Blood Donor Management System (LIFE LINK): A Location-Aware Emergency Request Dispatch Platform with Role-Based Donor, Organisation and Admin Portals

Anbu Priyatharsini V.[1], Ramya G.[2], Soundhranayaki V.[3], Swetha C.[4]
[1]Assistant Professor, Adhiyamaan College of Engineering, Hosur
[2,3,4]UG Scholars, Adhiyamaan College of Engineering, Hosur– 635130, India

***Abstract*— When a patient needs blood after an accident, during surgery or in a childbirth complication, the family usually starts a chain of phone calls, social-media posts and messages to find a donor of the right blood group who is close enough to reach the hospital in time. The information is scattered and often out of date, and nobody can tell who has seen the request or who is coming. The Blood Donor Management System, named LIFE LINK, is proposed to bring this into one platform. A donor registers with a blood group, a location and an availability flag. A hospital or organisation raises an emergency request with the blood group, the units, the hospital location and an urgency label (Critical, Urgent or Standard). The system selects the available donors of that group within a configured radius of the hospital, 8 km in the prototype, and stores a notification for each; a donor sees only the matching open requests, nearest first. Donor, organisation and admin users sign in through separate role-based portals, and each donation is kept in a history. The system is built with an HTML, CSS and JavaScript front end, a Node.js and Express back end reached through REST requests, and a MongoDB database; passwords are hashed with bcrypt and tokens expire. The prototype was demonstrated through its sign-in, donor, organisation and admin screens. Response times, matching accuracy and user satisfaction have not yet been measured and are reported as limitations. Blood group matching only organises information: eligibility and compatibility must be confirmed by medical staff.**

Index Terms*— *blood donor management, emergency blood request, location-based search, donor matching, geographic radius, role-based access, Node.js, Express, MongoDB, web application

I. INTRODUCTION

Blood cannot be manufactured, and hospitals depend on voluntary donors to supply it for surgery, accidents, childbirth complications and the treatment of many diseases [1]. In an emergency the difficulty is seldom that no donors exist. It is that the right donor, with the right blood group, close to the hospital and willing to come, cannot be found quickly. In practice the patient's relatives phone friends, post a request in messaging groups and wait. The request travels as a forwarded message, the same message arrives many times, and nobody knows which of the replies is current.

Three problems follow from this. First, the process is slow and unpredictable, because it depends on who happens to see the message and who is free to answer. Second, the information is scattered and goes out of date: a donor who gave blood last month, or who has moved to another city, may still be on the list. Third, the requests carry no structure. A donor who receives a message cannot tell whether the hospital is 2 km or 40 km away, or whether the need is critical or can wait a day, and a large part of the effort is wasted on donors who cannot help.

The Blood Donor Management System, named LIFE LINK, is built on a simple observation: the donor record, the request and the match between them should live in one database that every role can read. In the proposed system a donor registers once and keeps an availability flag and a location. A hospital or organisation raises a request with the blood group, the units, the hospital, its location and an urgency label. The back end then selects the available donors of the same blood group who lie within the configured radius of the

hospital and stores a notification for each of them. A donor opens a dashboard that lists only the open requests of his or her own blood group within the radius, nearest first, and keeps a history of past donations.

The work was carried out as an undergraduate mini project, and the claims are limited accordingly. The software workflow, from donor registration to the nearby-request view and the donation history, was built, and the screens of the prototype are shown in this paper. Measured response times, load tests and user surveys have not yet been collected. The contributions of this paper are as follows.

1. A single platform in which donor records, emergency requests, notifications and donation history share one backend and one database schema.
2. A role-based design with three portals (donor, organisation and admin), so that each person sees only the tasks that belong to the role.
3. A matching rule in which a request is offered to a donor only when the blood group is the same, the donor is marked available and the donor's stored location lies within the configured radius of the hospital, together with Critical, Urgent and Standard urgency labels for display.
4. A donor-side view that lists matching open requests nearest first and keeps a donation history, so that the donor does not have to search several message groups.
5. A description of the prototype, with screen-level results and a plain statement of what has and has not yet been measured.

The remainder of the paper is organised as follows. Section II reviews related work and compares the proposed system with the manual process. Section III describes the system architecture and workflow. Section IV presents the module design. Section V reports results from the prototype. Section VI discusses them together with security and privacy considerations, Section VII lists limitations and the proposed evaluation protocol, and Section VIII concludes.

II. RELATED WORK AND COMPARISON

A. Location-based donor finding

Several works use the donor's location to narrow the search. Kala [2] presents a location-based blood donor locator that uses a matching method, and Kamble et al. [3] describe a location-based donor finding system. Adeleke et al. [4] build the Geolocation-Integrated Emergency Blood Donation Response System (GI-BDERS), which combines geolocation tracking, automated donor matching and SMS communication, and is implemented with Django, PostgreSQL, the Google Maps API and an SMS service. Their testing reports donor response times between 7 and 19 minutes. These works show that location is the most useful single filter for emergency requests. The present work uses the same idea on a JavaScript stack, with a fixed radius around the hospital and no SMS service.

B. Real-time alerts and tracking

Nandini et al. [5] present BloodConnect, a proximity-based donation network built on the MERN stack. It uses GPS location services to link donors and recipients, sends browser push notifications to donors, and generates a QR code that is scanned at the donation centre to confirm attendance, which is tied to a reward system. The platform is deployed in Docker containers and its data flow is checked with Postman. BloodConnect is the closest relative of the present system, because it also uses Node.js and MongoDB. LIFE LINK in its current form stores notifications inside the application and does not yet send push messages or use QR check-in.

C. Matching methods and further related work

Other works go beyond a distance filter. Nwaukwa et al. [6] apply the K-Nearest Neighbor method to blood donation management, Kanna Deepa et al. [7] describe an AI-based donor connect platform that uses blood type matching with real-time emergency location detection, and Nithya and Maheshwari [8] study data-driven matching. Judging from their titles, these works use learned or data-driven matching. The present

system uses a fixed rule, which is easy to explain and test but does not learn from the past responses of donors. Further works address location-aware management of donation camps and emergency demands [9], blood inventory with donor clustering and geospatial visualisation [10], and a privacy-preserving donation social network [11].

D. Gap and the proposed contribution

The reviewed works show that location filtering, real-time alerts and data-driven matching are each feasible. The descriptions reviewed say little about three points that matter in a college-level system: separate portals for the donor, the requesting organisation and the administrator inside one small prototype, a request that immediately produces a stored notification for every matching donor, and a donor-side list that shows only matching open requests ordered by distance. The proposed system combines these in one workflow. Table I compares the manual process with the proposed system. The entries describe the process itself and do not report measured values.

TABLE I. COMPARISON OF THE MANUAL PROCESS AND THE PROPOSED SYSTEM

Aspect	Manual process	Proposed system
Finding donors	Phone calls and forwarded messages	Query on blood group, availability and location
Information	Scattered across contacts and groups	One database with donor and request records
Distance	Not known to the donor	Haversine distance, 8 km radius in the prototype
Urgency	Stated in the message text	Critical, Urgent or Standard label on each request
Availability	Asked again for every request	Availability flag kept by the donor
Request status	Unknown to donors	Open, Fulfilled or Cancelled
Donation records	Personal memory or paper	Donation history for each donor

III. SYSTEM ARCHITECTURE AND WORKFLOW

A. Architecture overview

Fig. 1 presents the architecture of the system as seven connected components. Donor Information holds the donor's name, blood group, location, verification status, availability and eligibility. Emergency Blood Request holds the required blood group, the units, the hospital, the distance, the urgency and the required time. LIFE LINK Core is the central processing component that connects the application logic, the database, security and integration. Request Matching and Prioritization organises requests by blood group, eligibility, availability, distance, urgency and required time. Geographic Awareness uses a configured radius, drawn as 8 km in the figure, to focus on nearby requests. Donor Dashboard and Request View lets the donor review nearby requests and request status, and Donation History and Recognition keeps the donation records and contribution summary.

The front end is written in HTML, CSS and JavaScript. It sends API requests to a back end built on Node.js with Express [12], which validates the input, checks the token, applies the matching rule and reads and writes a MongoDB database [13]. The database holds four kinds of record: donors, blood requests, donations and notifications. The browser Geolocation API [14] supplies coordinates when the user gives permission. The Core box in Fig. 1 also lists AI/ML-based analysis and real-time coordination. These are design intentions: the matching in the prototype described here is rule-based, and no machine-learning model is implemented.

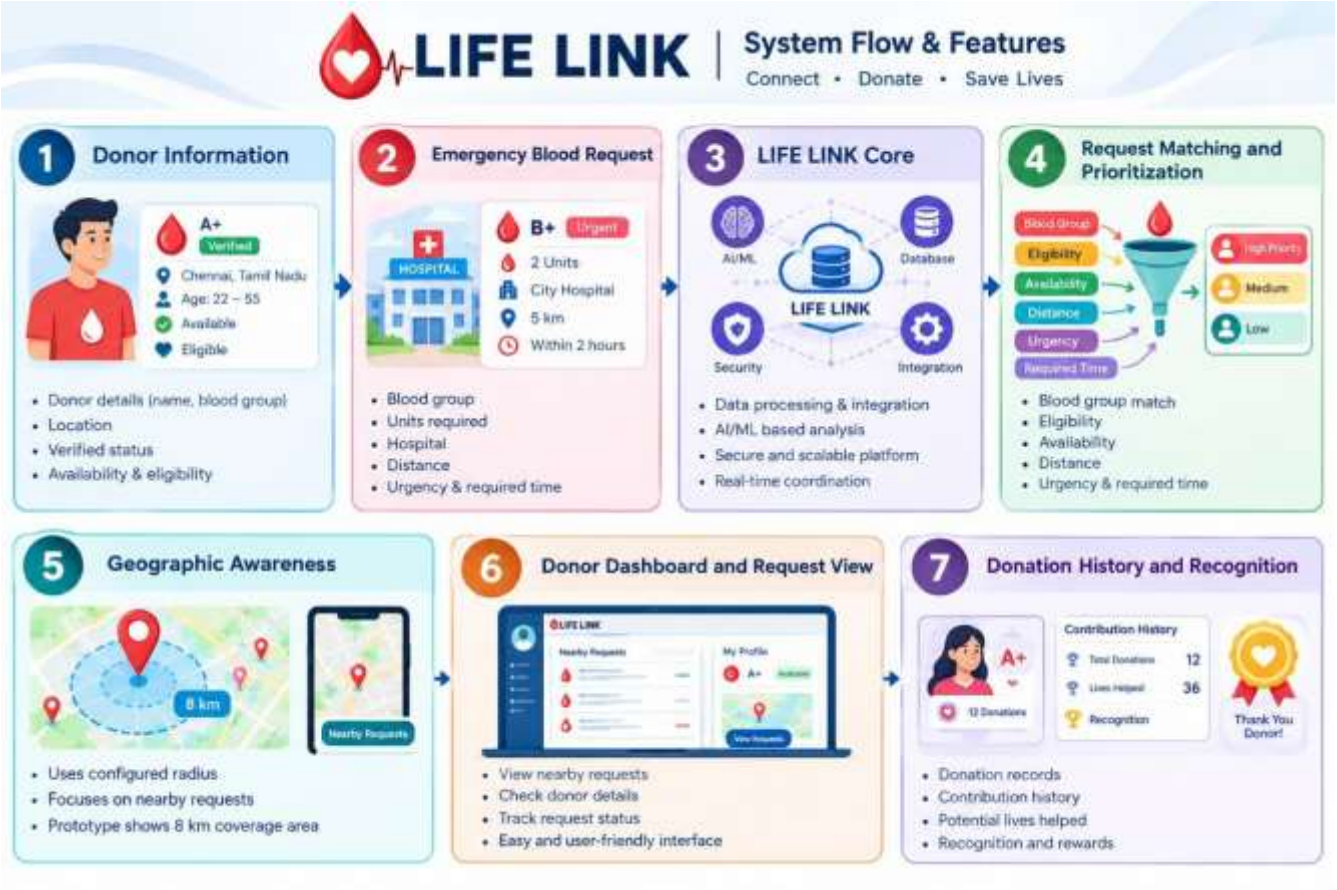


Fig. 1. LIFE LINK system flow and features (architecture).

The front end and the back end are separated deliberately. The browser holds only the interface and talks to the server through a small set of REST endpoints [15], so the database is never reachable from the client and the same back end could serve a mobile application later.

B. User roles and authentication

The system has three roles. A donor registers and signs in to the Donor portal, where the donor updates availability and location, views nearby requests and notifications, and records donations. An organisation, such as a hospital or blood bank, signs in to the Organisation portal to create requests and follow the response. An administrator signs in to the Admin portal to oversee donors, organisations and emergency activity. Fig. 2 shows the sign-in page, where the role is chosen with three tabs: Donor, Organization and Admin.



Fig. 2. LIFE LINK sign-in page with the Donor, Organization and Admin role tabs.

In the back end listed in the Appendix, registration needs a name, an email address, a password of at least eight characters and a blood group. The password is stored only as a bcrypt hash with cost factor 12, following the guidance of the OWASP password storage cheat sheet [16]. A successful login returns a signed JSON Web Token [17] that expires after two hours, and every protected endpoint rejects a missing or invalid token with error 401. A wrong email or password gives one generic message, so the response does not reveal which of the two was wrong. The sign-

in page in Fig. 2 shows donor sign-in with a mobile number and a one-time password. This differs from the email-and-password sign-in in the listed code, and the two should be aligned before deployment (Section VII).

C. Request lifecycle

The life of a request is a short chain, summarised in Table II. A donor registers and keeps an availability flag and a location. An organisation creates an emergency request, which is stored with the status Open. The back end selects the matching donors and stores a notification for each of them. A matching donor sees the request in the nearby list and responds outside the system or through the portal. When the need is met, or no longer exists, the request is set to Fulfilled or Cancelled. When a donor gives blood, the donation is recorded with its date, hospital and units.

TABLE II. STAGES OF A SINGLE REQUEST

No.	Actor	Action	Resulting state
1	Donor	Registers; sets availability and location	Donor record stored
2	Organisation	Creates request with group, units, hospital, urgency, location	Status Open
3	System	Selects available donors of the same group within 8 km; stores a notification for each	Alerts stored
4	Donor	Opens the nearby list: open requests of own group, nearest first	Request visible
5	Organisation	Updates the request when the need is met or withdrawn	Fulfilled / Cancelled
6	Donor	Records the donation (date, hospital, units)	History updated

The order is deliberate. A request is created only after its details pass validation, and only after it is stored does the system look for donors. A donor never sees a request of another blood group, or a request outside the radius, so the list stays short. The one piece of information the manual process never provides is also available here: the donor sees the distance to the hospital before deciding.

D. Data flow

Fig. 3 shows the data flow diagram. In the context-level diagram the user (donor, hospital or general user) sends donor details, blood requests and search queries to the LIFE LINK system and receives requests, donors and dashboard views in return, while the system stores and retrieves data in the database. The level 1 diagram divides the system into five processes: 1.0 Manage Donor Information, 2.0 Manage Blood Requests, 3.0 Filter and Match Requests, 4.0 Display Dashboard and 5.0 Manage Donation History. Every process reads and writes the same database, which holds the donor, request and history records.

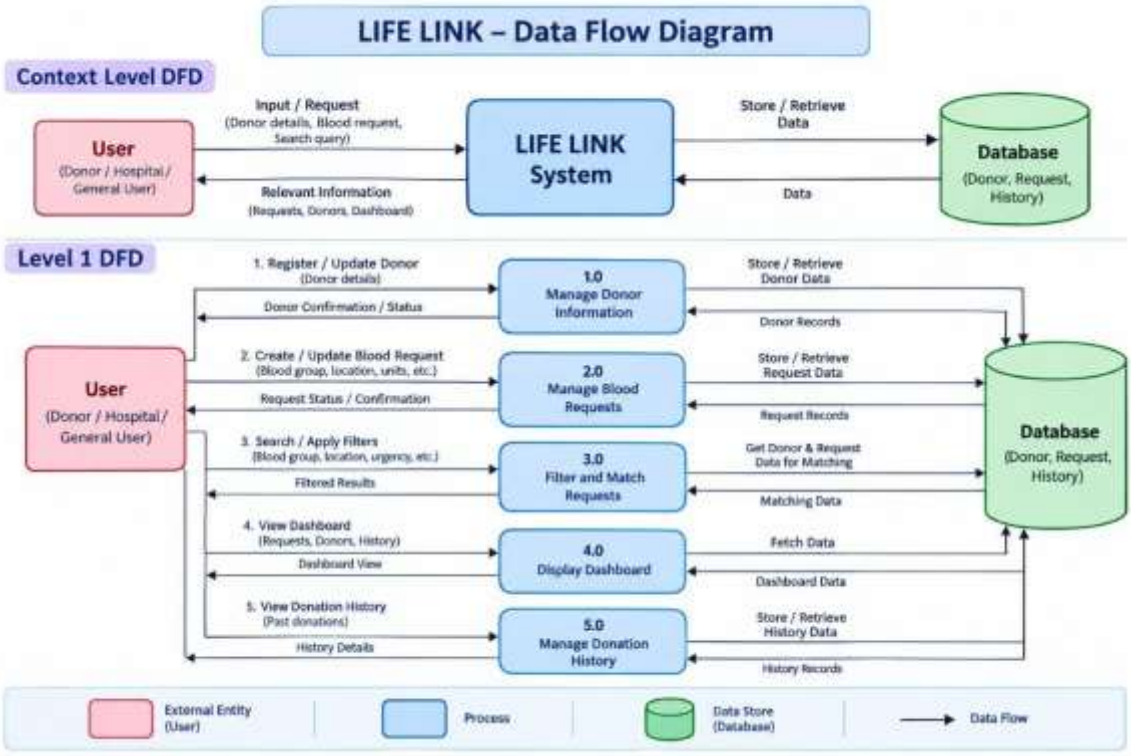


Fig. 3. Context-level and level 1 data flow diagrams of LIFE LINK.

E. Detailed workflow

The detailed workflow has nine steps. (1) Login: the user chooses a role and signs in. (2) Authentication: the back end verifies the credentials and returns a token. (3) Profile: the donor sets availability and location; the back end checks that the coordinates are numbers within the valid latitude and longitude ranges. (4) Request form: the organisation enters blood group, units, hospital, urgency and location. (5) Backend processing: the data are validated and stored in MongoDB. (6) Matching: the back end reads the available donors of the same group and computes the distance from each to the hospital. (7) Notification: a notification record is stored for every donor within the radius. (8) Nearby view: the donor opens the list of matching open requests. (9) Closure: the request status is updated and the donation is recorded.

F. Request processing procedure

Algorithm 1 summarises how the back end creates a request and finds the donors to notify. The order of the steps is deliberate: the cheap checks on the input come first, the database lookup follows, and the notifications are written last.

ALGORITHM 1. CREATING AN EMERGENCY REQUEST AND SELECTING DONORS

1. Receive blood group, units, hospital, urgency and location (lat, lng) from an authenticated user.
2. If the token is missing or invalid, return error 401.
3. If the blood group or hospital is missing, the units are not an integer of at least 1, or the coordinates are missing or outside the valid range, return error 400.
4. Store the request with status Open (urgency defaults to Standard).
5. Read the donors with the same blood group, marked available, that have a stored location.
6. For each donor, compute the haversine distance d between the donor and the hospital; keep the donor if $d \leq 8$ km.
7. Store one notification record for each kept donor.
8. Return the request id and the number of nearby donors (HTTP 201).

The distance in step 6 is the great-circle distance given by the haversine formula [18]: $d = 2R \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$, where $a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$, φ is latitude, λ is longitude and $R = 6371$ km. Algorithm 2 shows the donor's side of the same rule.

ALGORITHM 2. LISTING NEARBY REQUESTS FOR A DONOR

1. Authenticate the donor from the token; if the token is invalid, return error 401.
2. Load the donor record. If it has no stored location, return error 400 ("Update location first").
3. Read the requests with status Open and the same blood group as the donor.
4. For each request, compute the haversine distance between the donor and the hospital.
5. Keep the requests with distance ≤ 8 km and sort them in ascending order of distance.
6. Return the requests with the distance in km, rounded to two decimals.

IV. MODULE DESIGN AND IMPLEMENTATION

The work was organised as requirement analysis, system design, module implementation and testing of the screens and endpoints, with Postman as the tool for API testing. Requirement analysis identified the users (donors, requesting organisations and administrators) and the need for a centralised record, location-based selection, urgency labels and a donation history. System design produced the architecture, the database collections and the portal screens. The system was then implemented as the following modules.

A. Donor information management module. Stores the donor's name, email, hashed password, blood group, location (latitude and longitude) and availability flag, which defaults to true. The blood group is restricted to the eight values A+, A−, B+, B−, O+, O−, AB+ and AB−. Registration validates the required fields, the password length and the uniqueness of the email, and the availability and location endpoints validate their input.

B. Emergency blood request module. Stores the blood group, the units (at least 1), the hospital, the urgency (Critical, Urgent or Standard, default Standard), the hospital coordinates and the status (Open, Fulfilled or Cancelled, default Open). The status endpoint accepts only these three values and returns error 404 when the request does not exist.

C. LIFE LINK core processing module. Connects the interface, the Express routes and the MongoDB models. It checks the token on every protected route, validates the input and returns the error codes 400, 401, 404 and 500, so that a missing field or a database fault gives a defined response.

D. Request matching and prioritisation module. Offers a request to a donor only when the blood group is equal, the donor is available and the distance is within the radius. The urgency label is stored and displayed with the request. Equal blood group labels do not establish medical compatibility, so the result is a list of potentially relevant donors, not a decision.

E. Geographic awareness module. Computes the haversine distance and applies the 8 km radius, which is a configuration value. Coordinates come from the browser with the user's permission. If a donor has no stored location, the nearby-request endpoint asks the donor to update it first. The distance shown is a straight-line distance and not a travel time.

F. Donor dashboard and request view module. Presents the main interface of each portal. The donor portal shows a greeting, the blood group and city, the availability panel, the last donation date and the date from which the donor is next eligible, and the list of nearby requests relevant to the donor's blood group. The organisation portal offers Create Request, Active Requests, Donor Responses, Donor Referrals and Fulfilment, and the admin portal offers Emergency Management, Organization Management, Donors and Analytics.

G. Donation history and recognition module. Stores each donation with its date, hospital and units; a future date or a missing hospital is rejected. The history is returned newest first, and the notifications of a donor are returned newest

first in the same way. The donor portal has a Referrals and Recognition section, and recognition must rest on stored records, not on estimates of lives saved.

Testing and validation. Input validation and the error responses were built into the API, and the screens were checked in the browser. The evidence for each behaviour is listed in Table III. A formal test plan with recorded test cases has not yet been written, and no test results are claimed beyond the screens and code shown.

V. RESULTS

A. Application screens

Fig. 2 shows the sign-in page. The page offers three tabs, Donor, Organization and Admin, a mobile-number field with a Send OTP button for donors, and a Become a Donor link for new users. Fig. 4 shows the admin dashboard, which presents four summary cards (total donors, verified organisations, active requests and fulfilled requests), a featured active emergency case with its urgency label and the number of confirmed donors, the state of each dispatch round, and a list of recent platform activity. The counts and the case on these screens are sample data from the prototype and are not measurements.

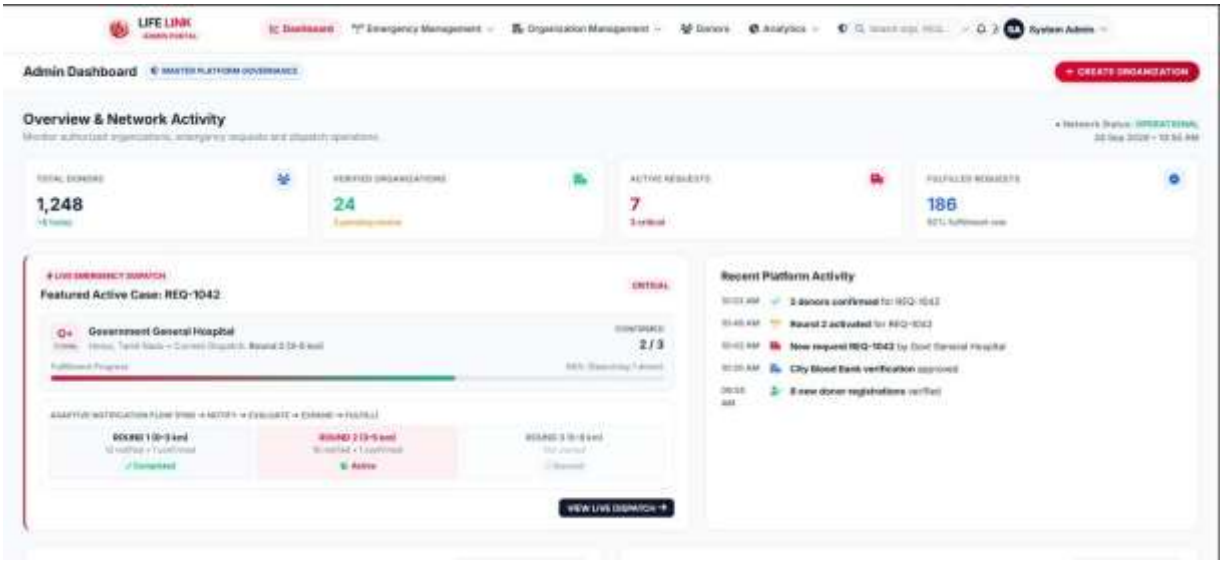


Fig. 4. Admin dashboard of the LIFE LINK prototype.

Fig. 5 shows the donor portal. The home page greets the donor and shows the blood group, the city and a verified-donor label. An availability panel shows the last donation date, the date from which the donor is next eligible and a button to pause donation requests, and a section titled "Someone nearby needs your blood" lists the open requests relevant to the donor's blood group. When no request matches, the page says so instead of showing unrelated requests. The tabs Home, My Requests, My Donations, Referrals and Recognition, Notifications and Profile give access to the other functions.

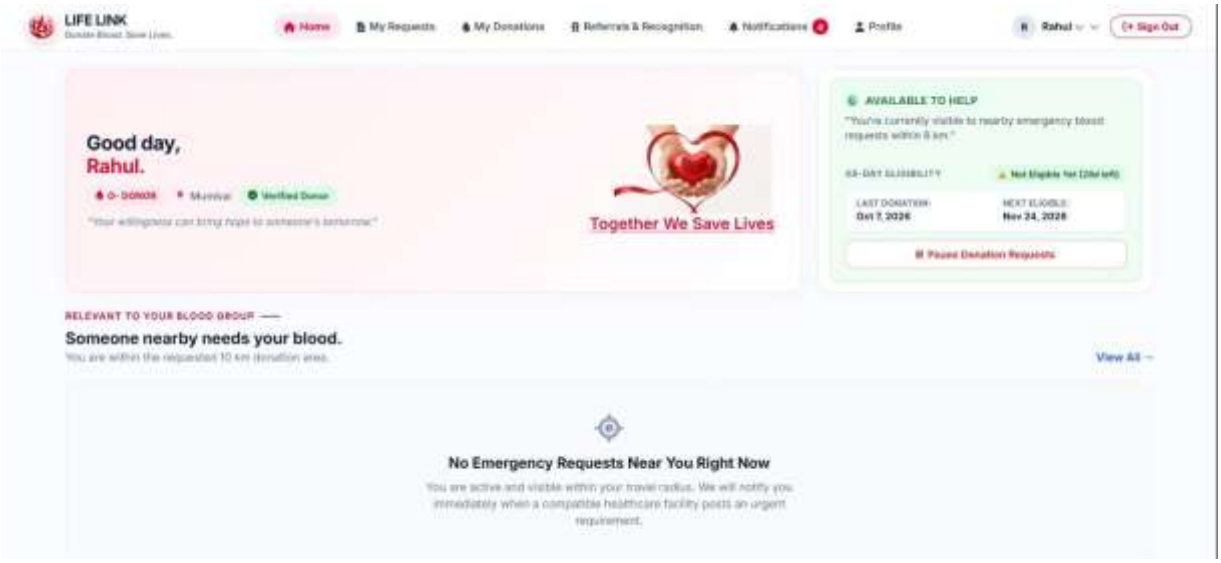


Fig. 5. Donor portal home page.

Fig. 6 shows the organisation portal, a dispatch control centre with four cards (active requests, donors notified, confirmed donors and fulfilment rate) and an adaptive dispatch panel. The panel presents the radius as three rounds: Round 1 for 0–3 km, Round 2 for 3–5 km and Round 3 for 5–8 km, each with its own state. The source code listed in the Appendix applies the full 8 km radius in a single step, so the staged rounds belong to the interface design and are listed as future work for the back end.

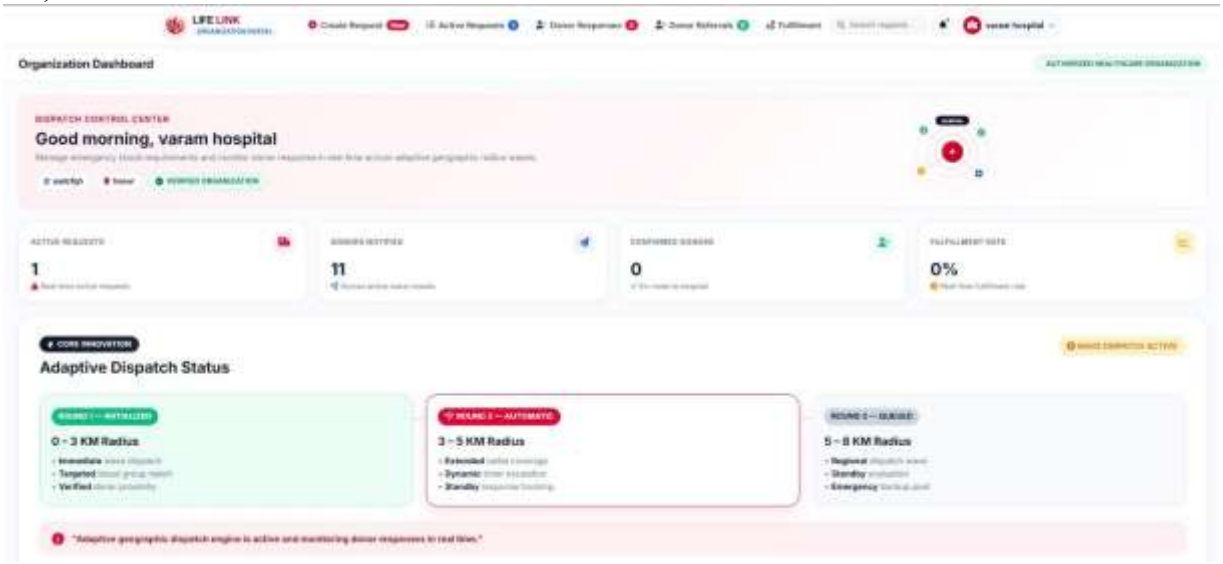


Fig. 6. Organisation portal with the adaptive dispatch status.

B. Features and the evidence for each

Table III lists the behaviours of the prototype together with the evidence in this paper. “Source code” means that the behaviour is present in the code listed in the Appendix. It does not mean that the behaviour has been measured under a test load.

TABLE III. FEATURES OF THE PROTOTYPE AND THE EVIDENCE FOR EACH

Feature	Behaviour	Evidence
Registration	Required fields, password of at least 8 characters, duplicate email rejected	Source code

Feature	Behaviour	Evidence
Login and token	One generic error for a wrong login; token expires after 2 h	Source code
Availability and location	Availability must be true or false; coordinates must be in range	Source code
Request creation	Invalid details rejected; request stored as Open	Source code
Donor selection	Same group, available, within 8 km of the hospital	Source code
Nearby view	Open requests of the donor's group, nearest first	Source code, Fig. 5
Status update	Only Open, Fulfilled or Cancelled accepted	Source code
Donation history	Future dates rejected; newest first	Source code
Role portals	Donor, organisation and admin screens	Figs. 2, 4–6
Staged dispatch rounds	0–3, 3–5 and 5–8 km rounds shown	Figs. 4, 6 (interface only)

C. Comparison with the manual process

In the manual process the person asking for blood must contact donors one by one, and every message goes to people regardless of blood group or distance. In the proposed system the selection is a database query on three attributes, and the donor sees only the requests that pass it. The medical decisions are unchanged, because eligibility and compatibility are still decided by medical staff. What changes is how the candidates are found. A numerical comparison of the time to find a donor would require the measurements described in Section VII, and no figure is claimed here.

VI. DISCUSSION

A. Observations

Three points stand out from the prototype. First, filtering on blood group, availability and distance removes most of the noise of a broadcast message, and the donor list stays short enough to read on a phone. Second, role-based portals keep each screen small. A donor never sees organisation actions, and an organisation sees the response to its own requests, so the risk of acting on the wrong item is low. Third, the organisation states the urgency and the system only selects and displays. This keeps the clinical judgement with the hospital and the clerical work of finding candidates with the software. The distance is a straight-line distance, so a donor 7 km away across a river may take longer to arrive than one 9 km away on a direct road, which is a reason to treat the radius as a filter and not as a promise of arrival time.

B. Security and privacy considerations

A donor record contains a name, a blood group, contact details and a location, which together reveal where a person lives or moves. The system limits access by role and by token, hashes passwords with bcrypt [16] and expires tokens after two hours. A production deployment also needs encrypted transport (HTTPS), rate limiting on login and on one-time-password requests, and consent for location sharing. The reference code checks that the caller is authenticated when a request is created or its status is changed, but it does not yet check the caller's role, so any signed-in user could create a request. A role check is required, together with verification of organisations before they can raise requests, because a false request would send donors to a hospital without need. The contact details of a donor should not be shown to a requester before the donor agrees. Finally, the system must be described as an information-support tool: donor screening, eligibility and compatibility testing belong to qualified medical personnel and authorised blood services [1].

VII. LIMITATIONS AND NEXT MEASUREMENTS

A. Known limitations

The limits of this work are stated so that the results are not read as more than they are.

1. The system has been demonstrated on a prototype with sample data. No response time, matching accuracy or number of concurrent users has been measured.
2. The sign-in page shows mobile-number and one-time-password login for donors, while the listed back end implements email and password. The two must be aligned, and a second factor is needed for real use.
3. The reference code checks authentication but not the caller's role when a request is created or its status is changed.
4. The listed code applies one fixed 8 km radius and orders the nearby list by distance only. The staged dispatch rounds and any use of urgency to order or widen the search are shown in the interface but are not in the listed code.
5. Notifications are stored in the database and can be read in the application. No push message, SMS or email is sent yet.
6. The interface shows an eligibility interval after a donation, but the listed code does not compute or enforce it. Equal blood group labels are not compatibility, and donor screening is outside the system.
7. The system has not been evaluated by donors or hospital staff, so usability claims are limited to the design intention.

B. Proposed evaluation protocol

Three experiments are planned. The first checks the correctness of matching on a test set of donors and requests with known coordinates and blood groups placed at several distances, including points just inside and just outside 8 km and donors with no stored location, and compares the donors selected by the system with a hand-computed answer. The second is a load test in which the number of donor records and of simultaneous requests is raised in steps, reporting the median and the 95th percentile response time of the request-creation and nearby-request endpoints and the number of failed calls. The third is a usability study with donors and hospital staff using a standard instrument such as the System Usability Scale (SUS), reported with the number of respondents. A field trial with a blood bank, measuring the time from a request to the first confirmed donor against the phone-call method, would follow. Only after these have been run can the system be described as validated for anything beyond a demonstration.

VIII. CONCLUSION AND FUTURE WORK

The Blood Donor Management System, LIFE LINK, joins the donor record, the emergency request, the location-based selection of donors, the notification and the donation history into a single workflow. By keeping all of them in one database and offering a separate portal to each role, it replaces scattered messages with a structured request that a donor can read at a glance: the blood group, the units, the hospital, the distance and the urgency. The prototype is built with HTML, CSS and JavaScript, Node.js with Express, and MongoDB, and was demonstrated through its sign-in, donor, organisation and admin screens. The measurements of matching accuracy, response time and usability are the next thing to prove.

The next steps are to carry out the three experiments in Section VII, to align the donor sign-in with a one-time password and add a role check on request creation, to implement the staged dispatch rounds and urgency-based ordering in the back end, to add push, SMS or email alerts, to add interactive maps, to connect with authorised hospitals and blood banks with donor and request verification, to send donation reminders based on medical guidance, to provide a mobile or progressive web application, and to move the system from a development environment to a secured, hosted deployment.

ACKNOWLEDGMENT

The authors thank Ms. V. Anbu Priyatharsini, Assistant Professor, Department of Computer

Science and Engineering, for her guidance as supervisor, Dr. G. Fathima, Professor and Head of the Department, Dr. R. Radhakrishnan, Principal, and the faculty of Adhiyamaan College of Engineering for the facilities and support provided during this work.

REFERENCES

[1] World Health Organization, "Blood safety and availability," Fact sheet. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/blood-safety-and-availability

[2] T. Kala, "A Location-Based Blood Donor Locator System Using Intelligent Matching Methodology," *Int. J. Innov. Res. Comput. Technol.*, 2023. [Online]. Available: https://www.ijirct.org/viewPaper.php?paperId=2603013

[3] V. Kamble, G. Chaudhari, and P. Thakur, "Location Based Blood Donor Finding System," *Int. J. Res. Appl. Sci. Eng. Technol.*, 2024, doi: 10.22214/ijraset.2024.63514.

[4] B. A. Adeleke, A. A. Awoseyi, and K. O. Ojetunde, "A Geolocation-Integrated Emergency Blood Donation Response System (GI-BDERS)," *Zenodo*, Nov. 2025, doi: 10.5281/zenodo.17607941.

[5] B. M. Nandini, S. B. Sharmila, A. Samantri, D. G., K. Bhat, and N. C. Nagarjun, "BloodConnect: Proximity Based Blood Donation Network with Real-Time Alerts and Tracking," *J. Sci. Ind. Res.*, vol. 84, no. 12, pp. 1291–1300, Dec. 2025, doi: 10.56042/jsir.v84i12.20668.

[6] J. C. Nwaukwa, E. C. Isok, M. I. Adaora, and I. Izegbu, "Blood Donation Management System using K-Nearest Neighbor," *J. Appl. Comput. Sci. Math.*, 2023, doi: 10.4316/JACSM.202302002.

[7] K. Deepa, C. Aarthy, and N. Udaya Kumar, "AI-Based Donor Connect Platform using Blood Type Matching and Real-Time Emergency Location Detection," *Int. J. Eng. Extended Technol. Res.*, 2026, doi: 10.15662/IJEETR.2026.0802042.

[8] R. Nithya and R. Maheshwari, "Blood Donation Management System Using Data Driven Matching," *Int. J. Res. Academic World*, 2026. [Online]. Available: https://academicjournal.ijraw.com

[9] G. Maji, S. Ghosh, A. Banerjee, A. Banerjee, and P. Ghosh, "Location Aware Blood Management System for Donation Camps and Emergency Demands," *Trans. Indian Natl. Acad. Eng.*, 2026, doi: 10.1007/s41403-026-00584-4.

[10] "Smart Blood Inventory and Real-Time Donor Clustering and Geospatial Visualization," *Int. J. Novel Trends Innov.*, 2026. [Online]. Available: https://rjpn.org/IJNTI/viewpaperforall.php?paper=IJNTI2603118

[11] "LifeLink: An Intelligent Privacy-Preserving Blood Donation Social Network," *Int. J. Emerg. Technol. Innov. Res.*, 2026. [Online]. Available: https://www.jetir.org/papers/JETIR2608399.pdf

[12] OpenJS Foundation, "Node.js documentation" and "Express.js documentation." [Online]. Available: https://nodejs.org and https://expressjs.com

[13] MongoDB, Inc., "MongoDB manual." [Online]. Available: https://www.mongodb.com/docs/manual/

[14] Mozilla, "Geolocation API," *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API

[15] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Univ. California, Irvine, CA, USA, 2000.

[16] OWASP Foundation, "Password storage cheat sheet," *OWASP Cheat Sheet Series*. [Online]. Available: https://cheatsheetseries.owasp.org

[17] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," IETF, RFC 7519, May 2015.

[18] R. W. Sinnott, "Virtues of the haversine," *Sky and Telescope*, vol. 68, no. 2, p. 159, 1984.