iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VII **Month of publication:** July 2026

DOI: https://doi.org/10.22214/ijraset.2026.84292

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII July 2026- Available at www.ijraset.com

Aero Face: Biometric Attendance & Administration System

Mycharla Madhavkumar[1], Tulasi Miryala[2]
[1]*Student, Master of Computer Applications (MCA), AQJ College for PG Studies (affiliated to Andhra University), Visakhapatnam, Andhra Pradesh, India*
[2]*Assistant Professor, Department of MCA, AQJ College for PG Studies (affiliated to Andhra University), Visakhapatnam, Andhra Pradesh, India*

Abstract: ***Even though most colleges and exam centers in Andhra Pradesh still use traditional attendance methods, such as paper registers or card machines, these approaches run into real problems in practice. Staff attendance is often marked as present by a colleague, even when that person is not actually there. Because most lab environments are dusty, fingerprint scanners jam frequently. And cloud-based systems go down whenever the institution's network goes down. In this paper, we propose a novel attendance system called AeroFace, which uses the webcam already available on the lab PC to capture images of employees, runs face recognition in the browser, and stores all data within the institution's local network, with no cloud dependency, no dedicated hardware, and no internet connection requirement after initial setup. The recognition engine uses face-api.js, which runs FaceNet-based neural network models through TensorFlow.js on the browser's WebGL-accelerated GPU pipeline. During enrollment, a 128-dimensional descriptor vector is extracted from an employee's photograph and stored; during live scanning, the same pipeline processes the employee's webcam feed, and a Euclidean-distance comparison is made against the stored descriptor vectors. The result of the comparison is logged along with information such as GPS location, a photo snapshot of the employee, and a confidence score. The back-end of the system is a 34-endpoint Node.js and Express API, which supports 12 administrative modules, including attendance logs, employee management, exam-slot booking for up to 120 systems, an income and expense ledger, salary computation with printable payslips, travel planning, and supervisor approval of transactions. The system was validated in a college laboratory environment through 25 functional test cases, divided into unit, integration, and multi-client network tests. The total size of the recognition pipeline is approximately 6.5 MB, and it requires no internet connection at runtime. This shows that a browser-native, deep-learning-based recognition system is a practical, low-cost, and hardware-independent alternative to commercial biometric attendance systems.***

Keywords: Biometric Attendance, Face Recognition, face-api.js, TensorFlow.js, FaceNet, Node.js, Express.js, REST API, Role-Based Access Control, GPS Attendance, Examination Management, Offline LAN Deployment

I. INTRODUCTION

The digital transformation of academic institutions and large-scale examination centers has given rise to the need for attendance systems that can function autonomously, in a tamper-proof fashion, without depending on fragile infrastructure. A common scene plays out at examination centers across Visakhapatnam: hundreds of candidates attend sessions conducted across many lab rooms. Before an examination begins, the supervisor of a particular lab must distribute a paper register to all candidates present and circulate among them to collect signatures. This process is time-consuming, prone to error, and often subject to manipulation. Typically, a candidate signs in on behalf of a late-arriving colleague, or a name gets entered twice into the register by mistake. By the end of the event, it is often not possible to obtain an accurate count of who actually attended.

The most common alternative to a paper-based register is a fingerprint-based attendance system. A fingerprint scanner typically costs between Rs.8,000 and Rs.25,000 and needs periodic calibration. Fingerprint scanners are also notoriously unreliable when chalk dust collects on the operator's fingers, and high humidity is a well-known cause of failed reads. Cloud-based biometric platforms remove the calibration requirement, but fail in a different way: a single network outage is enough to bring the entire attendance system to a halt on exam day. AeroFace is a system built primarily for colleges and examination centers in Andhra Pradesh. It does not depend on specialized biometric hardware or a cloud-based server. Facial recognition runs within the browser using GPU-accelerated deep learning inference, paired with a self-contained REST API and an admin panel. Attendance, finance, payroll, and logistics data are all managed within a single offline system. AeroFace runs on lab PCs with webcams that are already installed, on a network that colleges already have — so no extra hardware or subscription is required.

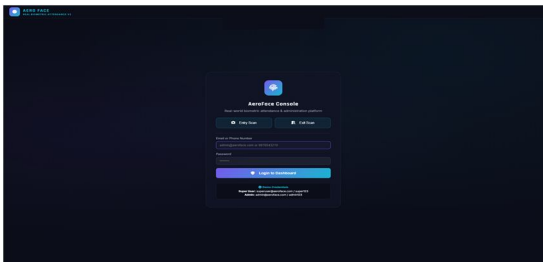


Fig. 1. AeroFace login screen with Entry/Exit biometric scan access and role-based sign-in.

A. *Problem Definition*

The operational challenges motivating this work are: proxy attendance marked by staff on behalf of absent colleagues; recurring maintenance costs and reliability issues from dedicated fingerprint hardware; disruption caused by dependence on cloud connectivity; fragmentation of attendance, financial, and booking records across separate registers and spreadsheets; the absence of a GPS-tagged, photo-verified audit trail for resolving disputes; and manual, error-prone salary computation from attendance data.

B. *Objectives*

1) Implement browser-based real-time face detection and recognition using face-api.js and TensorFlow.js, removing the need for specialized biometric hardware.
2) Develop a 34-endpoint RESTful API using Node.js and Express to manage attendance and administrative data through a file-based JSON database.
3) Provide GPS-tagged, photo-verified, timestamped attendance logging with a confidence score for each biometric scan.
4) Implement Role-Based Access Control with distinct Super User and Admin privilege levels.
5) Integrate a financial ledger, exam-slot booking with capacity management for up to 120 networked systems, and automated salary computation with printable payslips.
6) Support fully offline LAN deployment, with all recognition model weights bundled locally to eliminate internet dependency after installation.

II. LITERATURE REVIEW

Automated face recognition has progressed over six decades from geometric and statistical models to deep convolutional networks capable of surpassing human-level verification accuracy. Turk and Pentland's Eigenfaces [3] introduced Principal Component Analysis for representing faces as combinations of low-dimensional basis images — the first computationally practical recognition method. Belhumeur et al.'s Fisherfaces [8] used Linear Discriminant Analysis to improve robustness to changes in illumination, and later approaches, such as Local Binary Pattern Histograms, offered efficient feature extraction that could handle monotonic illumination changes. However, all of these methods are limited by their reliance on linear features, which cannot capture the overall structure of a face. Deep learning later transformed face recognition. In 2014, Taigman et al. [9] used a nine-layer convolutional network to reach 97.35% accuracy on the Labeled Faces in the Wild (LFW) benchmark — near human-level recognition. A year later, Schroff et al. proposed FaceNet [1], which used a triplet-loss function to map face images directly to a 128-dimensional Euclidean space, where images of the same person land close together and images of different people land far apart. AeroFace exploits this embedding space for recognition: at inference time, a single descriptor is computed for a new face image and compared by distance against the descriptors of already-enrolled faces. No retraining is required for the recognition model.

face-api.js, developed by Muhler [2], ports four neural networks — SSD MobileNetV1, TinyFaceDetector, FaceLandmark68Net/TinyNet, and a FaceNet-based recognition network — for execution in the browser through TensorFlow.js [5]. Inference is executed by TensorFlow.js using WebGL for GPU-based execution in any browser that supports WebGL 2.0 [11]. This is the technical foundation AeroFace's offline, browser-native pipeline is built on. On the server side, REST remains the dominant architectural style, as formalized in Fielding's dissertation [4], and Node.js with Express is used for its event-driven, non-blocking I/O model, which makes it easy to handle a high volume of concurrent attendance requests distributed across a shared LAN.

Many commercial biometric attendance products, such as fingerprint time-attendance terminals or cloud-based recognition APIs, suffer from several limitations: high dedicated hardware cost, a continuous internet connectivity requirement, or per-scan charges. In this paper, we present AeroFace, a face-recognition-based attendance system that combines FaceNet-based browser recognition with a self-hosted, offline Node.js application server.

III. SYSTEM ARCHITECTURE AND METHODOLOGY

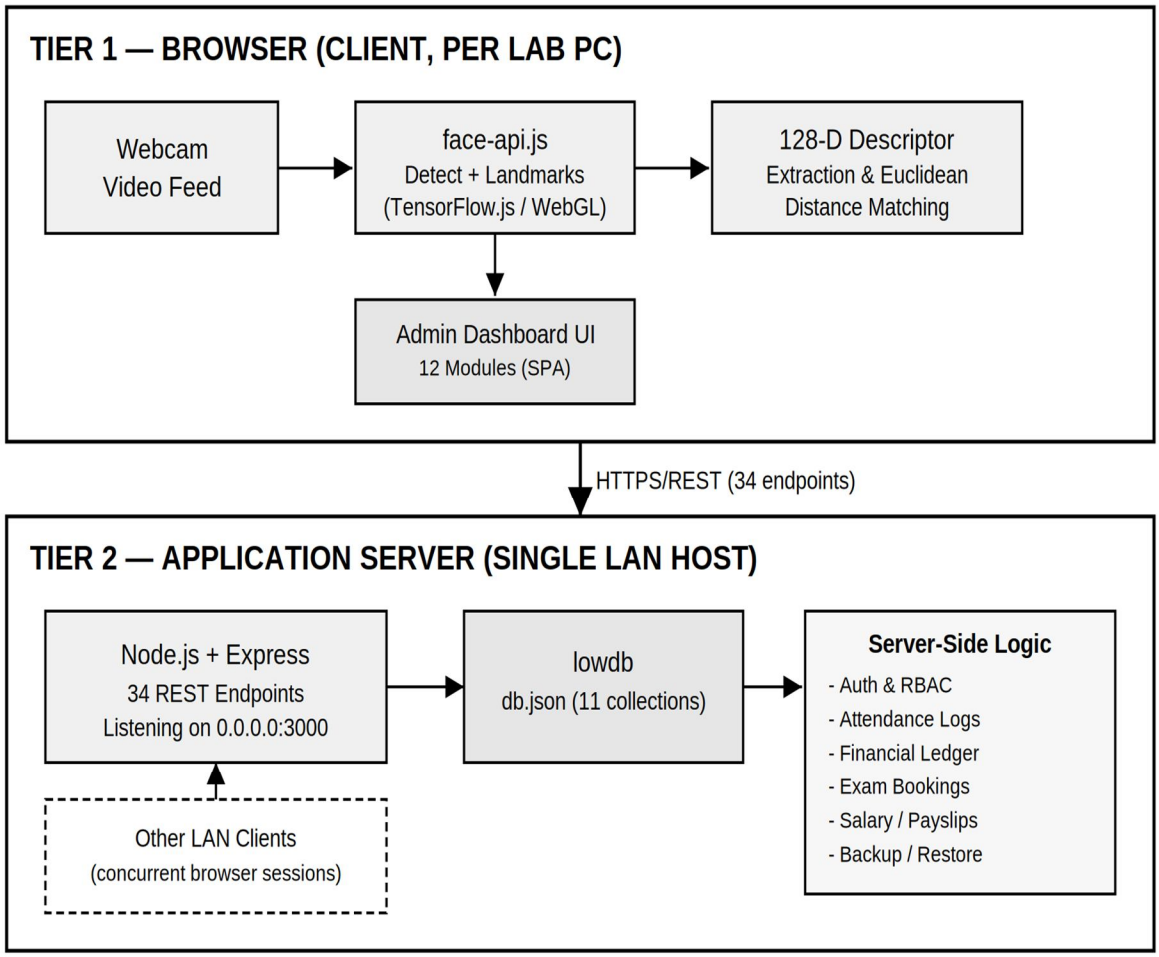


Fig. 2. The architecture of the two-tier AeroFace system: browser-based recognition (Tier 1) and the self-hosted Node.js/Express application server (Tier 2).

AeroFace is built as a two-tier system. The first tier is the presentation and recognition layer — a pure client-side solution that runs entirely in the browser. It captures video from the device's webcam, detects faces, localizes face landmarks, and extracts 128-dimensional face descriptors. Recognition is done by matching the live face descriptor against the descriptors of employees already enrolled in the system; no raw face image ever leaves the browser. Tier 2, the application layer, is a Node.js and Express server exposing 34 RESTful endpoints for authentication, employee management, attendance logging, financial records, exam bookings, and payroll, backed by lowdb, a zero-configuration JSON file database. The server listens on 0.0.0.0 so that any device on the institution's LAN can connect without internet access. Fig. 2 shows the overall two-tier architecture.

A. Face Enrollment

An admin registers each employee through the Employee Registry module, uploading or capturing a photograph along with name, ID, designation, and lab assignment. The photograph is processed by the TinyFaceDetector to locate the face, FaceLandmark68 to find 68 key points, and the FaceRecognition network to compute a 128-float descriptor, which is stored against the employee record in db.json. Fig. 3 shows the schema of the eleven lowdb collections and their relationships.

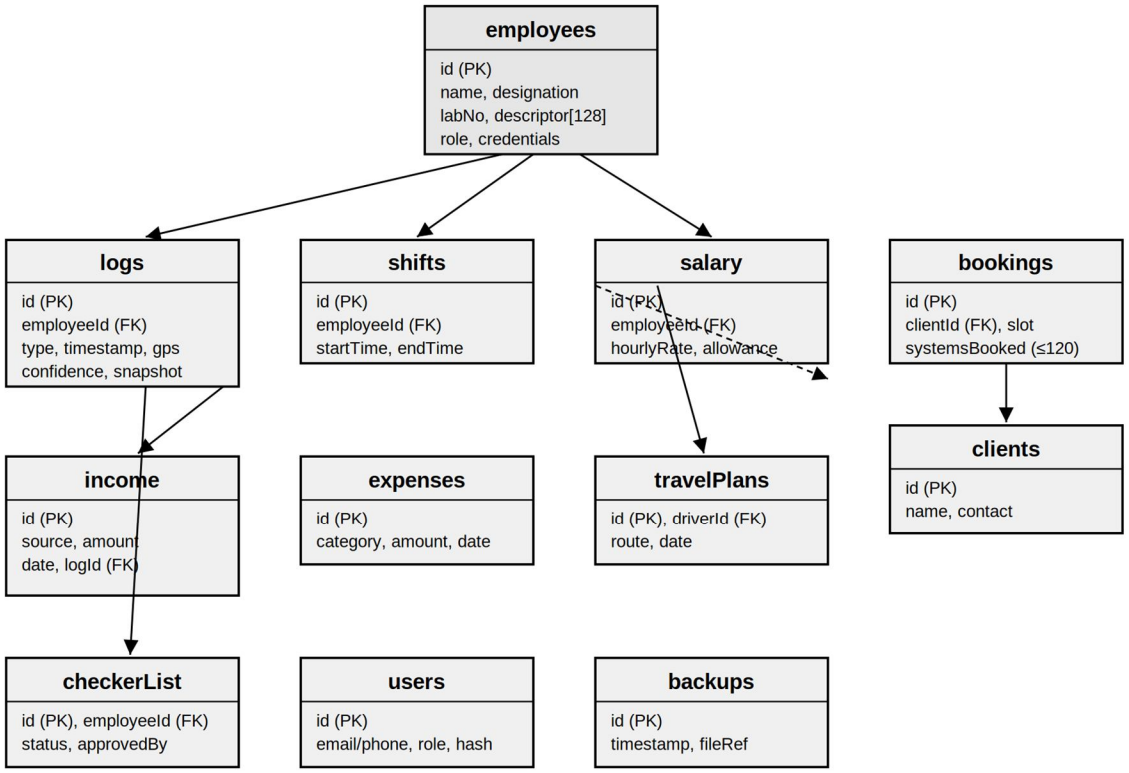


Fig. 3. Entity-relationship schema of the lowdb (JSON) data store spanning eleven collections.

B. Live Attendance Scanning

At the point of entry, the employee opens the AeroFace URL, grants camera permission, and the browser streams the webcam feed to a canvas element with a live bounding box drawn around any detected face. On clicking Identify and Record, the current frame is processed to extract a 128-float descriptor, which is compared against all stored descriptors using Euclidean distance:

$$d(p, q) = \sqrt{[\ \Sigma\ (p_i - q_i)^2\ ]},\ \ i = 1 \text{ to } 128$$

A FaceMatcher with a distance threshold of 0.5 is used; a match below this threshold identifies the employee, with confidence reported as (1 − distance) × 100%. For every scan, the system records the employee ID, scan type, a timestamp in ISO 8601 format, GPS coordinates, the terminal's IP address, the employee's assigned zone, their confidence score, and a JPEG snapshot. All of this is logged through a POST /api/logs request, which also creates a corresponding income record. Fig. 4 shows the complete pipeline for face recognition and attendance logging.

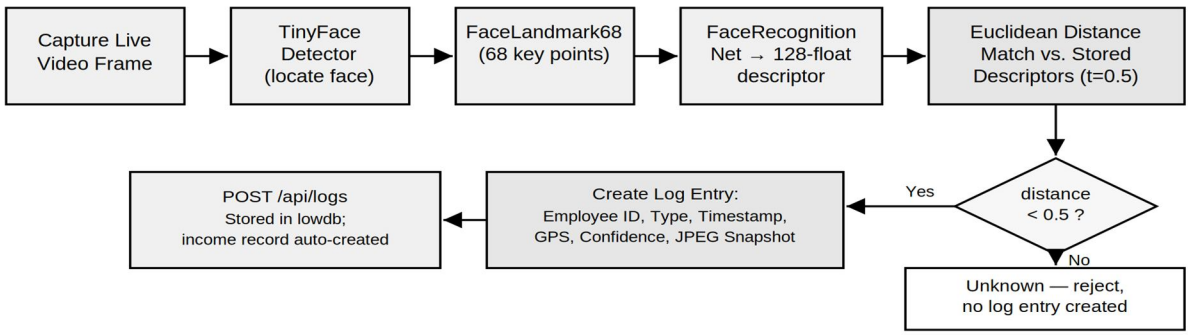


Fig. 4. Face recognition and attendance-logging pipeline, from frame capture to log entry.

C. Administrative Dashboard

AeroFace contains eleven more modules available behind the login screen: exam time-slot booking for up to 120 lab stations; a financial ledger that records daily income and expenses and automatically calculates a balance; a salary module that computes monthly pay for staff from attendance records, with printed payslips for individual staff; travel planning for the deployment of drivers and staff; a supervisor checker/approval workflow; and a backup module that exports the entire database to a timestamped file for restore. Roles are defined for both login and module access — Super Users can create bookings and change the salary rate for staff, while Admin users work with attendance records, the financial ledger, client records, and other day-to-day data. The biometric scanner is available from the login screen without authentication, so a person stationed at the entry point can scan arrivals as they come in. This is illustrated in Fig. 5, which also shows the scan-logging exchange between the browser and the server, and the login process for each role.

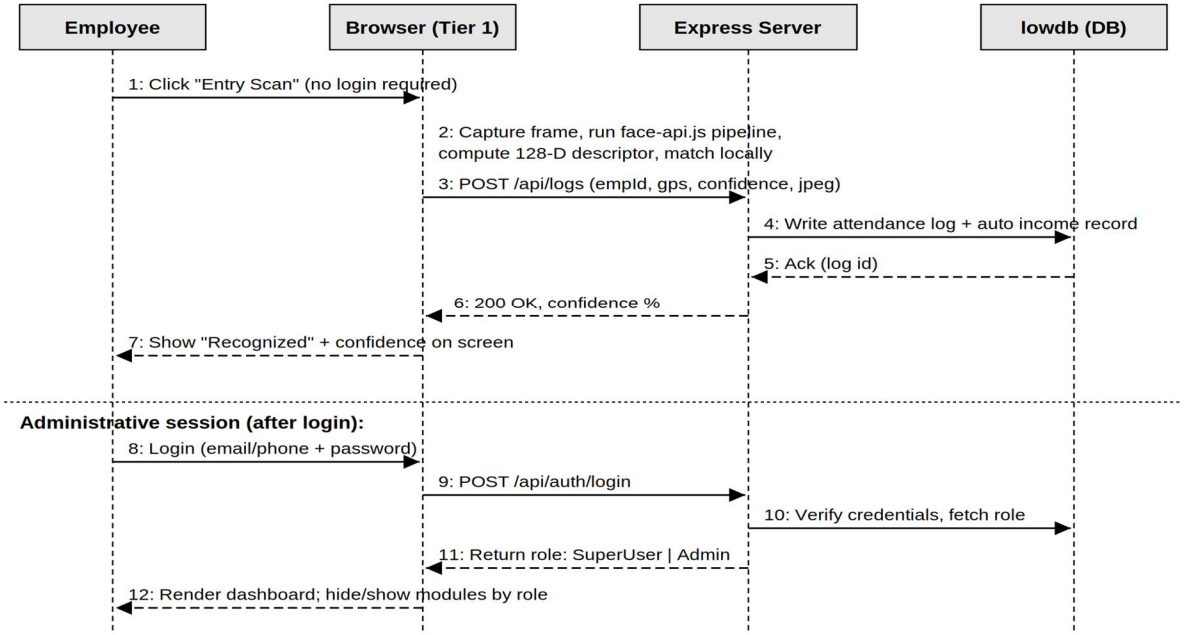


Fig. 5. Sequence diagram: biometric scan logging (top) and role-based authentication flow (bottom).

IV. RESULTS AND DISCUSSION

AeroFace was evaluated in the college laboratory at AQJ College for PG Studies, Visakhapatnam, with five employees who were all enrolled in the system, under standard fluorescent lighting. Twenty-five functional test cases were executed, covering user authentication, face enrollment, live face recognition, role-based access restrictions, financial ledger creation and editing, data backup and restore, and multi-client access from three different PCs connected to the same LAN. All 25 test cases passed. Figs. 6 and 7 are screenshots of the Overview Dashboard as it appears to the two access levels. Both show the same real-time terminal statistics, GPS-tagged system status, and live entry stream; the difference between the two is which modules appear in the left-side navigation, as described in Section III-C.

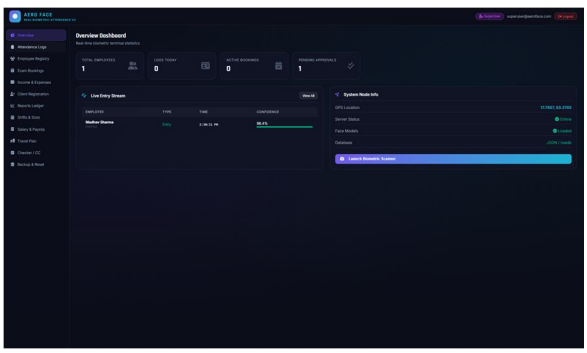


Fig. 6. Overview Dashboard — Super User session.

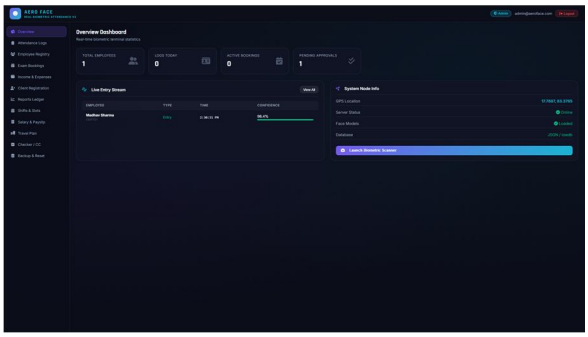


Fig. 7. Overview Dashboard — Admin session.

On the test hardware (Intel Core i5 machines with integrated graphics), face detection took approximately 180 ms per frame, and the full identify-and-record cycle completed in 320–380 ms with five enrolled employees. GPS acquisition functioned correctly when location permission was granted; when blocked, the system fell back to fixed lab coordinates without failure. API response time for CRUD operations over the LAN remained under 15 ms.

Testing with a non-enrolled face correctly returned an unrecognized result with no log entry created. Testing with a photograph of an enrolled person held to the webcam was, as expected given the current distance-threshold configuration, recognized as a match, confirming a known limitation around liveness detection that is addressed in Section VI.

The three bundled model files together total approximately 6.5 MB and require no internet connection at runtime. Because all component technologies (Node.js, Express, face-api.js, TensorFlow.js, and lowdb) are open-source and MIT-licensed, AeroFace carries no licensing cost, compared to commercial biometric terminals that typically cost Rs.15,000 to Rs.50,000 per unit plus annual maintenance.

V. CONCLUSION

AeroFace demonstrates that browser-native deep learning inference, built on face-api.js and TensorFlow.js, can deliver a hardware-independent biometric attendance solution suitable for colleges and examination centers. By coupling FaceNet-based recognition with a self-hosted Node.js/Express REST API and a twelve-module administrative dashboard, the system consolidates attendance, finance, payroll, booking, and logistics functions that are traditionally fragmented across manual registers — while operating entirely offline on a local network. The result is a cost-effective, low-maintenance alternative to commercial biometric hardware, one that remains fully functional without continuous internet connectivity.

A notable outcome of this work is how much administrative overhead could be consolidated into a single platform: attendance registers, expense sheets, salary calculations, booking logs, and approval chains that were previously maintained separately are now handled through one dashboard, backed by a single JSON data store that can be backed up and restored in one step.

VI. FUTURE SCOPE

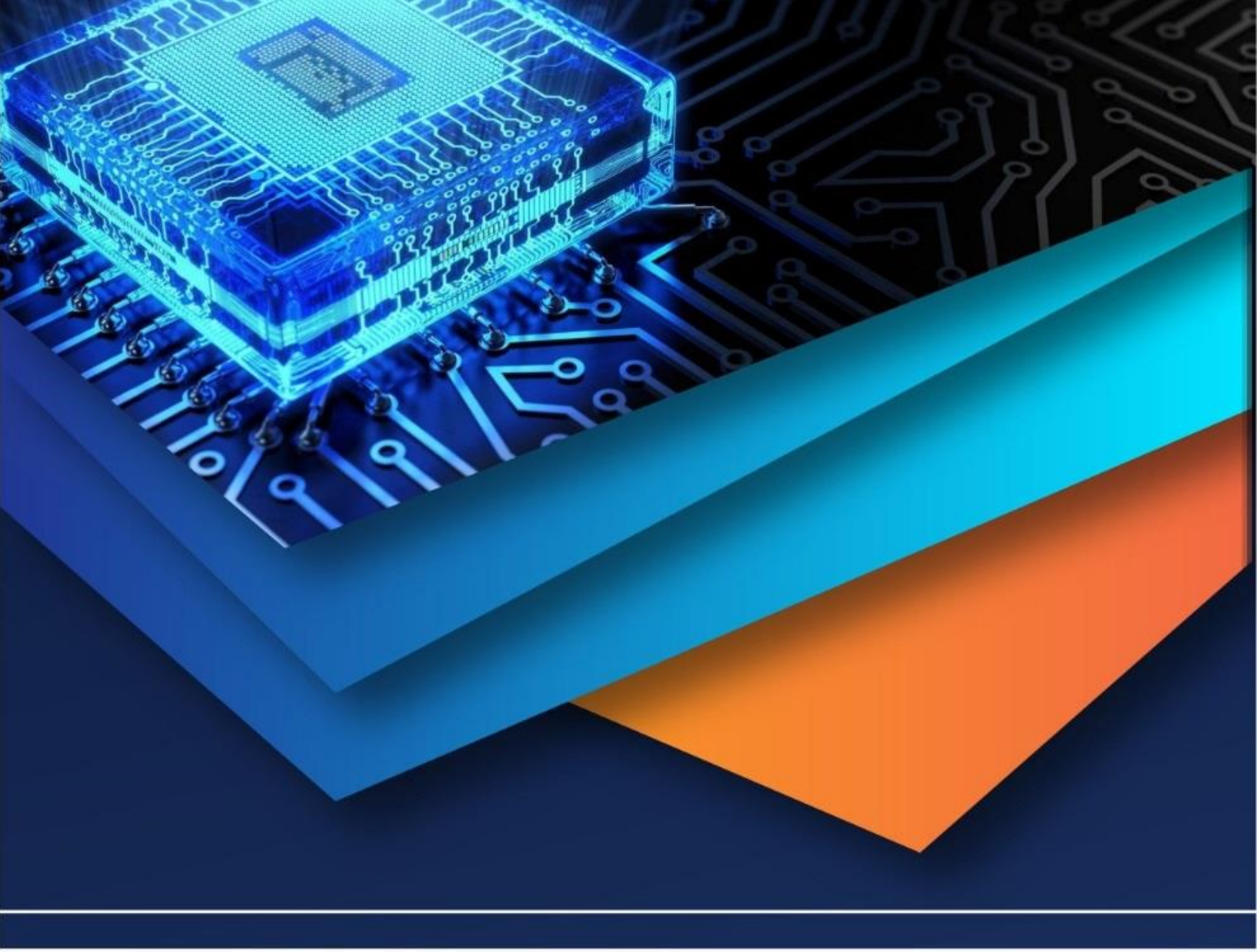
1) Liveness detection (eye-blink or depth-based) to guard against photograph-based spoofing, addressing the limitation noted in Section IV.
2) Multi-camera support for high-throughput entry and exit points during large examinations.
3) A React Native mobile client for Android and iOS attendance scanning.
4) Migration from lowdb to SQLite for improved performance at scale (500+ employees).
5) HTTPS support with self-signed certificates for secure LAN connections.
6) Automated PDF report generation with attendance and financial charts.
7) Dashboard analytics using Chart.js or D3.js for attendance and punctuality trends.

VII. ACKNOWLEDGMENT

The authors thank the Department of Computer Applications, AQJ College for PG Studies (affiliated to Andhra University), Visakhapatnam, for laboratory access and guidance during the development and testing of AeroFace. An AI writing assistant was used for wording and document formatting in the preparation of this manuscript; all design, implementation, testing, and resulting data, figures, and conclusions are the authors' own work.

REFERENCES

[1] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2015.
[2] V. Muhler, "face-api.js: JavaScript Face Recognition API for the Browser and Node.js Implemented on top of tensorflow.js core," GitHub Repository, 2018.
[3] M. Turk and A. Pentland, "Eigenfaces for Recognition," Journal of Cognitive Neuroscience, vol. 3, no. 1, pp. 71–86, 1991.
[4] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, Univ. of California, Irvine, 2000.
[5] N. Sinai, "TensorFlow.js: Machine Learning for the Web and Beyond," in Proc. Systems for Machine Learning Workshop, NeurIPS, 2018.
[6] Node.js Foundation, "Node.js Documentation v20 LTS," 2023. [Online]. Available: https://nodejs.org/en/docs
[7] Express.js Team, "Express.js 4.x API Reference," 2023. [Online]. Available: https://expressjs.com/en/4x/api.html
[8] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 711–720, 1997.
[9] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," in Proc. IEEE CVPR, 2014, pp. 1701–1708.
[10] MDN Web Docs, "MediaDevices.getUserMedia()," Mozilla Developer Network, 2024.
[11] W3C Working Group, "WebGL 2.0 Specification," World Wide Web Consortium, 2021.
[12] Ministry of Electronics and Information Technology, Government of India, "Digital Personal Data Protection Act, 2023," Gazette of India, 2023.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY