LifeLine — Web Project Document

Project: LifeLine — Smart Healthcare & Emergency Response System (Web)

Purpose: A web platform that coordinates ambulances, hospitals, organ transport, blood donation, volunteers, and city control centers to dramatically reduce emergency response times and improve resource allocation.

---

1. Executive Summary

LifeLine is a web application that connects citizens, ambulance teams, hospitals, blood banks, and volunteers through a unified dashboard and mobile-friendly web interface. It provides AI-powered routing, hospital auto-allocation, real-time tracking, organ transport monitoring, and blood availability management to ensure timely, data-driven emergency responses.

---

2. Problem Statement

Emergencies require fast, coordinated action. Current gaps include delayed ambulance arrival due to traffic, lack of hospital capacity visibility, poor coordination for organ and blood transport, and limited volunteer mobilization. These inefficiencies cause preventable mortality and resource wastage.

---

3. Solution Overview

A responsive web application that:

- Optimizes ambulance and organ transport routes using live traffic and AI.
- Auto-allocates hospitals based on real-time capacity and specialization.
- Provides a central command dashboard with analytics and heatmaps.
- Manages blood stock and donor matching.
- Connects trained volunteers to nearby emergencies.

---

4. Top 10 Features (Scope for this web project)

1. **AI-Based Route Optimization**
    - Fastest/safest routing using live traffic, weather, and incident feeds.
    - Auto-reroute when congestion or roadblocks are detected.
    - API hooks for traffic-signal priority (green-corridor).
2. **Hospital Auto-Allocation**
    - Automatic selection of nearest hospital with required ICU/ER capacity and specialists.
    - Real-time bed, ventilator, and staff availability display.
3. **Emergency Broadcast to Traffic**
    - Location-based alerts to nearby vehicles via push notifications and partnered navigation apps.
    - Interface to request temporary traffic-signal priority from local traffic management.
4. **Emergency QR for Patients**
    - User-generated medical QR containing blood group, allergies, chronic conditions, and emergency contact.
    - QR scan endpoint for paramedics/hospitals to fetch essential data instantly.
5. **Multi-Role Dashboard**
    - Public: request ambulance, see nearby hospitals, register as donor/volunteer.
    - Ambulance Staff: live navigation pane, patient data, hospital sync.
    - Hospital Staff: incoming patient list, bed management, blood/organ inventory.
    - Admin: city-wide active incidents, vehicle statuses, analytics.
6. **Central Command & Analytics Dashboard**
    - Heatmaps of emergencies, response time analytics, ambulance utilization.

    - Predictive alerts for hotspots and peak demand windows.

7. **Organ Transport Tracking**
    - Real-time GPS tracking and ETA broadcasting for organ couriers.
    - IoT sensor integration for temperature and container-condition alerts.
    - Chain-of-custody logs (timestamped handovers).
8. **Blood Donation & Availability System**
    - Live blood stock (type & quantity) per hospital/blood bank.
    - Smart donor matching and urgent request push notifications.
    - Donor scheduling and eligibility tracking.
9. **Emergency Volunteer Network**
    - Verification workflow for volunteers.
    - Geo-dispatch to nearest trained volunteers with role selection (first-aid, transport, traffic marshals).
    - Contribution tracking and rewards/badges.
10. **Disaster Mode / Mass Casualty Support**
    - Special workflow for disasters: multi-hospital coordination, triage prioritization, mass volunteer mobilization, and broadcast requests for blood/meds.

---

5. Target Users

- General public (requestors)
- Ambulance drivers and paramedics
- Hospital staff (ER, ICU coordinators)
- Blood bank administrators
- Volunteer coordinators and verified volunteers

- City traffic/control center admins

---

6. High-Level Architecture

- **Frontend:** React (web), responsive design (mobile-first), Progressive Web App capabilities for push notifications.
- **Backend:** Django REST Framework (Python) — REST APIs and WebSocket streams for real-time updates.
- **Database:** PostgreSQL for relational data; Redis for caching and realtime state; Time-series store (InfluxDB) optional for IoT/vitals.
- **Maps & Routing:** Google Maps API / OpenStreetMap + custom routing service with traffic feed.
- **Notifications:** Firebase Cloud Messaging + Twilio for SMS/voice.
- **IoT Integration:** MQTT broker for organ-temperature sensors and ambulance telemetry.
- **Authentication:** OAuth2 / JWT; role-based access control.
- **Hosting:** AWS (ECS / EBS / RDS) or Render/Vercel for frontend.
- **Architecture Diagram:** Frontend ↔ API Gateway ↔ Microservices (Routing, HospitalManager, Organs, BloodBank, Volunteers, Analytics) ↔ Databases & Message Broker

---

7. Security & Compliance

- TLS for all transport-level security.
- Role-based access and field-level encryption for sensitive medical data.
- Audit logging for all access to health records and organ/blood chain-of-custody.
- Compliance considerations: follow local health data regulations (HIPAA-equivalent awareness) and use patient-consent flows for record access.

---

8. UI / Pages (Web)

- Landing / Public info page
- Login / Signup (roles: public, ambulance, hospital, admin, volunteer)
- Public dashboard: Request ambulance, donate blood, view nearest hospitals
- Ambulance panel: Navigation, incident details, patient QR scan, live vitals feed
- Hospital panel: Incoming patients, bed & blood inventory, organ & blood requests
- Admin panel: City map, active incidents list, analytics, volunteer management
- Volunteer app page: Nearby requests, training materials, contribution history

---

9. Tech Stack (Expanded Details)

Frontend

- **Framework:** React.js for interactive SPA experience
- **UI/Styling:** TailwindCSS for rapid styling and responsiveness
- **Mapping & Visualization:** Google Maps API, Leaflet.js for dynamic maps and heatmaps
- **Charts & Analytics:** Recharts or Chart.js for dashboards
- **Push Notifications:** Firebase Cloud Messaging
- **PWA Features:** Service workers for offline support

Backend

- **Framework:** Django REST Framework (Python) for API endpoints
- **Realtime Communication:** Django Channels + WebSockets for live updates

- **Business Logic:** AI routing, hospital allocation, and prediction algorithms
- **Task Queue:** Celery + Redis for background jobs (alerts, notifications)

Database

- **Relational:** PostgreSQL for structured data (users, incidents, hospital info)
- **Caching / Realtime:** Redis for session management, ambulance location caching
- **Time-Series / IoT:** InfluxDB or TimescaleDB for organ temperature and vitals logging

Security & Auth

- **Authentication:** JWT tokens / OAuth2
- **Role Management:** RBAC (role-based access control) for multiple user types
- **Encryption:** TLS for network traffic, field-level encryption for sensitive data

DevOps & Hosting

- **Frontend:** Vercel for React deployment
- **Backend:** Render, AWS ECS, or Heroku for Django app
- **Database Hosting:** AWS RDS or managed PostgreSQL services
- **CI/CD:** GitHub Actions for automated builds and deployments

Notifications & Communication

- **SMS/Voice:** Twilio for emergency alerts
- **Push Notifications:** Firebase Cloud Messaging
- **Real-Time Events:** WebSockets via Django Channels or Socket.io

IoT & Device Integration

- **Sensors:** MQTT protocol for organ temperature sensors and ambulance telemetry

- **Mobile Integration:** Optional QR scanner on mobile devices for patient info

---

10. 24-Hour Hackathon Strategy

- **Goal:** Demonstrate all 10 features using live mock data, partial simulations, and interactive dashboards.
- **Execution Plan:**
    - Hour 0–3: Project setup, authentication, frontend layout.
    - Hour 4–7: Incident request, ambulance tracking, hospital auto-allocation mock.
    - Hour 8–12: QR simulation, organ transport mock, blood stock and donor tracking.
    - Hour 13–17: Multi-role dashboards, heatmap analytics, real-time updates.
    - Hour 18–21: Disaster mode toggle, notifications, UI polish.
    - Hour 22–24: Demo preparation, walkthrough, final testing.
- **Focus:** Visual clarity, interactive maps, real-time simulation to impress judges.
- **Tip:** Use mock IoT data for organ/blood transport, random route optimization, and dummy patient info for live demonstration.