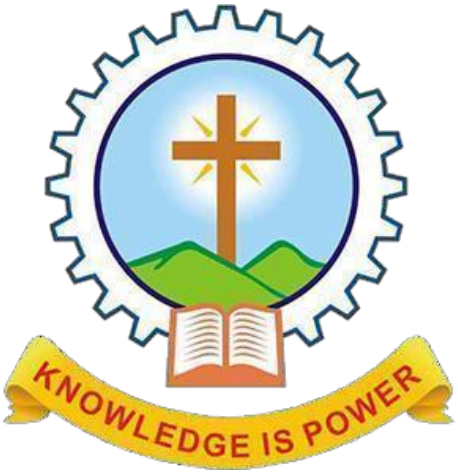
MAR ATHANASIUS COLLEGE OF ENGINEERING

(Affiliated to APJ Abdul Kalam Technological University, TVM)

KOTHAMANGALAM



Department of Computer Applications

Main Project Report

Organ Bank

Done by

DONA JINCE

Reg No: MAC23MCA-2024

Under the guidance of

Prof. Nisha Markose

2023-2025

MAR ATHANASIUS COLLEGE OF ENGINEERING

(Affiliated to APJ Abdul Kalam Technological University, TVM)

KOTHAMANGALAM

CERTIFICATE

Organ Bank

Certified that this is the bonafide record of project work done by

DONA JINCE

Reg No: MAC23MCA-2024

During the academic year 2024-2025, in partial fulfillment of requirements for award of the degree,

Master of Computer Applications

of

APJ Abdul Kalam Technological University, Thiruvananthapuram

Faculty Guide
Prof. Nisha Markose

Head of the Department
Prof. Biju Skaria

Project Coordinator
Prof. Sonia Abraham

External Examiner

ACKNOWLEDGEMENT

With heartfelt gratitude, I extend my deepest thanks to the almighty for His unwavering grace and blessings that have made this journey possible. May His guidance continue to illuminate my path in the years ahead.

I am immensely thankful to Prof. Biju Skaria, Head of the Department of Computer Applications, Prof. Sonia Abraham, our dedicated Project Coordinator and Prof. Nisha Markose my main Project Guide, for their invaluable guidance and timely advice, which played a pivotal role in shaping this project. Their guidance, constant supervision, and provision of essential information were instrumental in the successful completion of the main project.

I extend my profound thanks to all the professors in the department and the entire staff at MACE for their unwavering support and inspiration throughout my academic journey. My sincere appreciation goes to my beloved parents, whose guidance has been a beacon in every step of my path.

I am also grateful to my friends and individuals who generously shared their expertise and assistance, contributing significantly to the fulfillment of this endeavor.

Dona Jince

ABSTRACT

The Organ Bank is a comprehensive digital solution designed to streamline and automate the process of organ donation, matching, and transplantation. Traditional organ donation systems rely heavily on manual processes, resulting in inefficiencies, delays, and missed life-saving opportunities. While some organizations maintain static websites for organ donor registration, these systems lack real-time functionality for managing donor-recipient data, organ matching, and hospital coordination. The Organ Bank addresses these limitations by providing an integrated platform that ensures reliability, transparency, and efficiency in organ transplantation management.

This project introduces a real-time, automated matching system that enhances the efficiency of organ allocation. The platform supports all key stakeholders, including administrators, donors, recipients, and hospitals, facilitating seamless coordination. Core features include automated organ matching based on medical compatibility, real-time notifications for transplant schedules and compatibility testing, and a feedback mechanism to ensure continuous improvement and accountability.

The system is built using Flutter for a responsive and intuitive user interface, ensuring seamless experiences across mobile and web platforms. The backend is powered by Node.js, which efficiently handles server-side logic and matching algorithms, while MongoDB provides secure, scalable, and real-time data storage and management. The development environment integrates Visual Studio Code, enabling smooth application development.

By utilizing these modern technologies, the Organ Bank enhances operational efficiency, reduces manual errors, and accelerates the decision-making process. Its real-time features and improved communication foster trust among stakeholders, ultimately increasing the success rate of organ transplants. This innovative approach not only optimizes the organ donation workflow but also contributes to the noble cause of saving lives, making it an essential tool in modern healthcare management.

List of Tables

List of Figures

Contents

1 INTRODUCTION

The Organ Bank is a comprehensive digital solution developed as a mobile application to automate and streamline the process of organ donation, matching, and transplantation. This project eliminates the inefficiencies of traditional manual systems by providing an integrated, real-time platform that ensures reliability, transparency, and efficiency in managing organ donations and transplants.

The existing system for organ donation and transplantation often involves manual processes, leading to delays, inefficiencies, and missed opportunities for life-saving procedures. While some organizations maintain static websites for registration or informational purposes, these systems fail to provide dynamic solutions for managing donor and recipient data, organ matching, and hospital coordination. Such limitations result in underutilized resources, lack of transparency, and delayed decision-making, which can cost lives.

The Organ Bank addresses these shortcomings by introducing a dynamic and real-time platform designed to meet the needs of all stakeholders, including administrators, donors, recipients, and hospitals. Key features include automated organ matching based on medical criteria, real-time notifications for compatibility testing schedules, transplant dates, and a feedback mechanism to ensure continuous improvement and accountability.

The system uses Flutter for developing a responsive and intuitive user interface across both mobile and web platforms, ensuring seamless user experiences. Node.js powers the backend, efficiently handling server-side logic, organ matching algorithms, and business processes, while MongoDB serves as the database for secure, scalable, and real-time data storage and management. The development environment integrates Visual Studio Code for the mobile app.

By utilizing these modern technologies, the Organ Bank provides a robust and efficient solution that automates processes, reduces manual errors, and accelerates decision-making. Its real-time features and enhanced communication build trust among stakeholders while significantly improving the success rate of organ transplants. This innovative solution not only addresses critical challenges in organ donation and transplantation but also contributes significantly to the noble cause of saving lives, making it a vital tool for modern healthcare management.

2 SUPPORTING LITERATURE

2.1 Literature Review

Paper 1: Hawashin, D., Jayaraman, R., Salah, K., Yaqoob, I., Simsekler, M. C. E., & Ellahham, S. "Blockchain-Based Management for Organ Donation and Transplantation," IEEE Access, vol. 10, pp. 59013-59024, 2022. DOI: 10.1109/ACCESS.2022.3180008

The paper "Blockchain-Based Management for Organ Donation and Transplantation" by Diana Hawashin et al. explores technological advancements to enhance organ donation and transplantation processes. Traditional organ donation systems face challenges such as inefficiencies, lack of transparency, and security risks due to manual processing and centralized data storage. The study proposes automated donor-recipient matching, real-time organ tracking, and system-generated notifications to streamline the organ donation workflow and reduce administrative delays.

The study introduces six key algorithms that automate various phases of organ transplantation, including donor registration, organ removal, transportation, and recipient matching. The system implements ranked matching based on blood type, BMI, and age compatibility, ensuring fair and efficient organ allocation. Additionally, it emphasizes real-time scheduling and updates, ensuring that all stakeholders, including hospitals, doctors, and patients, receive timely notifications regarding compatibility tests, transplant schedules, and result updates.

The findings of this study align with the Organ Bank project, as both focus on optimizing organ donation through technology. The project, built using Flutter, Node.js, and MongoDB, aims to automate donor-recipient matching, ensure real-time data synchronization, and improve communication among hospitals and recipients. By integrating features such as automated scheduling, system-generated notifications, and optimized organ allocation algorithms, the Organ Bank project ensures efficient organ tracking, minimal delays, and enhanced coordination. This research provides a strong foundation for enhancing transparency, efficiency, and scalability in organ donation management.

Paper 2: Dajim, L. A., Al-Zuraib, A. A., Al-Farras, S. A., & Al-Shahrani, B. S. "Organ Donation Decentralized Application Using Blockchain Technology," 2019 IEEE Conference, 2019.

The paper "Organ Donation Decentralized Application Using Blockchain Technology" by Lama Abdulwahab Dajim et al. presents a technology-driven approach to improving organ donation and transplantation systems. Traditional organ donation processes face challenges such as long waiting times, lack of transparency, and inefficiencies in centralized systems. The proposed web-based system ensures that donor and recipient information, including medical ID, blood type, and organ type, is securely recorded and managed. The system operates on a first-in, first-out basis, prioritizing critical cases when necessary, thus enhancing fairness and efficiency in organ allocation.

The study highlights the advantages of automated organ matching, real-time data updates, and secure donor-recipient management. The system ensures data integrity, secure handling of organ availability records, and automated recipient matching based on blood type, medical urgency, and organ compatibility. Additionally, it introduces real-time scheduling for compatibility tests, transplant procedures, and notification updates to keep all stakeholders informed. The comparative analysis with traditional organ transplant management systems shows that automated donor-recipient matching and real-time communication improve efficiency and reduce delays in organ allocation.

This research is relevant to the Organ Bank project, as both systems aim to enhance organ donation management through technology. The project, built using Flutter, Node.js, and MongoDB, focuses on automating donor-recipient matching, ensuring real-time data synchronization, and improving coordination among hospitals and recipients. By implementing automated scheduling, instant notifications, and optimized allocation processes, the Organ Bank project ensures efficient organ tracking, minimal delays, and enhanced coordination. This research supports the adoption of automated processes to streamline organ donation, improve stakeholder communication, and enhance the overall efficiency of the transplantation workflow.

Paper 3: Behera, S. K., Sethy, P. K., & others. ”MongoDB Integration with Python and Node.js, Express.js,” 2024 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), 2024. DOI: 10.1109/ICAECT60202.2024.104

The paper ”MongoDB Integration with Python and Node.js, Express.js” by Santi Kumari Behera et al. presents a detailed study on the integration of MongoDB, a popular NoSQL database, with Python and Node.js for efficient data management. MongoDB's scalability, flexibility, and document-based architecture make it well-suited for handling semi-structured and unstructured data in modern web applications. The authors analyze key drivers and APIs, particularly PyMongo for Python and Mongoose for Node.js, highlighting their role in facilitating seamless database interactions. Additionally, the study discusses MongoDB's integration with the MEAN stack (MongoDB, Express.js, Angular, and Node.js), demonstrating its effectiveness in building robust web applications.

A significant focus of the study is query optimization and indexing, where B-trees and hash tables are utilized to enhance data retrieval performance. The authors emphasize the benefits of aggregation pipelines and MapReduce, which contribute to efficient data processing and real-time analytics. Security concerns such as unauthorized access and injection attacks are also addressed, with recommendations for implementing authentication, encryption, and role-based access control to strengthen database security and integrity.

The findings of this study are highly relevant to the Organ Bank project, as MongoDB's scalability and efficiency align with the system's need to handle large volumes of donor-recipient data, real-time organ matching, and hospital coordination. The use of Mongoose for Node.js and PyMongo for Python can enhance database operations, improve query execution speed, and ensure secure data management. By incorporating automated organ matching, real-time scheduling, and optimized security measures, the Organ Bank project can enhance performance, maintain data accuracy, and streamline organ transplantation workflows

2.2 Literature Review Summary

The paper ”Blockchain-Based Management for Organ Donation and Transplantation” by Diana Hawashin et al. presents a solution to enhance the efficiency and security of organ donation systems. Traditional systems suffer from manual processes, data manipulation risks, and lack of real-time tracking. The study proposes a structured approach to donor-recipient matching, organ removal, transportation, and transplantation, aiming to ensure secure and transparent organ allocation. The authors highlight the importance of automating key steps in the organ donation process to minimize delays and improve trust between stakeholders.

Similarly, the study ”Organ Donation Decentralized Application Using Blockchain Technology” by Lama Abdulwahab Dajim et al. supports the use of a digitized, automated platform for organ donation management. This research emphasizes the importance of data integrity, privacy, and transparency in organ transplantation. The proposed system ensures that patient records, organ availability, and recipient priority lists are managed efficiently and fairly. The study highlights that standardized organ allocation criteria and real-time communication between hospitals and medical teams play a crucial role in improving the organ donation process. Both studies advocate for automated donor-recipient matching systems to eliminate inefficiencies and enhance decision-making.

Efficient data management is crucial in organ donation systems, where large volumes of donor-recipient information require real-time access and updates. The paper ”MongoDB Integration with Python and Node.js, Express.js” by Santi Kumari Behera et al. explores the advantages of using MongoDB, a widely adopted NoSQL database, for handling semi-structured and unstructured data in modern web applications. MongoDB’s scalability, flexibility, and document-oriented structure make it an ideal choice for organ donation platforms, where dynamic data storage and rapid retrieval are essential.

The study examines PyMongo for Python and Mongoose for Node.js, both of which facilitate seamless integration of MongoDB with web applications. The research also explores indexing mechanisms like B-trees and hash tables, which enhance query performance by reducing retrieval time. Security vulnerabilities such as injection attacks, unauthorized access, and data leakage are also analyzed, with recommended best practices including encryption, authentication, and role-based access control. These findings are highly relevant to organ donation systems, which require secure, efficient, and scalable data management solutions.

The Organ Bank project aligns with these studies by leveraging automation, real-time data synchronization, and secure database management to streamline donor-recipient matching and improve transparency in the organ transplantation process. By incorporating efficient scheduling, system-generated notifications, and optimized data retrieval, the system ensures accuracy, efficiency, and reliability in organ donation management.

2.3 Findings and Proposals

The reviewed studies highlight the need for real-time updates and automated processes in organ donation management. A key proposal for the Organ Bank project is the implementation of real-time notifications for scheduling compatibility tests, transplant procedures, and result updates. This ensures that hospitals, donors, and recipients receive timely alerts, reducing delays in organ transplantation

Another crucial recommendation is automated donor-recipient matching based on blood type and organ compatibility. Every time a donation or request is submitted, the system should instantly analyze the available donor-recipient data and provide the best possible matches. The use of MongoDB indexing and querying mechanisms will allow for fast and efficient retrieval of donor-recipient pairs, ensuring a seamless and automated matching process.

Furthermore, email communication should be integrated into the system to facilitate coordination among stakeholders. Whenever an organ match is found, the system should automatically send email notifications to the recipient, donor, and medical personnel involved, ensuring efficient communication and scheduling. These proposals align with modern digital automation trends and will significantly enhance the efficiency, security, and transparency of the Organ Bank system.

In conclusion, automated donor-recipient matching ensures efficient and fair organ allocation, while MongoDB integration provides scalability, high-speed data access, and security. These technologies can significantly improve data integrity, transparency, and real-time decision-making in organ donation management. By incorporating these advanced solutions, the Organ Bank project can establish a robust, automated, and life-saving platform for efficient organ donation and transplantation.

3 SYSTEM ANALYSIS

3.1 Module Description

User Management

- Donors, recipients, and Hospitals can register themselves by providing necessary details such as personal information, contact details, and medical records (for donors and recipients).
- Hospitals provide their registration details, including location and address.
- Admin reviews the registration details.
- Approves or rejects registrations based on validity and authenticity of the provided information.
- Sends approval/rejection notifications to the respective users via mail.
- Donors and recipients can update their profiles after approval.
- Hospitals can edit their details, such as contact information and services provided.

Donation Module

- Donors can register and select the organ they wish to donate.
- Donors can update the availability status at any time before the match is approved by admin.
- Donors can view the status of the organ donated.

Request Module

- recipients can register and select the organ they wish to request.
- recipients can view the status of the organ requested.

Organ Matching Module

- Automatically matches registered donors and recipients based on compatibility factors such as blood group and organ type when a donation or request is submitted.
- Successful matches wait for admin approval.
- Admin approves the matches, and notifications are sent to the respective hospital.

Compatibility Testing Module

- Hospitals schedule onsite compatibility tests for approved matches.
- Updates the system with test results (success/failure).
- Notifications are sent to the donor and recipient.

Transplantation Management Module

- Hospitals conduct transplantation procedures after successful compatibility tests.
- Updates the system with transplantation outcomes (success or failure).
- Notifications are sent to Admin, Donors, and recipients about the procedure status.

Feedback and Complaint Management Module

- Donors, Recipients, and Hospitals can submit feedback on their experiences, system usability, and hospital services.
- Feedback is visible to the Admin for system improvement.
- Donors and recipients can raise complaints about hospital services, compatibility testing, or delays in processes.
- Hospitals can report any issues related to the system or donor/recipient cooperation.

Report Management Module

- Admin generates reports on day to day transplantation outcomes and details.

3.2 Business Rules

- Registrations for Donors, Recipients, and Hospitals must be reviewed and approved by the Admin before granting access to the system.
- Donation Consent can be revoked at any time before a match is approved by the Admin.
- Priority will be given based on medical urgency, compatibility, and waiting time.
- Matches must be reviewed and approved by the Admin before proceeding to compatibility tests.
- Only verified hospitals can perform compatibility tests and transplantation procedures.
- Admin must review all registrations (Donors, Recipients, Hospitals) within 24–48 hours of submission.
- Complaints must be addressed by the Admin.

3.3 Feasibility Analysis

3.3.1 Technical Feasibility

The Organ Bank system is technically feasible due to the use of a robust and widely adopted technology stack that ensures efficiency, scalability, and seamless integration. The frontend is built using Flutter, a cross-platform framework that allows the application to run smoothly on Android, iOS, and web platforms. Dart, the programming language used in Flutter, provides high-performance animations, a responsive UI, and efficient application execution, ensuring an intuitive and seamless user experience for donors, recipients, hospitals, and administrators.

The backend is powered by Node.js, a fast, scalable, and event-driven environment that efficiently handles server-side logic, donor-recipient matching algorithms, notifications, and authentication. The system employs RESTful APIs to ensure seamless communication between the frontend and the database, allowing real-time data exchange and updates. A key technical feature of the Organ Bank is its real-time scheduling system, which provides instant updates on compatibility tests, transplant scheduling, and result notifications, ensuring timely coordination among all stakeholders.

The database is managed using MongoDB, a highly scalable and flexible NoSQL database that efficiently handles semi-structured and unstructured data such as donor-recipient profiles, organ availability, and transaction logs. MongoDB's schema-less design allows for easy scalability and modification of records as new data is added. Indexing and querying mechanisms further enhance performance by enabling fast and efficient retrieval of donor-recipient matches. Every time a donation or request is submitted, the system automatically updates the matching records, ensuring an accurate and reliable process.

Additionally, the system incorporates email-based communication to keep stakeholders informed about critical updates. When a compatible donor-recipient match is found, an automated email notification is sent to relevant parties, including hospitals, doctors, and patients, ensuring prompt coordination and decision-making. The combination of Flutter, Node.js, and MongoDB provides a modern, efficient, and scalable solution that ensures reliability, security, and accessibility, making the Organ Bank a technically feasible system that can be successfully implemented and deployed.

3.3.2 Economical Feasibility

The Organ Bank project is economically feasible as it utilizes cost-effective open-source technologies, significantly reducing development and operational costs. The system is built using Flutter for frontend development, Node.js for backend logic, and MongoDB for database management, all of which are open-source frameworks that eliminate the need for expensive software licenses. This approach reduces initial setup costs and provides a financially viable

solution for hospitals and organizations looking to implement a modern organ donation system.

The availability of skilled developers further enhances the project's economic feasibility. Flutter, Node.js, and MongoDB are widely adopted technologies, making it easy to find experienced developers at competitive rates. This lowers hiring costs and allows for faster development and deployment, reducing the overall financial burden of building and maintaining the system.

Another key factor contributing to the system's cost-effectiveness is automation. The automated organ-matching algorithm eliminates the need for manual data processing, reducing the workload of hospital staff and minimizing administrative expenses. The system also includes automated real-time scheduling for compatibility testing, transplant coordination, and result notifications, ensuring that resources are used optimally. This efficiency translates into reduced labor costs, making the system an economically sound investment.

Although initial development, deployment, and maintenance costs are involved, the long-term benefits of the Organ Bank system justify the investment. By optimizing workflows, reducing errors, and improving decision-making, the system leads to cost savings in the healthcare sector. The automation of critical processes also increases the number of successful transplants, ultimately making the system a financially sustainable solution that improves efficiency, resource utilization, and patient outcomes.

3.3.3 Operational Feasibility

The Organ Bank system is highly feasible from an operational standpoint, as it streamlines the organ donation and transplantation process while enhancing efficiency, transparency, and trust among all stakeholders. The system's Flutter-based frontend ensures a seamless, user-friendly experience for donors, recipients, hospitals, and administrators, making it accessible across multiple platforms. The intuitive UI enables users to register, track organ availability, and receive real-time updates with minimal effort, ensuring smooth operations.

The backend, powered by Node.js and MongoDB, efficiently manages user authentication, organ matching, and transaction logs, ensuring real-time data access and high system availability. One of the most crucial operational components is the automated donor-recipient matching system, which eliminates delays in organ allocation. Whenever a new donation or request is submitted, the system instantly analyzes donor and recipient profiles, using blood type and organ compatibility as key matching criteria. This automation significantly improves matching accuracy and speed, increasing the likelihood of successful transplants. Another vital feature is real-time scheduling and updates. The system automates the compatibility testing process, notifying relevant stakeholders about test schedules, transplant procedures, and final results. These updates are delivered via automated email notifications, ensuring that all involved parties—including hospitals, doctors, and patients—are informed in a timely manner. The use of email-based communication enhances operational coordination, reducing miscommunication

and delays.

The Organ Bank system also enhances trust and transparency through its notification and feedback mechanisms. By keeping users informed at every stage of the process, the system ensures that donors, recipients, and medical staff remain engaged and aware of critical updates. Additionally, the automation of data logging and tracking minimizes human error and ensures compliance with medical regulations. The system's ability to optimize workflows, minimize manual interventions, and improve coordination makes it an operationally viable solution that can be effectively implemented in real-world healthcare settings. With continuous updates, dedicated technical support, and strong security measures, the Organ Bank system is positioned for long-term success and scalability.

3.4 System Environment

3.4.1 Software Environment

- **Frontend:** Flutter (Dart Language) for cross-platform mobile and web application development.
- **Backend:** Node.js for efficient server-side logic and API handling.
- **Database:** MongoDB for scalable, NoSQL-based data management.
- **Server OS:** Windows for high performance, reliability, and cost-effectiveness.
- **Development Tools:**
  - Android Studio for Flutter development.
  - VS Code for backend development.
  - Postman for API testing.
- **APIs and Frameworks:** RESTful APIs for frontend-backend communication.

3.4.2 Hardware Environment

- **Processor:**Intel i5 13th Gen 13500H
- **RAM:** 16 GB RAM
- **Storage:** 512 GB SSD for fast data retrieval
- **Internet Connectivity:** High-speed internet for real-time operations

3.5 Actors and roles

- Admin
  - Manages donor, recipient, and hospital registrations.
  - Monitors the system for new registrations and approves/rejects them.
  - Oversees the automated organ matching system and reviews potential matches.
  - Approves organ matches and adds them to the matched organs table.
  - Notifies donors, recipients, and hospitals about approved matches.
  - Tracks organ availability and generates detailed reports.
  - Manages system data and ensures proper functionality.
  - Monitors transplantation results and oversees feedback submissions.
- Donor
  - Registers with the system, providing consent for organ donation.
  - Updates personal details and organ donation preferences.
  - Receives notifications from the admin and hospital regarding matches, compatibility tests, and transplantation schedules.
  - Provides feedback about the system and hospital services.
- **recipient**
  - Registers with the system, requesting organ transplants.
  - Updates personal details and organ requirements.
  - Receives notifications about organ matches, compatibility tests, and transplantation schedules.
  - Tracks organ match statuses and compatibility test results.
  - Provides feedback about the system and hospital services.
- **Hospital**
  - Registers with the system and manages hospital-related data.
  - Conducts compatibility tests for matched donors and recipients.
  - Schedules and carries out transplantation procedures.
  - Notifies donors and recipients about test and transplantation schedules.
  - Updates the system with test results and transplantation outcomes.
  - Provides feedback on the organ bank system.

4 SYSTEM DESIGN

4.1 Use case Diagram

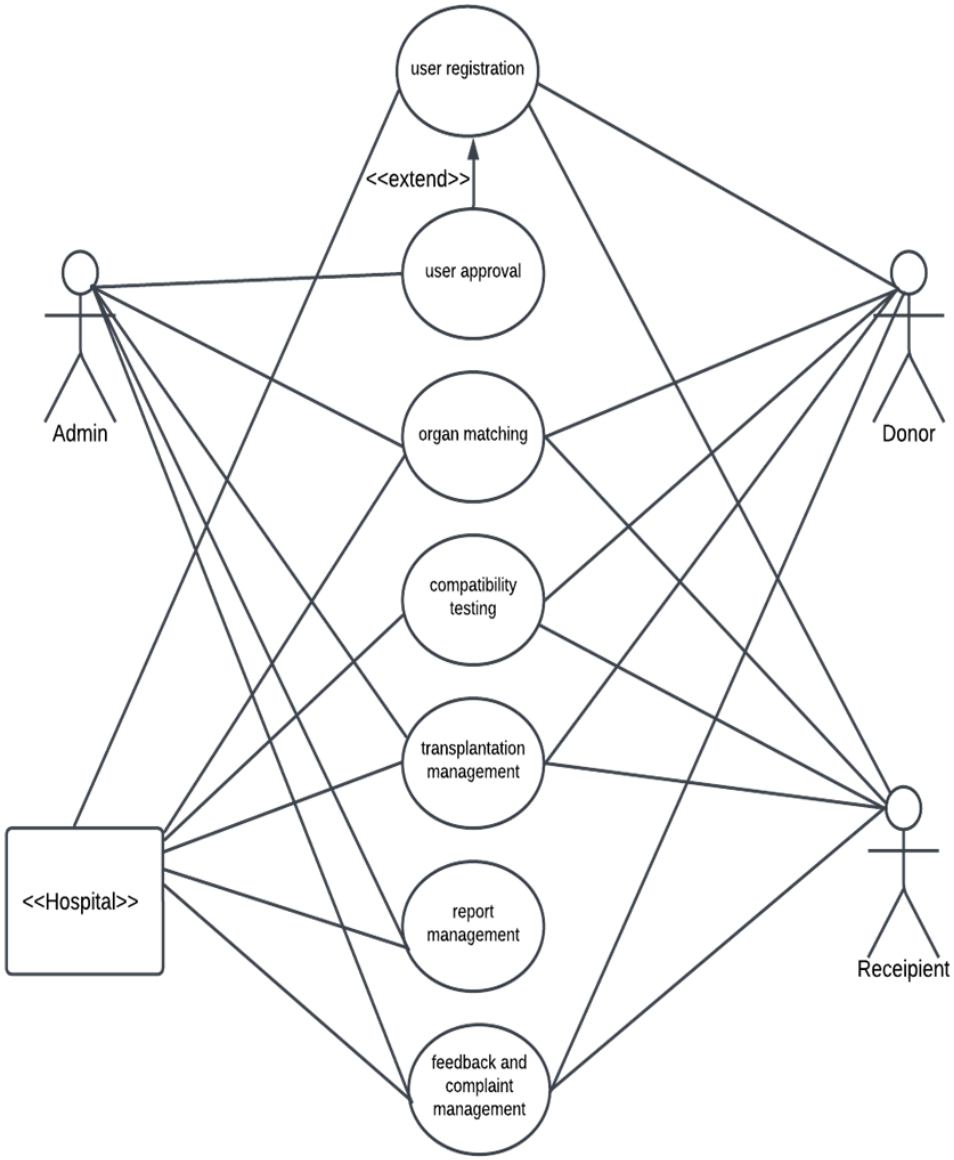


Figure 4.1: Use Case Model

Figure 4.1 shows the Use Case Diagram for the Organ Bank System. It represents how different users interact with the system's functionalities. Below is a description of the diagram:

Actors

- **Donors** – Individuals willing to donate organs.
- **Recipients** – Patients in need of an organ.
- **Admin** – System managers who oversee organ allocation and approvals.
- **Hospitals/Medical Staff** – Responsible for organ transplantation and compatibility testing.

Use Cases

- **User Registration** – Allows users (donors, recipients, and hospitals) to register on the platform.
- **User Approval** – Admin or hospital staff verify and approve registered users.
- **Organ Matching** – The system matches donors and recipients based on medical compatibility.
- **Compatibility Testing** – Ensures that the matched organ is suitable for transplantation.
- **Transplantation Management** – Manages the logistics and scheduling of organ transplants.
- **Report Management** – Stores and manages reports related to transplant procedures.
- **Feedback and Complaint Management** – Allows users to submit feedback or complaints regarding the system or process.

Relationships

- Lines connecting actors to use cases represent interactions between the users and system functionalities.
- Admin or hospital staff may be involved in approval, matching, and transplantation management.
- Donors and recipients interact with registration, organ matching, feedback, and reporting.

This diagram visually explains how different stakeholders engage with the Organ Bank System. It ensures a clear understanding of user roles and the system's main functions, helping in system design and implementation.

4.2 Activity Diagram

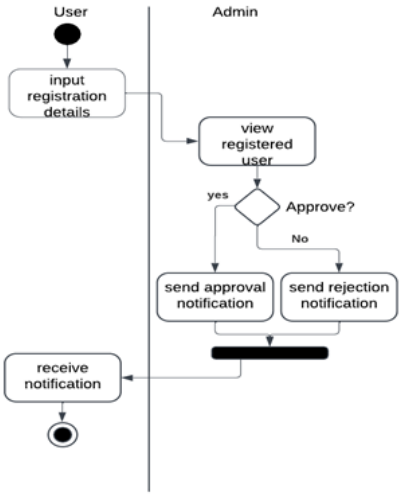


Figure 4.2: User Registration

Figure 4.2 shows the activity diagram for the user registration approval process in the Organ Bank System. The process begins with the user inputting their registration details. The request is then forwarded to the admin, who reviews the newly registered user. The admin decides whether to approve or reject the registration based on verification criteria. If approved, an approval notification is sent to the user; otherwise, a rejection notification is issued. Finally, the user receives the notification, informing them of their registration status. This diagram effectively illustrates the decision-making flow and communication between users and admin in the system.

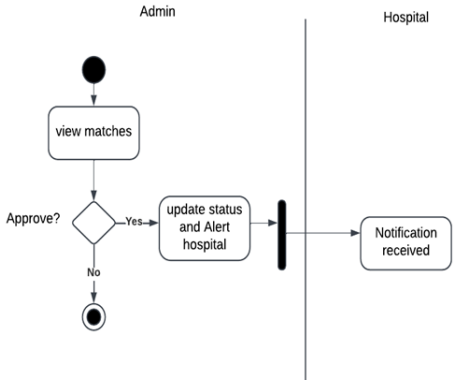


Figure 4.3: Organ Match Approval

Figure 4.3 shows the Activity Diagram that illustrates the organ match approval process in the Organ Bank System involving the Admin and Hospital. The process begins with the Admin viewing available organ matches. A decision point follows, where the Admin determines

whether to approve the match. If the match is not approved, the process terminates. If approved, the system updates the match status and sends an alert notification to the Hospital. The hospital then receives the notification, ensuring timely awareness of organ availability for transplantation. This diagram efficiently represents the decision flow and communication between system entities.

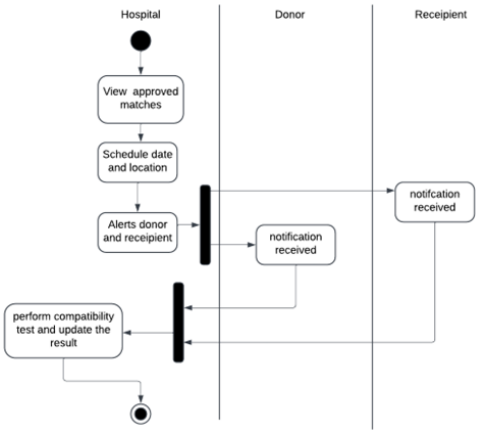


Figure 4.4: Compatibility Testing Schedule and Result Updation

Figure 4.4 shows the Activity Diagram that illustrates the compatibility testing schedule and result updation in the Organ Bank System, involving the Hospital, Donor, and recipient. The process starts with the Hospital viewing approved organ matches. The hospital then schedules a suitable date and location for the test Following this, the system sends notifications to both the Donor and recipient, informing them about the schedule. Once the notifications are received, the hospital proceeds with the compatibility test, ensuring that the organ is suitable for the recipient. The final step involves updating the test results in the system.

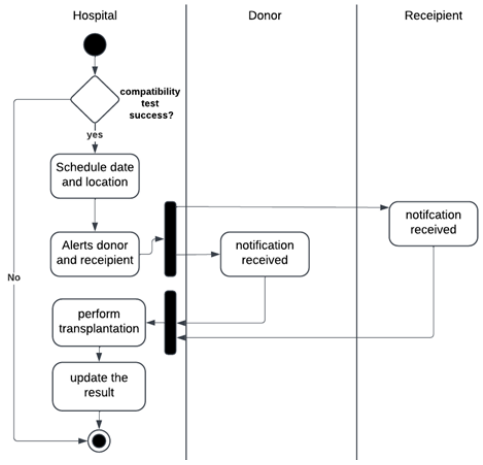


Figure 4.5: Organ Transplantation Process

Figure 4.5 illustrates the organ transplantation process in the Organ Bank System, involving the Hospital, Donor, and recipient. The process begins with a compatibility test conducted by the hospital. If the test is unsuccessful, the process terminates. If successful, the hospital proceeds

to schedule a date and location for the transplant. Notifications are then sent to both the Donor and recipient, informing them of the scheduled procedure. After receiving the notifications, the hospital performs the organ transplantation and subsequently updates the result in the system.

4.3 Sequence Diagram

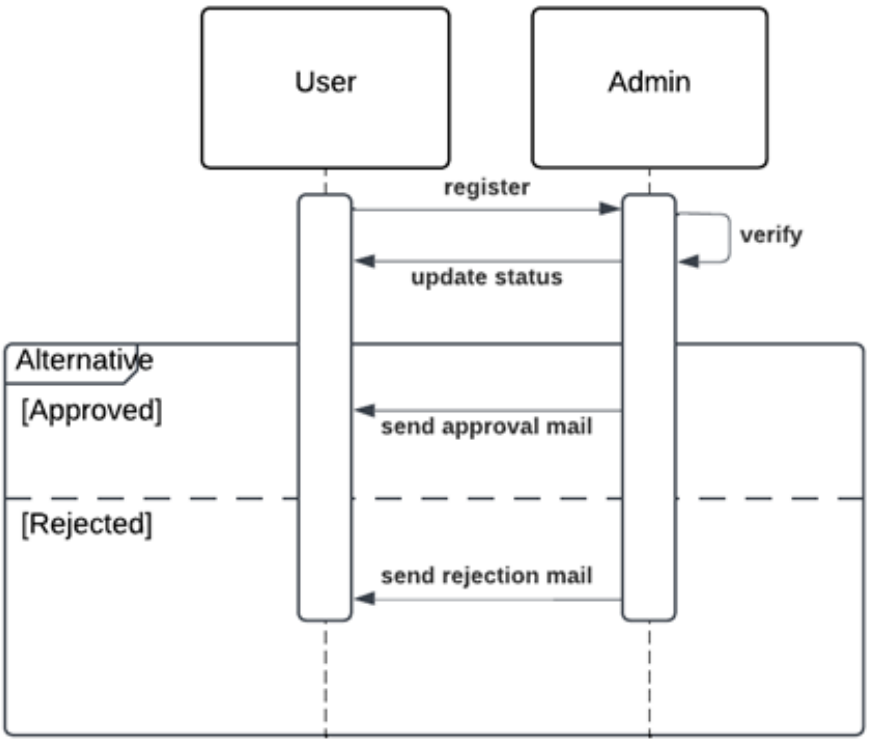


Figure 4.6: User Registration

Figure 4.6 shows the sequence diagram for user registration process , outlining how a new user joins the system. A user submits a registration request, which is then reviewed by the administrator. The administrator verifies the submitted details and updates the registration status accordingly. If the registration is approved, the system sends an approval email to the user, granting access to proceed with organ donation or request functionalities. However, if the registration is rejected, the system sends a rejection email, informing the user of the unsuccessful verification. This ensures that only verified individuals can participate in the organ donation and transplant process.

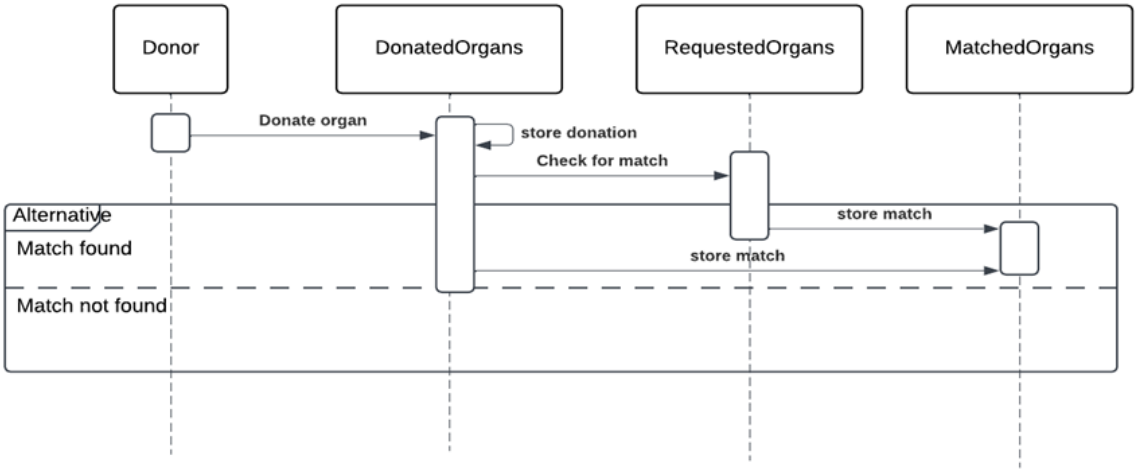


Figure 4.7: Organ Donation and Automated Matching

Figure 4.7 shows the sequence diagram that shows the process of a donor providing an organ and how the system finds a suitable recipient. Once a donor submits an organ donation request, the system stores the organ in the DonatedOrgans and then checks the RequestedOrgans to see if there is a recipient in need of the donated organ. If a match is found, the match is moved to the MatchedOrgans database. If no match is found, the donation remains in the system, waiting for a suitable recipient in the future.

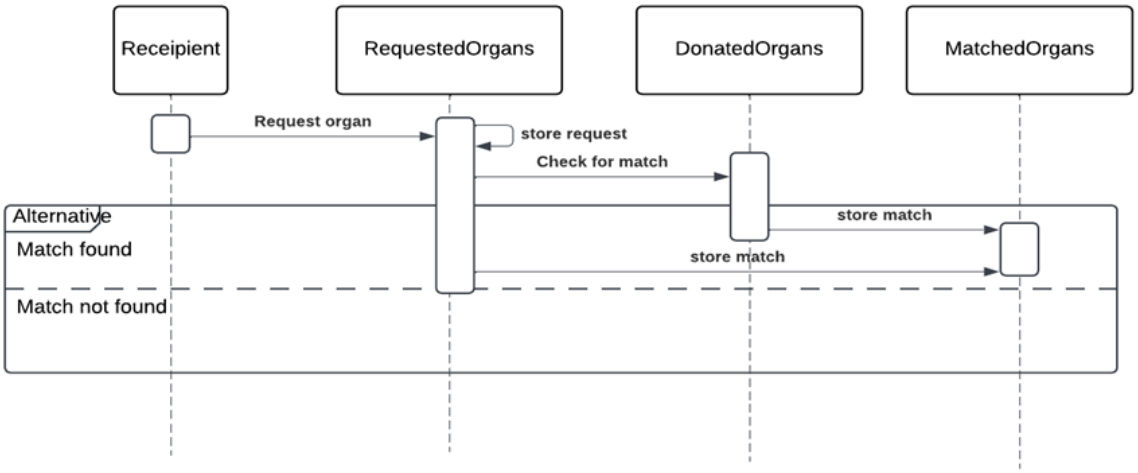


Figure 4.8: Organ Request and Automated Matching

Figure 4.8 shows the sequence diagram of organ request and matching , explaining how recipients request an organ and how the system attempts to find a match. When a recipient submits an organ request, the request is stored and then the system searches the DonatedOrgans database to check if a matching organ is available. If a match is found, it is recorded in the MatchedOrgans database, allowing the recipient to proceed with the next steps. If no match is available, the request remains in the system, and the recipient has to wait for a suitable donor.

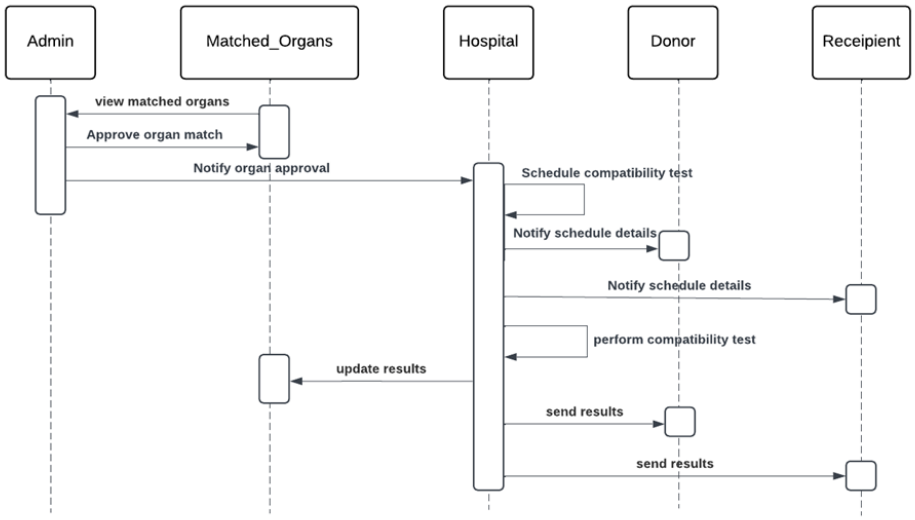


Figure 4.9: Match approval and Compatibility Test

Figure 4.9 shows the sequence diagram representing the approval process for matched organs and the subsequent scheduling of a compatibility test. Once an organ match is found, the administrator reviews the match and decides whether to approve it. Upon approval, the system notifies the hospital to arrange a compatibility test. The hospital then schedules the test and sends the details to both the donor and the recipient. After conducting the test, the hospital sends the results back to the system, which updates the records accordingly. If the test is successful, the transplantation process moves forward; otherwise, the system records the failed test, and the process is halted.

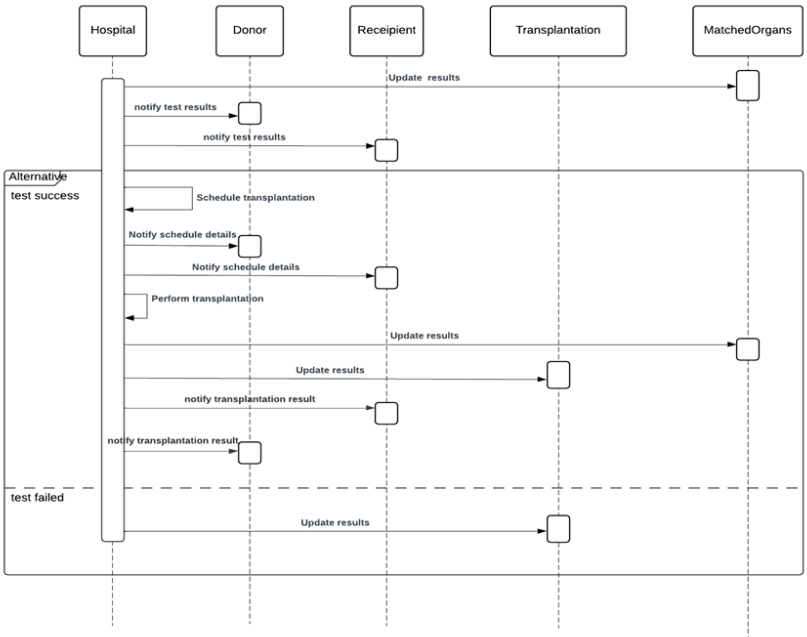


Figure 4.10: Organ Transplantation Process

Figure 4.10 shows the sequence diagram of final stage in the organ transplantation workflow, illustrating the execution of the transplantation procedure. The hospital first notifies both the donor and the recipient of the test results. If the compatibility test is successful, the hospital schedules the transplantation and informs both parties of the date and location. On the scheduled day, the transplantation is performed, and upon completion, the results are updated in the system under MatchedOrgans and Transplantation records. However, if the test fails, the system updates the records accordingly, and the recipient has to wait for another matching organ.

4.4 Identified classes, attributes, and their relationships

4.4.1 Identified Classes

The system consists of the following key classes:

1. **User** – Represents individuals who interact with the system.
2. **Admin** – Manages user registrations, organ approvals, and system operations.
3. **Hospital** – Handles organ compatibility testing, scheduling, and transplantation procedures.
4. **Donor** – Represents individuals donating organs.
5. **Recipient** – Represents individuals requesting organs.
6. **Feedback** – Stores feedback from users about the system.
7. **Complaint** – Manages user complaints.
8. **Donated_Organs** – Stores details of organs donated by donors.
9. **Requested_Organs** – Stores details of requested organs by recipients.
10. **Matched_Organs** – Maintains records of successfully matched donors and recipients.
11. **Transplantation** – Stores records of transplant procedures and their outcomes.

4.4.2 Identified Attributes

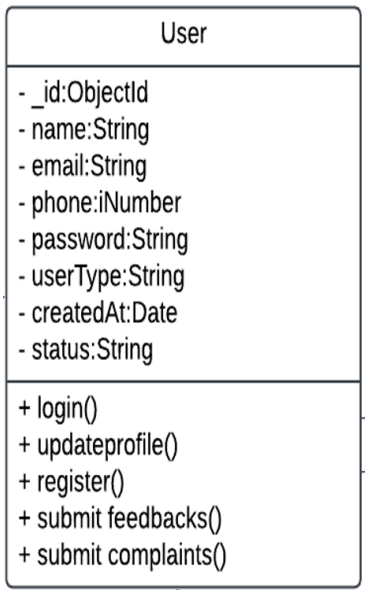


Figure 4.11: Snapshot of User Class

Figure 4.11 shows the User class represents individuals who interact with the system. Users can be of different types: donors, recipients,hospitals or admins. Each user has unique identification details, including name, email, phone number, and password. Users can perform various operations such as logging in, updating their profiles, register, submitting feedback and complaints.

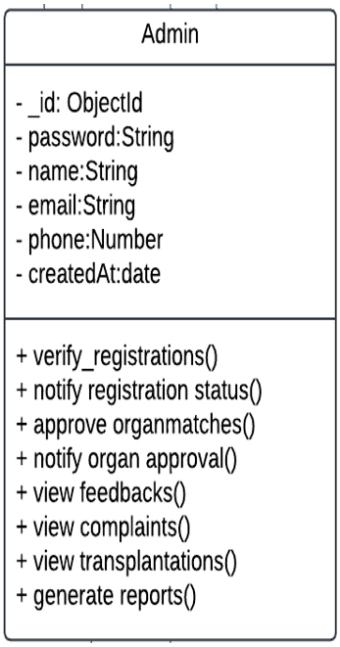


Figure 4.12: Snapshot of Admin Class

Figure 4.12 shows the Admin class is responsible for managing the system's operations. Admins can verify user registrations, approve organ matches, notify users about organ approval, and generate reports. Additionally, admins can view user feedback and complaints, ensuring smooth and transparent operations.

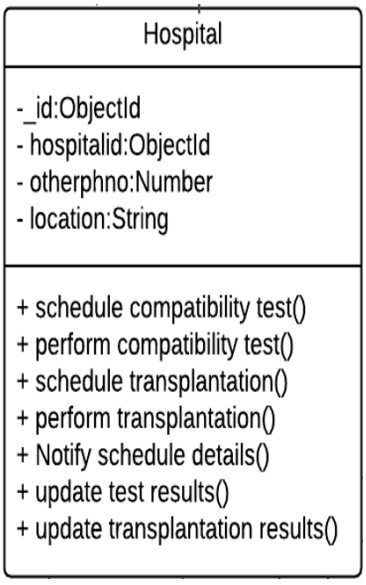


Figure 4.13: Snapshot of Hospital Class

Figure 4.13 shows the Hospital class plays a central role in the organ donation and transplantation process. Hospitals conduct compatibility tests, schedule transplantations, and notify users about test results. They ensure that only compatible organs are transplanted and that medical procedures are correctly followed.

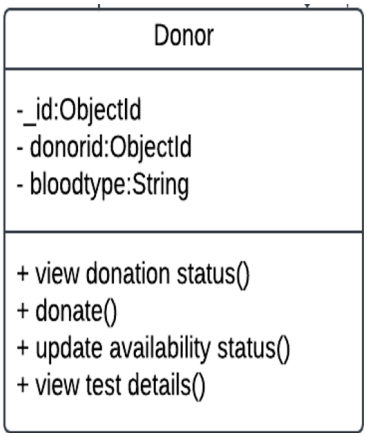


Figure 4.14: Snapshot of Donor Class

Figure 4.14 shows the Donor class represents individuals willing to donate organs. Each donor has a unique ID and blood type, which helps in determining compatibility with recipients. Donors can view their donation status and test results and update their availability status.

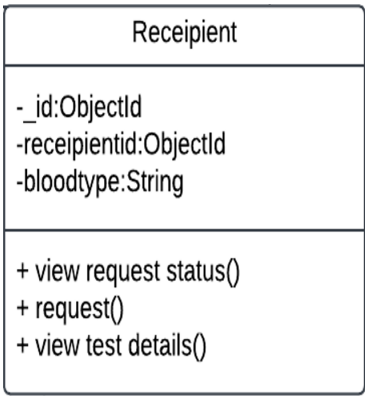


Figure 4.15: Snapshot of recipient Class

Figure 4.15 shows the recipient class represents individuals in need of organ transplants. Like donors, recipients have unique IDs and blood types, which are crucial for matching. recipients can submit organ requests, view their request status, and check compatibility test details.

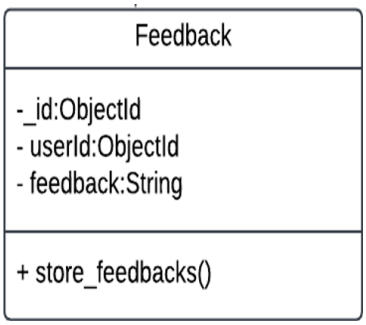


Figure 4.16: Snapshot of Feedback Class

Figure 4.16 shows the Feedback class stores feedback provided by users regarding their experiences with the system. Each feedback entry is associated with a user ID, ensuring traceability. This helps the system admin improve the efficiency and transparency of the organ donation process.

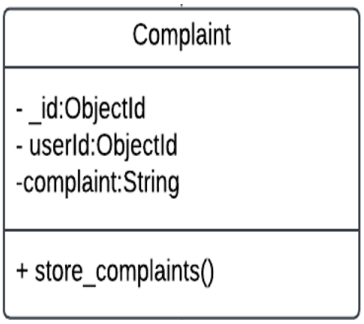


Figure 4.17: Snapshot of Complaint Class

Figure 4.17 shows the Complaint class stores complaints submitted by users regarding any issues faced during the process. Each complaint is linked to a user ID, allowing the admin to address concerns effectively and maintain a reliable system.

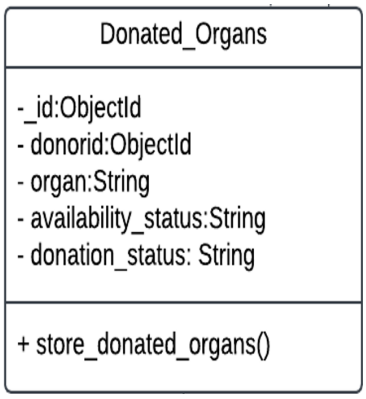


Figure 4.18: Snapshot of Donated_Organs Class

Figure 4.18 shows the Donated_Organs class maintains records of organs donated by registered donors. Each entry contains the donor ID, organ type, and availability status. This information is crucial for matching organs with recipients in need.

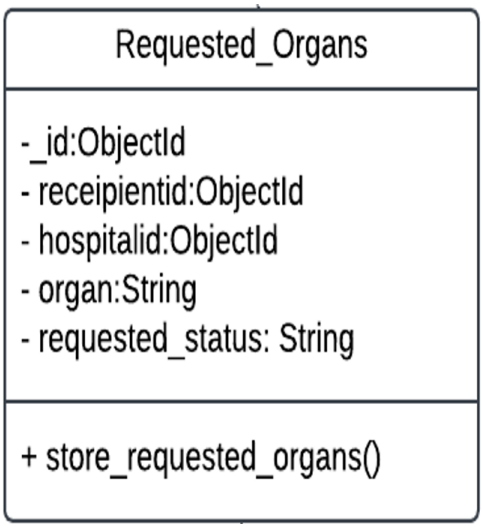


Figure 4.19: Snapshot of Requested_Organs Class

Figure 4.19 shows the Requested_Organs class stores details of organ requests submitted by recipients. It includes information such as recipient ID, requested organ type, hospital ID, and request status. This allows the system to process organ matches efficiently.

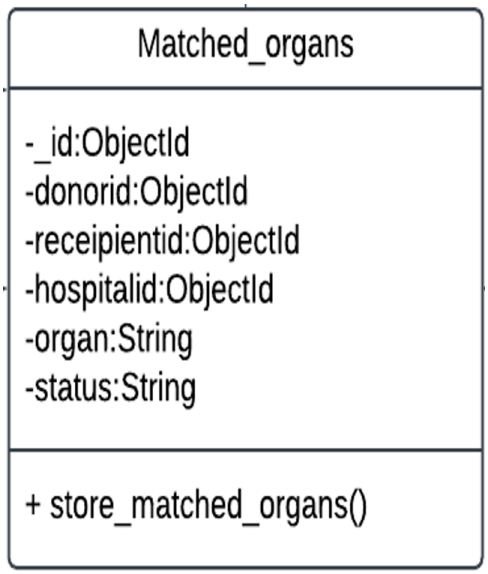


Figure 4.20: Snapshot of Matched_Organs Class

Figure 4.20 shows the Matched_Organs class records successful matches between donated and requested organs. It includes donor and recipient IDs, hospital ID, organ type, and match status. This ensures that only compatible organ transplants proceed to the next stage.

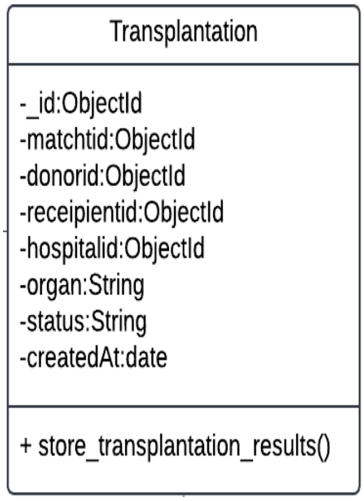


Figure 4.21: Snapshot of Transplantation Class

Figure 4.21 shows the Transplantation class maintains records of completed transplant procedures.This helps in tracking successful organ transplants and generating reports.

4.4.3 Identified Relationships

Class	**Related Class**	**Association Name**	**Cardinality**
User	Feedback	Submits Feedback	1 - *
User	Complaint	Submits Complaint	1 - *
Admin	User	Verifies Registration	1 - *
Admin	Hospital	Notify organ approval	1 - *
Admin	Matched_Organs	Update organ status	1 - *
Admin	Feedback	Views	1 - *
Admin	Complaint	Views	1 - *
Admin	Transplantation	Generate reports	1 - 1
Hospital	Donor	Update test results	1 - *
Hospital	Recipient	Update test results	1 - *
Hospital	Donor	Notify schedule details	1 - *
Hospital	Recipient	Notify schedule details	1 - *
Hospital	Matched_Organs	Update results	1 - *
Donor	Donated_Organs	Donates Organ	1 - *
Recipient	Requested_Organs	Requests Organ	1 - *
Matched_Organs	Transplantation	Leads to Transplantation	1 - 1

Table 4.1: Identified Relationships

Table 4.1 shows the related classes,it's association and relationships within Organ Bank.

4.5 Class Diagram

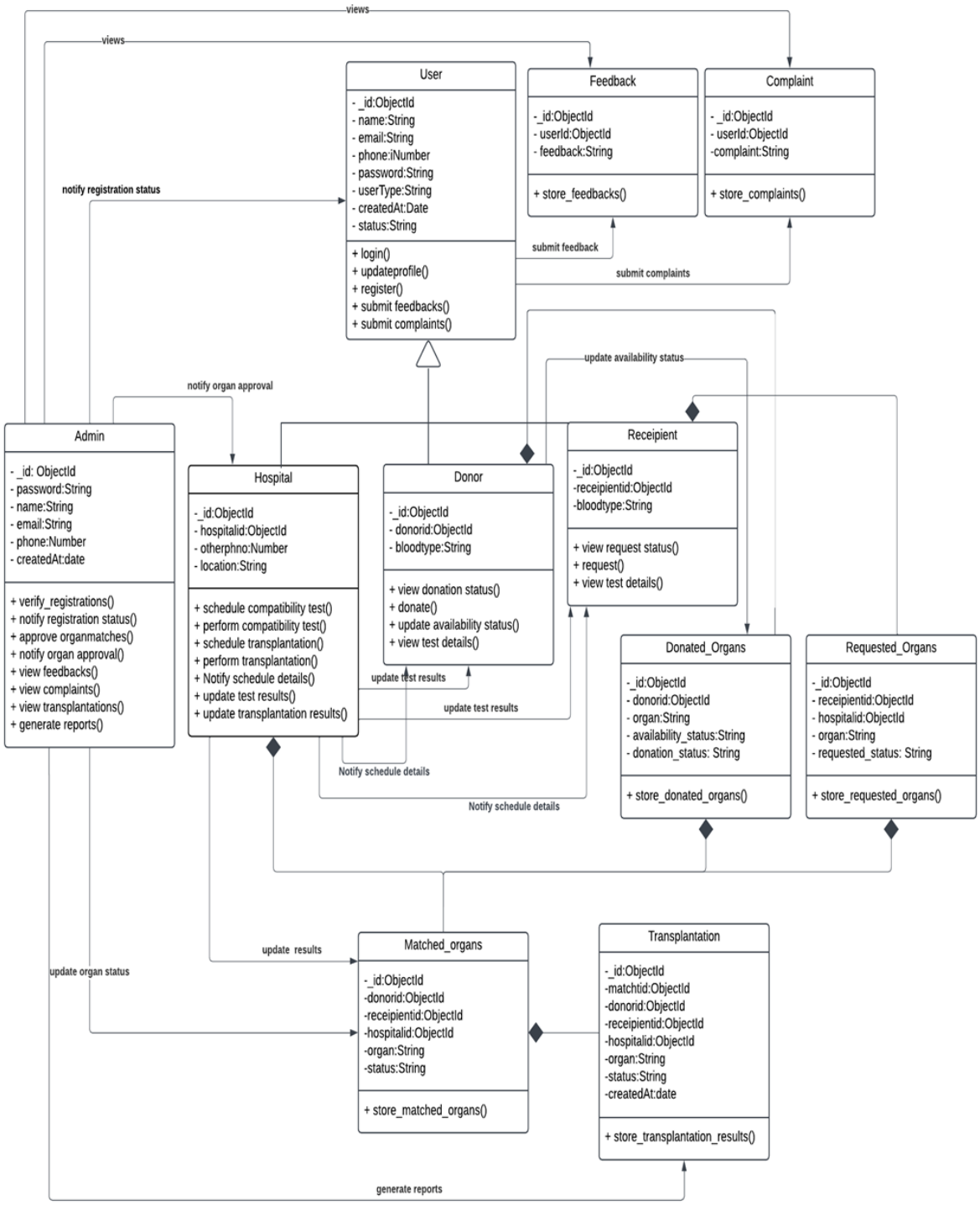


Figure 4.22: Snapshot of Transplantation Class

4.6 Database Design

Attributes	Datatype	Constraints	Description
_id	Objectid	Unique	Autogenerated id
name	String		Name of user
email	String	Unique	Email of user
phone	Number		Phone number
password	String		Password for login
userType	String		Role of user
createdAt	Date		Registered date
status	String		Status of registration

Table 4.2: User Table

Attributes	Datatype	Constraints	Description
_id	Objectid	Unique	Autogenerated id
donorid	Objectid	Ref(User._id)	Donor's id
bloodtype	String		Blood group of donor

Table 4.3: Donor Table

Attributes	Datatype	Constraints	Description
_id	Objectid	Unique	Autogenerated id
recipientid	Objectid	Ref(User._id)	recipient's id
bloodtype	String		Blood group of recipient

Table 4.4: Recipient Table

Attributes	Datatype	Constraints	Description
_id	Objectid	Unique	Autogenerated id
name	String		Name of admin
email	String		Email of admin
phone	Number		Phone number
password	String		Password for login
createdAt	Date		Created date

Table 4.5: Admin Table

Attributes	Datatype	Constraints	Description
_id	Objectid	Unique	Autogenerated id
hospitalid	Objectid	Ref(User._id)	Hospital's id
otherphno	Number		Other phone number
location	String		Location of hospital

Table 4.6: Hospital Table

Attributes	Datatype	Constraints	Description
_id	Objectid	Unique	Autogenerated id
donorid	Objectid	Ref(User._id)	Donor's id
organ	String		Donated organ
availability_status	String		Available or not
donation_status	String		Status of donation

Table 4.7: Donated Organs Table

Attributes	Datatype	Constraints	Description
_id	Objectid	Unique	Autogenerated id
recipientid	Objectid	Ref(User._id)	recipient's id
hospitalid	Objectid	Ref(User._id)	ID of the hospital selected by recipient
organ	String		Requested organ
requested_status	String		Status of request

Table 4.8: Requested Organs Table

Attributes	Datatype	Constraints	Description
_id	Objectid	Unique	Autogenerated id
donorid	Objectid	Ref(Donated_Organs: donorid)	Donor's id
recipientid	Objectid	Ref(Requested_Organs: recipientid)	Recipient's id
hospitalid	Objectid	Ref(Requested_Organs: hospitalid)	Hospital's id
organ	String		Matched organ
status	String		Status of organ

Table 4.9: Matched Organs Table

Attributes	Datatype	Constraints	Description
_id	Objectid	Unique	Autogenerated id
matchid	Objectid	Ref(Matched_Organs: _id)	Match's id
donorid	Objectid	Ref(Matched_Organs: donorid)	Donor's id
recipientid	Objectid	Ref(Matched_Organs: recipientid)	Recipient's id
hospitalid	Objectid	Ref(Matched_Organs: hospitalid)	Hospital's id
organ	String		Organ transplanted
status	String		Status of transplantation
createdAt	Date		Transplanted date

Table 4.10: Transplantation Table

Attributes	Datatype	Constraints	Description
_id	Objectid	Unique	Autogenerated id
userId	Objectid	Ref(User._id)	User's id
feedback	String		Feedback

Table 4.11: Feedback Table

Attributes	Datatype	Constraints	Description
_id	Objectid	Unique	Autogenerated id
userId	Objectid	Ref(User._id)	User's id
complaint	String		Complaint

Table 4.12: Complaint Table

4.7 UI Designs

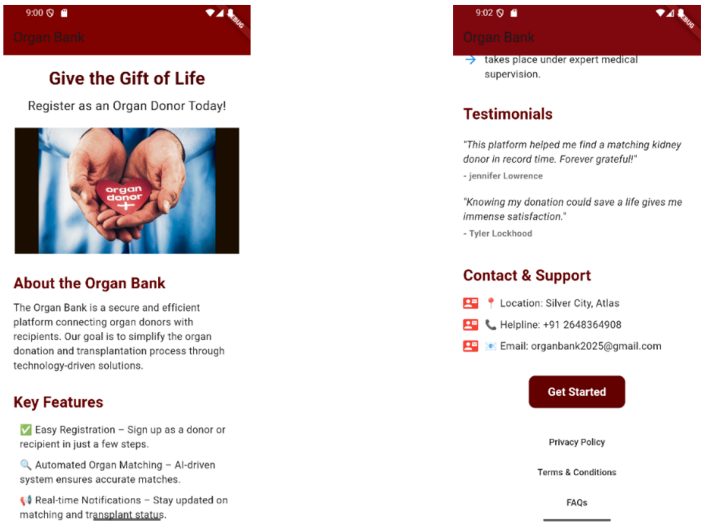


Figure 4.23: Snapshot of Home Screen

Figure 4.23 shows the home Screen of the Organ Bank app introduces the platform and encourages users to register as organ donors. It features a motivational banner with an image emphasizing the importance of organ donation, followed by an ”About the Organ Bank” section explaining its purpose. The ”Key Features” section highlights easy registration, AI-driven organ matching, and real-time notifications for updates on transplant status. The testimonials Screen showcases user experiences, reinforcing trust in the platform’s effectiveness. The ”Contact & Support” section provides essential details like location, helpline, email, and a ”Get Started” button for user engagement.

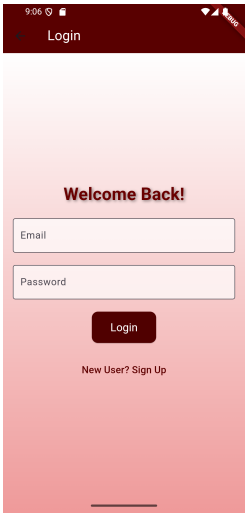


Figure 4.24: Snapshot of Home Screen

Figure 4.24 shows the login screen of the Organ Bank app has a simple and user-friendly design

with a gradient background. It includes a welcoming message, ”Welcome Back!”, encouraging users to log in. There are two input fields for email and password, ensuring a straightforward authentication process. The ”Login” button is prominently placed for easy access, and a ”New User? Sign Up” option is provided for first-time users, when clicked it will start with new user registration. The layout is clean, minimalistic, and visually appealing to enhance user experience.

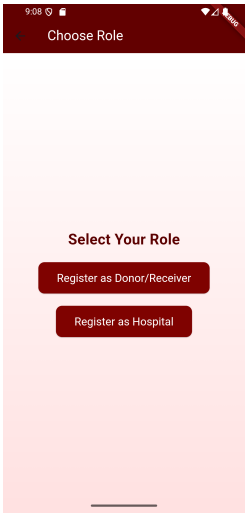


Figure 4.25: Snapshot of Choose Role Screen

Figure 4.25 shows the role selection screen allows users to choose their registration type in the Organ Bank app. It presents two distinct options: ”Register as Donor/Receiver” and ”Register as Hospital”, ensuring clarity in user categorization. This screen ensures a smooth onboarding experience by guiding users based on their intended role in the system.After clicking the any of the buttons, it will navigate to corresponding registration Screens.

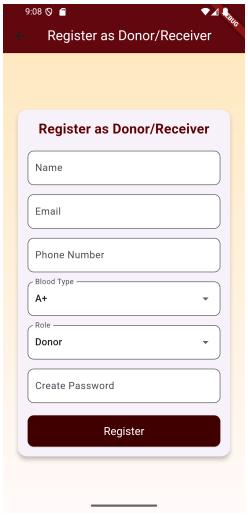


Figure 4.26: Donor/Receiver Registration Screen

Figure 4.26 shows the Donor/Receiver Registration Screen enables users to sign up by providing essential details. It includes input fields for name, email, phone number, blood type,

role selection (Donor or recipient), and password creation. A button at the bottom finalizes the process, guiding users to complete their registration and give a message indicating a successful registration.

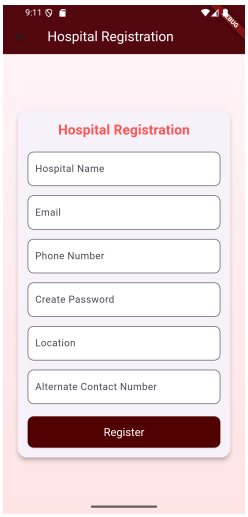


Figure 4.27: Hospital Registration Screen

Figure 4.27 shows the Hospital Registration Screen allows hospitals to sign up for the organ donation platform. It includes input fields for hospital name, email, phone number, password creation, location, and an alternate contact number. A ”Register” button at the bottom finalizes the registration process and give a message indicating a successful registration.This screen ensures that hospitals provide essential contact and location details to participate in the organ donation and transplantation network.

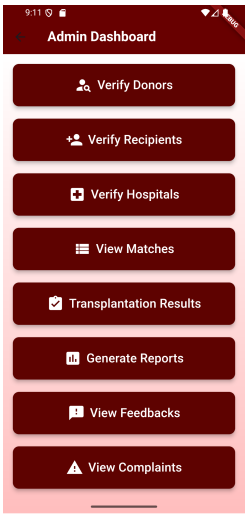


Figure 4.28: Snapshot of Admin Dashboard Screen

Figure 4.28 shows the Admin Dashboard Screen provides an interface for admin to manage the organ donation platform. It includes options to verify donors, recipients, and hospitals, ensuring only authorized users participate. Additional functionalities allow admins to view

matches, check transplantation results, generate reports, and monitor feedback and complaints. The layout consists of well-spaced, easily accessible buttons with clear labels for navigation. This screen serves as the central hub for managing and overseeing the platform's operations efficiently.

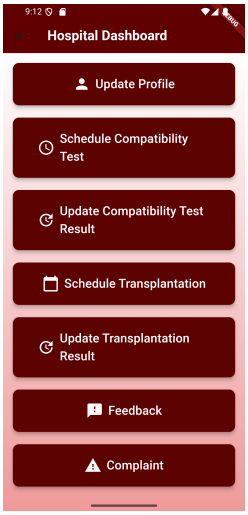


Figure 4.29: Snapshot of Hospital Dashboard Screen

Figure 4.29 shows the hospital dashboard provides functionalities specific to medical institutions, such as scheduling and updating compatibility tests and transplantations. Additional options include updating profiles, submitting feedback, and filing complaints. The interface is structured with clearly labeled buttons to guide hospital representatives through their tasks. The use of icons enhances usability, making it easier to identify each function at a glance. This dashboard ensures hospitals can seamlessly manage their responsibilities within the organ donation system.

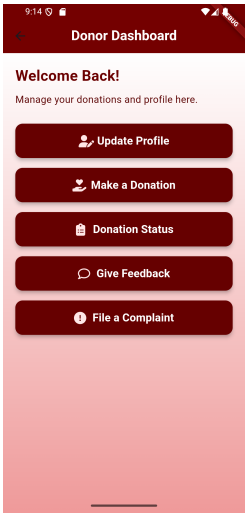


Figure 4.30: Snapshot of Donor Dashboard Screen

Figure 4.30 shows the donor dashboard welcomes donors and allows them to manage their donations and profile. It provides options to update their profile, make a donation, check donation

status, give feedback, and file complaints. The button layout is simple and user-friendly, ensuring easy access to essential features. The welcoming text at the top enhances user engagement by making them feel valued. This dashboard serves as a central hub for donors to track and manage their contributions efficiently.

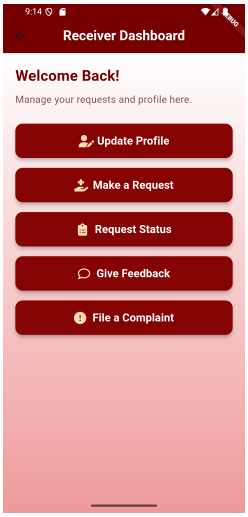


Figure 4.31: Snapshot of recipient Dashboard Screen

Figure 4.31 shows the recipient dashboard welcomes recipient and allows them to manage their requests and profile. It provides options to update their profile, make a request, check request status, give feedback, and file complaints. The button layout is simple and user-friendly, ensuring easy access to essential features. The welcoming text at the top enhances user engagement by making them feel valued. This dashboard serves as a central hub for recipients to track and manage their contributions efficiently.

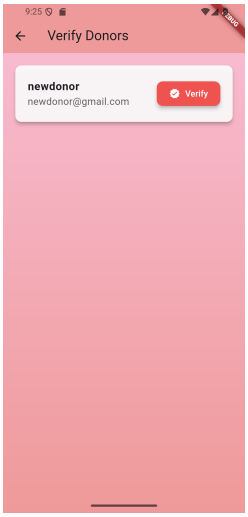


Figure 4.32: Snapshot of Verify Donor Screen

Figure 4.32 shows the donor check screen where all donors pending for admin approval are listed. By clicking the verify button, it will navigate to the other screen shown in the figure 4.33.

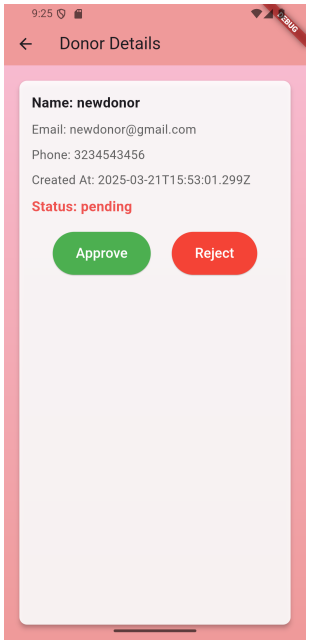


Figure 4.33: Snapshot of Donor Details Screen

Figure 4.33 shows the details of the selected donor along with two buttons.When clicking Approve button the donor is approved and allow him to login to the system.When clicking Reject button the donor is rejected preventing donor from accessing the system.Along with that mail is also send to donor regarding their registration status.

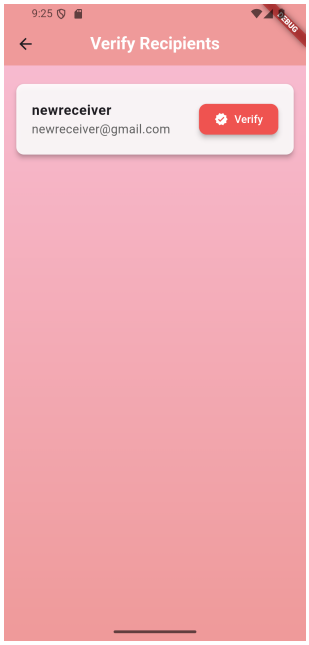


Figure 4.34: Snapshot of Verify recipients Screen

Figure 4.34 shows verify recipient Screen where all the pending recipients are listed for admin approval.Upon clicking verify button it will navigate to anouther Screen shown in Figure 4.35.

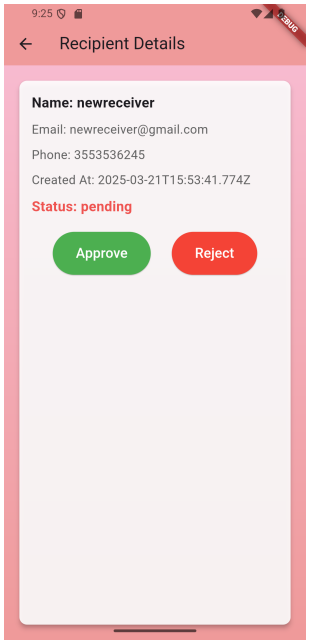


Figure 4.35: Snapshot of recipient Details Screen

Figure 4.35 shows the details of the selected recipient along with two buttons.When clicking Approve button the recipient is approved and allow him to login to the system.When clicking Reject button the recipient is rejected preventing recipient from accessing the system.Along with that mail is also send to recipient regarding their registration status.

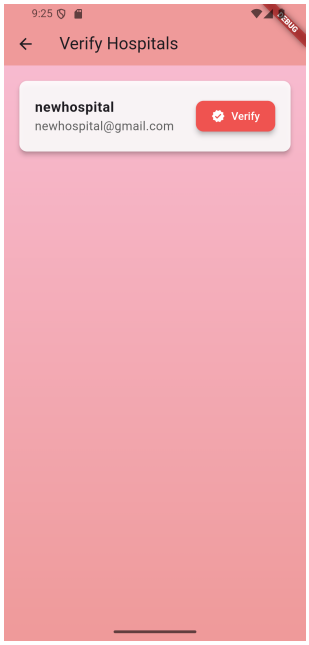


Figure 4.36: Snapshot of Verify Hospitals Screen

Figure 4.36 shows verify hospital Screen where all the pending hospitals are listed for admin approval.Upon clicking verify button it will navigate to anouther Screen shown in Figure 4.37 .

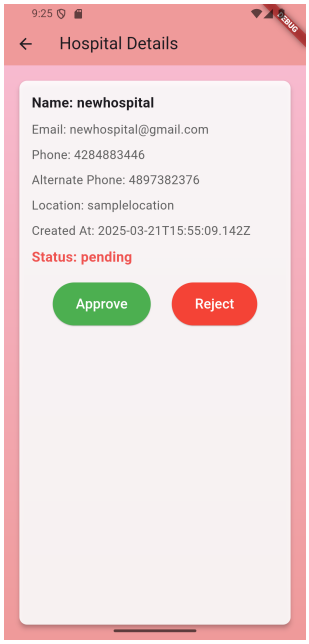


Figure 4.37: Snapshot of Hospital Details Screen

Figure 4.37 shows the details of the selected hospital along with two buttons.When clicking Approve button the hospital is approved and allow him to login to the system.When clicking Reject button the hospital is rejected preventing hospital from accessing the system.Along with that mail is also send to hospital regarding their registration status.

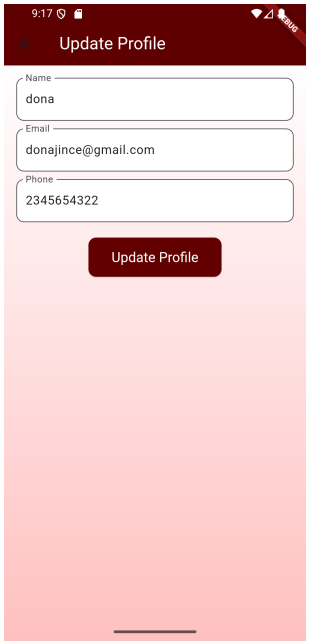


Figure 4.38: Snapshot of Update Profile Screen

Figure 4.38 shows the UI for updating donor/recipient profile.After updation click update profile for saving the changes.A message is also shown that conveys profile is updated successfully.

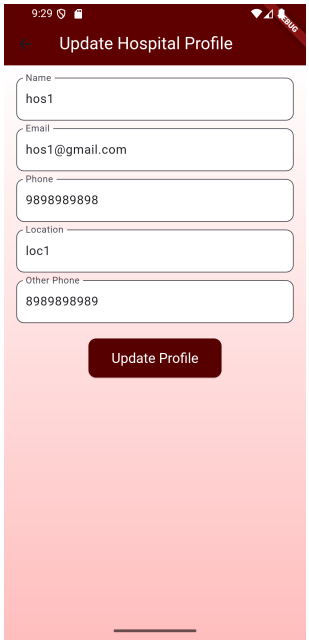


Figure 4.39: Snapshot of Update Hospital Profile Screen

Figure 4.39 shows the UI for updating hospital profile.After updation click update profile for saving the changes.A message is also shown that conveys profile is updated successfully.

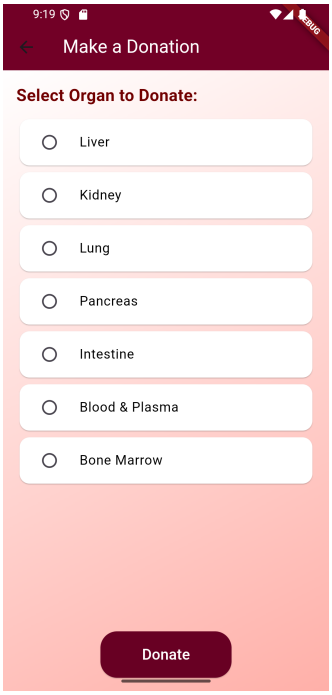


Figure 4.40: Snapshot of Make Donations Screen

Figure 4.40 shows the UI for making donation.After selecting the organ click donate to submit donation.A message is shown indicating donation made successfully.Along with that automated matching is also initiated and if there is a match, the match will be stored for admin approval.

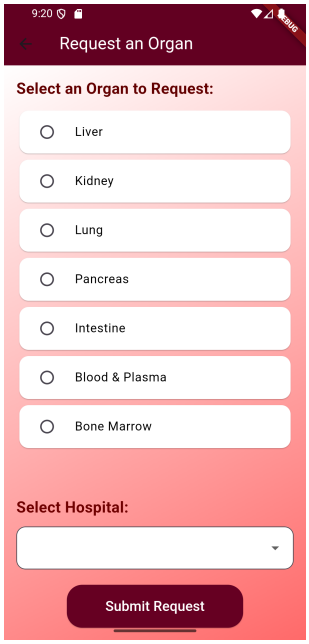


Figure 4.41: Snapshot of Make Request Screen

Figure 4.41 shows the UI for making request.After selecting the organ and hospital, click Submit Request to submit request.A message is shown indicating request made successfully.Along with that automated matching is also initiated and if there is a match, the match will be stored for admin approval.

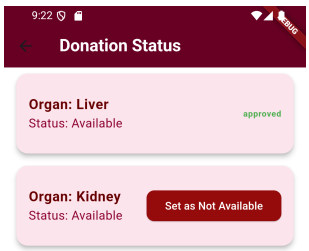


Figure 4.42: Snapshot of View Donation Status Screen

Figure 4.42 shows the UI for viewing donation status. If the status of the donation is pending ,donor will be able to change it's availability status, else donor can view the status.

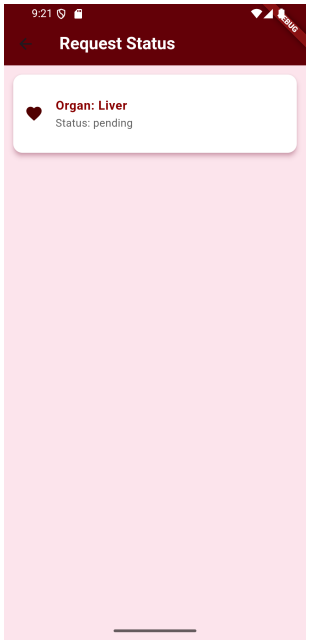


Figure 4.43: Snapshot of View Request Status Screen

Figure 4.43 shows the UI for viewing request status.

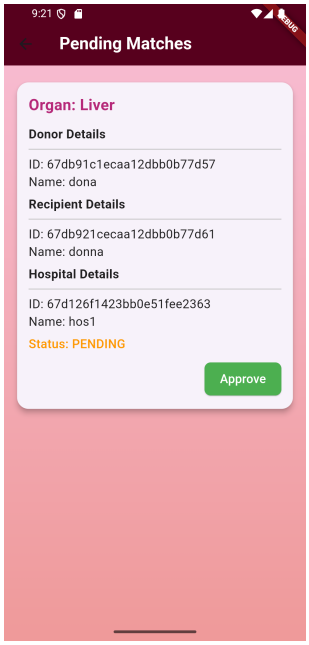


Figure 4.44: Snapshot of Pending Matches Screen

Figure 4.44 shows the UI for viewing matches by admin. On approving the match the respective hospital will be alerted by sending a mail that a match is found.

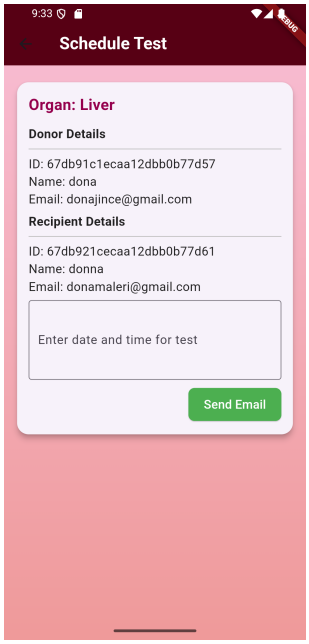


Figure 4.45: Snapshot of Schedule Compatibility Test Screen

Figure 4.45 shows the UI for scheduling the compatibility test.It displays all the approved matches and the details of the donor and recipient.It also has a text box for typing the date and time for the test.On clicking the button it will send mails to the respective donor and recipient.

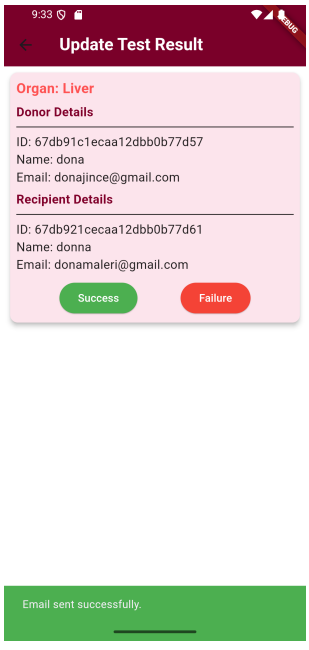


Figure 4.46: Snapshot of Update Test Result Screen

Figure 4.46 shows the UI for updating the test result.On clicking the Success button, status will be updated to “success”,else “failure”.

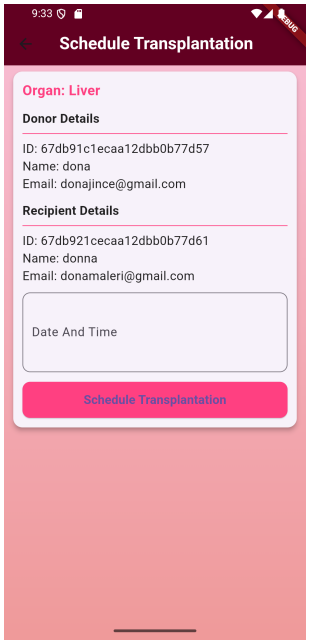


Figure 4.47: Snapshot of Schedule Transplantation Screen

Figure 4.47 shows the UI for scheduling the transplantation for the matches with the successful compatibility test.It displays the details of the donor and recipient.It also has a text box for typing the date and time for the transplantation. On clicking the button it will send mails to the respective donor and recipient.

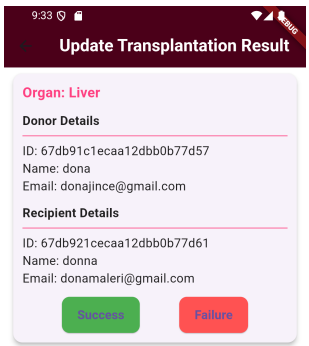


Figure 4.48: Snapshot of Update Transplantation Screen

Figure 4.48 shows the UI for updating the test result.On clicking the Success button, status will be updated to “transplantationsuccess”,else “transplantationfailure”.

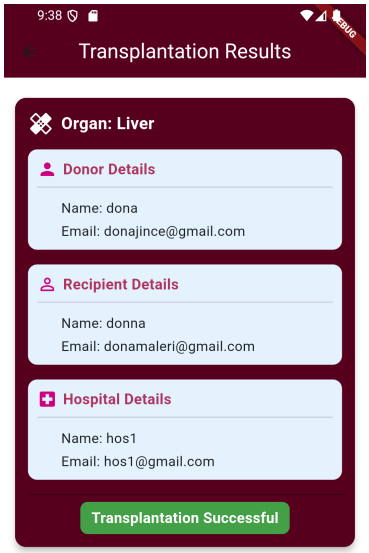


Figure 4.49: Snapshot of View Transplantation Screen

Figure 4.49 shows all the transplantation data along with participating donor, recipient, hospitals details and it's status.

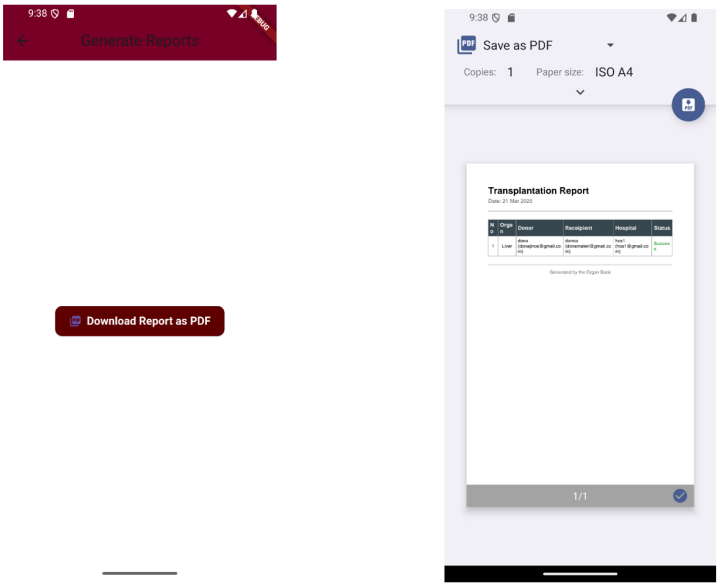


Figure 4.50: Snapshot of Generate Report Screen

Figure 4.50 shows the UI for generating reports.On clicking the button it will navigate to anouther Screen as shown in the figure where it displays a generated report of current day's transplantation along with it's associated details.On clicking the "PDF" icon we will be able to download and save the pdf.

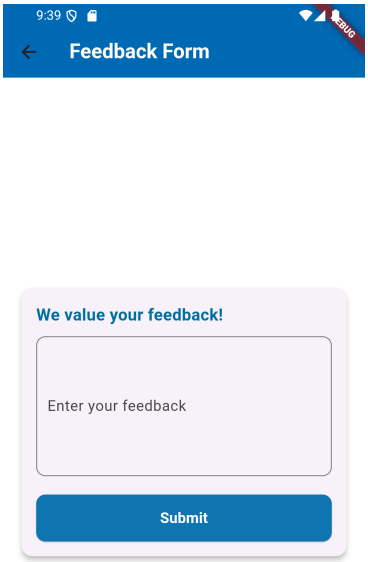


Figure 4.51: Snapshot of Submit Feedback Screen

Figure 4.51 shows the feedback form where user can enter the feedback and submit it. On successful feedback submission it will give a message indicating feedback is submitted successfully.

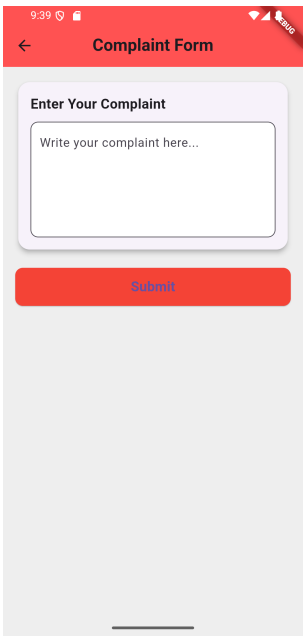


Figure 4.52: Snapshot of Submit Complaint Screen

Figure 4.52 shows the complaint form where user can enter the complaint and submit it. On

successful complaint submission it will give a message indicating complaint is submitted successfully.

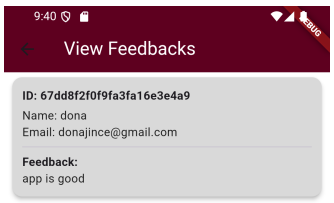


Figure 4.53: Snapshot of View Feedbacks Screen

Figure 4.53 shows the user interface that displays all feedback from users.

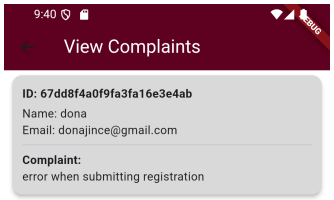


Figure 4.54: Snapshot of View Complaints Screen

Figure 4.54 shows the user interface that displays all complaints submitted by users.

5 TESTING

Software testing is a critical element of software quality assurance and represents ultimate review of specification, design and code generation. Once the source code has been generated the program should be executed before the customer gets it with the specific intend of finding and removing all errors, test must be conducted systematically and test must be designed using disciplined techniques. Software testing is a critical element of software quality assurance and represents the ultimate review of specifications, design and code generation. Once the source code has been generated, software must be tested to uncover as many errors as possible before delivery to the customer. In order to find the highest possible number of errors, tests must be conducted systematically and test case must be designed using disciplined techniques.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically attempts to execute a program or application with the intent of finding software bugs . Software testing can be stated as the process of verifying and validating that a software or application is bug free, meets the technical requirements as guided by it's design and development and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases.

The process of software testing aims not only to find faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability. It mainly aims at measuring specification, functionality and performance of a software program or application. Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. It can be either done manually or using automated tools. Software testing can be divided into two steps:

Verification: It refers to the set of tasks that ensure that software correctly implements a specific function.

Validation: It refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.

The testing process can be divided into five phases:

1. **Unit Testing**
2. **Integration Testing**
3. **Final/System Testing**
4. **Backend Testing**
5. **GUI Testing**

5.1 Unit Testing

In Organ Bank project, unit testing focuses on verifying the smallest functional components of the system, such as individual functions, services, or modules, before integrating them into the full application. Field testing ensures that input fields, such as donor and recipient registration forms, accept valid inputs while rejecting invalid ones. At the module level, core functionalities like user authentication, donor-recipient matching, and database operations are tested independently to ensure they perform as expected. Control path testing verifies that data flows correctly within each module, such as ensuring that donor details are properly processed and stored in MongoDB. Additionally, boundary condition testing is conducted to check how the system handles edge cases, such as maximum and minimum input values or unexpected inputs, ensuring robust performance. Lastly, unit testing ensures that all logical paths in a function or module are executed at least once, allowing for comprehensive validation of key features like the organ-matching algorithm. By implementing unit testing in both the Flutter frontend and Node.js backend, potential errors can be identified early, improving code reliability and ensuring seamless system functionality before progressing to integration and system testing.

5.1.1 Unit Test Cases

Sl.no	Procedure	Expected Output	Actual Output	Status
1	Validate Donor/Recipient Registration with Valid Data	Registration Success	Registration Success	Pass
2	Validate Donor/Recipient Registration with invalid data/missing data	Registration failed	Registration failed	Pass
3	Validate Hospital Registration with Valid data	Registration Success	Registration Success	Pass
4	Validate Hospital Registration with invalid data/missing data	Registration failed	Registration failed	Pass
5	Validate login by giving approved user credentials	Login successful	Login successful	Pass
6	Validate login by giving pending user credentials	Message indicating wait for admin approval	Message indicating wait for admin approval	Pass

Sl.no	Procedure	Expected Output	Actual Output	Status
7	Validate login by giving invalid user credentials	Message indicating it is invalid credentials	Message indicating it is invalid credentials	Pass
8	Validate user registration approval	Approved Successfully	Approved Successfully	Pass
9	Validate user registration rejection	Rejected Successfully	Rejected Successfully	Pass
10	Submit donation without selecting the organ	Donation unsuccessful	Donation unsuccessful	Pass
11	Submit donation by selecting an organ	Donation successful	Donation successful	Pass
12	Submit organ request by selecting the organ and hospital	Request successful	Request successful	Pass
13	Submit organ request by not selecting an organ or hospital	Request unsuccessful	Request unsuccessful	Pass
14	Approval of match	Match approved successfully and mail sent to hospital	Match approved successfully and mail sent to hospital	Pass
15	Schedule test by not providing date and time	Failed to schedule test and mailing	Failed to schedule test and mailing	Pass
16	Schedule test by providing date and time	Schedule test and mailing successful	Schedule test and mailing successful	Pass
17	Schedule transplantation by providing date and time	Schedule transplantation and mailing successful	Schedule transplantation and mailing successful	Pass
18	Schedule transplantation by not providing date and time	Failed to schedule transplantation and mailing	Failed to schedule transplantation and mailing	Pass
19	Submit feedback by providing feedback	Feedback submitted successfully	Feedback submitted successfully	Pass

Sl.no	Procedure	Expected Output	Actual Output	Status
20	Submit complaint by providing complaint	Complaint submitted successfully	Complaint submitted successfully	Pass

Table 5.1: Unit Test Cases

5.2 Integration Testing

In the Organ Bank project, integration testing plays a crucial role in systematically constructing the program structure while identifying errors in module interactions. This ensures that data is correctly transferred between modules without loss and that combined sub-functions deliver the expected major functionality. The project follows a top-down integration approach, where modules are integrated step by step, starting from the main module and progressively moving downward through the control hierarchy. Initially, individual modules such as user authentication, donor-recipient matching, and organ database management are unit tested to verify their correctness. Once validated, they are integrated one by one, ensuring seamless communication and proper data flow between them. After successful integration, the system was tested to confirm that all modules functioned as per the system design, with no uncovered errors.

5.2.1 Integration Test Cases

Sl.no	Procedure	Expected Output	Actual Output	Status
1	Integration of login and user dashboards	Approved users should be redirected to respective dashboards based on role	Approved users are redirected to respective dashboards based on role	Pass
2	Integration of make donation and organ matching	Donation submitted successfully and corresponding matches should be stored for admin approval	Donations submitted successfully and corresponding matches are stored for admin approval	Pass
3	Integration of make request and organ matching	Request submitted successfully and corresponding matches should be stored for admin approval	Request submitted successfully and corresponding matches are stored for admin approval	Pass

Sl.no	Procedure	Expected Output	Actual Output	Status
4	Integration of match approval and schedule test	After successful approval, hospital should be able to schedule test	After successful approval, hospital is able to schedule test	Pass
5	Integration of schedule test and update test result	After test scheduling, hospital should be able to update test result	After test scheduling, hospital is able to update test result	Pass
6	Integration of update test result and schedule transplantation	After updating result (success), hospital should be able to schedule transplantation	After updating result (success), hospital is able to schedule transplantation	Pass
7	Integration of schedule transplantation and update transplantation result	After scheduling transplantation, hospital should be able to update transplantation result	After scheduling transplantation, hospital is able to update transplantation result	Pass
8	Integration of make donations and donation status	After making donations, donor should be able to view the status of donation	After making donations, donor is able to view the status of donation	Pass
9	Integration of make request and request status	After making request, recipient should be able to view the status of request	After making request, recipient is able to view the status of request	Pass

Table 5.2: Integration Test Cases

5.3 System Testing

In the Organ Bank project, the implementation of a computer-based system requires careful preparation of test data and a structured approach to testing its components. Since the computer program component is a crucial sub-system, special attention is given to its testing at various stages of development. Errors can be introduced at any phase, making it essential to employ different techniques for detection and elimination. Testing is not limited to a single phase; instead, system testing includes both verification and validation, which are categorized into static and dynamic methods. In the static method, system behavior is analyzed without execution, focusing on aspects like code reviews and requirement validation. In contrast, the dynamic method involves executing the system to observe its behavior and ensure expected

outcomes. All modules in the Flutter frontend and Node.js backend were tested individually and successfully executed, ensuring the system functions correctly as designed.

Objective: Evaluate the integrated components of the Organ Bank system to ensure end-to-end functionality and performance within organizational environments. This comprehensive testing process encompasses various functionalities, including user registration, hospital registration, organ donation, organ request, and feedback submission, to validate their seamless integration and operation. Additionally, system testing evaluates the system's robustness, scalability, and security measures to identify and address any potential issues or vulnerabilities before deployment.

Test Case: Simulate end-to-end organ donation and request process.

Expected Output

1. **User Registration:** User data stored securely, confirmation message displayed upon successful registration.
2. **Hospital Registration:** Hospital details accurately captured, including name, email, phone number, and location.
3. **Organ Donation:** Donors can select organs to donate, and the system captures donation details accurately.
4. **Organ Request:** Recipients can request organs, and the system captures request details accurately.
5. **Feedback Submission:** System facilitates feedback submission and displays feedback for continuous improvement.
6. **Security Measures:** User authentication mechanisms ensure access control, data encryption safeguards sensitive information.

Actual Output

1. **User Registration:**
    - User data securely stored in the system.
    - Confirmation message displayed upon successful registration.
    - **Status: Pass**

2. **Hospital Registration:**
    - Hospital details accurately captured.
    - Confirmation message displayed upon successful registration.
    - **Status: Pass**
3. **Organ Donation:**
    - Donors can select organs to donate.
    - System captures donation details accurately.
    - Confirmation message displayed upon successful donation submission.
    - **Status: Pass**
4. **Organ Request:**
    - Recipients can request organs.
    - System captures request details accurately.
    - Confirmation message displayed upon successful request submission.
    - **Status: Pass**
5. **Feedback Submission:**
    - System facilitates feedback submission.
    - Feedback displayed for continuous improvement.
    - Confirmation message displayed upon successful feedback submission.
    - **Status: Pass**
6. **Security Measures:**
    - Effective implementation of user authentication and data encryption mechanisms.
    - Access control ensures only authorized users can access sensitive information.
    - **Status: Pass**

Conclusion: The Organ Bank system has successfully passed the system testing phase. All functionalities, including user registration, hospital registration, organ donation, organ request, and feedback submission, are working as expected. The system is robust, scalable, and secure, with effective user authentication and data encryption mechanisms in place to protect system integrity and user data.

5.4 Backend Testing

Backend testing, also known as database and server-side testing, is a critical component in ensuring the reliability and functionality of the Organ Bank project. This type of testing focuses on verifying server-side operations, data integrity, and interactions with the database. Without proper backend testing, the system may encounter severe issues such as data corruption, data inconsistency, security vulnerabilities, or transaction failures.

In the Organ Bank project, Postman is utilized as a powerful tool for conducting backend testing, particularly for validating the proper functioning of APIs. Postman allows testers to execute API requests independently of the user interface, ensuring seamless interaction with the backend services built using Node.js and MongoDB. During backend testing, testers send HTTP requests with different parameters and payloads to various API endpoints, such as user authentication, donor-recipient registration, organ availability checks, and matching algorithms. The responses received from the APIs are analyzed to verify whether they meet the expected specifications and return data in standard formats like JSON.

Additionally, backend testing involves database validation, ensuring that MongoDB properly stores, retrieves, and updates records without data loss or integrity issues. Various test cases include checking CRUD operations (Create, Read, Update, Delete), security mechanisms (authentication and authorization), and performance under different loads.

By rigorously testing the backend using Postman and other database testing techniques, the Organ Bank system ensures accurate data processing, secure transactions, and seamless functionality. This comprehensive testing approach significantly enhances the overall quality and performance of the system, ensuring a reliable platform for organ donor-recipient management.

5.4.1 Backend Test Cases

Sl.no	Procedure	Expected Output	Actual Output	Status
1	Call getTransplantations with no transplantations in the database	Response status 200 with an empty array	Response status 200 with an empty array	Pass
2	Call getTransplantations with transplantations in the database	Response status 200 with an array of transplantations including donor, recipient, and hospital details	Response status 200 with an array of transplantations including donor, recipient, and hospital details	Pass

Sl.no	Procedure	Expected Output	Actual Output	Status
3	Call getTodayTransplantationData with no transplantations created today	Response status 200 with an empty array	Response status 200 with an empty array	Pass
4	Call getTodayTransplantationData with transplantations created today	Response status 200 with an array of today's transplantations including donor, recipient, and hospital details	Response status 200 with an array of today's transplantations including donor, recipient, and hospital details	Pass
5	Call updateTransplantationResult with a valid match ID and status	Response status 200 with a success message and updated match, request, and donation details	Response status 200 with a success message and updated match, request, and donation details	Pass
6	Call updateTransplantationResult with an invalid match ID	Response status 404 with a "Match not found" message	Response status 404 with a "Match not found" message	Pass
7	Call sendTransplantationScheduleEmail with a valid match ID and email body	Response status 200 with a success message and emails sent to donor and recipient	Response status 200 with a success message and emails sent to donor and recipient	Pass
8	Call sendTransplantationScheduleEmail with an invalid match ID	Response status 404 with a "Match not found" message	Response status 404 with a "Match not found" message	Pass
9	Call getFeedback with no feedback in the database	Response status 200 with an empty array	Response status 200 with an empty array	Pass
10	Call getFeedback with feedback in the database	Response status 200 with an array of feedback including user details	Response status 200 with an array of feedback including user details	Pass

Sl.no	Procedure	Expected Output	Actual Output	Status
11	Call approveMatch with a valid match ID	Response status 200 with a success message and updated match, request, and donation details	Response status 200 with a success message and updated match, request, and donation details	Pass
12	Call approveMatch with an invalid match ID	Response status 404 with a ”Match not found” message	Response status 404 with a ”Match not found” message	Pass
13	Call getPendingMatches with no pending matches in the database	Response status 400 with a ”No pending matches found” message	Response status 400 with a ”No pending matches found” message	Pass
14	Call getPendingMatches with pending matches in the database	Response status 200 with an array of pending matches including donor, recipient, and hospital details	Response status 200 with an array of pending matches including donor, recipient, and hospital details	Pass

Table 5.3: Backend Test Cases

5.5 GUI Testing

For the Organ Bank project, GUI testing ensures that the user interface components function correctly, providing a seamless experience for donors, recipients, and administrators. The graphical user interface is a critical aspect of the system, as it enables users to interact efficiently with the platform for organ registration, matching, and request tracking. GUI testing is structured for different modules of the application to validate usability, responsiveness, and accessibility.

- **Test for Input Fields:** Ensure that all input fields, such as name, age, blood type, organ type, contact details, and medical history, accept the correct format and provide proper validation messages for incorrect entries.
- **Test for Buttons:** The ’Register Donor’ and ’Register Recipient’ buttons should successfully submit the form when valid information is provided and navigate users to a confirmation screen. The ’Search for Matching Organs’ button should display accurate donor-recipient matches.
- **Error Messaging:** Proper error messages should appear when users enter incomplete, invalid, or missing information. For example, an error should be displayed if a user tries

to register without selecting an organ type or providing an invalid blood group.

- **Navigation and Responsiveness:** Ensure that all menu options and links work correctly, leading users to the intended sections. The Organ Bank system should be tested on different screen sizes to confirm that it remains responsive across devices, including desktops, tablets, and mobile phones.
- **Form Validations:** The system should restrict duplicate entries, enforce required fields, and validate special conditions (e.g., ensuring age criteria match organ donation policies).

By conducting comprehensive GUI testing, the Organ Bank project ensures that users can navigate the system intuitively, interact with it efficiently, and complete their tasks without confusion or errors, ultimately enhancing the overall user experience.

5.5.1 GUI Test Cases

Sl.no	Procedure	Expected Output	Actual Output	Status
1	Navigate to the user registration page and enter valid details, then click 'Submit'.	The system securely stores user data and displays a confirmation message.	The system securely stores user data and displays a confirmation message.	Pass
2	Navigate to the hospital registration page, enter details, and click 'Submit'.	The system accurately stores hospital details and confirms registration.	The system accurately stores hospital details and confirms registration.	Pass
3	Navigate to the organ donation page, select an organ, and click 'Submit'.	The system captures donation details and confirms submission.	The system captures donation details and confirms submission.	Pass
4	Navigate to the organ request page, select an organ and hospital, then click 'Submit'.	The system records the request details and provides a confirmation message.	The system records the request details and provides a confirmation message.	Pass

Sl.no	Procedure	Expected Output	Actual Output	Status
5	Navigate to the feedback submission page, enter valid feedback, and click 'Submit'.	The system processes the feedback and displays a confirmation message.	The system processes the feedback and displays a confirmation message.	Pass
6	Navigate to the complaint submission page, enter a valid complaint, and click 'Submit'.	The system records the complaint and confirms submission.	The system records the complaint and confirms submission.	Pass
7	Navigate to the login page, enter valid credentials, and click 'Login'.	The system authenticates the user and redirects them to the appropriate dashboard.	The system authenticates the user and redirects them to the appropriate dashboard.	Pass
8	Log in as an admin, donor, hospital, or recipient and navigate	The system displays relevant content with smooth navigation.	The system displays relevant content with smooth navigation.	Pass
9	Log in as a donor and navigate to the 'View Donation Status' page.	The system displays the donor's current donation status accurately.	The system displays the donor's current donation status accurately.	Pass
10	Log in as a recipient and navigate to the 'View Request Status' page.	The system displays the recipient's organ request status accurately.	The system displays the recipient's organ request status accurately.	Pass

Table 5.4: GUI Test Cases

6 DEPLOYMENT

The deployment phase of the Organ Bank project involves transforming the conceptual design into a fully functional system that can be used by organizations to manage organ donations and transplants effectively. This phase is crucial as it marks the culmination of all the planning, development, and testing efforts, bringing the project to life.

Integrating new components with existing systems can be complex, especially when considering factors such as data formats, APIs, and architecture. Additionally, deciding how much existing code to reuse while maintaining compatibility with the new system requires careful consideration and testing. To address these challenges, the implementation process of the Organ Bank involves meticulous planning and coordination. This includes conducting thorough compatibility checks between the new system and existing infrastructure, identifying potential conflicts or issues, and developing strategies to mitigate them. Additionally, evaluated the feasibility of reusing code from previous projects or third-party libraries, weighing the benefits of efficiency gains against potential risks.

Despite the complexities inherent in implementation, the Organ Bank project benefits from careful planning and preparation, making it relatively easy to deploy and operate. By addressing compatibility challenges, maximizing code reuse, and providing thorough user training, the implementation phase sets the stage for the successful integration of the Organ Bank into organizational workflows. Ultimately, this facilitates efficient organ donation and transplantation management, empowering organizations to allocate resources effectively and achieve their strategic objectives.

The Organ Bank system has successfully transitioned from the conceptual design phase to a fully functional system ready for deployment. The implementation phase has addressed key challenges, ensuring compatibility and code reuse. This sets the stage for the successful integration of the Organ Bank into organizational workflows, facilitating efficient organ donation and transplantation management.

7 GIT HISTORY

Git Repository Organ Bank contains all js files, dart files, reports and three related research papers. It is maintained for systematic way of project presentation and mainly for future reference. Two folders are created for maintaining frontend and backend files seperately.

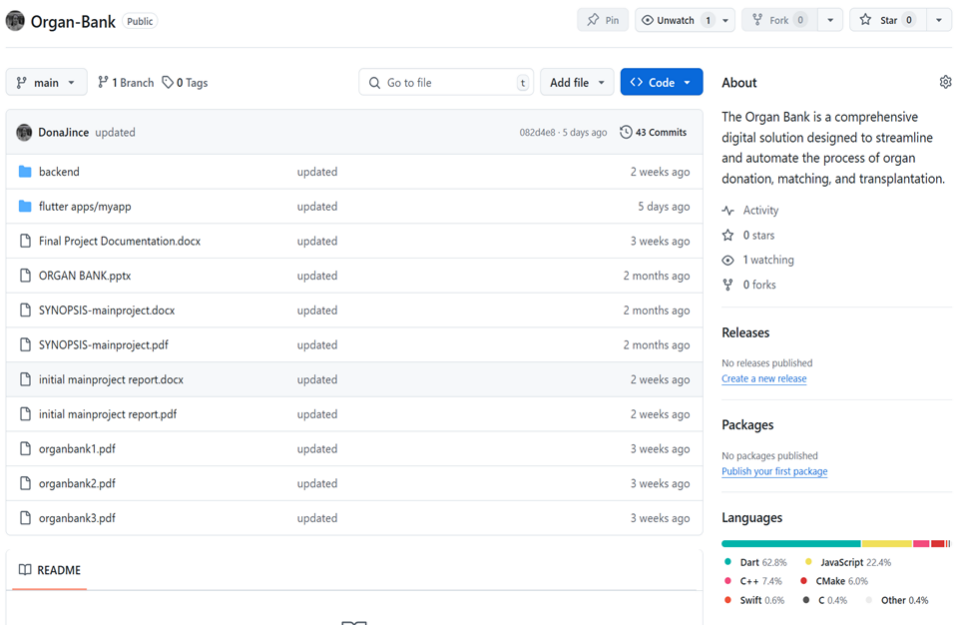


Figure 7.1: Git History

Figure 7.1 shows the git history of Organ Bank.

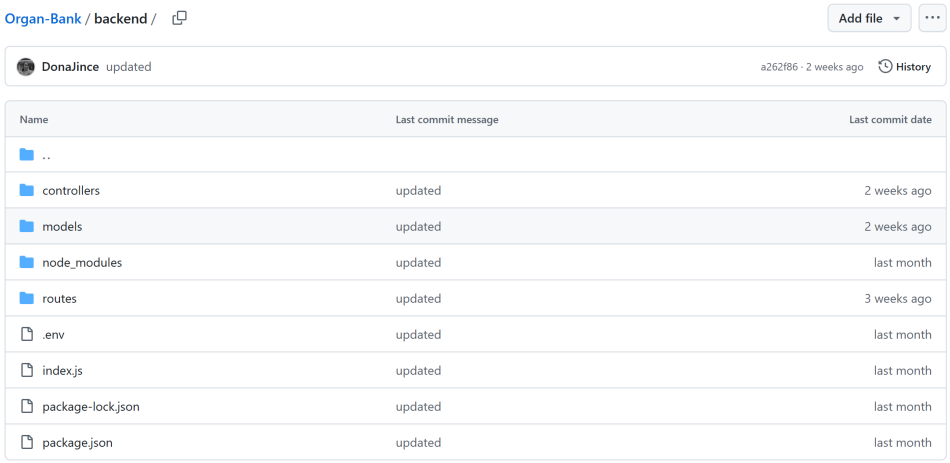


Figure 7.2: Git History of backend

Figure 7.2 shows the git history of Organ Bank backend.

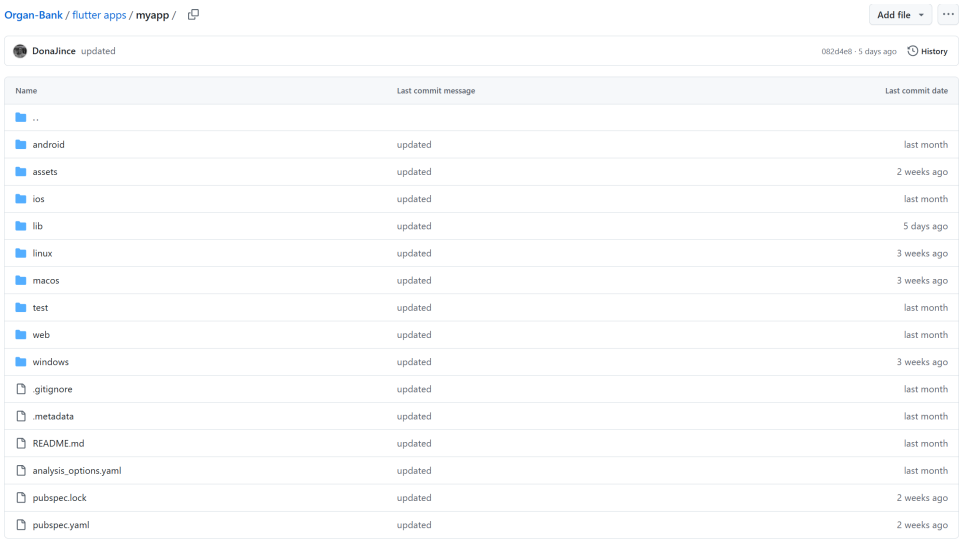


Figure 7.3: Git History of frontend

Figure 7.3 shows the git history of Organ Bank frontend.

8 CONCLUSION

The Organ Bank project revolutionizes the traditional organ donation system by introducing a real-time, automated matching platform that enhances efficiency, transparency, and accessibility. Unlike conventional manual processes that lead to delays and inefficiencies, this system ensures that organ allocation is streamlined and optimized, minimizing wait times and maximizing the chances of successful transplantation. By integrating all key stakeholders—administrators, donors, recipients, and hospitals—the platform facilitates seamless communication and coordination, ultimately improving patient outcomes.

One of the key strengths of the Organ Bank is its automated organ matching system, which leverages medical compatibility criteria to ensure accurate and timely matches between donors and recipients. The inclusion of real-time notifications for transplant schedules, compatibility testing, and hospital coordination ensures that all relevant parties are promptly informed, reducing administrative burdens and enhancing operational efficiency. Additionally, the system incorporates a feedback mechanism to track and improve the donation and transplantation process, ensuring continuous refinement and accountability.

The platform is developed using Flutter, offering a responsive and intuitive user experience across both mobile and web applications. On the backend, Node.js efficiently handles server-side logic and organ matching algorithms, while MongoDB provides a robust, scalable, and secure database system for real-time data storage. The integration of Visual Studio Code as the development environment ensures a smooth development workflow, allowing for quick iterations and improvements. These technologies collectively enable a highly reliable and scalable solution.

By leveraging modern technology, the Organ Bank enhances operational efficiency, reduces human errors, and accelerates critical decision-making processes in organ transplantation. The real-time features and improved communication channels foster trust among stakeholders, ensuring that every organ donation opportunity is effectively utilized.

In conclusion, the Organ Bank is a groundbreaking solution that addresses the shortcomings of traditional organ donation system. Through automation, real-time updates, and seamless integration of key stakeholders, it transforms the way organ donation and transplantation are managed. Moving forward, future enhancements such as AI-driven compatibility matching, blockchain-based data security, and predictive analytics could further improve its effectiveness. With these advancements, the Organ Bank has the potential to save countless lives, making it an indispensable tool in modern healthcare.

9 FUTURE WORK

The Organ Bank project lays a strong foundation for an efficient and automated organ donation and transplantation system. However, there are several opportunities to further enhance its capabilities and effectiveness. One key area of future development is the integration of Artificial Intelligence (AI) and Machine Learning (ML) to improve organ matching accuracy. By analyzing historical transplant data and medical records, AI-driven algorithms can predict compatibility more precisely, reducing the chances of rejection and increasing transplant success rates.

Another significant enhancement could be the adoption of Blockchain technology to improve the security and transparency of donor and recipient data. Blockchain ensures tamper-proof records, preventing fraudulent activities and unauthorized data modifications. It can also facilitate a decentralized and trustless system where hospitals, donors, and recipients can verify information without relying on a central authority. This would further strengthen trust and reliability within the organ donation ecosystem.

The expansion of real-time monitoring and tracking features is another crucial area for improvement. By integrating IoT-based health monitoring devices, hospitals can continuously track donor and recipient health status, ensuring that organs are transplanted at the optimal time. Additionally, GPS-based logistics tracking can be implemented to monitor the transportation of organs, ensuring timely and safe deliveries, especially for critical cases where delays can impact the viability of organs.

Furthermore, enhancing the user experience and accessibility of the platform can make organ donation more widely adopted. Developing a dedicated mobile application with multilingual support and voice-based interaction can help reach a broader audience, including rural and less technologically inclined users. Additionally, integrating the system with government health databases and hospital networks can facilitate automated donor registration and real-time updates on organ availability, further streamlining the process.

In the long run, expanding the Organ Bank project beyond a single region and implementing a nationwide or global organ-sharing network could significantly impact organ donation rates worldwide. Cross-border organ exchange programs, supported by international medical regulations and AI-driven global matching systems, could help connect donors and recipients from different countries, ultimately saving more lives. By continuously innovating and incorporating emerging technologies, the Organ Bank can evolve into a globally recognized, life-saving system that revolutionizes organ transplantation.

10 APPENDIX

10.1 Minimum Software Requirements

To ensure the smooth operation of the Organ Bank system on the client side, the following software requirements must be met:

1. **Operating System**
    - Android 8.0 or later / iOS 12 or later / Windows 10 or later / macOS
2. **Application Runtime**
    - Flutter framework (pre-installed within the app)
    - Dart runtime (bundled with the Flutter app)
3. **Browser Compatibility (For Web Application)**
    - Google Chrome / Mozilla Firefox / Microsoft Edge / Safari (latest versions)
    - JavaScript and cookies enabled
4. **Email Service**
    - Users must have access to valid email address to receive system notifications.
5. **Permissions Required**
    - Internet access

10.2 Minimum Hardware Requirements

1. **Processor (CPU)**
    - Android: Quad-core 1.8 GHz or higher
    - iOS: A11 Bionic chip or newer
    - PC/Laptop: Dual-core 2.0 GHz or higher
2. **Memory (RAM)**
    - Minimum: 4GB RAM
    - Recommended: 8GB RAM for better performance
3. **Storage (SSD/HDD)**
    - Minimum: 500MB of free space for app installation and cache
    - Recommended: 1GB for smooth operation
4. **Graphics & Display**
    - Mobile: 720p resolution or higher
    - PC/Laptop: Full HD (1920×1080) resolution recommended
5. **Internet Connectivity**
    - Stable broadband or mobile network (Wi-Fi/4G/5G)

11 REFERENCES

[1] D. Hawashin, R. Jayaraman, K. Salah, I. Yaqoob, M. C. E. Simsekler, and S. Ellahham, "Blockchain-Based Management for Organ Donation and Transplantation," *IEEE Access*, vol. 10, pp. 59013-59024, 2022. doi: https://doi.org/10.1109/ACCESS.2022.3180008

[2] L. A. Dajim, A. A. Al-Zuraib, S. A. Al-Farras, and B. S. Al-Shahrani, "Organ Donation Decentralized Application Using Blockchain Technology," *2019 IEEE Conference*, 2019.

[3] S.K.Behera, P.K.Sethy, et al., "MongoDB Integration with Python and Node.js, Express.js," *2024 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, 2024. doi: https://doi.org/10.1109/ICAECT60202.2024.10460546

[4] Flutter Official Documentation - https://docs.flutter.dev/

[5] Node JS Official Documentation - https://nodejs.org/docs/latest/api/

[6] MongoDB Official Documentation - https://www.mongodb.com/docs/