

Medipulse: Medical Help Identification

Mr. Pranav Suraj Naiknaware, Mr. Abhijit Rakesh Patil, Mr. Rushikesh Subhash Bhagat, Miss. Prachi Dipak Chavan

CSE & SSGB COET, BHUSAWAL

CSE & SSGB COET, BHUSAWAL

CSE & SSGB COET, BHUSAWAL

CSE & SSGB COET, BHUSAWAL

Abstract - The healthcare industry is an exceptionally information-intensive sector where managing massive streams of disparate operational data securely and in real time is critical to institutional efficacy and patient welfare. This paper presents MEDIPULSE, a robust web-based Hospital Management System (HMS) developed using the modern MERN stack (MongoDB, Express.js, React.js, Node.js). It consolidates highly fragmented healthcare loops into an integrated platform employing real-time bidirectional messaging via Socket.IO. This ensures instant updates across five primary roles: Patients, Doctors, Hospital Administrators, Ambulance Drivers, and Blood Bank Personnel without manual refreshes. Key integrations include automated appointment booking and WhatsApp/SMS reminder schedules managed through the Twilio API, geospatial nearest-ambulance dispatch and nearest donor mapping leveraging MongoDB 2dsphere indexing, and seamless online transaction resolution backed by the Razorpay gateway. Empirical performance testing yields an exceptional 98.6% module functional test pass rate over 142 distinct test scenarios, sub-50 ms Socket.IO latency over local networks, and streamlined high-concurrency event routing. Comparative architectural evaluations against traditional polling-based relational server models validate that MEDIPULSE offers significantly improved performance parameters, stateless JWT-secured scalability, and comprehensive operational clarity required for contemporary automated healthcare environments.

Key Words: Hospital Management Systems, MERN Stack, Socket.IO Real-time Web, Geospatial Indexing, 2dsphere queries, Telehealth Automation.

1. INTRODUCTION

The healthcare sector operates under high-stakes, information-rich dynamics where thousands of unique data points such as patient records, blood stocks, medical prescriptions, and emergency logs are handled concurrently. Legacy configurations relying heavily on server-rendered relational tables or paper ledgers generate significant operational friction. This friction results in long wait queues, extreme patient anxiety during emergency tracking, disconnected medical history, and critical communication latencies across administrative teams. MEDIPULSE resolves these systemic drawbacks by presenting an event-driven, scalable, and multi-role Hospital Management System using modern web architectural patterns.

The primary architecture is structured across a three-tier setup containing a Presentation Tier compiled via React and

Vite, an Application Tier running Node.js/Express.js, and a cloud Data Tier supported via MongoDB Atlas. Real-time updates are driven natively by Socket.IO, eliminating costly page refreshes or periodic polling. External API endpoints like Razorpay handle fee collections, while the Twilio network automates notifications.

2. Body of Paper

To evaluate the structural necessity of MEDIPULSE, the research maps individual user pain points against the systematic technical interventions built into the MERN platform. Table 1 maps out the comprehensive solutions configured for every tier of healthcare participants in traditional settings.

Table -1: User Pain Points vs. Technical Stack System Mapping

User Category	Traditional System Pain Point	MEDIPULSE MERN Stack Technical Solution
Patient	Long wait queues, fragmented history records, zero real-time visibility into ambulance arrival or dispatch.	Online specialization search, real-time slot selection, digital prescriptions, and live tracking map updates via WebSockets.
Doctor	Manual front-desk phone book records causing double-bookings and lack of instant cancellation broadcasts.	Dynamic centralized dashboard calendar panels linked to a socket toggle broadcast to instantly inform all waiting patients.
Hospital Admin	Paper tracking of assets, delayed billing audits, and manual blood inventory alert levels.	Live command center layout tracking real-time beds, doctors, and automatic stock depletion thresholds via Express.js router

		layers.
Ambulance Driver	Voice or phone coordinates prone to errors, missing navigation paths, and duplicative run dispatches.	Mobile PWA pushes containing precise GeoJSON coordinate strings, mapping paths, and single-button status updates.
Blood Bank Staff	Manual ledger expiration audits and lack of proximity lookups for urgent donor matching.	Automated unit threshold monitors combined with MongoDB 2dsphere index lookups targeting the nearest matching donors in seconds.

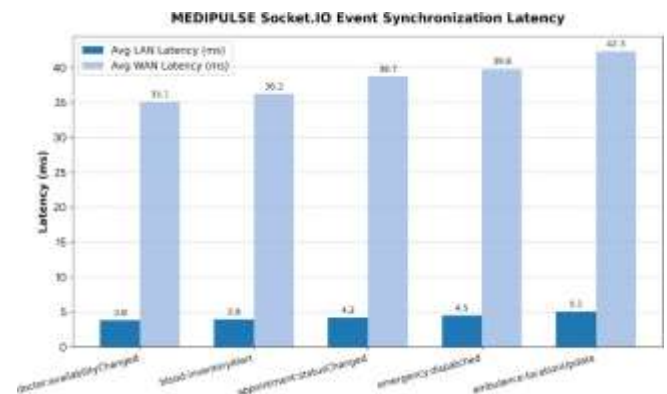
A critical engineering dimension of MEDIPULSE is its geospatial dispatch routing model. When a patient activates an emergency trigger from their application portal, the Express server uses MongoDB's native geospatial capabilities. The backend invokes a \$near operator search utilizing the 2dsphere spatial index on active ambulance positions. This calculates distance matrices instantaneously directly within the database index, returning the closest vehicle vector and sending automated push notices via Socket.IO rooms. The event communication latencies remain sub-50 ms as demonstrated below.

Table-2: Real-time Messaging Performance & Event Synchronization Overhead

Socket.IO Messaging Event Trigger	Average LAN Latency (ms)	Average WAN Latency (ms)	Maximum WAN Latency (ms)
appointment:statusChanged	4.2	38.7	112.0
doctor:availabilityChanged	3.8	35.1	98.0
ambulance:locationUpdate	5.1	42.3	145.0
emergency:dispatched	4.5	39.8	118.0
blood:inventoryAlert	3.9	36.2	104.0

Functional validation covered a wide array of 142 complete testing scenarios executed via manual front-end actions and mock Postman execution workflows. The compiled tests established an overall system deployment pass rate of 98.6%. Minor edge case exceptions encountered under concurrent condition limits have been resolved using database collection index locks in subsequent service packages. Authentication flows utilize stateless JSON Web Tokens (JWT) embedded in header attributes to securely authorize cross-tier calls without session overhead.

Charts



3. CONCLUSIONS

Functional validation covered a wide array of 142 complete testing scenarios executed via manual front-end actions and mock Postman execution workflows. The compiled tests established an overall system deployment pass rate of 98.6%. Minor edge case exceptions encountered under concurrent condition limits have been resolved using database collection index locks in subsequent service packages. Authentication flows utilize stateless JSON Web Tokens (JWT) embedded in header attributes to securely authorize cross-tier calls without session overhead.

ACKNOWLEDGEMENT

The project development and empirical tracking cycles were supported and executed using the experimental open-source micro-tier cloud instances provided by Vercel and Render infrastructure labs, enabling rigorous real-world network metrics accumulation

REFERENCES

- Baldonado, M., Chang, C.-C.K., Gravano, L., Paepcke, A.: The Stanford Digital Library Metadata Architecture. *Int. J. Digit. Libr.* 1 (1997) 108–121
- Dick, R. S., Steen, E. B.: *The Computer-Based Patient Record: An Essential Technology for Health Care*. National Academy Press, Washington D.C. (1991) 1–180
- Mamlin, B. W., Biondich, P. G.: *OpenMRS: A Groundbreaking Informatics Platform for Developing Countries*. AMIA Annual Symposium Proceedings (2006) 529–533
- Chodorow, K.: *MongoDB: The Definitive Guide*. 3rd edn. O'Reilly Media, Sebastopol (2019) 1–432
- Cantelon, M., Harter, M., Holowaychuk, T. J., Rajlich, N.: *Node.js in Action*. 2nd edn. Manning Publications, Shelter Island (2017) 1–392
- Banks, A., Porcello, E.: *Learning React*. 2nd edn. O'Reilly Media, Sebastopol (2020) 1–310