R. Vigneshwaran, 2026, 14:6
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

QR-Buddy: HMAC-Based Dynamic Attendance Tokens with Atomic Replay Control and Geofencing

[1]R. Vigneshwaran, [2]N. Mageshwaran, [3]K. Ramkumar, [4]M. N. Vasu, [5]M. Lavanya
Department of Computer Science and Engineering,
Sree Sakthi Engineering College, Coimbatore, Tamil Nadu, India.

Abstract- QR-based attendance systems can reduce manual record keeping, but a scanned code alone does not establish that a submitted credential is current, unused, or associated with an acceptable location. This paper presents QR-Buddy, a prototype attendance-verification system that combines rotating QR credentials, HMAC-SHA-256-based token construction, server-side expiry checks, conditional single-use token consumption, student-registration validation, and a geofencing rule. The documented configuration refreshes the displayed QR value every 30 seconds and sets a maximum token-record validity of 120 seconds. The token construction uses department and session identifiers, a time-window value, and a 128-bit random nonce. The documented software components comprise a React/TypeScript client, a Node.js/Express verification service, a Python/Flask administration portal, and MongoDB Atlas persistence. The location rule calculates great-circle distance from the coordinates submitted by the client using the Haversine formula and accepts a request when the calculated distance is no greater than the configured 1,500 m threshold. Project records report tests involving 600 adversarial payloads distributed across six categories; all 600 were recorded as rejected in that selected test set. A separate synthetic API benchmark is reported as 500 requests with 10 concurrent workers, an average latency of 1.85 ms, a median latency of 2 ms, a 99th-percentile latency of 13 ms, a maximum latency of 26 ms, and peak synthetic throughput of 4,545.45 requests per second. These values are reported project measurements: the underlying test scripts, raw logs, hardware details, software versions, database tier, and workload specification were not available for independent reproduction during manuscript preparation. Consequently, the reported rejection outcome does not establish resistance to all attack classes, and the API benchmark must not be interpreted as end-to-end classroom capacity. In particular, a Haversine calculation does not authenticate the source of client-reported coordinates or independently prevent GPS spoofing or live QR relaying. The paper describes the architecture and verification logic, reviews closely related attendance-security approaches, reports the available measurements with their evidence limitations, and identifies reproducibility, concurrency, failure-handling, and location-manipulation tests required for a stronger evaluation.

Keywords- Attendance security; dynamic QR code; HMAC-SHA-256; replay prevention; geofencing

I. INTRODUCTION

Attendance systems are used to record presence in educational and organizational settings. Paper registers require manual handling and may permit proxy marking. Digital QR-based systems reduce friction, yet static or long-lived QR values can be copied and forwarded. Location checks add context, but a distance calculation over client-reported coordinates does not authenticate those coordinates. A defensible design needs distinct controls for credential integrity, freshness, single-use consumption, roster membership, and geographic policy.

QR-Buddy is a prototype combining a rotating QR display with server-side verification. Its documented design uses HMAC-SHA-256 over contextual values, a 128-bit random nonce, token-expiry records, a conditional single-use state transition, roster validation, and Haversine-distance checking. The software is described as a React-based client, a Node.js/Express verification API, a Python/Flask administration portal, and MongoDB Atlas persistence.

© 2026 R. Vigneshwaran. his is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly credited.

Prior work already combines dynamic QR codes, geofencing, and device identifiers for attendance fraud mitigation (Nwabuwe et al., 2023). AttendEase reports time-limited dynamic tokens, duplicate-registration checks, device binding, and Haversine geofencing, evaluated with 48 students across five courses (Gaftandzhieva & Lyavova, 2026). A recent IEEE conference paper describes a reusable library with dynamic QR and geolocation validation, including fake-location tests (Oktavian et al., 2025). Thus, QR rotation plus geofencing is not a defensible novelty claim by itself.

This paper presents QR-Buddy as an implementation and bounded evaluation report. Its intended contribution is a documented combination of HMAC-based contextual token construction and a database-backed atomic single-use transition in a split verification/administration architecture, together with explicit limits on the current evidence. The questions are: (RQ1) how are freshness and single-use checks represented? (RQ2) what do the recorded tests measure? (RQ3) what further evidence is needed before generalizing results? The answers are limited to the available project record.

II. RELATED WORK

Dynamic QR attendance and geofencing. Nwabuwe et al. (2023) describe a fraud-mitigation system combining dynamic QR codes, venue geofencing, and IMEI checks. AttendEase combines time-limited tokens, duplicate-registration checks, device binding, and Haversine geofencing, with an initial evaluation involving 48 students across five courses (Gaftandzhieva & Lyavova, 2026). These works overlap with QR-Buddy's application domain. The paper must therefore be differentiated by implementation detail and evidence, not by the broad idea of using QR and location together.

Reusable location and QR security. A 2025 IEEE conference paper describes SecureQR and GeoValidator modules for reusable QR and geolocation fraud mitigation; its reported evaluation includes tests against ten fake-GPS applications (Oktavian et al., 2025). That work addresses location manipulation more directly than QR-Buddy's current evidence, which documents only a distance check and does not demonstrate GPS-spoofing detection. This is a limitation, not an area where QR-Buddy can claim superiority.

Cryptographic token integrity. HMAC is a keyed message-authentication construction for integrity and authenticity when parties share a secret (Krawczyk et al., 1997). A nonce can improve token uniqueness, but it is not a replay defense on its own. Replay prevention depends on server-side expiry and single-use state. The final implementation specification should define canonical serialization, field encoding, key storage and rotation, and token representation.

Atomic database state. MongoDB documents atomicity for single-document writes and recommends including the expected current value in an update filter to avoid conflicting concurrent updates (MongoDB, n.d.). A conditional update can support one-time token consumption if unused and unexpired predicates are part of the same atomic write. This does not automatically make the entire workflow atomic across token consumption and attendance insertion; failure recovery and transaction boundaries require testing.

Geographic checks and QR symbology. The Haversine calculation estimates great-circle distance from latitude/longitude pairs (Sinnott, 1984). It enforces a rule over supplied coordinates but does not authenticate their source. ISO/IEC 18004 specifies QR Code symbology (ISO/IEC, 2024), but encoding compliance does not secure the application protocol inside a QR value.

Research gap. The reviewed literature already covers dynamic QR attendance, geofencing, device binding, and fake-location testing. The narrow contribution supportable by current QR-Buddy evidence is a documented implementation pattern for HMAC-based contextual tokens and atomic single-use token state, with a bounded report of existing tests. Stronger novelty claims require a systematic literature review and reproducible comparisons.

III. METHODOLOGY

This manuscript reports an existing prototype and measurements recorded in project documentation. No new experiment was performed for this revision, and reported benchmark values have not been independently reproduced. The available source materials include the previously submitted paper and journal template, but not the source repository, raw benchmark logs, or test scripts. Missing details are therefore identified as validation requirements rather than inferred.

1. System Overview

The documented architecture has a client, verification API, administration portal, and shared data layer. The client scans or requests the active QR value and submits it with a registration identifier and reported location. Node.js/Express handles token generation and attendance verification; Python/Flask supports administration and auditing; MongoDB Atlas stores token, roster, and audit data. Figure 1 summarizes documented logical components, not a source-code-derived deployment diagram.

The documented token expression is $T = \text{HMAC-SHA-256}(K, \text{DeptID} \parallel \text{SessionID} \parallel \tau \parallel \eta)$, where K is the secret key, DeptID is the department identifier, SessionID identifies the session, τ is a time-window value, η is a 128-bit random nonce, and $\parallel$ denotes concatenation. The display refresh interval is 30 seconds and maximum configured token-record validity is 120 seconds. These parameters are distinct. A reproducible specification must define serialization, HMAC-output encoding, server lookup/validation, and key storage/rotation.

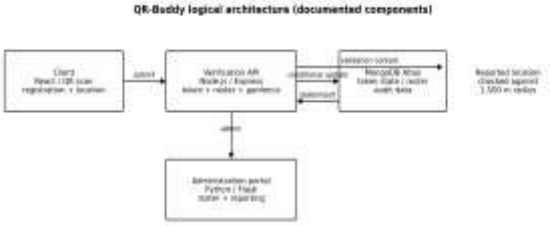


Figure 1. Logical architecture based on documented components.

2. Verification Workflow and Replay Control

The reported workflow begins with token generation for an active department/session context. The display refreshes every 30 seconds. The client submits the scanned token and attendance attributes. The server is described as checking freshness, single-use state, student registration, and geographic distance before recording attendance. Figure 2 presents the logical sequence; exact validation order and failure handling must be confirmed from source code.

The replay-control operation is a conditional update that matches a token only when used=false and expires_at is later than server time, then sets used=true and used_at=now. Attendance must not proceed if the update did not match a document. This is implementation pseudocode, not a mathematical equation. The implementation must define behavior if token consumption succeeds but later validation, network activity, or attendance persistence fails. Concurrent tests should verify that at most one request consumes a token and no duplicate attendance record is created.



Figure 2. Documented verification sequence; order requires source-code confirmation.

3. Geofencing

Let φ_{user} and λ_{user} denote reported user coordinates, and φ_{campus} and λ_{campus} the configured campus center. Convert all coordinates from degrees to radians. Define $\Delta\varphi = \varphi_{campus} - \varphi_{user}$ and $\Delta\lambda = \lambda_{campus} - \lambda_{user}$. Then $a = \sin^2(\Delta\varphi/2) + \cos(\varphi_{user})\cos(\varphi_{campus})\sin^2(\Delta\lambda/2)$, $c = 2\,\text{atan2}(\sqrt{a}, \sqrt{(1-a)})$, and $d = R \cdot c$, where $R = 6{,}371{,}000$ m. The reported rule accepts when $d \le 1{,}500$ m. The available record does not establish that the radius was calibrated against campus geometry or location-error measurements.

The geofence check is not anti-spoofing by itself. If an attacker manipulates coordinates supplied by the client, the server can calculate an in-range distance

from false coordinates. The documented test categories mention out-of-bounds locations but do not establish testing against mock-location applications, rooted devices, location-provider manipulation, or live QR relaying. These threats must not be described as solved.

4. Implementation and Reproducibility

The project documentation identifies React/TypeScript, Node.js/Express, Python/Flask, and MongoDB Atlas. It names POST /api/qr/generate and POST /api/submit as API paths. The final paper should confirm paths, package versions, schema, indexes, validation order, timeout settings, and transaction behavior against the repository. A reproducibility package should record hardware, OS, runtime/dependency versions, database tier and region, network topology, benchmark tool, request mix, warm-up policy, repetitions, and raw logs. These details were absent from the submitted PDF and must be recovered or measured.

IV. RESULTS AND DISCUSSION

The project record reports a security test of 600 adversarial payloads across six categories, with 100 payloads per category. It reports all 600 rejected and no successful bypass observed in the selected test set. Table 1 preserves that reported result without extending it beyond the finite test cases.

Table 1. Project-reported security-test outcomes

Attack category	Tests	Rejected	Bypassed
Expired replay	100	100	0
Used-token replay	100	100	0
Tampered token	100	100	0
Duplicate submission	100	100	0
Outside geofence	100	100	0
Invalid registration	100	100	0
Total	600	600	0

1. Reported Security Results

The reported categories are expired-token replay, used-token resubmission, forged or tampered token, duplicate attendance submission, out-of-bounds geofence, and invalid student registration. Each category is reported as 100 tests, 100 rejected, and zero bypasses. This is a 100% observed rejection rate in the selected test set, not evidence of zero false acceptance across all possible attacks or deployments. The source material does not include scripts, raw responses, reset procedures, or formal definitions for each payload class; independent reproduction remains necessary.

2. Reported API Performance

The synthetic benchmark is reported as 500 requests and 10 concurrent workers. It reports 1.85 ms average latency, 2.00 ms p50, 2.00 ms p90, 3.00 ms p95, 13.00 ms p99, 26.00 ms maximum latency, and peak synthetic throughput of 4,545.45 requests/s. These are project-reported measurements pending reproduction. The available record does not specify hardware, operating system, runtime versions, database tier/region, request distribution, warm-up, repetitions, measurement tool, errors/timeouts, or the throughput calculation.

The throughput figure is an API benchmark, not end-to-end attendance capacity. It does not include a verified measure for camera scanning, GPS acquisition, mobile-network delay, user interaction, or the complete attendance workflow. No controlled baseline was supplied, so the evidence does not support a claim that QR-Buddy is faster or more secure than other systems.

3. Comparative Positioning

Table 2 compares features explicitly described in the cited publications. It is a literature-based qualitative comparison, not a controlled benchmark. It highlights that dynamic QR and geofencing are established approaches, while HMAC construction and atomic token consumption are the implementation details relevant to QR-Buddy's positioning. Current evidence does not establish stronger protection or better performance than the compared systems.

Table 3 lists evaluation claims and evidence status. It separates observations recorded in project documentation from claims requiring further experiments. This distinction is necessary for a credible submission.

Table 2. Qualitative comparison with related work

Work	Features described	Evaluation / implication
Nwabuwe et al. (2023)	Dynamic QR; geofence; IMEI	Fraud-mitigation system; overlap means QR + location is not novel.
Gaftandzhieva & Lyavova (2026)	Time-limited token; duplicate registration; device binding; geofence	48 students across 5 courses; close application overlap.
Oktavian et al. (2025)	Dynamic QR; geolocation validation; fake-GPS tests	Location manipulation tested more directly than in QR-Buddy.
QR-Buddy	HMAC context; nonce; atomic single-use state	600 payloads and synthetic API benchmark reported; scripts/logs unavailable.

Table 3. Evidence status and required validation

Claim	Available evidence	Required next step
Security rejection	600/600 rejected reported	Provide scripts, payloads, raw logs, reset protocol.
Latency / throughput	Reported mean 1.85 ms; p99 13 ms; peak 4,545.45 req/s	Document environment, workload, repetitions, formula.
Concurrent replay	Conditional atomic update described	Run race test; verify one attendance record.
GPS spoofing	Not demonstrated	Run dedicated mock-location/manipulation tests.
Performance comparison	No controlled baseline supplied	Run identical workload against defined baseline.

V. CONCLUSION

QR-Buddy is a prototype attendance-verification system combining dynamic QR credentials, HMAC-SHA-256-based contextual token construction, a conditional single-use token state transition, roster validation, Haversine geofencing, and separate verification and administration services. Project records report rejection of 600 selected adversarial payloads and low latency in a synthetic API benchmark. These observations are bounded and do not prove universal security, GPS anti-spoofing, or production-scale attendance capacity.

The contribution should be understood as a documented implementation and preliminary evaluation, not the invention of QR-plus-geofencing attendance. The next steps are to recover source code and logs, reproduce tests, test concurrent replay and failure cases, evaluate location manipulation separately, and compare against a defined baseline.

Acknowledgements and AI-use disclosure

The authors acknowledge the institutional resources and technical support used during development and evaluation. Any funding or specific assistance should be confirmed by the authors. Generative AI assistance was used for language editing and manuscript preparation; the authors must verify the technical content, references, and final text and take responsibility for the submission.

REFERENCES

1. Nwabuwe, A., Sanghera, B., Alade, T., & Olajide, F. (2023). Fraud mitigation in attendance monitoring systems using dynamic QR code, geofencing and IMEI technologies. International Journal of Advanced Computer Science and Applications, 14(4), 938–945. https://doi.org/10.14569/IJACSA.2023.01404104
2. Gaftandzhieva, S., & Lyavova, G. (2026). AttendEase: Fraud-resistant attendance tracking in higher education using dynamic tokens and geofencing. International Journal of Advanced Computer Science and Applications, 17(9). https://doi.org/10.14569/IJACSA.2026.0170923

3. Oktavian, A., Gunawan, W., & Kesuma, M. E.-K. (2025). A reusable library for secure attendance systems: Fraud mitigation based on QR code and geolocation. In Proceedings of the 2025 IEEE International Conference on Data and Software Engineering (ICoDSE) (pp. 37–41). IEEE. https://doi.org/10.1109/ICoDSE68111.2025.11351816
4. Krawczyk, H., Bellare, M., & Canetti, R. (1997). HMAC: Keyed-hashing for message authentication (RFC 2104). RFC Editor. https://www.rfc-editor.org/rfc/rfc2104
5. MongoDB. (n.d.). Atomicity and transactions. MongoDB Manual. https://www.mongodb.com/docs/manual/core/write-operations-atomicity/
6. Sinnott, R. W. (1984). Virtues of the Haversine. Sky & Telescope, 68(2), 159.
7. International Organization for Standardization & International Electrotechnical Commission. (2024). Information technology—Automatic identification and data capture techniques—QR code bar code symbology specification (ISO/IEC 18004:2024). https://www.iso.org/standard/83389.html