International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Design and Implementation of an IOT-Based Smart Attendance Monitoring System Using Fingerprint Authentication

Prof. Mayank Kurchaniya[1], Prof. Nitin Jain[2], Prof. Deepti Jain[3], Dr. Sweta Singh[4]

[1234] Department of Computer Science & Engineering, Sagar Institute of Science & Technology,

Gandhi Nagar, Bhopal, Madhya Pradesh, India

Abstract

Attendance recording in colleges and workplaces is still largely manual, which wastes lecture time, invites proxy attendance and produces records that are hard to analyse. This paper presents the design and implementation of an IoT-based Smart Attendance Monitoring System that authenticates users through fingerprint biometrics and manages records through a web application built on the MERN stack (MongoDB, Express.js, React.js and Node.js). The hardware node uses an ESP32 Wi-Fi microcontroller interfaced with an R307 optical fingerprint sensor, an OLED display and a buzzer. Fingerprint templates are matched on the sensor itself, and only the matched template ID, device ID and timestamp are sent over HTTPS to a RESTful Node.js/Express server. The server validates the device, maps the template ID to a registered user, prevents duplicate entries within a session and stores the record in MongoDB. A React.js dashboard gives administrators and faculty role-based access to live attendance, subject-wise and date-wise reports, defaulter lists and CSV/PDF export. The system was built and tested in a classroom setting. The results show that it records attendance in about one to two seconds per person, removes proxy attendance, and gives an accurate, real-time and low-cost alternative to manual registers and standalone biometric machines.

Keywords: Internet of Things (IoT), Biometric Authentication, Fingerprint Recognition, ESP32, R307 Sensor, MERN Stack, MongoDB, REST API, Smart Attendance System, Real-Time Monitoring

I. Introduction

Attendance is a basic measure of student engagement and employee discipline. Many Indian institutions also need it to enforce minimum-attendance rules before examinations. Yet most classrooms still use paper registers or roll calls. A roll call in a class of 60 students can take five to ten minutes of each lecture, and the resulting records must be typed into spreadsheets by hand before they can be analysed.

Manual methods have three well-known weaknesses. First, they are slow and repetitive. Second, they are easy to manipulate, because one student can answer or sign for another (proxy attendance). Third, the data is stored in a form that cannot be searched, shared or reported on in real time. RFID cards and QR codes speed up marking, but a card or a code can be handed to someone else, so they do not prove that the right person is present [1], [2].

Biometric traits such as fingerprints are unique to each person and cannot be lent or forgotten [3]. Low-cost optical fingerprint modules now perform enrolment, storage and matching on the module itself, and Wi-Fi microcontrollers such as the ESP32 can send results directly to a cloud server [4], [5]. This makes it possible to build an attendance terminal that is both secure and connected, at a fraction of the cost of commercial biometric machines.

This paper proposes a complete IoT-based Smart Attendance Monitoring System. Its main contributions are:

1. A fingerprint terminal built on the ESP32 and R307 sensor that authenticates users in about one second and sends attendance events over Wi-Fi.
2. A RESTful back end built with Node.js and Express.js, with MongoDB as a flexible document database for users, subjects, devices and attendance logs.
3. A responsive React.js dashboard with role-based access for administrators, faculty and students, giving live attendance, analytics and report export.
4. Security measures that include device API keys, JWT-based user authentication, encrypted transport and duplicate-entry prevention.

The rest of the paper is organised as follows. Section II reviews related work. Section III states the problem and objectives. Section IV describes the system architecture and hardware. Section V presents the software design. Section VI covers implementation and results, Section VII discusses security and limitations, and Section VIII concludes the paper.

II. Literature Review

Research on automated attendance falls into four groups: card-based, vision-based, biometric standalone and IoT-connected biometric systems.

A. Card-based systems. Arulogun et al. [1] built an RFID system in which each student carries a tag that is read at the classroom door. RFID is fast and contactless, but a tag identifies the card rather than the person, so proxy attendance remains possible. QR-code and Bluetooth methods share the same weakness.

B. Vision-based systems. Kar et al. [2] used face recognition to mark attendance from classroom images. Face recognition needs no contact, but accuracy drops with poor lighting, pose changes and occlusion, and it needs a camera and higher processing power, which raises cost.

C. Biometric foundations. Jain, Ross and Prabhakar [3] describe fingerprints as one of the most mature biometric traits because they are unique, stable over a lifetime and cheap to capture. Maltoni et al. [6] explain minutiae-based matching, which is now built into low-cost optical modules such as the R307 [4].

D. IoT-connected fingerprint systems. Kamble et al. [7] combined a NodeMCU ESP8266 with an R307 sensor and stored attendance in an online Google Sheet. Their design is simple, but a spreadsheet offers no user roles, data validation or analytics. Olawole and Adebayo [8] used an ESP32, a fingerprint module and an OLED display with a locally hosted PHP web interface, which limits remote access. Abhirami et al. [9] proposed GateSense, an ESP32 system that combines a fingerprint sensor, RFID and IR sensors with a Firebase back end to track continuous presence.

E. Research gap. Most reported systems use spreadsheets, local PHP/MySQL servers or vendor cloud services. Few offer a modern full-stack web application with a secure REST API, role-based dashboards and a scalable document database. The proposed system fills this gap by pairing an ESP32 fingerprint terminal with a MERN-stack application.

Table I. Comparison of existing attendance approaches

Approach	Proxy-proof	Real-time access	Typical cost	Main limitation
Paper register / roll call	No	No	Very low	Slow, error-prone, no analytics
RFID / QR code [1]	No	Yes	Low	Card or code can be shared
Face recognition [2]	Yes	Yes	High	Lighting and pose sensitivity, needs camera and compute
Standalone biometric machine	Yes	No	Medium	Data locked in device, manual download
IoT fingerprint + spreadsheet / PHP [7], [8]	Yes	Partial	Low	Limited security, roles and analytics
Proposed: ESP32 fingerprint + MERN	**Yes**	**Yes**	**Low**	Needs Wi-Fi (offline buffer reduces impact)

III. Problem Statement and Objectives

A. Problem statement. Institutions need an attendance method that proves the physical presence of the correct person, records it instantly, and makes the data available to authorised users anywhere. Existing low-cost systems either allow proxies (cards, codes) or keep data in places that are hard to secure, query and scale (spreadsheets, local servers, standalone devices).

B. Objectives. The system was designed to meet the following objectives:

1. Authenticate each user by fingerprint so that attendance cannot be marked by another person.
2. Send each attendance event from the terminal to a central server in real time over Wi-Fi.
3. Store users, subjects, sessions and attendance in a secure, scalable MongoDB database.
4. Provide a role-based web dashboard for administrators, faculty and students.
5. Generate subject-wise, date-wise and student-wise reports and defaulter lists, with CSV/PDF export.
6. Keep the hardware cost of one terminal low enough for every classroom.
7. Continue to record attendance during short network outages and sync it later.

IV. System Architecture and Hardware Design

The system follows a three-layer IoT architecture: a perception layer (fingerprint terminal), a network and application layer (REST server), and a presentation layer (web dashboard). Fig. 1 shows how the layers connect.

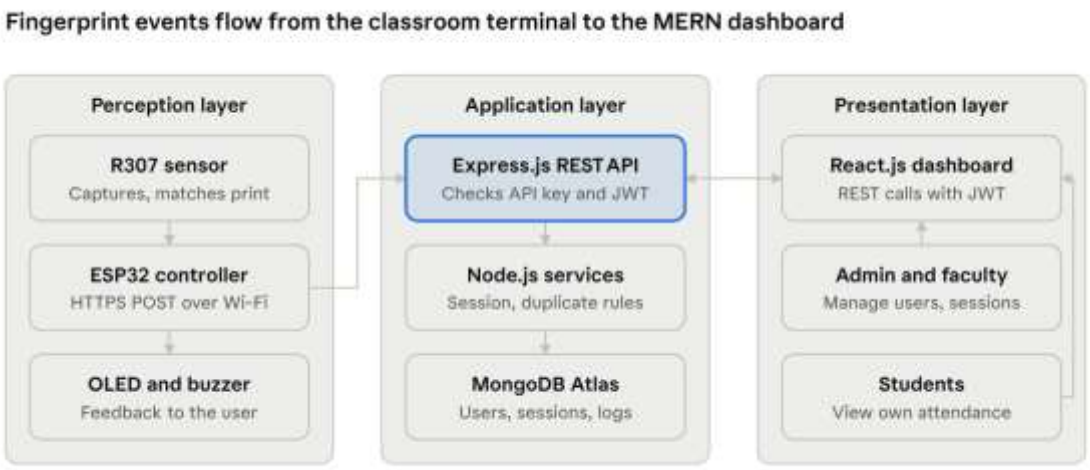


Fig. 1. Three-layer system architecture

Fig. 1. Three-layer architecture of the proposed system. The terminal posts events to the Express.js API, which stores them in MongoDB; the React.js dashboard reads them through the same API.

A. Perception layer. Each classroom has one terminal. The R307 sensor captures the fingerprint, extracts minutiae and searches its internal template library. On a match it returns a template ID and a confidence score to the ESP32 over UART. The ESP32 then shows the result on the OLED, sounds the buzzer and sends the event to the server.

B. Network and application layer. The ESP32 posts a JSON payload to the Express.js API over HTTPS. The Node.js server checks the device API key, resolves the template ID to a user, checks the active session and duplicate rules, and writes the record to MongoDB Atlas.

C. Presentation layer. The React.js single-page application calls the same API with a JWT token. Administrators manage users and devices, faculty start sessions and view live attendance, and students view their own percentage.

D. Hardware components. Table II lists the components of one terminal. Specifications are taken from the manufacturer datasheets [4], [5].

Table II. Hardware components of the fingerprint terminal

Component	Key specification	Role in system
ESP32 DevKit V1	Dual-core 32-bit Xtensa LX6, up to 240 MHz, 520 KB SRAM, 802.11 b/g/n Wi-Fi, Bluetooth	Main controller and Wi-Fi client
R307 optical fingerprint sensor	500 dpi, 1000-template storage, FAR < 0.001 %, FRR < 1.0 %, search time < 1 s, UART 57600 baud	Capture, store and match fingerprints
0.96-inch OLED (SSD1306)	128 × 64 pixels, I2C interface	Shows prompts, name and status
Buzzer and LEDs (green, red)	3.3 V digital outputs	Audio and visual feedback
5 V adapter / power bank	5 V, 1 A minimum	Power supply

E. Circuit connections. The R307 is connected to the second hardware serial port of the ESP32, and the OLED uses the default I2C pins, as shown in Table III.

Table III. Pin connections

Module pin	ESP32 pin
R307 VCC / GND	5 V (VIN) / GND
R307 TX	GPIO 16 (RX2)
R307 RX	GPIO 17 (TX2)
OLED SDA / SCL	GPIO 21 / GPIO 22
Buzzer	GPIO 25
Green LED / Red LED	GPIO 26 / GPIO 27

The approximate cost of one terminal is ₹2,000 to ₹2,500 at Indian retail prices, well below commercial networked biometric machines.

V. Software Design

A. Technology stack. The application uses the MERN stack, so JavaScript runs on every tier and JSON is the single data format from the device to the database [10]–[13]. The firmware is written in C++ using the Arduino core for ESP32 and the Adafruit Fingerprint Sensor library.

Table IV. Software stack

Tier	Technology	Purpose
Firmware	Arduino C++ (ESP32 core), Adafruit Fingerprint, HTTPClient, ArduinoJson, SSD1306	Sensor control, Wi-Fi, HTTPS requests, display
Database	MongoDB Atlas with Mongoose ODM	Document storage, schema validation, indexes
Back end	Node.js, Express.js	REST API, business rules, authentication

Tier	Technology	Purpose
Front end	React.js, React Router, Axios, Chart.js	Dashboard, routing, charts
Security	bcrypt, JSON Web Token (JWT), HTTPS, Helmet, CORS	Password hashing, sessions, transport security

B. Database design. MongoDB stores data as flexible documents grouped into collections. Table V shows the main collections. A compound unique index on studentId, sessionId in the attendance collection stops the same student being marked twice in one session at the database level.

Table V. MongoDB collections

Collection	Key fields	Description
users	_id, name, email, passwordHash, role (admin, faculty, student), rollNo, department	All system users
fingerprints	userId, deviceId, templateId, enrolledAt	Maps each sensor template ID to a user
devices	deviceId, apiKeyHash, classroom, status, lastSeen	Registered terminals
subjects	code, name, facultyId, semester, department	Courses and their faculty
sessions	subjectId, facultyId, deviceId, startTime, endTime, isActive	One lecture or lab period
attendance	studentId, sessionId, subjectId, timestamp, status, source	One record per student per session

C. REST API design. The Express.js server exposes the endpoints in Table VI. Device routes are protected by a per-device API key sent in a request header, and user routes are protected by a JWT with role checks.

Table VI. Main API endpoints

Method	Endpoint	Access	Function
POST	/api/auth/login	Public	Verify credentials, return JWT
POST	/api/users	Admin	Create a user
POST	/api/fingerprints/enroll	Admin	Link a template ID to a user
POST	/api/sessions/start	Faculty	Open an attendance session for a subject
PATCH	/api/sessions/:id/end	Faculty	Close the session
POST	/api/device/attendance	Device key	Record a fingerprint match
GET	/api/attendance/live/:sessionId	Faculty	Live list of present students
GET	/api/reports/subject/:id	Admin, Faculty	Subject-wise report and defaulters
GET	/api/attendance/me	Student	Own attendance percentage

A typical device request body is:

```
{
 "deviceId": "CLS-101",
 "templateId": 23,
 "confidence": 142,
 "timestamp": "2026-11-10T09:05:12+05:30"
}
```

D. Enrolment algorithm. An administrator selects a student on the dashboard and puts the terminal into enrolment mode.

1. The server sends the next free template ID to the device.
2. The student places a finger; the sensor captures image 1 and converts it to character file 1.
3. The student lifts and places the same finger again; the sensor creates character file 2.
4. The sensor combines both files into one model and stores it at the given template ID.
5. The device reports success, and the server saves the mapping in the fingerprints collection.

E. Attendance marking algorithm.

1. Faculty start a session for a subject; the server marks the session active for that device.
2. The terminal waits for a finger and captures an image.
3. The sensor searches its library. If no match is found, the device shows "Not recognised" and sounds a long beep.
4. On a match, the ESP32 sends deviceId, templateId, confidence and timestamp to /api/device/attendance.
5. The server verifies the API key, finds the user for the template ID and checks that a session is active.
6. If the student is already marked for that session, the server replies "Already marked".
7. Otherwise it saves the record as Present and returns the student's name.
8. The device shows the name with a green LED and a short beep. The React dashboard refreshes the live list.
9. When the session ends, students with no record are stored as Absent.

F. Offline handling. If Wi-Fi is down, the ESP32 stores events in flash memory using its file system and retries in the background. Stored events keep their original timestamp, which is synchronised from an NTP server at start-up, so late upload does not change the attendance time.

VI. Implementation, Testing and Results

A. Prototype. The terminal was assembled on a perforated board and placed in a 3D-printed enclosure with an opening for the sensor window and the OLED. The back end was deployed on a cloud Node.js host connected to a MongoDB Atlas cluster, and the React.js build was served as a static site.

Fig. 2. Hardware prototype of the fingerprint terminal. [Insert photograph of your device here.]

Fig. 3. Faculty dashboard showing the live attendance of an active session. [Insert screenshot here.]

Fig. 4. Subject-wise attendance report with defaulter list. [Insert screenshot here.]

B. Functional testing. Each module was tested against the cases in Table VII.

Table VII. Functional test cases

No.	Test case	Expected result	Status
1	Enrol a new fingerprint	Template stored and linked to the student	Pass
2	Registered finger during an active	Name shown, record saved as Present	Pass

No.	Test case	Expected result	Status
	session		
3	Unregistered finger	"Not recognised" shown, no record saved	Pass
4	Same student scans twice in one session	"Already marked" shown, no duplicate	Pass
5	Scan when no session is active	Request rejected with a message	Pass
6	Request with an invalid device API key	HTTP 401, event ignored	Pass
7	Wi-Fi disconnected during scans	Events buffered and uploaded on reconnect	Pass
8	Student opens a faculty-only route	HTTP 403 Forbidden	Pass
9	Session closed by faculty	Remaining students stored as Absent	Pass
10	Export subject report	CSV / PDF downloaded with correct data	Pass

C. Performance evaluation. The system was tested with [N] enrolled users over [M] sessions. Table VIII summarises the measured values.

Table VIII. Performance results

Parameter	Measured value
Number of enrolled users	[N]
Total verification attempts	[Total]
Correct acceptances	[__]
False rejections (genuine user not matched on first try)	[__] %
False acceptances (wrong user matched)	[__] %
Average time from finger placement to on-screen result	[__] s
Average API response time (device to server and back)	[__] ms
Time for dashboard to show a new record	[__] s

D. Time saving. If marking takes about 1.5 seconds per student, a class of 60 is recorded in about 90 seconds, and students can scan while the lecture begins. A verbal roll call for the same class typically takes several minutes of teaching time. The records are also available in the dashboard instantly, so no manual data entry is needed.

E. Observations. Recognition failures mostly occurred with dry, wet or dusty fingers and with fingers placed at a steep angle. Enrolling two fingers per student and keeping the sensor glass clean reduced repeat scans.

VII. Discussion, Security and Limitations

A. Discussion. The results confirm that pairing on-sensor fingerprint matching with a MERN web application gives secure and immediate attendance at low cost. Because matching happens inside the R307 module, the ESP32 only handles small JSON messages, which keeps the firmware simple and the network load light. MongoDB's document model fits attendance data well: new fields such as geo-location or temperature can be added without schema migrations, and aggregation pipelines produce subject-wise percentages directly in the database.

B. Security and privacy. Biometric data is personal data, so the design limits where it travels and who can see results.

- Fingerprint images and templates stay inside the sensor. Only a numeric template ID leaves the device, so a database breach does not expose biometric data.
- Each terminal has its own API key, stored as a hash on the server, so an unregistered device cannot post events.
- All traffic between device, server and browser uses HTTPS (TLS).
- User passwords are hashed with bcrypt, and every user request carries a short-lived JWT checked against the user's role [14].
- The unique index on student and session, together with server-side session checks, blocks duplicate and out-of-session entries.
- Enrolment needs the student's informed consent, in line with India's Digital Personal Data Protection Act, 2023 [15].

C. Limitations.

- Optical sensors can be fooled by high-quality fake fingers, because the R307 has no liveness detection.
- Templates are stored per terminal, so a student enrolled on one classroom device must be enrolled again, or the template synced, to use another.
- Recognition is less reliable for users with worn, injured or very dry fingers.
- Contact-based scanning raises hygiene concerns that contactless methods avoid.
- Long network outages delay dashboard updates, though no records are lost.

VIII. Conclusion and Future Scope

This paper presented the design and implementation of an IoT-based Smart Attendance Monitoring System that uses fingerprint authentication and a MERN-stack web application. An ESP32 terminal with an R307 sensor verifies each user on the module and sends only the template ID to a secure Express.js API. MongoDB stores the records, and a React.js dashboard gives role-based, real-time access to attendance and reports. Testing showed that the system removes proxy attendance, records each student in a second or two, prevents duplicate entries and keeps working during short network outages. At about ₹2,000 to ₹2,500 per terminal, it is affordable for every classroom.

Future work will focus on:

1. Capacitive sensors with liveness detection to resist fake fingerprints.
2. Central template management, so one enrolment works on every terminal.
3. Multi-factor options that combine fingerprint with RFID or face recognition.
4. Automatic SMS, e-mail or WhatsApp alerts to parents and students below the attendance threshold.
5. A mobile app and integration with the institution's ERP or learning management system.
6. Machine-learning analytics to predict students at risk of falling below the required attendance.

Acknowledgment

The authors thank the management and the Department of Computer Science & Engineering, Sagar Institute of Science & Technology, Gandhi Nagar, Bhopal, for providing the laboratory facilities and support needed to complete this work.

References

[1] O. T. Arulogun, A. Olatunbosun, O. A. Fakolujo, and O. M. Olaniyi, "RFID-based students attendance management system," *Int. J. Sci. Eng. Res.*, vol. 4, no. 2, Feb. 2013.

[2] N. Kar, M. K. Debbarma, A. Saha, and D. R. Pal, "Study of implementing automated attendance system using face recognition technique," *Int. J. Comput. Commun. Eng.*, vol. 1, no. 2, pp. 100–103, 2012.

[3] A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 1, pp. 4–20, Jan. 2004.

[4] Hangzhou Grow Technology Co., Ltd., *R307 Fingerprint Module User Manual*.

[5] Espressif Systems, *ESP32 Series Datasheet*. [Online]. Available: https://www.espressif.com/en/support/documents/technical-documents

[6] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*, 2nd ed. London, U.K.: Springer, 2009.

[7] S. Kamble, R. Chouhan, K. Memane, V. Ingale, and N. Fatima, "Fingerprint based attendance system by IoT," *Int. Res. J. Eng. Technol. (IRJET)*, vol. 9, no. 5, May 2022.

[8] E. T. Olawole and O. F. Adebayo, "Development of an IoT based fingerprint biometric attendance system," *LAUTECH J. Eng. Technol.*, vol. 19, no. 3, pp. 79–87, 2025.

[9] N. K. Abhirami, A. P. Shafin Sharaf, K. C. Amidha, L. Thomas, and A. M. George, "GateSense: A smart continuous attendance system," *J. Appl. Sci. Eng. Technol. Manag. (JASETM)*, vol. 4, no. 1, pp. 35–41, 2026.

[10] MongoDB, Inc., *MongoDB Documentation*. [Online]. Available: https://www.mongodb.com/docs/

[11] OpenJS Foundation, *Express – Node.js Web Application Framework*. [Online]. Available: https://expressjs.com/

[12] OpenJS Foundation, *Node.js Documentation*. [Online]. Available: https://nodejs.org/docs/

[13] Meta Open Source, *React – The Library for Web and Native User Interfaces*. [Online]. Available: https://react.dev/

[14] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," IETF RFC 7519, May 2015.

[15] Government of India, *The Digital Personal Data Protection Act, 2023* (Act No. 22 of 2023), Ministry of Law and Justice, Aug. 2023.