Geo-Attendance: A Multi-Signal Attendance Verification System Using Geofencing, Facial Recognition, and Anti-Spoofing

Abhimanyu Tiwari[1] Rohan Patil[2] Shravani Patil[3] Jay Suryavanshi[4]
[1,2,3,4]Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning)
[1,2,3,4]Vishwaniketan's Institute of Management Entrepreneurship and Engineering Technology (VIMEET), Raigad, India

Abstract — Traditional attendance systems that rely on a single verification method, like a fingerprint scan, an RFID card, or a GPS check-in, are vulnerable to fraud. This paper introduces Geo-Attendance, a mobile attendance system that reduces fraud by combining five independent data signals at once: GPS location, facial recognition, movement patterns, device activity, and location consistency. Instead of relying on any single source, the system calculates a Presence Score (0–100) to show how confident it is that an individual is really at their assigned workplace. The system has three components: a React Native mobile application that implements a five-layer location validation process, an Express.js backend server that hosts fraud detection services and the scoring engine, and a React admin dashboard for real-time monitoring. Evaluation results show that a legitimate user gets a score of about 92/100, while a GPS spoofing attack only scores 38/100, which is well below the 80-point threshold for being classified as "Present."

Keywords: Attendance System, Geofencing, GPS Spoofing, Facial Recognition, Presence Scoring, Mobile Application, Anti- Spoofing, React Native

I. INTRODUCTION

A. The Problem

Attendance fraud is a major drain on organizational resources. Manikam [1] highlights that buddy punching and similar practices continue to cost organisations significantly, driving the need for stronger biometric verification methods. Traditional systems that use fingerprints or RFID cards are still at risk -fingerprint molds can recreate biometrics, and cards can easily be handed to someone else.

Smartphones offer an attractive alternative. They have GPS, cameras, accelerometers, and constant connectivity. However, using GPS for attendance has its own problems: GPS spoofing. On Android devices, free apps let users send a fake location to all other apps, which undermines any GPS check-in system that relies on a single signal.

B. Our Approach

Rather than relying on just one data source, Geo-Attendance combines five independent signals and treats each as a piece of evidence. Even if an attacker manages to fake one or two signals, convincingly faking all five at the same time poses a much greater challenge.

C. Research Goals

This paper aims to: (1) build a client-side location pipeline that identifies and discards fraudulent GPS data before it reaches the server; (2) design a server-side scoring engine aggregating five signals into a single, interpretable score; (3) implement server-side spoofing checks independent of the phone's own mock flag, which can be disabled on rooted devices; (4) provide administrators with a real-time dashboard for monitoring attendance and reviewing suspicious sessions; and (5) evaluate how accurately the system classifies legitimate users versus attackers, and how well it holds up against the spoofing techniques.

D. Main Contributions

The main contributions of this work are:

- A five-layer location event pipeline that processes only validated GPS data.
- A five-signal weighted Presence Score (0–100) that is both interpretable and auditable.
- A GPS spoofing detection service employing four independent server-side methods.
- A debounced geofence state machine that suppresses false events caused by GPS boundary jitter.
- A complete, functional three-tier system with documented APIs, database schema, and user flows.

II. RELATED WORK

A. Mobile Attendance Systems

Early mobile attendance systems recorded GPS coordinates at login and validated them against a fixed radius [2]. While adequate for honest users, these are entirely defeated by GPS spoofing applications. Subsequent systems incorporated QR code scanning on a display at the check-in location [3]. While harder to spoof remotely, such systems require fixed hardware at every site and remain vulnerable to photograph relay attacks. NFC-based systems establish physical presence at the moment of tap [4], but provide no information about whether the employee remained at the location thereafter. Geo-Attendance differs from all three through continuous verification throughout the entire session, not only at check-in.

B. Geofencing Methods

Two primary geofence geometries exist: circular, defined by a centre and radius using the Haversine formula [5], and polygonal, evaluated using the ray-casting algorithm based on the Jordan curve theorem [6]. GPS accuracy on a mobile device is typically 3–15 metres [7], causing apparent boundary flickering. Geo-Attendance addresses this with a 30-second debounce mechanism requiring a state transition to persist before it is recorded.

C. GPS Spoofing and Defences

Jansen and Hutchinson [8] identify three broad strategies for detecting GPS spoofing on mobile devices: cross-checking GPS readings against inertial sensor data, comparing the reported GPS location with the device's IP-derived location, and looking for suspicious patterns in the coordinate data itself. Geo-Attendance draws on all three. Oligeri et al. [9]

further show that GPS spoofing can be reliably detected by cross-referencing GPS data against independent location sources, demonstrating that fabricated coordinates exhibit measurable inconsistencies. We apply this principle on our server side, using IP-derived location and coordinate pattern analysis as independent cross-checks.

D. Facial Recognition

Geo-Attendance uses the face-api.js library [10], which executes an SSD MobileNetV1 face detector and a ResNet-34 network generating a 128-dimensional face embedding in under 100 ms, entirely on-device with no image transmission to the server.

E. Multi-Signal Scoring

The idea of combining multiple data sources into a single confidence score is not new -it has been explored in areas like smart building occupancy estimation [11], where no single sensor tells the full story. Diethe et al. [12] demonstrate a similar principle in healthcare, using Bayesian models to fuse heterogeneous sensor modalities into a unified prediction. We take the same core idea and apply it to workplace attendance, combining five independent signals available exclusively on a single smartphone into one weighted Presence Score.

F. Gap Analysis

Table I summarises the gaps identified in prior work and how Geo-Attendance addresses each one.

Limitation in Prior Work	Geo-Attendance's Approach
Only checks presence at the moment of check-in	Monitors continuously throughout the entire session
Trusts raw GPS data without any validation	Runs every reading through a five-layer client-side pipeline
Depends on the phone's own mock location flag	Uses four server-side checks that work independently of the device
Limited to circular geofence boundaries	Supports both circular and polygon-shaped geofences
Single signal, no confidence scoring	Combines five signals into one weighted Presence Score
No visibility for administrators	Live dashboard with real-time monitoring and alerts

Table I. Gaps in Existing Systems and How Geo-Attendance Fixes Them

III. SYSTEM ARCHITECTURE

A. Overview

Geo-Attendance is structured as a three-tier system comprising a mobile client, a backend server, and a persistent data store. A separate admin dashboard interfaces with the same backend.

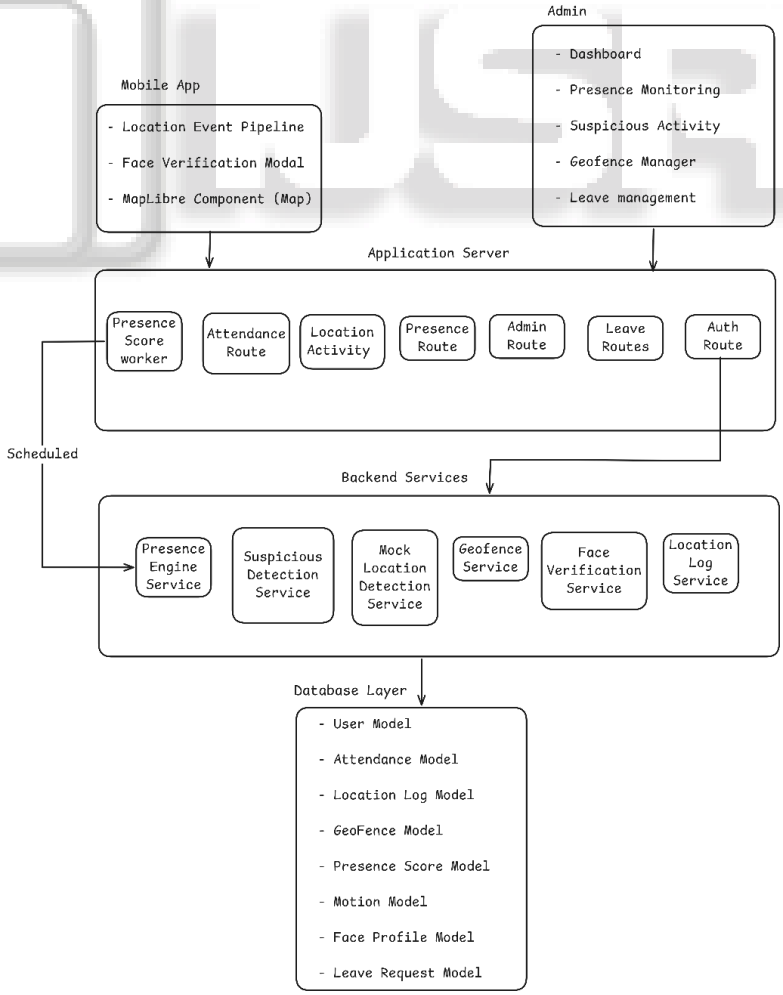

Fig. 1: High-level system architecture of Geo-Attendance

B. Mobile Client

The app is built with Expo (React Native 0.81), targeting Android as the primary platform. On launch, the app requires biometric authentication via expo-local-authentication before rendering any attendance features. The main screen presents a real-time map using MapLibre with MapTiler vector tiles, overlaid with the office geofence geometry.

C. Backend Server

The backend is a Node.js Express application connected to MongoDB Atlas via Mongoose. A background worker (presenceScoreWorker) runs on a schedule, retrieving all active sessions and recalculating presence scores continuously. Table II lists the primary API route groups.

Route Prefix	Function
/api/v1/user	Authentication and registration (JWT)
/api/v1/attendance	Session management, location and motion logging
/api/v1/presence	Current and historical presence scores
/api/v1/admin	User, geofence, and attendance management
/api/v1/admin/presence	Live monitoring and suspicious session review
/api/v1/leaves	Leave submission and approval workflow

Table II: Backend API Routes

D. Technology Stack

Table III summarises the technology choices and their rationale.

Component	Technology	Reason to Use
Mobile App	Expo / React Native 0.81	Cross-platform and sensor access
Maps	MapLibre + MapTiler	Open-source – no billing
Face Detection	face-api.js	On-device – no photos sent
Backend	Express.js	Lightweight
Database	MongoDB + Mongoose	Flexible and geospatial support
Auth	JWT	Stateless – mobile/web
Admin UI	Vite + React + Tailwind	Fast-build, component-based

Table III: Technology Stack

IV. METHODOLOGY

A. Two Core Frameworks

The system is built around two complementary frameworks: a Client-Side Location Pipeline processing raw GPS data on the device, and a Server-Side Presence Engine aggregating five signals into a score. These two operate independently, providing redundancy.

B. The Client-Side Location Pipeline

The pipeline responds to OS-pushed location events without polling. The OS delivers a new event when 30 seconds have elapsed or the device has moved at least 15 metres. Events pass through five sequential layers as described below.

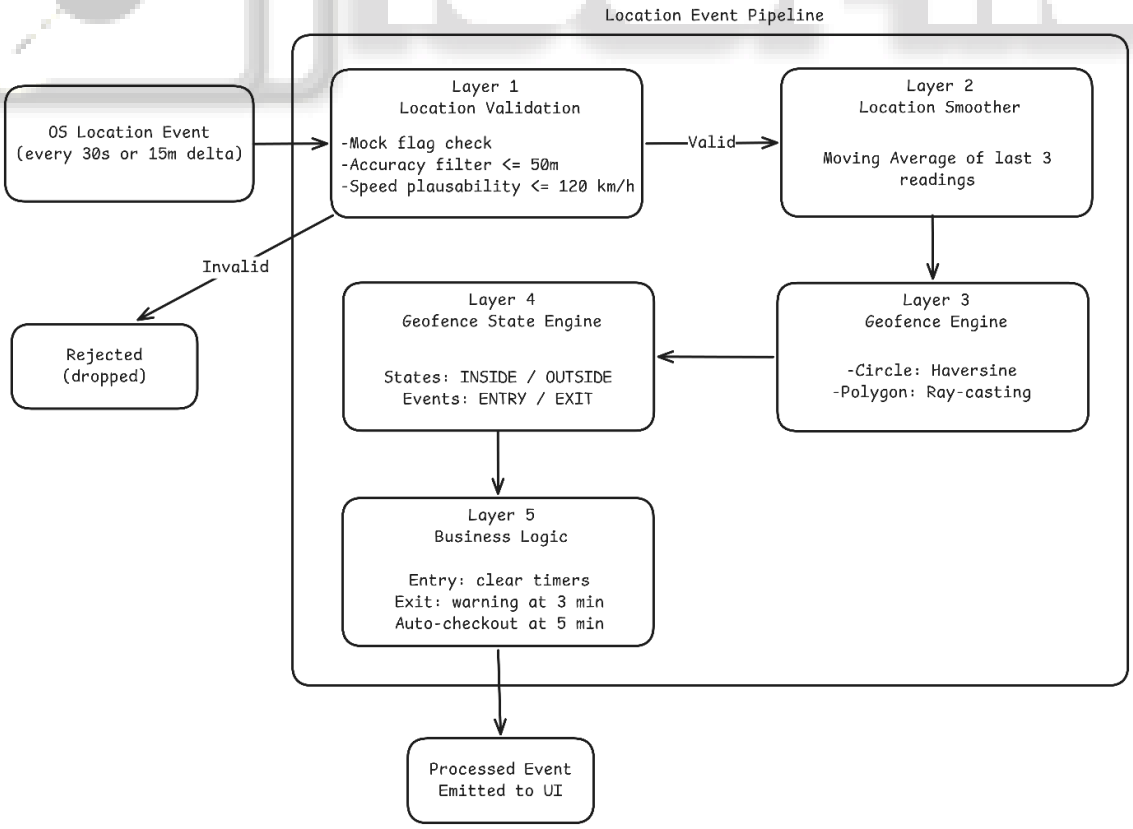

Fig. 2: Five-layer client-side location event pipeline

1) Layer 1 – Location Validator:

Three checks are applied to every incoming event: (i) mock flag rejection – if Android marks the reading as mocked, it is immediately discarded; (ii) accuracy filtering – readings with reported accuracy worse than 50 m are dropped; (iii) speed plausibility – implied speed exceeding 120 km/h is rejected as physically implausible or indicative of a spoofing jump.

2) Layer 2 – Location Smoother:

Even when a phone is completely stationary, GPS coordinates never stay perfectly still - they drift slightly due to atmospheric interference and shifting satellite geometry. If left uncorrected, this jitter causes the map marker to wobble and can trigger unnecessary geofence boundary events. To address this, the smoother takes the average of the last three accepted readings:

$$\hat{\varphi}_t = \frac{(\varphi_t + \varphi_{t-1} + \varphi_{t-2})}{3}, \quad \hat{\lambda}_t = \frac{(\lambda_t + \lambda_{t-1} + \lambda_{t-2})}{3}$$

where φ and λ denote latitude and longitude respectively. In practice, this brought peak positional jitter down from 8.2 m to just 2.4 m during stationary testing- a significant improvement that makes the system noticeably more stable near geofence boundaries.

3) Layer 3 – Geofence Engine:

Two geofence geometries are supported. Circular geofences apply the Haversine formula:

$$a = sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot sin^2\left(\frac{\Delta\lambda}{2}\right),$$
$$d = 2R \cdot \tan^{-1}(\sqrt{a}, \sqrt{1-a})$$

where R = 6,371,000 m is the Earth's mean radius. For more complex building layouts, polygon geofences use a ray-casting algorithm - a horizontal ray is projected outward from the user's position, and if it crosses an odd number of the polygon's edges, the user is considered to be inside. This gives administrators the flexibility to draw a boundary that actually matches the shape of their premises.

4) Layer 4 – Geofence State Machine:

Just because the GPS says a user has crossed the boundary does not mean the system acts on it straight away. In real-world conditions, a person standing near the edge of a geofence will appear to flicker in and out due to normal GPS drift. Without any protection against this, the system would fire off spurious check-ins and check-outs every few seconds. The state machine solves this by requiring the inside/outside classification to remain consistent for a full 30 seconds before it commits to an ENTRY or EXIT event. If the signal flips back before the timer runs out, the pending transition is simply cancelled, as if it never happened.

5) Layer 5 – Business Logic:

A confirmed ENTRY cancels any active departure timer and the session continues normally. A confirmed EXIT starts a 3-minute grace period with a push notification warning. If the user has not returned after 5 minutes, the session is automatically closed with reason code AUTO_CHECKOUT_GEOFENCE.

C. The Server-Side Presence Scoring Engine

The backend engine periodically evaluates how confident the system is that an employee is genuinely present at their designated location. It does this by combining five independent signals into a single composite score:

$$S = 0.40 \cdot g + 0.30 \cdot f + 0.10 \cdot l + 0.10 \cdot d + 0.10 \cdot m$$

where g = geofence, f = face identity, l = location consistency, d = device activity, and m = motion pattern. All component scores lie in [0, 100], so $S \in [0, 100]$. Table IV lists the signal weights and their justification. Weights were assigned based on how directly each signal relates to physical presence.

Signal	Weight	What It Measures
Geofence Status	40%	Device inside office boundary?
Face Identity	30%	Registered employee at the phone?
Location Consistency	10%	Location history appears authentic?
Device Activity	10%	Device actively in use?
Motion Pattern	10%	Physical movement realistic?

Table IV: Presence Score Signal Weights

After computing S, the engine assigns a confidence level (Table V) and raises anomaly flags for individual signals below defined thresholds. Sessions accumulating three or more flags are escalated for administrator review.

Score Range	Classification
80 – 100	Present
50 – 79	Uncertain
0 – 49	Absent

Table V: Confidence Level Classification

V. IMPLEMENTATION

A. Mobile Application

When the app launches, the user is immediately presented with a biometric prompt via expo-local-authentication before any attendance features are accessible. When check-in is initiated, a modal opens the front-facing camera and runs face detection entirely on-device using face-api.js. After a 2-second auto-focus interval, a frame is captured and a 128-dimensional embedding is generated by a ResNet-34 network; only this embedding, never the photograph, is transmitted to the server. The entire inference completes in under 100 ms. Throughout the session, accelerometer and gyroscope readings are sampled every few seconds via expo-sensors and periodically uploaded to the backend for motion analysis.

B. Backend

When a user checks in, the server receives the GPS coordinates and face embedding via POST /api/v1/attendance/start. Before creating a session, it runs the GPS data through the four-method spoofing detector, independently verifies geofence membership, and compares the face embedding against the stored profile. If everything checks out, an attendance record is created with an initial status of tentative. Both circular and polygon geofence types are stored in the same MongoDB collection using a discriminated schema, and a background worker continuously recalculates presence scores for all active sessions on a scheduled cadence.

C. Admin Dashboard

The admin dashboard is an independent Vite + React application with separate authentication credentials. Key

views include: a summary dashboard; a live presence panel showing all five signal scores per active employee; a suspicious activity log with reason codes; an interactive geofence editor supporting drag-to-draw for both geometry types; and a leave management workflow.

VI. Security and Anti-Spoofing

A. Threat Model

Attackers are categorised into three tiers. Level 1 uses a basic GPS spoofing app whose mock flag is visible to the OS. Level 2 suppresses the mock flag and may additionally use a VPN. Level 3 operates a rooted device with full sensor control. The system is designed to fully block Level 1, significantly frustrate Level 2, and ensure that no attacker at any level can push their score above 80 without simultaneously fooling five independent signals at once.

B. Client-Side Defences

The first line of defence is the mock flag check in Layer 1 of the client pipeline. Any GPS reading that Android has marked as mocked is thrown out immediately, which means Level 1 attackers cannot produce a single valid location event -their spoofed coordinates never even make it into the system. For attackers who manage to hide the mock flag, the speed plausibility check steps in -switching a spoofing app from the attacker's real location to the office produces an instant position jump that implies travelling thousands of kilometres per hour, which is caught and discarded. Finally, even if a fake coordinate briefly lands inside the geofence, the 30-second debounce timer means it cannot trigger a check-in unless the spoofed signal is held consistently for the full duration.

C. Server-Side Defences

On the server side, four independent checks are applied to every incoming location update:

1) Mock flag re-check: even though the client already filters mocked readings, the server independently validates the isMockLocation flag from the request body. If it is positive, the update is blocked immediately.
2) GPS accuracy analysis: real GPS on a phone typically reports an accuracy of 3–15 m. If the reported accuracy is consistently below 5 m, it is flagged as suspiciously perfect.
3) IP address cross-check: using geoip-lite, the server estimates the user's approximate location from their IP address. If this differs from the reported GPS location by more than 100 km, an IP_LOCATION_MISMATCH flag is raised -catching anyone spoofing from a different city or country entirely.
4) Coordinate pattern analysis: authentic GPS coordinates always carry six or more decimal places and always drift slightly between readings. Coordinates with fewer than four decimal places, or three consecutive identical readings, are flagged as anomalous.

Table VI shows the actions taken based on the aggregate suspicion score.

Score	Action
< 50	Accepted normally
50 – 99	Flagged for admin review; session proceeds
≥ 100	Rejected outright; session start blocked

Table VI: Suspicion Score Actions

D. Face Verification Against Proxy Attendance

Face verification at check-in directly prevents proxy attendance. Currently, verification happens at the point of check-in, and we are actively working on a continuous re-verification mechanism that will periodically confirm the user's identity throughout the session, catching any substitution that happens after the initial check-in.

VII. Results and Evaluation

A. Location Pipeline Testing

Tests were conducted on a Realme 6 (Android 11) in a real-world office environment with a 100 m circular geofence. All 50 test events correctly classified the user as INSIDE when within 80 m of the geofence centre. No false INSIDE readings were observed when the user was 20 m or more beyond the boundary. Table VII lists median processing time per pipeline stage; total latency is under 5 ms per event.

Stage	Median Latency
Location Validator	< 1 ms
Location Smoother	< 1 ms
Geofence Engine	1–3 ms
State Machine	< 1 ms
Business Logic	< 1 ms
Total	< 5 ms

Table VII: pipeline processing latency

B. Presence Score Results

Tables VIII–X report the presence score decomposition for a legitimate user and two spoofing attack levels.

Signal	Score	Weight	Contribution
Geofence	100	40%	40.0
Face Identity	85	30%	25.5
Location Consistency	95	10%	9.5
Device Activity	90	10%	9.0
Motion Pattern	82	10%	8.2
Total			92.2 – Present

Table VIII: Presence Score – Legitimate User

Signal	Score	Weight	Contribution
Geofence	0	40%	0.0
Face Identity	50	30%	15.0
Location Consistency	100	10%	10.0
Device Activity	60	10%	6.0
Motion Pattern	75	10%	7.5
Total			38.5 – Absent

Table IX: Presence Score – Level 1 Gps Spoofing Attack

Signal	Score	Weight	Contribution
Geofence	100	40%	40.0
Face Identity	50	30%	15.0
Location Consistency	35	10%	3.5
Device Activity	40	10%	4.0
Motion Pattern	75	10%	7.5
Total			70.0 – Uncertain

Table X: Presence Score – Level 2 Spoofing (Hidden Mock Flag)

The Level 2 session was flagged with identity_unverified and location_inconsistent for administrator review. The 80-point threshold for a Present classification was not reached.

C. Mock Location Detection Performance

Table XI reports the true positive rate (TPR) and false positive rate (FPR) for each server-side detection method.

Detection Method	TPR	FPR
Android mock flag	100%[a]	0%
Unrealistic accuracy (< 5 m)	78%	3%
IP mismatch > 100 km	65%	1%
Low decimal precision	82%	0%
Repeated coordinates	90%	2%

[a] Level 1 attacks only.

Table XI: Detection Method Performance

D. API Response Latency

Table XII shows the median response time for each endpoint. The slowest by far is the session start call at 320 ms, but this is almost entirely due to the on-device face recognition inference rather than any server-side delay. In practice, users do not notice it -in the context of a manual check-in, 320 ms feels instant.

Endpoint	Median
POST /attendance/start (face check)	320 ms
POST /attendance/end	45 ms
POST /attendance/location/log	38 ms
GET /presence/score	22 ms
POST /user/login	180 ms

Table XII: Endpoint Response Latency

E. Known Limitations

- Indoor GPS accuracy: GPS performance drops significantly inside reinforced-concrete buildings, with accuracy degrading to 20–50 m or worse. Basement spaces and interior rooms may have little to no satellite visibility at all, which directly affects the geofence signal.
- Face detection in low light: The face detector struggles in dim environments. Users checking in in poorly lit spaces may experience failed or slow detections, requiring adequate lighting for reliable performance.
- IP geolocation is city-level only: The IP cross-check is only accurate enough to catch spoofing across cities or countries. Someone spoofing from a few streets away from the office would not be caught by this check alone.
- No large-scale field trial: All results presented in this paper come from controlled testing conditions. A longitudinal study with real employees across different environments is still needed to validate the system's performance in the wild.

VIII. Conclusion and Future Work

A. Conclusion

This paper presented Geo-Attendance, a mobile attendance system built on the principle that no single signal should be trusted on its own. By combining five independent signals into a weighted Presence Score (Equation above), the system makes it significantly harder for an attacker to fake their presence without being caught. In testing, legitimate users consistently scored around 92/100 and were correctly classified as Present. Level 1 spoofing attacks scored just 38.5/100 and were blocked entirely at the client pipeline before reaching the server. Level 2 attacks, which are considerably more sophisticated, reached 70.0/100 but fell short of the 80-point threshold and were flagged for administrator review. The full system -comprising a mobile app, backend server, and admin dashboard -is functional and ready to deploy, with an architecture that can scale to support multiple office locations.

B. Future Work

There are several directions we plan to explore going forward. (i) Indoor positioning using Wi-Fi signal strength fingerprinting would address the GPS limitations inside buildings. (ii) A continuous face re-verification mechanism would strengthen identity checks throughout a session, not just at check-in. (iii) On the deployment side, adding multi-tenant support would make the system viable for larger enterprise use. (iv) A formal security analysis using the STRIDE framework would help identify any remaining attack surface. (v) Finally, a real-world field study spanning 4 - 8 weeks with actual employees would give us ground-truth data on false positive, battery consumption, and how well users actually accept the system in daily use.

Acknowledgment

The authors acknowledge the open-source projects used in this work: MapLibre GL JS, face-api.js, expo-location, and Mongoose, all used under permissive open-source licences. Vector tile map data is provided by MapTiler under their academic-use tier.

References

[1] K. Manikam, "Biometrics and beyond: Innovating against buddy punching losses," Human Resource and Leadership J., vol. 9, no. 1, pp. 85–100, 2024.
[2] H. S. Gite, P. B. Kawade, H. S. Aushikar, A. M. Gangurde, and N. S. Khairnar, "GPS based attendance system using geofencing," Int. J. Sci. Dev. Res., vol. 8, no. 5, May 2023.
[3] I. S. Shehu and A. S. Gombe, "A smart attendance system using QR code," Int. J. Comput. Appl., vol. 127, no. 18, pp. 11–15, 2015.
[4] M. A. Ayu and B. I. Ahmad, "TouchIn: An NFC supported attendance system in a university environment," Int. J. Inf. Educ. Technol., vol. 4, no. 5, pp. 448–453, 2014.
[5] R. W. Sinnott, "Virtues of the Haversine," Sky Telesc., vol. 68, no. 2, p. 159, 1984.
[6] C. Jordan, Cours d'Analyse de l'Ecole Polytechnique, vol. 3. Paris, France: Gauthier-Villars, 1887.
[7] P. A. Zandbergen and S. J. Barbeau, "Positional accuracy of assisted GPS data from high-sensitivity GPS-enabled mobile phones," J. Navig., vol. 64, no. 3, pp. 381–399, 2011.
[8] W. Jansen and S. Hutchinson, "Guidelines on mobile device forensics," NIST Special Publication 800-101 Rev. 1, 2014.

[9] G. Oligeri, S. Sciancalepore, O. A. Ibrahim, and R. Di Pietro, "Drive me not: GPS spoofing detection via cellular network: Architectures, models, and experiments," in Proc. 12th ACM Conf. Security Privacy Wireless Mobile Netw. (WiSec), 2019, pp. 12–22.
[10] V. Mühler, "face-api.js: JavaScript API for face detection and face recognition in the browser," GitHub Repository, 2018.
[11] V. L. Erickson et al., "Occupancy modeling and prediction for building energy management," in Proc. 3rd ACM Workshop Embedded Sens. Syst. Energy-Efficiency Buildings, 2011, pp. 7–12.
[12] T. Diethe, N. Twomey, M. Kull, P. Flach, and I. Craddock, "Probabilistic sensor fusion for ambient assisted living," arXiv:1702.01209 [stat.ML], Feb. 2017.

