INITIAL REPORT

ON

ORGAN BANK

Submitted By:

Dona Jince

MAC23MCA-2024

Faculty Guide:

Prof. Nisha Markose

Project Coordinator:

Prof. Sonia Abraham

ABOUT THE PROJECT

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.

ACTORS AND THEIR ROLES

The Organ Bank involves multiple actors, each playing a significant role in ensuring the efficient and transparent functioning of the system. These actors include Admin, Donor, Recipient, and Hospital.

1. 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.

2. 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.

3. Receipient

- 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.

4. 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.

DESCRIPTION OF MODULES

1. **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.

2. **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.

3. **Request Module**

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

4. **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.

5. **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 receipient.

6. **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.

7. **Feedback and Complaint Management Module**

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

8. **Report Management Module**

   Admin generates reports on transplantation outcomes and success rates.

BUSINESS RULES

- Registrations for Donors, Recipients, and Hospitals must be reviewed and approved by the Admin before granting access to the system.
- Donors must provide explicit consent for organ donation during registration.
- Consent can be revoked at any time before a match is approved by the Admin.
- Receipients must provide valid medical records during registration to justify their request for an organ.
- 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, Receipients, Hospitals) within 24–48 hours of submission.
- Complaints must be addressed by the Admin .

TECHNOLOGY/FRAMEWORK

Frontend Development

- Flutter: Used for creating the user interface for both the mobile app and the web-based system. Ensures cross-platform compatibility, providing a consistent and seamless experience across Android, and web devices. Offers interactive and user-friendly interfaces for donors, recipients, hospitals, and the admin. Built using the Dart programming language for high performance and a smooth user experience.

Backend Development

- Node.js:Powers the backend by providing a fast, scalable, and event-driven environment. Manages server-side logic, including user authentication, organ matching, notifications, and admin approval workflows. Ensures efficient communication between the frontend and database via RESTful APIs.

Database Management

- MongoDB: A NoSQL database used for storing all system data in a flexible, JSON-like format.Manages critical data, and ensures data consistency and supports complex relationships within the system.

PROJECT FEASIBILITY

A project feasibility study is essential to evaluate the technical, economic, and operational viability of the proposed Organ Bank. The study ensures that the project can succeed within the defined scope, resources, and timeline while identifying potential challenges.

1. Technical Feasibility

The Organ Bank is technically feasible due to its use of robust and widely adopted technologies:

- **Frontend**:
    - Built using **Flutter**, a cross-platform framework delivering a seamless experience across Android, iOS, and web platforms.
    - **Dart**, the programming language used in Flutter, ensures smooth animations, responsive UI, and efficient application performance.

- **Backend**:
    - Developed using **Node.js**, a fast, scalable, and event-driven environment ideal for handling server-side logic, organ matching algorithms, and notifications.
    - RESTful APIs ensure efficient communication between the frontend and database.
- **Database**:
    - **MongoDB**, a NoSQL database, manages unstructured and semi-structured data like donor and recipient profiles, organ availability, and transaction logs.
    - MongoDB's flexible schema supports scalability, fast data retrieval, and ensures data consistency.

This technology stack provides a modern, reliable, and efficient solution that is widely supported. It ensures compatibility with various devices, making the system accessible to all users, including donors, recipients, hospitals, and administrators.

2. Economic Feasibility

The Organ Bank is cost-effective due to the following:

- **Open-source Technologies**: The use of **Flutter**, **Node.js**, and **MongoDB** eliminates licensing fees, significantly reducing development costs.
- **Availability of Skilled Developers**: These technologies are widely adopted, making it easy to find skilled professionals at reasonable rates.

- **Automation Benefits**: The organ-matching system, notification module, and reporting features reduce manual administrative costs and time investment.

While initial development, deployment, and maintenance will require investment,the system's ability to reduce operational costs, optimize resource utilization, and improve decision-making provides strong economic justification.

3. Operational Feasibility

The Organ Bank streamlines operations and builds trust through:

- **Frontend**:
    - **Flutter** delivers an intuitive and responsive user interface for all stakeholders.
    - The cross-platform nature ensures a consistent experience across mobile and web platforms.

- **Backend**:
    - **Node.js** efficiently handles backend processes, including user authentication, organ matching, and notifications.
    - The **MongoDB** database ensures fast and reliable data storage and retrieval, supporting complex queries and maintaining data integrity.
- **Automation**:
    - Automates organ matching, notifications, compatibility testing management, and data logging, significantly reducing manual effort and minimizing errors.
- **Trust and Transparency**:
    - The notification and feedback systems enhance transparency and communication, building trust among donors, recipients, hospitals, and administrators.

By automating critical processes and reducing manual interventions, the system optimizes workflows and enhances operational efficiency. Regular updates, skilled developers, and strong security measures will ensure long-term success and scalability.

SYSTEM ENVIRONMENT

Hardware Environment

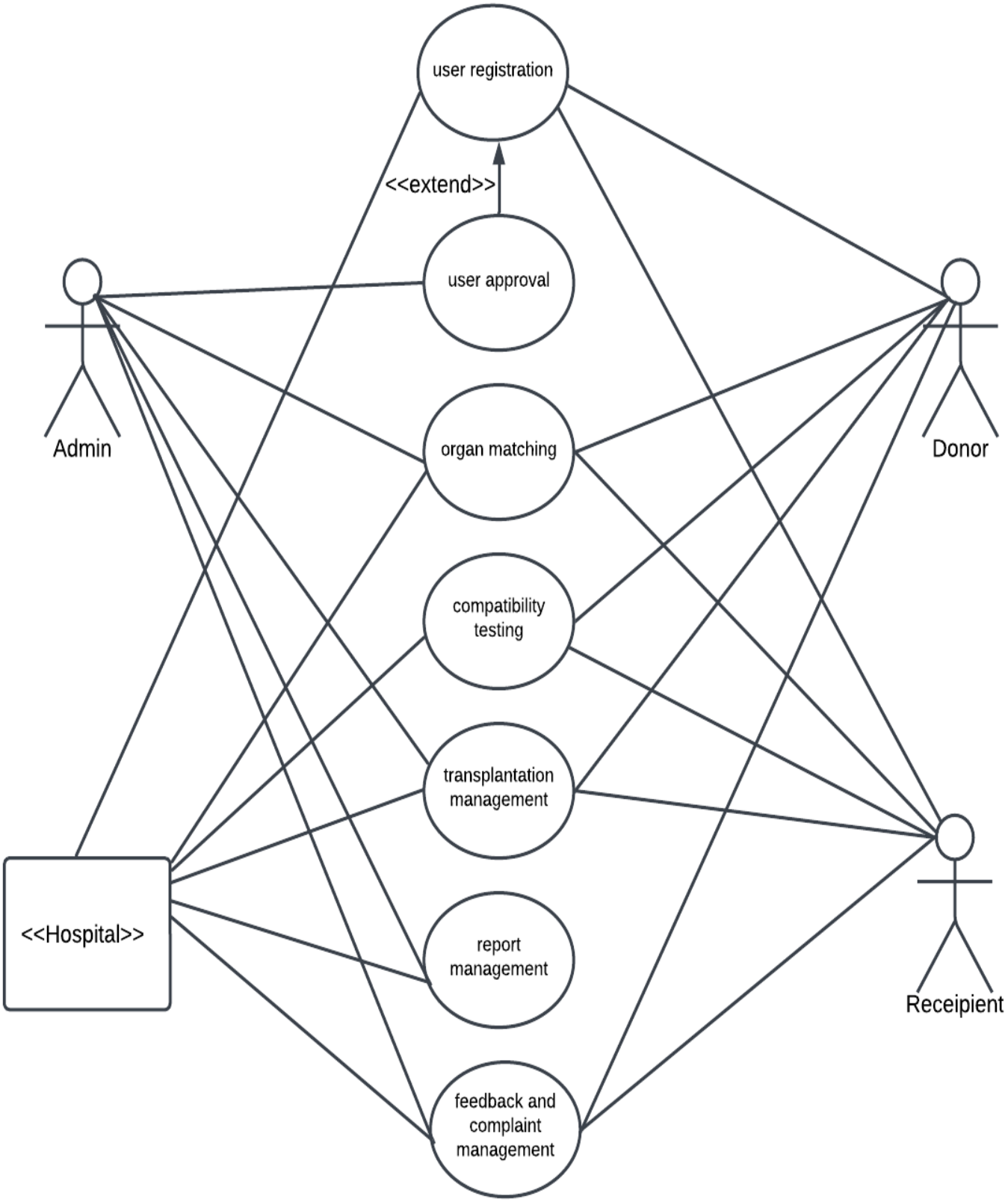
1. Client Devices:
    a. Smartphones and PCs for donors, recipients, and hospital staff.
    b. Minimum: Dual-core processor, 4GB RAM, 500MB storage.
2. Server Requirements:
    a. High-performance server with the following:
        i. Quad-core processor or higher.
        ii. 16GB RAM minimum.
        iii. 1TB storage with SSD for fast data access.
        iv. Reliable internet connectivity for seamless operations.
3. Networking:
    a. Secure, high-speed network for communication between client devices and the server.

Software Environment

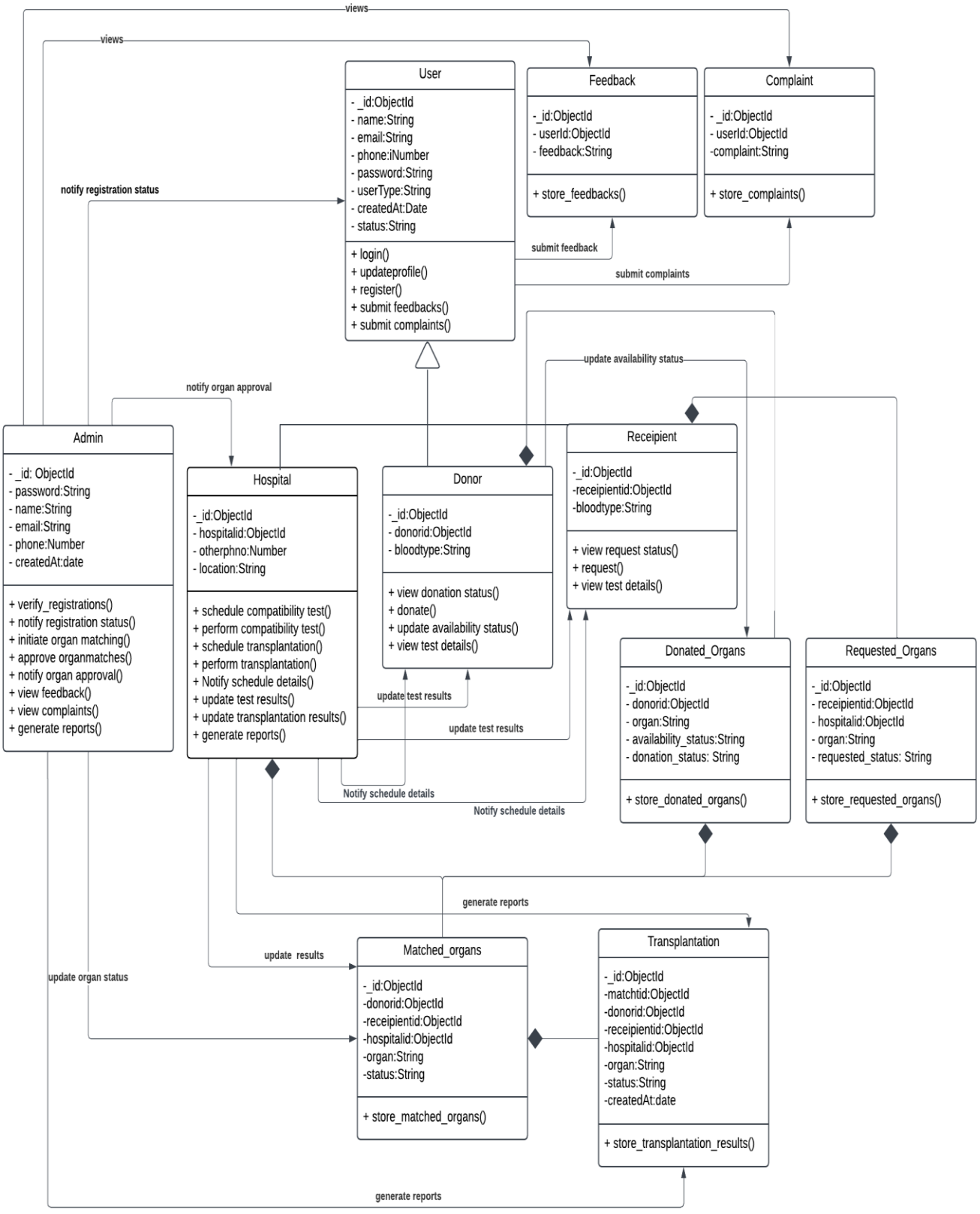
1. Frontend: Flutter (Dart Language) for cross-platform mobile and web application development.
2. Backend: Node.js for efficient server-side logic and API handling.
3. Database: MongoDB for scalable, NoSQL-based data management.
4. Server OS: Linux (Ubuntu) for high performance, reliability, and cost-effectiveness.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

UML Diagrams

1. UseCase Diagram

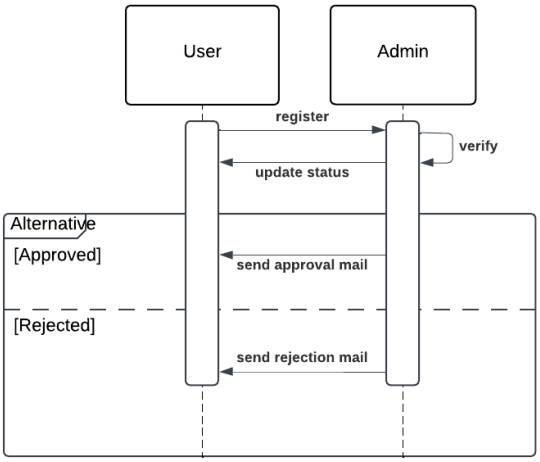


2. Class Diagram

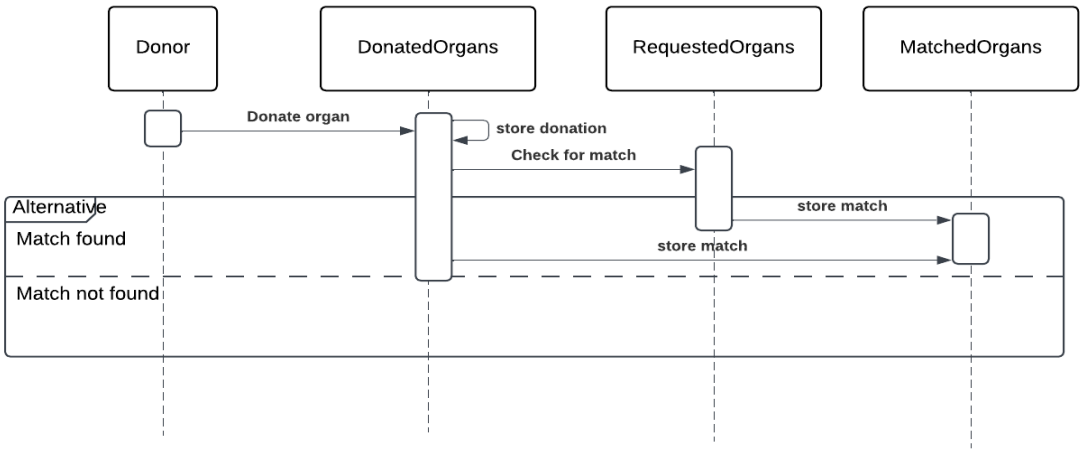


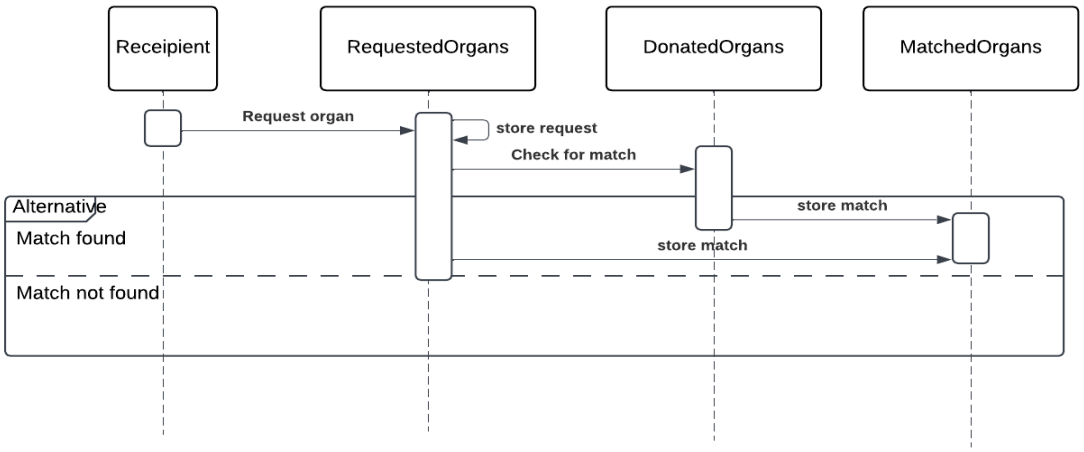
3. Sequence Diagram

3.1 User Registration

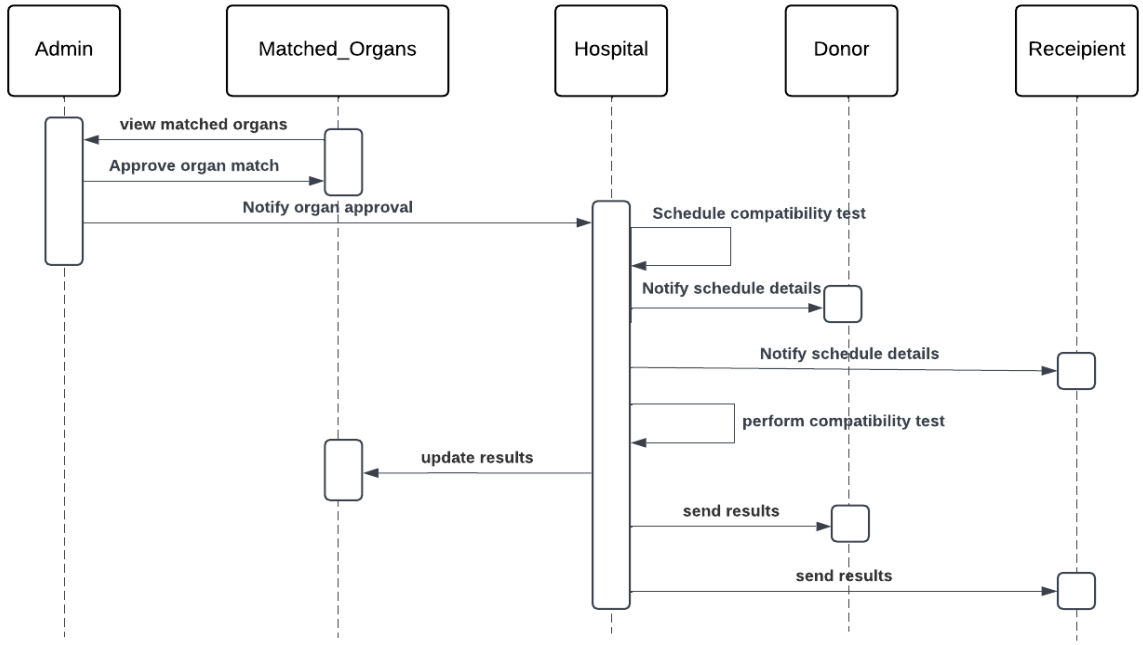


3.2 Organ Matching

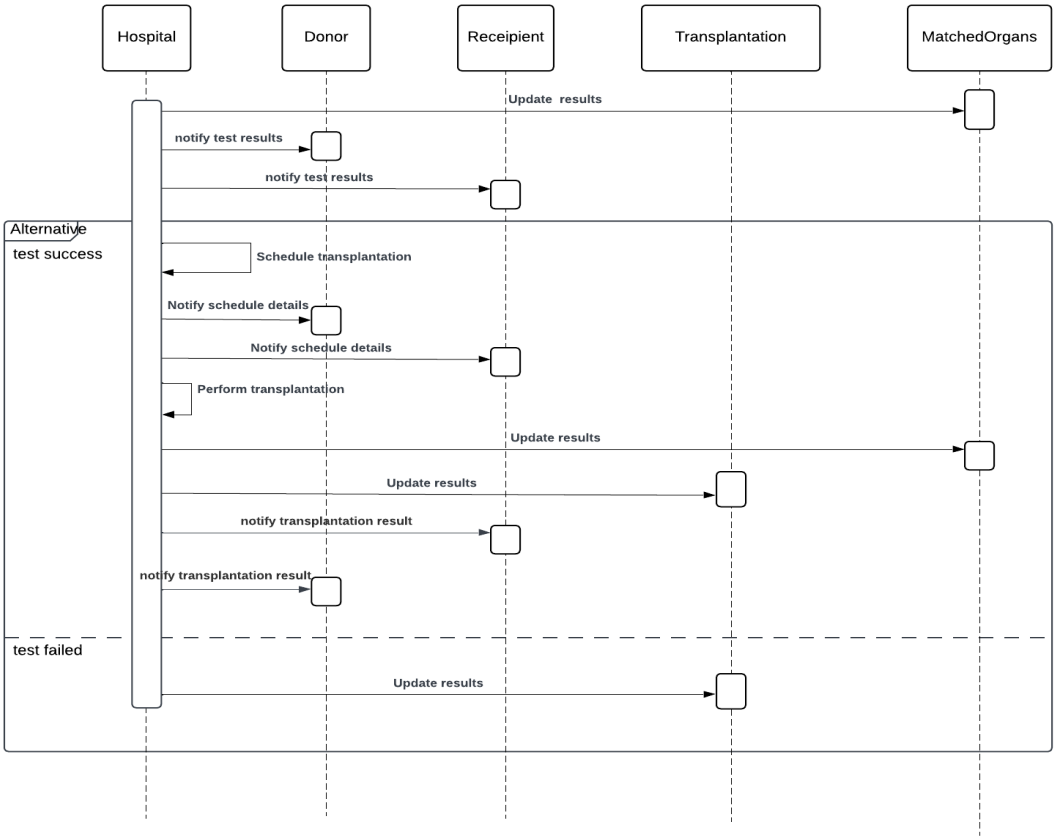




3.3 Compatibility Test

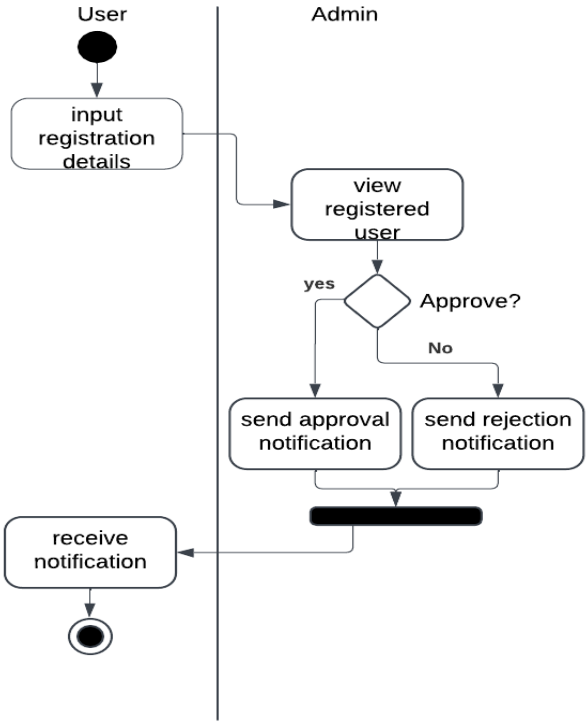


3.4 Transplantation Module

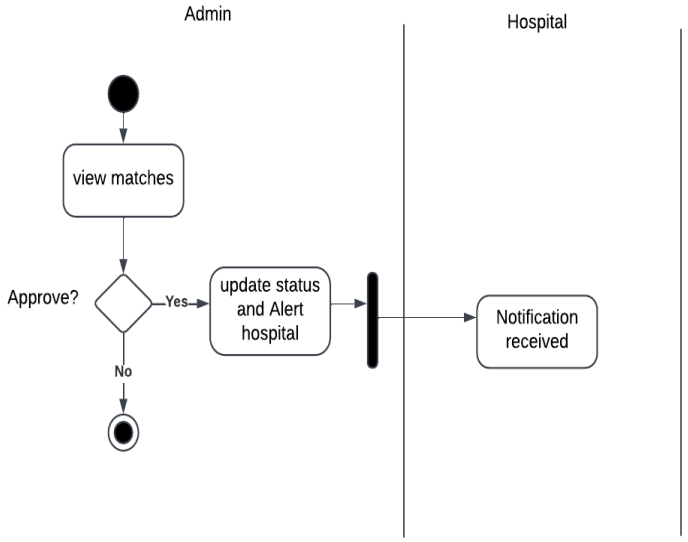


4. Activity Diagram

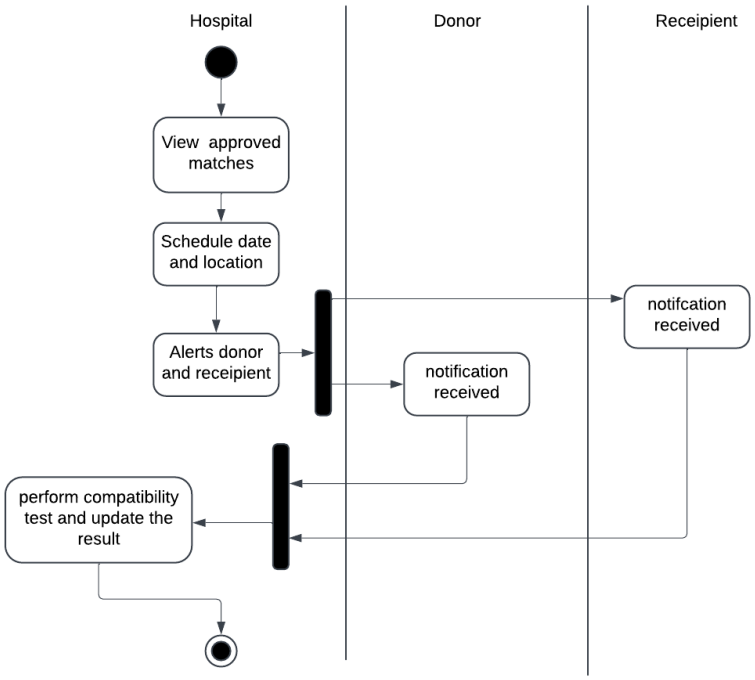
4.1 User Registration



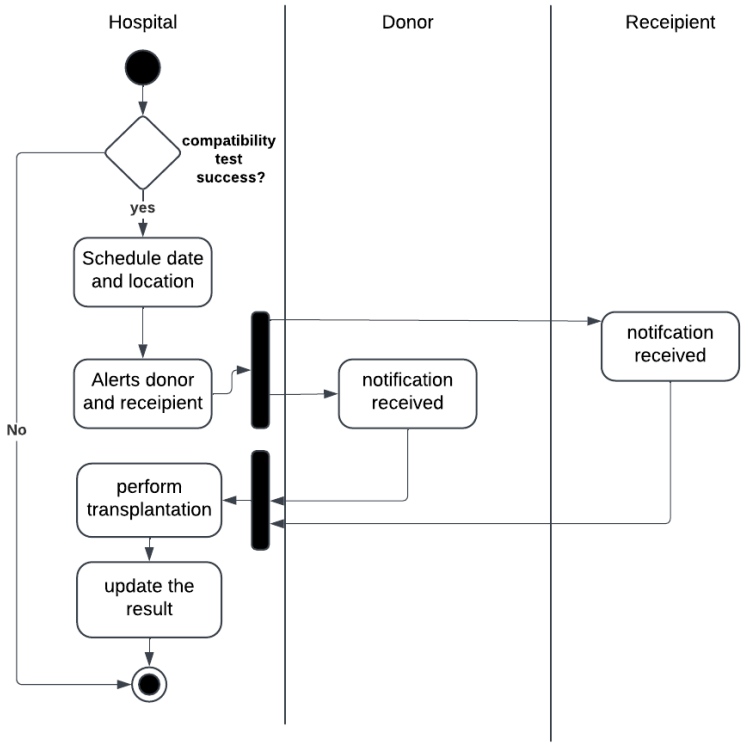
4.2 Match Approval Module



4.3 Compatibility Test Scheduling Module



4.4 Transplantation Scheduling Module



‘

DATABASE DESIGN

1. User

Attributes	Datatype	Constraints	Description
_id	Objectid	Unique	Autogenerated id
name	String		Name of user
email	String		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

2. Donor

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

3. Receipient

Attributes	Datatype	Constraints	Description
_id	Objectid	Unique	Autogenerated id
receipientid	Objectid	ref(User:_id)	Receipient's id
bloodtype	String		Blood group of receipient

4.Admin

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

5. Hospital

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

6.Donated_Organs

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

7. Requested_Organs

Attributes	Datatype	Constraints	Description
_id	Objectid	Unique	Autogenerated id
receipientid	Objectid	ref(User:_id)	receipient's id
hospitalid	Objectid	ref(User:_id)	id of the hospital ,receipient selected during request submission
organ	String		Requested organ
requested_status	String		Status of request

8. Matched_Organs

Attributes	Datatype	Contraints	Description
_id	Objectid	Unique	Autogenerated id
donorid	Objectid	ref(Donated_Organs: donorid)	donor's id
receipientid	Objectid	ref(Requested_Organs: receipientid)	receipient's id
hospitalid	Objectid	ref(Requested_Organs: hospitalid)	hospital's id
organ	String		Matched organ
status	String		Status of matched organ

9. Transplantation

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
receipientid	Objectid	ref(Matched_Organs: receipientid)	receipient's id
hospitalid	Objectid	ref(Matched_Organs: hospitalid)	hospital's id
organ	String		Organ transplanted
status	String		Status of transplantation
createdAt	Date		Transplanted date

10. Feedback

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

11. Complaint

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

SUPPORTING LITERATURE

1. Literature Review

Paper 1: Hawashin, D., Jayaraman, R., Salah, K., Yaqoob, I., Simsekler, M. C. E., & Ellahham, S. (2022). Blockchain-Based management for organ donation and transplantation. *IEEE Access*, *10*, 59013–59025. https://doi.org/10.1109/access.2022.3180008

The paper "Blockchain-Based Management for Organ Donation and Transplantation" by Diana Hawashin et al. explores the integration of blockchain technology 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 authors propose a private Ethereum blockchain-based system that ensures decentralization, security, traceability, and auditability in organ donation management. The use of smart contracts automates key processes, including donor-recipient matching, organ tracking, and real-time updates, ensuring a seamless and tamper-proof workflow.

The study introduces six key algorithms that automate various phases of organ transplantation, including donor registration, organ removal, transportation, and recipient matching. The blockchain-based system enables a ranked matching process that considers factors like blood type, BMI, and age compatibility, thereby ensuring fair and efficient organ allocation. Security analysis using the Oyente tool confirms that the smart contracts are resistant to vulnerabilities like transaction-ordering dependence and re-entrancy attacks. Additionally, a comparative study highlights blockchain's advantages over conventional organ transplant management systems, emphasizing its potential to eliminate fraud, enhance real-time tracking, and improve trust among stakeholders.

The findings of this study align with the Organ Bank project, as both aim to optimize organ donation through technology. While your project is built using Flutter, Node.js, and MongoDB, integrating blockchain technology could further improve data security, transparency, and trust in organ matching and allocation. By adopting blockchain-based smart contracts, the system can prevent data manipulation, enhance organ traceability, and ensure real-time collaboration among hospitals, donors, and recipients. This research provides a strong foundation for future enhancements in your project, making it a more reliable, efficient, and scalable healthcare solution.

Paper 2: Dajim, L. A., Al-Farras, S. A., Al-Shahrani, B. S., Al-Zuraib, A. A., & Mathew, R. M. (2019). Organ Donation decentralized application using blockchain technology. *Organ Donation Decentralized Application Using Blockchain Technology*, 1–4. https://doi.org/10.1109/cais.2019.8769459

The paper "Organ Donation Decentralized Application Using Blockchain Technology" by Lama Abdulwahab Dajim et al. presents a blockchain-based approach to improving organ donation and transplantation systems. Traditional organ donation processes face challenges such as long waiting times, lack of transparency, and security vulnerabilities in centralized systems. The proposed decentralized web application 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 blockchain technology in maintaining data integrity, security, and transparency. By decentralizing organ donation records, the system eliminates risks associated with data manipulation and cyber threats. The authors discuss the use of smart contracts to automate key processes, such as donor registration, recipient matching, and organ tracking, ensuring real-time updates and preventing fraudulent activities. A comparative analysis of centralized, distributed, and decentralized systems underscores blockchain's benefits, particularly in security, speed, scalability, and reliability.

This research is relevant to your Organ Bank project, as both systems aim to enhance organ donation management through technology. While your project is built using Flutter, Node.js, and MongoDB, integrating blockchain can further strengthen data security, improve transparency, and ensure real-time collaboration among donors, recipients, and hospitals. The study supports the idea that blockchain-based organ donation platforms can eliminate inefficiencies, enhance trust among stakeholders, and ultimately save more lives through a streamlined and secure process.

Paper 3: Somasundar, A., Chilakarao, M., Raju, B. R. K., Behera, S. K., Ramana, C. V., & Sethy, P. K. (2024). MongoDB integration with Python and Node.js, Express.js. *MongoDB Integration With Python and Node.js, Express.js*, 1–5. https://doi.org/10.1109/icaect60202.2024.10469546

The paper "MongoDB Integration with Python and Node.js, Express.js" by Santi Kumari Behera et al. presents a comprehensive study on integrating MongoDB, a widely used NoSQL database, with Python and Node.js. MongoDB's scalability, flexibility, and document-centric structure make it an excellent choice for handling semi-structured and unstructured data in modern web applications. The authors examine the key drivers and APIs, particularly PyMongo for Python and Mongoose for Node.js, demonstrating their role in facilitating seamless database interactions. They also explore how MongoDB integrates with the MEAN stack (MongoDB, Express.js, Angular, and Node.js), a widely adopted web development framework.

A major focus of the study is indexing and query optimization in MongoDB, where techniques such as B-trees and hash tables are employed to improve query performance. The authors highlight the advantages of aggregation pipelines and MapReduce for efficient data processing, making MongoDB an ideal choice for big data analytics and real-time applications. Additionally, they address security vulnerabilities, such as unauthorized access and injection attacks, recommending best practices like authentication, encryption, and role-based access control to enhance database security.

This study is highly relevant to your Organ Bank project, as it validates the scalability, efficiency, and security of MongoDB for handling large volumes of donor-recipient data, real-time organ matching, and hospital coordination. The findings suggest that using Mongoose for Node.js and PyMongo for Python can streamline database interactions while ensuring data integrity and optimal query performance. By adopting the recommended security measures, your system can prevent unauthorized data manipulation and ensure reliable organ transplantation management.

2. Literature 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.

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.