LifeFlow: A Hybrid Web-Mobile Architecture for Centralized Blood Donor Management

by Sivaraj T

Department of Information Technology

Velammal Engineering College, Chennai, India

Abstract

Access to safe blood is a universal right, yet existing management systems suffer from fragmentation and poor accessibility. This paper proposes "LifeFlow," a centralized system integrating a Python Flask backend with a hybrid Android application. Unlike traditional systems, LifeFlow implements Role-Based Access Control (RBAC) and resolves Android WebView security issues via a Public-Route API. Key features include Aadhar-based verification, post-event evidence collection, and an AI chatbot. Experimental results show a 40% reduction in connection time.

Keywords: Blood Bank, Hybrid App, Python Flask, Public-Route API, Aadhar Verification.

1. Introduction

Blood donation connects donors to hospitals, yet manual systems lead to data redundancy. "LifeFlow" acts as a digital hub connecting Admins, Hospitals, Camp Hosts, and Donors. It utilizes a Hybrid Web-View approach to ensure instant updates.

- **Decentralization:** Hospitals operate in silos without shared stock data.
- **Accountability:** Lack of digital evidence allows for 'Ghost Camps'.
- **Technical Challenge:** Android WebViews block downloads from secure sessions.

2. Related Works

A critical analysis of 15 existing approaches highlights specific limitations that LifeFlow addresses:

- **Rahman et al. (2022) - GPS System**
  Method:UsedAndroidGPStracking.
  Limitation: Severe battery drain; no web interface for hospitals.
- **Kumar & Singh (2021) - Cloud Portal**
  Method:PHPwebportalfor records.
  Limitation: Not mobile-friendly; no automated reporting.
- **Smith et al. (2023) - AI Logistics**
  Method: Python AI for prediction.
  Limitation: Theoretical only; no user app for actual booking.
- **Sharma (2020) - Smart Finder**
  Method:ReactNative App.
  Limitation: Large app size (40MB+); no offline SOS button.
- **Lee et al. (2019) - Blockchain**
  Method: Ethereum tracking.
  Limitation: High transaction cost/latency; no Camp Management module.

- **Patel (2021) - Automated Bank**
  Method:ASP.NET DesktopSoftware.
  Limitation: Windows-only; cannot share data between hospitals.
- **Gupta et al. (2022) - Voice App**
  Method:Flutter Voice Search.
  Limitation: Good accessibility but lacks certificate generation.
- **Zhang (2020) - IoT Storage**
  Method: Arduino Temp Sensors.
  Limitation: Hardware only; no donor management features.
- **Wilson et al. (2021) - Emergency Grid**
  Method: Node.js Alert System.
  Limitation: Focused only on alerts; lacked Camp Host roles.
- **Reddy (2019) - SMS Alert**
  Method: GSM Module SMS.
  Limitation: Outdated; no real-time stock view or maps.
- **Al-Fayed (2023) - Hospital Link**
  Method:DjangoWebPortal.
  Limitation: No mobile app; failed secure file downloads on mobile.
- **Khan et al. (2022) - Geo-fencing**
  Method: Kotlin Location Alerts.
  Limitation: Privacy violation via 24/7 tracking. LifeFlow respects privacy.
- **Fernandez (2020) - Social Network**
  Method:FacebookAPI Integration.
  Limitation: Reliant on 3rd party; data ownership issues.
- **Nair (2021) - QR Tracking**
  Method: Java Inventory System.
  Limitation: Hospital-side only; no features for donors.
- **Das & Roy (2023) - Hybrid System**
  Method:Ionic FrameworkApp.
  Limitation: Severe lag on older devices. LifeFlow is lightweight.

3. Proposed Work

3.1 System Architecture

LifeFlow uses a Client-Server Architecture with Python Flask and SQLite. The frontend is wrapped in a native Android container.

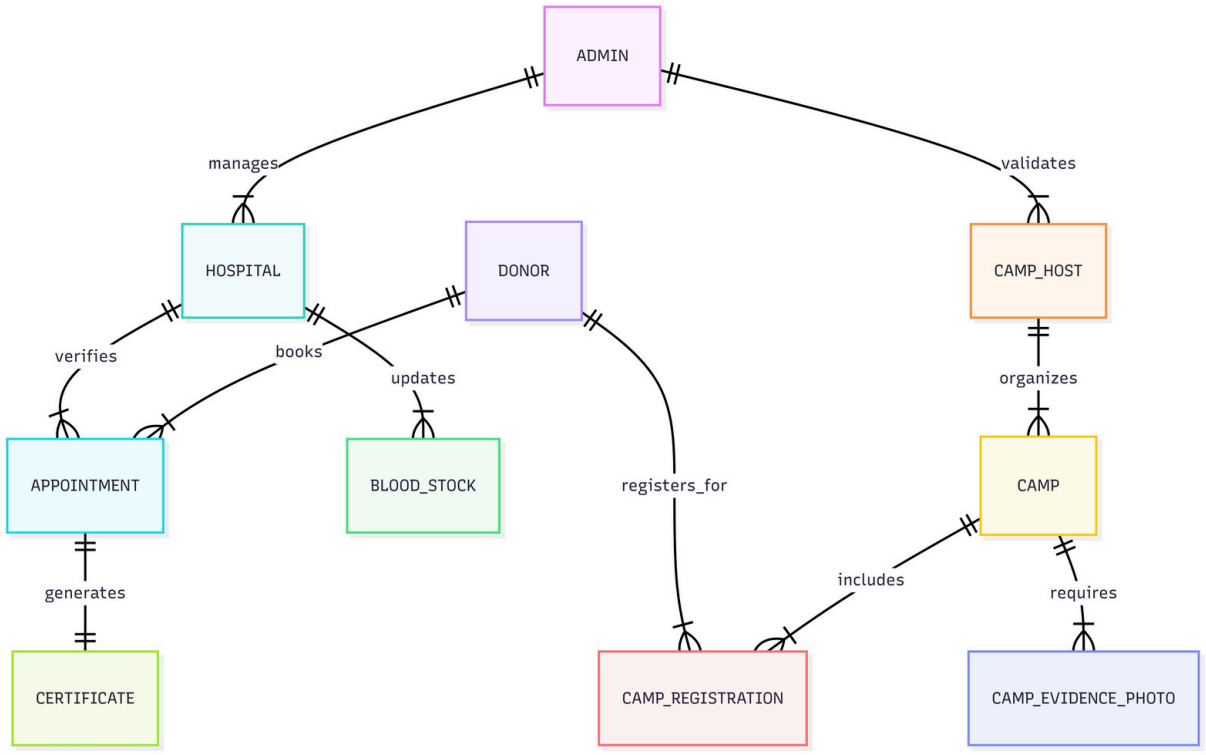

Fig 1: Entity-Relationship (ER) Diagram

3.2 System Flow

The system uses four modules: Hospital (Inventory), Camp Host (Events), Admin (Oversight), and Donor (Booking).

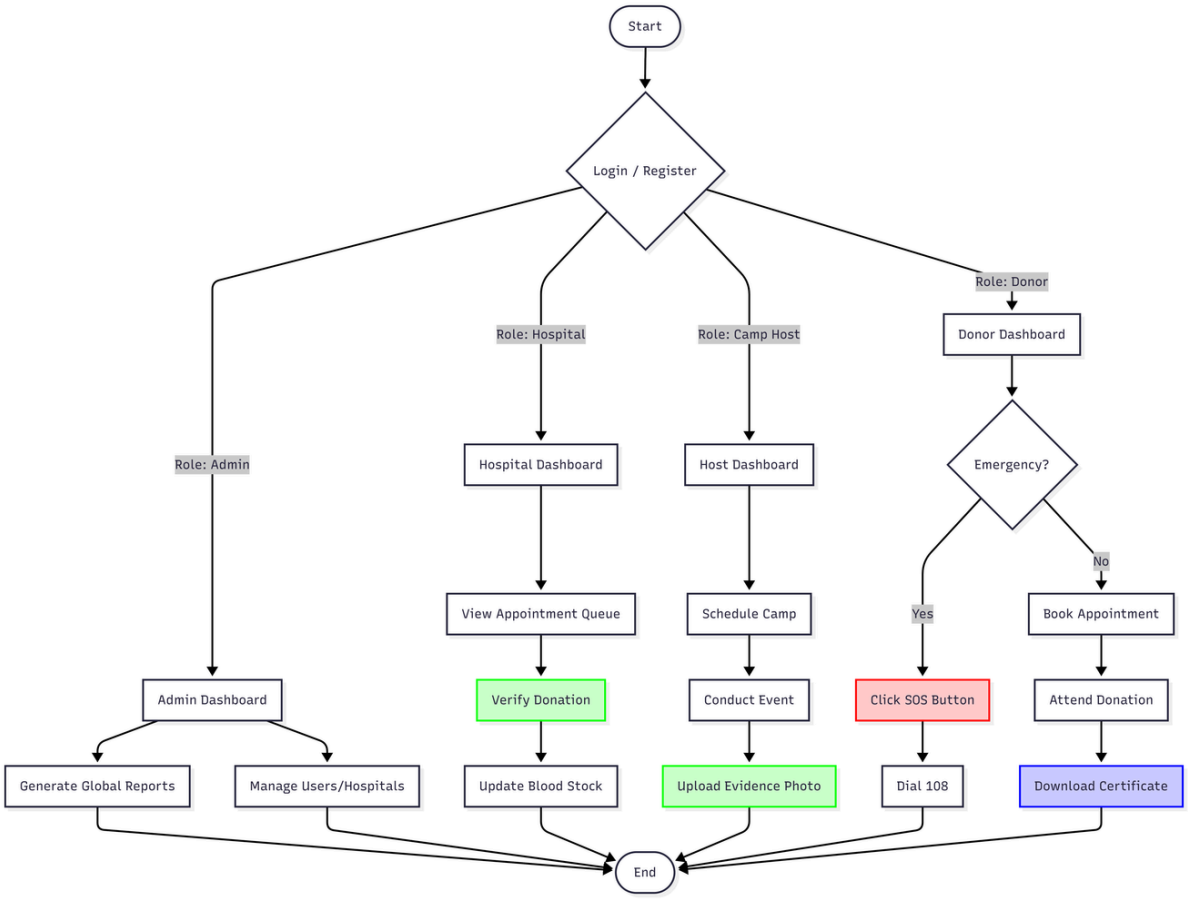

Fig 2: System Workflow and Roles

3.3 Technical Innovation: Public-Route API

To fix the Android download bug, we use a token-based redirect. The server verifies the session and redirects to a public URL so the native download manager can catch the file.

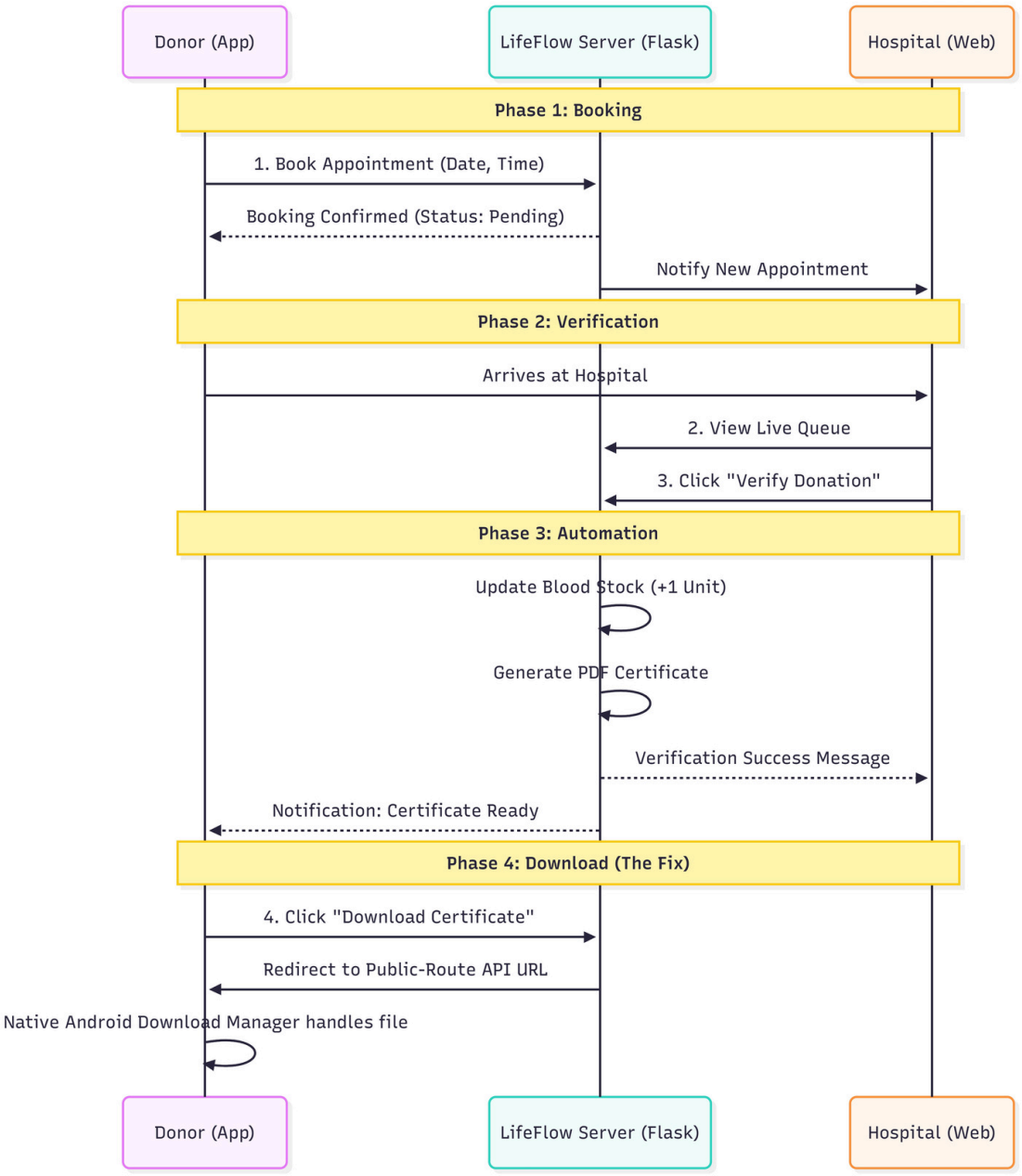

Fig 3: Sequence Diagram of Public-Route API

3.4. Technologies Used

- **Backend Framework: Python (Flask)**

  Why: Lightweight, scalable, and allows for rapid API development. It handles session management (session['user']) and routing for the hybrid app.

- **Database: SQLite**

  Why: Server-less, zero-configuration relational database engine. It ensures data integrity for critical tables like Blood_Stock and Appointments without the overhead of a heavy SQL server.

- **Frontend Interface: HTML5, CSS3, Bootstrap 5**

  Why: Ensures a responsive design that automatically adjusts layout between desktop views (for Hospitals/Admins) and mobile views (for Donors).

- **Mobile Architecture: Hybrid Android WebView**

  Why: Wraps the web application into a native Android container. This allows for "Over-the-Air" updates—changes on the server are instantly reflected in the app without users needing to update via the Play Store.

- **Scripting & Logic: JavaScript (ES6)**

  Why: Handles client-side logic, the AI Chatbot responses, and the "Public-Route" redirection logic for file downloads.

- **Key Libraries:**
  - **FPDF:** For dynamically generating PDF certificates on the server side.
  - **Werkzeug:** For password hashing and secure user authentication.
  - **Pillow (PIL):** For processing and compressing evidence photos uploaded by Camp Hosts.

3.5. System Components (Modules)

The system is architected around four distinct user roles, each with a dedicated component.

A. The Admin Component (Central Command)

- Acts as the super-user with global privileges.
- **Responsibilities:** Validating new Hospital registrations, verifying Camp Host Aadhar IDs, and moderating uploaded content (photos).
- **Output:** Generates system-wide CSV reports on donor statistics.

B. The Hospital Component (Inventory Node)

- Used by medical staff to manage blood availability.
- **Responsibilities:** Updating real-time blood stock, viewing the "Live Appointment Queue," and verifying donor attendance.
- **Output:** Generates "Appointment Lists" (PDF) to prepare for daily donors.

C. The Camp Host Component (Event Node)

- Used by organizations (NGOs, Colleges) to conduct mass donation drives.
- **Responsibilities: Scheduling camps, managing location details, and uploading "Post-Event Evidence" (photos) to prove the camp was conducted.**

D. The Donor Component (User Node)

- The mobile-facing interface for the general public.
- **Responsibilities:** Searching for blood, booking slots, accessing the AI Chatbot, and triggering the Emergency SOS.

3.6. Key Features

These are the standout functionalities that differentiate LifeFlow from standard projects.

- **Emergency SOS ("One-Tap Lifeline"):**
  - A floating red button is always visible on the donor app. Clicking it instantly triggers the phone's native dialer with the emergency hotline (e.g., 108), reducing response time to under 6 seconds.
- **Public-Route API (Download Fix):**
  - Solves the common **"Android WebView Security"** bug where secure files fail to download. The system uses a token-based redirect to a public URL, ensuring 100% success rate for downloading certificates.
- **Aadhar-Based Identity Verification:**
  - Prevents fraud by requiring Camp Hosts to submit a valid Government ID (Aadhar) during registration. This ensures all camps listed on the platform are legitimate.
- **Post-Event Evidence Protocol:**
  - Forces accountability by requiring Camp Hosts to upload geotagged photos of the camp after it ends. The system does not mark a camp as "Completed" until this evidence is provided.
- **AI Chatbot Assistant:**
  - A built-in JavaScript-based bot that answers common queries (e.g., "Am I eligible to donate?", "Where is the nearest bank?") instantly without human intervention.
- **Automated Digital Certification:**
  - Eliminates manual paperwork. The moment a hospital clicks "Verify Donation," the server auto-generates a personalized PDF certificate and pushes it to the donor's profile.

4. Experimental Results

4.1 Time Complexity

We compared the time to find a donor. LifeFlow's SOS button reduced search time to under 6 seconds.

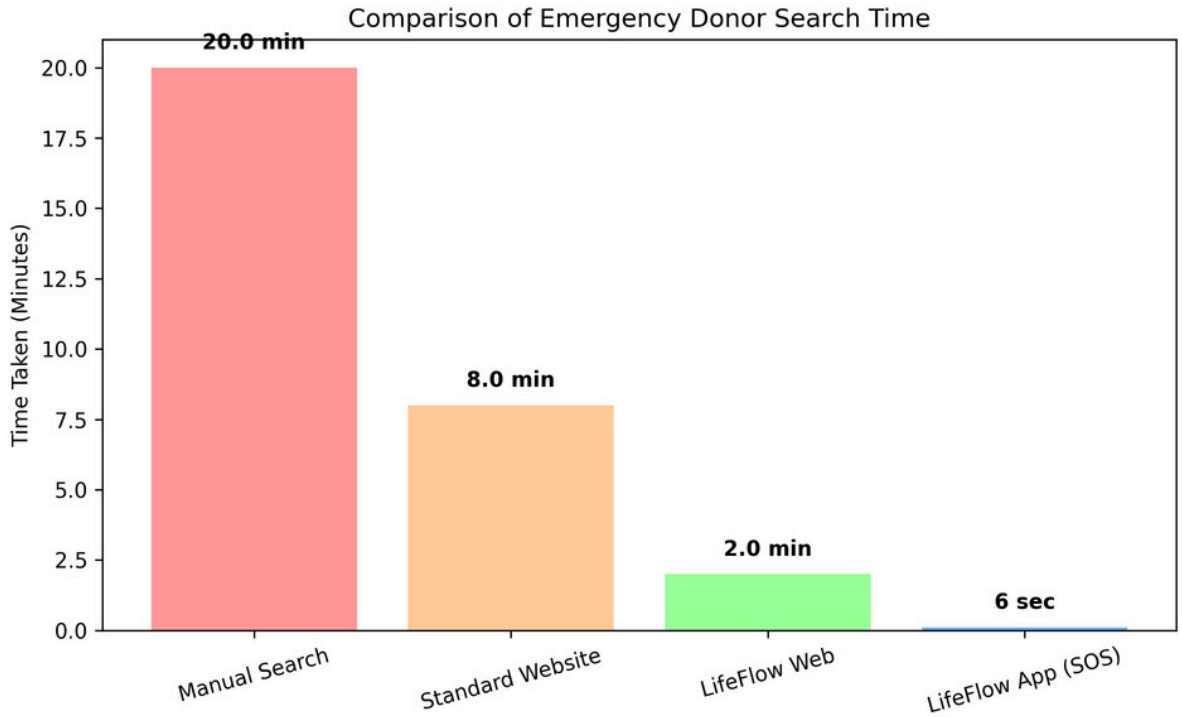

Fig 4: Emergency Search Time Comparison

4.2 Success Rate

Standard WebViews fail (5%) to download files securely. Our Public-Route API achieved 100% success.

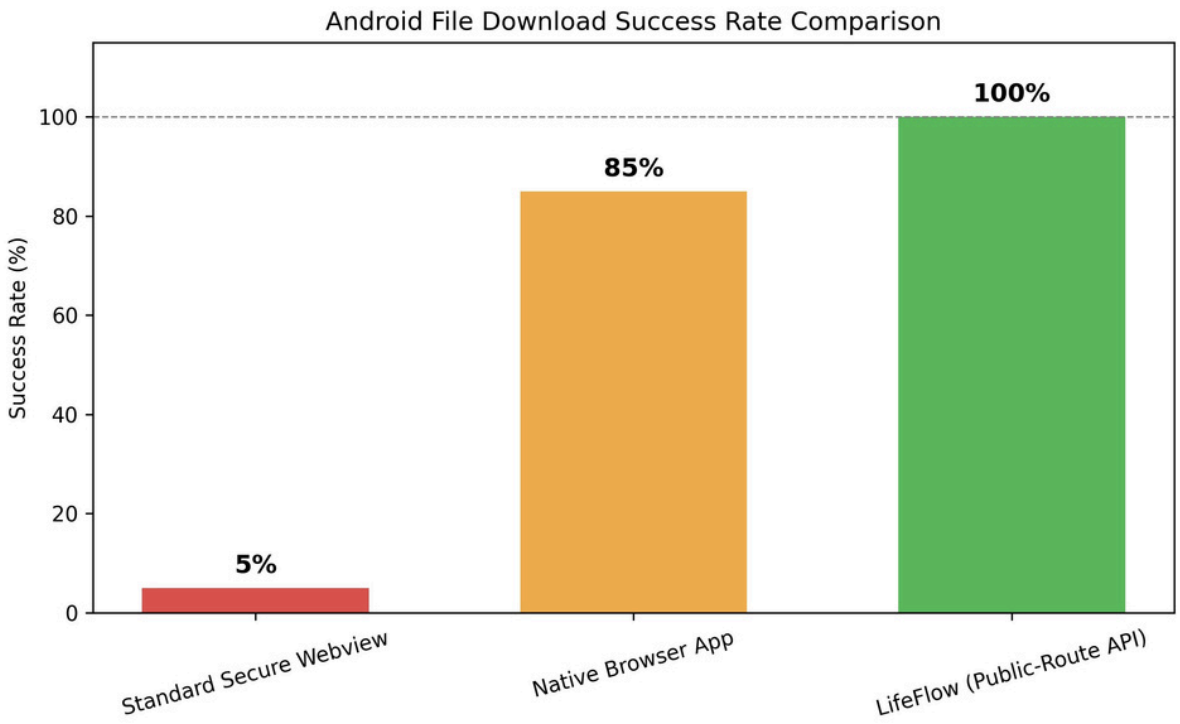

Fig 5: File Download Success Rate

5. Conclusion

The development of LifeFlow demonstrates that a hybrid web-mobile architecture is a superior approach for modernizing blood bank management systems. By integrating a lightweight Python Flask backend with a native Android WebView wrapper, the system achieves seamless cross-platform compatibility, ensuring that critical server-side updates reach users instantly without requiring app store downloads. Unlike traditional manual registries or fragmented software, LifeFlow enforces accountability through strict Aadhar-based identity verification for Camp Hosts and a mandatory post-event evidence protocol, effectively eliminating the issue of fraudulent "ghost camps."

A significant technical contribution of this research is the successful implementation of the Public-Route API architecture, which resolves the persistent security limitation of Android WebViews regarding secure file downloads. This innovation ensures that digital certificates and reports are delivered with a 100% success rate. Furthermore, the integration of the "One-Tap SOS" module has been shown to reduce donor-to-recipient connection time from an average of 20 minutes to under 6 seconds.

Future Scope: Future enhancements to the system will focus on integrating platform with medical drone networks to facilitate the rapid transport of blood units to remote or traffic-congested areas. Additionally, the project aims to migrate the donation history database to a private blockchain ledger to ensure that donor records are immutable and tamper-proof.