Contents

Acknowledgment

We extend our deepest gratitude to the Department of Computer Science at St. Mary's University for providing us with the opportunity to undertake this meaningful project. Our sincere thanks go to our advisor Deme H, whose unwavering guidance, patience, and expertise were instrumental in shaping our work from conception to completion. We are also indebted to the faculty members who shared their insights during the review process, helping us refine our approach and overcome technical challenges.

Special appreciation goes to the National Blood Bank Services (NBBS) of Ethiopia, hospital staff, and blood donors who generously participated in interviews and surveys, providing invaluable real-world perspectives that grounded our solution in practical needs. We thank our families and friends for their encouragement and understanding throughout this demanding yet rewarding journey. Finally, we acknowledge the open-source community and the developers behind tools like React, Node.js, whose technologies empowered us to build a system we hope will save lives across Ethiopia.

Chapter 1

1.1 Background of the Project

Blood donation is a vital process in healthcare systems worldwide because it plays a critical role in saving lives. In Ethiopia, however, there are numerous challenges that hinder an efficient donation process. Issues such as limited public awareness, unorganized blood banks, and the absence of modern technological solutions lead to frequent shortages and delays in obtaining compatible blood. Many patients face life-threatening delays—especially in emergency situations—because they cannot quickly find suitable donors.

The traditional system used in many areas is outdated. It lacks the essential capabilities to track donations in real time as well as proper channels for communication and coordination among donors, patients, and hospitals. Recognizing these shortcomings, the project proposes the development of a web-based Blood Donation System. This system will utilize modern technologies to help users locate nearby donors, schedule donations, monitor blood stock levels, and receive timely notifications. In doing so, it aims to transform how blood donations are managed and encourage a more responsive, efficient process throughout Ethiopia.

1.2 Statement of the Problem

Despite the life-saving importance of blood donation, the current system in Ethiopia faces several critical challenges:

Inefficient Donor Database: The absence of a centralized donor database causes delays when matching patients with compatible blood donors.

Lack of Real-Time Tracking: Without a facility to monitor blood types and available stock at donation centers and hospitals in real time, responding effectively to emergencies becomes very difficult.

Low Public Awareness & Motivation: Inadequate public awareness about donation drives and limited incentives for voluntary donations create an imbalance between supply and demand.

Poor Communication: Communication gaps between hospitals and donors render urgent blood requests ineffective, further hampering rapid response in critical times.

The project aims to address these problems by developing a centralized digital system that connects donors, patients, and hospitals. By employing geolocation, instantaneous notifications, and real-time data updates, the system aspires to streamline the entire donation process and ultimately save lives.

1.3 Objective of the Project

1.3.1 General Objective

The primary aim is to develop a responsive and efficient web platform that facilitates blood donation and its management by connecting donors, hospitals, and patients in real time.

1.3.2 Specific Objectives

User Interface Design: Create an intuitive, user-friendly interface tailored for blood donors, hospital staff, and system administrators.

Real-Time Monitoring: Implement real-time blood stock monitors and donation request handling to ensure timely updates.

Geolocation-Based Search: Develop functionality that uses geolocation technology to identify nearby donors.

Hospital Features: Enable hospitals to request specific blood types and update their current blood stock continuously.

Notifications: Integrate Firebase Cloud Messaging to deliver timely donation reminders and urgent alerts.

Educational Outreach: Provide dynamic educational content to raise awareness about blood donation, its benefits, and relevant safety measures.

Accessibility: Ensure the platform is multilingual and accessible to all user demographics.

Security: Implement secure role-based access controls and robust user authentication systems.

Full Deployment Cycle: Complete thorough testing, quality assurance, and eventual deployment of the system.

1.4 Methodology

1.4.1 Data Collection

Data will be gathered using a variety of methods to ensure comprehensive insight into current challenges and user needs:

Interviews: Conduct one-on-one sessions with hospital staff and frequent blood donors.
Surveys: Target potential users, including students and healthcare workers, to assess broader needs.
Observations: Directly observe existing blood donation processes to identify workflow inefficiencies.
Document Analysis: Review existing systems and manual procedures to pinpoint gaps that the new system can address.

1.4.2 System Development Process Model

The project will adopt the Agile development model. Agile enables iterative development with continuous feedback and adaptive planning, which is essential given the evolving requirements and user needs related to healthcare technology.

1.4.3 Design Pattern

The system is built on the Model-View-Controller (MVC) design pattern. Here:

Model: Handles data management.

View: Manages UI/UX components.

Controller: Processes user inputs and orchestrates interactions between the Model and View.

This separation enhances the system's maintainability, scalability, and overall performance.

1.4.4 Programming Languages

Frontend (Web): Developed using JavaScript with React.js to create a dynamic and responsive user interface.

Backend: Developed using JavaScript with Node.js and the Express framework, chosen for its asynchronous capabilities, fast performance, and extensive community support.

1.5 Tools

1.5.1 Hardware Tools

Personal Computers: Used during the development phase.

Internet-Enabled Devices: For accessing and using the web platform across various user devices.

1.5.2 Software Tools

React.js: Provides the basis for the web frontend.

Node.js & Express: Serve as the backbone for the backend API development.

Firebase Auth: Used to handle secure user authentication.

Google Maps API: Powers the geolocation functionality.

Figma / Adobe XD: Tools used for designing UI/UX, ensuring a user-centered design.

Lucidchart: Utilized for drawing ER diagrams and system flowcharts to better plan and visualize the system architecture.

1.6 Scope and Limitations of the Project

1.6.1 Scope of the Project

Nationwide Coverage: The system is designed to work across Ethiopia.

User Types: Serves blood donors, hospital staff, and system administrators.

Accessibility: The web platform is available at all times, supporting both scheduled donations and emergency requests.

Core Features:

User registration and secure login.

Real-time blood stock updates.

Geolocation-based donor search.

Notification system via integrated messaging.

Educational content to inform users about donation procedures and benefits.

1.6.2 Limitations of the Project

Integration: The system does not integrate with national health databases.

Donor Availability: It cannot guarantee that a donor will be available during every emergency.

Internet Access: The solution is not suitable for users without reliable access to the internet.

1.7 Significance of the Project

The proposed Blood Donation System is expected to deliver significant improvements across various dimensions of healthcare and community engagement:

For Hospitals & Blood Banks

Streamlined Management: Facilitates real-time monitoring of blood inventory, helping hospitals plan surgeries and manage emergencies efficiently.

Reduced Wastage: Alerts regarding blood expiry help ensure that stocks are used effectively, thereby reducing wastage.

For Blood Donors

Timely Notifications: Donors receive alerts about donation drives, urgent needs, and nearby blood donation centers.

Health Tracking: The platform maintains a record of donor history and suggests eligibility dates for the next donation.

Recognition and Motivation: Donor contributions may be recognized through badges and public acknowledgment, encouraging regular donations.

For Patients

Rapid Emergency Response: Patients (or their families) can post blood requests and quickly get matched with compatible donors, significantly reducing response times.

Emotional Relief: Minimizes the stress of manually searching for blood, providing transparency on request status.

For the Public Health System

Digital Transformation: Modernizes a critical aspect of emergency healthcare in line with Ethiopia's digital health transformation goals.

Educational Impact: Raises public awareness about blood donation through integrated educational content.

Data-Driven Decisions: Enables health authorities to review and plan based on aggregated data regarding donation patterns, regional needs, and blood demand trends.

For NGOs, Donor Organizations, and the Government

Campaign Support: Acts as a backbone for nationwide blood donation initiatives, streamlining event coordination and impact tracking.

Resource Allocation: Real-time and historical data assist in directing mobile blood banks, staffing, and awareness campaigns more effectively.

Social Impact

Cultural Integration: Fosters a culture of solidarity and altruism, promoting selfless community service.

Empowerment Through Technology: Empowers citizens by providing tools for rapid response in emergencies and democratizing access to life-saving services.

1.8 Feasibility Study

The feasibility study ensures that the project can be successfully completed within current constraints by evaluating technical, economic, and operational factors.

1.8.1 Technical Feasibility

Tool Availability: The chosen platforms (React.js, Node.js, Firebase, Google Maps API) are widely available, robust, and supported by strong communities.

Team Expertise: The development team has experience in Java, JavaScript, and Firebase, which enhances the likelihood of successful implementation.

1.8.2 Economic Feasibility

Cost Efficiency: Low development expenses due to reliance on open-source tools.

Affordable Hosting: Services like Firebase and GCP offer cost-effective hosting solutions, with only minimal recurring costs for internet and server fees.

1.8.3 Operational Feasibility

User Interface: The system is built with a user-friendly interface that caters to all levels of digital literacy.

Real-World Integration: Designed for smooth integration into hospital workflows, with minimal disruption.

Reliability and Scalability: Hosted on reliable cloud platforms, the system is scalable and offers high uptime, automatic backups, and built-in security.

Low Maintenance: Routine maintenance, updates, and bug fixes can be managed by a small team.

Stakeholder Acceptance: By addressing the critical needs of hospitals, donors, and patients, the system is highly likely to be adopted widely.

Mitigation Strategies: Plans include visual and multilingual training materials and a lightweight design for low-bandwidth areas.

1.9 Risk Assessment

Risk assessment identifies potential challenges and outlines strategies to mitigate them.

1.9.1 Risks and Mitigation

Internet Availability Issues: Particularly in rural areas, limited connectivity may hinder system access. *Mitigation:* Optimize for low-bandwidth situations and include caching where possible.

User Resistance: Some users may be reluctant to adopt new digital tools due to limited familiarity with technology. *Mitigation:* Provide comprehensive training, demo videos, and community outreach.

System Bugs or Deployment Delays: Development setbacks may occur despite rigorous testing. *Mitigation:* Employ continuous testing and update cycles.

1.9.2 Assumptions

Users are assumed to have access to web browsers and basic digital literacy.

Hospitals and other stakeholders are expected to cooperate in implementing and using the system.

External services (Firebase, Google Maps API) will remain free or affordable throughout the development period.

1.9.3 Constraints

- **Time:** There is a limited window for development and testing.
- **External Dependencies:** The system relies on third-party APIs (e.g., Maps, FCM) which could introduce potential issues.

- **Budget:** Financial constraints may restrict extensive deployment or additional features.

1.10 Work Breakdown Structure (WBS)

The project is divided into eight distinct phases to ensure a systematic approach:

Phase 1: Requirement Gathering and Analysis

Identify and engage key stakeholders (hospitals, donors, NGOs).

Conduct comprehensive interviews, surveys, and observational studies.

Define both functional and non-functional system requirements.

Document and validate requirements with stakeholder feedback.

Phase 2: System Design

Design the overall system architecture based on the MVC pattern.

Develop a coherent database schema using Firebase and PostgreSQL.

Create detailed wireframes and mockups for the web interface.

Finalize UI/UX design and establish security models and access control mechanisms.

Phase 3: Backend Development

Set up the server environment using Node.js and Express.

Develop secure user authentication with Firebase Auth.

Implement RESTful APIs for both user and hospital modules.

Integrate blood stock tracking and donation management functionalities.

Connect backend services with databases (Firestore, PostgreSQL).

Phase 4: Frontend Web Development

Set up the React.js framework for a responsive web application.

Develop separate dashboards for donors, hospital staff, and administrators.

Incorporate geolocation features using Google Maps API.

Integrate Firebase Cloud Messaging (FCM) for real-time alerts.

Ensure full mobile responsiveness across various devices.

Phase 5: Testing and Quality Assurance

Perform unit tests on individual components and modules.

Conduct integration and system-wide testing to ensure all parts work together.

Execute user acceptance testing (UAT) with real stakeholders.

Identify, document, and resolve bugs, refining system functionality.

Phase 6: Deployment and Hosting

Deploy backend services to cloud platforms (e.g., Firebase Functions, GCP).

Host the frontend on suitable platforms (e.g., Firebase Hosting, Vercel).

Configure database security rules and optimize performance settings.

Conduct final system validations and performance optimization.

Phase 7: Documentation and Reporting

Prepare thorough technical documentation covering APIs, database schemas, and UI designs.

Develop comprehensive user manuals and training content.

Compile the final project report and create presentation slides.

Submit the project deliverables for evaluation.

Phase 8: Final Presentation and Handover

Present the completed project to advisors and evaluators.

Officially deliver the final product to all stakeholders.

Provide support and maintenance guidelines for future operations.

CHAPTER TWO

Business Area Analysis and Requirement Definition

2.1 Introduction

The National Blood Bank Services (NBBS) in Ethiopia is responsible for ensuring a safe and adequate supply of blood and blood products across the country. Currently, its operations rely on manual processes and paper-based systems for donor registration, blood collection, screening, distribution, and quality control. The introduction of a **digitalized system through a web-based platform** will significantly improve efficiency, data accuracy, accessibility, and transparency. This chapter outlines the business area analysis, current system evaluation, proposed system, and requirement gathering to facilitate the development of an improved, digital solution.

2.2 Business Area Analysis

2.2.1 Detailed Analysis

NBBS plays a critical role in health service delivery by managing the collection, testing, and distribution of blood supplies. The current system is **fragmented and lacks automation**, leading to inefficiencies in tracking donor data, inventory levels, and distribution logistics. A web-based platform will centralize operations, automate workflows, and provide real-time monitoring capabilities.

2.2.2 Current System

The existing system is **primarily paper-based**, with limited digital tools for donor management, inventory tracking, and communication among blood banks and health facilities. This results in **data inconsistencies, slow processing, and difficulty in accessing reports** on blood availability across different locations.

2.2.3 Players of the Existing System

Key stakeholders involved in the NBBS include:

- **Blood donors**: Individuals who voluntarily donate blood.

- **Blood collection teams**: Mobile and central clinic teams responsible for organizing blood collection campaigns.
- **Laboratory personnel**: Conduct screening for infectious diseases and prepare blood components.
- **Health facilities**: Hospitals and clinics that require blood supplies for patient transfusion.
- **Quality control teams**: Ensure compliance with safety standards in testing and distribution.
- **Marketing and communication personnel**: Promote voluntary blood donation and awareness campaigns.
- **Data management team**: Record donor information, track inventory, and generate reports.
- **Government and regulatory bodies**: Oversee NBBS operations and set policies for blood transfusion services.

2.2.4 Proposed System

The new digital system will introduce a **web-based platform** that integrates:

- **Online donor registration** with appointment scheduling.
- **Automated inventory tracking** for blood availability.
- **Digital communication system** to connect blood banks and health facilities.
- **Centralized data management** for performance reports and compliance tracking.
- **Secure access and authentication** for personnel involved in blood bank operations.

2.2.5 Forms and Reports Used

The digital system will replace existing **manual forms** with **dynamic digital reports**, such as:

- **Donor registration forms** with health history.
- **Donor Ranking Badge** with Donation history.
- **Blood test result reports** from the laboratory.
- **Blood inventory tracking** across regional blood banks.
- **Notify Donor using Notification Methods** (SMS or Notification Bar)
- Digitation Donte Form user for uploading Donator Paper fill in to the system

- **Distribution reports** for hospitals and clinics.
- **Quality control audit logs** ensuring compliance with safety protocols.

2.3 Requirement Gathering

2.3.1 Requirement Gathering Techniques

To collect accurate requirements for the system, the following techniques will be used:

- **Interviews** with NBBS personnel to identify operational needs.
- **Survey/questionnaires** for blood donors to understand their expectations.
- **Document analysis** of existing manual records and workflows.
- **Observation** of current processes to identify pain points and inefficiencies.

2.4 Method of Communication

2.4.1 Communication Techniques

To ensure the smooth implementation and transition to the new system, communication strategies include:

- **Stakeholder meetings** to discuss system requirements and expectations.
- **Digital dashboards** for real-time tracking of blood availability.
- **Automated SMS/email alerts** for donor reminders and inventory updates.
- **Training sessions** to educate staff on system usage.

2.5 Requirement Definition

2.5.1 Functional Requirement

Functional requirement focus on the main function ,which the new application system will provide. The system to be develop has different functions . The functional requirement that will provided by the new system include the following activities.

Functional Requirement 1(FR1): User Registration and profile management

The system shall able to allows donors to create personal accounts by submitting essential information such as name, contact details, age, blood type, and medical history (e.g., recent illnesses, medications, or travel to high-risk regions). The system validates this data to ensure accuracy and compliance with eligibility criteria (e.g., minimum age or weight requirements). Once registered, donors can update their profiles, view donation history, and track eligibility for future donations. Medical staff and blood ban also have role-based profile while admin accounts provide privileges to manage user roles, generate reports, and monitor system-wide compliance. Robust security measures, such as encrypted passwords and role-based access control, protect sensitive data.

Functional Requirement 2(FR2): Donor appointment scheduling

Donor Appointment Scheduling lets donors easily schedule, change, or cancel their donation appointments online. The system shows available time slots at nearby centers and reminds donors of basic rules (like age or health requirements) before they confirm. Once scheduled, donors get automatic text or email reminders so they don't forget. They can also add appointments to their personal calendar. Blood bank can view all appointments in one place, adjust schedules during emergencies, or focus on urgent blood needs. This makes scheduling fast, reduces waiting times, and helps donors donate smoothly.

Functional Requirement 3(FR3): Blood Donation & Inventory Management

The system shall store in an organized database the following data:

1. Records donation of details like blood type,quantity and collection date.
2. Profile and history of the donation
3. System tracks inventory (e.g., blood type, expiration dates) and alerts for low stock.

Functional Requirement 5(FR5):Reporting

The system helps admins create easy reports, like tracking how many donations happen each month or showing details about donors (e.g., age, location, or blood type). It also monitors how blood is used for example, how much is given to patients in hospitals or if it expires or isn't safe.

The reports show admins regular trends, like which months have the most donations or which locations need more donors. This helps them make smarter decisions to reduce waste.

Functional Requirement 6(FR6):Donation history

The system keeps track of all donations safely. Donors can log in to see their past donations like when and where they donated, and their blood type. Blood banks can look up donor details to check if someone can donate again. Admins check all records to make sure everything is correct and fix any mistakes.

2.5.2 Non-Functional Requirement and Quality Requirements

These requirement are focused with reliability,maintainability,availabilty,efficiency ,usability and so on.

Non-Functional Requirement 1(NFR1): Performance

The system must work quickly, even when lots of people use it at the same time. For example, pages should load within 2 seconds normally and not take longer than 5 seconds during busy times (like emergencies). If many people join a blood donation event, the system should handle up to 10,000 users without slowing down. Important actions, like confirming an appointment or updating blood stock, should happen instantly.

Non-Functional Requirement 2(NFR2): Availability & Reliability

The system should almost never crash. It must stay online 99.9% of the year, except for planned maintenance (which users will be warned about a day before). Even if part of the system breaks (like the database), the rest should keep working. All blood type records, expiration dates, and donor details must stay accurate to avoid mistakes.

Non-Functional Requirement 3(NFR3): Security

Donor data (like medical history) must be protected. Admins must use two-step login (like a password + SMS code) for extra safety. Sensitive information is locked with strong encryption, both when stored and sent over the internet. The system must follow privacy laws (like HIPAA or GDPR), and only let users see what they need—for example, donors can’t see hospital inventory.

Non-Functional Requirement 4(NFR4): Usability

The system should be easy for everyone to use. It must work well on phones and tablets, and support features like screen readers for people with disabilities.

Non-Functional Requirement 5(NFR5): Compatibility

The system must work smoothly on all major browsers (Chrome, Firefox, Safari, Edge). It should also connect with other tools hospitals use (like patient record systems) and let donors add appointments to their Google Calendar.

Non-Functional Requirement 6(NFR6): Maintainability

The system should be easy to update. For example, fixing the appointment feature shouldn't break the inventory section. Updates should take less than 15 minutes and happen at times when fewer people are using the system (like late at night).

Non-Functional Requirement 7(NFR7): Notification Reliability

Alerts (like appointment reminders or low stock warnings) must reach users 95% of the time within 5 minutes. If an SMS fails, the system will try sending an email or an in-app message instead.

Non-Functional Requirement 8(NFR8): Notification & Alerts

The system shall automatically sends reminders to donors about their upcoming appointments and when they're eligible to donate again. It also alerts blood bank if blood units are about to expire or if stock levels are critically low. This helps prevent waste and ensures hospitals never run out of blood when needed most. Simple messages via email or SMS keep everyone informed and ready to act.

2.6 System Modeling: Blood Donation System

2.6.1 Introduction

System modeling is a method used to visualize, specify, construct, and document the components and behavior of a system. For the Blood Donation System, system modeling provides a clear understanding of how the system will interact with users such as donors, hospitals, and

administrators. It defines system functionality through diagrams and textual descriptions, ensuring that requirements are met and helping developers and stakeholders align on the system's scope.

2.6.2 System Use Case

Key Use Cases:

- User Registration
- User Login
- Search for Donors
- Request Blood
- Donate Blood
- Update Blood Stock
- Send/Receive Notifications
- Enter Information to the system

2.6.3 UI Identification

Donor UI:

- Registration screen
- Login screen
- Blood donation eligibility checker
- Appointment scheduler
- Notification center
- Reward Center

Hospital Staff UI:

- Dashboard to manage blood stock
- Blood request form
- Approved Blood request form
- Donor search and communication tools
- Hospital Staff Admin

Blood Bank UI:

- User management panel
- Content management system
- System logs and reports
- Enter Information to the system

2.6.4 Business Rules Identification

Business Rules:

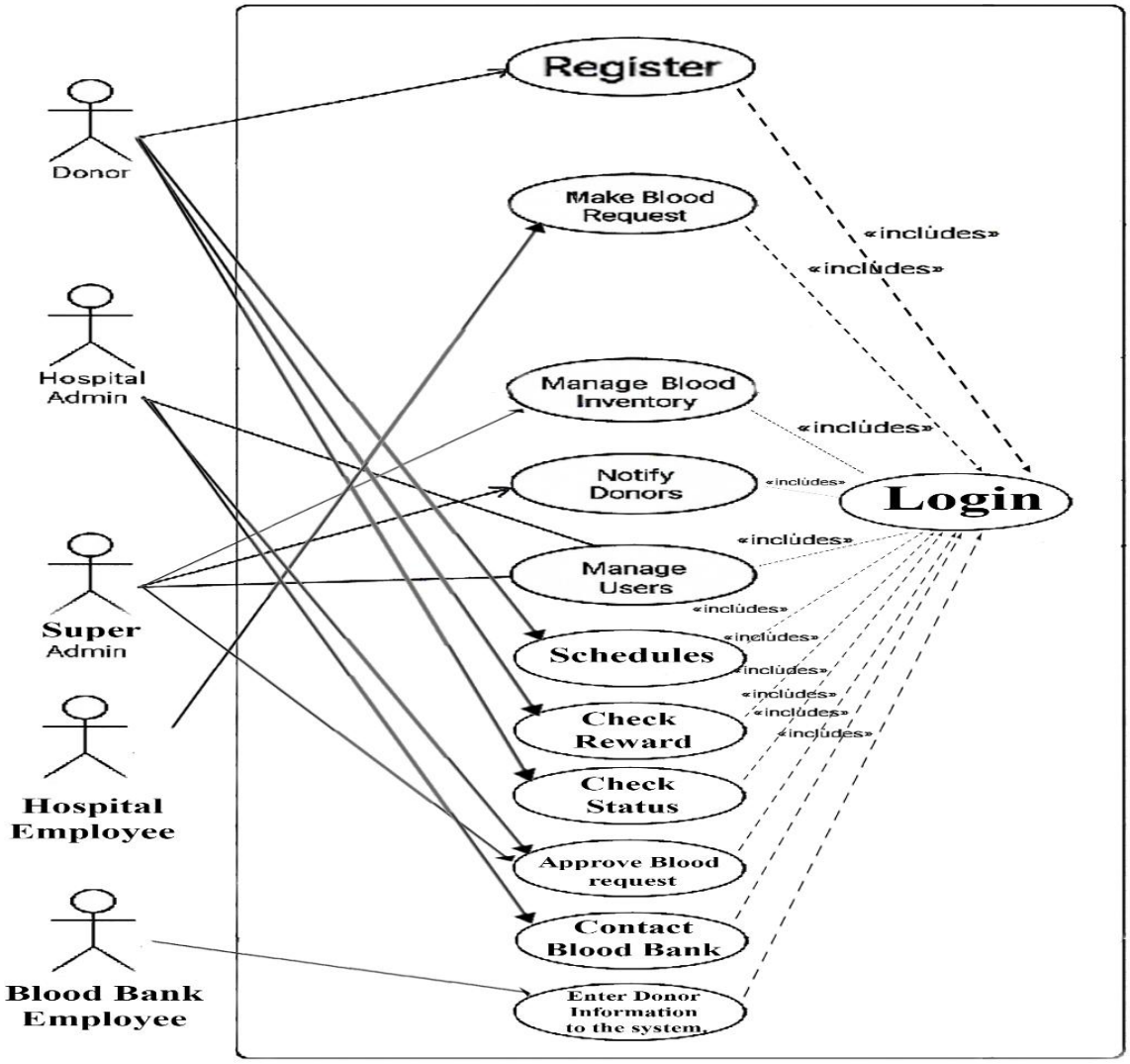
• Eligibility Rule: Only users aged 18–60, in good health, and who haven't donated in the last 3 months can donate.
• Verification Rule: Hospital accounts must be verified by an admin before submitting blood requests.
• Donation Interval Rule: A donor must wait at least 90 days between donations.
• Matching Rule: Blood request matches are prioritized by location proximity and blood type compatibility.
• Notification Rule: Donors receive automatic alerts for appointments, matching requests, and reminders.
• Stock Update Rule: Hospitals must update blood stock immediately after a donation is received.

2.6.5 Actor Identification

Actors:

- **Donor:** Registers, Login, schedules, Check Reward and Check Status.
- **Hospital_Admin:** Login, Approve Blood requests, and Contacts Blood Bank.
- **Hospital_Employee:** Login, Make Blood requests
- **Super_Admin:** Login, Mange the Blood Stock, Manages User, Approve Blood requests, Notify Donor for Emergency Case.
- **Blood_Bank_Employee:** Login, Enter Donor Information to the system.

2.6.6 Use Case Diagram



2.6.7 Use Case Description

2.6.7.1 Use Case 1: Register

Identification	UC-01
Name	Register
Actor(s)	Donor
Description	Allows users to register a new account.
Preconditions	Donor must be register in.
Post conditions	User gains access to system functionalities
Main Flow	System grants access and redirects to dashboard.
Alternative Flows	Invalid credentials → Show error message.
Include	Login
Extend	Logout

2.6.7.1 Use Case 1: Login

Identification	UC-01
Name	Login
Actor(s)	All
Description	Allows users to Login in Existing Account.
Preconditions	They must be have a Account.
Post conditions	User gains access to system functionalities
Main Flow	User enters credentials. System validates credentials. System grants access and redirects to dashboard.
Alternative Flows	Invalid credentials → Show error message.
Include	Login
Extend	Logout

2.6.7.2 Use Case 2: Make Blood Request

Identification	UC-01
Name	Make Blood Request
Actor(s)	Donor
Description	A donor or recipient submits a request for a specific blood type.
Preconditions	User must be logged in.
Post conditions	Request is stored and forwarded to relevant hospitals.
Main Flow	User selects blood type and location. Submits request with details. System stores and forwards to matching hospitals.
Alternative Flows	No hospitals available → System notifies user.
Include	Login
Extend	Logout

2.6.7.4 Use Case 4: Manage Blood Inventory

Identification	UC-01
Name	Manage Blood Inventory
Actor(s)	Super_Admin
Description	Blood Bank track and update blood stock levels.
Preconditions	Blood Bank is authenticated.
Post conditions	Inventory is updated.
Main Flow	Hospital logs in. Views current stock. Adds/updates units.
Alternative Flows	Invalid blood type entry → Show error.
Include	Login
Extend	Logout

2.6.7.5 Use Case 5: Notify Donors

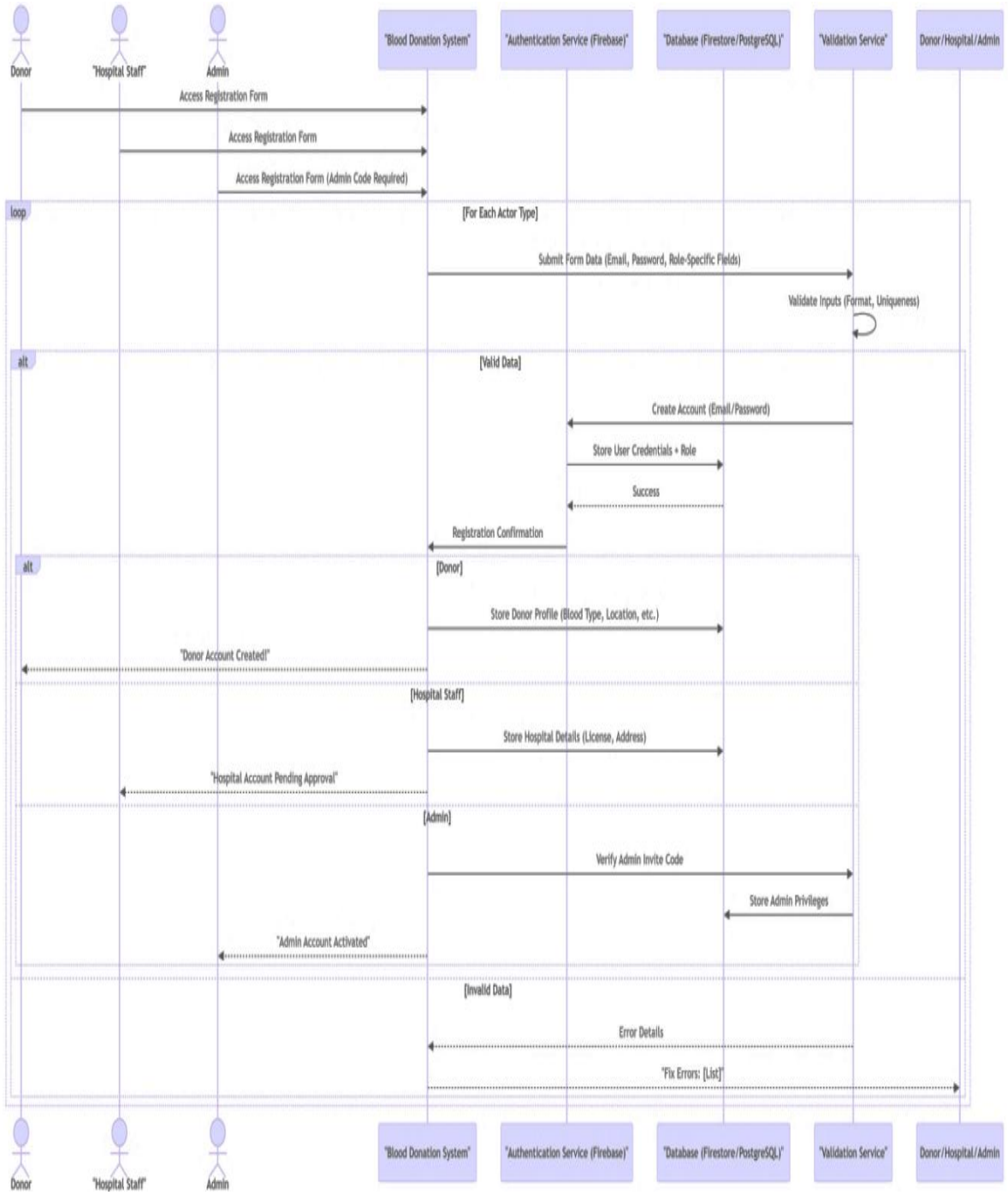
Identification	UC-01
Name	Notify Donors
Actor(s)	Super_Admin
Description	Sends notifications to eligible or nearby donors.
Preconditions	Blood stock is low or demand is urgent.
Post conditions	Notifications are sent via SMS/FCM.
Main Flow	Hospital identifies blood need. Selects donor list. System sends message.
Include	Login
Extend	Logout

2.6.7.6 Use Case 6: Manage Users

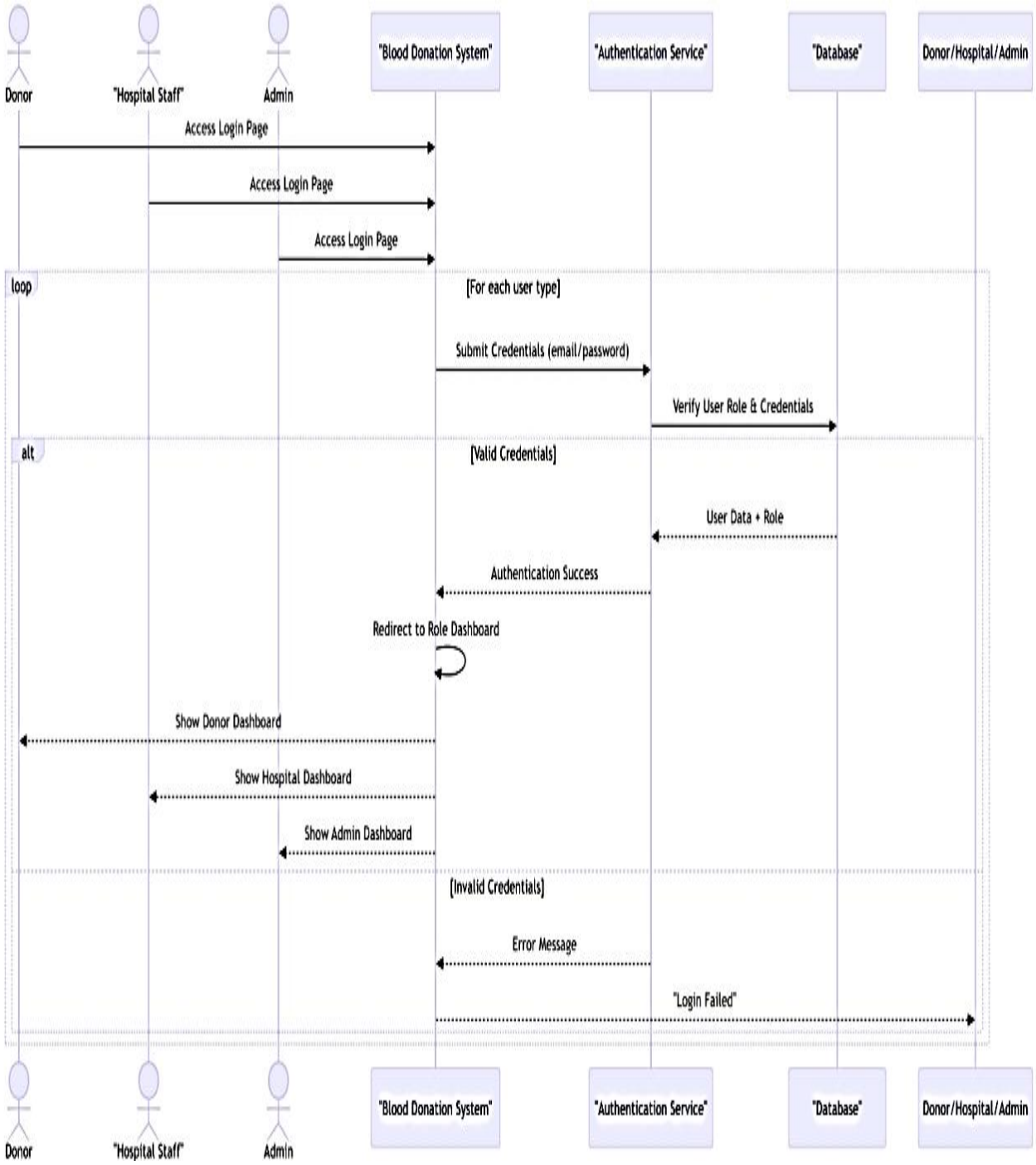
Identification	UC-01
Name	Manage Users
Actor(s)	Super_Admin
Description	Admin can view, update, or delete user accounts.
Preconditions	Admin login is successful.
Post conditions	User records are modified.
Main Flow	Admin logs in. Views list of users. Edits, deactivates, or deletes accounts.
Include	Login
Extend	Logout

2.6.8 Sequence Diagram

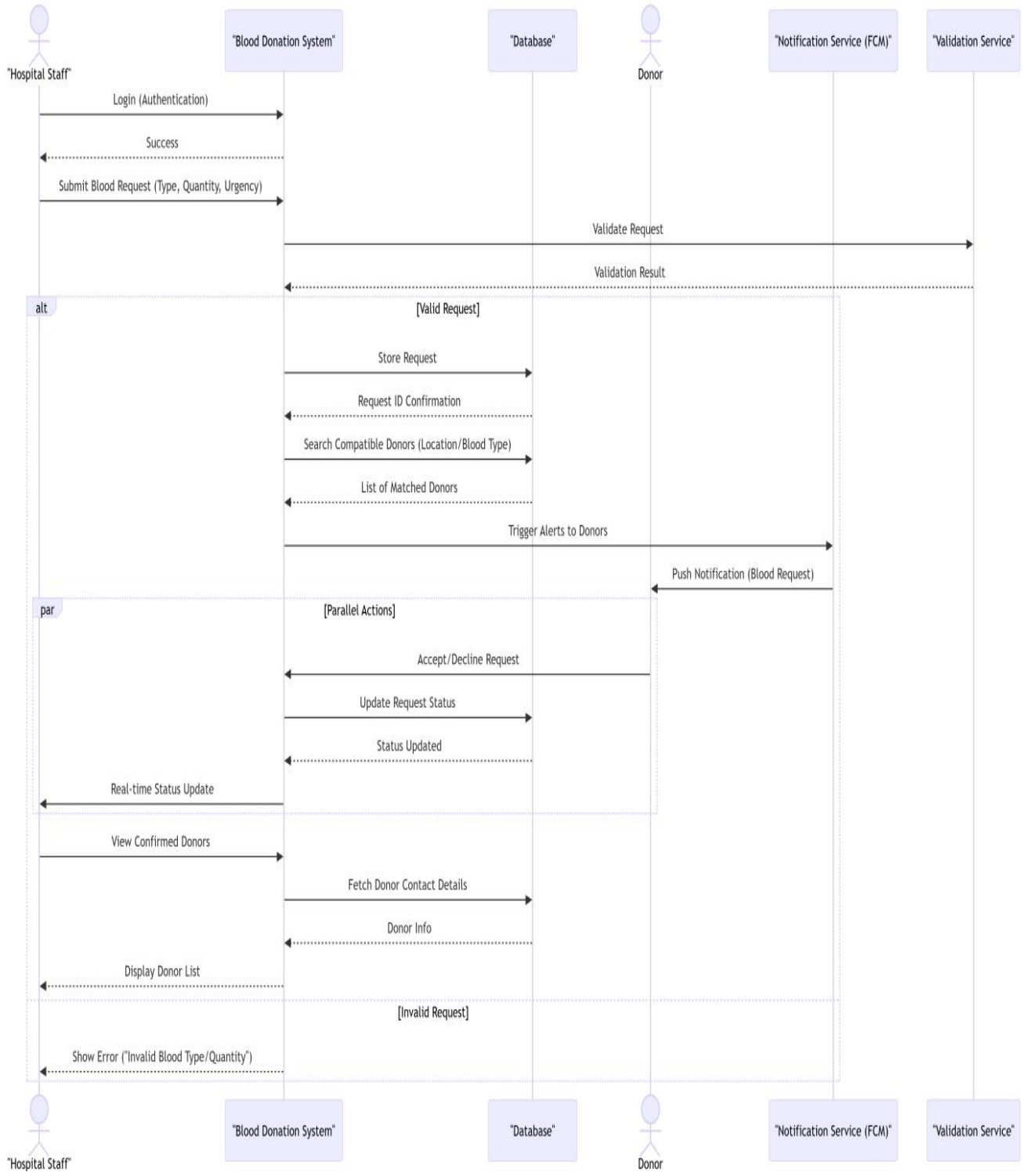
2.6.8.1 **Sequence diagram for Register** Process



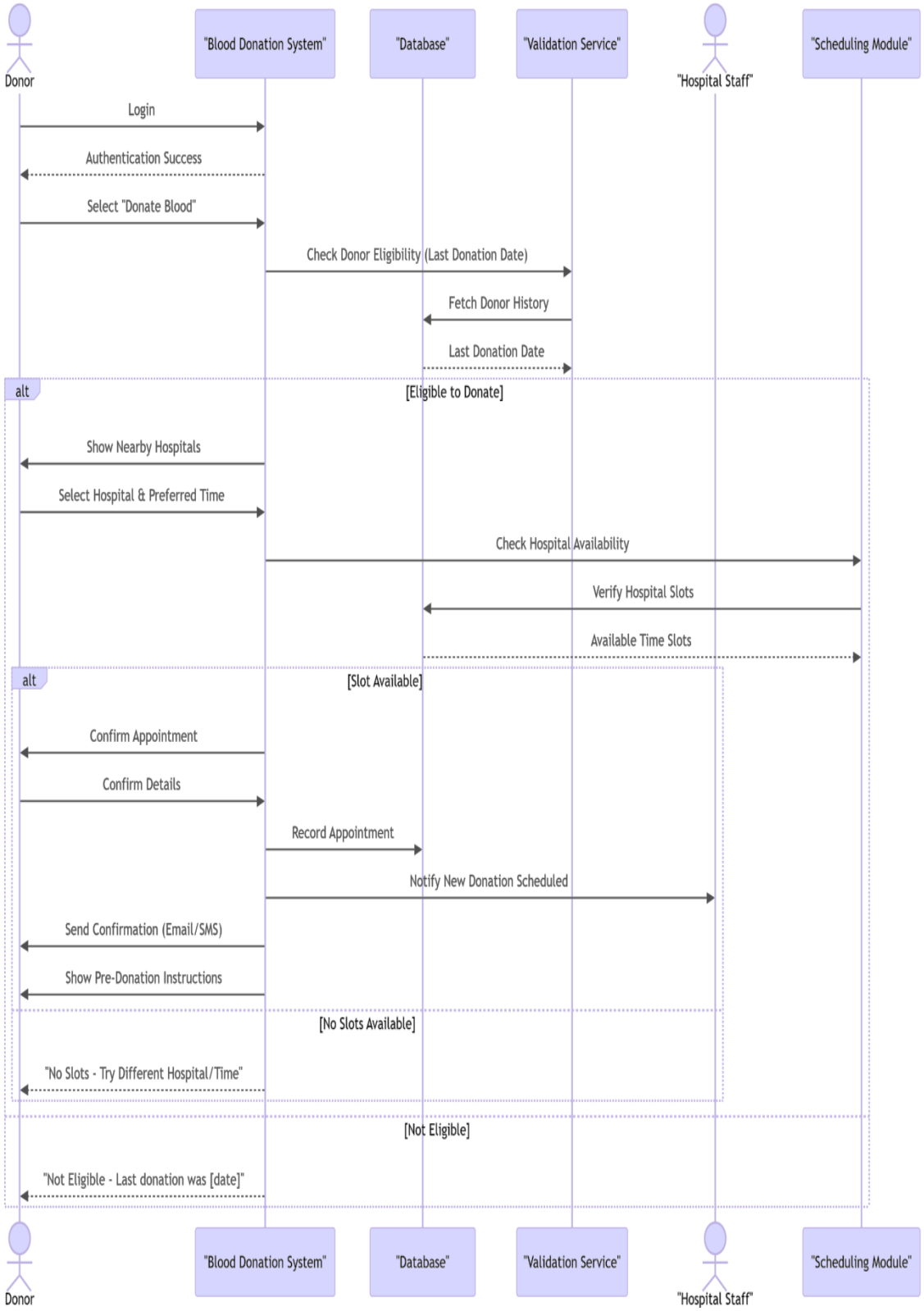
2.6.8.2 **Sequence diagram for Login** Process



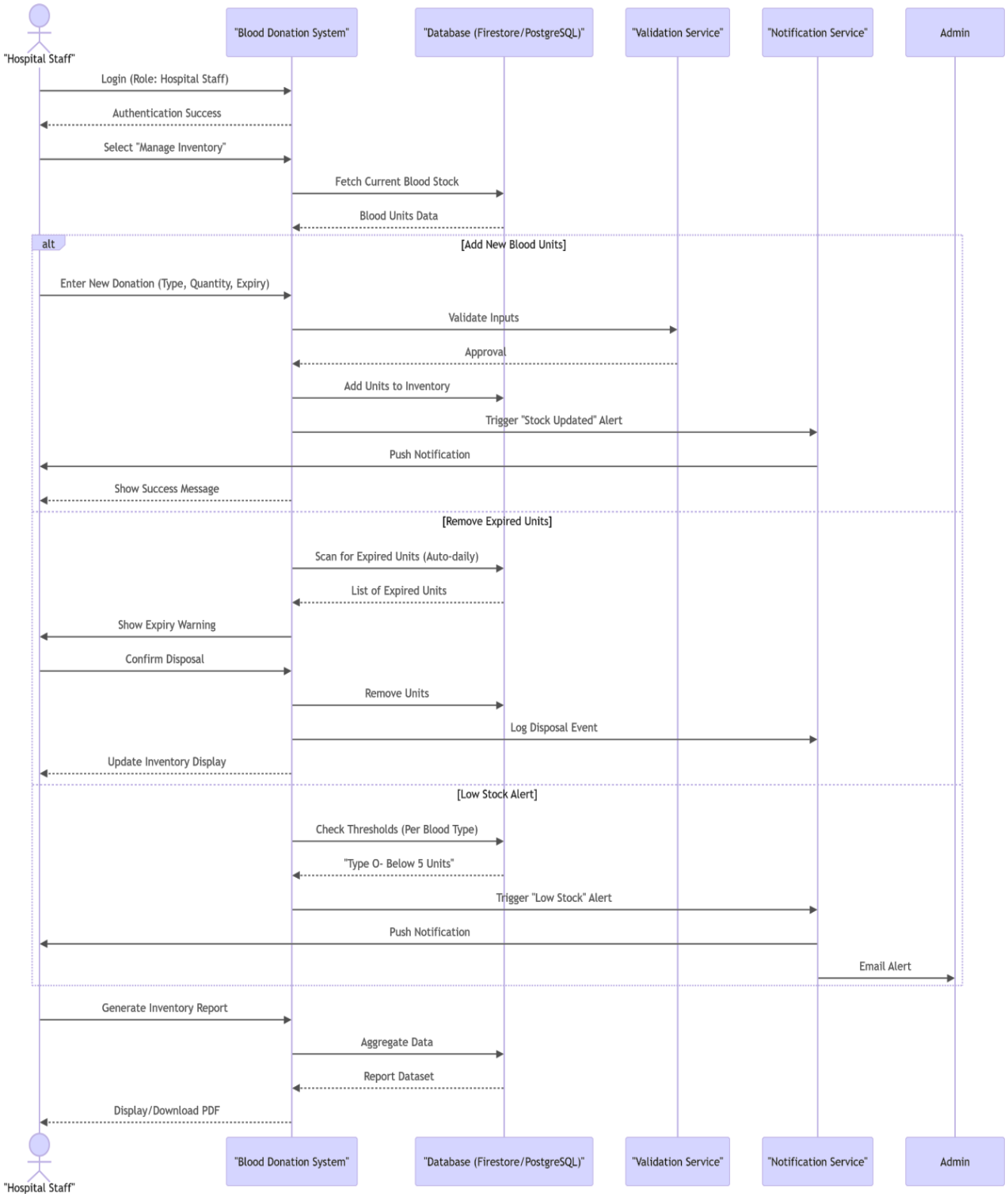
2.6.8.3 Sequence diagram for Make Blood Request



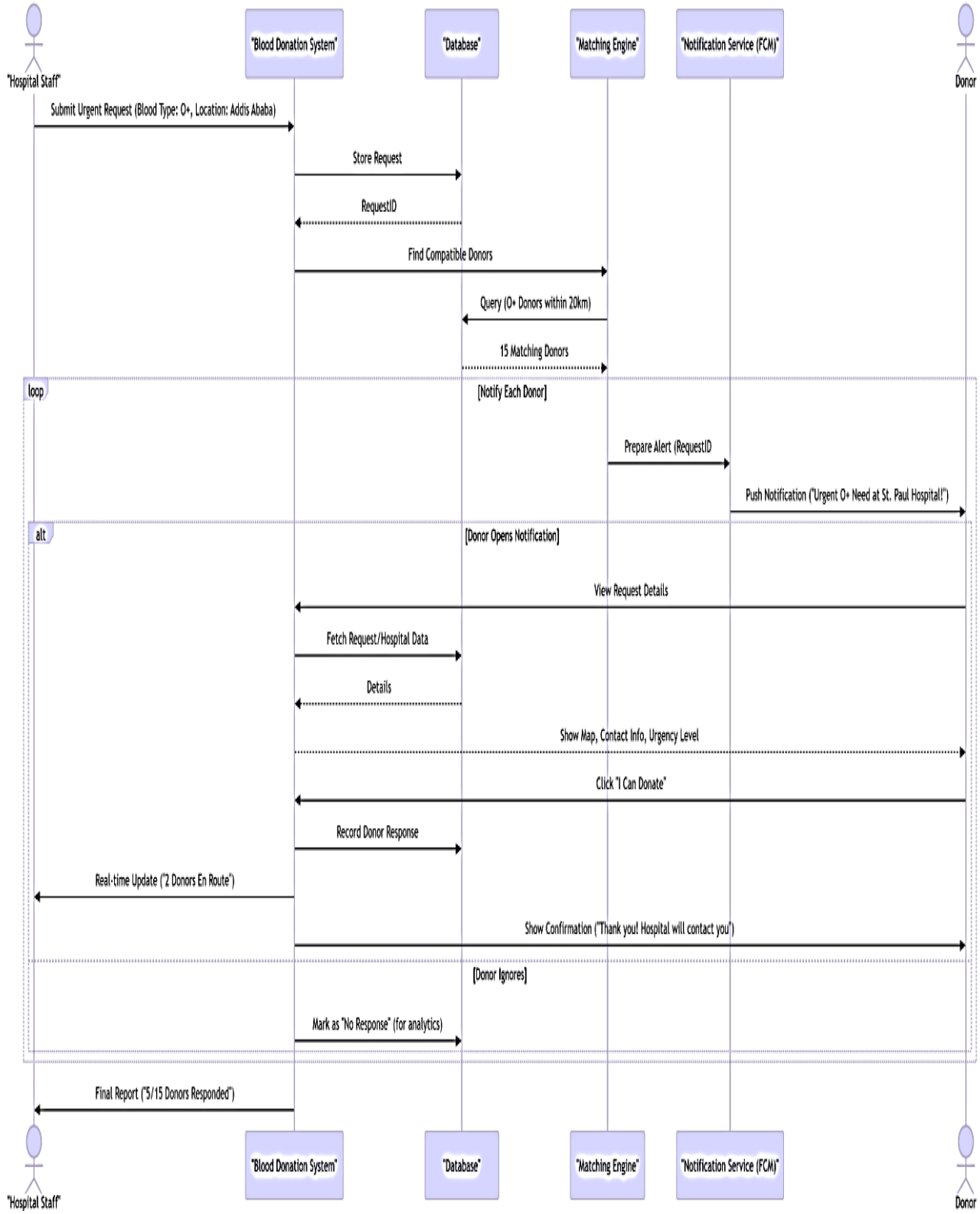
2.6.8.4 Sequence diagram for Donate Blood



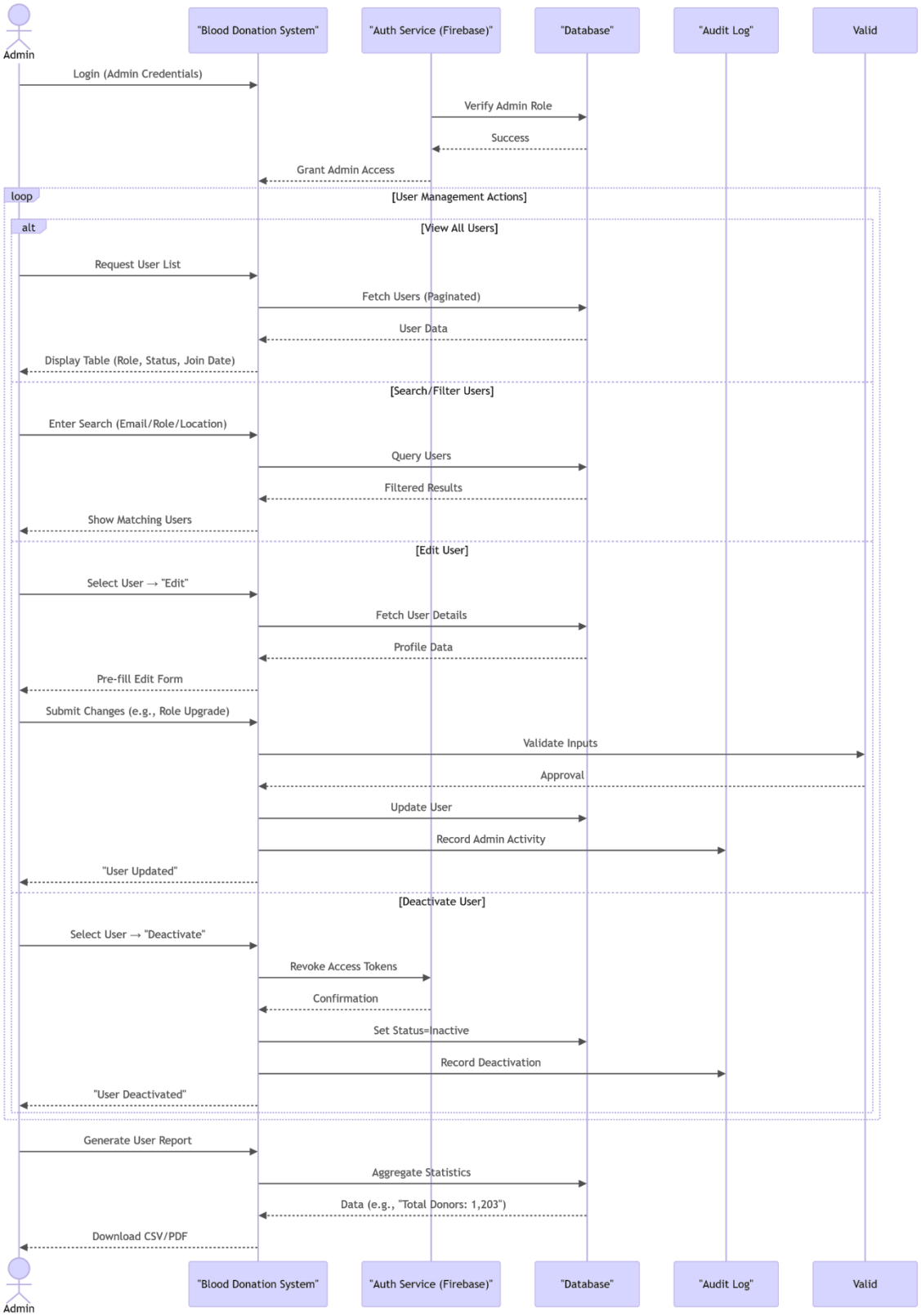
2.6.8.5 Sequence diagram for Manage Blood Inventory



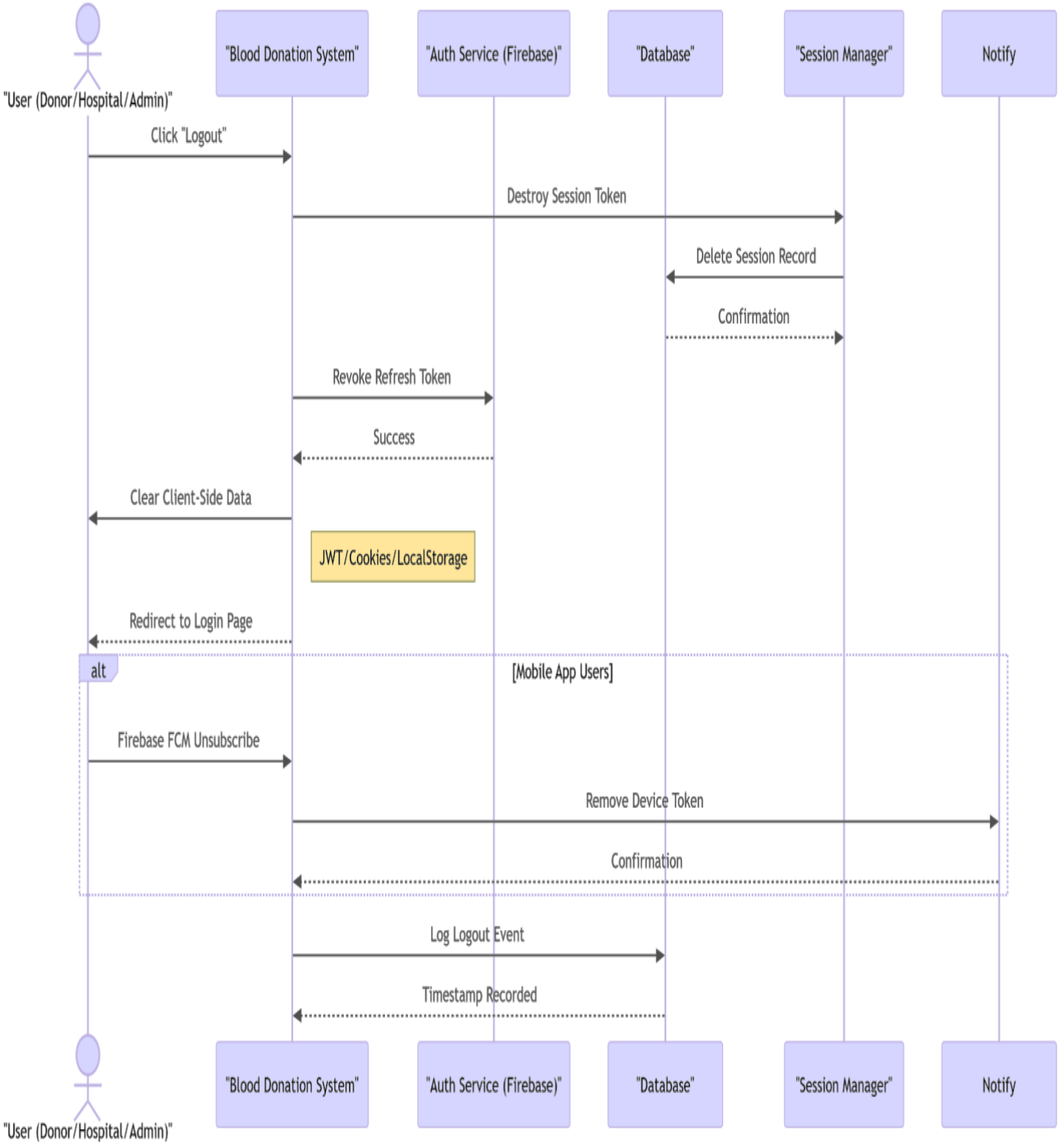
2.6.8.6 Sequence diagram for Notify Donors



2.6.8.7 Sequence diagram for Manage Users

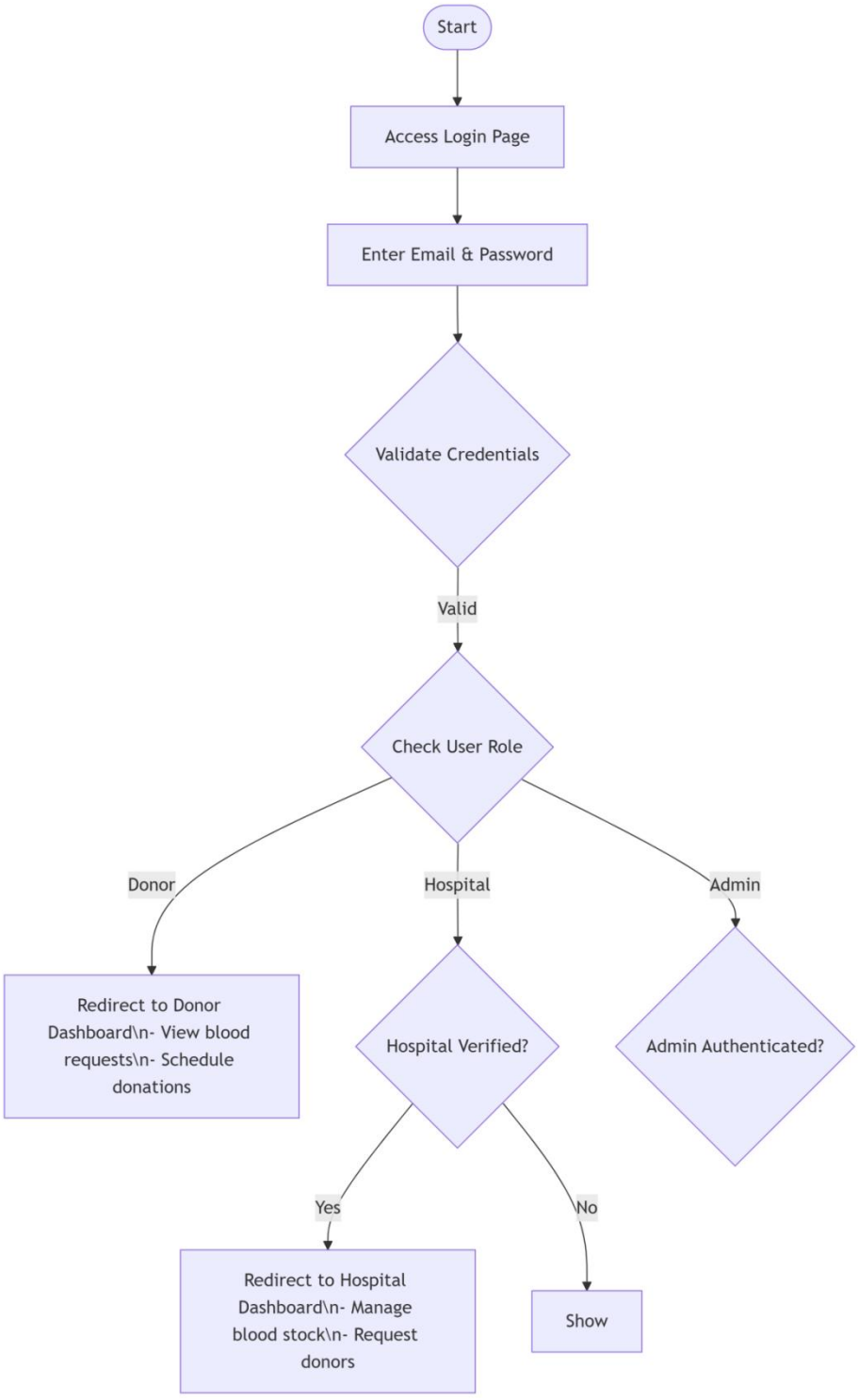


2.6.8.8 Sequence diagram for Logout

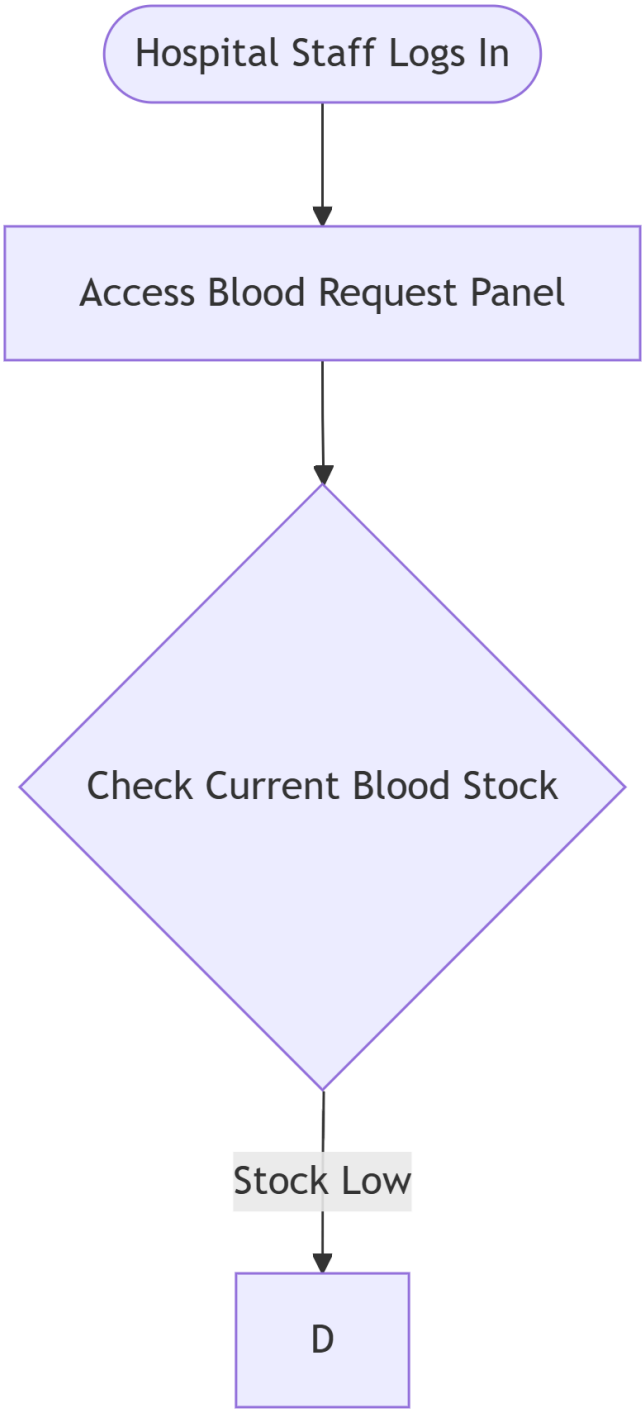


2.6.9 Activity Diagramming

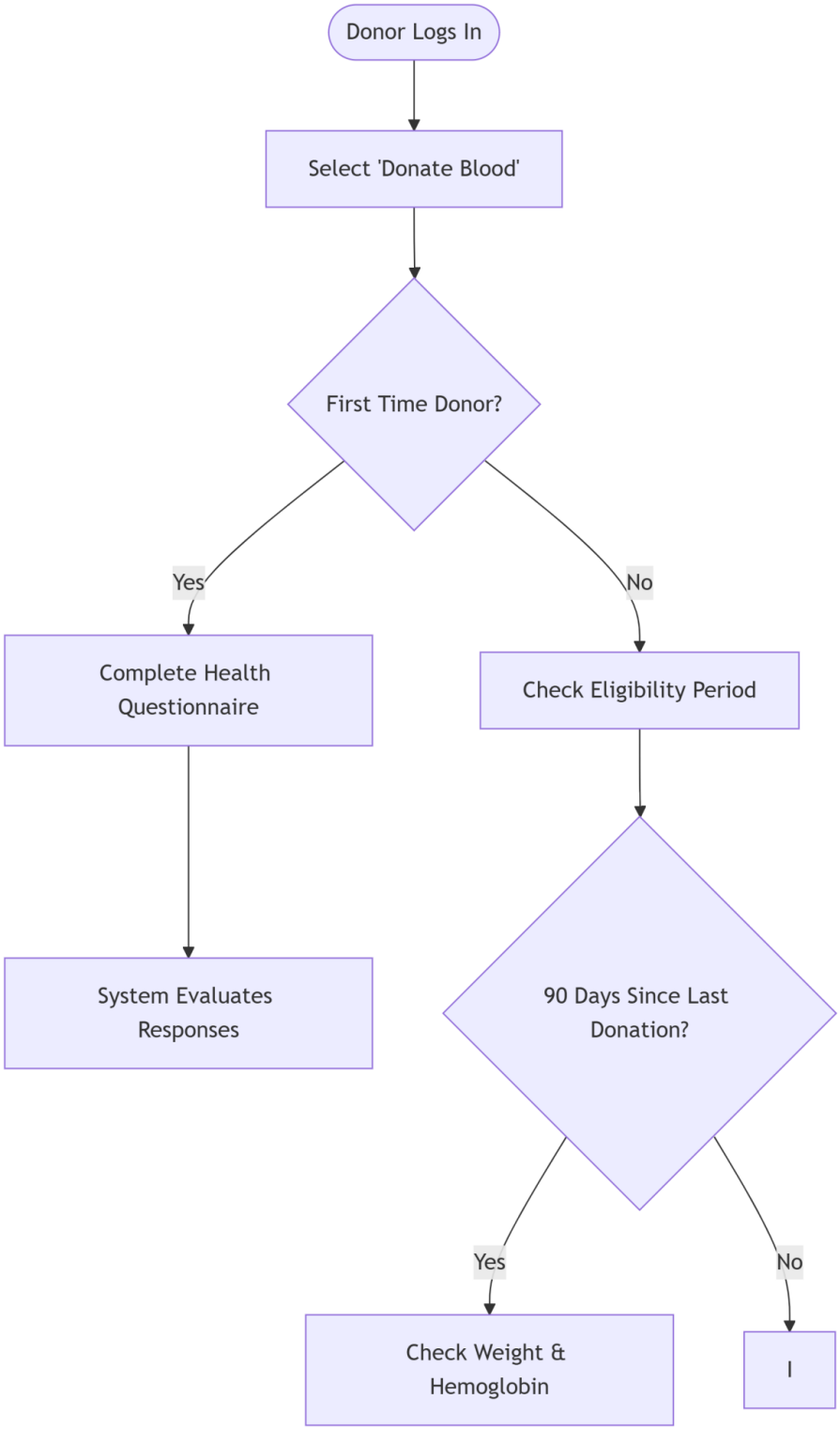
2.6.9.1 Activity Diagramming for Login



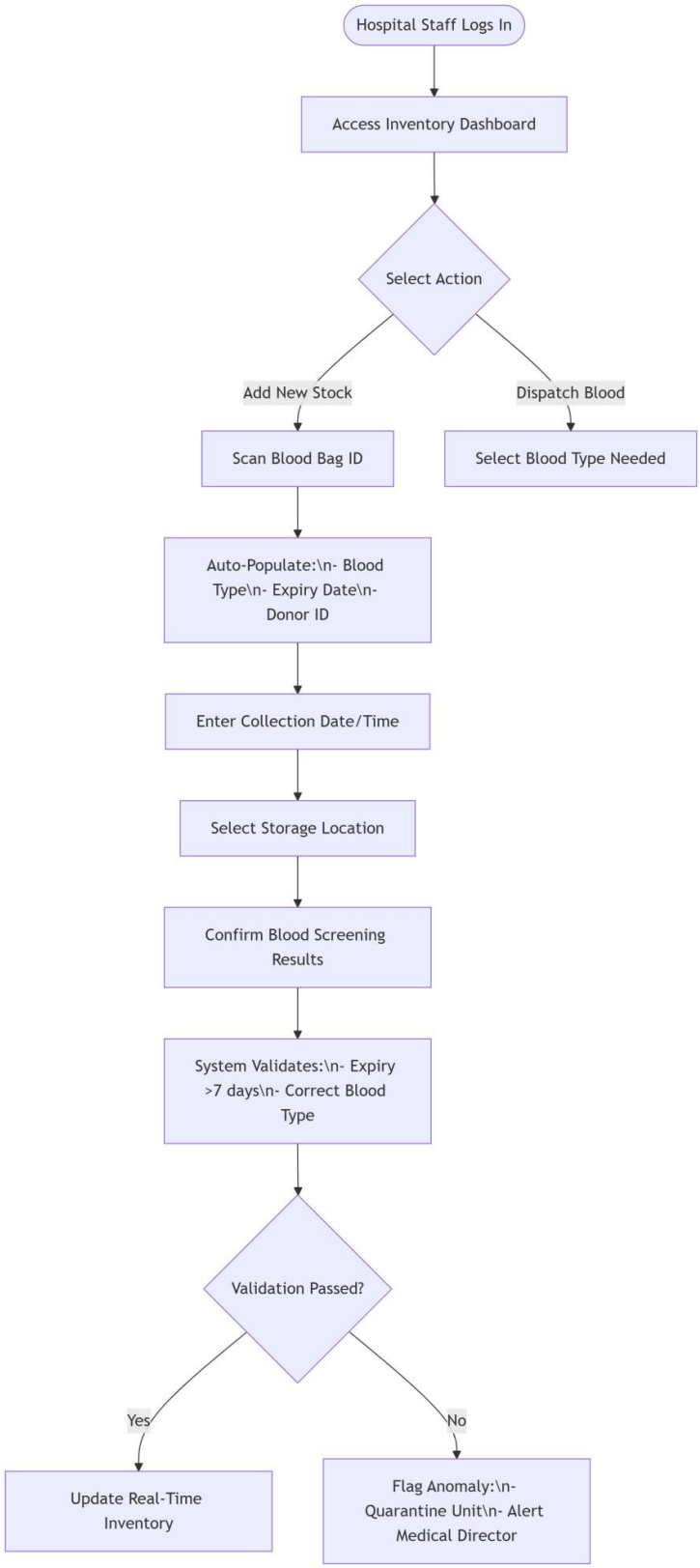
2.6.9.2 Activity Diagramming for Make Blood Request



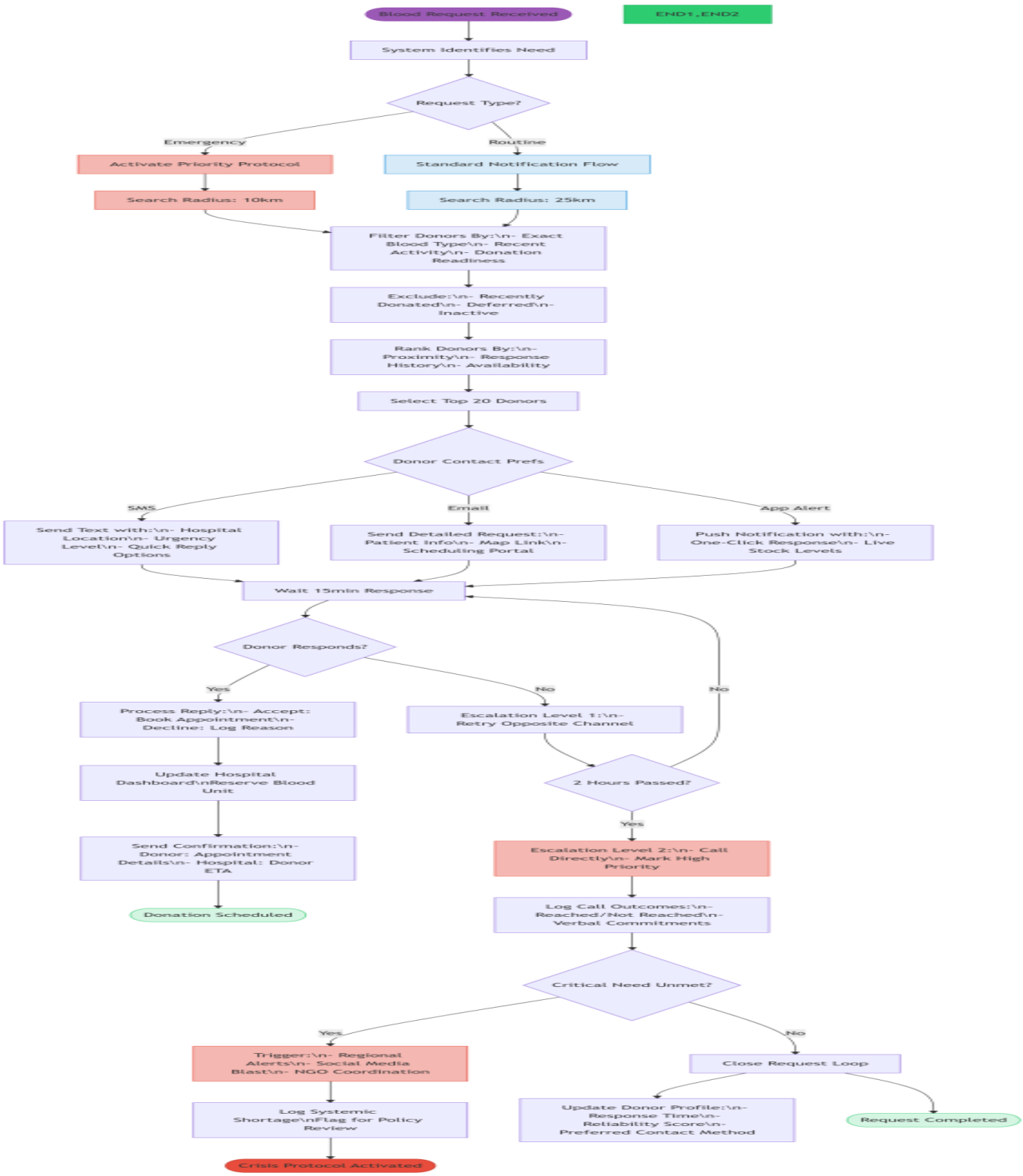
2.6.9.3 Activity Diagramming for Donate Blood



2.6.9.4 **Activity Diagramming for** Manage Blood Inventory

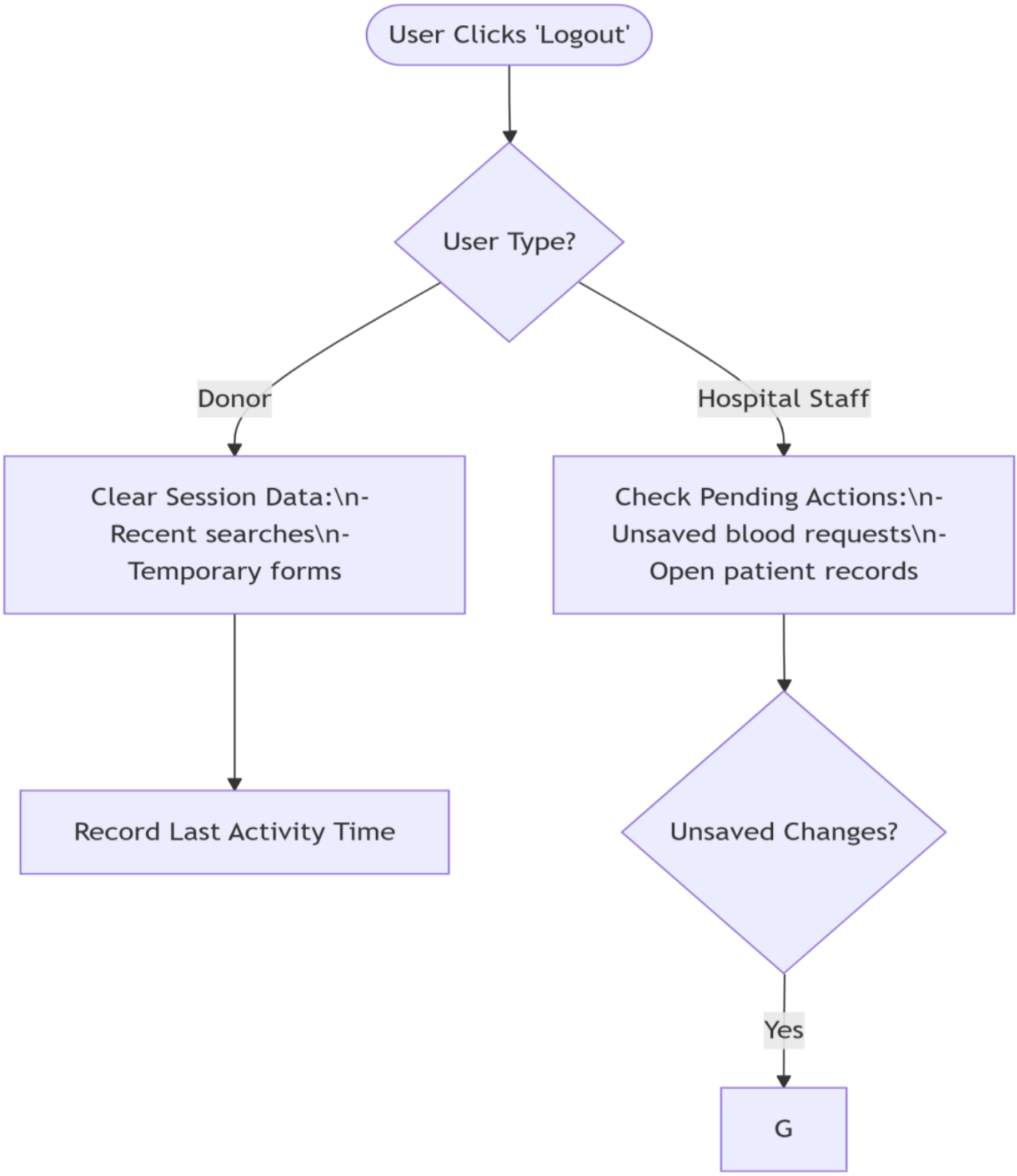


2.6.9.5 Activity Diagramming for Notify Donors

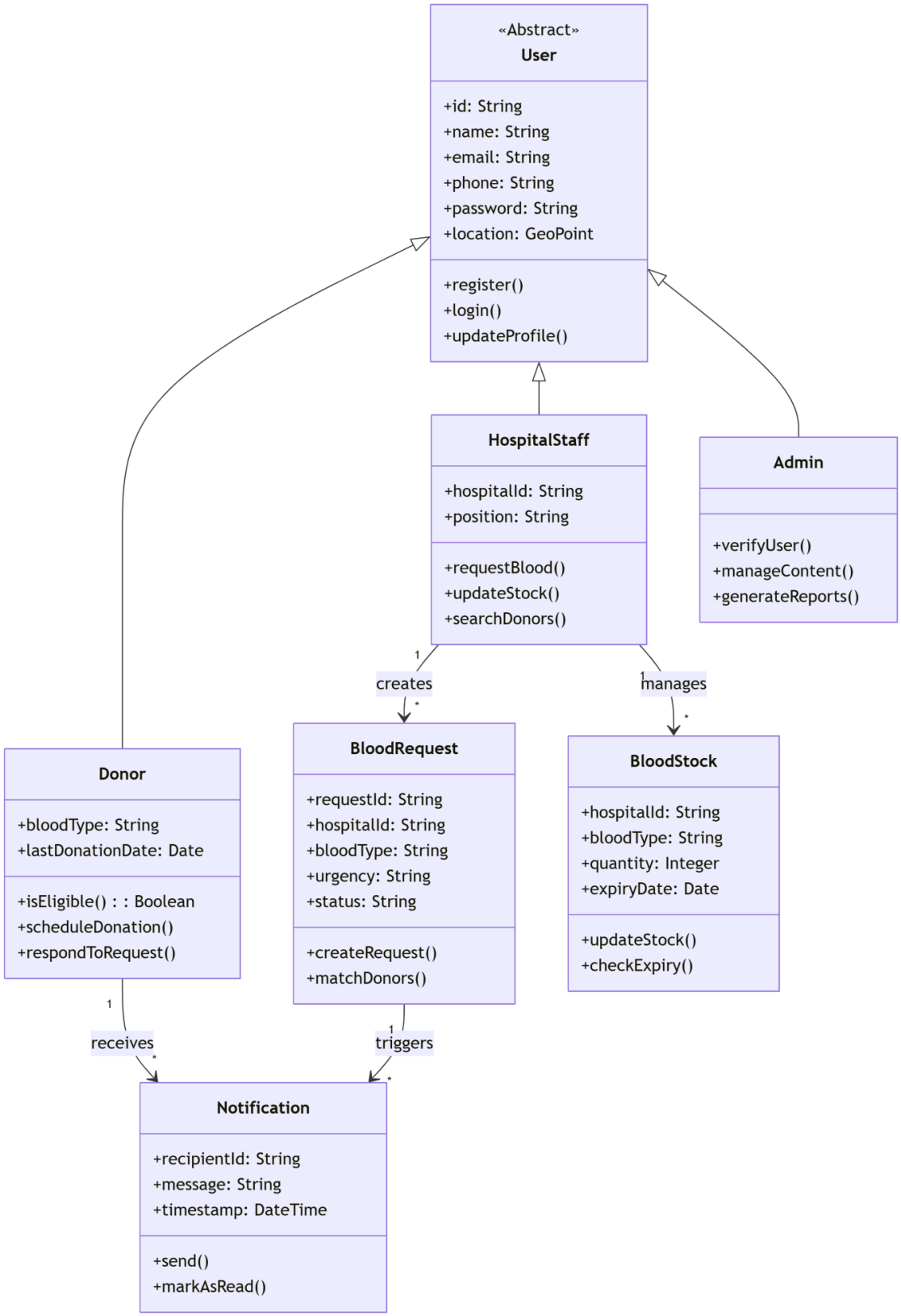


2.6.9.6 Activity Diagramming for Manage Users

2.6.9.7 Activity Diagramming for Logout

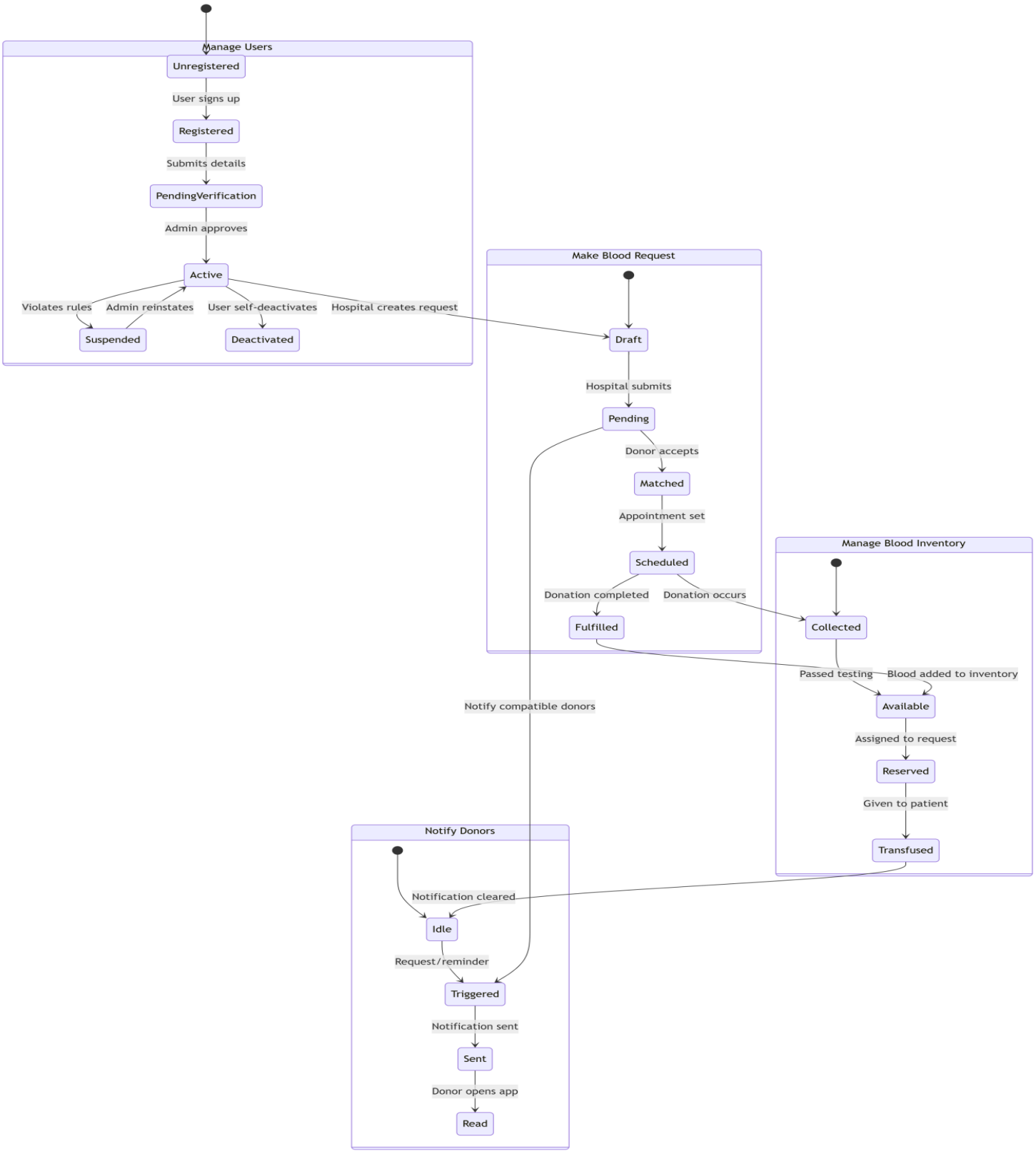


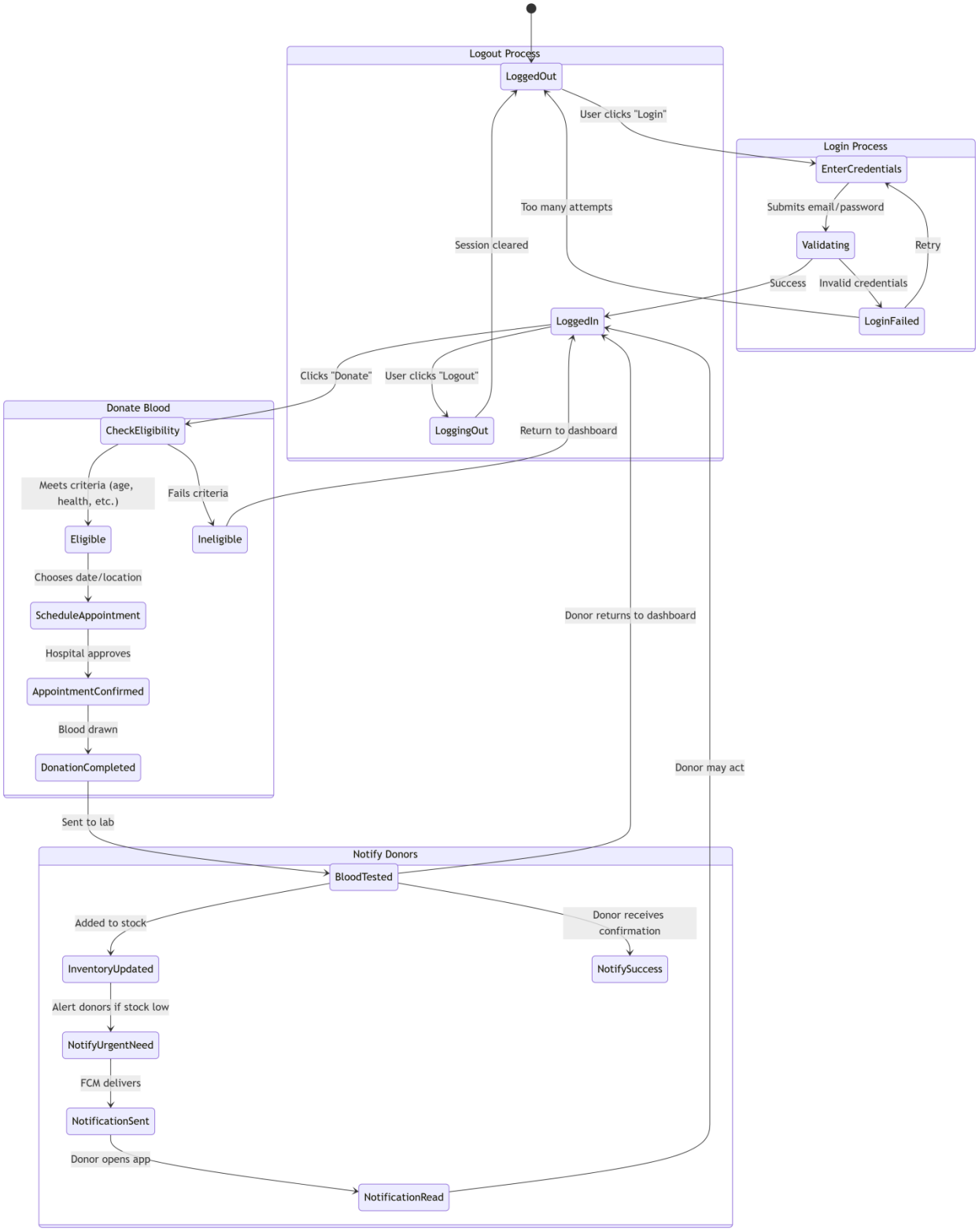
2.6.10 Class diagram



2.6.11 State chart diagram

2.6.11.1 State Chart Diagram for Manage Users, Make Blood Request ,Notify Donors, Manage Blood Inventory




Logout Process
LoggedOut
User clicks "Login"
Login Process
EnterCredentials
Submits email/password
Validating
Retry
Success
Invalid credentials
LoginFailed
Too many attempts
Session cleared
LoggedIn
Clicks "Donate"
User clicks "Logout"
LoggingOut
Return to dashboard
Donate Blood
CheckEligibility
Meets criteria (age, health, etc.)
Fails criteria
Eligible
Ineligible
Chooses date/location
ScheduleAppointment
Hospital approves
AppointmentConfirmed
Blood drawn
DonationCompleted
Donor returns to dashboard
Donor may act
Sent to lab
Notify Donors
BloodTested
Added to stock
Donor receives confirmation
InventoryUpdated
NotifySuccess
Alert donors if stock low
NotifyUrgentNeed
FCM delivers
NotificationSent
Donor opens app
NotificationRead

2.6.12 User interface Prototyping (High fidelity Prototype)

The high-fidelity prototype for the Ethiopian Blood Donation System represents a fully interactive, pixel-perfect mockup of the final web application, meticulously designed to address the critical challenges in the country's blood donation ecosystem. Built primarily in Figma with complementary use of Adobe XD for complex animations, this prototype will serve as the visual and functional blueprint for development. The design system encompasses three distinct yet interconnected user interfaces tailored to donor, hospital staff, and administrator needs. For donors, the prototype features an engaging landing page with compelling statistics about blood donation needs, followed by a multi-step registration process that collects personal details, health information, and location data through an intuitive form interface with real-time validation. The donor dashboard serves as the central hub, displaying upcoming appointments, donation history, and a notification center for urgent blood requests. A key innovation is the geolocation-powered donor search interface that integrates Google Maps API, allowing users to visually locate nearby hospitals and view blood requests filtered by blood type compatibility and proximity.

For hospital staff, the prototype demonstrates a comprehensive blood management dashboard that visualizes real-time inventory levels by blood type with color-coded alerts for critical shortages and impending expirations. The interface includes streamlined workflows for submitting blood requests with patient details and urgency levels, as well as tools to search for and directly contact compatible donors through the system. Administrative features are equally robust, with tools for verifying hospital accounts, monitoring system-wide donation patterns, and managing educational content about blood donation benefits and eligibility requirements. The prototype pays special attention to accessibility considerations, implementing high-contrast color schemes, responsive layouts for various device sizes, and multilingual support for both Amharic and English speakers.

Interactive elements are carefully crafted to mimic the final user experience, including clickable buttons with subtle hover animations, form fields with intelligent validation, and simulated push notifications through Firebase Cloud Messaging. The visual design employs a purposeful color palette dominated by red (#FF0000) to evoke the life-saving nature of blood donation, balanced with clean whites and trustworthy blues. Typography follows modern web standards with Poppins

Bold for headings and Open Sans Regular for body text, ensuring optimal readability across all user demographics. Material Design icons provide visual consistency, while carefully considered spacing and grouping adhere to UX best practices for information hierarchy.

The prototyping process follows an iterative approach, beginning with low-fidelity wireframes that evolve into fully-realized screens with authentic Ethiopian user scenarios in mind. Each user flow is meticulously mapped, from initial registration through donation scheduling and emergency request fulfillment. Advanced Figma features like interactive components, smart animate transitions, and prototyping presets bring the interface to life, demonstrating how users will navigate between screens and complete critical tasks. Particular attention is given to micro interactions - the small but meaningful animations and feedback cues that significantly enhance usability, such as loading indicators during search operations and success confirmation dialogs after form submissions.

Prior to development handoff, the prototype will undergo rigorous usability testing with representative Ethiopian users, including both tech-savvy urban donors and rural hospital staff with varying digital literacy levels. Test scenarios will evaluate the system's effectiveness in real-world conditions, measuring metrics like time-to-complete-critical-tasks and error rates during emergency blood requests. The insights gained will inform refinements to the interface, ensuring the final product meets the specific needs and constraints of Ethiopia's healthcare context. This comprehensive prototyping approach bridges the gap between conceptual design and technical implementation, providing developers with clear specifications for building the React.js frontend and Node.js backend while minimizing costly revisions during the development phase.

Chapter 3: System Design

3.1 Introduction

This chapter presents a comprehensive technical blueprint for the Ethiopian Blood Donation System, a transformative digital platform designed to modernize the nation's blood donation ecosystem. The system architecture is meticulously crafted to replace Ethiopia's current fragmented, paper-based processes with an integrated, cloud-native solution that bridges donors, hospitals, and blood banks real time. Built on a foundation of scalability, reliability, and security, the Ethiopian blood donation addresses critical gaps in the existing infrastructure, including inefficient donor matching, lack of centralized inventory tracking, and delayed emergency response times.

The system's design adopts an Incremental Development Methodology, allowing for phased implementation and continuous refinement based on stakeholder feedback. This approach ensures the platform can evolve alongside Ethiopia's healthcare infrastructure while delivering immediate, measurable benefits at each development stage. The Model-View-Controller (MVC) architectural pattern provides a structured separation of concerns, with React.js powering the dynamic frontend View layer, Node.js/Express handling the Controller logic for business operations, and PostgreSQL/Firebase serving as the Model layer for robust data management.

3.2 Purpose of the System

The Ethiopian Blood Donation System is designed to transform the nation's blood management infrastructure by creating a unified digital ecosystem that connects all stakeholders in real-time. The primary purpose of this system is to address critical gaps in Ethiopia's current blood donation processes by replacing fragmented, paper-based methods with an efficient technological solution. By integrating donor databases, hospital inventories, and emergency request systems into a single platform, Ethiopian Blood Donation System dramatically reduce response times during life-threatening situations while optimizing blood supply chain management. The system specifically target the bridge between blood bank and hospitals.

Beyond operational efficiency, EBDS serves as a strategic tool for strengthening Ethiopia's public health infrastructure. The system is purpose-built to enhance transparency in blood distribution, improve donor retention through engagement features, and provide data-driven insights for national health planning. the platform ensures accessibility across Ethiopia's diverse population

and varying levels of technological infrastructure. Basically , this system represents a vital step toward modernizing Ethiopia's healthcare system and saving countless lives through timely access to safe blood supplies.

3.3 Design Goals

The Ethiopian Blood Donation System is designed with clear objectives to create an effective and sustainable solution. First and foremost, the system prioritizes usability and accessibility, ensuring it can be easily adopted across Ethiopia's diverse user base. responsive design guarantees full functionality on mobile devices, which are widely used even in low-bandwidth areas, while streamlined workflows simplify critical processes like donor registration and emergency blood requests. These user-centric design choices aim to maximize engagement and minimize training requirements for healthcare staff and donors.

Security and performance form the foundation of the system's technical architecture. The platform is optimized for high-performance operation, capable of handling sudden surges in demand during emergencies through cloud auto-scaling, efficient database indexing, and intelligent caching of frequently accessed data like donor locations and blood type availability. These technical considerations ensure rapid response times when lives are at stake while maintaining strict data privacy standards.

The system's infrastructure emphasizes scalability, reliability, and maintainability to support nationwide deployment and long-term operation. A modular microservices architecture allows different components (donor matching, inventory management, notifications) to scale independently based on demand. Redundant cloud deployments with automated failover mechanisms guarantee continuous operation even during infrastructure disruptions, while comprehensive monitoring provides real-time system health visibility. The design incorporates cost-efficiency through smart resource allocation and open-source technologies, ensuring sustainable operation within Ethiopia's healthcare budget. These architectural decisions create a flexible platform that can grow with the country's evolving digital health ecosystem while remaining resilient in challenging operating environments.

3.4 Current Software Architecture

Description of Existing System

Ethiopia's blood donation management currently operates through a patchwork of manual and semi-digital processes. Most hospitals and blood banks rely on paper-based donor registration forms and handwritten inventory logs. Larger facilities may use standalone digital tools like Excel spreadsheets or basic Access databases, but these systems are not interconnected. Communication between hospitals and blood banks primarily occurs through phone calls, text messages, or in-person visits, creating significant delays in emergency situations.

The system lacks centralized coordination, with critical components operating in isolation. Donor information is stored locally at each collection site, making it impossible to search for compatible donors across different locations. Blood inventory levels are updated manually, often leading to outdated information. Emergency requests require staff to physically visit nearby facilities or make numerous phone calls to locate available blood units.

Key Limitations and Challenges

Several critical limitations plague the current architecture. Data fragmentation creates major obstacles - with no central donor database, hospitals cannot quickly locate potential donors during emergencies. The manual inventory tracking system frequently leads to stock discrepancies, resulting in both shortages and wastage of expired blood units. Communication bottlenecks delay emergency responses, sometimes for hours or even days when rare blood types are needed.

The system suffers from high error rates due to manual data entry, with occasional mismatches in blood type recording. There is no standardized process for tracking donor eligibility or donation frequency. Rural health centers face even greater challenges, as many lack any digital tools and rely entirely on paper records. Security vulnerabilities are prevalent, with sensitive donor information stored in unsecured spreadsheets or physical files that could be lost or damaged.

Need for Architectural Modernization

The current system's deficiencies create urgent need for a modernized architecture. Ethiopia's growing population and healthcare demands require a solution that can provide real-time visibility into blood supplies nationwide. The paper-based system cannot support rapid emergency response times that could save lives during accidents or medical crises. Data integrity issues threaten patient safety and undermine trust in the blood donation system.

A new architecture must address these gaps by implementing centralized, digital coordination. Automated processes could eliminate manual errors and reduce response times. Secure cloud-based storage would protect sensitive data while making it accessible to authorized personnel across different locations. Mobile integration would engage more potential donors and provide rural clinics with better tools. The proposed system aims to transform this critical healthcare infrastructure into a reliable, efficient, and transparent national service.

3.5 Proposed Software Architecture

The Ethiopian Blood Donation System adopts a cloud-native microservices architecture built on Google Cloud Platform (GCP) to deliver scalability, reliability, and real-time responsiveness. The system is structured into decoupled components—React.js frontend, Node.js/Express backend APIs, and specialized microservices (user management, donor matching, inventory tracking)—communicating via gRPC for internal calls and REST for external integrations. Data is persisted across Firestore (for real-time donor profiles) and PostgreSQL (for transactional records), with Redis caching high-frequency queries. This architecture directly addresses core design goals: scalability through auto-scaling Cloud Run instances, reliability via multi-region deployment (Europe-west1/west4), and performance via edge caching (Firebase CDN) and optimized geospatial queries.

Compared to Ethiopia's current paper-based and siloed digital systems, the proposed solution eliminates critical bottlenecks:

Data fragmentation → Unified database with real-time sync

Emergency delays → Instant notifications (SMS) replacing phone trees

Limited accessibility → Progressive Web App (PWA) supporting 2G/3G networks

3.5.1 Subsystem Decomposition

The Ethiopian Blood Donation System is architecturally decomposed into six specialized subsystems, each designed to handle specific functional requirements while maintaining clear interfaces for system-wide integration. This modular approach ensures maintainability, scalability, and the ability to independently update components without disrupting core operations.

Core Subsystems and Their Functions

User Management Subsystem Serves as the foundation of the system, handling all aspects of user registration, authentication, and profile management. Built on Firebase Auth, it provides secure identity verification and enforces role-based access controls (RBAC) across the platform. This subsystem maintains the master dataset of all donors, hospital staff, and administrators, feeding essential identity information to other components while protecting sensitive data through encryption and strict access policies.

Donor Matching Subsystem The intelligent core that processes emergency requests and identifies suitable donors through a multi-factor matching algorithm. It combines geolocation data (via Google Maps API) with blood-type compatibility analysis and donation history to generate prioritized donor lists. During matching operations, it pulls fresh donor availability data from the User Management Subsystem and coordinates closely with the Notification Subsystem to ensure rapid alert delivery.

Inventory Management Subsystem Maintains real-time visibility of blood supplies across all participating hospitals through PostgreSQL. This subsystem tracks critical metrics including stock levels, expiration dates, and utilization rates, automatically triggering replenishment alerts when supplies fall below predefined thresholds. It features specialized logic to prioritize rare blood types and integrates with the Matching Subsystem during emergencies to guide allocation decisions.

Reporting & Analytics Subsystem Transforms operational data into actionable insights through customizable dashboards and automated reports. Using a combination of materialized views and cached datasets, it provides healthcare administrators with real-time visibility into donation patterns, regional demand fluctuations, and system performance metrics while maintaining strict read-only access to preserve data integrity.

Admin Control Subsystem Offers comprehensive system management capabilities through an audited, privilege-tiered interface. This includes user account administration, content updates for educational resources, and system configuration management. All administrative actions generate immutable audit trails with detailed before/after snapshots to support compliance requirements.

Subsystem Dependencies and Integration

The architecture employs a hierarchical dependency model to maintain system coherence while allowing independent scalability. The User Management Subsystem serves as the primary upstream dependency, providing essential identity data to all other components. Downstream, the Matching and Inventory subsystems initiate cascading operations by engaging the Notification Subsystem when time-sensitive actions are required.

Critical cross-cutting services include

A shared Redis caching layer that reduces database load for frequently accessed data

A centralized API gateway that manages authentication and rate limiting

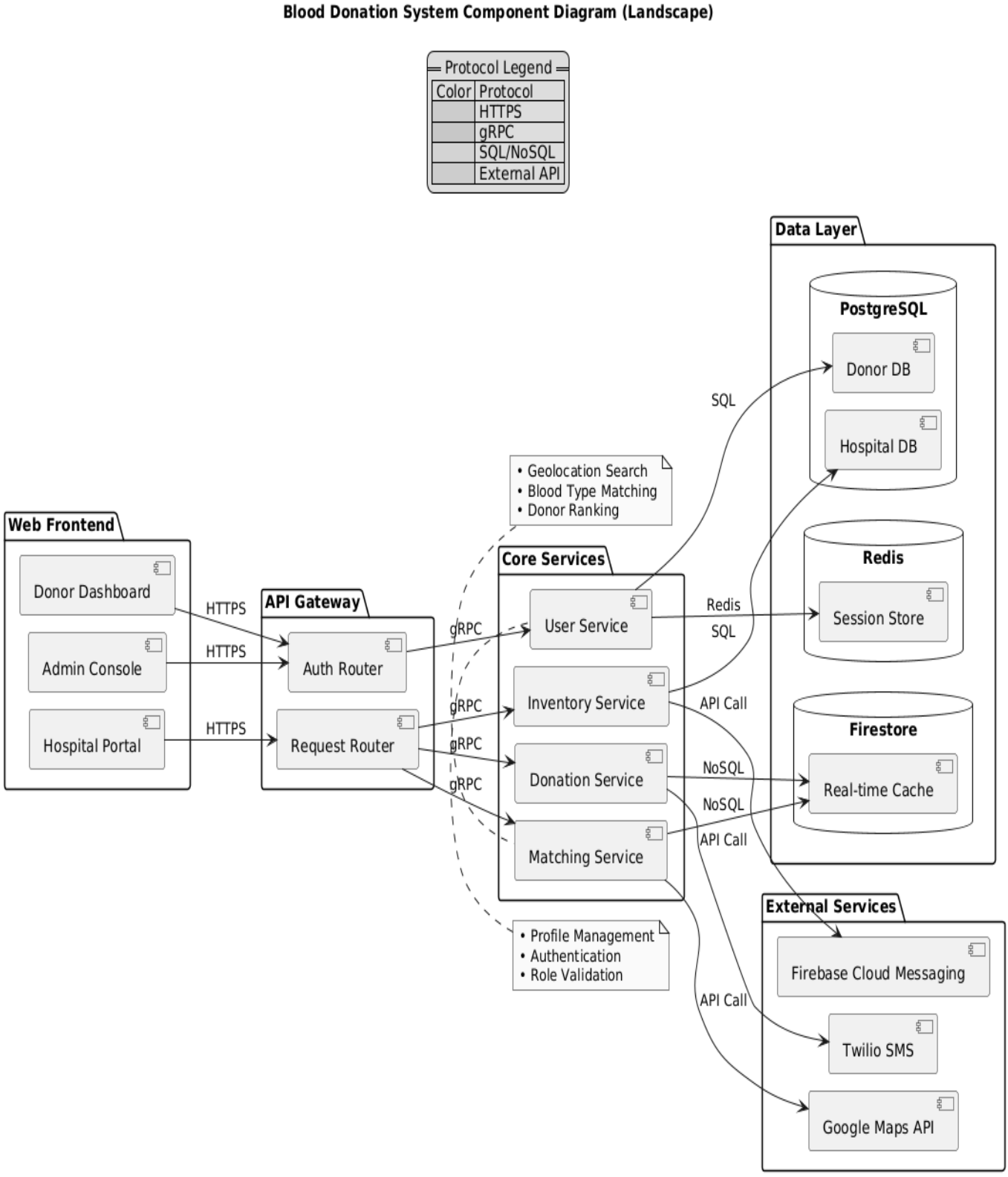
A monitoring service that tracks inter-subsystem communication health

Ethiopia-Specific Adaptations

The decomposition incorporates several features tailored to Ethiopia's operational environment:

Multi-tiered notification fallback (SMS → email) for unreliable networks.

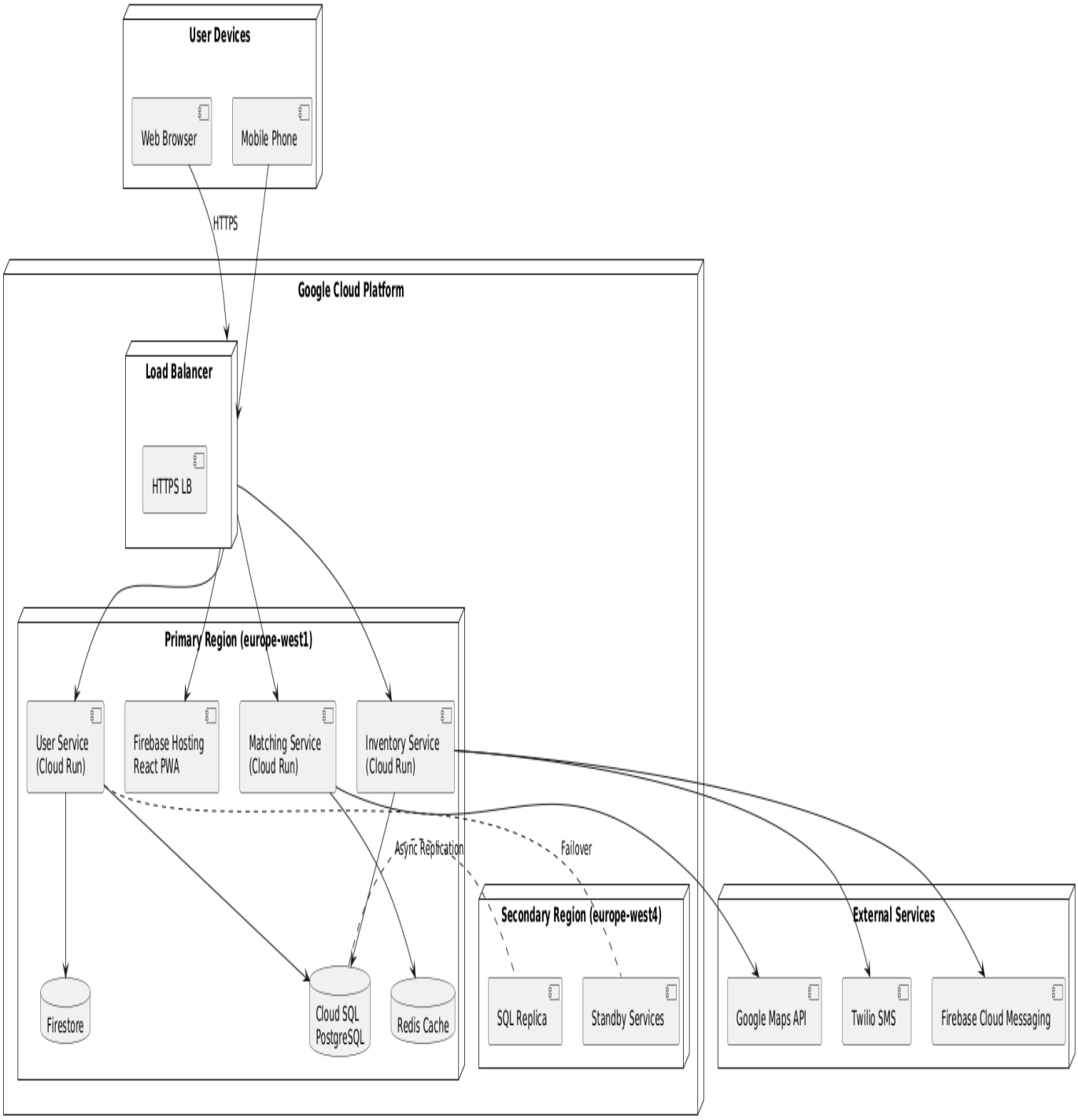
3.5.2 Component Diagram



3.5.3 Deployment Diagram

The Ethiopian Blood Donation System follows a cloud-native deployment architecture hosted entirely on Google Cloud Platform (GCP) for maximum reliability and performance. The system components are physically distributed across multiple layers - the frontend Progressive Web Application built with React.js is deployed on Firebase Hosting with global CDN distribution to ensure low-latency access across all regions of Ethiopia. Backend microservices including user management, donor matching, and inventory tracking run on Cloud Run containers that automatically scale based on demand, while data is persisted across three specialized databases: PostgreSQL for ACID-compliant transaction processing, and Memory store Redis for high-speed caching of frequently accessed data like blood availability status.

Performance is optimized through multiple architectural features: edge caching via Firebase CDN brings first-byte times under 500ms nationwide, Redis caching reduces database load for common queries, and protocol buffer serialization minimizes payload sizes for areas with slower connectivity. Security is enforced through VPC network isolation, identity-aware proxies for service communication, and automatic TLS certificate rotation. Monitoring and logging through Cloud Operations Suite provides real-time visibility into system health across all components. The diagram below illustrates these deployment relationships with boxes representing hardware nodes and lines showing communication protocols between components. This cloud-native approach delivers both the scalability needed for nationwide coverage and the reliability required for life-critical blood donation operations.


User Devices
Web Browser
Mobile Phone
HTTPS
Google Cloud Platform
Load Balancer
HTTPS LB
Primary Region (europe-west1)
User Service (Cloud Run)
Firebase Hosting React PWA
Matching Service (Cloud Run)
Inventory Service (Cloud Run)
Async Replication
Failover
Firestore
Cloud SQL PostgreSQL
Redis Cache
Secondary Region (europe-west4)
SQL Replica
Standby Services
External Services
Google Maps API
Twilio SMS
Firebase Cloud Messaging

3.5.4 Persistent Data Management

The Ethiopian Blood Donation System employs a structured yet flexible approach to managing persistent data, ensuring reliability, security, and compliance with healthcare regulations. The system stores multiple types of data, including user profiles (donor, hospital, and administrator details), blood donation records (donation dates, blood types, and test results), inventory data (real-time blood stock levels and expiry dates), geolocation information (donor and hospital coordinates for proximity-based matching), and notification logs (SMS and email). Future

For database technology, the system utilizes a hybrid architecture real-time user data and PostgreSQL (SQL) for structured transactional records. Firestore handles donor profiles and live updates with automatic synchronization for offline-capable devices, while PostgreSQL enforces strict data integrity through relational models, complex queries, and ACID compliance. Redis serves as an in-memory cache for high-speed access to frequently requested data, such as emergency blood availability or active alerts. External services like Google Maps API (geolocation) and Twilio (SMS) integrate seamlessly via secure APIs.

3.5.5 Detailed Database Design

3.5.5.1 Relational tables

The system employs a hybrid database architecture, with PostgreSQL 14 serving as the primary relational database for structured transactional data. Below is the complete schema design with detailed table definitions, constraints, and indexing strategies:

Table: donors

Stores comprehensive donor information, including eligibility criteria and contact details.

Column	Type	Constraints	Description
donor_id	UUID	PRIMARY KEY	Unique system identifier
national_id	VARCHAR(20)	UNIQUE, NOT NULL	Government-issued ID
first_name	VARCHAR(50)	NOT NULL	
last_name	VARCHAR(50)	NOT NULL	
phone	VARCHAR(15)	NOT NULL	Encrypted storage
blood_type	CHAR(3)	CHECK (valid blood type)	Values: A+, A-, B+, etc.
last_donation_date	TIMESTAMPTZ		NULL for first-time donors
geohash	VARCHAR(12)		Optimized for location queries
is_active	BOOLEAN	DEFAULT TRUE	Soft deletion flag

Table: hospitals

Tracks healthcare facilities authorized to request blood.

Column	Type	Constraints	Description
hospital_id	UUID	PRIMARY KEY	
name	VARCHAR(100)	NOT NULL	Facility name
region	VARCHAR(50)	NOT NULL	Ethiopian administrative region
latitude	DECIMAL(10,7)		GPS coordinates
longitude	DECIMAL(10,7)		
verification_status	VARCHAR(20)	CHECK (valid status)	Values: pending, verified, rejected

2. Transactional Tables

Table: donations

Records completed blood donations with medical metadata.

Column	Type	Constraints	Description
donation_id	UUID	PRIMARY KEY	
donor_id	UUID	FOREIGN KEY REFERENCES donors(donor_id) ON DELETE CASCADE	
hospital_id	UUID	FOREIGN KEY REFERENCES hospitals(hospital_id)	
blood_type	CHAR(3)	NOT NULL	Redundant for query performance
expiry_date	TIMESTAMPTZ	GENERATED ALWAYS AS (donation_date + INTERVAL '42 days') STORED	
test_results	JSONB		Structured lab da

Table: inventory

Real-time blood stock levels per hospital.

Column	Type	Constraints	Description
inventory_id	UUID	PRIMARY KEY	
blood_type	CHAR(3)	NOT NULL	
units_available	INTEGER	CHECK (units_available >= 0)	
last_audit	TIMESTAMPTZ	DEFAULT NOW()	

3. Operational Tables

Table: requests

Manages blood requests from hospitals.

Column	Type	Constraints	Description
request_id	UUID	PRIMARY KEY	
priority	VARCHAR(10)	CHECK (priority IN ('routine','urgent','emergency'))	
status	VARCHAR(20)	CHECK (status IN ('open','fulfilled','expired'))	

3.5.5.2 Normalization

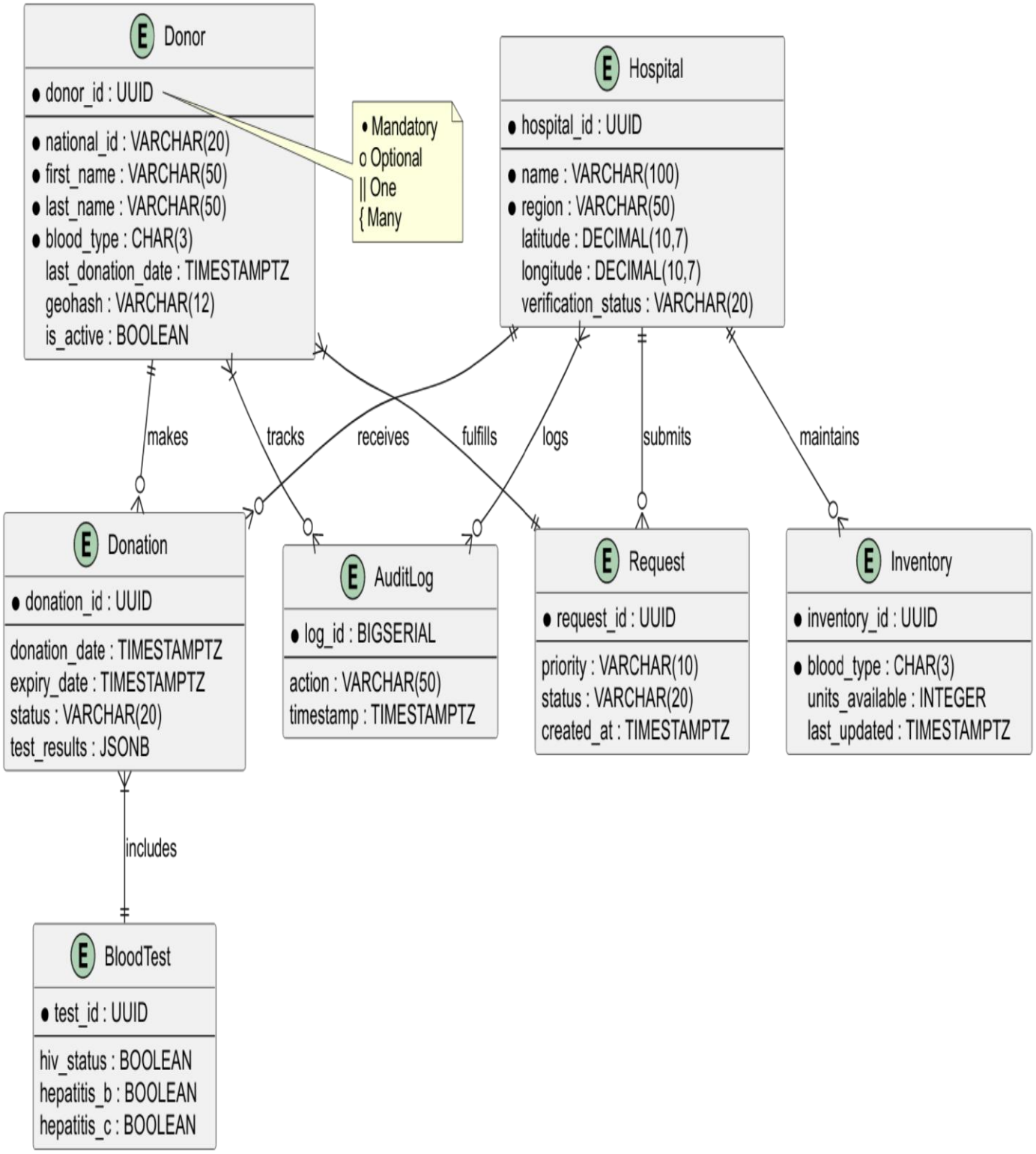
The database schema adheres to third normal form (3NF) to eliminate redundancy and ensure data integrity while maintaining query performance. All tables undergo the following normalization steps:

First Normal Form (1NF): Each table enforces atomic values by eliminating repeating groups. For example, the donations table stores each blood donation as a separate record rather than aggregating multiple donations in an array. Composite attributes like addresses are split into discrete columns (e.g., region, city in the hospitals table), and multi-value fields such as test results are stored as normalized JSONB to preserve atomicity while accommodating flexible medical data.

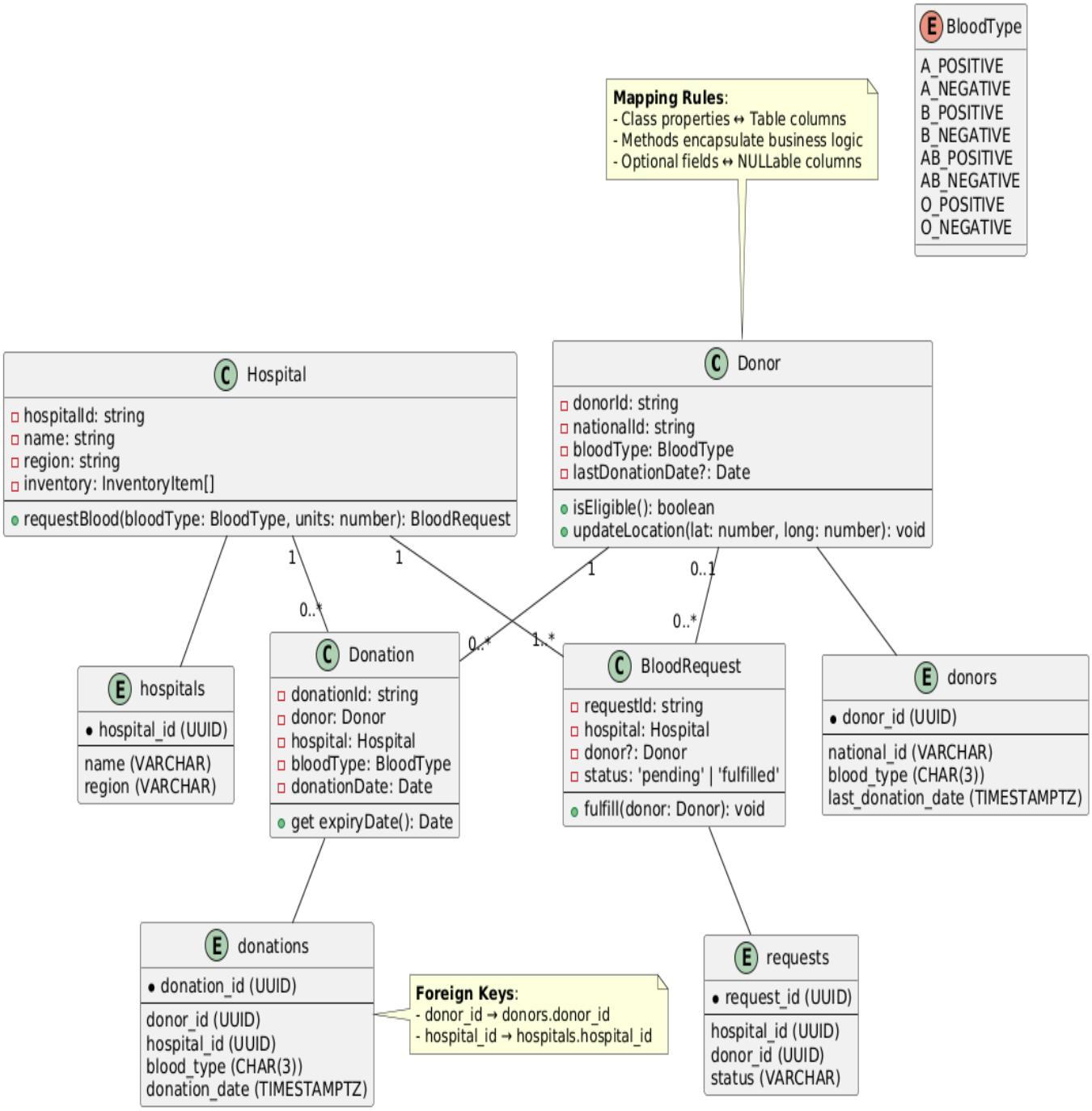
Second Normal Form (2NF): Partial dependencies are removed by ensuring all non-key attributes fully depend on the primary key. The inventory table, for instance, includes a composite unique constraint on (hospital_id, blood_type) to prevent duplicate entries, while units_available depends entirely on this combination. Similarly, the donors table separates location data into geohash coordinates rather than embedding them in a single text field.

Third Normal Form (3NF): Transitive dependencies are eliminated by isolating attributes not directly dependent on the primary key. The donations table avoids storing derived donor details (e.g., names or blood types redundantly) and instead references the donors table via foreign keys. Hospital verification status in the hospitals table depends solely on hospital_id, not on secondary attributes like location.

3.5.5.3 EER


Donor
donor_id : UUID
national_id : VARCHAR(20)
first_name : VARCHAR(50)
last_name : VARCHAR(50)
blood_type : CHAR(3)
last_donation_date : TIMESTAMPTZ
geohash : VARCHAR(12)
is_active : BOOLEAN
Mandatory
Optional
One
Many
Hospital
hospital_id : UUID
name : VARCHAR(100)
region : VARCHAR(50)
latitude : DECIMAL(10,7)
longitude : DECIMAL(10,7)
verification_status : VARCHAR(20)
makes
tracks
receives
fulfills
logs
submits
maintains
Donation
donation_id : UUID
donation_date : TIMESTAMPTZ
expiry_date : TIMESTAMPTZ
status : VARCHAR(20)
test_results : JSONB
AuditLog
log_id : BIGSERIAL
action : VARCHAR(50)
timestamp : TIMESTAMPTZ
Request
request_id : UUID
priority : VARCHAR(10)
status : VARCHAR(20)
created_at : TIMESTAMPTZ
Inventory
inventory_id : UUID
blood_type : CHAR(3)
units_available : INTEGER
last_updated : TIMESTAMPTZ
includes
BloodTest
test_id : UUID
hiv_status : BOOLEAN
hepatitis_b : BOOLEAN
hepatitis_c : BOOLEAN

3.5.5.4 OO-Relational mapping



3.5.6 Access control and security

The Ethiopian Blood Donation System enforces a comprehensive security framework designed to safeguard sensitive data while ensuring controlled access to system resources. Authentication is implemented through Firebase Authentication, combining email/password login with SMS-based OTP verification for critical actions, while hospital staff and blood bank undergo stricter two-factor authentication (2FA) using hardware tokens. Role-Based Access Control (RBAC) governs authorization, with predefined roles—Donor, Hospital Staff, Admin —each granted granular permissions to restrict access to only relevant data and functions. For instance, donors can view and modify their profiles, whereas hospital staff may update inventory levels but cannot access other hospitals' records.

To ensure data privacy, all sensitive information—including personal identifiers and medical records—is encrypted using AES-256 at rest and TLS 1.3 in transit, with field-level encryption applied to confidential health data via Google Cloud KMS. Database security is further reinforced through row-level policies in PostgreSQL, restricting queries to only data owned by the authenticated user or institution. Audit logs capture every access and modification event, stored immutably to support compliance reviews and forensic investigations.

For Ethiopia-specific requirements, the system adheres , including localized SMS alerts or email. TPOTP for users in low-bandwidth areas. Regular penetration testing and automated anomaly detection (e.g., for brute-force attacks) further harden the system against threats. This multi-layered approach balances robust protection with usability, ensuring compliance with both Ethiopian regulations and global standards (HIPAA/GDPR) while maintaining performance across diverse network conditions.

3.5.7 Global Software Control

The Ethiopian Blood Donation System employs a centralized orchestration approach to manage control flow, ensuring seamless coordination between subsystems while maintaining scalability and fault tolerance. Requests are initiated through the API Gateway, which acts as the single entry point for all client interactions—whether from donors, hospitals, or administrators.

Subsystems synchronize through event-driven architecture, leveraging Cloud Pub/Sub for real-time communication. For example, when a hospital submits a blood request, the Inventory Service first checks stock levels; if unavailable, it publishes an event to the Matching Service, which identifies compatible donors via geolocation and blood type, then triggers the Notification Service to dispatch alerts. This decoupled design ensures subsystems operate independently yet remain synchronized, with transactional outbox patterns guaranteeing message delivery even during failures.

For data consistency, manage distributed transactions—such as donation recording, which updates the donor's last donation date (User Service), reduces inventory (Inventory Service), and logs the transaction (Reporting Service). The Circuit Breaker pattern prevents cascading failures by isolating malfunctioning services (e.g., if the SMS gateway fails, notifications default to email).

Scheduled cron jobs (via Cloud Scheduler) handle batch processes like nightly expiry checks for blood units, while idempotent APIs ensure safe retries for interrupted requests. All control flows are logged and monitored through Cloud Operations Suite, providing real-time visibility into system health. This structured yet flexible control mechanism ensures reliability across Ethiopia's diverse infrastructure conditions, balancing rapid response times with data integrity.

3.5.8 Boundary Conditions

The Ethiopian Blood Donation System implements robust boundary condition handling to maintain system stability across all operational extremes. During startup sequences, the system executes pre-flight checks including database connectivity verification, third-party service handshakes (SMS gateways, mapping APIs), and configuration validation before accepting live traffic, with a progressive rollout of services to prevent cascading failures. For controlled shutdowns, the system implements graceful degradation - completing active blood matching operations within a 90-second timeout period while rejecting new requests, persisting all transaction states, and emitting comprehensive shutdown telemetry for audit purposes.

The system demonstrates particular resilience in error scenarios: invalid API requests trigger immediate schema validation with localized Amharic/English error messages, while failed dependency calls (e.g., unavailable geolocation services) activate automated failover to cached ZIP-code based matching algorithms. Under extreme load conditions such as mass casualty events

triggering thousands of concurrent blood requests, the architecture responds through multi-layered throttling - prioritizing emergency requests via a weighted queue system and temporarily simplifying matching algorithms to maintain sub-2-second response times. For infrastructure failures, regional database outages automatically trigger read-only mode with cached inventory data.

Special consideration is given to Ethiopia-specific challenges: during prolonged internet outages, hospital clients can continue limited operations through pre-cached donor lists and emergency protocol PDFs, with all changes automatically synchronizing when connectivity restores using conflict-resolution algorithms. The system maintains functionality even under resource exhaustion scenarios through aggressive memory caps on analytics processes and automated log purging when disk utilization exceeds 80%. All boundary behaviors are continuously monitored through adaptive health checks that adjust sensitivity based on real-time system load, ensuring both stability and performance across Ethiopia's diverse technological infrastructure conditions.

Summary

The Blood Donation Management System is a comprehensive web-based platform designed to modernize Ethiopia's blood donation ecosystem. It addresses critical challenges like fragmented donor databases, inefficient communication, and lack of real-time inventory tracking by creating a centralized digital solution. The system connects three key user groups:

Donors can register profiles, schedule appointments, track donation history, and receive urgent alerts via SMS/app notifications.

Hospital staff can request specific blood types, update real-time inventory levels, and search for nearby donors using geolocation.

Administrators manage user roles, monitor nationwide donation patterns, and generate reports to optimize blood distribution.

Built with React.js (frontend) and Node.js/Express (backend), the system leverages Google Cloud Platform for scalability and for authentication/notifications. Key innovations include:

1. Geolocation-powered donor matching using Google Maps API.

2. Automated alerts for emergencies, appointments, and blood expiry.

3. Robust security through role-based access controls and encryption.

The project followed an Incremental methodology, ensuring iterative development aligned with stakeholder feedback. By digitizing paper-based processes, the system aims to reduce response times during emergencies, minimize blood wastage, and foster a culture of regular donation.

Conclusion

This project represents a significant step toward transforming Ethiopia's healthcare infrastructure through technology. The Blood Donation Management System directly tackles life-threatening inefficiencies by enabling real-time coordination between donors, hospitals, and blood banks. Its user-centric design ensures accessibility across diverse demographics, while cloud-based architecture guarantees scalability for nationwide deployment.

Though challenges like internet reliability in rural areas persist, features such as offline caching and SMS fallbacks provide practical solutions. The system's success hinges on widespread adoption, which can be achieved through partnerships with health authorities and NGOs.