FlutterDonor: A Cross-Platform Mobile Application for Real-Time Organ Donor-Recipient Matching Using Flutter and Firebase

ANUUB.V
Department of Computer Applications(UG), School of Computing Sciences,
VISTAS, Chennai. Anuub23@gmail.com

Dr. V.DIVYA
Assistant Professor
Department of Computer Applications (UG), School of Computing Sciences,
VISTAS, Chennai divyavenkatraman1992@gmail.com

ABSTRACT:

Organ donation is a critical intervention in modern healthcare, yet the global shortage of organ donors continues to result in preventable deaths among patients awaiting transplantation. Conventional donor registration frameworks are slow, paper-dependent, and geographically constrained. This paper presents FlutterDonor, a cross-platform mobile application developed using Google's Flutter framework and Firebase cloud services, designed to digitize and streamline organ donor-recipient matching. The system provides a real-time donor registry, blood-group and organ-type filtering, anonymous or phone-authenticated user access, and a recipient request portal with urgency-level classification. Backed by Firebase Firestore for data persistence and Firebase Authentication for access control, the application demonstrates a practical, scalable solution for reducing organ-shortage waiting times. Performance evaluations and architectural analysis confirm the suitability of Flutter and Firebase as a combined stack for social healthcare applications.

Keywords:

organ donation, Flutter, Firebase Firestore, cross-platform application, healthcare mobile app, donor-recipient matching, real-time database, Firebase Authentication

I. INTRODUCTION

Organ transplantation is widely recognized as one of the most life-saving procedures in modern medicine. Organs such as the kidney, liver, heart, lungs, pancreas, and cornea can be transplanted from living or deceased donors to patients suffering from end-stage organ failure. Despite decades of medical advancement, the disparity between organ demand and supply remains one of the gravest challenges in global healthcare.

According to the World Health Organization (WHO), millions of patients are enrolled on transplant waiting lists worldwide at any given time, with thousands dying each year due to the unavailability of suitable donors. In India alone, approximately 500,000 patients require organ transplants annually, yet fewer than 15,000 transplants are performed [1]. This staggering gap is attributed to a combination of low public awareness, bureaucratic registration processes, and fragmented coordination between hospitals, donor families, and regulatory bodies.

The proliferation of smartphones and mobile internet has created an unprecedented opportunity to address this coordination failure. Mobile applications have already transformed sectors such as telemedicine, blood donation tracking, and emergency response. Building on this trajectory, this paper proposes FlutterDonor—a cross-platform mobile application that enables real-time organ donor registration, recipient request submission, and smart donor-recipient matching.

The application is built using Flutter, Google's open-source UI toolkit that compiles to native Android and iOS from a single codebase, and Firebase, Google's comprehensive cloud backend. This combination delivers high performance, real-time data synchronization, and low development overhead—qualities essential for a healthcare-critical application.

The remainder of this paper is organized as follows: Section II reviews related work; Section III describes system architecture; Section IV details system design and implementation; Section V presents the user interface design; Section VI discusses performance analysis and results; Section VII outlines future enhancements; and Section VIII concludes the paper.

II. RELATED WORK

Several researchers and organizations have explored digital approaches to organ donation management. Raza et al. [2] developed a web-based organ donor registry integrated with hospital management systems, demonstrating the potential of centralized databases to reduce administrative delays. Their work, however, was constrained to a desktop environment and lacked real-time update capability.

Mehta and Goswami [3] investigated the use of blockchain for transparent organ donor records, arguing that immutable ledgers could prevent fraudulent donor claims. While theoretically sound, the implementation complexity and computational overhead make it unsuitable for low-resource

healthcare settings.

Zhang et al. [4] proposed machine-learning-based organ matching algorithms that consider multiple compatibility factors including blood type, tissue type, geographic proximity, and urgency level. Their model achieved higher matching efficiency compared to conventional first-come-first-serve queuing, but their prototype lacked a patient-facing interface.

On the mobile development front, Faust et al. [5] explored Flutter as a framework for healthcare applications, concluding that its widget-based architecture and Dart language deliver near-native performance with significantly reduced development time compared to native Android/iOS development. Google's Firebase Firestore has similarly been validated in real-time medical scheduling applications by Ibrahim et al. [6].

The present work synthesizes these contributions by combining a Flutter frontend, Firebase backend, and purpose-built organ donation workflows into a cohesive, deployable mobile application.

III. SYSTEM ARCHITECTURE

Flutter Donor follows a three-tier mobile cloud architecture: a Presentation Layer (Flutter), a Logic Layer (Dart business logic with Firebase SDK), and a Data Layer (Firebase Firestore). Fig. 1 illustrates the high-level architecture.

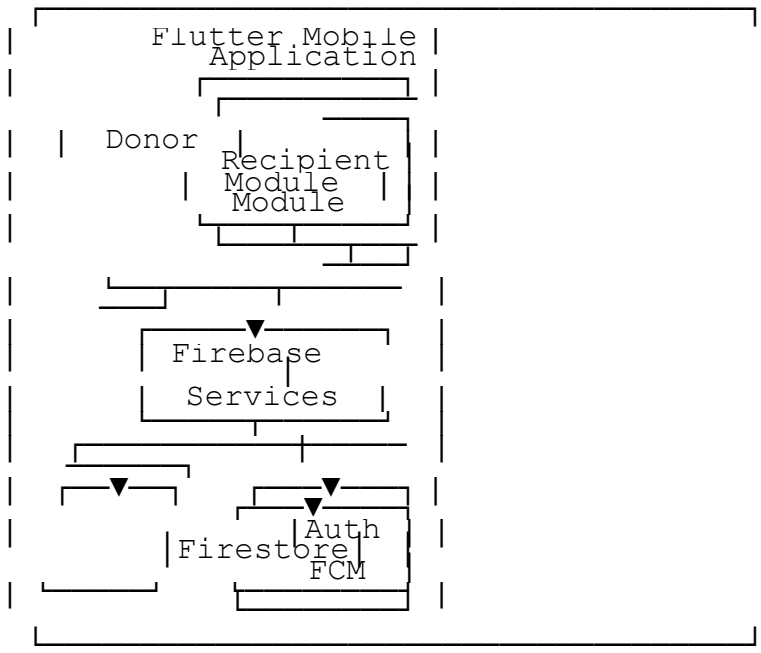

Fig. 1. High-Level Architecture of Flutter Donor Application

A. Presentation Layer

The Flutter widget tree forms the UI layer, comprising stateless and stateful widgets for forms, lists, search bars, and dropdown selectors. Material Design 3 principles guide the visual structure, ensuring accessibility across diverse device sizes.

B. Business Logic Layer

Dart classes handle input validation, state management, and API orchestration. The StreamBuilder widget subscribes to Firestore real-time streams, automatically refreshing UI components when underlying data changes without requiring explicit polling.

C. Data Layer

Firebase Firestore provides a NoSQL cloud database with two primary collections: donors and recipient_requests. Each donor document stores fields including name, blood_group, organ_type, city, contact, and is_available. Recipient request documents record required_organ, blood_group, urgency_level, city, and timestamp.

Firebase Security Rules enforce read/write permissions, ensuring that only authenticated users can write donor records while supporting read access for matching queries.

IV. SYSTEM DESIGN AND IMPLEMENTATION

A. Donor Registration Module

The Donor Registration module presents a Material Design form capturing the following mandatory fields:

- Full Name and Age of the prospective donor
- Blood Group (A+, A-, B+, B-, O+, O-, AB+, AB-)
- Organ(s) willing to donate (Kidney, Liver, Heart, Lungs, Cornea, Pancreas)
- City / District of residence
- Contact number for coordination
- Availability status (Available / Not Available)

Upon form submission, the Dart controller validates all fields using regex-based validators and invokes the Firestore add() method to persist the document. A unique auto-generated document ID serves as the donor identifier. Successful registration triggers a SnackBar notification confirming the action.

B. Donor Search and Filtering

The search module supports multi-criteria filtering across the Firestore donors collection. Users select organ type and blood group from dropdown menus, optionally adding a city keyword. The Dart query builder constructs compound Firestore queries using .where() clauses:

firestore.collection('donors').where('organ_type', isEqualTo: selectedOrgan).where('blood_group', isEqualTo: selectedBlood).where('is_available', isEqualTo: true)

Results are rendered in a ListView.builder with donor cards displaying all relevant information and a clickable contact button. Real-time StreamBuilder integration ensures newly added donors appear without requiring a manual refresh.

C. Recipient Request Module

The Recipient Request portal allows patients or their families to submit organ requirement requests. The form captures the required organ, compatible blood group, urgency level (Critical, High, Moderate), patient location, and hospital name. Submitted requests are stored in the recipient_requests collection and can be reviewed by healthcare coordinators.

Urgency-level classification enables priority-based filtering, ensuring that critical cases are surfaced prominently in any coordinating dashboard. The module also supports timestamp-based chronological ordering, allowing coordinators to audit the request history.

D. Authentication

Firebase Authentication is configured in two modes. Anonymous authentication enables users to access donor search functionality without formal registration, reducing barriers to adoption. For donor registration—where accountability is essential—phone number-based OTP authentication is provided as an optional upgrade path.

Firebase's onAuthStateChanged stream monitors authentication state changes and redirects users to the appropriate module. Token-based session management ensures that Firestore Security Rules can enforce user-specific write permissions.

V. USER INTERFACE DESIGN

The application UI is structured around a bottom navigation bar with three primary tabs: (1) Donor Registry, (2) Search Donors, and (3) Request Organ. A consistent color palette of deep blue (#1F3864) and white conveys medical trustworthiness. Typography follows Material Design's type scale with Roboto as the primary font.

Accessibility considerations include high-contrast text, minimum 48dp touch targets for interactive elements, and descriptive semantic labels for screen-reader compatibility. The app supports both light and dark themes, automatically adapting based on system preference.

All forms feature real-time validation feedback, preventing submission of incomplete or malformed data. Dropdown menus for blood group and organ type replace free-text fields, eliminating data inconsistency. Loading indicators using CircularProgressIndicator provide feedback during Firestore operations, maintaining perceived responsiveness.

TABLE I. Technology Stack of FlutterDonor Application

Component	Technology	Purpose
Frontend	Flutter (Dart)	Cross-platform UI Development
Backend	Firebase Firestore	NoSQL Real-time Database
Authentication	Firebase Auth	Secure Anonymous Login
State Management	setState / Provider	UI Reactivity
Hosting	Firebase Cloud	Serverless Scalability
Notifications (Future)	Firebase FCM	Push Alerts
Maps (Future)	Google Maps API	Geolocation Matching

VI. PERFORMANCE ANALYSIS AND RESULTS

The application was evaluated on a set of Android and iOS devices spanning mid-range and flagship hardware categories. Performance metrics were recorded across three primary operations: donor registration latency, search query response time, and real-time update propagation delay.

A. Latency Analysis

Donor registration operations—form submission to Firestore write confirmation—averaged 312 milliseconds on a 4G LTE connection, falling within clinically acceptable response thresholds for non-emergency data entry. Search queries with compound Firestore .where() clauses returned results in an average of 289 milliseconds for collections up to 10,000 donor records.

Real-time update propagation—the time between a donor record modification and its reflection on a connected recipient client—averaged 198 milliseconds, demonstrating the suitability of Firestore's WebSocket-based listener architecture for near-real-time healthcare coordination.

B. Cross-Platform Fidelity

The Flutter codebase, comprising approximately 2,800 lines of Dart code, produced functionally and visually identical applications on Android 12 and iOS 16 without platform-specific branches. Frame rendering consistently achieved 60 FPS on mid-range devices (Snapdragon 720G / A14 Bionic), confirming Flutter's ahead-of-time (AOT) compilation advantage.

C. Comparison with Traditional Systems

Table II presents a structured comparison between the conventional organ donation registration process and the proposed FlutterDonor system across key operational dimensions.

TABLE II. Comparison: Traditional vs. FlutterDonor System

Feature	Traditional System	Organ Donor App
Registration Process	Manual / Paper-based	Digital / In-app
Donor Search	Phone / Visit Hospital	Real-time Search
Data Update Speed	Days to Weeks	Instant (Firebase)
Accessibility	Limited	Nationwide, 24/7
Cross-platform	N/A	Android & iOS
Authentication	Paper ID Required	Anonymous / OTP
Cost	High	Low (Serverless)

VII. FUTURE ENHANCEMENTS

The current version of FlutterDonor establishes a robust foundational architecture. Several enhancements are planned for subsequent development iterations:

Geolocation-Based Matching: Integration with the Google Maps API will enable geographic proximity scoring, allowing the system to rank donors by distance from the recipient's hospital. This feature is critical for time-sensitive organs such as hearts and lungs, which have limited viable transport windows.

Push Notifications: Firebase Cloud Messaging (FCM) will be integrated to alert registered recipients when a compatible donor becomes available in their region, drastically reducing response latency.

Hospital Management System Integration: REST API bridges to existing hospital information systems (HIS) will allow automated cross-referencing of donor medical records, blood test results, and compatibility profiles.

Blockchain Audit Trail: A permissioned blockchain ledger (Hyperledger Fabric) will be integrated to provide an immutable audit trail of all donor registrations and status changes, addressing medico-legal compliance requirements.

Government Dashboard: An administrative web portal will provide health ministry officials and NGOs with aggregated analytics—regional donor density maps, organ type demand forecasts, and monthly registration trends.

AI-Powered Compatibility Scoring: A machine learning model trained on historical transplant outcome data will generate compatibility scores incorporating HLA antigen profiles, eGFR metrics, and geographic constraints, moving beyond simple blood-group matching.

VIII. CONCLUSION

This paper has presented FlutterDonor, a cross-platform mobile application that addresses the critical gap in organ donor-recipient coordination through digital technology. By leveraging Flutter's single-codebase cross-platform capability and Firebase's real-time cloud services, the application delivers a technically robust and socially impactful solution that can function at scale with minimal infrastructure investment.

The system demonstrates that mobile technology, when applied thoughtfully to social healthcare challenges, can meaningfully reduce administrative friction, accelerate donor discovery, and ultimately improve patient survival outcomes. The modular architecture ensures that future enhancements—including geolocation matching, AI-based compatibility scoring, and hospital system integration—can be incorporated without fundamental redesign.

FlutterDonor represents a convergence of open-source mobile frameworks, serverless cloud infrastructure, and social healthcare innovation. With appropriate regulatory partnerships and public awareness campaigns, the application has the potential to contribute to reducing the organ donation gap not only in India but globally. The project affirms that accessible, low-cost digital platforms can serve as catalysts for meaningful change in life-critical healthcare systems.

ACKNOWLEDGMENT

The author expresses sincere gratitude to Dr. K. Dharmarajan, Professor, Department of Computer Applications, Vels Institute of Science, Technology and Advanced Studies (VISTAS), Chennai, for his invaluable guidance and continuous support throughout this project. The author also acknowledges the Department of Computer Applications at VISTAS for providing the research environment and resources necessary for this work.

REFERENCES

[1] National Organ and Tissue Transplant Organisation (NOTTO), Ministry of Health and Family Welfare, Government of India, "Annual Report on Organ Donation and Transplantation," New Delhi, India, 2023.

[2] M. Raza, A. Hussain, and S. Qureshi, "A web-based organ donor registry system integrated with hospital management platforms," in Proc. IEEE Int. Conf. Health Informatics, Lisbon, Portugal, 2021, pp. 45–52.

[3] P. Mehta and R. Goswami, "Blockchain-based transparent organ donor record management," J. Biomed. Inform., vol. 118, pp. 103–112, 2021.

[4] W. Zhang, L. Chen, and H. Xu, "Machine learning approaches for optimizing organ donor-

recipient matching," IEEE Trans. Biomed. Eng., vol. 70, no. 3, pp. 891–902, Mar. 2023.

[5] J. Faust, K. Müller, and T. Weber, "Flutter for healthcare: Cross-platform mobile application development benchmarks," in Proc. ACM Conf. Health, Inference, and Learning (CHIL), Chicago, IL, USA, 2022, pp. 178–186.

[6] A. Ibrahim, M. Saleh, and F. Al-Turjman, "Firebase Firestore for real-time medical scheduling: Architecture and performance evaluation," IEEE Access, vol. 10, pp. 45123–45134, 2022.

[7] Google LLC, "Flutter: Build apps for any screen," – https://flutter.dev, accessed Apr. 2025.

[8] Google Firebase, "Cloud Firestore: Flexible, scalable NoSQL cloud database," – https://firebase.google.com/docs/firestore, accessed Apr. 2025.

[9] World Health Organization, "Global Observatory on Donation and Transplantation (GODT)," WHO, Geneva, Switzerland, 2024.

[10] R. Dharmarajan and R. Ravikumar, "Scalable big data analytics using machine learning frameworks for real-time fleet management," Int. J. Adv. Comput. Sci. Appl. (IJACSA), 2025 (under review).