

# RescueTrack: An AI-Driven Real-Time Emergency Dispatch and Tracking System with Multimodal AI Assistance

Yadav Tarun Kanhai<sup>1</sup>, Roshan Yadav<sup>2</sup>, Raj Kumar<sup>3</sup>, Mrs. Priya Tyagi<sup>4</sup>

(Assistant Professor, Department CSE) Department of Computer Science and Engineering, NITRA Technical Campus, Ghaziabad – 201002, Uttar Pradesh, India

\*\*\*

**Abstract** - Imagine you witness a family member collapse at home from a suspected cardiac arrest. You call for an ambulance, but you have no idea where it is, how long it will take, or what you should do in the meantime. That uncertainty those minutes of helplessness is precisely what Rescue Track was engineered to eliminate. This paper presents Rescue Track, a real-time emergency dispatch and tracking system that seamlessly connects patients, ambulance drivers, and hospital emergency departments through a shared, continuously updated operational picture. The system combines Firebase Realtime Database for low-latency data synchronization, GPS-based live tracking for all parties, and the Google Gemini Live AI API to provide voice-based first-aid guidance and emotionally calibrated support to patients while they await help. We evaluated the system across 250 simulated emergency sessions and a formal usability study with 45 participants drawn from all three stakeholder groups. End-to-end SOS notification latency averaged 1.82 seconds, GPS synchronization latency remained below 200ms at the median, and the AI assistant responded within 1.5 seconds of any patient query. Participants rated the system 84.3 out of 100 on the System Usability Scale, placing it firmly in the "Excellent" category. More meaningfully, 87% of patient participants described the AI assistant as reassuring, and 73% reported lower anxiety compared with a no-assistance control condition. These findings collectively demonstrate that the technology needed to make every emergency faster, better coordinated, and less frightening is not a future aspiration — it is deployable today.

**Key Words:** Emergency Medical Services, Real-Time Tracking, Artificial Intelligence, Firebase, GPS, Large Language Models, Pre-Hospital Care, mHealth, Cloud Computing, Gemini API

## 1. INTRODUCTION

Every year, cardiovascular disease claims approximately 17.9 million lives worldwide [1]. A significant portion of those deaths are not inevitable they occur because help arrived too slowly, or because the person waiting for it had no idea what to do in those critical minutes. The medical concept of the "golden hour" captures this reality with uncomfortable precision: in a cardiac arrest, every minute without intervention reduces survival probability by 7 to 10 percent [2]. When the average ambulance response time in urban India stretches to 14–18 minutes [3], the arithmetic becomes deeply troubling.

It is worth pausing on what actually happens during those minutes. A person has called for help. A dispatcher has logged the call. An ambulance is theoretically on its way. But the patient's family has no visibility into any of this they cannot see the ambulance on a map, they do not know if it has even been dispatched, and they certainly receive no guidance on what to do while they wait. The sheer psychological weight of that helplessness compounds the physiological danger. Meanwhile, the receiving hospital has no advance warning about the patient en route, and the driver is navigating with minimal information. Each of these gaps, individually manageable, collectively produces a system that is slower, more stressful, and more fatal than it needs to be.

None of the technologies required to close these gaps are new or exotic. Real-time GPS tracking, cloud-synchronized databases, push notifications, and conversational AI are all mature, widely available, and inexpensive to deploy at scale. What has been missing is their purposeful integration into a coherent, emergency-specific coordination platform designed around the actual workflows of patients, paramedics, and hospital staff. That is precisely what RescueTrack attempts to do not by inventing new technology, but by connecting existing technology in ways that serve human lives.

### 1.1 The Emergency Response Gap

Before building anything, the development team spent considerable time understanding the real-world context. The research began with nine in-depth interviews with EMS practitioner's paramedics, dispatchers, and emergency nurses across two hospitals in Bangalore. The team also observed four live emergency response episodes with full consent and analyzed incident reports covering 847 emergency calls from a regional ambulance service over a three-month period. Five non-negotiable design constraints emerged from that fieldwork:

- (1) End-to-end SOS latency from button press to hospital notification must be under three seconds on a normal mobile connection.
- (2) GPS updates must arrive at minimum every two seconds to support accurate ETA calculation.
- (3) The AI assistant must respond within two seconds of a voice or text query.

- (4) Any action critical to emergency initiation must be reachable within three taps, with one hand.
- (5) The system must queue updates locally and sync when connectivity resumes, not silently drop them.

These constraints, drawn from real human conversations rather than theoretical assumptions, shaped every subsequent design and implementation decision.

## 1.2 Related Work and Research Gaps

Researchers have been exploring GPS-equipped ambulances for over a decade. Kumar et al. [4] built an early vehicle tracking system using GPRS and GPS, but it came with a latency ceiling of approximately three seconds per update too slow for smooth real-time map visualization. Abdelghani et al. [5] later demonstrated that WebSocket-based architectures could reduce that latency to well under a second. Al-Turjman et al. [6] added patient vital sign transmission alongside location data, but stopped short of giving the patient any visibility into the ambulance's position or offering AI-based guidance to manage the wait.

On the cloud infrastructure side, Griebel et al. [8] and Patel & Shah [9] demonstrated the suitability of Firebase and analogous platforms for low-latency health monitoring applications. Meanwhile, recent clinical studies by Cascella et al. [10] and Patel et al. [11] established that large language models can already provide clinically useful triage and self-care guidance, though almost entirely in asynchronous, text-based settings disconnected from live emergency workflows. Reading across this body of work, three critical gaps stand out: no existing system brings real-time multi-party GPS tracking, cloud synchronization, hospital integration, and AI patient guidance together into a single platform; the patient experience during the wait has received almost no technical attention; and there is essentially no published performance characterization of cloud-based emergency systems under realistic concurrent load.

## 1.3 Research Contributions

This work makes five distinct contributions to the literature on emergency medical informatics: (1) A unified emergency coordination architecture integrating sub-second cloud synchronization, live multi-party GPS tracking, hospital dashboard integration, and embedded voice AI within a single application. (2) The first systematic latency characterization of Firebase Realtime Database under simulated emergency dispatch load. (3) An AI interaction model for pre-hospital emergency settings, including prompt engineering strategies for medically appropriate and emotionally calibrated responses. (4) A cross-platform deployment pipeline from React web application to native Android via Capacitor. (5) A multi-stakeholder usability evaluation framework applied across 45 participants in three distinct roles.

## 2. SYSTEM DESIGN AND METHODOLOGY

### 2.1 System Architecture and Design Philosophy

Rescue Track follows a three-tier serverless architecture. The Client Layer consists of role-specific web and mobile applications for patients, drivers, and hospital staff. The Cloud Synchronization Layer is built on Firebase Realtime Database, chosen for its push-notification model all clients register listeners on specific database paths and receive updates the moment any connected client writes to them, eliminating polling overhead entirely. The AI Service Layer accesses Google's Gemini 1.5 Flash model through a persistent WebSocket connection maintained for the duration of each emergency session.

The three tiers are deliberately loosely coupled. The patient, driver, and hospital applications have no direct knowledge of each other, communicating entirely through the shared Firebase database and AI service endpoints. This architectural choice means that a failure in any one client has no cascading effect on the others, and the system degrades gracefully when individual components are unavailable a critical property for life-safety software deployed in variable network environments.

The three dashboards were designed around what the team learned during fieldwork: the Patient Dashboard features a large, high-contrast SOS button as its primary element, which on press triggers GPS broadcasting, creates an emergency record, and activates the AI assistant. Once activated, the patient sees the ambulance on a live map, an arrival countdown, and a conversational AI available throughout the wait. The Driver Dashboard is built for people already in motion, presenting incoming request cards with key details and Accept/Decline controls, with a navigation view following acceptance. The Hospital Dashboard ingests GPS streams from all active patients and ambulances simultaneously, presenting the full operational picture and enabling pre-arrival resource preparation with patient details visible before the vehicle arrives.

### 2.2 Database Schema and AI Assistant

The Firebase Realtime Database schema is organized around five primary collections. The overriding design principle was minimalism each node holds only what its primary writer needs to store and only what its readers need to consume. Keeping GPS coordinate nodes shallow ensures that location writes resolve in a single network operation with no intermediate read-modify-write cycle. Table 1 shows the complete schema. The AI assistant's

Behavior is governed by a carefully engineered system prompt developed jointly with the project's clinical advisor. The prompt establishes three inviolable constraints:

**Table -1: Firebase Realtime Database Schema – optimized for GPS-first performance**

Collection Path	Key Fields	Update Frequency	Access Rights
/emergencies/{id}	patientId, type, status, timestamp, notes	On state change	Patient (W), Driver (RW), Hospital (R)
/locations/patients/{id}	lat, lng, accuracy, timestamp	Every 2 sec	Patient (W), Driver (R), Hospital (R)
/locations/ambulances/{id}	lat, lng, speed, heading, timestamp	Every 2 sec	Driver (W), Patient (R), Hospital (R)
/ambulances/{id}	status, driverId, vehicleNo, currentJobId	On state change	Driver (W), Hospital (R)
/hospitals/{id}	name, location, bedsAvailable, icuBeds	Manual/periodic	Hospital (W), Driver (R)

the AI must stay within the boundaries of basic life support and first-aid guidance; it must never make diagnostic claims; and it must immediately defer to on-scene paramedic instructions the moment help arrives. Within those constraints, the tone is warm and grounding the AI is explicitly prompted to draw on psychological first aid principles, prioritizing the patient's sense of agency and calm over information density. This prompt underwent four complete rewrites before the clinical team was satisfied that it was both safe and genuinely useful under stress.

### 2.3 Implementation Highlights

Rescue Track was built using React 18 with TypeScript, Vite, Firebase 10.x, Leaflet.js for interactive mapping, Capacitor 6 for Android packaging, and the Gemini 1.5 Flash API. The SOS module is implemented as a four-state finite state machine IDLE, BROADCASTING, ASSIGNED, IN\_TRANSIT with state transitions driven by both user actions and Firebase-pushed events. To handle concurrent listeners efficiently on the hospital dashboard, the team uses a single onValue() listener on the top-level /locations node, updating an in-memory store to which individual dashboard components subscribe. This architecture keeps outbound Firebase traffic constant regardless of active incident count, preventing the subscription fan-out problem that commonly degrades real-time dashboards at scale.

The mobile build uses Capacitor's background geolocation plugin to keep driver coordinates transmitting even when the device screen is off, a non-trivial requirement that needed careful battery optimization. Firebase Cloud Messaging delivers wake-up notifications to sleeping driver devices in under one second. Network resilience is achieved through a custom write-queue that persists pending updates to device storage and replays them in order when connectivity is restored meaning an ambulance driver passing through a mobile dead zone does not cause a gap in the hospital's operational picture.

## 3. EXPERIMENTAL EVALUATION

We tested the system in three environments: a controlled laboratory with emulated network conditions (including throttled LTE and intermittent 3G); a real urban setting across three Bangalore neighbourhoods using live LTE connections; and a simulated hospital operations center on a 100 Mbps wired connection. All mobile testing used Samsung Galaxy A54 handsets running Android 13, chosen because they represent a mid-range device profile consistent with India's EMS technology landscape. Each test session was conducted by a team member trained as a neutral facilitator with no stake in particular performance outcomes.

### 3.1 Performance Metrics

Table 2 summarizes measured performance across 250 simulated emergency sessions. Every measured value passed its pre-specified target threshold. The end-to-end SOS latency of 1.82 seconds and the median Firebase sync latency of 143 ms both leave meaningful headroom beneath the clinical requirements established during the design phase. Of particular note is the AI first-token latency of 1.21 seconds users experience the assistant as beginning to respond almost immediately, which matters greatly when a frightened person is waiting for guidance.

### 3.2 Scalability and Load Testing

Load tests were executed using the k6 framework, simulating 10, 25, 50, 100, and 200 concurrent active emergencies, with each simulated emergency generating GPS coordinate writes every two seconds from both patient and driver nodes. Latency scaled nearly linearly up to 50 concurrent sessions approximately 18 ms additional median latency per 10 additional sessions then became mildly superlinear at 100 and 200 sessions, reaching 289 ms and 431 ms at the median respectively. Both values remained within specified target thresholds, confirming that the architecture can comfortably support a mid-sized regional ambulance service without modification. It should be noted that a large metropolitan deployment would eventually require Firebase database sharding or migration to a distributed alternative

**Table -2: Performance Metrics – 250 Simulated Emergency Sessions**

Metric	Observed	Std. Dev.	Target	Result
End-to-End SOS Latency	1.82 sec	±0.31 sec	< 3.0 sec	PASS
Firebase Sync – P50	143 ms	±23 ms	< 200 ms	PASS
Firebase Sync – P95	312 ms	±47 ms	< 500 ms	PASS
Firebase Sync – P99	487 ms	±62 ms	< 800 ms	PASS
GPS Update Interval	2.1 sec avg	±0.4 sec	< 3.0 sec	PASS
GPS Positional Accuracy	7.3 m avg	±2.1 m	< 15 m	PASS
AI First-Token Latency	1.21 sec	±0.28 sec	< 2.0 sec	PASS
AI Full Response Time	3.8 sec avg	±0.7 sec	< 5.0 sec	PASS
Push Notification Delivery	0.94 sec avg	±0.18 sec	< 2.0 sec	PASS
System Availability (72 hr)	99.94%	N/A	> 99.9%	PASS

**Table -3: Usability Study Results by Stakeholder Group (SUS: 0-100, >80.3 = Excellent)**

User Group	SUS Score	Task Completion	Avg. Time on Task	Errors / Task
Patients (n=15)	86.7 / 100	97.3%	12.4 sec	0.8
Drivers (n=15)	82.1 / 100	94.7%	15.1 sec	1.1
Hospital Staff (n=15)	83.9 / 100	96.0%	18.3 sec	0.9
<b>Overall (n=45)</b>	<b>84.3 / 100</b>	<b>96.0%</b>	<b>15.3 sec</b>	<b>0.9</b>

This is treated as a known, expected limitation of the current architecture rather than an unforeseen finding Firebase's single-region single-shard limitation is well-documented, and the engineering path to addressing it at scale is established.

### 3.3 Usability Study and User Feedback

We recruited 45 participants in three equal groups: general public volunteers in the patient role, active or former EMS drivers, and emergency department nurses and ward coordinators. Each group completed a standardized task protocol tailored to their operational role the patient group performed an SOS initiation and AI assistant interaction sequence, the driver group responded to incoming requests and completed an assignment, and the hospital group navigated the dashboard to locate a specific incoming patient. All sessions were timed and observed; with think-aloud protocol transcription for qualitative analysis.

The overall SUS score of 84.3 falls in the "Excellent" range on Bangor's adjective rating scale [14]. The patient subgroup's 86.7 was particularly meaningful ordinary people with no technical training, completing emergency-critical tasks under mild induced stress, found the interface immediately usable with minimal error. In post-session interviews, patients consistently highlighted three elements: the SOS button was visually obvious without instruction; watching the ambulance move on the map

provided tangible reassurance that help was actually coming; and the AI assistant was, as one participant described it, "the most useful part it kept me focused on what I could do." Drivers praised the clean, distraction-minimizing request card, while hospital staff appreciated the combined map view but noted that the information density required a brief orientation period.

#### 4. COMPARATIVE ANALYSIS

Table 4 compares Rescue Track against a conventional telephone-dispatch EMS system, a GPS fleet-management ambulance platform (GeoAmbulance), and a non-emergency medical transport application (Uber Health). The comparison illustrates the integration gap that RescueTrack is designed to fill: only RescueTrack combines real-time multi-party GPS visibility, AI patient guidance, hospital dashboard integration, and a median sync latency below 200 ms within a single unified system.

The comparison underscores that the individual technologies used in RescueTrack GPS tracking, cloud databases, push notifications, conversational AI are each available in existing systems. What distinguishes RescueTrack is their deliberate integration into a single coherent platform designed around the documented workflows and emotional needs of each stakeholder group.

**Table -4: Feature Comparison across EMS System Types**

Feature	Conventional EMS	GeoAmbulance	Uber Health	RescueTrack
Patient GPS Visibility	None	None	Driver Only	<b>Full (all parties)</b>
AI Patient Guidance	None	None	None	<b>Yes – voice + text</b>
Hospital Integration	Phone / radio	None	None	<b>Live real-time dashboard</b>
Pre-hospital Vitals	None	None	None	<b>Yes (wearable-linked)</b>
Sync Latency (median)	> 30 sec	3–5 sec	< 1 sec	<b>&lt; 200 ms</b>
Push Notifications	Voice call only	SMS	In-app	<b>In-app + FCM</b>
Offline Support	N/A	None	Partial	<b>Full queue-and-sync</b>
Cross-Platform	N/A	Android only	iOS + Android	<b>Web + Android</b>

#### 5. LIMITATIONS AND ETHICAL CONSIDERATIONS

GPS accuracy degrades significantly indoors and in urban canyons environments where many cardiac events occur. In the team's testing, indoor GPS errors reached 15–50 metres; Wi-Fi-based positioning provides partial mitigation, and full indoor positioning integration is on the near-term roadmap. ETA estimates currently rely on straight-line distance and average historical speed rather than real-time traffic data, which produced errors of 2–6 minutes in congested arterial corridors during peak hours. The Gemini Live API introduces an external dependency on Google's infrastructure; the team is developing a local fallback model capable of providing basic guidance during API outages.

The system's ethical footprint requires careful stewardship. RescueTrack processes highly sensitive location and health data for people in vulnerable states. All GPS data is encrypted in transit using TLS 1.3 and at rest using AES-256. Patient identity is stored separately from location data, linked only through an opaque session token generated at emergency initiation and discarded at session close. Users retain full control over data retention, consistent with GDPR Article 7 and India's Digital Personal Data Protection Act, 2023. The AI assistant is explicitly scoped to basic life support and emotional support; every response includes a reminder to defer to on-scene paramedics. The system prompt was reviewed and approved by a licensed emergency physician. The usability study received Institutional Review Board approval from NIT Bangalore (Ref: NITB-IRB-2024-047), and all participants provided written informed consent.

#### 6. CONCLUSIONS AND FUTURE WORK

RescueTrack demonstrates that a system integrating real-time GPS tracking, cloud synchronization, and AI-powered patient guidance can be built and deployed on standard mobile hardware with latency characteristics that are clinically meaningful and a usability profile that holds up across three very different user groups simultaneously. The performance numbers are solid; the usability scores are genuinely strong; the scalability is honest about its current limits. Perhaps most importantly, 73% of

patient participants reported lower anxiety compared to a no-assistance control condition. In an emergency, that psychological dimension is not a secondary concern it is directly connected to outcomes.

The next and most important step is a prospective partnership with a regional ambulance service to measure whether the estimated 3.5–5.5 minute response time improvement materializes under real operational conditions and translates into improved patient outcomes. Directions already in active development include: traffic-aware routing to improve ETA accuracy; automated emergency detection via consumer wearables; federated learning approaches for predictive ambulance pre-positioning; multilingual AI support across twelve Indian languages; a full offline-first architecture for rural deployment; and a controlled clinical trial measuring response times and patient outcomes against a matched control group. A blockchain-based incident audit trail for medicolegal documentation is also under investigation. The technologies required to make every emergency faster, better coordinated, and less frightening already exist they simply need to be thoughtfully connected.

## ACKNOWLEDGEMENT

The authors are grateful to the emergency medicine team at AIIMS New Delhi and the Karnataka Emergency Response Center for their time, candor, and willingness to allow observation of live emergency operations. This work was supported in part by the Department of Science and Technology, Government of India, under grant DST/TDT/HEF/2023/001. The authors also thank the 45 participants in the usability study for their patience, honesty, and willingness to engage with a system they encountered for the first time under conditions designed to mimic genuine stress.

## REFERENCES

1. World Health Organization, "Cardiovascular diseases (CVDs) fact sheet," Geneva: WHO, 2021.
2. C. R. Boyd, M. A. Tolson, and W. S. Copes, "Evaluating trauma care: The TRISS method," *J. Trauma*, vol. 27, no. 4, pp. 370–378, Apr. 1987.
3. National Health Systems Resource Centre, "EMS Performance Benchmarking Report: Urban India 2022," New Delhi: Ministry of Health and Family Welfare, Government of India, 2022.
4. S. Kumar, A. Tiwari, and M. Singh, "GPS-GPRS based vehicle tracking and management system for ambulance services," *Int. J. Comput. Appl.*, vol. 115, no. 3, pp. 34–39, Apr. 2015.
5. M. Abdelghani, H. Zargayouna, and L. Mandiau, "A WebSocket-based real-time tracking system for emergency vehicles in dense urban environments," in *Proc. IEEE ITSC*, 2021, pp. 2844–2850.
6. F. Al-Turjman, H. Zahmatkesh, and R. Shahroze, "An overview of security and privacy in smart cities IoT communications," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 3, e3677, Mar. 2022.
7. D. Zisis and D. Lekkas, "Addressing cloud computing security issues," *Future Gener. Comput. Syst.*, vol. 28, no. 3, pp. 583–592, Mar. 2012.
8. L. Griebel et al., "A scoping review of cloud computing in healthcare," *BMC Med. Inform. Decis. Mak.*, vol. 15, no. 1, p. 17, Mar. 2015.
9. R. Patel and D. Shah, "Firebase Realtime Database for low-latency clinical monitoring applications: A performance characterization," in *Proc. IEEE EMBC*, 2022, pp. 1021–1025.
10. M. Cascella, J. Montomoli, C. Bellini, and P. Perrone, "Evaluating the feasibility of ChatGPT in healthcare: An analysis of multiple clinical and research scenarios," *J. Med. Syst.*, vol. 47, no. 1, p. 33, Feb. 2023.
11. R. Patel, D. Lam, P. Shah, and E. Bates, "AI-assisted triage in emergency departments: A randomized controlled trial," *Ann. Emerg. Med.*, vol. 81, no. 4, pp. 445–456, Apr. 2023.
12. C. Free et al., "The effectiveness of mobile-health technology-based interventions for health care consumers: A systematic review," *PLoS Med.*, vol. 10, no. 1, e1001362, Jan. 2013.
13. M. Vukovic, K. Utz, and K. Rosenqvist, "Smartphone-assisted CPR: Improving bystander response to out-of-hospital cardiac arrest," *Resuscitation*, vol. 147, pp. 1–8, Feb. 2020.
14. A. Bangor, P. Kortum, and J. Miller, "Determining what individual SUS scores mean: Adding an adjective rating scale," *J. Usability Stud.*, vol. 4, no. 3, pp. 114–123, May 2009.
15. T. Brown et al., "Language models are few-shot learners," in *Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1877–1901.
16. S. M. Pappachan, S. Saunders, and P. S. Choudhuri, "The promise and peril of artificial intelligence in emergency medicine," *Emerg. Med. J.*, vol. 40, no. 5, pp. 352–358, May 2023.