

EMERGENCY SERVICES

An Integrated Emergency Ambulance Response, Tracking and Hospital Coordination System

Priyank Nagori, Abhijeet Raskar, Rutuja Patil, Sachin Rathod, Ashwini Garkhedkar

Department of MCA, Progressive Education Society's Modern College of Engineering, Pune, Maharashtra, India

Published October 2026

Abstract

Emergency medical response is time-sensitive, and delays in locating the patient, assigning an ambulance, and coordinating with a receiving hospital can directly affect patient outcomes. This paper presents RAAH Connect, an integrated emergency ambulance response, tracking, and hospital coordination prototype. The system comprises four purpose-built components: a Patient Android application (Kotlin/Jetpack Compose) for SOS initiation and real-time ambulance tracking on a live map; a Driver Android application (Kotlin/Jetpack Compose) for emergency offer management, step-by-step dispatch workflow, foreground GPS telemetry, and siren-alert notifications; a Dispatcher web dashboard (React 19/TypeScript/Vite) for fleet monitoring, emergency coordination, and hub management; and a trusted server-authoritative backend of Firebase Cloud Functions (Node.js) that handles nearest-driver ranking, atomic claim/decline transactions, and cascade dispatch. Google Firebase (Firestore, Realtime Database, Authentication, Cloud Messaging) provides the shared cloud data layer. The prototype demonstrates an end-to-end emergency workflow from patient SOS to ambulance arrival and hospital coordination in a single integrated digital system. System limitations and directions for future work are discussed.

Keywords: Ambulance Tracking; Android; Cloud Functions; Emergency Dispatch; Emergency Medical Services; Firebase; Foreground Service; Hospital Coordination; Jetpack Compose; Location-Based Services; React; Real-Time Database

1 Introduction

Emergency medical services provide immediate medical assistance, transport patients to appropriate receiving facilities, and coordinate across multiple actors in a compressed time window. The timeliness and quality of that coordination directly influence patient outcomes. Digital platforms that connect the patient, the ambulance crew, a dispatcher, and the receiving hospital through a shared information layer can reduce communication delays and provide each actor with current, consistent situational awareness.

Ambulance operations become difficult when emergency information is spread across telephone calls, manual coordination, and disconnected records. A driver who does not have the patient's precise location, or a dispatcher who cannot see which ambulance is nearest, must rely on voice communication and manual judgement. A common digital record, updated in real time, replaces that manual exchange with structured, queryable data visible to every role.

Locating the patient correctly during an emergency is another practical problem. A patient may not be able to give a complete address under stress, and a verbal description is difficult to relay accurately. The RAAH Connect Patient application captures the device GPS position at the moment the SOS is raised and stores it in a structured record, giving the dispatcher and driver a coordinate-based reference rather than a verbal description. The driver application renders that coordinate on a

live map using Google Maps and launches turn-by-turn navigation directly from the dispatch workflow.

The proposed system addresses these coordination needs through a common digital workflow involving the patient, dispatcher, ambulance driver, and hospital coordination component. The RAAH Connect prototype integrates four purpose-built applications sharing a single Firebase cloud data layer: a Patient Android app, a Driver Android app, a Dispatcher web dashboard, and a trusted backend of Firebase Cloud Functions. Each component operates on the same Firestore emergency record, so status transitions made by one actor are immediately visible to all others.

2 Literature survey

2.1 World Health Organization ambulance operational guidance (2025)

The World Health Organization's operational guidance for ambulance systems names emergency communication and dispatch centres as core components of prehospital emergency care. It notes that ambulance systems need organized communication, dispatch processes, trained personnel, transport capability, and coordination with the wider health system [1]. This supports the need for a structured digital workflow in which emergency requests are captured, communicated, and coordinated, instead of depending entirely on unstructured manual communication.

2.2 World Health Organization Emergency Care System Framework (2018)

The WHO Emergency Care System Framework gives a broader view of emergency care and describes functions at the scene of illness or injury, during transport, and in emergency-unit and early inpatient care [2]. It is relevant to this work because it shows that emergency care does not end with ambulance transport. Coordination between prehospital and facility-based care is part of the overall system.

For a digital emergency platform, this means that information collected when the emergency is reported should stay useful to the later stakeholders. The present system therefore treats patient request information, ambulance response information, and hospital-related coordination as connected stages of one workflow.

2.3 Shetab-Boushehri et al. (2022)

Shetab-Boushehri, Rajabi, and Mahmoudi studied the location of emergency medical service stations, ambulance allocation, and routing, taking recurrent traffic congestion and variability in emergency events into account [3]. Their work shows how complicated it is to decide where emergency resources should be placed and how ambulances should be routed when demand and travel conditions change.

The study considered unpredictable emergency calls, ambulance availability, and changing traffic conditions. The researchers formulated the location and routing decisions and used optimization and heuristic approaches to obtain workable solutions. This literature is relevant to RAAH because it shows why ambulance availability and location matter operationally, although the current RAAH prototype does not implement the optimization algorithms described in the paper.

The contribution of this reference to the present study is therefore conceptual: ambulance response needs shared resource information and location-aware decisions. RAAH provides the digital coordin-

ation layer in which emergency requests and ambulance states can be shared among the system participants.

2.4 Chou, Chiang and Chen (2022)

Chou, Chiang, and Chen studied emergency medical response in mass casualty incidents, considering traffic congestion near the incident site and delays at hospitals [4]. Their work shows that ambulance response cannot be judged by travel distance alone, since traffic conditions and delays at the receiving hospital also affect emergency performance.

The authors used traffic-flow modelling and optimization methods to examine emergency response under congestion and hospital delays. Their results show that both transportation and healthcare-facility conditions need to be considered in emergency planning. In the present work, this is the reason hospital coordination is included as a separate stakeholder and ambulance transport is not treated as an isolated process.

The mathematical traffic and optimization models used in that study are not implemented in the current RAAH prototype. They only provide a research basis for possible future routing and decision-support extensions.

2.5 Hospital selection research (2014)

Lee examined the role of hospital selection in ambulance logistics and discussed how the choice of destination hospital can influence emergency transport decisions [5]. Hospital selection may depend on travel distance, treatment capability, waiting conditions, and available capacity. Choosing a hospital is therefore not only a geographic problem, because the destination must also suit the patient's emergency needs.

The study supports including hospital information and coordination in an emergency-response system. For this reason RAAH has a dispatcher-side hospital coordination interface and a separate hospital-side dashboard concept. The current prototype demonstrates hospital information and coordination interfaces, while automated hospital ranking or capacity-based selection logic would need further verification and development.

Hospital coordination is part of the intended workflow, but the system should not be described as implementing a validated medical decision algorithm for hospital selection.

2.6 Emergency vehicle traffic signal control (2019)

Cao, Shuai, and Li investigated emergency-vehicle-centered traffic signal control in intelligent transportation systems [6]. Their work deals with reducing delay for emergency vehicles at signalized intersections and shows how infrastructure-level traffic control can support emergency response.

The study is related to the broader problem because ambulance travel time is affected by intersections and traffic conditions. However, traffic-signal control requires integration with road infrastructure and signal controllers. This is outside the verified implementation scope of the current RAAH platform and is treated as a possible future extension.

The literature suggests that digital ambulance coordination could later be combined with intelligent transportation infrastructure, but RAAH does not perform traffic-signal control in its present prototype.

2.7 IoT-based smart signal and emergency vehicle priority (2024)

Kanimozhi, Kishore Reddy, and Rahul Kumar presented an IoT-based smart traffic signal approach for traffic-density management and emergency-vehicle priority [7]. The study describes vehicle detection, traffic-density information, and signal adaptation to give priority to emergency vehicles.

This work shows one way in which an ambulance-response platform could later interact with external transportation systems. The current RAAH system does not include AI-based vehicle counting, IoT traffic-signal hardware, or direct signal control.

Overall, the reviewed literature covers emergency-care coordination, ambulance allocation and routing, hospital selection, and traffic infrastructure. RAAH addresses the gap between these operational participants at the application and cloud-coordination level.

3 Research gap

The literature shows that several individual aspects of emergency response have been studied in detail, including ambulance allocation, route optimization, hospital selection, emergency-care organization, and traffic-signal priority [1]–[7]. These studies show that emergency response involves both healthcare and transportation considerations.

From an end-to-end operational view, however, the emergency workflow can still be fragmented across the patient requesting help, the dispatcher coordinating resources, the ambulance driver accepting and completing the response, and the hospital preparing to receive the patient. A system may provide location information without structured ambulance acceptance, or support ambulance operations without keeping a common emergency-request record.

The research gap addressed by this work is the integration of emergency request initiation, location capture, ambulance-driver coordination, dispatcher monitoring, shared emergency information, and hospital coordination in a common mobile and cloud-based workflow. The contribution is not a new routing or traffic-control algorithm. It is an integrated prototype that brings these stakeholders into one structured emergency-response process.

4 System architecture

The system architecture is built around four primary actors: the patient, the dispatcher, the ambulance driver, and the hospital-side coordinator. Five technical components serve these actors: the Patient Android application, the Driver Android application, the Dispatcher web dashboard, the Firebase/Cloud data layer, and the hospital coordination interface. All components share a single Firebase project (project ID: raahconnect-5) and operate on the same Firestore collections and Realtime Database paths.

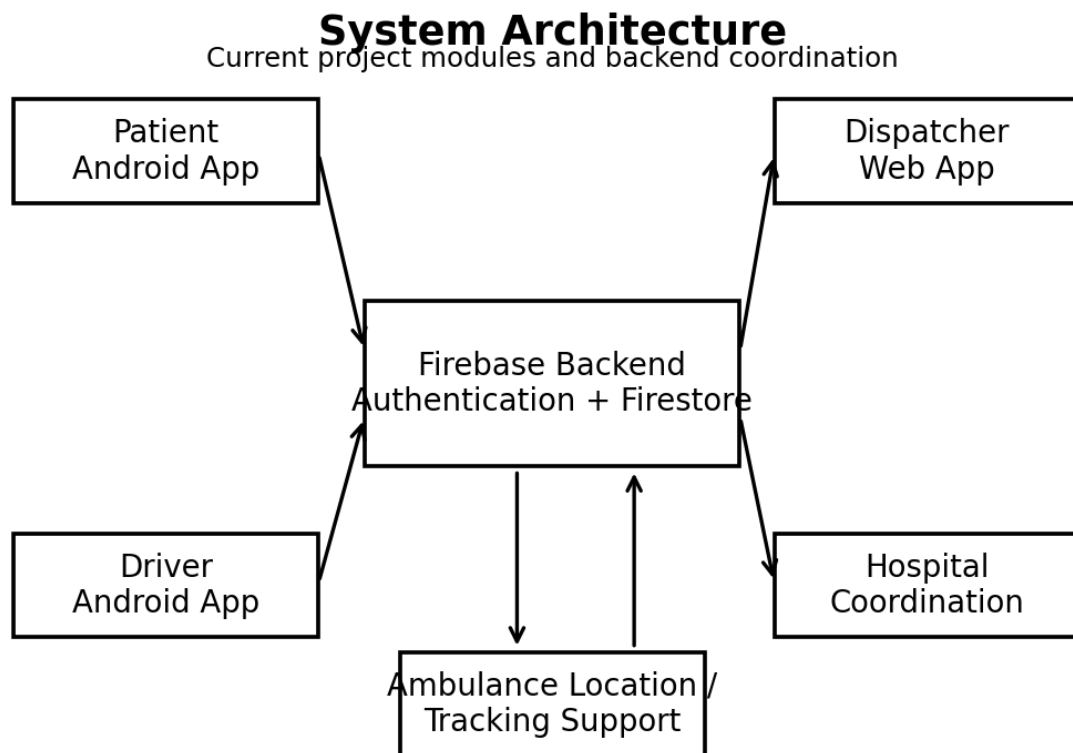


Figure 1. System architecture of the proposed emergency response platform.

The responsibilities of each layer are summarised in Table 1.

Table 1. Major layers and responsibilities of the proposed RAAH Connect architecture.

Module	Main Responsibility
Patient Application	Authentication (Firebase Auth), SOS initiation with emergency-type selection (SELF / OTHER), one-time GPS capture via FusedLocationProviderClient, hospital pre-selection from Firestore, emergency-request creation in Firestore (emergencyRequests collection), and real-time ambulance tracking on a live Google Maps Compose surface with ambulance-location polling from Firebase Realtime Database (ambulanceLocations/{ambulanceId}).
Firebase / Cloud Data Layer	Shared Firestore collections (users, ambulances, emergencyRequests, hospitals, hubs, assignments) and Firebase Realtime Database paths (ambulanceLocations/{ambulanceId}) used by all components. Firebase Authentication (email/password) manages identity. Firebase Cloud Messaging delivers high-priority siren-alert push notifications to driver devices. Firestore Security Rules enforce role-based access at the document level.
Dispatcher Web Application	React 19 / TypeScript / Vite single-page web application. Provides dispatcher authentication, real-time emergency-request monitoring via Firestore listeners, fleet and hub management views, live ambulance positions on a MapLibre GL map, and manual dispatch override controls. Communicates with Firestore using the Firebase JS SDK v12.
Ambulance Driver Application	Android application (Kotlin / Jetpack Compose, minSdk 24, targetSdk 37, Firebase BOM 34.19.0). Supports driver login and registration with Hub Admin approval workflow, daily ambulance selection, online/offline toggle, emergency offer accept/decline, full step-by-step dispatch workflow (EN_ROUTE_TO_PATIENT to ARRIVED_AT_PATIENT to PATIENT_ONBOARD to EN_ROUTE_TO_HOSPITAL to ARRIVED_AT_HOSPITAL to COMPLETED), foreground GPS telemetry service (DriverLocationService writing to Realtime Database every 2.5 s), and siren-alert FCM push notifications with screen-wake/keyguard-dismiss for background alerts.

Module	Main Responsibility
Hospital Coordination	Hospital information stored in Firestore (hospitals collection). Dispatcher views hospital data and can associate a hospital with an active emergency. A separate hospital-side interface displays incoming emergency details, assigned ambulance, patient information, and current ambulance status for receiving-facility situational awareness.

4.1 Patient application layer

The patient application is the entry point for an emergency. The implemented application supports registration and login via Firebase Authentication, a home screen with quick-access SOS and profile navigation, and a dedicated SOS screen where the patient selects the emergency type (for self or for another person), optionally provides brief context, and chooses a hospital. On submission, the app captures the current device GPS coordinate using FusedLocationProviderClient and writes a new emergency-request document to the Firestore emergencyRequests collection. After submission, the patient is shown a live tracking screen: a Google Maps Compose surface that polls Firebase Realtime Database for the assigned ambulance GPS position and renders it as a moving marker, giving the patient real-time visibility of the incoming ambulance. Voice notifications and status-change alerts are implemented through a VoiceNotificationManager that uses Android TextToSpeech to narrate key status transitions such as ambulance acceptance and arrival.

4.2 Emergency coordination layer

The coordination layer holds the shared emergency-request record used by all participating components. An emergency record in Firestore progresses through a well-defined status lifecycle: REQUESTED, OFFERED, ACCEPTED, EN_ROUTE_TO_PATIENT, ARRIVED_AT_PATIENT, PATIENT_ONBOARD, EN_ROUTE_TO_HOSPITAL, ARRIVED_AT_HOSPITAL, COMPLETED, or CANCELLED. A server-authoritative Firebase Cloud Function (onEmergencyCreated) triggers on new document creation, queries all AVAILABLE ambulances, ranks them by Haversine distance from the patient GPS coordinate, and writes the nearest ambulance as targetAmbulanceId together with a candidateAmbulanceIds list for cascaded dispatch. Atomic claim and decline operations are implemented as Firebase HTTPS Callable functions (claimEmergency, declineEmergency, expireOfferAndCascade) using Firestore transactions to prevent double-acceptance race conditions. A markPatientPickedUp callable function validates that only the assigned driver can advance the status to PATIENT_PICKED_UP.

4.3 Location and routing layer

Location tracking is implemented as a persistent Android Foreground Service (DriverLocationService) running on the driver device. The service uses FusedLocationProviderClient with PRIORITY_HIGH_ACCURACY and a 2,500 ms update interval (minimum 1,500 ms) to obtain continuous GPS fixes. Each location fix is written to Firebase Realtime Database at ambulanceLocations/{ambulanceId} with fields latitude, longitude, speed, heading, bearing, accuracy, timestamp, driverId, and emergencyId. The fix also updates the corresponding Firestore ambulances/{ambulanceId} document for dispatcher and hospital-side map visibility. The service persists through app backgrounding and external navigation launches using START_STICKY and a low-importance foreground notification. The AmbulanceLocationTracker component exposes a StateFlow of AmbulanceLocation. The Patient app reads the same Realtime Database node to render the ambulance marker on its tracking map. Navigation from the driver app to patient or hospital is launched via a Google Maps intent with the target coordinate.

4.4 Ambulance service layer

The ambulance-driver application is implemented in Kotlin with Jetpack Compose (Material 3) targeting Android SDK 37 (minimum SDK 24). Navigation is managed by a sealed AppScreen state machine in MainActivity. After login and Hub Admin approval (PENDING to AUTHORIZED), the driver selects a daily ambulance from the hub fleet. The DashboardScreen and DashboardViewModel (AndroidViewModel) coordinate the driver session: an online/offline toggle writes the ambulance status (AVAILABLE or OFFLINE) to Firestore; incoming emergency offers arrive via a Firestore real-time listener and trigger a siren-alert notification (EmergencyAlertNotificationHelper playing a custom siren sound on a high-priority notification channel); the driver accepts or declines the offer, and on acceptance the full dispatch step-by-step workflow is presented with action buttons for each status transition. The DriverLocationService foreground service is started when the driver goes online and stopped when the shift ends. A Hub Admin screen (HubAdminScreen) provides fleet management: adding ambulances, approving or rejecting driver registrations, and monitoring today assignments.

4.5 Hospital coordination layer

The hospital coordination component stores hospital records in the Firestore hospitals collection (fields: hospitalId, name, address, latitude, longitude, phone, capacity). The Patient application lists available hospitals from this collection when the patient raises an SOS and allows pre-selection of the destination. The Dispatcher web dashboard shows hospital information alongside fleet data and allows the dispatcher to associate a hospital with an active emergency record (writing hospitalId, hospitalName, hospitalLat, hospitalLng to the emergencyRequests document). The hospital-side interface displays the incoming emergency details, the assigned ambulance identifier and vehicle number, current dispatch status, and the patient basic information, giving the receiving facility preparation time before ambulance arrival.

5 Working of the system

The workflow begins when a patient logs in to the Android application and initiates an SOS. The patient selects the emergency type, optionally chooses a hospital, and submits the request. The application captures the device GPS coordinate via FusedLocationProviderClient and writes the emergency-request document to the Firestore emergencyRequests collection with status REQUESTED. The Firebase Cloud Function onEmergencyCreated triggers immediately, queries all AVAILABLE ambulances, ranks them by Haversine distance, and updates the document with targetAmbulanceId, candidateAmbulanceIds, and dispatchStatus OFFERED. A high-priority FCM push notification with a siren sound is sent to the targeted driver.

The dispatcher web application (React 19 / TypeScript, served via Vite) monitors emergency requests and fleet data through Firestore real-time listeners and displays ambulance positions on a MapLibre GL live map. The ambulance-driver application receives the FCM push notification and plays a siren alert even when backgrounded. The driver sees the incoming offer on the DashboardScreen and can accept or decline. If the driver accepts, the claimEmergency Cloud Function runs atomically: it verifies the status is OFFERED or REQUESTED, confirms the caller is the targeted driver, updates the emergency record to ACCEPTED, assigns ambulanceId and driverId, and sets the ambulance status to BUSY. If declined, declineEmergency cascades the offer to the next nearest candidate.

After acceptance, the driver follows the step-by-step workflow: tapping En Route to Patient sets status EN_ROUTE_TO_PATIENT; Arrived at Patient sets ARRIVED_AT_PATIENT; Patient Onboard calls

the `markPatientPickedUp` Cloud Function (`PATIENT_ONBOARD`); En Route to Hospital sets `EN_ROUTE_TO_HOSPITAL`; and Arrived at Hospital sets `ARRIVED_AT_HOSPITAL` before final completion. Throughout the active emergency, `DriverLocationService` writes GPS coordinates to Firebase Realtime Database every 2.5 seconds. The patient tracking screen polls the same path and renders the ambulance marker moving on a live Google Maps surface. The dispatcher live map also displays the ambulance position.

Hospital coordination runs in parallel: the emergency record carries `hospitalId` and hospital coordinates, which the driver application uses to launch Google Maps navigation to the hospital after patient pickup. The hospital-side interface reads the emergency document and displays incoming ambulance details, patient information, and current status to allow the receiving facility to prepare.

The response is complete when the driver taps Complete Emergency. The emergency record is updated to `COMPLETED`, the ambulance status returns to `AVAILABLE`, and the driver is presented with a summary. The driver may then start a new shift cycle or end the shift.

5.1 Conceptual emergency response flow

Patient SOS → Firestore `emergencyRequests` (`REQUESTED`) → Cloud Function ranks nearest drivers → FCM siren push to targeted driver → Driver accepts (`claimEmergency` transaction) → `EN_ROUTE_TO_PATIENT` → `DriverLocationService` GPS telemetry → `ARRIVED_AT_PATIENT` → `markPatientPickedUp` (`PATIENT_ONBOARD`) → `EN_ROUTE_TO_HOSPITAL` → `ARRIVED_AT_HOSPITAL` → `COMPLETED`

6 Limitations

- GPS accuracy varies with device hardware, the physical environment, and network conditions. The `DriverLocationService` uses `PRIORITY_HIGH_ACCURACY` and a 2,500 ms update interval, but location should be treated as an operational aid rather than a certified metric. Indoor or underground environments with poor satellite visibility will reduce fix quality.
- The cloud-based workflow depends on network availability. Loss of connectivity on any device can delay request synchronization, status updates, or GPS telemetry writes to Firebase Realtime Database. The current prototype does not implement offline queuing or conflict resolution for status transitions. Firestore offline persistence is enabled for reads, but write operations require connectivity.
- Ambulance availability depends on the actual fleet and participating drivers. The dispatch algorithm ranks all ambulances whose Firestore status field is `AVAILABLE` and whose GPS coordinates are non-zero; ambulances that are offline, have not started their location service, or have stale coordinates may be ranked incorrectly. The system depends on drivers correctly toggling their online/offline status.
- Hospital data is stored in Firestore and must be maintained manually or through an administrative import. The current backend includes a Node.js administrative utility (`RAAHconnectBackend`) for importing hospital master data. Real-time bed availability and department capacity are not yet integrated; the hospital coordination component shows static information and live emergency status only.
- Patient location coordinates and emergency information are stored in Firestore under security rules that restrict write access to authenticated users and read access to assigned drivers, dispatchers, and hospital staff. Before production deployment, independent security review, data

minimisation, consent mechanisms, and compliance with applicable healthcare data regulations are required.

7 Conclusion

Emergency ambulance response is a coordinated healthcare process involving the patient, ambulance service, dispatcher, and receiving hospital. The literature reviewed in this paper covers emergency-care coordination, ambulance allocation and routing, hospital selection, and intelligent transportation integration, each area studied in relative isolation. The RAAH Connect prototype addresses the integration problem at the application and cloud-coordination level.

The implemented prototype combines structured SOS initiation with GPS capture (Patient app), server-side nearest-driver ranking and atomic dispatch transactions (Cloud Functions), foreground GPS telemetry to Firebase Realtime Database (DriverLocationService), siren-alert FCM push notifications for background awareness (EmergencyAlertNotificationHelper), a real-time dispatcher monitoring dashboard (React/TypeScript/Vite), and hospital coordination through a shared Firestore record. The end-to-end emergency lifecycle from patient SOS to ambulance arrival and handover at the hospital is demonstrated in a single integrated digital workflow.

The system should be viewed as a prototype integration platform, not a validated emergency-response optimization system. Future work includes load testing under concurrent emergencies, integration with real-time hospital bed availability APIs, integration with intelligent traffic signal systems for ambulance priority corridors, an offline-first resilience layer for connectivity gaps, and formal clinical trials to measure response-time impact. Security hardening, penetration testing, and regulatory compliance review are prerequisites before any production deployment.

References

- [1] World Health Organization. *Prehospital emergency care: Operational guidance for ambulance systems*. Geneva: WHO; 2025.
- [2] World Health Organization. *WHO Emergency Care System Framework*. Geneva: WHO; 2018.
- [3] Shetab-Boushehri SN, Rajabi P, Mahmoudi R. Modeling location-allocation of emergency medical service stations and ambulance routing problems considering the variability of events and recurrent traffic congestion: A real case study. *Healthcare Analytics*. 2022;2:100048. <https://doi.org/10.1016/j.health.2022.100048>
- [4] Chou CC, Chiang WC, Chen AY. Emergency medical response in mass casualty incidents considering the traffic congestions in proximity on-site and hospital delays. *Transportation Research Part E: Logistics and Transportation Review*. 2022;158:102591. <https://doi.org/10.1016/j.tre.2021.102591>
- [5] Lee S. The role of hospital selection in ambulance logistics. *IIE Transactions on Healthcare Systems Engineering*. 2014;4(2):105–117. <https://doi.org/10.1080/19488300.2014.914608>
- [6] Cao M, Shuai Q, Li VOK. Emergency vehicle-centered traffic signal control in intelligent transportation systems. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019. p. 4525–4531. <https://doi.org/10.1109/ITSC.2019.8916879>
- [7] Kanimozhi T, Kishore Reddy GRV, Rahul Kumar N. IoT-based smart signal system for traffic density management and emergency vehicle priority. In: *2024 International Conference on Intelligent Systems and Advanced Applications (ICISAA)*. Pune, India; 2024. <https://doi.org/10.1109/ICISAA62385.2024.10828938>
- [8] World Health Organization. *Prehospital toolkit*. Geneva: WHO [cited 2026 Oct 8].
- [9] World Health Organization. *Prehospital trauma care systems*. Geneva: WHO; 2005.
- [10] Jana A, Sarkar A, Parmar V, Saunik S. Examining district-level disparity and determinants of timeliness of emergency medical services in Maharashtra, India. *Scientific Reports*. 2023;13:21239. <https://doi.org/10.1038/s41598-023-48713-1>

-
- [11] Ayyan MS, et al. Developing a high-quality patient-centric integrated model for emergency care system in selected districts of India: An implementation research protocol (INDIA-EMS Study). *PLOS ONE*. 2025;20(9):e0331290. <https://doi.org/10.1371/journal.pone.0331290>