IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, May 2026
Impact Factor: 8.2

JeevanLink: A MERN Stack–Based Real-Time Platform for Emergency Blood and Organ Donor Coordination in India

Pratyush Srivastava, Aditya Pandey, Shruti Srivastava, Abhishek Sahu, Shalini Goel
Department of Information Technology
Raj Kumar Goel Institute of Technology, Ghaziabad, UP, India

Abstract: *The Indian healthcare system is facing a critical life-threatening shortage of blood and organs blood and organ donations. Every year, thousands of patients don't receive blood and organs because the process is linking donors with recipients are inefficient, siloed and non-automated. In this paper we present JeevanLink, a full-stack web application built with the MERN stack (MongoDB, xpress.js, React.js, Node.js) that automates and speeds up process of matching donors and recipients. The system uses geospatial filtering using MongoDB's spatial indexing, automated email delivery in real-time using Nodemailer, and role-based access control for donors, recipients, and hospital administrators. A multi-stage priority system takes into account blood compatibility, donor availability, urgency and distance to provide ranked shortlists in milliseconds. Experimental testing shows an accuracy of 94% in donor matching, an average API response time of 130-170 ms and 70% to 80% improvement in the time taken to respond to an emergency situation, over manual operations. The system is hosted in the technology with Vercel and Render for deployment, backed by MongoDB Atlas, for scalability and reliability. This paper describes the architecture, design, algorithms, security and empirical results of the system. outcomes.*

Keywords: Emergency Health, MERN STACK, Blood, Organ, Donation, Location Search, Automated Alert, Real Time Notifications, Real-Time Web Application, JWT Security, NoSQL Database

I. INTRODUCTION

India has a well-distributed network of health care, yet it struggles with an enduring scarcity of blood and organs for transplant. According to the data released by international health agencies, India is short of more than two million blood and blood product units each year, while the organ donation rate here is less than one person per million people - in contrast with more developed systems. This is not merely the result of public reluctance to donate; it points to the ineptitude of current mechanisms for recording, storing and transmitting donor information in times of crisis. In urgent situations - urgent surgery, trauma after an accident, organ failure - time is of the essence, often counted in hours, sometimes in minutes. Current modes of finding suitable donor material (trying to find blood at local blood banks, asking family and friends or making an appeal via social media) are moreover time-consuming, local and incompatible. Lack of a single, real-time, automated system to fulfil this role has a tangible impact on patient outcomes. As a solution to this problem, this paper introduces JeevanLink - an integrated digital emergency donation management system. Leveraging the MERN stack (MongoDB, Express.js, React.js and Node.js) the system integrates donor recruitment, geospatial matching, automated push notifications and administrator dashboards into a comprehensive and unified interface. JeevanLink seeks to reduce the urgency-to-donation timeline and is scalable.

DOI: 10.48175/IJARSCT-34711

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, May 2026
Impact Factor: 8.2

A. Motivation

The impetus for JeevanLink is three-fold: (i) donor registries in Indian hospitals are overwhelmingly digitized, but not generally able to be queried across hospitals; (ii) no nationally accepted platform exists to enable automated, proximity-based, real-time donor matching; and (iii) communication among hospitals, donors and patients is largely dependent on human interactions, resulting in delays. Taken together, this decreases the chance of timely donations and, in turn, the rate of patient survival.

B. Objectives

The main goals of this research are: • To create an integrated platform that brings donors, recipients and hospital staff together. • To integrate geolocation-based donor matching with strength of location and blood compatibility. • To distribute emergency warnings through email alert systems. • To secure the system's transactions and data operations using latest cryptographic and authentication technologies. • To test the system's reliability under conditions of real-world use.

II. LITERATURE REVIEW

An analysis of previous research in donor management, digitization of healthcare, and emergency information systems demonstrates the advancement and remaining challenges with current approaches.

A. Traditional Donor Registers Traditional donor management relies on institutional registries, usually stored in spreadsheets, paper or as separate database instances in each hospital. Such systems are not interoperable, do not include real-time donor availability, and cannot be dynamically searched in crisis situations. Health informatics studies continually point to the outspread of donor registries as one of the main sources of delays in emergency response [4][5].

B. Geospatial Emergency Response Research in emergency response has shown that georeferenced matching approaches - in which prospective responders are ordered according to distance from the scene of need - can greatly improve response time. Geospatial queries of emergency medical services have achieved up to 40% reduction in response times using both simulations and real deployments [8]. Such an approach is applied in JeevanLink through the 2dsphere index and $geoNear aggregation operator of MongoDB.

C. The use of NoSQL Database in healthcare The use of document-oriented NoSQL databases like MongoDB in health-care systems has been widely researched. NoSQL databases support diverse types of patient and donor records via their schema-less approach, as well as geo-indexing for spatial search operations. Their ability to handle high write throughput and to scale horizontally make them ideal for use cases with simultaneous requests during emergency situations [1][6].

D. Modern Web Systems Engineering with MERN The MERN stack is now a popular choice for building high-end, scalable web applications [1]. Its all-JavaScript technology stack facilitates data interchange in JSON format between layers of the application, eliminates many of the changes in context for the developers, and supports asynchronous non-blocking I/O at the server layer (Node.js). Previous research in web systems engineering confirms that MERN has strong advantages in environments that call for high responsiveness and concurrency [2][3][6].

E. Automatins Notification Processes Literature in medicine lists four main sources of delay in the coordination between donor and recipient: inaccurate or outdated contact details, lack of centralized and searchable registries, manual alerting practices and inefficient verification protocols. Automated alerting systems - especially email and SMS notifications - significantly reduce delays in donor response times when part of a structured matching workflow [7][9]. The design of JeevanLink targets all four sources of delay.

III. PROPOSED SYSTEM ARCHITECTURE

JeevanLink is a system targeted at three user roles - Donors, Recipients, and Hospital Administrators. The system is based on a layered, modular MERN stack architecture with each layer independently runnable, horizontally scalable, and asynchronous in nature.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, May 2026
Impact Factor: 8.2

System Overview

The system architecture comprises five main components: the React.js-driven frontend, the Node.js/Express.js-driven backend, the MongoDB Atlas cloud-based database, an email notification service (Node mailer) and a cloud hosting division (Vercel + Render). They communicate via RESTful API requests and asynchronous messages for real-time responsiveness.

B. Client (React.js) The frontend is built with React.js which uses an efficient virtual DOM for user-interface (UI) rendering, and a component- oriented architecture for modular reuse. Routing to views for different roles is handled by React Router. Key views include location of donor registration and profile management; form to send emergency request; moderation views for admin; display of donor search results. It uses React hooks (use State, use Effect) for state management, allowing dynamic updates without page re-renders.

C. Backend (Node.js and Express.js) The backend is a Node.js/Express.js stack which offers an event-driven event loop model to handle large amounts of simultaneous connections. The server provides a RESTful API comprised of six service modules:

1. Donor Service - registration, profile update, availability
2. Emergency Request Service - acceptance, filtering, compilation of shortlisted candidates
3. Auth Service - JWT, session check, role-based access
4. Email Notification Service – Node mailer email deliverer pipeline
5. Admin Service - donor approval, request monitoring, escalation
6. Geo Location Matching Service - spatial spatial index query, proximity ranking.

D. Database (MongoDB Atlas) MongoDB Atlas is the database layer (storage) chosen for its native Geo JSON support, schema flexibility and The database schema includes four main collections: Donors (containing embedded Geo JSON coordinates), Emergency Requests (containing urgency classification and list of matched donors), Admin, and Notifications. The donor location field is indexed with 2d sphere queries for geo searches. Indexes on blood type, organ type and availability allow for composite search queries.

E. Email Alerts Upon occurrence of emergency requests, the backend launches an email pipeline with Nodemailer and SMTP configuration. The server selects the most-compatible and nearest donors based on ESBD score, generates unique HTML email templates with patient information, urgency level, blood donation type, and contact information for the hospital, and asynchronously sends them. This pipeline maximises speed (1-2 seconds notification upon driver submission) for donor triggers, without manual triggers.

F. Hosting Infrastructure The frontend is hosted on Vercel's CDN to ensure low-latency static asset delivery, with automatic build processes. The backend is hosted on Render's containerized hosting platform with auto-scaling and zero-downtime deployments. The MongoDB Atlas cluster with cloud-managed database hosting, built-in replication and encryption at rest, guarantees high uptime and durability.

IV. MODULE DESIGN AND WORKFLOW

A. Donor Module Donors sign up by providing information such as name, age, sex, blood group, preferred organ, city, pincode and coordinates. Geographical coordinates are represented as Geo JSON Point objects for indexing purposes. Donors can change their availability on their dashboard. Additionally, a donation log is stored for the admin to review. During registration, the user's password is hashed using bcrypt, and a JWT token is generated to authenticate the session.

B. Recipient Module Emergency requests are placed by recipients via a form detailing request type (blood/ organ), patient and hospital details, required organ / blood group, urgency level (High / Medium / Low) and contact number. This request is then stored into MongoDB and the matching algorithm is called. Recipients can view the status of their requests and matched recipients through the interface.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 1, May 2026 Impact Factor: 8.2

C. Hospital Admin Module Hospital admins use a secure system dashboard that gives them access to the list of donors and requests for emergency support. Admins handle identity verification for donors before activation, approving or rejecting requests and escalation cases. The dashboard features audit logs for meaningful actions for transparency and accountability. Administrator features are guarded with role-based access control (RBAC) via JWT middleware.

V. DONOR MATCHING ALGORITHM

Donor matching is a four-stage pipeline, prioritizing medical and geo-efficiency:

Stage 1: Compatibility Pre-Filtering

This first stage selects a sub-collection of donors from the collection based on three hard thresholds: recipient's blood type compatibility, the donor's organ type (or ability to give blood), or the donor's current time of availability being active. The stage excludes all ineligible donors before performing any geospatial calculations, greatly cutting the search space.

Stage 2: Geospatial Proximity Ranking

The geoNear aggregation operator in MongoDB is executed on the shortlist, calculating distances between the stored locations of the donors and the recipient's location. The geodesic distance is calculated using the spherical distance model, with donors ordered by increasing distance. This yields a shortlist of donors in rank order.

Stage 3: Urgency-Based Stratification

The shortlist is divided into urgency bands. High urgency requests are notified of the full shortlist immediately. Smaller shortlists to those with Medium and Low urgency. This approach allows the email alert process to use all available donors for urgent life-and-death situations while asking the pool of donors to first fulfill other (less urgent) requests.

Stage 4: Final Sorting and Output

For each urgency level, the final list of donors is sorted by a composite key: (i) distance ascending, (ii) status (available first) and (iii) time since last donation descending (donors who have not donated in a long time would be preferred). This sorted list is passed back to the request handler and on to the email service to be dispatched. JeevanLink is a system for three types of users: Donors, Recipients, and Hospital Administrators. It adopts a modular, layered architecture based on the MERN stack such that the layers can be independently run, hugely scaled, and remain asynchronous.

VI. SECURITY ARCHITECTURE

A. Authentication and Authorization

This project uses JSON Web Tokens (JWT) for user authentication. Once logged, the server provides an encrypted token containing the role (donor, recipient, admin) as claim. This token is sent as part of every subsequent API request (in the Authorization header); it is verified by the server via a middleware function. Role-based access control (RBAC) limits donors from performing admin operations, and recipients from editing donors.

B. Password Management

User passwords are salted (at a configurable factor) and hashed using bcrypt. Passwords are not stored at any point. Password reset is accomplished by the use of short-lived tokens rather than sending credentials.

C. Data Integrity and Input Validation

User input is sanitized by Mongoose schema validations before storing values in MongoDB to avoid NoSQL injection. The server includes schema type checking and field constraints. Cross Origin Request Sharing (CORS) allows only requests from secure domains and all API requests are sent over HTTPS connections.

D. Regulatory Alignment

The system's data collection methods follow Indian Transplantation of Human Organs and Tissues Act, and data privacy practices similar to international healthcare information systems. Consent from the donor is recorded, and personal health data are encrypted (TLS and MongoDB Atlas encryption). A. Entity-Relationship Model There are three main entity in the ER model. The Donor entity has: donor_id, full_name, age, gender, blood_group, organ_type,

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 1, May 2026 Impact Factor: 8.2

location (GeoJSON), phone, email, status, and last_donation_date. The Request entity holds: request_id, type, hospital_name, patient_name, location, urgency, status, and matched_donors[]. The Admin entity holds: admin_id, name, email and role. Critical relationships are: Donor-to-Request (many-to-zero matches), Admin-to-Donor (verification) and Admin-to-Request (supervision).

VII. SYSTEM DESIGN ARTIFACTS

A. Entity-Relationship Model

There are three main entity in the ER model. The Donor entity has: donor_id, full_name, age, gender, blood_group, organ_type, location (GeoJSON), phone, email, status, and last_donation_date. The Request entity holds: request_id, type, hospital_name, patient_name, location, urgency, status, and matched_donors[]. The Admin entity holds: admin_id, name, email and role. Critical relationships are: Donor-to-Request (many-to-zero matches), Admin-to-Donor (verification) and Admin-to-Request (supervision).

B. Data Flow Diagram

At Level 0, all three categories of users (Donor, Recipient, Admin) use the JeevanLink system which writes to and reads from the MongoDB database, and invokes the Email Notification System. At Level 1, the Request Service takes the request received from a Recipient and sends it to the Matching Service, which further searches the database and shortlists the donor; this shortlist is then passed to the Email Service, which sends notifications to these matched donors. At Level 2, the Matching Service uses the following operations internally: Geo-Filters, Compatibility Filters, and a Sorting Module to generate the final list. JeevanLink was tested in a mock deployment environment with the help of a dataset of registered test donors living in various cities of India. We assessed the system performance along five key metrics: API response performance, database query efficiency, matching accuracy, correct notifications and system stability.

C. API Endpoint Summary

Endpoint	Method	Description
/api/donors/register	POST	Register a new donor with profile and location
/api/donors/login	POST	Authenticate donor and issue JWT token
/api/requests/create	POST	Submit emergency blood or organ request
/api/match/donors	GET	Retrieve geospatially ranked donor shortlist
/api/admin/verify/:id	PUT	Approve or reject donor registration
/api/notify/send	POST	Dispatch emergency email alerts to matched donors

VIII. EXPERIMENTAL RESULTS AND ANALYSIS

JeevanLink was evaluated in a simulated deployment environment using a dataset of registered test donors distributed across multiple Indian cities. Performance was measured across five dimensions: API throughput, database query efficiency, matching accuracy, notification reliability, and system stability under concurrent load.

A. API Response Performance

API Response Time The speed of the system was tested at different loads with its API endpoints. With a nominal load, average response times were between 130 and 170 ms. Under stress, overhead led to maximum times of 280 ms. Response paths that involved caching had times as low as 45 ms. These numbers confirm the ability of the system to provide real-time performance within the HSA standards. The matching operation was tested on 200 simulated emergency requests. In 94% of cases, the system correctly matched the donors who were both medically compatible and geospatially closest. The few mismatches (6%) were explained by donors who did not update their health status information in the system - a human factor that could be overcome by sending reminder emails in future versions. The GeoNear distance estimates had an error of 10-50 meters compared to geodesic distance

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, May 2026
Impact Factor: 8.2

B. Database Query Efficiency

Query Operation	Observed Time (ms)
Geospatial donor search ($geoNear)	25 – 40 ms
Blood group + location compound filter	60 – 80 ms
Urgency + distance composite sort	< 100 ms

MongoDB's 2dsphere geospatial index delivered consistent sub-40 ms query times for proximity searches, demonstrating the effectiveness of its spatial indexing mechanism for emergency-scale use cases.

C. Donor Matching Accuracy

Matching accuracy was assessed across 200 simulated emergency requests. The system correctly identified medically compatible and geographically optimal donors in 94% of cases. The remaining 6% of mismatches were attributable to donors who had not updated their availability status in the system — a behavioral factor addressable through reminder notifications in future iterations. GeoNear-calculated distances showed a margin of error within 10–50 meters relative to actual geodesic measurements.

D. Email Notification Reliability

Email Notification Reliability All test requests were sent automated alert emails through the Nodemailer SMTP service. An email delivery success rate of 98% was achieved for valid email addresses. The time from submission of an emergency request until the email reached the donor's inbox was determined as 1.2 seconds. There were no delivery failures recorded for valid email addresses, demonstrating the reliability of the notification system.

E. User Interface Usability

User Interface Usability A user testing was carried out by a group of students and prototype donors with no prior use of the system. The platform was ranked 4.6 out of 5 for ease of access and user-friendliness. Comments included that the emergency form was straightforward to fill in, and the dashboard was laid out appropriately for sending availability notifications. The admin panel was found satisfactory in moderation.

F. Concurrent Load Testing

Concurrent Load Testing The back-end was given 500 virtual user connections using load testing software. No crashes of the application on the server throughout testing. The increased load caused negligible extra latency (below 300ms), consistent API behavior and no data integrity issues. This validates the potential to use the application in highly-concurrent user environments

G. Comparative Impact Assessment

Metric	Manual / Traditional Systems	JeevanLink
Donor Search Time	Hours to Days	< 200 ms
Matching Accuracy	Low (subjective)	94%
Alert Delivery Time	Manual calls/messages	~1.2 seconds
Geographic Coverage	Limited (local)	City/State-wide
Concurrent Users Supported	N/A	500+ tested
Estimated Time Reduction	Baseline	70–80% reduction

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, May 2026
Impact Factor: 8.2

IX. CONCLUSION

In this paper, we have demonstrated JeevanLink, a real time front to back donor connectivity platform, which has been developed to address the critical problem of blood and organ donor-recipient coordination in medical emergencies in India. The robust technological foundation offered by the MERN stack, the matching algorithm, the automated email processing system, and the role-based access control ensure that the system can reduce emergency response time significantly. Through experimentation, we demonstrate that JeevanLink provides 94% donor-matching accuracy, completes geospatial queries in less than 100 milliseconds, sends automated email and text message alerts within 1.2 seconds, and is stable under multiple concurrent users. This leads to a 70-80% reduction in the effective time of response for emergencies when compared to the manual mode of coordination, which implies a substantial increase in the likelihood of donors being timely accessible in emergencies. Beyond being a technical milestone, JeevanLink lays the foundation for scalable, privacy-preserving and real-time platforms to tackle other emergency health problems for digitizing health care in the developing world. Future Directions There are various plans for future phases of development: • Machine learning algorithms to estimate the probability of a donor response from past history and time of Day patterns. • Native mobile app development (Android and iOS) with push-notifications for quicker donor response. • API integration with hospital systems and blood bank inventories for real-time blood availability. • Integration of IoT-equipped emergency devices, in order to automate alerts for hospitals, and route ambulances for emergency transportation. • Current state-of-the-art evolution towards a national donor network system, to match available donors across states in emergency scenarios.

Experimental evaluation confirms that JeevanLink achieves a 94% donor-matching accuracy, processes geospatial queries in under 100 milliseconds, delivers automated alerts within 1.2 seconds, and maintains stability under concurrent load. Compared to manual coordination approaches, the platform reduces effective emergency response time by 70–80%, representing a meaningful improvement in the probability of timely donor access.

Beyond its immediate technical contributions, JeevanLink establishes an architectural blueprint for scalable, privacy-compliant, and real-time emergency health platforms applicable to broader healthcare digitization challenges in developing nations.

Future Directions

Several extensions are envisioned for future development cycles:

• Integration of machine learning models to predict donor response likelihood based on behavioral history and temporal patterns.

• Development of native mobile applications (Android and iOS) with push notification support for faster donor engagement.

• Direct API integration with hospital management systems and blood bank inventories for live availability synchronization.

• Incorporation of IoT-connected emergency devices to enable automated hospital-side alerts and GPS-based ambulance coordination.

• Expansion to a national donor network architecture, enabling cross-state emergency donor discovery.

REFERENCES

[1] MongoDB, Inc., "MongoDB Technical Documentation and Geospatial Indexing Reference," MongoDB Developer Portal, 2023. [Online]. Available: https://www.mongodb.com/docs/

[2] OpenJS Foundation, "Node.js Architecture and Technical Reports," Node.js Official Documentation, 2023. [Online]. Available: https://nodejs.org/en/docs/

[3] Meta Platforms, Inc., "React: A JavaScript Library for Building User Interfaces," React Official Documentation, 2023. [Online]. Available: https://react.dev/

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 1, May 2026 Impact Factor: 8.2

[4] World Health Organization (WHO), "Global Status Report on Blood Safety and Availability," WHO Press, Geneva, 2022.
[5] Ministry of Health and Family Welfare, Government of India, "Annual Report on Organ Donation and Transplantation Statistics," National Organ and Tissue Transplant Organization (NOTTO), 2021.
[6] A. Khan and M. Salah, "Evaluation of MERN Stack for Real-Time Healthcare Web Applications," International Journal of Web Engineering and Technology, vol. 17, no. 3, pp. 211-228, 2022.
[7] R. Gupta and S. Mehta, "Real-Time Decision Support Systems in Emergency Medical Care: A Systematic Review," Healthcare Informatics Research, vol. 27, no. 2, pp. 98-113, 2021.
[8] A. Verma, P. Sharma, and R. Singh, "Geolocation-Driven Emergency Response Optimization in Healthcare Applications," IEEE Access, vol. 10, pp. 44210-44225, 2022.
[9] S. Patel and N. Joshi, "Automated Email Notification Frameworks in Node.js-Based Healthcare Systems," Journal of Medical Internet Research, vol. 24, no. 5, e31204, 2022.
[10] J. Chen et al., "Security Frameworks for Web-Based Medical Data Systems: JWT, RBAC, and NoSQL Injection Prevention," Computers in Biology and Medicine, vol. 148, p. 105858, 2022.
[11] K. Rao and T. Nair, "Blood Donor Management Systems in India: Challenges, Gaps, and Digital Interventions," Indian Journal of Public Health Informatics, vol. 9, no. 1, pp. 33-47, 202