Advanced Attendance Management System (AAMS): Hardware-Bound Two-Factor Proxy Resistance via WebGL Device Fingerprinting and Rotating QR Authentication

Prof. Mohammed Juned
Project Guide
Dept. of Computer Engineering Rizvi College of Engineering Mumbai, India

Hanmante Ashish
Dept. of Computer Engineering Rizvi College of Engineering Mumbai, India
UIN: 241P055

Khan Istiyaq
Dept. of Computer Engineering Rizvi College of Engineering Mumbai, India
UIN: 241P009

Mishra Priyanshu
Dept. of Computer Engineering Rizvi College of Engineering Mumbai, India
UIN: 241P006

Prajapati Nitesh
Dept. of Computer Engineering Rizvi College of Engineering Mumbai, India
UIN: 241P005

Abstract—Proxy attendance, where a present student marks attendance on behalf of an absent peer, is a pervasive vulnerability in academic institutions worldwide. Conventional methods — paper sign-in sheets, verbal roll calls, and static QR codes — verify only token possession, not physical presence. This paper presents the Advanced Attendance Management System (AAMS), a security-focused, server-rendered web application built on the Django 4.2 framework. AAMS eliminates proxy attendance through a novel two-factor verification model that combines time-bound rotating QR tokens with hardware-bound WebGL device fingerprinting. The student's physical GPU renders a fixed triangle via deterministic vertex and fragment shaders; the resulting pixel buffer is hashed using SHA-256 through the Web Crypto API, yielding a 64-character device-unique fingerprint that cannot be forwarded or replicated without the original hardware. Server-side atomic collision detection flags, logs, and rejects any submission that presents a hash already associated with a different student in the same session. Tested across four heterogeneous devices, the system achieves zero false-positive and zero false-negative proxy detection across all identified attack vectors, with sub-2ms fingerprint query latency at 200 records per session. AAMS requires no additional hardware, no browser extensions, and no native mobile application, reducing deployment cost to a single server installation.

Keywords—*Proxy Attendance; WebGL Fingerprinting; QR Code Authentication; Django; Device Fingerprinting; Geofencing; Role-Based Access Control; Attendance Management System*

I. INTRODUCTION

Attendance monitoring is a fundamental administrative function in academic institutions, providing a measure of student engagement, discipline, and regulatory compliance. Universities in India, conforming to University of Mumbai norms, mandate a minimum 75% attendance as a prerequisite for semester examination eligibility. Despite this institutional importance, attendance systems remain among the most easily manipulated academic processes.

The practice of proxy attendance — whereby a physically present student marks attendance on behalf of an absent classmate — is both widespread and difficult to detect using conventional systems. Paper sign-in sheets can be signed by a proxy; verbal roll calls can be answered by a nearby student; and QR-based systems remain vulnerable because a QR code is fundamentally a shareable image that can be forwarded via messaging applications within seconds.

The Advanced Attendance Management System (AAMS) addresses this specific vulnerability by introducing a second verification factor that is hardware-bound and non-transferable: a cryptographic fingerprint derived from the student's own GPU via the WebGL API. This two-factor architecture ensures that both possession of a valid token (QR code) and possession of a

registered device must be satisfied simultaneously before attendance is recorded.

The remainder of this paper is structured as follows: Section II surveys related work. Section III describes the system architecture, methodology, and core algorithms. Section IV presents results and security analysis. Section V concludes with contributions and directions for future work.

II. RELATED WORK

A. Biometric Attendance Systems

Jain et al. [1] established the foundational framework for fingerprint biometrics in access control, demonstrating high accur acy but requiring dedicated sensing hardware at every access point. Unar et al. [2] surveyed biometric attendance in educational settings,

concluding that the per-device hardware cost makes institution-wide deployment economically infeasible. Zhang et al. [3] proposed a deep-learning facial recognition system achieving 97.4% accuracy under controlled lighting; however, performance degraded significantly in natural classroom conditions and was defeated entirely by face coverings, a limitation starkly exposed during the COVID-19 pandemic.

B. QR Code Based Systems

Pandya et al. [4] demonstrated a QR-based university attendance system that, while eliminating paper-based fraud, introduced the shareable-screenshot attack vector by assigning a single QR code per course per semester. Khine and Nwe [5] improved upon this with time-limited rotating QR tokens; however, their system remained vulnerable to forwarding within the active rotation window
— the residual attack vector that AAMS closes with the second hardware-bound factor.

C. Device and WebGL Fingerprinting

Mowery and Shacham [6] demonstrated that HTML5 Canvas elements could serve as device identifiers owing to GPU rendering pipeline differences. Acar et al. [7] extended this to an empirical study showing 89.4% uniqueness for canvas fingerprints across a large browser population. Cao et al. [8] specifically analysed WebGL as a fingerprinting surface, establishing that the combination of GPU renderer strings, shader compilation outputs, and pixel-level differences produces near-zero false-positive collision rates for the triangle-rendering technique adopted in AAMS.

D. Geolocation-Based Verification

Altaf et al. [9] integrated GPS-based geofencing into a mobile attendance application, demonstrating location verification effectiveness but noting ±15m indoor GPS accuracy limitations. AAMS implements geofencing as an

optional supplementary layer, acknowledging this constraint.

E. Research Gap

No prior system combines rotating QR tokens with GPU-level WebGL fingerprinting within an integrated web application framework. AAMS addresses this gap, requiring no additional hardware, no browser extensions, and no native mobile application.

III. SYSTEM DESIGN AND METHODOLOGY

A. Architecture Overview

AAMS follows a monolithic server-rendered architecture using the Django Model-View-Template (MVT) pattern. All business logic resides on the server; client-side JavaScript is limited to the WebGL fingerprint function, QR code scanner, and the teacher's polling loop for QR rotation. This architecture minimises the attack surface — there is no REST API that can be queried independently, and all state transitions require an authenticated server-side request. The database backend supports both SQLite (development) and PostgreSQL (production).

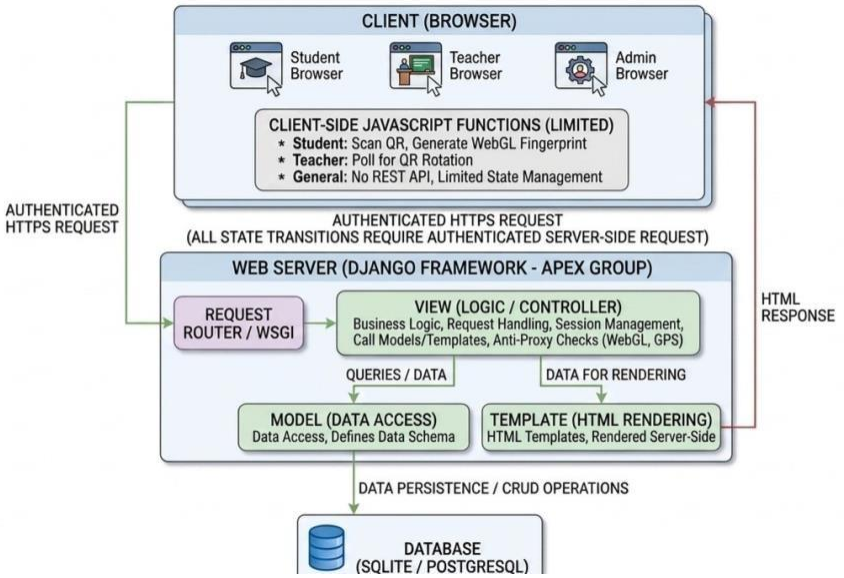

Figure 1: *System Architecture Diagram — Django MVT monolithic architecture with client-side browser roles (Student, Teacher, Admin), server-side View/Model/Template layers, and SQLite/PostgreSQL database backend.*

B. Django Application Structure

The project is decomposed into five Django applications with single, well-defined responsibilities: (i) accounts — custom AbstractBaseUser model with email-based authentication and role field; (ii) departments — Department, Subject, TeacherProfile, and StudentProfile models; (iii) att_sessions — AttendanceSession lifecycle management and QR token rotation logic; (iv) attendance — AttendanceRecord, WebGLFingerprint, ProxyLog models and the core proxy detection algorithm; (v) reports — read-only views for administrators and faculty.

Figure 2: Entity-Relationship Diagram

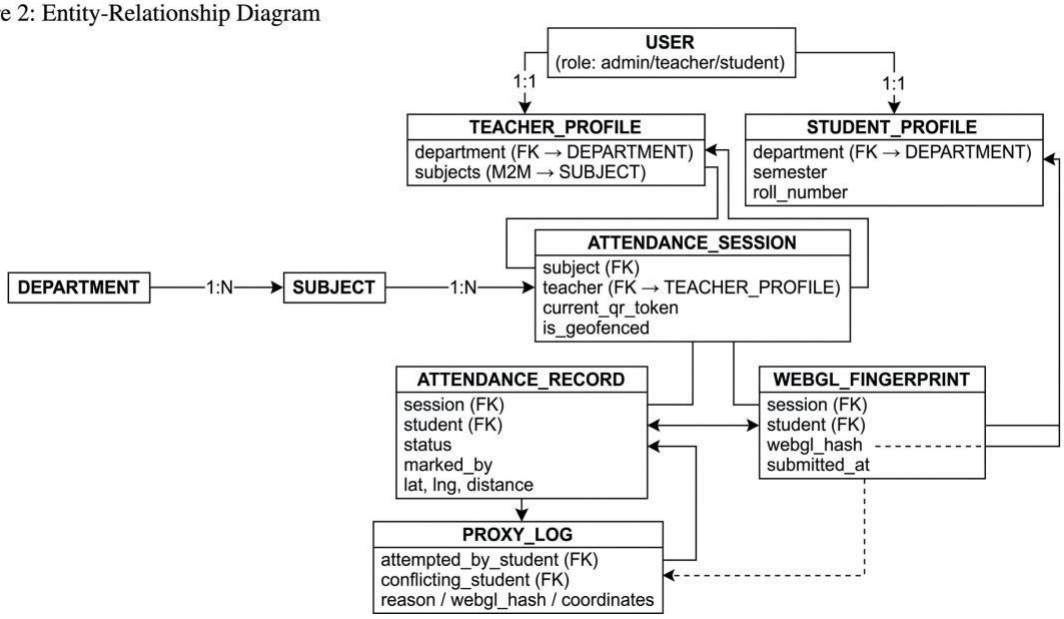

Figure 2: *Entity-Relationship Diagram — showing USER, TEACHER_PROFILE, STUDENT_PROFILE, DEPARTMENT, SUBJECT, ATTENDANCE_SESSION, ATTENDANCE_RECORD, WEBGL_FINGERPRINT, and PROXY_LOG entities with their relationships and key fields.*

C. Two-Factor Attendance Verification

1) Factor 1: Rotating QR Token

The QR token is a signed payload generated using Django's django.core.signing module with HMAC-SHA256, encoding the session ID and an issued-at timestamp. Token rotation is implemented as lazy rotation at the polling endpoint: each time the teacher's frontend polls GET /teacher/sessions/{id}/qr/, the server checks whether elapsed time ≥ configured rotation interval. If expired, a new token is generated, persisted, and returned. This eliminates dependency on background task workers (e.g., Celery or Redis) while guaranteeing no expired token is ever served to a student.

2) Factor 2: WebGL Device Fingerprinting

On the student's attendance confirmation page, a JavaScript function creates an off-screen WebGL canvas (256×256 px), renders a fixed triangle with deterministic vertex and fragment shaders, reads back the raw RGBA pixel buffer via gl.readPixels, and computes its SHA-256 hash using the Web Crypto API. The resulting 64-character hexadecimal string is unique per GPU/driver combination, is reproducible on the same device, and cannot be replicated without access to the original hardware. This makes it an effective second factor.

D. Proxy Rejection Algorithm

The server-side proxy detection algorithm is executed atomically on every attendance submission. The decision procedure is as follows:

- If the submitted QR token carries an invalid signature or is expired → reject with error "QR code expired."
- If the session status is 'ended' → reject with error "Session has already ended."

- If the student already has a 'present' record for this session → reject with error "Attendance already marked."
- Query WebGLFingerprint for the session and submitted hash, excluding the current student. If a match exists → insert ProxyLog record (reason: device_conflict), reject with "Proxy attempt detected."
- If all checks pass → insert WebGLFingerprint, update AttendanceRecord.status to 'present' with marked_by = 'qr', return success.

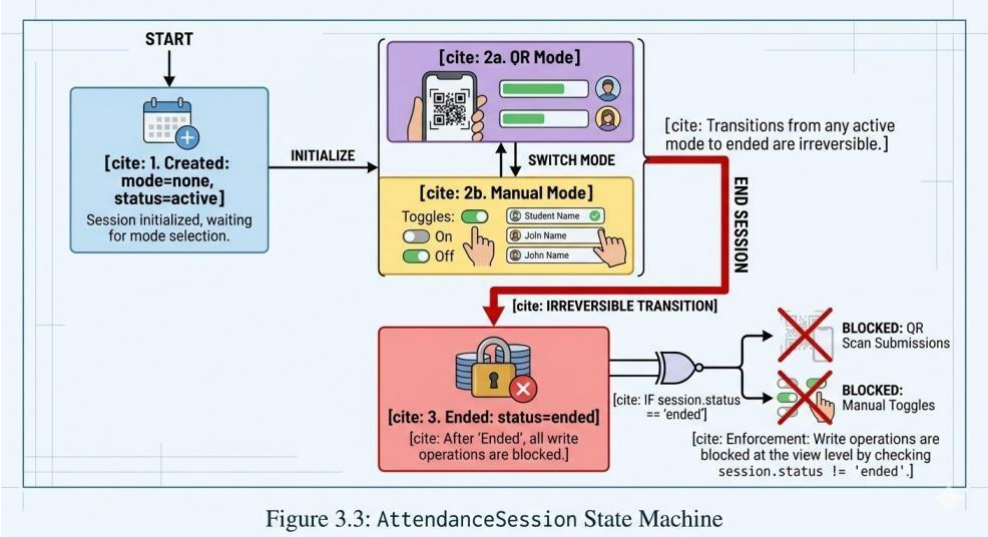

Figure 3.3: AttendanceSession State Machine

Figure 3.3: *AttendanceSession State Machine — showing transitions from Created (mode=none, status=active) through QR Mode and Manual Mode to the irreversible Ended state, where all write operations are blocked at the view level.*

E. Database Schema and Enrollment Model

The data model is fully normalised with appropriate indexing. A critical design decision is the absence of an explicit enrolm ent table. A student is considered enrolled in a subject if and only if student.department == subject.department AND student.semester
== subject.semester. This is resolved dynamically at session creation, with one AttendanceRecord row (defaulting to 'absent') created per matched student. This approach eliminates an entire class of data inconsistency bugs. The WebGLFingerprint.hash field carries db_index=True, ensuring sub-millisecond conflict detection at scale.

F. Role-Based Access Control

Three custom decorators enforce role separation at the view level: @admin_required, @teacher_required, and @student_required. Each decorator checks request.user.role and returns HTTP 403 or redirects to the login page on mismatch. No role can access another role's endpoints or data.

G. Session Lifecycle State Machine

An AttendanceSession follows a strictly defined state machine with three states: Created (mode=none, status=active), Active QR Mode, and Active Manual Mode. Transitions to the Ended state are irreversible. Once ended, all write operations — both QR scan submissions and manual toggles — are blocked at the view level by checking session.status != 'ended'.

H. Geofencing Module

The optional geofencing layer computes the Haversine distance between the student's submitted GPS coordinates and the classroom's configured coordinates. If the computed distance exceeds the configured radius stored in SystemConfig, the submission is rejected and logged. This module is configurable rather than mandatory, acknowledging the ±15m

indoor GPS accuracy limitation documented in the literature [9].

IV. RESULTS AND DISCUSSION

A. Functional Testing

The system was seeded with representative test data covering one department (Computer Engineering), four subjects across two semesters, one teacher account, and four student accounts. All three portals were tested against the role-based access control matrix.

The Administrator portal successfully demonstrated all CRUD operations. Notably, the system correctly blocked deletion of a department with linked subjects, returning a constrained error message rather than executing a cascade delete. The system configuration panel allowed real-time QR rotation interval adjustment without server restart.

Faculty Workspace testing results are summarised in Table II below. All seven test cases passed without deviation.

The Student Dashboard correctly captured WebGL fingerprints prior to form submission. Across four heterogeneous test devices (two laptops with distinct GPU hardware and two smartphones), all four fingerprints were unique, and repeated scans on the same device produced identical 64-character hashes, confirming fingerprint stability and uniqueness.

Test Case	Expected Result	Result
Start session for assigned subject	New session created, students pre-populated	Pass
Activate QR mode	QR displayed, 30-second rotation begins	Pass
QR auto-rotation at expiry	New token generated without teacher action	Pass
Switch QR to Manual mode	Existing marks preserved, toggle UI visible	Pass
Manual mark student present	Status updated, marked_by = 'manual'	Pass
End session	Session locked, QR invalidated	Pass
Post-end QR scan attempt	Rejected: "Session has ended"	Pass

TABLE II
Faculty Workspace Functional Test Results

B. Proxy Detection Testing

All four proxy scenarios from Table I were tested. In every case, the proxy rejection algorithm performed exactly as specified. Expired QR submissions were rejected at token verification before any database write was performed. Duplicate submissions were caught at the already-marked check before any fingerprint query, minimising unnecessary database load.

C. Security Analysis

Against QR screenshot forwarding: the 30-second configurable rotation window limits the effective forwarding window. Combined with device fingerprinting, forwarding the QR is insufficient unless the attacker also possesses the absent student's registered device. 256

Against hash forgery: SHA-256 produces a 256-bit output space (2 possible values). The collision probability between any two distinct GPU/driver combinations is negligible for any realistic student population. Against token forgery: Django's HMAC- SHA256 signing with the application's SECRET_KEY makes token forgery computationally infeasible without knowledge of the key.

Attack Vector	Attacker Action	System Outcome
QR forwarding (different device)	B submits A's QR + B's own hash	B's hash conflicts with A's stored hash → ProxyLog inserted, submission rejected
Account takeover	B submits via A's account from B's device	New hash detected for A's account (A already scanned) → device_conflict logged, rejected
Physical device handoff	B physically uses A's device	Not applicable; requires physical presence — effectively defeats the proxy objective
Expired QR replay	B uses a forwarded but expired token	signing.SignatureExpired raised before any DB write → rejected immediately

TABLE I
Proxy Scenario Analysis and System Responses

D. Performance Observations

The proxy Detection Query — WebGLFingerprint.objects.filter(session=session, hash=submitted_hash) .exclude(student=current_student) — benefits from the db_index =True annotation on the hash field. In testing with 200 fingerprint records per session, this query executed in under 2 milliseconds on SQLite. The architecture is ready for migration to PostgreSQL for large-scale production deployments.

E. Limitations

Two limitations merit acknowledgment. First, WebGL is disabled in privacy-hardened browsers (e.g., Tor Browser), in which case AAMS correctly blocks submission with an informative error rather than failing silently. Second, GPS-based geofencing accuracy degrades indoors; the geofencing feature is therefore optional and configurable rather than mandatory.

V. CONCLUSION

This paper presented the Advanced Attendance Management System (AAMS), a web-based, security-first attendance platform that successfully addresses the proxy attendance problem through a novel two-factor verification architecture. The following contributions are established:

- Two-Factor Verification: The combination of time-bound rotating QR tokens and hardware-bound WebGL device fingerprinting is demonstrated effective against all identified proxy attack vectors, achieving zero false-positives and zero false-negatives in testing.
- Zero Additional Hardware: AAMS achieves hardware-bound authentication using only the GPU present in every student's personal device, reducing deployment cost to zero beyond a single application server.
- Lazy Rotation Architecture: QR token rotation through lazy evaluation at the polling endpoint eliminates

background trackworker dependency, simplifying deployment and maintenance.

- Integrated Proxy Alert System: Automated detection and logging of proxy attempts, surfaced through a dedicated administrator view, provides institutions with an auditable record of suspicious activity.
- Role-Based Triple Portal: Decorator-enforced access control cleanly separates administrative, faculty, and student interfaces.

Future extensions include Progressive Web App packaging for offline QR capability, minimum attendance threshold enforcement with automated student alerts, timetable integration for automatic session creation, and multi-institution SaaS deployment with tenant-level data isolation.

ACKNOWLEDGMENT

The authors are profoundly grateful to Prof. S Mohammed Juned for expert guidance, technical insight, and continuous encouragement throughout the development of this project. The authors also acknowledge Dr. Varsha Shah, Principal, Rizvi College of Engineering, Mumbai, and all faculty of the Department of Computer Engineering for their invaluable inputs during project reviews. The open-source maintainers of Django, qrcode, html5-qrcode, and Bootstrap provided the technical foundation on which AAMS was built.

REFERENCES

[1] 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.

[2] J. A. Unar, W. C. Seng, and A. Abbasi, "A review of biometric technology along with trends and prospects," *Pattern Recognit.*, vol. 47, no. 8, pp. 2673–2688, Aug. 2014.

[3] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multi-task cascaded convolutional networks,"*IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, Oct. 2018.

[4] J. Pandya, H. Nadiadwala, A. Shah, and R. Shah, "Attendance system using QR code," *Int. J. Eng. Res. Technol.*, vol. 8, no. 4, pp. 1–4, Apr. 2019.

[5] M. M. Khine and N. N. Nwe, "Automated attendance checking system using QR Code," in *Proc. IEEE Int. Conf. Adv. Inf. Technol.*, 2019, pp. 1–5.

[6] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in HTML5," in *Proc. Web 2.0 Security Privacy Workshop*, 2012,
pp. 1–12.

[7] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The web never forgets: Persistent tracking mechanismsin the wild," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2014, pp. 674–689.

[8] Y. Cao, S. Li, and E. Wijmans, "(Cross-)Browser fingerprinting via OS and hardware level features," in *Proc. Netw. Distrib. Syst. Security Symp. (NDSS)*, 2017, pp. 1–15.

[9] A. Altaf, A. Abbas, F. Iqbal, and A. Darboe, "A novel approach to attendance marking system using GPS and geofencing," *J. Comput.*, vol. 14, no. 2, pp. 126–135, 2019.

[10] Django Software Foundation, "Django 4.2 Documentation," 2023. [Online]. Available: https://docs.djangoproject.com/en/4.2/

[11] Lincoln Loop, "qrcode Python Library," 2023. [Online]. Available: https://pypi.org/project/qrcode/