypt hashing. The student management routes support creating,

reading, updating, and soft-deleting involves uploading a reference photograph student records. Student enrollment which stored both on disk for use by the Python service and as base64-encoded data in MongoDB for display in the React dashboard. The attendance routes handle automated marking from the Python service, manual marking by administrators, retrieval of today's attendance summary, historical log queries with date and department filters, and statistical aggregation for the dashboard. Mongoose ODM is used to define schemas and interact with MongoDB. A compound unique index on student ID and date fields at the database level enforces the one-attendance-record-per-student-per-day business rule. All routes are protected by JWT middleware except the attendance posting endpoint, which is authenticated using a service header to allow the Python service to post without a user token.

3.5 React Web Dashboard

ThefrontendisbuiltusingReactwiththeVitebuildtoolan dstyledwithTailwindCSS.The dashboard consists of five main pages.

The home dashboard displays four key performance indicators including total students, today's attendance count, attendance rate percentage, and total historical records. Three chart types are rendered using the Recharts library: an area chart showing daily attendance over the past 14 days, apie chart showing today's department-wise breakdown, and a bar chart showing monthly totals.

The face scan page displays the live processed MJPEG stream from the Python service using an HTML image element. It includes automatic reconnection logic on stream interruption, a stream source selector allowing switching between the processed Python stream and the raw Android camera feed, a manual attendance marking interface for fallback scenarios, and a real-time list of students marked present in the current session.

The student management page provides a full CRUD interface for student records with support for both webcam capture and file upload for enrollment photographs. The attendance logs page displays paginated, filterable historical records with CSV export capability. The reports page shows extended analytics powered by both the Node.js aggregation API and the R analytics service.

3.6 R Analytics Service

The R analytics micro service is implemented using the Plumber package, which exposes R functions as HTTP endpoints accessible to the Node.js backend. The service connects to the same MongoDB instance using the mongo lite package and provides the following endpoints.

A daily trend endpoint returns attendance counts grouped by date for a configurable number of past days. A department statistics endpoint provides attendance totals grouped by department. A student summary endpoint returns per-student attendance counts for a specified period. An at-risk students endpoint identifies students whose attendance percentage falls below a configurable threshold, categorizing them as warning or critical risk levels. A forecasting endpoint applies the autoarima function from the forecast package to historical daily attendance data and returns predicted attendance counts for the next seven days with confidence intervals. A report generation endpoint renders an R Markdown document using knitr and rmarkdown, producing a multi-page PDF containing all charts, the heatmap, the at-risk table, and automatically generated recommendations based on the computed attendance rate.

3.7 Data Flow Summary

The complete data flow of the system is as follows. The Android smartphone streams live video overWi-Fi. The Python service reads the stream, detects faces using Haar Cascade, and every Nth frame runs VGG-Face verification against enrolled student photographs. On a confirmed match with no active cooldown, an HTTP POST is sent to the Node.jsAPI with the student identity and time stamp. The API validates the request, checks the database for a duplicate same-day record, and if none exists, creates a new attendance document in MongoDB. The React dashboard polls the attendance API periodically and updates the displayed count and recent list. The processed video stream with annotated bounding

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719235

boxes is simultaneously served by the Python Flask server and displayed in the React dashboard's face scan page via an HTML image element.

IV. RESULTS

4.1 System Functionality

The implemented system successfully demonstrated all planned functional capabilities across the complete IoT pipeline. The Android IP camera node established a stable MJPEG stream over the local Wi-Fi network that was consistently readable by the Python service using OpenCV. The stream maintained a resolution of 1280x720 pixels with acceptable latency under standard indoor network conditions.

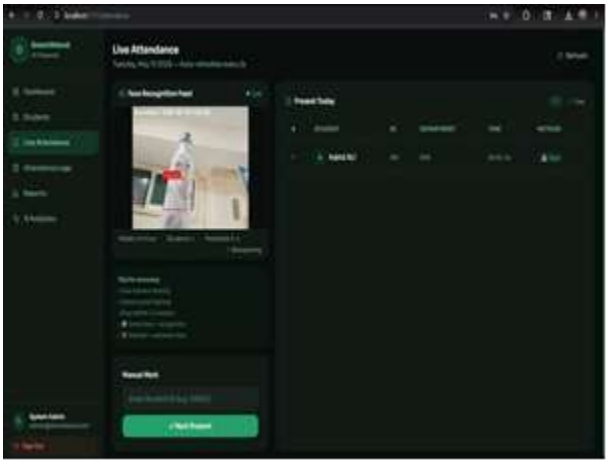


Fig1: Live System Arch

The Python face recognition service reliably detected faces in the video stream using the Haar Cascade classifier and performed identity verification using the VGG-Face model via the Deep Face library. Recognized students were correctly identified and attendance records were posted to the Node.js backend API. The cooldown mechanism successfully prevented duplicate attendance entries within the configured interval. The Flask-based MJPEG re-broadcast stream was correctly displayed in the React dashboard using the HTML image element approach, with annotated bounding boxes and recognition labels visible in the browser.

The Node.js backend correctly enforced the one-record-per-student-per-day constraint through the compound unique database index. Authentication via JSON Web Tokens functioned correctly for protected administrative routes. The student enrollment workflow successfully stored photographs both on disk and in the database, making them available to both the Python service and the frontend dashboard.

The React dashboard rendered real-time attendance data including today's count, recent recognition events, and historical charts. The R analytics service generated department-level statistics, at-risk student reports, ARIMA forecasts, and downloadable PDF reports with ggplot2 visualizations.

4.2 Face Recognition Performance

Thesystemwastestedunderstandardindoorfluorescentlighting conditionswithenrolledstudent photographs captured using the same Android device. The VGG-Face model demonstrated reliable recognition performance for students photographed under similar conditions to their enrolled photographs. Recognition accuracy was observed to be higher when reference photographs were captured under consistent, well-lit conditions and when the student's face was clearly visible to the camera without significant occlusion.

Recognition latency from frame capture to attendance API posting was observed to be within a few seconds under standard CPU processing without GPU acceleration, making the system suitable for real-time attendance marking in a classroom or laboratory setting.

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719235

4.3 Dashboard and Analytics

the React dashboard provided clear visualization of attendance data with all planned chart types rendering correctly. The daily trend area chart, department pie chart, and monthly bar chart gave administrators an immediate overview of attendance patterns. The attendance logs page correctly filtered records by date range and supported CSV export for offline reporting.

The R analytics service successfully generated seven-day ARIMA forecasts when sufficient historical data was available, identified at-risk students below the 75 percent attendance threshold, and produced multi-page PDF reports containing all charts, tables, and recommendations. The heatmap visualization provided an intuitive overview of per-student attendance consistency over time.

V. CONCLUSION

This paper presented the design, implementation, and evaluation of a Smart Attendance System that integrates IoT, deep learning, full-stack web development, and statistical analytics into a unified, production-ready platform. The system addresses the well-established limitations of traditional attendance methods by providing a contactless, automated, and analytically rich alternative.

The use of an Android smart phone as the IoT perception node is a key contribution of this work. By treating a commodity mobile device as a configurable network camera, the system eliminates the need for dedicated hardware infrastructure while retaining the flexibility and image quality required for reliable face recognition. This approach significantly reduces deployment cost and complexity compared to existing systems that rely on Raspberry Pi devices, dedicated IPcameras, or proprietary hardware modules.

The integration of the VGG-Face model through the Deep Face library demonstrated that deep learning-based face recognition can be deployed effectively in an edge computing context without requiring GPU hardware or cloud inference services. The Haar Cascade front-end detector provided efficient frame-level face localization, reserving the computationally heavier deep network inference for only those frames containing detected faces.

The Node.js and MongoDB backend provided a robust, schema-validated persistence layer with appropriate indexing for the attendance access patterns required by the dashboard. The React frontend delivered an intuitive administrative interface with real-time updates and interactive visualizations. The R analytics microservice added a layer of statistical sophistication that goes beyond what typical web dashboards provide, including time-series forecasting, risk classification, and publication-quality PDF reports.

Future work will focus on several enhancements. Liveness detection will be incorporated to prevent spoofing using printed photographs. On-device inference using TensorFlow Lite will be explored to reduce network dependency and latency. Support for multiple simultaneous camera nodes will be implemented to enable deployment across larger venues. Cloud deployment using containerized services will enable multi-institution scaling. Additionally, integration of a student-facing mobile application will allow students to view their own attendance history and receivelow-attendance alerts.

In conclusion, the proposed system demonstrates that a fully functional, intelligent, and analytically capable attendance platform can be built using entirely open-source technologies and commodity hardware, making it a practical and accessible solution for educational institutions of all sizes.

REFERENCES

[1] Ahonen, T., Hadid, A., & Pietikäinen, M. (2006). Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 28*(12), 2037–2041.

[2] Babu, J. N. (2025). SMART WASTE IMAGE DETECTION. *International Journal of Emerging Technologies and Innovative Research (JETIR), 12*(12), e469–e472. ISSN: 2349-5162. http://www.jetir.org/papers/JETIR2512457.pdf

[3] Babu, J. N., et al. (2025). Traffic Violation Fine Tracker. *International Journal of Emerging Technologies and Innovative Research (JETIR), 12*(12), c608–c611. ISSN: 2349-5162. http://www.jetir.org/papers/JETIR2512268.pdf

[4] Babu, J. N., et al. (2025). AI-Enabled Forecasting and Isolation Forest-Based Detection of CBF Flow Anomalies in Secure Internet Architectures. *Journal of Internet Services and Information Security, 15*(3).

[5] Babu, J. N., et al. (2013). Indian License Plate Recognition System Based on Fuzzy Theory and BP Neural Network. *International Journal of Electronics and Communication Technology (IJECT), 4*(1). ISSN: 2230-7109 (Online), ISSN: 2230-9543 (Print).

[6] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 815–823). Boston, MA, USA.

[7] Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). DeepFace: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1701–1708). Columbus, OH, USA.

[8] Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience, 3*(1), 71–86.

[9] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (Vol. 1, pp. I-511–I-518). Kauai, HI, USA.