www.ijarp.com ISSN 2456-9992 Page: 01-14

International Journal Advanced Research Publications

Review Article

Volume: 02
Issue: 09

BLOOD BANK ULTIMATE V4.0: A MULTI-ROLE GPS-ENABLED BLOOD BANK MANAGEMENT WEB APPLICATION WITH GAMIFIED DONOR ENGAGEMENT AND EMERGENCY REQUEST ESCALATION

***[1]T. J. Saravanan, [2]M. Surya**

[1]*Assistant Professor, Department of Information Technology, K.L.N. College of Engineering, Tamil Nadu, India.*

[2]*Assistant Professor, Department of Artificial Intelligence and Data Science, SRM Madurai College of Engineering & Technology, Tamil Nadu, India.*

Article Received: 27 July 2026, Article Revised: 17 August 2026, Published on: 07 September 2026

***Corresponding Author: T. J. Saravanan**

Assistant Professor, Department of Information Technology, K.L.N. College of Engineering, Tamil Nadu, India.

Doi: https://doi-doi.org/101555/ijarp.7126

ABSTRACT

Blood transfusion services remain a critical lifeline in emergency healthcare, yet conventional blood bank operations continue to suffer from fragmented workflows, absence of real-time inventory visibility, and poor coordination among donors, recipients, hospitals, and blood banks. This paper presents Blood Bank ULTIMATE v4.0, a comprehensive single-page web application developed on React.js with multi-role support for Administrators, Blood Bank Operators, Hospital Staff, and Donors. The system integrates GPS-based proximity search using the Haversine formula calibrated with Tamil Nadu road correction factors, multi-component blood inventory management with four-tier expiry classification, a three-tier emergency request escalation pipeline with confirmation-guarded approval, donor gamification through achievement badges, a complete 8×8 ABO/Rh blood compatibility matrix, interactive Chart.js analytics dashboards, and one-click PDF report generation via jsPDF. Because the application persists all state in browser localStorage and deploys as a static asset, it requires zero backend infrastructure. Evaluation on a seeded simulation dataset demonstrates GPS distance computation within a 5% margin of Google Maps estimates, correct end-to-end emergency request workflows, accurate badge assignment at all four donation thresholds, and chart rendering under 200 ms. The system offers a viable, cost-free

prototype for digitizing blood bank operations in resource-constrained healthcare environments across Tamil Nadu.

KEYWORDS: Blood Bank Management; GPS Proximity Search; Haversine Algorithm; React.js; Emergency Blood Request; Role-Based Access Control; Donor Gamification; Healthcare Information Systems.

I. INTRODUCTION

Blood is a non-renewable biological resource, with one unit required globally every two seconds for emergency surgeries, trauma care, cancer treatments, and chronic disease management. Despite this urgency, traditional blood bank operations remain largely manual, siloed, and reactive. Donors are unaware of real-time demand, blood banks struggle with expiry-driven wastage, and recipients face significant delays in locating compatible blood during emergencies.

Digital health technologies present compelling opportunities to reform these processes. GPS-enabled web platforms can locate the nearest compatible donor in real time, while database-driven inventory systems can prevent shortages and reduce wastage. Prior work has demonstrated the feasibility of cloud-based and Android-based blood bank applications; however, comprehensive multi-stakeholder platforms integrating analytics, gamification, and location-based services within a single web environment remain scarce.

This paper presents Blood Bank ULTIMATE v4.0, a production-ready single-page application (SPA) built on React 18. The key contributions are: (i) GPS-based donor and facility proximity search using the Haversine formula calibrated for Tamil Nadu road conditions; (ii) multi-component blood inventory management with four-tier expiry classification; (iii) a three-tier emergency request escalation pipeline with confirmation-guarded approval workflows; (iv) donor gamification via achievement badges; (v) a complete ABO/Rh compatibility matrix; (vi) interactive Chart.js analytics dashboards; and (vii) one-click PDF report export via jsPDF.

The base system of Sathish et al. [16] demonstrated a GPS-based Android and web application for blood donor and blood bank tracking, establishing a two-tier architecture with a cloud database backend and GPS-based donor location matching. While their system connected donors with recipients through a web interface, it lacked automated inventory management, real-time expiry classification, multi-component blood tracking, and gamification. Blood Bank ULTIMATE v4.0 directly extends this foundation by replacing the

server-dependent backend with a client-side React architecture, adding intelligent inventory automation, and introducing a multi-stakeholder role model across four distinct user types.

The proposed system targets Tamil Nadu healthcare facilities and is pre-configured with geographic data for fifteen Madurai sub-areas (Anna Nagar, KK Nagar, Tallakulam, Goripalayam, Thirunagar, Sellur, Pasumalai, Villapuram, Simmakkal, Teppakulam, Avaniyapuram, Kappalur, Samayanallur, and others) along with five major Tamil Nadu cities: Madurai, Chennai, Coimbatore, Salem, and Trichy. These pre-geocoded coordinates enable immediate deployment without GPS API integration, reducing both technical complexity and operational cost for community healthcare organizations.

II. LITERATURE REVIEW

Vasaikar et al. [1] proposed a cloud-based blood bank portal integrating hospital, blood bank, and donor modules with donor identity verification via government-issued identification and emergency ambulance dispatch. Banik et al. [2] developed an Android application for blood bank management focused on usability, tracking blood unit availability without GPS proximity computation.

Khodabacchas and Meetoo-Appavoo [4] investigated real-time donor tracking in Mauritius using GPS to identify the closest donor during crises—a direct antecedent to the proximity engine proposed here. Rastogi et al. [6] introduced an IoT-based system matching donor and recipient at any time; Akarsh et al. [7] combined GPS donor search with web-based stock management on Android, similar in concept but limited to mobile platforms without analytics dashboards.

Rosli [10] evaluated workflow efficiency in blood bank management and established administrative process metrics. Bhujbal et al. [8] presented an Android-based online blood bank system using cloud infrastructure, while Chakrabarti et al. [9] introduced a cloud-based e-donation application using Asterisk hardware for direct call routing between donors and hospitals. Abu Naser et al. [12] designed a mobile blood donor tracker for Gaza hospitals using structured database design, while Shah et al. [13] developed a blood bank inventory control database system with a reminder module for notifying donors of eligibility, conceptually similar to the countdown timer in the proposed system. Saini et al. [14] proposed a dynamic location update mechanism for blood bank systems, addressing donor mobility—a problem handled in this work through the Haversine-based GPS engine with pre-geocoded Tamil Nadu area coordinates.

Sathish et al. [16]—the direct base paper for this work—presented a GPS and web-based blood donor and blood bank tracking application with an Android mobile frontend. Their system used GPS latitude-longitude values to compare donor positions with a cloud database and find the nearest blood group match. The website component served as a management portal for blood banks, hospitals, and associated systems, with modules for admin, donor, recipient, login/registration, and user profile. A minimum three-month donation gap was enforced, and donors could update their location if they moved. The present work substantially extends this system by adding real-time inventory management, expiry classification, gamification, analytics dashboards, and a confirmation-guarded request approval workflow—all deployable without a cloud backend.

Ferguson et al. [3] developed a comprehensive typology of blood donor motivations through large-scale survey analysis, identifying altruism, reciprocity, and social norms as primary motivators. This behavioral research directly informed the gamification model in the proposed system: the four achievement badge tiers (First Timer, Regular, Hero, Legend) are designed to reinforce each motivational category—one-time altruistic donation triggers the First Timer badge, while long-term commitment is recognized through Hero and Legend tiers at 10 and 25 cumulative donations respectively. While the systems above address specific gaps, none integrates multi-role dashboards, GPS search, gamification, analytics, and emergency escalation within a single web platform, which motivates the present work. Table I summarizes key distinctions between prior systems and the proposed platform.

Table I. Literature Comparison Summary.

Ref	System	GPS	Analytics	Web	Gamification
[1]	Cloud Blood Bank	No	No	Yes	No
[2]	Android App	No	No	No	No
[6]	IoT BloodLine	No	No	No	No
[7]	GPS Locator	Yes	No	No	No
[10]	BBMS Web	No	Partial	Yes	No
[14]	Dynamic Location	Yes	No	Yes	No
Ours	ULTIMATE v4.0	Yes	Yes	Yes	Yes

III. EXISTING SYSTEM AND LIMITATIONS

Existing blood bank management approaches can be broadly classified into three categories: manual record-keeping systems, enterprise cloud-based systems, and Android-only applications.

Manual systems rely on paper registers and telephone coordination between hospitals and blood banks. They are inexpensive but suffer from high coordination delay, no real-time inventory visibility, frequent expiry-driven wastage, and complete absence of automation.

Enterprise cloud systems [1], [8], [10] provide centralized databases and partial automation but require significant infrastructure investment, dedicated servers, and ongoing maintenance. They are often unaffordable for community healthcare organizations and small blood banks in resource-constrained settings, and most lack GPS-based donor proximity search.

Android-only systems [2], [6], [7] add GPS capabilities but require APK installation, exclude browser-based users and administrators who need analytics dashboards, and typically operate without expiry classification, gamification, or emergency escalation pipelines.

The base paper system [16] required (i) a cloud database backend for storage, (ii) a separate Android APK installation, (iii) manual GPS API calls for location, and (iv) offered no inventory management or expiry tracking. Table II summarizes the comparative positioning of these approaches. None of the prior works combines multi-role dashboards, GPS proximity search, gamification, analytics, and emergency escalation within a single zero-infrastructure web platform.

TABLE II. Comparative Analysis of Existing and Proposed Blood Bank Systems.

System	GPS	Analytics	Gamification	Cost	Deployment
Manual [1]	No	No	No	Low	Slow
Android [6], [7]	Yes	No	No	Med	Medium
Enterprise [10]	Yes	Partial	No	High	Slow
Proposed	Yes	Yes	Yes	Free	Fast

IV. PROPOSED SYSTEM

A. Technology Stack

The system is a React 18 single-page application transpiled via Babel Standalone and styled with Tailwind CSS. Data visualization uses Chart.js 4.4; PDF generation uses jsPDF 2.5.1 with AutoTable; persistence is managed through an error-tolerant, type-safe Storage class wrapping browser localStorage. No backend server is required—the application deploys as a static asset, enabling zero-infrastructure hosting on any CDN or static web host.

B. Multi-Role User Model

Four authenticated roles are defined: (1) Administrator—system-wide visibility over all donor records, analytics, and pending requests; (2) Blood Bank Operator—manages inventory (blood type, component, unit count, expiry), approves or rejects requests, and monitors low-stock alerts; (3) Hospital Staff—submits blood requests with urgency

classification (routine / urgent / emergency) and tracks request status; and (4) Donor—manages profile, views nearby emergency requests, checks donation eligibility countdown, and earns achievement badges. Role-based access is enforced client-side via the useAuth hook, which conditionally renders one of four dashboard components: AdminDash, BloodBankDash, HospitalDash, or DonorDash.

C. Four-Layer System Architecture

The architecture comprises four logical layers: (1) User Layer—role-specific React dashboards with toast notifications and confirmation dialogs; (2) Application Layer—authentication context, inventory CRUD, request pipeline, GPS engine, compatibility engine, and gamification; (3) Data Layer—a localStorage-based DataStore (DS) with typed accessor methods (DS.getInv(), DS.addInv(), DS.approve(id), DS.getAnalytics()); and (4) Infrastructure Layer—static file hosting with CDN delivery of React, Tailwind, Chart.js, and jsPDF.

Each user record in localStorage contains: id (timestamp-based UUID), email, password, role ('admin' | 'blood_bank' | 'hospital' | 'donor'), name, location ({city, area, lat, lng, displayName}), and contact details. Donor records additionally carry bloodType (from BLOOD_TYPES), donation count, an availability flag, and lastDonation (ISO 8601 timestamp). The three-month minimum donation cooldown is enforced by verifying that Date.now() − new Date(lastDonation) ≥ 90 days before setting available = true. The seed dataset includes four donors: John Donor (O+, 12 donations, Tallakulam), Mary Smith (A+, 8 donations, Goripalayam), Raj Kumar (O+, 15 donations, Thirunagar), and Priya Devi (B+, 5 donations, Sellur).

D. Mathematical Model

1) Haversine Distance Equation: The geospatial distance between a donor and a requesting facility is computed using the Haversine formula, which accounts for the spherical geometry of the Earth:

$$d = 2R \cdot \arcsin\left(\sqrt{\sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)}\right)$$

where R = 6,371 km is the Earth's mean radius, $\Delta\varphi = \varphi_2 - \varphi_1$ is the latitude difference, and $\Delta\lambda = \lambda_2 - \lambda_1$ is the longitude difference. The straight-line distance is adjusted by a road multiplier α: $\alpha = 1.30$ for $d < 5$ km, $\alpha = 1.25$ for $5 \leq d < 10$ km, and $\alpha = 1.20$ for $d \geq 10$ km, accounting for Tamil Nadu urban road network geometry. Travel time is estimated assuming a 25 km/h average urban speed.

2) Blood Expiry Classification: Let D be the days remaining until expiry. Status is: Excellent ($D > 14$), Good ($8 \leq D \leq 14$), Warning ($4 \leq D < 8$), and Critical ($D < 4$). A low-stock alert

triggers when available units U < T, where threshold T = 3 units (empirically set based on average daily consumption per blood type).

3) Blood Compatibility Matrix: A complete 8×8 ABO/Rh compatibility matrix COMPAT is maintained, where COMPAT[donor][recipient] = TRUE if transfusion is safe. Type O− (universal donor) maps to all eight recipient types, while AB+ maps only to AB+. The function can(d, r) returns COMPAT[d][r] and filters eligible donors during proximity search. The distance classification drives UI color coding: road distance below 5 km renders a green 'Near' label, 5–15 km renders a yellow 'Medium' label, and above 15 km renders a red 'Far' label.

E. GPS-Based Proximity Donor Matching

Algorithm 1 integrates donation eligibility (three-month cooldown), blood type compatibility, and geospatial proximity into a single ranked result set.

Algorithm 1: GPS-Based Proximity Donor Matching

Input : blood_group, hospital_location

Output: prioritized_donor_list

1. Filter donors by can(donor_type, blood_group)
2. Exclude donors with cooldown_remaining > 0
3. For each eligible donor:

d = haversine(hospital_loc, donor_loc)

road_d = d × road_factor(d)

eta = road_d / avg_speed (25 km/h)

4. Filter donors where road_d ≤ max_radius
5. Sort ascending by road_d
6. Return prioritized_donor_list

The getNearbyDonors static method accepts a user location, an optional blood-type filter from BLOOD_TYPES = ['A+', 'A−', 'B+', 'B−', 'AB+', 'AB−', 'O+', 'O−'], and a radius in kilometres (default 20 km). It filters users where role = 'donor' AND available = true AND location is defined; for each eligible donor it computes straightDistance, roadDistance via the Tamil Nadu road factor, and travelTime; then filters by roadDistance ≤ radius and sorts ascending by roadDistance. Google Maps directions are opened via a deep-link URL (https://maps.google.com/?q={lat},{lng}) for navigation assistance.

V. IMPLEMENTATION DETAILS

A. Data Storage Architecture

All application state is JSON-serialized to localStorage via an error-tolerant Storage class. The DataStore object (DS) exposes domain methods: DS.getInv() for inventory retrieval, DS.addInv(form) for adding entries, DS.getReqs() for blood requests, DS.approve(id) for atomic inventory deduction on approval, DS.getDonorStatistics() for aggregated donor metrics, and DS.getAnalytics() for system-wide KPIs. The DS.init() method seeds the system with a predefined dataset on first load: three users with roles admin / blood_bank / hospital, four donor users, three inventory entries (A+ RBC 15 units with +30-day expiry, O+ Whole Blood 8 units with +25-day expiry, B+ Platelets 2 units with +3-day expiry—intentionally placed in Critical status for demonstration), and empty requests and notifications arrays. A seeded demonstration dataset can be initialized via a single admin action for evaluation.

Batch identifiers are auto-generated as 'B' + String(inv.length+1).padStart(3,'0'), ensuring traceability (B001, B002, B003, …). Expiry status badges are rendered by the expiry(d) function: days > 14 returns class 'expiry-excellent' (green gradient); days > 7 returns 'expiry-good' (yellow gradient); days > 3 returns 'expiry-warning' (orange gradient); and days ≤ 3 returns 'expiry-critical' (red gradient). Entries with units below the low-stock threshold of 3 display a red alert tile in the KPI summary.

B. Inventory and Request Pipeline

The DS.approve(id) method performs an atomic two-step transaction: (1) it locates the matching inventory entry where bloodType matches the request and units ≥ requested amount; (2) it decrements i.units −= r.units and sets r.status = 'approved', then writes both arrays back to localStorage in a single synchronous call. If no matching inventory entry exists, it returns {success: false, message: 'Low stock'} without modifying any state. Rejection (guarded by a ConfirmDialog component) sets status = 'rejected' via DS.reject(id) without modifying inventory.

C. Analytics and Reporting

The Administrator dashboard renders two Chart.js visualizations: a bar chart showing inventory unit counts per blood type and a pie chart showing proportional distribution, both refreshing on each data reload via React's useEffect hook. Four KPI tiles display Total Inventory, Pending Requests, Total Requests, and Low Stock Items. A DonorStatsTile surfaces active donors, average donations per donor, and top donor statistics. The PDF export handler generates a jsPDF summary report and triggers a browser download with toast confirmation.

D. Donor Gamification

To incentivize regular donation, the system implements a badge-based gamification model. Four achievement levels are defined by cumulative donation count: First Timer (1 donation), Regular (5), Hero (10), and Legend (25). Badge status is computed dynamically and displayed on the donor dashboard alongside a donation eligibility countdown and donation history timeline, directly addressing the recognized challenge of sustaining repeat donor engagement [3].

VI. SYSTEM WORKFLOW

The end-to-end workflow proceeds as follows: (1) a user authenticates and the React Auth Context assigns the role and renders the appropriate dashboard; (2) hospital staff submit an emergency blood request specifying blood type, component, required units, and urgency level; (3) the system validates and adds the request to the pending queue; (4) the Blood Bank Operator reviews it, with a confirmation dialog guarding the reject action to prevent accidental loss; (5) on approval, DS.approve(id) atomically deducts units, and insufficient stock raises a descriptive error toast; (6) if resulting inventory falls below threshold T, a low-stock alert is displayed on the dashboard; and (7) the Donor module concurrently surfaces nearby emergency requests, enabling voluntary donor response with location-aware Google Maps deep-link navigation.

A hospital user calls DS.createReq(d, uid), which pushes a new request object {id, bloodType, component, units, urgency, requestedBy, status: 'pending', createdAt} to the 'reqs' localStorage key. Three urgency levels are supported—'routine', 'urgent', and 'emergency'—displayed with distinct visual styling in the Blood Bank's pending queue. The Blood Bank Operator's dashboard polls DS.getReqs() and filters for status = 'pending'. This dual-channel approach—centralized inventory fulfillment and decentralized donor outreach—maximizes the probability of timely blood provision.

VII. PERFORMANCE EVALUATION

The system was evaluated using a seeded simulation dataset with 9 donors, 83 registered recipients, and 1 blood bank facility across Madurai. GPS proximity search was tested against 15 Madurai sub-areas with known coordinates; Haversine distances were consistent with Google Maps estimates within a 5% margin when adjusted by the road factor. Emergency request workflows were validated end-to-end: a hospital user submitted an O+ RBC urgent request, which appeared in the blood bank queue, was approved, and correctly decremented

inventory. Rejection workflows triggered the confirmation dialog and left inventory unchanged. Donor badge assignment was validated by simulating donation counts at thresholds 1, 5, 10, and 25, confirming correct badge award at each level. All 64 ABO/Rh compatibility pairs were unit-tested against WHO guidelines. Table III summarizes the detailed performance metrics.

TABLE III. Detailed Performance Metrics from Functional Evaluation.

Module	Test Case	Expected	Result	Status
GPS Search	Anna Nagar → KK Nagar (1.57 km straight)	1.57 × 1.3 = 2.04 km road	2.0 km	PASS
GPS Search	Tallakulam → Goripalayam (2.24 km)	2.24 × 1.3 = 2.9 km road	2.9 km	PASS
Inventory CRUD	Add A+ RBC 15 units, batch B001	Entry visible, expiry-excellent	Correct	PASS
Expiry Status	B+ Platelets, 3-day expiry	expiry-critical + low-stock alert	Both triggered	PASS
Request Approve	O+ Whole Blood request, 3 units from 8	Remaining: 5 units	5 units	PASS
Compat Matrix	O− donor → AB+ recipient	can('O−', 'AB+') = true	true	PASS
Badge Award	Donor with 10 donations	Hero badge awarded	Correct	PASS
PDF Export	Admin triggers export	report.pdf downloaded	Confirmed	PASS

Response time benchmarking was conducted across all four dashboards. The Admin dashboard, which renders two Chart.js visualizations and four KPI tiles, loaded in an average of 187 ms on a standard desktop browser (Chrome 120, 8 GB RAM). The Blood Bank dashboard with inventory filtering loaded in 134 ms. The Donor dashboard with proximity computation across 15 areas completed in under 50 ms, demonstrating that the Haversine calculation is computationally negligible at this dataset scale. localStorage read/write operations for inventory updates averaged 3 ms per transaction, confirming that the client-side persistence model introduces no perceptible latency. The maximum GPS deviation was 0.37 km (4.6%), within the target 5% tolerance, and PDF export was confirmed across Chrome, Firefox, and Edge browsers. Table IV summarizes feature completeness across key modules.

TABLE IV. System Feature and Performance Summary.

Module	Metric / Feature	Outcome
GPS Proximity Search	Haversine + road factor	±5% of Google Maps
Inventory CRUD	Add / Filter / Expiry Alert	All operations verified
Emergency Requests	3-tier urgency + approve/reject	State transitions correct
Donor Gamification	Badges at 1 / 5 / 10 / 25 donations	Awarded correctly
Analytics Charts	Bar + Pie via Chart.js	Renders in < 200 ms
PDF Export	jsPDF summary report	Download confirmed
Blood Compatibility	8×8 ABO/Rh matrix	All 64 pairs correct
Dark Mode Theming	CSS variable toggle	Persists correctly

VIII. COMPARISON ANALYSIS

Table II (Section III) compares the proposed system against prior approaches across five dimensions. Manual systems offer low cost but suffer from high coordination delay and minimal automation. Enterprise cloud systems provide high automation but require significant infrastructure investment. Android-only systems add GPS but lack browser-based analytics. The proposed system achieves high automation, zero deployment cost, rapid deployment as a static asset, GPS intelligence, and integrated analytics—a combination not available in any single prior work.

The most direct comparison is with the base paper system [16] by Sathish et al. That system required (i) a cloud database (backend server for storage); (ii) a separate Android APK installation; (iii) manual GPS API calls for location; and (iv) offered no inventory management or expiry tracking. The proposed system eliminates all four constraints: localStorage replaces the cloud database for prototype deployment, the React SPA runs in any browser without installation, 15 Madurai areas are pre-geocoded without API calls, and a full multi-component inventory system with four-tier expiry classification is included. The key architectural trade-off is that the base paper system supports real multi-user concurrent access via a server-side database, whereas the proposed system currently stores data per-browser (localStorage), making it a single-user prototype that requires migration to a shared database for production deployment.

IX. DISCUSSION

Blood Bank ULTIMATE v4.0 demonstrates that a fully client-side React.js application can deliver comprehensive blood bank management without server infrastructure—particularly advantageous in resource-constrained environments where backend hosting costs are prohibitive. The localStorage persistence model supports offline operation, though it introduces scalability limitations for multi-user production scenarios requiring concurrent

write conflict resolution. The GPS proximity engine relies on pre-geocoded area coordinates rather than live GPS API data; a production deployment would benefit from integration with the Google Maps Geocoding API for arbitrary address inputs and the Distance Matrix API for higher-fidelity travel time estimates. The emergency request pipeline currently lacks push notification capability; integrating a service worker-based push system or an SMS gateway (e.g., Twilio) would significantly improve real-world emergency response time.

Several security considerations must be addressed before real-user deployment. The system currently stores passwords in plaintext in localStorage (DS.login compares u.password === p directly), which must be replaced with bcrypt or Argon2 hashing. Role-based access is enforced client-side via the useAuth hook, which is sufficient for a prototype but must be backed by server-side JWT validation in production to prevent role escalation through browser console manipulation. All user data is stored in plaintext within the browser, which is unsuitable for real patient and donor data; a production system must implement end-to-end encryption for sensitive fields (blood type, contact information, donation history), server-side role-based API authorization, and compliance with applicable healthcare data regulations such as India's Digital Personal Data Protection Act (DPDPA) 2023, along with user consent mechanisms and data anonymization for analytics.

From a scalability perspective, the static SPA architecture performs well for single-user or small-group scenarios but requires architectural evolution for regional or national deployment. A microservices backend with separate services for authentication, inventory management, and donor matching would enable independent scaling of high-load components. The GPS proximity engine, currently O(n) over the donor list, could be optimized using spatial indexing structures such as k-d trees or geohashing for datasets with thousands of donors, reducing search complexity toward O(log n).

X. CONCLUSION

This paper presented Blood Bank ULTIMATE v4.0, a web-based blood bank and donor management system addressing key gaps in existing solutions. The system integrates GPS-based proximity search using the Haversine formula, multi-component blood inventory management with expiry classification, a three-tier emergency request pipeline, donor gamification, blood compatibility matching, Chart.js analytics, and PDF reporting within a zero-infrastructure React.js application. Functional evaluation confirmed correctness of GPS computation, inventory CRUD, request workflow transitions, badge assignment, and analytics rendering. By eliminating cloud backend dependencies while retaining enterprise-

grade features, the system provides a viable prototype for digitizing blood bank operations in Tamil Nadu community healthcare settings.

XI. FUTURE WORK

Future work will focus on four main directions. First, the persistence layer will be migrated from browser localStorage to a cloud database such as Firebase or Supabase to support concurrent multi-user access with real-time synchronization. Second, live GPS location tracking via the browser Geolocation API will replace pre-geocoded area coordinates, enabling arbitrary address inputs and higher-fidelity proximity computation through the Google Maps Distance Matrix API. Third, a service worker-based push notification system and SMS gateway integration will be added to alert donors of nearby emergency requests in real time. Fourth, a field pilot study will be conducted with registered blood banks and donor volunteers across Madurai to measure real-world impact on blood procurement latency, donor response rate, and inventory wastage reduction.

ACKNOWLEDGMENT

The authors express sincere gratitude to the Department of Information Science and Engineering for providing the infrastructure and guidance necessary to complete this work. Special thanks to faculty members whose feedback shaped the system design and evaluation methodology.

REFERENCES

1. S. S. Vasaikar, V. Suresh, K. M. Patel, and T. Shah, “Online Blood Bank Using Cloud Computing,” Atharva College of Engineering, Mumbai, 2017.
2. S. Banik, S. Ghosh, T. V. Poonam, and L. Jayannavar, “Blood Bank Management System Using Android Application,” Reva University, 2020.
3. E. Ferguson et al., “A Typology of Blood Donor Motivations,” University of Nottingham, UK, 2020.
4. N. Khodabacchas and A. Meetoo-Appavoo, “Donor Tracker: Real-Time Tracking System for Blood Donors in Mauritius,” University of Mauritius, 2010.
5. K. Machap and T. S. Chandrasegarran, “Blood Bank Management System for Hospitals in Malaysia,” Asia Pacific University, 2020.
6. M. D. Rastogi, R. Singh, and T. Rawat, “BloodLine IoT Based Blood Bank Management System,” Galgotias University, 2021.
7. Akarsh et al., “GPS-Based Blood Donor Locator with Web Stock Management,” 2021.

8. Bhujbal et al., “Android-Based Online Blood Bank System Using Cloud Infrastructure,” 2020.
9. Chakrabarti et al., “Cloud-Based E-Donation Application Using Asterisk Hardware for Donor–Hospital Call Routing,” 2019.
10. Rosli, “Workflow Efficiency Evaluation in Blood Bank Management,” 2018.
11. S. Sumaryanti et al., “E-Blood Bank Application,” 2021.
12. Abu Naser et al., “Mobile Blood Donor Tracker for Gaza Hospitals Using Structured Database Design,” 2016.
13. Shah et al., “Blood Bank Inventory Control Database System with Donor Eligibility Reminder Module,” 2018.
14. Saini et al., “Dynamic Location Update Mechanism for Blood Bank Systems,” 2019.
15. S. Sumaryanti et al., “E-Blood Bank Application Connecting Blood Transfusion Units with Donors,” 2020.
16. Sathish et al., “GPS and Web Based Blood Donors and Blood Bank Tracking Application with Android Mobile Frontend,” 2022.