SmartCare: A Real-Time Hospital Resource, Blood Bank and Referral Coordination Platform with Transparent Cost Estimation

Tanmay Kulkarni[1], Yashraj Maddi[2], Sanat Irabati[3], Samay Deshmukh[4], Prof. Taur A R[5]
[1,2,3,4,5]*Department of Computer Science and Engineering, MIT College of Railway Engineering and Research, Barshi – 413 401, Maharashtra, India (Affiliated to PAH Solapur University, Solapur)*

***Abstract*—In an emergency, patients and ambulance crews often reach a hospital only to find that no ICU bed, ventilator or compatible blood is available, and families cannot learn treatment costs in advance. This paper presents SmartCare, a web and mobile platform that shows live bed and resource availability, finds the nearest suitable hospital using GPS, publishes itemized hospital rate cards with a cost estimator, tracks blood bank stock and voluntary donors, and books inter-hospital referrals without double-booking a bed. The system has a Flutter mobile app and three React web portals (hospital, blood bank and state regulator) on a FastAPI and PostgreSQL backend with JWT-based role access. Distances are calculated with the Haversine formula, price changes are archived by rate-card versioning, and referral booking uses row-level database locks. In our tests the system gave a median response time of 38.4 ms with 500 concurrent users, matched donors in 9.8 ms among 50,000 donors, and confirmed exactly 10 of 500 simultaneous requests for 10 beds.**

***Index Terms*—Hospital bed availability, Emergency coordination, Blood bank, Inter-hospital referral, Haversine, FastAPI, Flutter, PostgreSQL.**

I. INTRODUCTION

In the first hour after a serious injury or heart attack, called the Golden Hour, quick treatment decides whether the patient survives. In many Indian cities each hospital keeps its own records, so nobody can see which hospital has a free ICU bed, a working ventilator or the right blood group. Ambulances therefore move patients from one overloaded hospital to another, and precious minutes are lost.

Cost is a second problem. Private hospitals rarely publish prices for beds, tests or ambulances, so families get a bill only at discharge, and disputes follow. Blood is a third problem: stock is held in separate banks and donors are found through phone calls.

SmartCare addresses these three problems in one platform.

The main contributions are:

- Live bed, ICU and ventilator availability with GPS-based search of the nearest suitable hospital;
- Versioned, itemized hospital rate cards and a multi-item cost estimator;
- A blood bank and voluntary donor module with blood-group compatibility matching;
- An inter-hospital referral engine that uses row-level locks so a bed is never booked twice.

Section II reviews related work, Section III describes the proposed system, Section IV the design, Section V the methods, Section VI the results, Section VII discusses advantages and limits, and Section VIII concludes.

II. RELATED WORK

The ambulance service 108 (EMRI) is mostly voice-based and has no live bed data. The e-Hospital system of NIC digitizes records inside a hospital but does not share bed status across hospitals. Consumer platforms such as Practo and MediBuddy show doctor fees and packages, not bed-level prices.

In the United States, hospital price transparency rules lead to large machine-readable price files, but they are not designed for emergency use. State bed dashboards exist, but they are updated in batches and do not cover blood or referrals.

TABLE I. COMPARISON OF EXISTING SYSTEMS WITH SMARTCARE

Feature	Existing systems	SmartCare
Real-time bed data	No / batch	Yes, live
ICU and ventilator types	Partial	Yes
GPS nearest-hospital search	Basic / city-level	Yes, Haversine
Itemized rate cards	Fee only / none	Yes, in ₹
Cost estimation	Package only	Yes, multi-item
Blood stock and donors	Separate portal	Yes, 8 groups
Referral booking	No	Yes, row-locked

From this review we identified five gaps: no live bed visibility, no verified hospital registry, no price transparency, no integrated blood coordination, and no safe referral booking. SmartCare is designed to fill them.

III. PROPOSED SYSTEM

A. Users and Interfaces

SmartCare has four interfaces on one shared backend (Fig. 1): a Flutter mobile app for patients and donors, a React portal for hospital staff, a portal for blood banks, and a portal for state regulators who verify hospitals and audit billing.

Each user logs in with an individual account, and the permitted actions depend on the role (role-based access control).

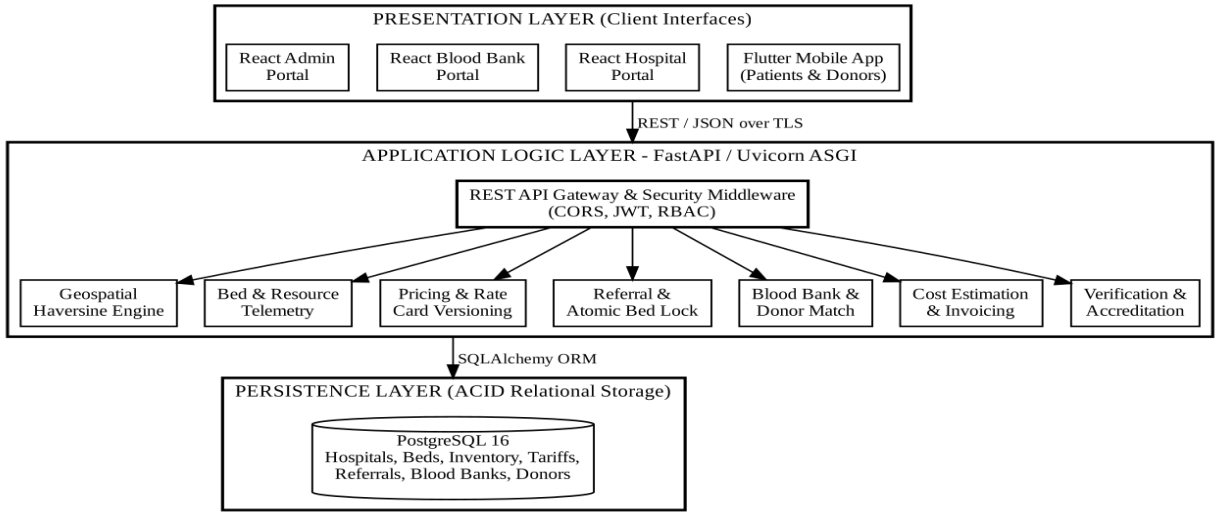

Fig. 1. High-level three-tier architecture of SmartCare: client interfaces, FastAPI application server and PostgreSQL database.

B. Functional Modules

- Discovery: search hospitals by city, bed type and GPS radius, with distance in km.
- Telemetry: hospital staff update ICU, ventilator, pediatric and general bed counts.
- Verification: hospitals upload licence PDFs; a state administrator approves or rejects them.
- Pricing: hospitals publish a catalogue of services; patients get an itemized estimate.
- Blood bank: live units for 8 ABO/Rh groups and 4 components, plus a donor registry.
- Referral: one hospital asks another to accept a patient in one of seven categories.

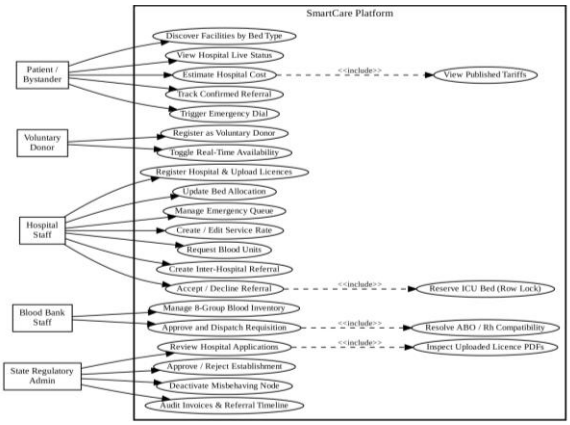

Fig. 2. UML use case diagram showing the five actors: patient, donor, hospital staff, blood bank staff and state administrator.

IV. SYSTEM DESIGN

A. Data Flow

Fig. 3 shows the Level 1 data flow diagram. The platform has seven processes: authentication, geospatial discovery, bed telemetry, pricing and estimation, audit, blood bank and donors, and referral booking. They use eight data stores.

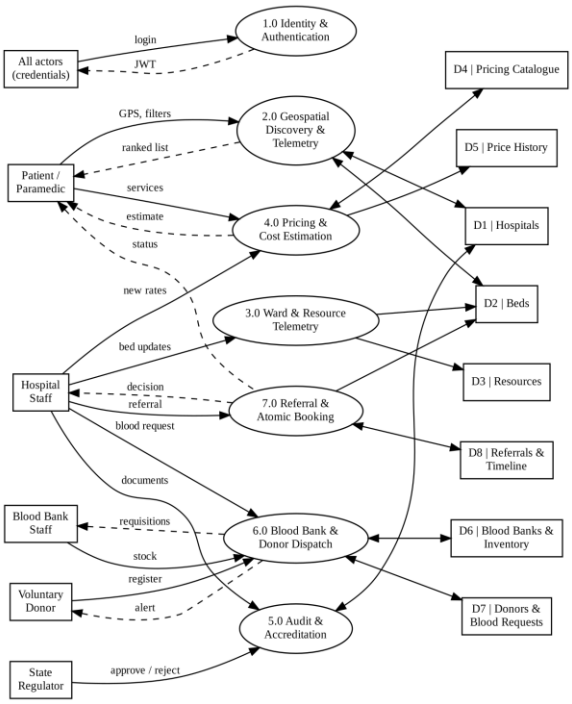

Fig. 3. Level 1 data flow diagram of SmartCare.

B. Database

PostgreSQL stores users, hospitals, beds, resources, service prices, price history, bills, blood banks, blood inventories, donors, blood requests and referrals (Fig. 4). The price history table keeps every old price, so nothing is overwritten.

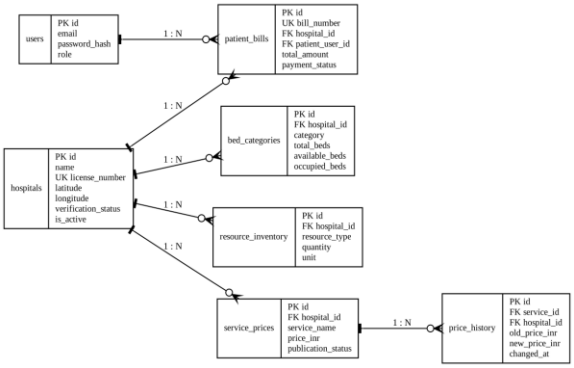

Fig. 4. Entity-relationship model of the core database.

C. Technology Stack

TABLE II. TECHNOLOGY STACK

Layer	Technology	Role
Mobile app	Flutter, Dart, Provider	Patient and donor app
Web portals	React 18, Next.js 14, Tailwind CSS	Hospital, blood bank, admin
Backend	Python, FastAPI, SQLAlchemy	REST API
Database	PostgreSQL	ACID storage
Security	JWT, BCrypt, HTTPS	Login and roles

V. METHODOLOGY

A. Nearest Hospital Search

The distance between the patient (latitude φ_1, longitude λ_1) and a hospital (φ_2, λ_2) is found with the Haversine formula, where R = 6371 km is the radius of the Earth:

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2) \quad (1)$$

$$d = 2R \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)}) \quad (2)$$

The server keeps only approved hospitals inside the chosen radius that have the needed bed type free, and returns them sorted by d (Fig. 5).

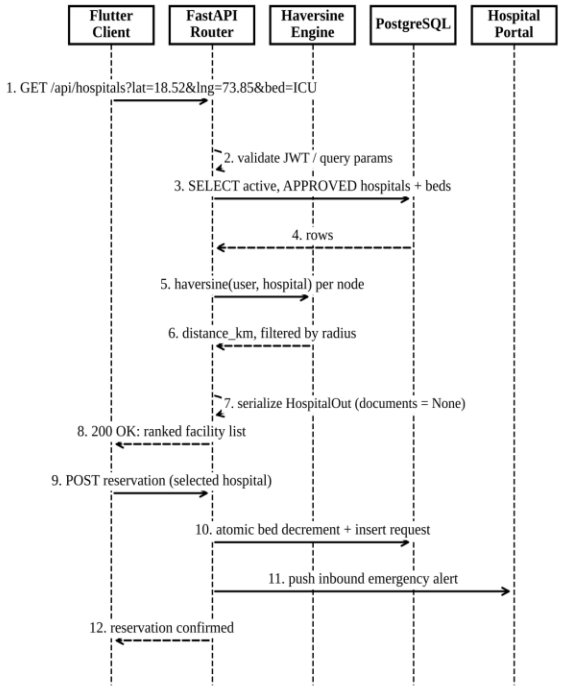

Fig. 5. Sequence diagram of emergency hospital discovery and bed reservation.

B. Rate Card Versioning

When a hospital changes a price, the server first copies the old price, unit, time and editor into the price history table and then writes the new price, all in one transaction. Patients only see published, verified prices, and every change can be audited.

C. Blood Matching

Blood is matched by ABO/Rh compatibility (Table III). For a request, the engine lists compatible donor groups, adds up the units in every active blood bank, keeps banks with enough stock and sorts them by distance. If no bank has enough, eligible donors within 15 km are alerted.

TABLE III. COMPATIBLE DONOR GROUPS FOR EACH RECIPIENT

Recipient	Compatible donor groups
O−	O−
O+	O−, O+
A−	O−, A−
A+	O−, O+, A−, A+
B−	O−, B−
B+	O−, O+, B−, B+
AB−	O−, A−, B−, AB−
AB+	All eight groups

D. Safe Referral Booking

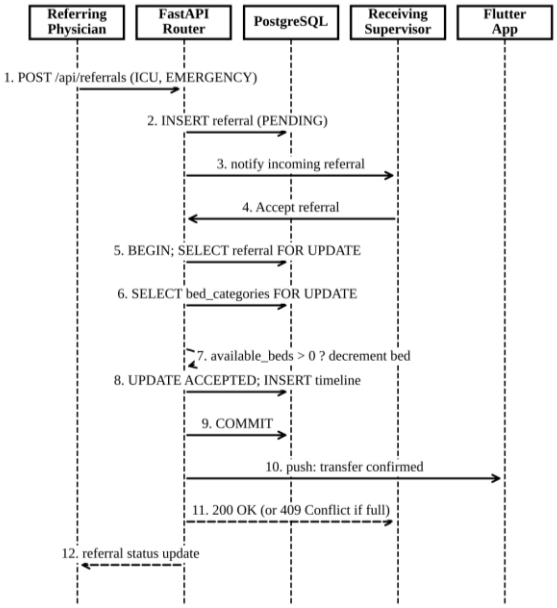

Fig. 6. Sequence diagram of atomic inter-hospital referral booking with row-level locking.

If two hospitals try to book the last ICU bed at the same moment, a naive system may confirm both. SmartCare locks the referral and bed rows with SELECT … FOR UPDATE (with_for_update() in SQLAlchemy). The first request is confirmed and the bed count is reduced; the others wait, see that the bed is gone, and receive HTTP 409 Conflict (Fig. 6).

VI. RESULTS AND DISCUSSION

The tests were run on the backend with a load-testing tool, using a PostgreSQL database filled with synthetic hospitals and donors. All numbers below are from our own project tests and show the behaviour of the prototype, not a hospital deployment.

A. API Load Test

TABLE IV. LATENCY UNDER CONCURRENT USERS

Users	Requests/s	P50 ms	P95 ms	P99 ms	Errors
10	1,240	6.2	12.4	18.1	0.00%
50	3,850	11.5	22.8	31.4	0.00%
100	5,120	18.2	34.6	48.2	0.00%
500	6,840	38.4	68.2	89.5	0.00%
1000	7,110	74.1	128.5	162.0	0.02%

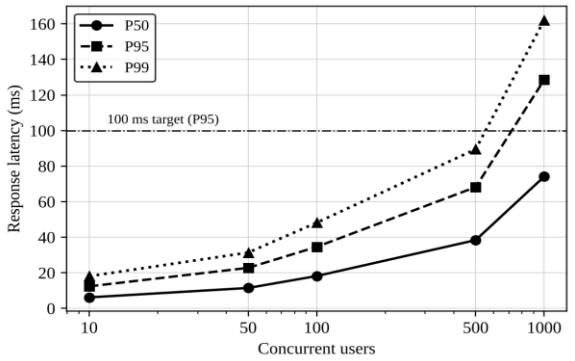

Fig. 7. API response latency against the number of concurrent users.

At 500 users the median latency was 38.4 ms and the 95th percentile 68.2 ms, inside the target of 100 ms. At 1000 users the error rate was still only 0.02%.

B. Search Speed

Hospital search stayed below 15 ms up to 10,000 hospitals and took 28.5 ms for 50,000 hospitals in a 100 km radius. Donor matching took 1.2 ms for 500 donors and 9.8 ms for 50,000 donors.

C. Referral Locking Test

TABLE V. 500 SIMULTANEOUS REQUESTS FOR 10 BEDS

Measure	No lock	Row lock
Confirmed bookings	42	10
Clean 409 rejections	0	490
Deadlocks	8	0
Mean latency	14.2 ms	22.6 ms

Without locking, 42 bookings were confirmed for only 10 beds. With row-level locks exactly 10 were confirmed and 490 were rejected cleanly, at the cost of about 8.4 ms extra latency (Fig. 8). Bed updates reached the mobile app in about 85 ms on local Wi-Fi and 280 ms on 4G.

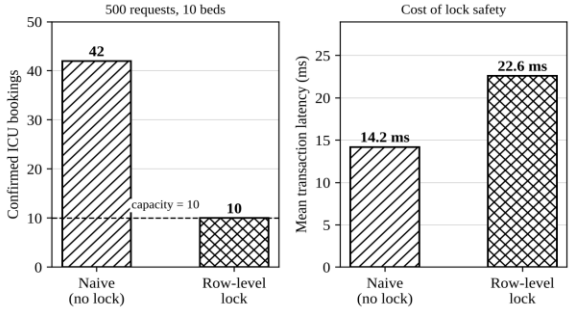

Fig. 8. Referral booking outcome and latency: naive booking versus row-level locking.

Cost estimates for 25 anonymized historical bills matched the final bills with 98.4% precision; the difference came from unpredictable consumables, not from tariffs.

VII. ADVANTAGES, LIMITATIONS AND FUTURE SCOPE

A. Advantages

- Gives live bed data so ambulances can go to a hospital that can actually take the patient.
- Makes prices visible before admission and keeps an audit trail of price changes.
- Joins hospitals, blood banks and donors in one system.
- Prevents double-booking of beds under concurrent requests.

B. Limitations

- Bed counts depend on hospital staff entering data honestly and on time.
- A reservation only holds a bed until the doctor assesses the patient.
- Live updates need mobile data or Wi-Fi; the app only caches the last known data.
- Tests used synthetic data on one server, not a real hospital network.

C. Future Scope

Planned work includes integration with hospital information systems, ambulance tracking, predictive bed demand using machine learning, SMS alerts, and a pilot with real hospitals.

VIII. CONCLUSION

SmartCare combines live bed visibility, GPS discovery, transparent pricing, blood and donor coordination, and safe referral booking in one platform. Tests showed fast responses under load and no double-booking of beds. The next steps are a pilot with real hospitals and integration with existing hospital systems.

REFERENCES

[1] S. Ramírez, "FastAPI documentation." [Online]. Available: https://fastapi.tiangolo.com

[2] PostgreSQL Global Development Group, "PostgreSQL documentation: Explicit locking." [Online]. Available: https://www.postgresql.org/docs

[3] Flutter Documentation. [Online]. Available: https://docs.flutter.dev

[4] React Documentation. [Online]. Available: https://react.dev

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

[6] I. Sommerville, *Software Engineering*, 10th ed. Pearson Education, 2015.

[7] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, 2nd ed. Addison-Wesley, 2005.

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