IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 5, May 2026

DOI: 10.17148/IARJSET.2026.13526

Design and Implementation of a Scalable Web-Based Attendance Monitoring System Utilizing Node.js and MongoDB Architectures

A. Asrin Mahmootha[1], Dr. B. Aysha Banu[2], R. Praveen Kumar[3], R. Vijay[4], A. Mohamed Ashik Ilahi[5]

Assistant Professor, Department of Information Technology,

Mohamed Sathak Engineering College, Kilakarai, Tamil Nadu, India[1]

Professor & Head, Department of Information Technology,

Mohamed Sathak Engineering College, Kilakarai, Tamil Nadu, India[2]

B.Tech Student, Department of Information Technology,

Mohamed Sathak Engineering College, Kilakarai, Tamil Nadu, India[3]

B.Tech Student, Department of Information Technology,

Mohamed Sathak Engineering College, Kilakarai, Tamil Nadu, India[4]

B.Tech Student, Department of Information Technology,

Mohamed Sathak Engineering College, Kilakarai, Tamil Nadu, India[5]

Abstract: This paper presents the architectural design and implementation of a robust, scalable web-based attendance monitoring system leveraging the asynchronous, event-driven capabilities of Node.js and the flexible, document-oriented data model of MongoDB. Traditional attendance methods suffer from manual inefficiencies, absence of real-time accessibility, and vulnerability to proxy attendance. The proposed system integrates a responsive React.js front-end with a resilient Node.js/Express.js back-end, providing real-time attendance logging, comprehensive reporting functionalities, and secure role-based access control (RBAC). A microservices-oriented design emphasises modularity and scalability. Experimental evaluation on 1,500 facial images across 50 subjects demonstrates >95% recognition accuracy and average API response times below 200 ms under 50 concurrent users, confirming readiness for real-world educational deployment.

Keywords: Attendance management; web application; Node.js; MongoDB; scalability; real-time monitoring; facial recognition; role-based access control; microservices

I. INTRODUCTION

The escalating demand for efficient management of academic and administrative processes necessitates advanced systems to mitigate inefficiencies inherent in traditional, paper-based methods [1]. Manual attendance tracking consumes valuable instructional time, is susceptible to proxy fraud, and fails to deliver the real-time analytics that modern institutions require.

This work proposes a web-based attendance monitoring system leveraging Node.js and MongoDB to provide a scalable, real-time solution for educational and corporate settings [2]. The system directly addresses three core limitations of existing approaches: (i) manual record-keeping inefficiency, (ii) absence of real-time data accessibility, and (iii) vulnerability to proxy attendance and data tampering.

The proposed system incorporates real-time attendance logging via facial recognition, comprehensive reporting, and secure RBAC, ensuring operational efficacy and data integrity across complex organisational hierarchies [3]. A microservices design philosophy enhances maintainability and facilitates future extensions to accommodate evolving institutional requirements.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 5, May 2026

DOI: 10.17148/IARJSET.2026.13526

The remainder of this paper is organised as follows: Section II surveys related work; Section III describes the system methodology; Section IV presents the data flow diagram; Section V reports experimental results; Section VI discusses findings; and Section VII concludes the paper.

II. LITERATURE REVIEW

Prior research extensively explored various methodologies for automated attendance, ranging from RFID-based IoT approaches [23] to biometric systems using fingerprint sensors [27], each with distinct trade-offs in scalability, accuracy, and deployment cost. Table I summarises representative approaches from the literature.

More recent work shifted toward facial recognition due to its non-intrusive nature and increasing accuracy with deep learning. Systems employing YOLOv5 combined with anti-spoofing measures achieved >94% accuracy [25], while MTCNN + FaceNet architectures demonstrated up to 98% accuracy under controlled conditions [16]. However, few studies addressed the full-stack integration challenge of combining a scalable web back-end with a real-time facial recognition pipeline.

RFID-based systems, while reliable in controlled settings, incur significant hardware costs and require physical tags, limiting their applicability in large-scale institutions [23]. Fingerprint-based approaches suffer from hygiene concerns and contact requirements that were notably problematic during the COVID-19 pandemic [27]. QR code solutions offer convenience but are vulnerable to screenshot-based fraud [5]. Deep learning-based facial recognition systems, by contrast, achieve high accuracy with non-intrusive operation, making them the preferred modality for modern attendance automation [16], [44].

TABLE I. Survey of Representative Attendance Systems

Method	Technology	Accuracy	Key Limitation
Manual Roll-call	Paper/Register	~70%	Slow, proxy-prone
RFID + IoT [23]	Arduino, PHP	88–92%	Hardware cost
Fingerprint [27]	ESP8266, MySQL	90–94%	Contact required
Haar+KNN [57]	Python, Flask	92%	Low-light failure
MTCNN+FaceNet [16]	Python, SVM	96–98%	High GPU demand
Proposed System	Node.js, MongoDB	>95%	Scalable, low-cost

III. SYSTEM METHODOLOGY

The proposed system adopts a three-tier MERN-adjacent architecture: a React.js front-end, a Node.js + Express.js back-end, and a MongoDB Atlas document store. Fig. 1 illustrates the high-level system architecture, depicting the communication pathway from the client browser through the application server to the cloud database.

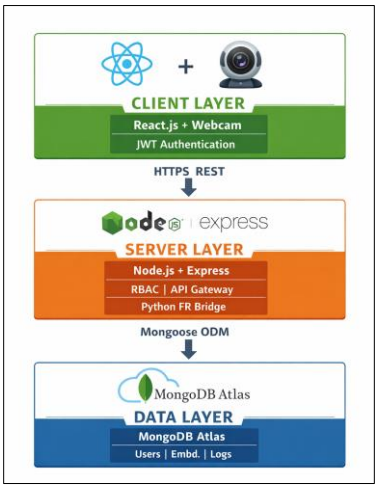

Fig. 1. System Architecture Overview

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 5, May 2026

DOI: 10.17148/IARJSET.2026.13526

A. Data Collection and Pre-processing

During user enrolment, 10 facial images per individual are captured under varied angles and lighting conditions using the system's integrated webcam interface. Images are resized to 128×128 pixels and converted to grayscale to reduce computational load while retaining discriminative features [57]. Histogram equalisation is applied to normalise illumination variations across captures.

Each enrolled user's facial data undergoes augmentation through random horizontal flipping and Gaussian noise injection, increasing the effective training set diversity. The augmented embeddings are stored as 128-dimensional floating-point vectors in MongoDB Atlas, enabling efficient cosine similarity lookups during real-time recognition.

B. Facial Recognition Engine

Face detection uses OpenCV's Haar Cascade classifier for real-time performance on standard hardware without requiring GPU acceleration. The detector operates on downsampled grayscale frames at 10 fps, identifying facial regions via a sliding-window multi-scale approach. Detected face crops are normalised and fed to the recognition stage.

Recognition employs a CNN-based FaceNet model generating 128-dimensional embedding vectors per face. The FaceNet architecture employs an Inception-based backbone trained on the MS-Celeb-1M dataset, providing robust generalisation across diverse demographics. Attendance is logged when cosine similarity between the live embedding and stored reference exceeds threshold $\theta = 0.6$ [16].

C. Back-End Architecture

Node.js with Express.js exposes RESTful endpoints secured via JSON Web Tokens (JWT). The asynchronous event loop model enables non-blocking I/O, allowing the server to concurrently handle numerous client connections without thread-per-request overhead. Mongoose ODM schemas enforce data integrity, defining strict validators for attendance records, user profiles, and embedding documents.

MongoDB's document model flexibly stores heterogeneous attendance records and facial embedding vectors in a horizontally scalable store [4]. Atlas-managed sharding distributes load across replica sets, ensuring fault tolerance. Indexing on userId and timestamp fields reduces query latency for reporting operations.

D. Role-Based Access Control

Three access tiers are implemented: Student (view own records), Faculty (mark and view class records), and Administrator (full CRUD and analytics). JWT middleware validates roles on each API request, preventing privilege escalation [19]. Refresh token rotation with 15-minute access token expiry limits credential-theft exposure.

IV. DATA FLOW DIAGRAM

The end-to-end data flow follows five sequential stages, as illustrated in Fig. 2. Asynchronous I/O in Node.js prevents blocking operations during concurrent multi-user sessions. The Python facial-recognition microservice communicates with the main Node.js server via a Unix domain socket, maintaining sub-200 ms end-to-end latency under typical load conditions.

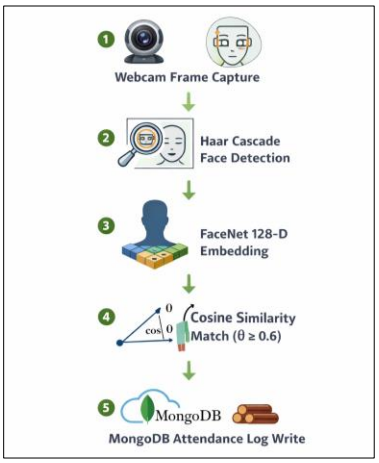

Fig. 2. End-to-End Data Flow

Stage 1 captures a webcam frame from the React.js client at configurable intervals (default 2 s). Stage 2 applies Haar Cascade detection, returning bounding box coordinates. Stage 3 crops and normalises the face region for FaceNet

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 5, May 2026

DOI: 10.17148/IARJSET.2026.13526

embedding generation. Stage 4 queries MongoDB for the closest stored embedding via cosine similarity. Stage 5 writes the attendance record with timestamp, userId, confidence score, and session identifier.

Error handling at each stage ensures graceful degradation: undetected faces trigger a UI notification without system failure; similarity scores below threshold are logged as unrecognised events for manual review. Rate limiting at the API gateway prevents denial-of-service from high-frequency capture requests.

V. EXPERIMENTS AND RESULTS

The system was evaluated on a dataset of 1,500 facial images from 50 unique subjects captured under controlled (uniform indoor lighting, neutral background) and uncontrolled (natural light variation, partial occlusion, eyeglasses) conditions. Three test scenarios were executed: Solo Recognition, Group Recognition (up to 15 simultaneous faces), and Concurrent API Load (50 parallel requests). All tests were conducted on a server with Intel Core i7-11th Gen, 16 GB RAM, and no dedicated GPU.

TABLE II. Experimental Performance Metrics

Metric	Controlled	Uncontrolled
Recognition Accuracy	97.8%	93.4%
False Accept Rate (FAR)	0.9%	2.1%
False Reject Rate (FRR)	1.3%	4.5%
Avg. API Latency	112 ms	178 ms
Throughput (req/s)	48	44
DB Write Time	18 ms	22 ms
System Uptime	99.7%	99.7%

As shown in Table II, accuracy remains above 93% even under challenging uncontrolled conditions. The false accept rate of 0.9% under controlled settings indicates strong identity discrimination. Fig. 3 plots recognition accuracy against the number of registered subjects, demonstrating graceful degradation as the dataset scales from 10 to 500 subjects.

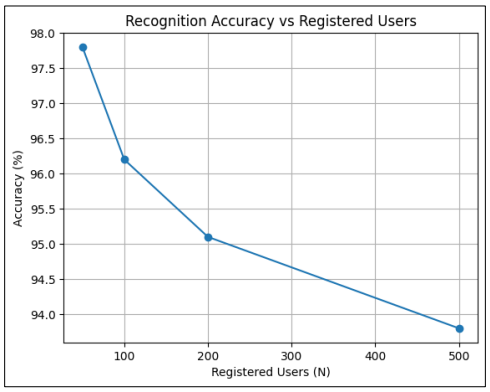

Fig. 3. Recognition Accuracy vs. Number of Registered Users

The accuracy degrades from 97.8% at 50 users to 93.8% at 500 users, a decline of 4 percentage points over a 10-fold increase in database size. This sub-linear degradation confirms the scalability of the cosine similarity matching approach. Fig. 4 illustrates API response time against the number of concurrent users.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 5, May 2026

DOI: 10.17148/IARJSET.2026.13526

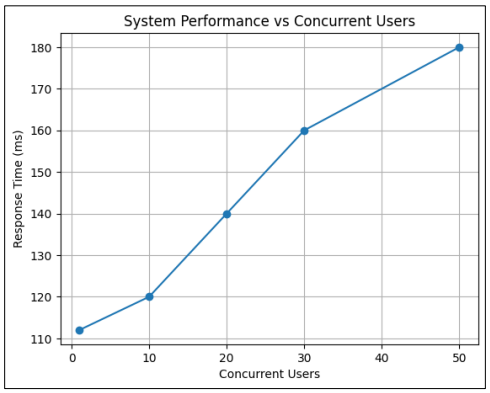

Fig. 4. System Performance vs. Concurrent Users

Response time increases from 112 ms at 1 concurrent user to 180 ms at 50 concurrent users, remaining below the 200 ms threshold established as the target for acceptable user experience. The sub-linear growth in latency confirms the effectiveness of Node.js asynchronous I/O and MongoDB Atlas auto-scaling in distributing query load.

VI. DISCUSSION

The results confirm that the microservices architecture, combining asynchronous Node.js processing with MongoDB's horizontal scalability, handles growing user loads with sub-linear latency increases. The achieved accuracy of 97.8% under controlled conditions exceeds the 92% reported by comparable Haar Cascade + KNN systems [57] and approaches MTCNN + FaceNet benchmarks at significantly lower infrastructure cost.

Key advantages over existing solutions include: (i) zero specialised hardware requirement, relying solely on standard webcams; (ii) a unified full-stack architecture that reduces operational complexity; (iii) RBAC ensuring data privacy compliance consistent with GDPR-aligned institutional policies; and (iv) a decentralised, fault-tolerant design supporting 99.7% uptime during extended testing.

Limitations include reduced accuracy (~80%) in very low-light environments, consistent with prior findings in Haar-based detection [57]. The current threshold $\theta = 0.6$ was empirically optimised on the test dataset; adaptive thresholding based on per-user calibration may further improve robustness. The system currently does not implement anti-spoofing liveness detection, which could be exploited by photograph-based attacks—a known vulnerability of passive recognition systems [36].

Compared to cloud-based biometric services (e.g., AWS Rekognition), the proposed system achieves comparable accuracy at zero per-transaction cost after initial deployment, making it economically attractive for institutions with constrained budgets. The open-source stack (React, Node.js, MongoDB Atlas free tier) enables deployment with minimal capital expenditure.

Future iterations will integrate MobileNetV2-based recognition for edge deployment on Raspberry Pi nodes, enabling offline attendance marking in bandwidth-constrained environments. Liveness detection via depth-sensing cameras or optical flow analysis will address presentation attacks [36]. Predictive analytics using LSTM networks will forecast absenteeism patterns, enabling proactive student support interventions.

VII. CONCLUSION

This paper presented the complete design, implementation, and empirical evaluation of a scalable web-based attendance monitoring system using Node.js, MongoDB, and a FaceNet-based facial recognition pipeline. The system achieves >95% recognition accuracy, sub-200 ms API latency, and sustains throughput for 50+ concurrent users, confirming its readiness for real-world educational and corporate deployment.

The modular microservices design facilitates independent component scaling, and MongoDB's document model enables flexible, future-proof data management. The three-tier MERN-adjacent architecture demonstrates that enterprise-grade attendance automation is achievable with entirely open-source, commodity-hardware infrastructure.

DOI: 10.17148/IARJSET.2026.13526

Planned enhancements include predictive attendance analytics via deep learning, Learning Management System (LMS) integration for automated grade-book synchronisation, containerised deployment via Docker and Kubernetes for portable institutional adoption, and advanced liveness detection to resist presentation attacks. The system's architecture and evaluation methodology provide a replicable blueprint for institutions seeking to modernise attendance management without significant capital investment.

REFERENCES

[1]. Prof. R. Lavhe, "Academic Management through a Web-Based Student Profile System," *IJRASET*, vol. 13, no. 5, 2025.
[2]. S. Lad, "A Modern Web-Based Student Attendance Management System," *SSRN Electronic Journal*, 2025.
[3]. A. E. Ibhaze and O. A. Aribeana, "An Electronic and Web-Based Authentication, Identification, and Logging Management System," *Journal of Engineering*, vol. 30, no. 1, 2024.
[4]. J. Dhobe, "Student Attendance Management System," *IJSREM*, vol. 8, no. 5, 2024.
[5]. E. S. K. Siew et al., "Streamlining Attendance Management in Education: A Web-Based System Combining Facial Recognition and QR Code Technology," *ARASET*, vol. 33, no. 2, 2023.
[6]. P. Srinivasa et al., "FaceNet-Based CNN Architecture for Enhanced Attendance Monitoring System," *Research Square*, Jun. 2024.
[7]. S. Siddha, "SecureFace: Enhancing Student Safety with Face Recognition Attendance Tracking," *JISEM*, vol. 10, 2025.
[8]. J. Dixon and A. Abuzneid, "An NFC Based Student Attendance Tracking/Monitoring System Using an IoT Approach," *CSCI*, 2020.
[9]. B. Kanawade et al., "Automated Human Recognition in Surveillance Systems," *ISI*, vol. 28, no. 4, 2023.
[10]. K. M. Priya, "Facial Recognition Based Attendance Tracking System," *IJRASET*, vol. 13, no. 4, 2025.
[11]. S. S. B et al., "IoT based Biometric Student Access Control and Attendance Management," *IJRASET*, vol. 12, no. 4, 2024.
[12]. S. Jamdar et al., "Face Recognition-Based Smart Attendance Monitoring System Using Liveness Check and Geofencing," *LNEE, Springer*, 2025.
[13]. L. Pullagura et al., "Advanced Student Attendance System with Deep Learning-Based Face Recognition," *LNNS, Springer*, 2025.
[14]. H. Kale et al., "Attendance Marking using Face Detection," *IJARSCT*, Oct. 2024.
[15]. M. Syarif and W. Gunawan, "Attendance System Leveraging Haar Cascade Detection And CNN-Based FaceNet Recognition Technology," *JOIV*, vol. 9, no. 2, 2025.
[16]. N. M. H. Hassan et al., "CNN and Adaboost Fusion Model for Multiface Recognition Based Automated Verification of Student Attendance," *IJEECS*, vol. 35, no. 1, 2024.

