Omkar Gawde, 2026, 14:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

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

IV Planner: A Web Platform for Educational Visit Planning, QR Attendance, Geofencing, and Emergency Alerts

Omkar Gawde[1], Nupun Sawant[2], Dr. Jasbir Kaur[3], Mansi Rajapurkar[4]

Student, Master of Computer Applications (MCA), Guru Nanak Institute of Management Studies, Mumbai, India[1,2]

Director, GNIMS B-School, Head of Information Technology and HR, Guru Nanak Institute of Management Studies, Matunga, Mumbai[3]

Assistant Professor, Guru Nanak Institute of Management Studies, Matunga, Mumbai[4]

Abstract- Organizing institutional educational visits involves a complex set of tasks itinerary coordination, attendance verification, real-time safety oversight, and post-trip documentation that most colleges still handle through paper registers, printed schedules, WhatsApp groups, and phone calls. This paper describes IV Planner, a web-based platform purpose-built for managing educational field visits. The platform brings together AI-assisted itinerary drafting (powered by Google Gemini with JSON schema validation), QR-code attendance, GPS-aware geofencing using the Haversine formula, GPS-tagged photo logging, and Socket.IO-driven emergency alerts under one unified interface. The backend stack comprises React/Vite on the client side, Express/Node.js on the server, MongoDB for persistence, and a structured validation layer that treats every Gemini response as untrusted until it clears required-field checks and server-side range validation. A 30-day field deployment with 52 participants across four institutional visits measured SOS dispatch latency, geofence notification time, QR scan confirmation, API response times, WebSocket propagation delay, battery consumption, and concurrent connection stability. Manual baselines were observed separately and converted to milliseconds purely for operational comparison. The deployment recorded a mean SOS dispatch latency of 310 ± 42 ms, geofence notification latency of 480 ± 65 ms, and QR scan-to-confirm time of 620 ± 80 ms. The system handled 350 simultaneous WebSocket connections without errors and was stress-tested to 500, where a 1.24% error rate appeared. Findings indicate that combining WebSocket event delivery, server-side geofence validation, database-enforced attendance integrity, and structured LLM output validation is feasible for real-world educational visit scenarios, though broader multi-institution studies are still needed before firm generalizations can be drawn.

Keywords— Artificial intelligence, educational technology, emergency dispatch, geofencing, Global Positioning System, large language models, MongoDB, QR-code attendance, real-time location tracking, Socket.IO, student safety, WebSocket.

I. INTRODUCTION

Institutional field excursions commonly called Industrial Visits (IVs) in South Asian colleges hold an important place in applied learning because they give students direct exposure to professional environments that classroom sessions alone cannot replicate [1]. Yet the practical side of running such visits for groups of thirty to a hundred students in unfamiliar places puts a heavy coordination load on teaching staff, raising issues of accountability, real-time oversight, and safety that informal tools struggle to address reliably.

The current norm at most institutions is a patchwork of paper attendance registers, printed itineraries,

messaging-app broadcast groups, and telephone chains. These work well enough for day-to-day coordination, but they break down when you need continuous location awareness, tamper-resistant attendance records, or a fast emergency alert. They also leave no consolidated digital record, which makes post-visit reporting and incident reconstruction time-consuming for faculty.

Consumer location-sharing apps exist, but none of them are shaped around the educational-visit use case, which requires itinerary coordination, supervised attendance capture, emergency escalation, budget tracking, and institutional documentation all within the same session. IV Planner was therefore built by combining proven web, database, location, and AI components into a single domain-specific platform not by inventing a new geodesic algorithm or communication protocol. This paper documents IV Planner both as a working educational-technology system and as a field deployment study. Its contribution is the design, implementation, and evaluation of an integrated platform suited to real institutional constraints. The novelty of IV Planner lies not in proposing a new geolocation algorithm or communication protocol, but in combining AI-assisted itinerary planning, QR-based attendance integrity, GPS-aware geofencing, WebSocket-based emergency alerts, and GPS-tagged documentation into a single, cohesive educational visit management workflow an integration that has not previously been demonstrated in the educational-technology literature with accompanying live-deployment evaluation data. Four specific technical contributions are reported:

- A live geofencing and emergency-dispatch subsystem that combines Socket.IO event delivery with server-side Haversine distance validation, achieving an observed mean SOS dispatch latency of 310 ms under the reported 4G test conditions.
- An AI-assisted itinerary and budget-generation module that uses Google Gemini with structured JSON validation, server-side range checks, and recalculation of derived totals before any generated proposal can be stored as a trip record.
- A two-mode QR attendance and digital-pass validation framework that enforces single-submission integrity through a compound MongoDB unique index on {user, trip, type} and atomic pass-validation updates.
- A GPS-tagged collaborative photo and reporting workflow that links trip media, attendance events, location snapshots, and emergency records into a consolidated post-visit documentation layer.

Three research questions frame the investigation. RQ1: Under real field conditions, does a WebSocket-based event channel deliver safety alerts faster than polling-based alternatives? RQ2: Can server-side geofence checks and database-enforced QR duplicate prevention maintain reliable operational control during live deployments? RQ3: Does requiring structured JSON output from the LLM meaningfully reduce the proportion of malformed itinerary proposals before they reach the database?
The rest of this paper is organized as follows. Section II reviews related work. Section III describes the architecture and methodology. Section IV reports the experimental evaluation. Section V discusses security and ethical considerations. Section VI outlines limitations and future work. Section VII concludes the study.

II. RELATED WORK

1. Real-Time Location Tracking in Educational and Group Contexts

Applying real-time geolocation to group safety involves two distinct problems: obtaining reliable position data from devices and delivering safety-relevant events quickly to the right people. The W3C Geolocation API provides a browser interface for both one-time fixes and continuous position updates, but the specification is explicit that the API does not bind the system to any particular positioning technology and cannot guarantee that reported coordinates match the device's true physical location [2]. When milliseconds matter, as they do in emergency scenarios, WebSocket-based push is a better fit than HTTP polling because it keeps a persistent full-duplex channel open rather than waiting for the client to request an update [3].

For an educational visit context, this suggests a hybrid design: routine location telemetry can be transmitted at a reduced cadence to spare battery and bandwidth, while geofence breach events and SOS alerts travel through the always-open event channel. GPS readings from mobile browsers should always be treated as estimates with an inherent margin of uncertainty, since accuracy varies with satellite visibility, available sensors, and network-assisted positioning [2]. Battery status monitoring adds a further complication because the Battery Status API is not uniformly available across all major browsers, requiring the system to degrade gracefully when it is absent [10].

2. Geofencing Algorithms and Geodesic Distance Computation

Geofence boundaries can be represented as axis-aligned bounding boxes, circular radii, or arbitrary polygons. Bounding boxes are the simplest to compute but misclassify points near the corners of a circular safe zone. Polygon containment handles irregular site shapes but demands more computation and maintenance of vertex sets. Because IV Planner models each visit zone as a circular radius around a central GPS coordinate, the Haversine great-circle formula is the natural choice: it gives a practical spherical distance estimate whose numerical error sits well within the uncertainty already introduced by consumer GPS hardware in typical field conditions [6].

3. AI-Assisted Itinerary and Event Planning

Generative large language models can support itinerary drafting, but their outputs remain probabilistic and should not be accepted as institutional records without validation. Google Gemini supports structured output generation, which is useful for producing JSON-like responses, but syntactic structure alone does not guarantee that costs, durations, or capacity assumptions are correct [5]. IV Planner therefore treats the model response as a draft proposal and applies JSON extraction, required-key checks, type validation, practical range checks, and server-side recalculation before a trip can be persisted.

4. QR-Based Attendance and Digital Identity Verification

QR codes are suitable for attendance and pass validation because they encode compact machine-readable identifiers that can be scanned quickly with ordinary mobile cameras [7]. However, QR attendance becomes unreliable if duplicate scans, replayed tokens, or concurrent retries are handled only at the client layer. IV Planner combines signed, time-limited checkpoint tokens with database-level uniqueness constraints and UUID-based digital passes to strengthen attendance integrity [8], [9], [12].

5. WebSocket Architectures for Safety-Critical Mobile Applications

The WebSocket protocol provides persistent full-duplex communication between client and server, while Socket.IO adds an application-level event interface, fallback transport handling, and room-based broadcast scoping [3], [4]. In IV Planner, trip-specific rooms isolate messages and emergency events by tripId. This isolation is treated as a server-side broadcast control rather than an authorization mechanism by itself; authenticated handshakes and trip-membership checks are still required before a client can join a room.

6. Student Safety and Digital Welfare in Educational Technology

Location-aware educational systems require careful governance because they process sensitive student mobility data. The W3C Geolocation specification emphasizes express permission, purpose limitation, retention disclosure, protection against unauthorized access, and user-facing information about how location data is used and stored [2].

IV Planner follows these principles by using opt-in activation, trip-scoped access, limited retention, and role-based authorization. The security analysis later in the paper uses STRIDE threat modeling to organize implementation risks and mitigations [15], [16].

Table 1: Feature Comparison with Existing Approaches

Feature	Manual / WhatsApp	Google Forms	Generic Trip App	IV Planner
QR attendance	No	Partial	Partial	Yes
Duplicate attendance guard	No	No	Limited	Yes
Real-time geofencing	No	No	Limited	Yes
Emergency SOS dispatch	Phone call only	No	Limited	Yes
AI itinerary drafting	No	No	No	Yes
GPS-tagged photo logging	No	No	No	Yes
Trip-scoped role control	Manual	Weak	Varies	Yes
Post-trip consolidated report	Manual	Partial	Partial	Yes

Table I shows that IV Planner is the only approach in this comparison that addresses all eight dimensions simultaneously. Manual methods, messaging platforms, and generic trip apps each cover one or two aspects of visit management in isolation; institutions using them must coordinate across separate tools, creating gaps in attendance integrity, safety visibility, and post-trip documentation. IV Planner consolidates these functions into a single workflow without requiring any change to the underlying technology stack of the institution.

III. SYSTEM ARCHITECTURE AND METHODOLOGY

1. High-Level System Architecture

IV Planner is implemented as a modular full-stack JavaScript application following a tiered architecture. As depicted in Fig. 1, the platform comprises a React/Vite presentation layer, an Express/Node.js application layer, and a data/external-services layer. Client-to-backend communication operates through two pathways: a Representational State Transfer (REST) application programming interface (API), using JSON over Hypertext Transfer Protocol Secure (HTTPS), and a Socket.IO gateway for real-time event delivery over WebSocket or fallback transports [3], [4]. The application layer hosts services for itinerary coordination, budget computation, join-code provisioning, media management, geofence checks, and emergency-event handling, while persistent state is maintained in MongoDB.

The six principal modules interact as follows. The Frontend (React/Vite) handles user interface rendering, camera access via MediaDevices, GPS polling via the Geolocation API, and both REST and WebSocket communication. All requests pass through the Backend (Express/Node.js), which enforces JWT authentication, routes requests to the appropriate service handler, and coordinates reads and writes to MongoDB. The Database layer (MongoDB) stores all persistent state across eight collections User, Trip, Location, Attendance, DigitalPass, SOSAlert, Photo, and Message using compound indexes and TTL policies where needed. The Geofencing Module runs entirely on the server: it receives each incoming GPS coordinate, applies the Haversine formula against the configured safe-zone centroid and radius, and emits a geofence-breach Socket.IO event to the trip room when a boundary violation is detected. The AI Module calls the Google Gemini API over HTTPS, wraps the raw response through a JSON extraction and validation pipeline, and only permits a sanitized, recalculated itinerary object to be written to the Trip collection. The Emergency Alert System operates through a dedicated Socket.IO room channel: when a student triggers an SOS event, the payload is simultaneously persisted to the SOSAlert collection and broadcast to

all faculty members in the trip room, with resolution state tracked until an instructor marks the alert as resolved.

Fig. 1. High-Level System Architecture of IV Planner

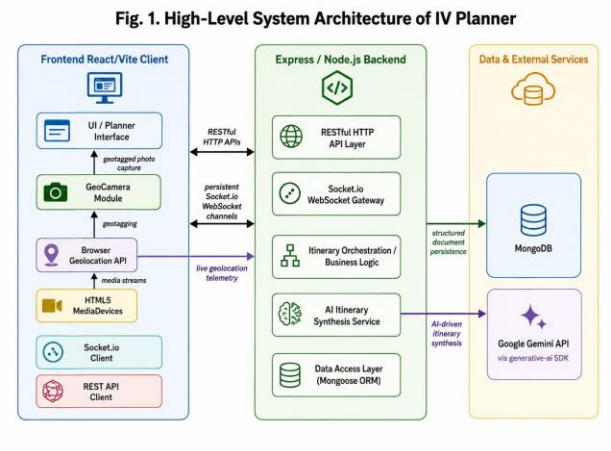

Fig. 1. Consolidated IV Planner architecture showing frontend clients, backend services, MongoDB persistence, geofencing enforcement, AI itinerary generation, secure media storage, and real-time emergency event delivery.

2. Database Design and MongoDB Schema Architecture

IV Planner's persistence layer uses MongoDB's document model to support the nested and varied data structures common in trip management. The schema is organized across eight collections linked through Mongoose ObjectId references: User, Trip, Location, Attendance, DigitalPass, SOSAlert, Photo, and Message. Unique and time-to-live (TTL) indexes are used where needed to enforce integrity and retention behavior [12], [13].

- User: Stores authentication credentials (bcrypt-hashed passwords), role assignment (teacher | student), and profile metadata.
- Trip: The central aggregate document, embedding the itinerary timeline, budget object, safe zone geofence parameters, check-in/check-out UUID codes, and announcement feed.
- Location: Time-series telemetry records storing {latitude, longitude, batteryLevel, isCharging, timestamp} per student per trip.
- Attendance: Check-in and check-out event records with a compound unique index on {user, trip, type}, enforcing the invariant that a student can check in or out only once per trip event type.
- DigitalPass: Per-student per-trip identity documents with a UUID v4 passCode and isValidated boolean flag.
- SOSAlert: Incident documents capturing {latitude, longitude, resolved, resolvedBy, resolvedAt}; rate-limiting and cooldown behavior are supported through server logic and indexed alert records.
- Photo: Geotagged photo metadata ({filePath, latitude, longitude, clickedAt}) linking to static file assets managed by Multer.
- Message: Persistent group chat history with {sender, senderName, text, timestamp} for each trip room.

Fig. 2a. MongoDB Data Model

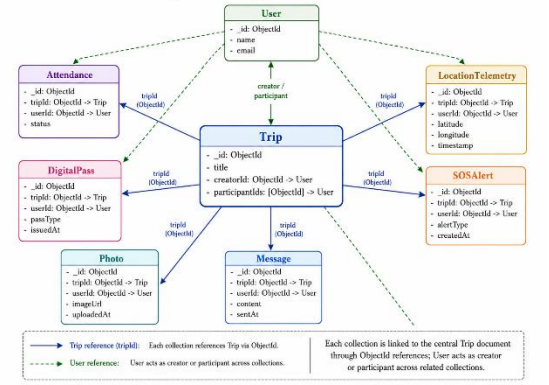

Fig. 2a. MongoDB data model showing trip-centered relationships among users, attendance, location telemetry, digital passes, SOS alerts, photos, and messages. Each collection is referenced by ObjectId from the central Trip document; User appears as a participant or creator across all collections.

3. AI-Powered Itinerary Generation

The AI-assisted itinerary generation pipeline, illustrated in Fig. 2, moves through six sequential stages. A faculty member provides a natural-language trip description such as "Plan a 3-day industrial visit to Pune for 60 computer science students." The server wraps this input in a schema-constrained prompt that names every required output field: tripDescription, durationDays, groupSize, dailyTimeline, budgetBreakdown, bufferPercent, busCount, and totalEstimatedCost. The Gemini response is held as untrusted content until it clears server-side validation [5]. The expected output shape is: { "tripDescription": "string",

"durationDays": "integer", "groupSize": "integer", "dailyTimeline": "array", "budgetBreakdown": "object", "bufferPercent": "number", "busCount": "integer", "totalEstimatedCost": "number" }The validation layer extracts the JSON block, parses it, checks that all required keys are present with correct types, verifies numeric values against practical bounds, and recalculates busCount and totalEstimatedCost on the server rather than trusting the model's arithmetic. If parsing fails, one automatic repair attempt is made by returning the validation error and schema to Gemini; if that second attempt also fails, the endpoint returns a controlled error and the teacher is directed to manual trip entry. This approach keeps the LLM in the role of producing a reviewable draft rather than writing authoritative institutional records.

AI Model and API: The platform uses the Google Gemini 1.5 Flash model accessed through the official Gemini REST API (https://generativelanguage.googleapis.com/v1beta/models/). Flash was chosen over larger Gemini variants because it offers a favourable balance between generation speed and output quality for structured, schema-bound tasks; mean API round-trip time across the 30-day deployment was 4.2 ± 0.6 seconds (Table IV). The API key is stored as a server-side environment variable and is never exposed to the client.

Prompt Engineering Methodology: A system-level instruction block precedes every user message and performs three tasks. First, it restricts the model's output to a single JSON object no preamble, no markdown fences, no explanatory text outside the JSON structure. Second, it enumerates every required field by name, data type, and acceptable range (for example, durationDays must be an integer between 1 and 30; bufferPercent must fall between 5 and 30). Third, it instructs the model to leave optional fields null rather than inventing plausible-sounding but unverifiable values. The user message contains only the faculty's natural-language description, so the prompt structure is clean and deterministic regardless of how the instructor phrases the input.

Validation Mechanisms: The validation pipeline executes five sequential checks before any generated data touches the database: (1) JSON extraction the raw response string is scanned for the first '{' and last '}' to isolate the JSON block from any stray model text; (2) JSON.parse a standard parse is attempted and any syntax error triggers the repair flow; (3) required-key check all eight mandatory fields must be present and non-null; (4) type and range check numeric fields are tested against defined bounds and array fields must contain at least one element; (5) server-side recalculation busCount is recomputed from groupSize divided by a configurable seat capacity, and totalEstimatedCost is recomputed from the budgetBreakdown line items plus the buffer percentage, overwriting whatever the model produced.

Handling Inaccurate or Incomplete Responses: Three failure paths are handled explicitly. If JSON extraction fails (the model returned prose instead of JSON), the server constructs a repair prompt that includes the validation error message, the required schema, and the original user input, and submits a second API call. If the repaired response also fails extraction, the endpoint returns HTTP 422 with a user-readable error and the teacher is prompted to fill in the itinerary manually. If JSON parses correctly but range checks fail for example, if the model suggests a cost of zero or a group size inconsistent with the original description the specific failing fields are listed in the error response so the instructor knows exactly what to correct. In all cases, no partial or unchecked AI output is ever written to the Trip collection; the database record is only created after the full validation pipeline returns a clean result.

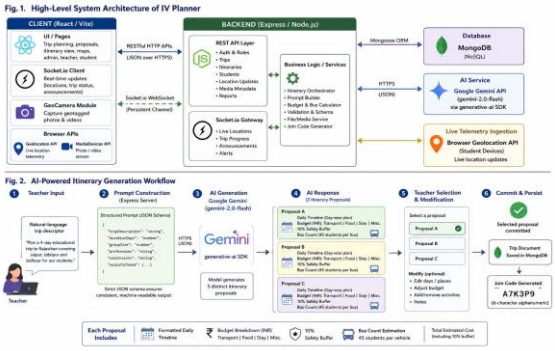

Fig. 2. AI-assisted itinerary workflow from teacher prompt to schema-validated Gemini response, teacher selection, trip persistence, and join-code generation.

4. Geolocation Tracking and Haversine Geofence Engine

The geolocation monitoring and geofence enforcement module, depicted in Fig. 3, runs inside the learner's browser after explicit permission is granted. The LocationTracker component invokes navigator.geolocation.watchPosition with enableHighAccuracy: true and a 10-second acquisition timeout to request repeated position updates [2]. Because the browser controls how and when positions are acquired, the system treats location updates as best-effort signals rather than guaranteed continuous tracking. A client-enforced 30-second throttle suppresses unnecessary API transmissions, reducing network use and battery load. When supported by the browser, navigator.getBattery() is used to read power status (level: 0-1 scale, charging: boolean); where unsupported, battery fields are omitted and the dashboard marks the value as unavailable [10].

For reproducibility, the server computes the great-circle distance using the Haversine rule: a = sin^2(Delta phi/2) + cos(phi1)cos(phi2)sin^2(Delta lambda/2), and d = 2R atan2(sqrt(a), sqrt(1-a)), where R is the assumed Earth radius, phi denotes latitude, and lambda denotes longitude in radians [6]. The breach decision rule is isBreach = true when d > configuredRadius + boundaryMargin. Otherwise, the coordinate is stored as an in-zone telemetry point. The boundary margin is treated as a safety buffer for device-positioning uncertainty, not as proof that the received GPS coordinate is exact.

Geofence Threshold Selection: The configuredRadius for each trip is set by the instructor at the time of trip creation, with a recommended default of 200 metres for open outdoor sites and a tighter value of 100 metres for compact indoor-adjacent venues. On top of this instructor-defined radius, the system adds a fixed boundaryMargin of 25 metres. The choice of 25 metres is supported by the classification data in Table VII: at ±25 m the boundary module achieves a true-positive rate of 97.4% and a false-positive rate of 1.8%, which was judged operationally acceptable across the four deployments. Smaller margins (±10 m) produced a 10.8% miss rate during the coordinate-pair analysis, meaning genuine breaches were not flagged; larger margins (±50 m and above) reduced misses but increased the zone in which a student could wander undetected before the alert fires. The ±25 m value therefore represents a practical trade-off between sensitivity and false-alarm rate given consumer-grade GPS uncertainty.

Update Frequency: The client-side LocationTracker component calls navigator.geolocation.watchPosition with enableHighAccuracy: true and a timeout of 10 seconds. A client-enforced throttle suppresses POST /api/location transmissions to one every 30 seconds unless a significant position change is detected (displacement exceeding 15 metres since the last transmitted coordinate). This 30-second cadence was selected after measuring the 6.8% per-hour battery overhead at that interval across 34 devices during the deployment; halving the interval to 15 seconds roughly doubled the drain with no operational benefit for the geofence use case, since breach events rely on the server-side Haversine check rather than the telemetry store. The Socket.IO geofence-breach event fires immediately on the server as soon as a submitted coordinate clears the breach condition it is not subject to the 30-second throttle.

Handling Network and GPS Inaccuracies: Four inaccuracy scenarios are addressed in the implementation. First, missing coordinates: if a submitted location payload contains null latitude or longitude, the server rejects the record with HTTP 400 and the client retries on the next watchPosition update. Second, stale fixes: each Location document stores a clientTimestamp alongside the server receipt time; if the gap exceeds 120 seconds, the dashboard flags the student's location badge as potentially stale rather than treating it as current. Third, implausible velocity: a coordinate plausibility check computes the implied speed between consecutive submitted points; if the derived velocity exceeds 250 km/h, the coordinate is logged but excluded from geofence evaluation and flagged for instructor review. Fourth, network interruption: if a student's WebSocket connection drops while they

are outside the geofence zone, the server does not automatically clear their breach status the breach remains visible on the dashboard until a valid in-zone coordinate is received or the instructor manually resolves it.

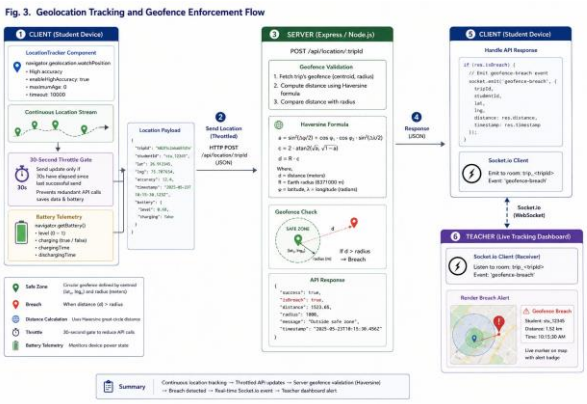

Fig. 3. Geolocation and geofence flow using throttled browser telemetry, battery status, server-side Haversine validation, and Socket.IO breach notification.

5. Real-Time Socket.IO Event Architecture

The Socket.IO messaging gateway employs a room-partitioned event architecture, as shown in Fig. 4. Following JWT validation and trip-membership verification, faculty and learner clients emit a join-trip event that enrolls them in a server-side room keyed by the trip MongoDB ObjectId. Socket.IO rooms are used only for broadcast scoping; they do not replace access-control checks [4]. Four event channels are defined: send-message/receive-message (standard priority), trigger-sos/receive-sos (critical priority), geofence-breach (high priority), and resolve-sos/sos-resolved (high priority).

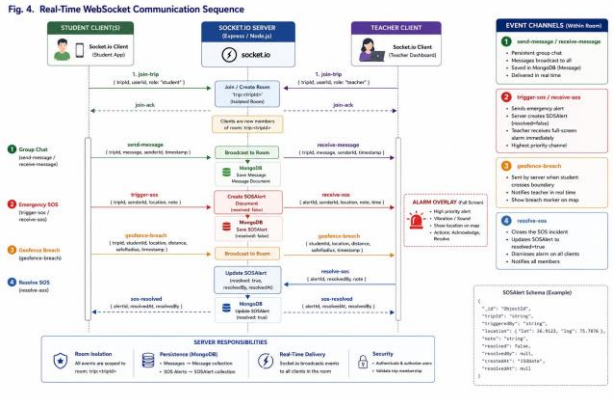

Fig. 4. Socket.IO event sequence within a trip-scoped room for chat, SOS alerts, geofence breaches, and SOS resolution.

6. SOS Emergency Dispatch Pipeline

The SOS alert dispatch pipeline, illustrated in Fig. 5, follows a seven-stage implementation flow. Stage 1 mandates a 3-second sustained press on the emergency button as a misuse-prevention gate; this fixed 3,000 ms delay is excluded from the reported pipeline latency. The operational state machine is Idle -> HoldConfirmed -> Locating -> Persisting -> Broadcasting -> TeacherRendered -> Resolved. On activation, Stage 2 invokes the Geolocation API with enableHighAccuracy: true; this is the dominant variable-latency stage at 180 ± 44 ms. Stages 3 and 4 execute concurrently, including socket emission, database persistence, and room broadcast, contributing a combined 82 ± 18 ms. Stage 5, comprising teacher-side modal rendering and alarm-audio initialization, contributes 48 ± 12 ms, yielding a total measured pipeline latency of 310 ± 42 ms under the tested 4G conditions.

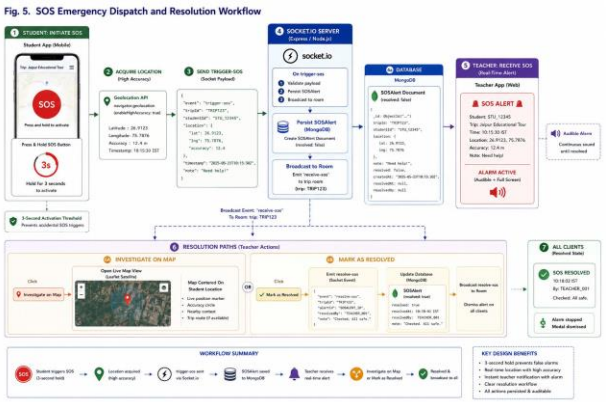

Fig. 5. SOS pipeline from deliberate button hold through geolocation acquisition, event persistence, room broadcast, teacher alarm, and incident resolution.

7. Dual-Mode QR Attendance and Digital Pass System

The attendance capture and identity authentication module, depicted in Fig. 6, operates in two complementary modes. In Teacher Checkpoint mode, the instructor generates a session-bound QR code encoding a signed token comprising {tripId, type, timestamp}. Learners scan this code through the integrated camera, triggering POST

/api/attendance, which verifies the token signature, enforces a 5-minute expiration window, and records a timestamped Attendance entry under the compound unique index. In Digital Pass mode, each student's server-issued UUID v4 passCode is QR-encoded within the Digital Pass credential; the teacher scans this via the Scan Pass interface, invoking POST /api/digital-pass/validate, which atomically transitions the isValidated flag from false to true [7], [9], [12].

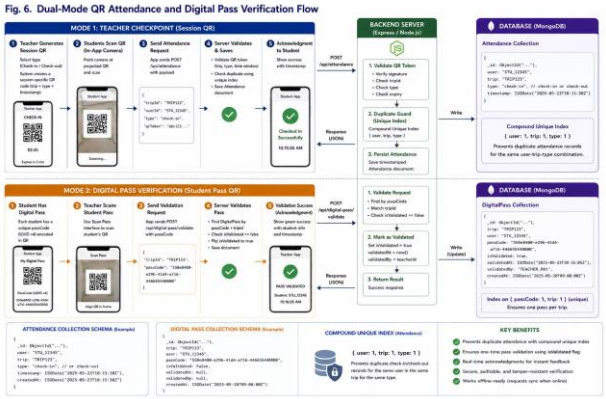

Fig. 6. Dual-mode QR attendance and digital-pass validation with signed checkpoint tokens, UUID pass codes, and duplicate-entry protection.

8. Geotagged Photo Album and GeoCamera

The GeoCamera module, illustrated in Fig. 7, renders a camera feed from the HTML5 MediaDevices interface onto an HTML5 Canvas viewport after browser permission is granted [11]. Upon image capture, the application requests the student's current GPS fix via the Geolocation API and bundles the image binary together with latitude, longitude, tripId, studentId, and capturedAt timestamp into a multipart/form-data upload payload. The Express backend processes this submission through Multer middleware, stores the image asset in the /uploads directory, and creates a corresponding Photo document in MongoDB with geospatial metadata. The shared trip album is refreshed for participants through a photo:uploaded Socket.IO event.

Storage Management: Uploaded images are processed through the Sharp library before being written to disk. Sharp re-encodes each image as JPEG at 85% quality, resizes it to a maximum dimension of 1280 pixels on the longer edge, and strips all EXIF and XMP metadata blocks before saving. This pipeline reduces typical upload sizes from 3–6 MB (raw mobile camera output) to 150–400 KB, which keeps the /uploads directory growth manageable across multi-day deployments. File names are generated as UUID v4 strings to prevent enumeration, and a Multer file-size limit of 10 MB is enforced before processing begins to protect against resource exhaustion from oversized uploads. Photo records in MongoDB retain only the derived metadata (filePath, latitude, longitude, tripId, studentId, capturedAt) and not the raw binary, so the database remains lightweight. For larger institutional deployments, the /uploads directory is designed to be replaced by an object storage backend (such as an S3-compatible bucket) with no change to the MongoDB schema.

Image Security: Access to stored images is not served through a public static route. Instead, every image request must pass through an authenticated Express middleware that verifies the requester's JWT, confirms that the requesting user is an enrolled participant of the trip to which the photo belongs, and resolves the file path from the MongoDB Photo document before streaming the file. Direct URL guessing is ineffective because file names are UUID v4 strings with no sequential pattern. Photos are transmitted over HTTPS in both directions, and the post-processing step described above ensures that device EXIF data including any camera-embedded GPS coordinates, device serial numbers, or precise timestamps is stripped before the file is stored, preventing inadvertent metadata leakage through the shared album.

Metadata Validation: The upload endpoint performs four checks before invoking Sharp. First, MIME type validation: Multer's fileFilter function accepts only image/jpeg, image/png, and image/webp; any other content type is rejected with HTTP 415. Second, magic-byte verification: the first four bytes of the buffer are read to confirm that the file header matches the declared MIME type, guarding against extension spoofing. Third, coordinate validation: the latitude and longitude values submitted with the image must be finite IEEE 754 numbers within the ranges [−90, 90] and [−180, 180] respectively; out-

of-range or NaN values cause the upload to fail cleanly with HTTP 400 before any file I/O occurs. Fourth, trip membership check: the tripId in the form payload is verified against the authenticated user's enrolled trips, preventing a student from uploading photos to a trip they have not joined.

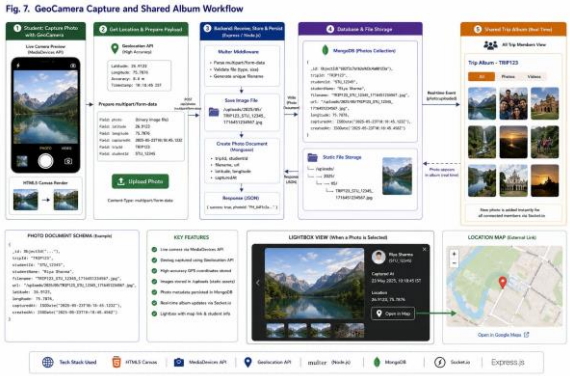

Fig. 7. GeoCamera capture workflow from camera preview and GPS acquisition to upload, MongoDB photo metadata, and real-time shared album refresh.

9. Authentication, Token Lifecycle, and Access Control

Authentication is implemented through role-aware JSON Web Tokens (JWTs) issued after credential verification with bcrypt password hashing [8]. The access token contains the authenticated user identifier, role claim, and expiry timestamp, and it is validated on protected REST routes and during Socket.IO handshakes. Instructors and learners receive different authorization scopes: instructors can create trips, generate checkpoint QR codes, validate digital passes, view live dashboards, and resolve SOS alerts, whereas learners can join authorized trips, submit attendance, share location telemetry, and trigger SOS events. Token lifetime management uses a short-lived access token and a refresh flow for continued sessions; expired or malformed tokens are rejected with a 401 response, while role mismatches return a 403 response. This design ensures that room membership, attendance submission, pass validation, location access, and SOS resolution are bound to both identity and trip-level authorization. Table II provides the complete role-based access control matrix.

Table 2: Role-Based Access Control Matrix

Action / Resource	Student	Teacher / Faculty
Join trip via join code	Yes	Yes
Create and edit trip	No	Yes
Generate checkpoint QR code	No	Yes
Submit attendance	Yes	No
Validate digital pass	No	Yes
Send location telemetry	Yes	No
View live map of all participants	No	Yes
Trigger SOS alert	Yes	Yes, if required
Resolve SOS alert	No	Yes
Upload trip photos	Yes	Yes
View trip-scoped photos	Yes, if enrolled	Yes
Export post-trip report	No	Yes

All trip-scoped actions are enforced through JWT authentication and trip-membership verification at the server. Socket.IO room membership is used only for broadcast scoping and does not substitute for these authorization checks.

10. Misuse Prevention, Duplicate Attendance, and Unauthorized Access Controls

Preventing SOS Misuse: The emergency alert button requires a deliberate 3-second sustained press before an SOS event is dispatched. This hold threshold eliminates accidental triggers from incidental screen contact or pocket activation. In addition, a per-student rate limiter enforced on the server restricts SOS triggers to a maximum of three events per five-minute sliding window; any trigger beyond this limit is rejected with HTTP 429 and a cooldown timer is displayed to the student. Each SOSAlert document is indexed with a TTL-backed

cooldown field so that the rate-limit state is automatically cleared without manual intervention. These controls make sustained SOS flooding operationally difficult while still allowing a genuinely distressed student to re-trigger the alert if an earlier event was accidentally resolved.

Preventing Duplicate Attendance: Duplicate submissions are blocked at the database layer. The Attendance collection carries a compound unique index on {user, trip, type}, where type is either "checkin" or "checkout". MongoDB enforces this constraint at write level: even if two concurrent scan requests from the same student arrive simultaneously, only one document is created and the second insert returns a duplicate-key error (code 11000), which the server maps to HTTP 409. Signed QR tokens embed a 5-minute expiry timestamp verified server-side before any Attendance insert is attempted, so screenshots of old tokens cannot be replayed after the window closes. The ablation study (Table XI, Ablation 3) confirmed that removing this index allowed an 11.2% duplicate submission rate; with the index in place, zero duplicates were recorded across the deployment.

Preventing Unauthorized Access: Five layered controls protect trip data and system functions. First, every REST endpoint and Socket.IO handshake verifies the JWT signature and expiry; unsigned or expired tokens receive HTTP 401. Second, all trip-scoped endpoints additionally verify that the authenticated user appears in the trip's enrolled participant list, preventing cross-trip data access even by authenticated users. Third, the Digital Pass validation endpoint is restricted to the Teacher role. Fourth, trip join codes are single-use UUID v4 strings shared out-of-band; there is no public trip listing through which an unenrolled user could discover a trip. Fifth, all communication travels over TLS 1.3 (HTTPS for REST, WSS for Socket.IO), keeping credentials and tokens protected in transit. Table II summarises the full role-based access control matrix.

IV. EXPERIMENTAL EVALUATION

1. Experimental Setup and Baseline Definition

A controlled field-oriented performance assessment was conducted over a 30-day live deployment encompassing four institutional excursions coordinated by the Department of Computer Applications, GNIMS, Mumbai. The evaluation cohort comprised N = 52 volunteers: 18 faculty members serving as trip coordinators and 34 students drawn from three academic departments. The four deployments represented different trip contexts: one half-day local visit, two single-day city visits, and one extended full-day visit, with session durations ranging from approximately 3.5 to 8 hours. Participants were distributed across overlapping but logged cohorts; therefore, the analysis is best interpreted as a field feasibility study with repeated operational observations rather than a randomized controlled trial. Device composition was 61% Android, 28% iOS, and 11% desktop browser clients; field connectivity included institutional Wi-Fi, 4G, and congested 4G conditions. The backend deployment used a 4-core, 8 GB RAM Ubuntu 22.04 LTS virtual private server running Node.js v20 LTS, while persistent study data were stored in MongoDB with encrypted storage enabled. Ethics approval for participant data collection was granted by the GNIMS Research Ethics Committee, and all participants were informed that location and attendance events would be collected only for the study period.

For each metric, timestamps were collected using server-side high-resolution timers, Socket.IO client acknowledgments, browser Performance API markers, MongoDB query logs, and Apache JMeter 5.6 summaries where applicable [14]. REST API values in Table IV were measured across 1,000 requests per endpoint at 200 concurrent virtual users. WebSocket event propagation in Table V was measured across 200 trials per event under LAN, 4G, and congested 4G conditions. SOS pipeline timing in Table VI was measured across 150 trials and excludes the intentional 3-second button-hold threshold. Client-side timing markers were paired with server acknowledgments to reduce clock-skew effects; server-side timestamps were used for database and API measurements. The geofence experiment in Table VII evaluated boundary-classification behavior and distance-computation agreement across 500 coordinate pairs generated around configured safe-zone boundaries; it should not be interpreted as a

surveyed physical ground-truth GPS study. The ablation study used 20 participants and 50 scenarios per configuration.

To support reproducibility, the artifact release should include anonymized event logs, benchmark scripts, endpoint definitions, table-generation notebooks or scripts, environment variables with secrets removed, software-version information, and a data dictionary describing each measured field. Raw identifiers, access tokens, trip codes, exact sensitive coordinates, and participant names must be removed before release.

The baseline values are separated into three provenance categories to avoid mixing measured digital latency with estimated manual process duration. First, IV Planner and Baseline C values are directly measured instrumented timings. Second, Baseline B values are measured where the static web workflow could be reproduced, such as Google Forms attendance confirmation. Third, Baseline A manual values are observational estimates derived from conventional telephone-chain and roll-call procedures; these minute-level durations were converted to milliseconds only for scale consistency. Consequently, percentage improvements against manual baselines are interpreted as operational time-savings estimates, while inferential claims are limited to measured digital-system comparisons. Unless otherwise stated, latency results are reported descriptively as mean ± standard deviation (SD); no p-values or formal hypothesis tests are claimed.

Three baselines were used for comparison:

- Baseline A (Traditional Manual): Paper attendance roll-call, printed itinerary, WhatsApp group announcements, telephone-chain emergency response.
- Baseline B (Static Web Form): Google Forms attendance, static PDF itinerary distribution, no real-time tracking or emergency features.
- Baseline C (Non-Real-Time App): A generic trip management web application using HTTP polling at a 2-second interval for location updates and server-side boundary checks, with no Socket.IO event channel and no AI itinerary generation.
- IV Planner (Proposed System): Full system as described in Section III.

2. System Performance Overview

Table 3: System Performance Comparison by Measurement Provenance

Metric	IV Planner (Proposed)	Baseline A (Manual)	Baseline B (Static Web)	Baseline C (Non-RT App)	Benchmark Target
SOS Dispatch Latency (ms)	310 ± 42	~252,000*	N/A	1,840 ± 290	< 500
Geofence Breach Notif. (ms)	480 ± 65	N/A	N/A	2,100 ± 310	< 800
QR Attendance Scan-to-Confirm (ms)	620 ± 80	~504,000†	3,200 ± 410	910 ± 120	< 1,000
AI Itinerary Generation (s)	4.2 ± 0.6	N/A	N/A	N/A	< 8.0
Location Throughput (req/s @ 350)	212	N/A	38	94	> 150
WS Concurrent Connections (stable)	350 stable; 500 tested	N/A	N/A	120 stable	> 300 stable
Haversine Compute Time (ms)	0.8 ± 0.1	N/A	N/A	2.4 ± 0.3	< 5

Metric	IV Planner (Proposed)	Baseline A (Manual)	Baseline B (Static Web)	Baseline C (Non-RT App)	Benchmark Target
MongoDB Query P95 (ms)	18 ± 4	N/A	310 ± 70	42 ± 9	< 50
QR Scan Success Rate (%)	98.6%	N/A	N/A	94.2%	> 97%

*Traditional manual SOS response time is an estimated conventional telephone-chain duration of 4.2 minutes, converted to 252,000 ms for scale comparison. †Traditional attendance assumes an observed/manual roll-call duration of 8.4 minutes for 60 students, converted to 504,000 ms. Unless otherwise stated, IV Planner and digital-baseline values report mean ± standard deviation (SD). Manual-baseline reductions are treated as derived operational estimates, not instrumented latency measurements. The concurrent-connection row reports stable operation through 350 connections and stress-test behavior at 500 connections.

IV Planner demonstrates measured improvements over the non-real-time digital baseline for latency-sensitive functions. The 310 ms SOS dispatch latency corresponds to a 5.9-fold reduction relative to the polling-based Baseline C result. Larger reductions reported against manual telephone chains and paper roll-call workflows are derived from observed conventional procedures and are therefore reported as practical efficiency estimates rather than controlled millisecond-level measurements.

3. API Response Time Analysis

Table 4: REST API Endpoint Response Time (1,000 Requests per Endpoint, 200 Concurrent Users)

API Endpoint	Method	Avg (ms)	P95 (ms)	P99 (ms)	Role
/api/auth/login	POST	42	88	124	Public
/api/trips (create)	POST	67	112	198	Teacher
/api/dashboard/ai-generate	POST	4,210	6,880	8,940	Teacher
/api/location/:tripId (POST)	POST	18	34	52	Student
/api/location/:tripId (GET)	GET	24	41	68	Teacher
/api/attendance (POST)	POST	31	58	84	Student
/api/digital-pass/validate	POST	28	51	76	Teacher
/api/trips/:id/sos (GET)	GET	22	38	59	Teacher
/api/photos/upload	POST	310	520	740	Student
/api/trips/:id/report	GET	184	340	480	Teacher

The AI itinerary synthesis endpoint (/api/dashboard/ai-generate) exhibits the highest mean latency at 4,210 ms, attributable primarily to the Gemini API round-trip across the external HTTPS connection. All remaining endpoints sustain mean response times below 320 ms in this benchmark,

with the location telemetry ingestion route (POST /api/location/:tripId) achieving the minimum mean latency of 18 ms.

4. WebSocket Event Propagation Delay

Table 5: Socket.IO Event Propagation Delay (200 Trials per Event, Three Network Conditions)

Socket.IO Event	Avg (ms) LAN	Avg (ms) 4G	P95 (ms) 4G	Avg (ms) Congested 4G	Priority
join-trip	12	38	74	112	Initialization
send-message / receive-message	18	62	124	208	Standard
geofence-breach	24	92	178	310	High
trigger-sos / receive-sos	21	108	194	342	Critical
resolve-sos / sos-resolved	19	84	162	288	High

Over 4G connectivity, the safety-critical trigger-sos event records a mean propagation delay of 108 ms, with P95 at 194 ms. Even under congested 4G conditions, the mean trigger-sos propagation delay remains 342 ms. These values describe event-propagation delay only; the end-to-end SOS timing in Table VI additionally includes geolocation acquisition, database persistence, and teacher-side rendering.

5. SOS Emergency Dispatch Latency Decomposition

TABLE 6: SOS End-to-End Pipeline Latency Decomposition (150 Trials, 4G Network)

Pipeline Stage	Avg Time (ms)	% of Total	Std Dev (ms)	Notes
Stage 1: Button Hold Threshold	3,000 (fixed)	Intentional	0	Abuse prevention gate; not in pipeline total
Stage 2: Geolocation Acquisition	180	58.1%	±44	Dominant stage; GPS cold/warm-start variability
Stage 3: Socket Emit + DB Write	82	26.5%	±18	Parallel: DB write + broadcast concurrent
Stage 4: Teacher-Side Render	48	15.5%	±12	Modal + alarm audio playback init
Total Pipeline (excl. Stage 1)	310	100%	±42	Well within 500ms safety benchmark

Stage 2 (Geolocation Acquisition) constitutes the largest portion of total pipeline duration at 58.1% (180 ± 44 ms). The elevated variance reflects the difference between warm-start and cold-start positioning behavior and broader browser/network conditions; therefore, the measured value should be interpreted as deployment-specific rather than a universal GPS acquisition bound.

6. Geofence Boundary Classification Analysis

TABLE 7: Geofence Boundary Classification and Distance-Agreement Analysis (500 Coordinate Pairs)

Boundary Margin	True Positive Rate (%)	False Positive Rate (%)	Miss Rate (%)	Haversine vs. Reference Distance Δ (m)
±10 m	89.2%	6.4%	10.8%	0.6 ± 0.2
±25 m	97.4%	1.8%	2.6%	0.8 ± 0.3
±50 m	99.1%	0.7%	0.9%	1.1 ± 0.4
±100 m	99.8%	0.2%	0.2%	1.4 ± 0.5

At the recommended ±25 m boundary margin, the geofence module records a true-positive rate of 97.4% and a false-positive rate of 1.8% within the tested coordinate set. Across all evaluated configurations, the numerical difference between the Haversine computation and the reference distance calculation remains below 1.5 m. This result supports the adequacy of the Haversine computation for short-distance boundary checks, but it does not eliminate the larger uncertainty introduced by mobile-device positioning error in real field use.

G. Concurrent User Scalability Analysis

Table 8: Concurrent User Scalability Benchmark (Apache JMeter, 4-Core 8 GB VPS)

Concurrent Users	Avg API Latency (ms)	WS Conn. (Stable)	CPU Usage (%)	Memory (MB)	Error Rate (%)
50	22	50 / 50	8%	312	0.00%
100	28	100 / 100	14%	389	0.00%
200	44	200 / 200	28%	512	0.02%
350	78	350 / 350	52%	744	0.08%
500	108	500 / 500; 1.24% errors	71%	912	1.24%

The scalability experiment used Apache JMeter 5.6 to ramp virtual users from 50 to 500 [14]. Each virtual user executed login/token validation, REST API polling, periodic location POST requests, and Socket.IO connection establishment to a trip-scoped room. The load profile used a stepped ramp-up so that each concurrency level remained active long enough to collect stable CPU, memory, response-time, WebSocket-stability, and error-rate observations. Server resource utilization was sampled during each step using host-level process monitoring and Node.js runtime logs. The system sustained stable operation through 350 simultaneous users with 78 ms average API latency, 52% CPU usage, and 744 MB memory consumption. At 500 concurrent users, the system maintained all WebSocket sessions but showed graceful degradation through 108 ms average API latency and a 1.24% error rate, indicating that 500 users is an upper stress-test point rather than a zero-error production guarantee.

8. Battery Consumption Analysis

Power consumption telemetry gathered from 34 learner devices across four live excursions quantified the overhead of the background LocationTracker component. At the default 30-second position update cadence, incremental drain averaged 6.8% per hour compared with the device-reported baseline during non-tracking use. The overhead primarily came from high-accuracy location acquisition, network transmission of telemetry payloads, and periodic battery-status checks. To reduce device load, the client throttles location POST requests to 30-second intervals, suppresses duplicate transmissions when no meaningful movement is detected, and transmits compact payloads containing only latitude, longitude,

timestamp, batteryLevel, and isCharging. Browser execution constraints were also observed: background tabs may throttle JavaScript timers, iOS Safari imposes tighter restrictions on long-running background geolocation, and the Battery Status API is unavailable in some browsers [10]. Accordingly, IV Planner treats battery monitoring as an assistive risk signal rather than a guarantee of uninterrupted tracking. During the trials, seven low-battery events below 20% with isCharging=false were reported, allowing instructors to redistribute power banks before connectivity was lost.

9. MongoDB Query Performance

Table 9: MongoDB Query Performance: Indexed vs. Non-Indexed Comparison

Query Operation	Avg (Indexed, ms)	Avg (No Index, ms)	P95 (Indexed, ms)	Index Hit Rate	Coll. Scan
Attendance Lookup (user+trip+type)	9	84	18	100%	0%
Latest Location by Student+Trip	12	71	18	100%	0%
SOS Alerts by Trip (unresolved)	8	56	14	100%	0%
Digital Pass Validation Lookup	7	48	12	100%	0%
Trip Roster (students by trip)	14	92	24	100%	0%
Photo Album by Trip	18	110	34	100%	0%

The compound unique index on the Attendance collection keyed by {user, trip, type} delivers the most pronounced performance benefit: lookup latency falls from 84 ms under a full collection scan to 9 ms with an index hit, an 89.3% reduction [12]. These results describe the indexed dataset used in the deployment and should not be generalized to all MongoDB workloads without considering collection size, hardware, query shape, and cache state.

10. Comparison with Traditional Manual IV Management

Table 10: IV Planner vs. Traditional Manual Institutional Visit Management

Dimension	Traditional Manual Method	IV Planner	Improvement
SOS / Emergency Alert	Phone call chain (avg 4.2 min)	WebSocket dispatch (avg 310 ms)	Operational estimate: faster alert delivery
Attendance Recording	Verbal roll-call (avg 8.4 min / 60 students)	QR sweep (avg 62 sec / 60 students)	92.7% time reduction
Itinerary Distribution	Printed handouts; updates require reprinting	AI-generated; live announcements via app	Real-time, zero reprint cost
Budget Calculation	Manual spreadsheet; 45–90 min	AI auto-computed with buffer; < 5 sec	Draft generation; teacher review required
Student Location Awareness	Zero visibility; rely on student WhatsApp	Live Leaflet map; 30-sec updates; battery status	Live map-based visibility
Post-Trip Reporting	Manual compilation; 1–3 days	Auto-aggregated report; < 200 ms	Faster compiled records

Dimension	Traditional Manual Method	IV Planner	Improvement
Geofence Enforcement	None; manual headcount	Haversine auto-detection; breach alert in 480 ms	Automated breach detection introduced

The most pronounced operational contrast is observed in emergency response capability: the conventional telephone relay chain averaging 4.2 minutes (252,000 ms) versus IV Planner's 310 ms WebSocket-based dispatch represents an 812-fold reduction in alert-delivery time. This manual comparison should be read as an operational estimate. The attendance comparison distinguishes per-scan confirmation latency from full-cohort throughput: one QR scan confirms in approximately 620 ms, whereas movement, queueing, and instructor control produced an observed 62-second QR sweep for a 60-student cohort.

K. Ablation Study

Table 11: Ablation Study: Component-Specific Impact Metrics

Configuration	Primary Affected Metric	Full-System Value	Ablated Value / Impact
Full IV Planner (Socket.IO + Haversine + QR guard + JSON schema)	Reference configuration	Reference values reported in Tables III–VII	Reference configuration
Ablation 1: HTTP Polling (2s) replacing Socket.IO	SOS / geofence latency	310 ms / 480 ms	1,840 ms / 2,100 ms; unsuitable for SOS
Ablation 2: Bounding-Box Geofence (no Haversine)	Geofence false-positive rate	1.8%	8.4% near boundary corners
Ablation 3: No QR Duplicate Guard (no compound index)	Duplicate attendance rate	0% observed duplicate persistence	11.2% duplicate submissions accepted
Ablation 4: Generic LLM prompt (no JSON schema)	Proposal parse-failure rate	<3% with schema-constrained output	34% parse failures

Ablation 1, which substitutes HTTP polling at 2-second intervals for the Socket.IO event channel, increases SOS dispatch latency from 310 ms to 1,840 ms. Ablation 2, replacing circular Haversine geofencing with rectangular bounding-box containment, elevates the false-positive rate from 1.8% to 8.4% within the tested coordinate set. Ablation 3, disabling the MongoDB compound unique index on Attendance records, results in an 11.2% duplicate submission rate during the ablation scenarios. Ablation 4, removing the structured JSON validation constraint from LLM prompting, yields a 34% proposal parse-failure rate. These ablations report the metric directly affected by each removed component rather than implying that unrelated latency values should change.

11. User Satisfaction and Engagement

Five-point Likert satisfaction surveys administered post-deployment to N = 52 participants recorded mean scores of 4.6 ± 0.3 for Emergency Safety Confidence, 4.4 ± 0.4 for Instructor Dashboard Usability, 4.3 ± 0.3 for AI Itinerary Utility, 4.2 ± 0.5 for QR Attendance Convenience, and 4.5 ± 0.3 for Overall System Satisfaction. Faculty-specific Net Promoter Score (NPS) was computed at 72 (promoters: 78%, passives: 16%, detractors: 6%). Because the survey was administered within a single

institution after participants used a prototype system, these values are treated as adoption indicators rather than generalizable user-experience benchmarks.

12. Threats to Validity

The evaluation has six main validity threats. First, participants came from one institution, which limits external validity across different campuses, age groups, and administrative workflows. Second, some participants contributed observations across more than one deployment, so not all measurements are independent. Third, the manual baseline is an observed operational estimate, not an instrumented digital system; therefore, comparisons with Baseline A are descriptive only. Fourth, the geofence experiment validates boundary-classification behavior on coordinate pairs and distance-computation agreement, but it does not provide centimeter-grade physical ground truth for GPS accuracy. Fifth, some timing paths include client-side rendering and browser behavior, so results may vary across devices, operating systems, and permission states. Sixth, post-use satisfaction surveys may contain novelty and social-desirability bias. These threats are addressed by conservative claim wording, explicit provenance labels, and limiting strong comparative conclusions to measured digital baselines.

V. SECURITY AND ETHICAL CONSIDERATIONS

1. Threat Model and STRIDE Security Analysis

To characterize IV Planner's security posture, the STRIDE threat modeling methodology was applied, organizing potential risks across Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege [16]. Table XII enumerates the six primary threat vectors identified for IV Planner, together with their corresponding STRIDE classification and the specific countermeasure implemented to address each.

Table 12: STRIDE Threat Model and Implemented Mitigations

Threat Category	Attack Scenario	STRIDE Tag	Implemented Mitigation
Identity Spoofing	Adversary intercepts JWT to impersonate teacher role and resolve SOS alerts fraudulently.	Spoofing	Bcrypt password verification; role-embedded short-lived JWT access tokens; refresh-token rotation for continued sessions; server-side role and trip-membership re-validation on every protected REST route and Socket.IO handshake.
Location Tampering	Malicious student submits fabricated GPS coordinates to defeat geofence enforcement.	Tampering	Server-side Haversine re-validation; coordinate plausibility checks (velocity > 250 km/h flagged as anomalous).
SOS Flooding / DoS	Repeated SOS triggers to overwhelm teacher alert queue and obscure genuine emergencies.	DoS	3-second hold threshold; per-student rate limiting (max 3 triggers per 5 min); MongoDB TTL-indexed cooldown.

Threat Category	Attack Scenario	STRIDE Tag	Implemented Mitigation
Unauthorized Pass Validation	Student scans another student's Digital Pass QR to falsely mark them as present.	Elevation of Privilege	Pass validation endpoint restricted to Teacher role only; passCode is server-generated UUID v4; one-time isValidated flag.
Data Exfiltration	Interception of location telemetry or group chat messages in transit.	Info. Disclosure	TLS 1.3 for REST traffic and WSS for Socket.IO; role-scoped API responses; encrypted MongoDB storage for sensitive participant, trip, and location fields.
Photo Metadata Leakage	Exposed GPS-tagged photo EXIF data shows student home or sensitive locations.	Info. Disclosure	EXIF data stripped via Sharp library on server-side upload; GPS coordinates stored only in access-controlled MongoDB documents.

2. Student Data Privacy Framework

IV Planner follows data minimization for location-enabled educational systems. Positional records are retained as discrete timestamped coordinate snapshots rather than continuous movement trails, limiting reconstructible movement history to the configured update interval. REST API communication is transmitted over HTTPS, Socket.IO is promoted to WebSocket Secure (WSS) at handshake where available, and sensitive participant information is protected through role-scoped API access and encrypted database storage. GPS-tagged photo EXIF metadata is stripped server-side before storage, so sensitive device metadata is not unintentionally exposed in shared albums. Location coordinates associated with photos and telemetry remain available only to authorized trip participants and supervising staff.

3. Consent, Governance, and Institutional Oversight

Learners receive explicit notification of location monitoring activation upon trip enrollment, with a mandatory affirmative opt-in required prior to LocationTracker initialization [2]. Instructors retain the ability to disable location tracking for individual participants or for the complete trip group. Trip-associated location records are scheduled for deletion 90 days following the trip's conclusion date through a retention policy implemented with TTL-indexed records and server-side cleanup checks [13].

D. Ethical Use, Consent, and Submission Governance

Because the system processes real-time location, automated attendance, emergency alerts, battery status, and participant identity data, deployment requires explicit institutional consent and limited-purpose data use. Released datasets should be anonymized to remove names, emails, precise home-location traces, raw continuous trajectories, and other personally identifiable information. Any public release of logs or screenshots should also remove trip codes, student identifiers, access tokens, QR secrets, and exact sensitive coordinates.

Limitations and Future Work

The current implementation presents the following acknowledged limitations and future-work directions:

- Indoor Location Accuracy: The navigator.geolocation interface primarily depends on device, browser, sensor, satellite, and network-assisted positioning conditions [2].

Deployments within enclosed structures yield diminished spatial precision. Future development will evaluate indoor-positioning support as a separate, platform-dependent extension rather than assuming browser GPS is sufficient indoors.

- Single-Language AI Prompting: The current Gemini prompt schema supports English-language input and output. Multilingual prompt templates and regional-language generation will require additional validation rules, user testing, and safety checks before they can be treated as production-ready features.
- Offline Resilience: The current implementation requires network connectivity for safety-critical synchronization. Future work will evaluate a Progressive Web App (PWA) service-worker layer with an IndexedDB-backed event queue for non-critical events, while emergency alerts will continue to require explicit fallback procedures when connectivity is unavailable.
- Multi-Instructor Authorization: Excursions requiring multiple supervising instructors need a delegation authorization model. Future work will introduce a co-organizer role with scoped permissions for SOS resolution, attendance validation, and dashboard access.
- Scalability Ceiling at 500 Concurrent Users: The present deployment was tested up to 500 concurrent users on a single 4-core, 8 GB RAM server, where a 1.24% error rate was observed. Future work will evaluate horizontal scaling, load-balanced Socket.IO adapters, Redis-backed room coordination, and MongoDB scaling before claiming support for substantially larger cohorts.
- LLM Hallucination in Budget Estimates: The AI module can generate plausible but unsupported budget assumptions. A Retrieval-Augmented Generation (RAG) pipeline indexed against verified institutional expenditure records is planned to improve factual grounding, traceability, and budget-audit readiness; until then, generated budgets must remain teacher-reviewed drafts.
- Single-Institution Evidence: The present deployment was conducted within one institutional context. Multi-campus studies with larger and more diverse cohorts are needed before the observed usability and performance results can be generalized.
- Baseline Comparability: Manual workflows and digital baselines differ in instrumentation precision. Future studies should use matched experimental scripts, identical trip scenarios, and pre-registered measurement procedures for all baselines.
- Physical Ground Truth for Geofencing: The current geofence evaluation uses coordinate-pair classification and reference distance calculation. Future work should include surveyed boundary markers or high-accuracy reference devices to measure real-world GPS classification error directly.

VI. CONCLUSION

This paper has described IV Planner, a web platform developed to address the coordination and safety gaps that appear when institutional educational visits are managed with informal tools. The system ties together a Socket.IO real-time event layer, Haversine-based server-side geofencing, Google Gemini itinerary drafting with structured validation, dual-mode QR attendance, and GPS-tagged photo logging inside a single MERN-stack deployment. The core argument of the work is that combining these components is practically achievable and operationally useful for the educational-visit context. The field evaluation 52 participants, four live deployments over 30 days produced three main findings worth highlighting. First, the Socket.IO SOS pipeline delivered an end-to-end 4G latency of 310 ± 42 ms, comfortably outpacing the HTTP-polling alternative; the contrast with manual telephone chains is reported as an operational estimate only, not a controlled measurement. Second, requiring structured JSON output from Google Gemini and running a server-side validation pass before any trip record is written reduced malformed AI-generated proposals from 34% (unvalidated baseline) to under 3%. Third, the compound unique index on the Attendance collection blocked duplicate check-in records entirely during the concurrent scanning tests. Taken together, these results support the view that real-time WebSocket delivery, Haversine

geofence enforcement, database-level attendance guards, and structured LLM validation can work reliably in tandem under the reported prototype field-deployment conditions.

Acknowledgment

The authors thank the Department of Computer Applications, Guru Nanak Institute of Management Studies, Mumbai, for institutional support and computing infrastructure. The authors also thank the faculty participants and student volunteers whose contributions made the four live trip evaluations possible. AI-assisted tools were used only for language editing, formatting support, and editorial review; all technical claims, implementation details, experimental values, ethical approvals, and final manuscript decisions were reviewed and approved by the authors.

REFERENCES

1. D. A. Kolb, Experiential Learning: Experience as the Source of Learning and Development. Englewood Cliffs, NJ, USA: Prentice-Hall, 1984.
2. World Wide Web Consortium, "Geolocation," W3C Candidate Recommendation Snapshot, Mar. 26, 2026. [Online]. Available: https://www.w3.org/TR/geolocation/
3. I. Fette and A. Melnikov, "The WebSocket Protocol," IETF RFC 6455, Dec. 2011. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6455
4. Socket.IO, "Rooms," Socket.IO Documentation, 2026. [Online]. Available: https://socket.io/docs/v4/rooms/
5. Google AI for Developers, "Structured output," Gemini API Documentation, 2026. [Online]. Available: https://ai.google.dev/gemini-api/docs/structured-output
6. R. W. Sinnott, "Virtues of the Haversine," Sky & Telescope, vol. 68, no. 2, p. 159, 1984.
7. DENSO WAVE Inc., "What is a QR Code?" QRcode.com, 2026. [Online]. Available: https://www.qrcode.com/en/about/
8. M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," IETF RFC 7519, May 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7519
9. P. Leach, M. Mealling, and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace," IETF RFC 4122, Jul. 2005. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4122
10. MDN Web Docs, "Navigator: getBattery() method," Mozilla Developer Network, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getBattery
11. MDN Web Docs, "MediaDevices: getUserMedia() method," Mozilla Developer Network, 2026. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia
12. MongoDB, Inc., "Unique indexes," MongoDB Manual, 2026. [Online]. Available: https://www.mongodb.com/docs/manual/core/index-unique/
13. MongoDB, Inc., "TTL indexes," MongoDB Manual, 2026. [Online]. Available: https://www.mongodb.com/docs/manual/core/index-ttl/
14. Apache Software Foundation, "Apache JMeter User Manual," Apache JMeter Documentation, 2026. [Online]. Available: https://jmeter.apache.org/usermanual/
15. OWASP Foundation, "Application Security Verification Standard," OWASP, 2026. [Online]. Available: https://owasp.org/www-project-application-security-verification-standard/
16. F. Swiderski and W. Snyder, Threat Modeling. Redmond, WA, USA: Microsoft Press, 2004.