A SOPHISTICATED PLATFORM OF BLOOD DONATION WITH ADVANCED WEB TECHNOLOGIES

Submitted in partial fulfillment of the
requirements for the degree

of

Bachelor of Science in Information and Communication Engineering

by

Kamrul Hasan Ridoy
201-50-002
Mst Ayesha Khatun
201-50-020

Under the Supervision of
Dipto Biswas
Lecturer
Department of ICE

Daffodil International University

September 2024

APPROVAL

This is to certify that the entitled "**A Sophisticated Platform of Blood Donation with Advanced Web Technologies"**, submitted by Kamrul Hasan Ridoy (ID: 201-50-002) and Mst Ayesha Khatun (ID: 201-50-020) are undergraduate students of the Department of ICE has been examined. Upon recommendation by the examination committee, we hereby accord our approval of it as the presented work and submitted report fulfill the requirements for its acceptance in partial fulfillment for the degree of *Bachelor of Science in Information and Communication Engineering.*

BOARD OF EXAMINERS

Md. Taslim Arefin Associate Professor and Head Department of ICE Daffodil Internation University	**Chairman**
Prof. Dr. A.K.M. Fazlul Haque Professor Department of ICE Daffodil Internation University	**Internal Examiner**
Ms. Tasnuva Ali Assistant Professor Department of ICE Daffodil Internation University	**Internal Examiner**
Dr. Engr. M. Quamruzzaman Professor and Head Department of EEE World University of Bangladesh	**External Examiner**

DECLARATION OF AUTHORSHIP

Project Title	A Sophisticated Platform of Blood Donation with Advanced Web Technologies
Authors	*kamrul Hasan Ridoy, Mst Ayesha Khatun*
Supervisor	Dipto Biswas, Lecturer, Department of ICE, DIU

We, Kamrul Hasan Ridoy and Mst Ayesha Khatun, hereby declare that this capstone project titled "**A Sophisticated Platform of Blood Donation with Advanced Web Technologies"** is entirely our own work unless otherwise referenced or acknowledged. The content of this project is the result of our own research and efforts, and we have complied with all ethical guidelines and academic standards.

We are aware of the consequences of academic dishonesty and understand that any violation of ethical standards in this project may lead to disciplinary actions as defined by Daffodil International University's policies.

Kamrul Hasan Ridoy

Kamrul Hasan Ridoy, 201-50-002

Mst Ayesha Khatun

Mst Ayesha Khatun, 201-50-020

8/9/24

Supervisor

Dipto Biswas
Lecturer
Department of ICE

ACKNOWLEDGEMENT

First and foremost, all thanks go to the Almighty Allah, merciful, whose countless showers of blessings and His guidance let us accomplish this task successfully, which was not possible but for His grace and help.

Our supervisor, **Dipto Biswas, Lecturer**, Department of Information and Communication Engineering, Daffodil International University, Birulia, Savar, Dhaka, has given us invaluable advice, mentoring, and support for which we are very much indebted. In determining the course and success of our work, his profound knowledge and keen interest in "Software Engineering and Information Systems" have been very helpful.

We also want to express my heartiest gratitude to the **Head of the department, Md. Taslim Arefin, Department of ICE,** for your kind help finishing our project. We also want to express our deep gratitude to other faculty members and mentors who have contributed immeasurably with their knowledge and support, advising on the proper application of the several technologies involved in the project.

We would also like to take this opportunity to greatly appreciate our family and friends for their support and trust. Their encouragement has been with us all along this journey and also kept us on course, particularly at those crucial times.

Last but not least, we would like to include a special thank you to our fellow students and other members in this group. Their cooperation, feedback, and support have contributed to the completion of this project, both enriching and rewarding.

ABSTRACT

The present blood donation process is inefficient and fragmented, which more often than not delays finding eligible donors for recipients, especially in emergency cases. The current process suffers from huge problems of communication between donors and recipients and maintenance of the 90-day time gap of donations, therefore impacting the overall efficiency of the system. The aim of this project is to create a web application that will connect blood seekers and blood donors to contact directly as end-to-end users. A single-page application design has been adopted. The design consists of all the donors registered in the application, real-time notification, searching of donors based on both blood group and area, map for the clarification of donor's given location required Google map integration, sending requests to the donor, accepting requests, and updating database based on the different actions performed by users. A blood donation web application has the potential to revolutionize blood donation by improving efficiency, reducing the time to look for a donor, and promoting blood donation while making donors follow the rules and ethics of donating blood. However, there are some challenges to consider such as potential technical issues and the need for the proper maintenance, and upkeep to optimal performance of the application. Overall, the web applications benefits outweigh the drawbacks, and further development and research are needed to improve the application and make it more accessible to a wider range of the country.

Keywords: Web Application; React JS; Express JS; MongoDB; Node JS; Notification; Map; Blood Donation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

DFD	Data Flow Diagram
API	Application Programming Interface
IDE	Integrated Development Environment
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
HTTP	Hypertext Transfer Protocol
REST	Representational State Transfer
BSON	Binary Javascript Object Notation
VAPID	Voluntary Application Server Identification
JSON	JavaScript Object Notation
IDE	Integrated Development Environment

Chapter 1: Introduction

1.1 Introduction

In recent years, there has been a rapid increase in blood donations and blood seeker numbers. Every year patients with various diseases, from accidents need blood. Some patients need blood on a regular basis, some need it in emergency surgery. Various private, and public organizations work to provide safe blood to save so many lives. But there are many cases where a donor cannot be found at a crucial time, which leads to difficulty and even to death of a patient. There are also many cases where a donor gives blood without following the restriction period for donating blood which even causes major serious issues. One such solution to reduce the issues is a web application for blood donations. With the use of the react.js library, node.js, express.js, and MongoDB database, this project focuses on presenting a design and implementation of one such web application. The blood donation web application is designed for users where it needs to be functional, convenient, and in an emergency where the user seeks for donor.

1.2 Project Background

Blood donation is a life-saving experience and also an honorable practice that saves millions of lives every year. Every year a number of people die due to the lack of blood transfusion in time. Patients from various illnesses or patients from accidents need blood. Some patients need blood on a regular basis, patients with thalassemia, and cancer need blood regularly. In 1950, the Dhaka Medical College Hospital in Bangladesh started offering blood transfusion services. By 2000, 47% of donated blood in Bangladesh came from professional donors, who made up the majority of blood donors[1]. It was widely acknowledged that there was a chance of contamination in the supply and that voluntary donors were required. According to a 2011 estimate, only 25% of the needed units of blood needed yearly are donated voluntarily, 20%–25% are provided by paid donors, and 50–55% are given once for a particular patient[2].

Numerous studies have revealed that patients require blood in a variety of situations, including those involving deliveries, surgeries, kidney disorders, thalassemia, anemia, and various accident-related injuries[2]. Another crucial issue to be concerned about is the fact that many applicants state that handling each bag of blood takes them between 19 and 24 hours. Additionally, the blood seekers say they obtain blood via blood banks, several referrals, blood donation organizations, and families. Most blood donors say they have to go a considerable distance in order to donate[3]. Blood donors deal with a number of issues both before and during the donation process, including transportation issues, handling emergency situations, health-related issues, and a drawn-out cross-checking procedure. Once more, a lot of people wish to voluntarily donate blood. However, because there is no way for them to get in touch with blood donors, they are unable to contribute[3].

The goal of this project is to create a web application that effectively connects blood donors and seekers, streamlining the blood donation procedure. In the current environment, where a prompt and adequate blood supply can impact emergency medical care, the project's significance is paramount. With a streamlined approach to managing and contacting potential

donors, hospitals, blood banks, and humanitarian organizations will gain from this system. Enhanced donor participation, decreased shortages, and quicker reaction times in emergency situations are some of the possible effects.

1.3 Project Definition

1.3.1 Problem Statement

The main issue this project attempts to solve is the present blood donation process's inefficiency and lack of connectivity, which frequently causes delays between blood donors and those in need. Finding eligible blood donors for recipients in need is a significant problem, particularly in emergency cases when time is of importance. For patients in need of emergency blood transfusions, traditional methods of coordinating donors and recipients such as phone calls or emails are sometimes unreliable and slow, which can cause delays and even fatal outcomes. Also donating blood without following the code of blood donation, like a person cannot donate between the last donation which is 90 days period. In order to solve these problems, this project will create a web-based application that will expedite the blood donation procedure, provide better communication between donors and receivers, and increase the system's overall effectiveness.

1.3.2 Context and Scope

This project's scope includes multiple important components:

- **Donor Management:** Managing donor information, and contact details. Searching donors based on blood type and area.
- **Recipient Requests:** Requesting donors based on criteria on blood type and location accurately that matches with available donors.
- **Communication:** Enhancing communication between donors and recipients, including notifications and updates on donation requests.
- **Donation Frequency Compliance:** Implementing a system to ensure that donors comply with the regulation that they cannot donate blood more frequently than once every 90 days, to safeguard donor health.
- **Data Security:** Ensuring the privacy and security of all user data.

This project does not cover aspects such as medical testing of blood donations, physical logistics of blood transport, or integration with hospital management systems.

1.3.3 Significance and Implications

The blood donation process can be made much more efficient by addressing inefficiencies and connectivity issues. By guaranteeing that blood is accessible when and where it is most needed, this might possibly save lives. In addition, the project contributes to the preservation of donor health and guarantees a consistent and secure blood supply by enforcing adherence to donation frequency regulations. This initiative intends to improve patient outcomes and the overall efficacy of blood donation operations by building a more integrated and efficient system.

1.3.4 Survey

We looked into the targeted users of our project by conducting a survey amongst individuals that may provide us with better insight into the situation and thoughts of blood donors and seekers. The survey report is as follows:

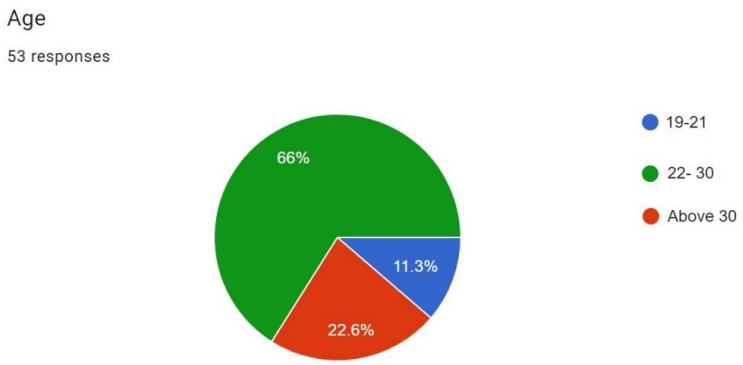

Fig.1 The pie chart visually represents the distribution of age groups

From the information given in the pie chart Fig.1, the most popular age rank taking this survey was 22-30 years old, the other age ranges did take the survey however 22-30 was by far the most popular category in this blood donation survey.

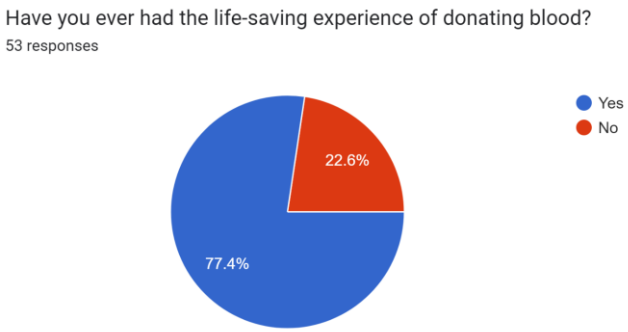

Fig.2 This pie chart in the survey report displays the ratio of donating blood.

The pie chart Fig.2 clearly indicates that the majority of people have voted "yes," demonstrating that a significant number of individuals have experience with blood donation.

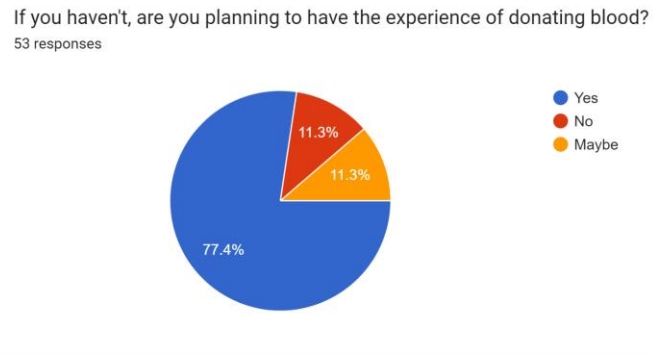

Fig.3 This pie chart represents individuals who plan to donate blood in the future.

From the information shown in this pie chart Fig.3, the percentage distribution of individuals who have expressed their intention to donate blood in the future.

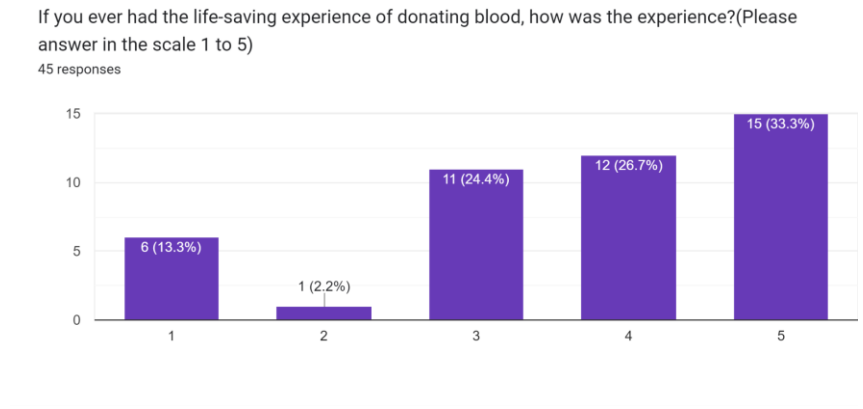

Fig.4 The column chart displays the rating of blood donation experience

The evidence from this bar chart Fig.4 included in the comprehensive survey report visually presents the ratings and feedback provided by blood donors about their donation experience.

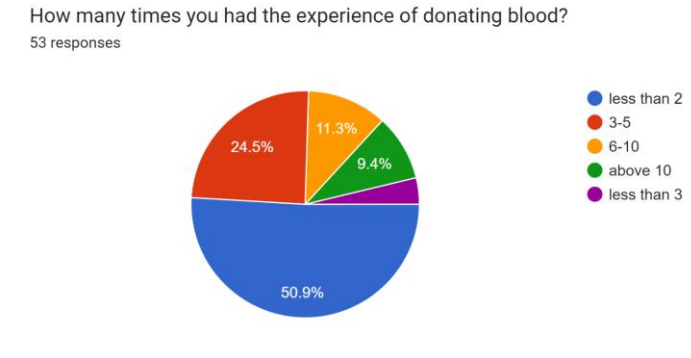

Fig.5 The pie chart displays the number of times they donated blood.

The pie chart Fig.5 provides a visual breakdown of the frequency of blood donations by individuals. Each section of the chart represents the number of times a person has donated blood.

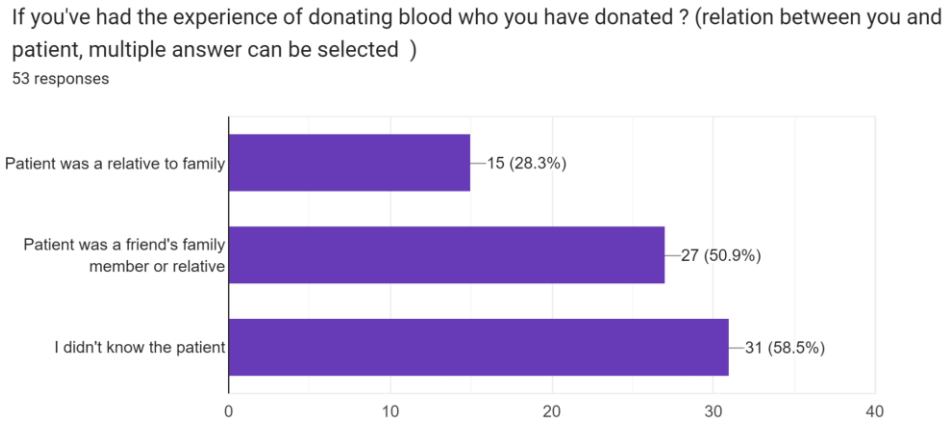

Fig.6 Here the bar chart represents the relationship between the donor and patient

The graph displays the survey results Fig.6 regarding the relationship between the donor and the patient. It is clear from the graph chart that the majority of the connections are recorded as unknown.

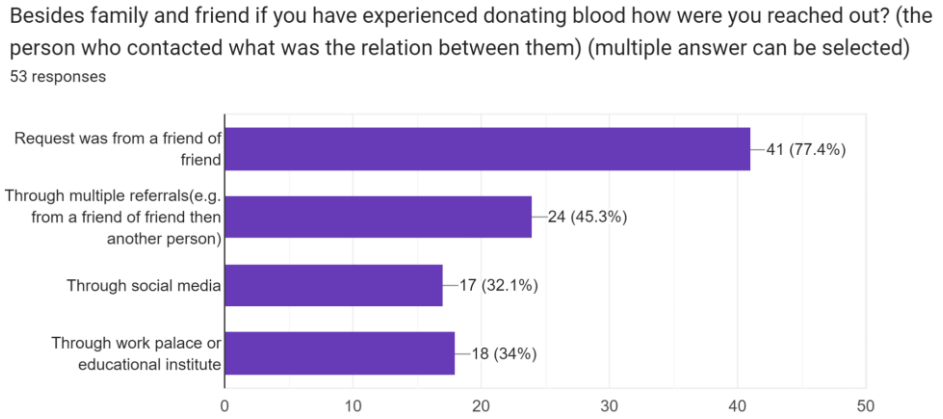

Fig.7 The bar chart represents the method of contacting for blood donation requests.

Based on the data in the graph chart Fig.7, it is evident that the majority of blood donations come from sources other than family members. Additionally, most blood requests are made by friends of friends. Besides, it also helps with social media and educational institutes.

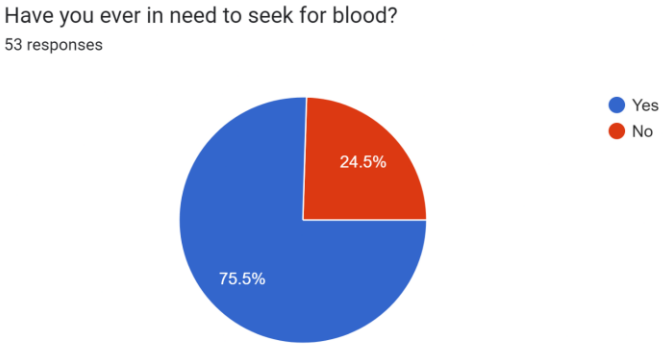

Fig.8 This pie chart is about the experience of seeking blood.

From the information in the above pie chart, Fig.8 we can say that most of the people were in need of blood. The ratio says that the majority is 75.5% from our survey.

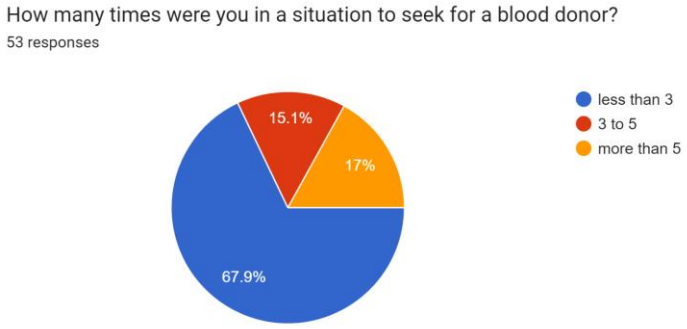

Fig.9 This pie chart represents the number of times seeking blood donors

The above pie chart Fig.9 represents people seeking a donor and the number of times they were seeking them.

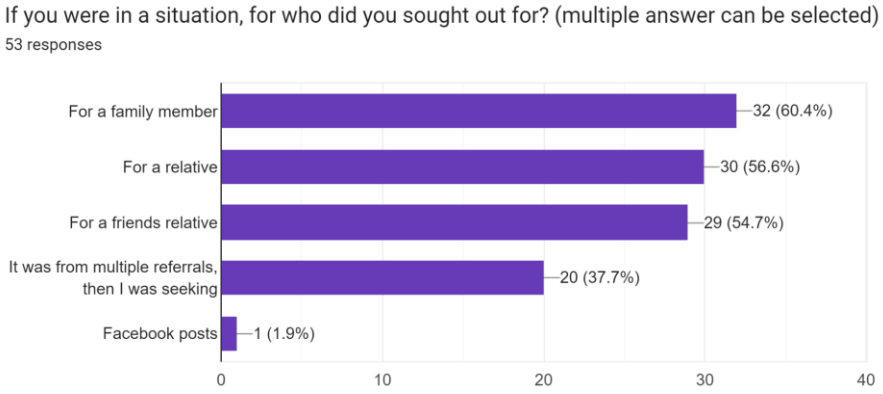

Fig.10 The bar chart shows the recipient of the blood donor search.

In the pie chart, Fig.10 people were asked questions on who were they seeking for blood and their relation to them. The majority was for their family or relatives. But there is also a major part where they sought donors they don't even know. They were contacted by multiple referrals.

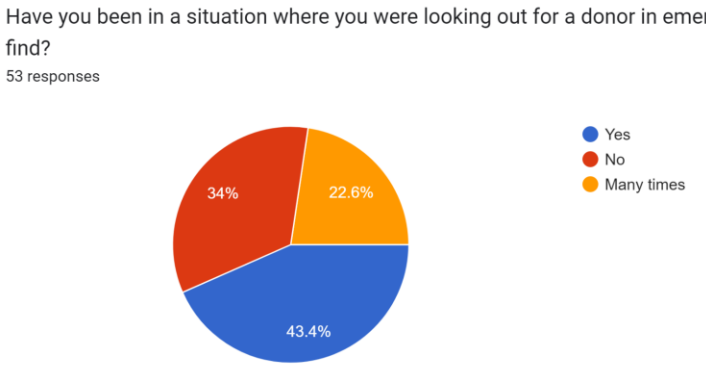

Fig.11 This pie chart is about the difficulty in finding blood donors

From the information given in this pie chart Fig.11, there is a significant challenge in locating blood donors during emergencies, and it is clear in this survey that the large areas voted 'yes' which is included in the problem.

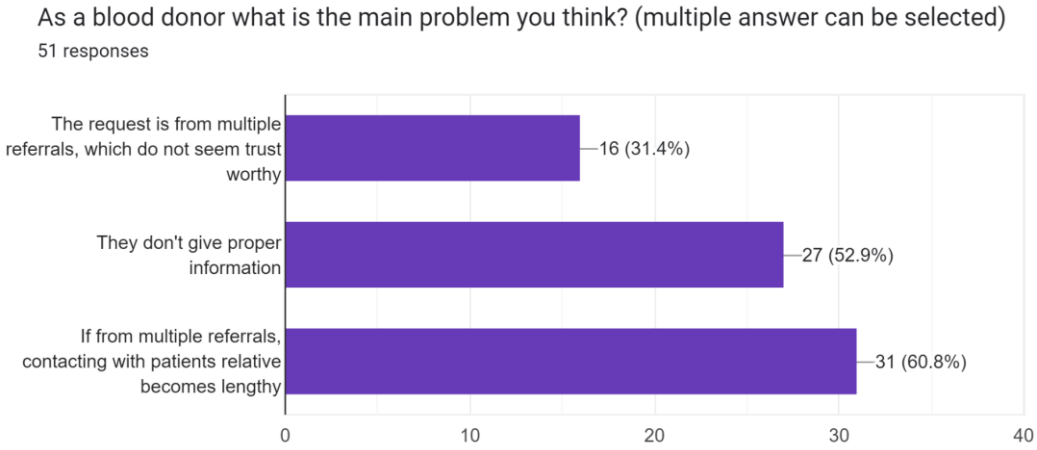

Fig.12 The bar chart is about the main problems identified by blood donors.

The bar chart Fig.12 from the survey report highlights that a significant number of people are encountering challenges, with the majority facing difficulties with multiple referrals and experiencing prolonged interactions with patients' relatives. Sometimes they provide wrong information or trust issues are created.

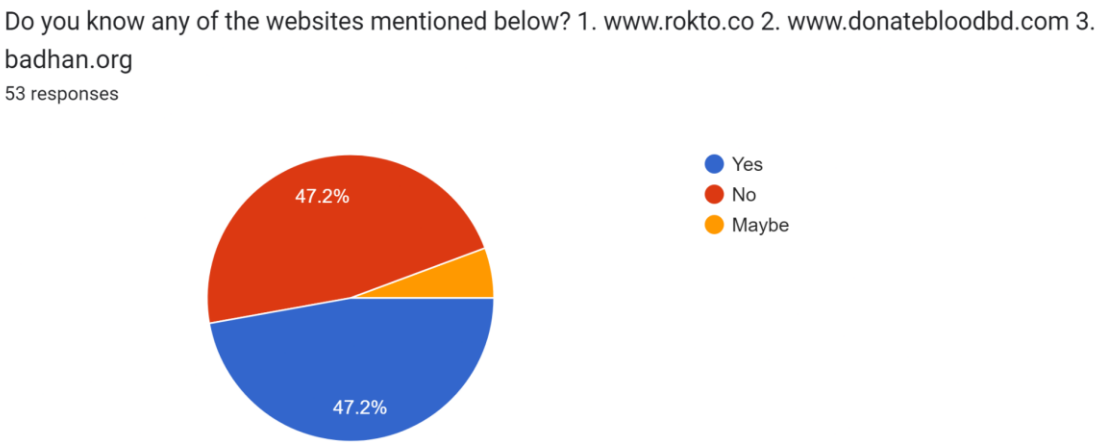

Fig.13 This pie chart represents the awareness of blood donation websites.

As shown in the pie chart Fig.13 the awareness of blood donation websites among the surveyed individuals. Here this gives the idea that most people are aware of those websites.

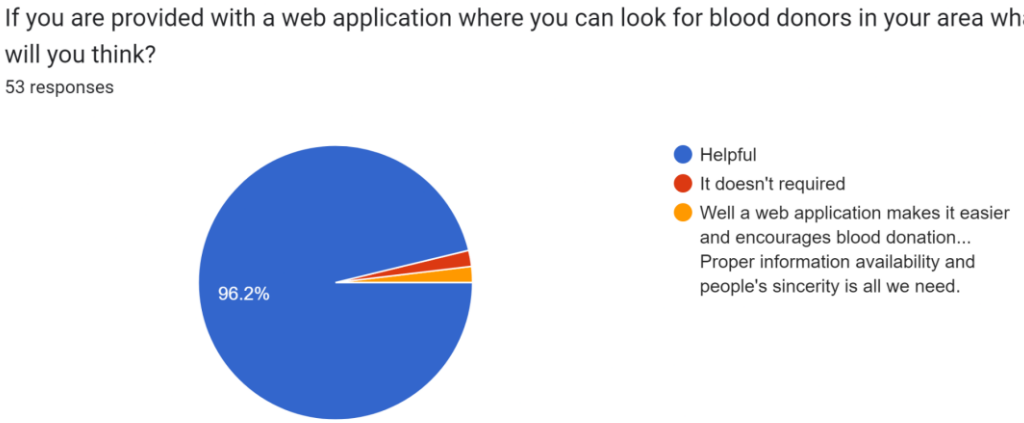

Fig.14 The pie chart is about interests in using web applications

In this chart, Fig.14 the majority of people said that they are interested in using web applications for finding blood donors. The information from this survey shows that most people are interested in using web apps to find blood donors. This important development is significant proof that a web application should be created specifically for this purpose. However, in conclusion, it is evident that often the donation of blood and seeking blood donors becomes distressful. Hence blood donors and seekers tend to lean toward more easy alternative way. In this regard, a web application will become the bridge between the seekers and donors. This project will help people who are looking for a donor to contact the donor directly near the seeker's location without creating multiple referrals.

1.4 Project Objectives

1.4.1 Objectives

1. Create a System for Effective Database Management of Blood Donor and Seeker

- To handle and keep track of user data, such as blood type, contact information, and donation history, establish a centralized database.
- Easy access and updating of user records and to register new users.

2. Procedure for Recipient Requests

- Implementation of a user-friendly interface for seekers to request blood donations quickly based on their criteria.
- Creating and applying an algorithm to match seekers with suitable donors based on blood type and location they need, ensuring timely and effective responses.

3. Enhance Communication Between Donors and Recipients

- Implementing a web browser notification system to inform donors of requests for donations and to inform seekers of the progress of their requests.

4. Integration of Geolocation Feature

- Utilize geolocation APIs to allow recipients to ensure the donor's location which will display the location in the donor's information page.

5. Ensure Compliance with Donation Frequency Regulations

- A system to track donation dates and enforce a 90-day interval between donations to protect donor health. During this interval, seekers won't be able to send requests.

6. Improve Overall Effectiveness of Blood Donation Efforts

- Encourage more donors to participate and to stay involved by providing an effective, user-friendly platform.
- In the end, improve patient outcomes and save lives by expediting and enhancing blood supply.

1.5 Project Specification

The project specification document describes all features, capabilities, behavior, and user interaction with the software system. It explains what functions will be offered by the software, the proper conduct for each, and the expected results of user interaction. The key specifications necessary to ensure the blood donation web application project's success are depicted in the TABLE I.

TABLE I. Blood Donation Web Application Key Specifications

Category	Technologies, Tools, Details and Features
User Interface(Frontend)	React.js for dynamic UI rendering
	Responsive design using Tailwind CSS
	Integration with Google Maps API for geolocation visualization
Backend	Node.js and Express.js for server-side logic
	RESTful API design for frontend-backend communication
Web Push	Implementation of web push notifications
	Service Worker API for background processes
	Integration with VAPID keys for secure communication
Database	MongoDB for storing data. It’s a NoSQL database.
	Provides high performance, high availability, and easy scalability
Geolocation API	Collects the user's latitude and longitude
	Google Maps integration for displaying donor locations.
Technical Needs	Centralized database using MongoDB
	Geolocation APIs for mapping donor locations
	Web push notification service and RESTful API for frontend-backend communication
Performance Expectations	Real-time data updates and synchronization and high volume handling.
Functionalities	User authentication and secure login
	Donor registration and profile management
	Recipient request and donor response

	Notification system and Geolocation for donor mapping
	System to track donation date and interval of 90 days
Firebase	To authenticate registration, login of users, and deployment
Compatibility	Support major web browsers such as Chrome, Firefox, Safari, and Edge.
	Responsive across devices such as desktop, and mobile
Deployment	Firebase for frontend and vercel for backend deployment.
Testing	Jest and React Testing Library for frontend testing, and Rapid API for backend testing
Version Control	Git and GitHub for version control and collaboration

1.6 Conclusion

The inefficiency and lack of connectedness in the present blood donation procedure, which frequently causes delays between blood donors and recipients, is the main problem this initiative attempts to address. By providing a simple and effective platform for donors and receivers to connect, our blood donation web application has effectively addressed these issues. With the use of modern web technologies, the application ensures that eligible donors are matched with those in need, which reduces the time taken to facilitate a blood donation significantly which is very crucial in emergencies where timely blood transfusions can save lives.

The application has essential features such as finding donors based on blood type and area, Firebase authentication for secure user authentication, and MongoDB for data storage. Additionally, the web push notification system ensures that donors are always informed about the status of their requests and geolocation integration for the authenticity of the donor's given location. Our application also addresses the importance of the 90-day interval between donations, ensuring that donors follow the necessary guidelines and maintain their health while contributing to the blood supply.

In summary, the development of our blood donation web application represents a significant step forward in enhancing and empowering the efficiency and effectiveness of the blood donation process. This initiative has the potential to save lives and enhance the blood donation process as a whole by bridging the gap between donors and receivers and guaranteeing quickness and trustworthy contact.

Chapter 2: Software Design Approach

2.1 Introduction

A Sophisticated Platform of Blood Donation with Advanced Web Technologies is a new innovative approach to make end to end interaction between blood seekers and donors. With the increase in blood transfusion practice and the need for emergency cases, the development of blood donation web application has become a necessary solution.

The main goal of the blood donation web application is to provide a hassle-free and faster connection between blood donor and seeker. This is achieved by developing real-time sending requests, notifications to donors, and responses from donors. Also for seekers' reliability to donors geolocation service and Google map have been integrated.

2.2 Multiple Design Approach

After examining a number of designs, we conducted research on their features to ensure that they met our needs. Our main goal is to create a web application that will make it easier to find donors by allowing requests for blood to be sent directly to donors.

Existing Design Approach 1: Development and Implementation of a Web-Based Blood Bank Management System for Efficient Blood Donation and Distribution[4] a paper by Gollapelly Manika, Venkata Sridhar Reddy, Kammampati SaiVamshi, Ms.Prashanthi proposes a solution for the improvement of blood donation. The approach applies to blood banks. The approach is a seeker will look for available donors in the data sheet and send the request to the blood bank, then the blood bank will contact the donor. Here is the block diagram of the existing design approach 1.

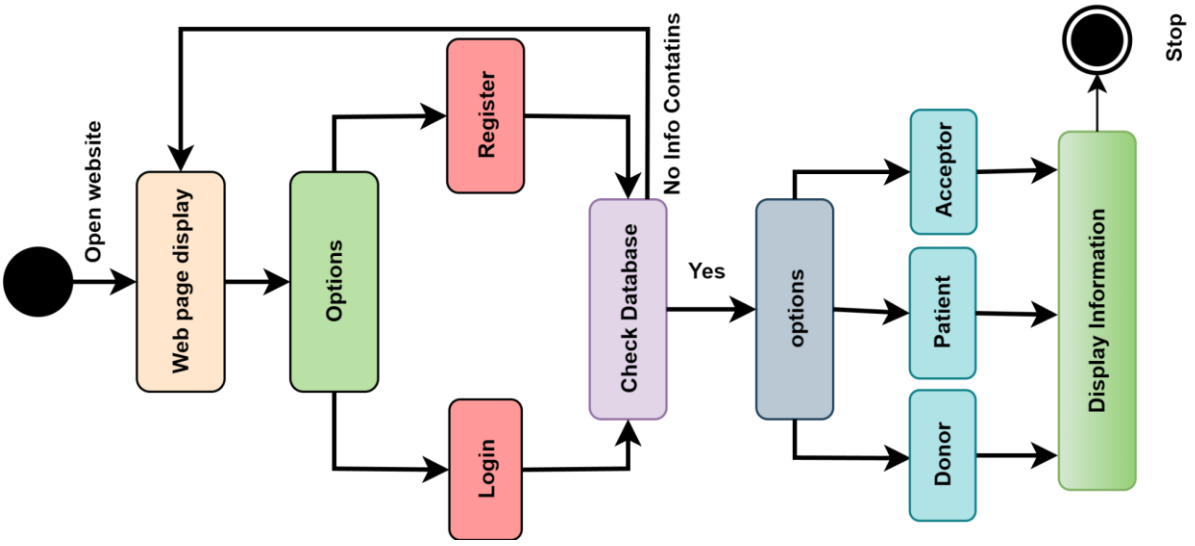

Fig.15 Existing Design Approach 1 Block Diagram

In the Fig.15, the block diagram shows the design approach procedure. At first, the data web page is displayed and the user can choose the options of login or register. If information is found it will show options to show what the user wants to perform. Users can be patients or donors. Patients can make requests to the bank for blood donations. Donors can accept requests via the approval of a blood bank.

Existing Design Approach 2: Online Blood Donation Management System[5] M Vinod, Keerthi Gaddam, and Likhitha Inavolu provide a blood donation agent to create e-information about the donor and organization that are related to donating blood. Additionally, any public customer may use this website to place an online blood request. The primary authority, admin, has the ability to add, remove, and modify content as needed[5].

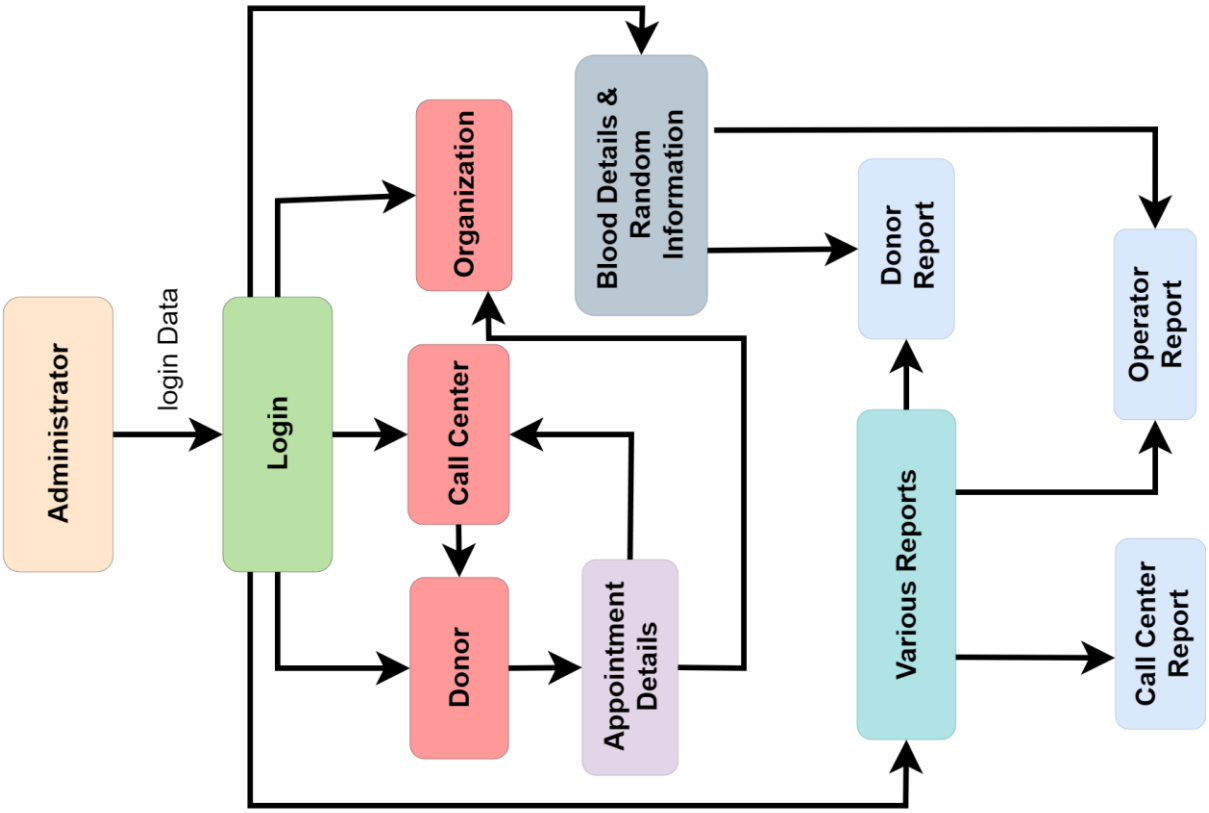

Fig.16 Existing Design Approach 2 Block Diagram

The block diagram in Fig.16 shows that there is an admin in the system, and there are also call centers, organizations, and donors are in the system. The call center can make appointments if needed with donors and organizations. The call center will assign donors to related requests. various reports are generated and given. Such reports are from the call center, operator, and donor. Donors can also explore random information. So mostly all the work is done via the call center, and no direct communication is made with the donor patient or seeker until the donation time. All the blood donated is stored in the organization.

2.3 Proposed Design Approach

A Sophisticated Platform of Blood Donation with Advanced Web Technologies. In our design, we introduced direct interaction between donor and seeker. We created a notification system to make donors aware of requests that were received in the application. For the seeker's trust in the donor location, we have integrated a geolocation service and Google Maps.

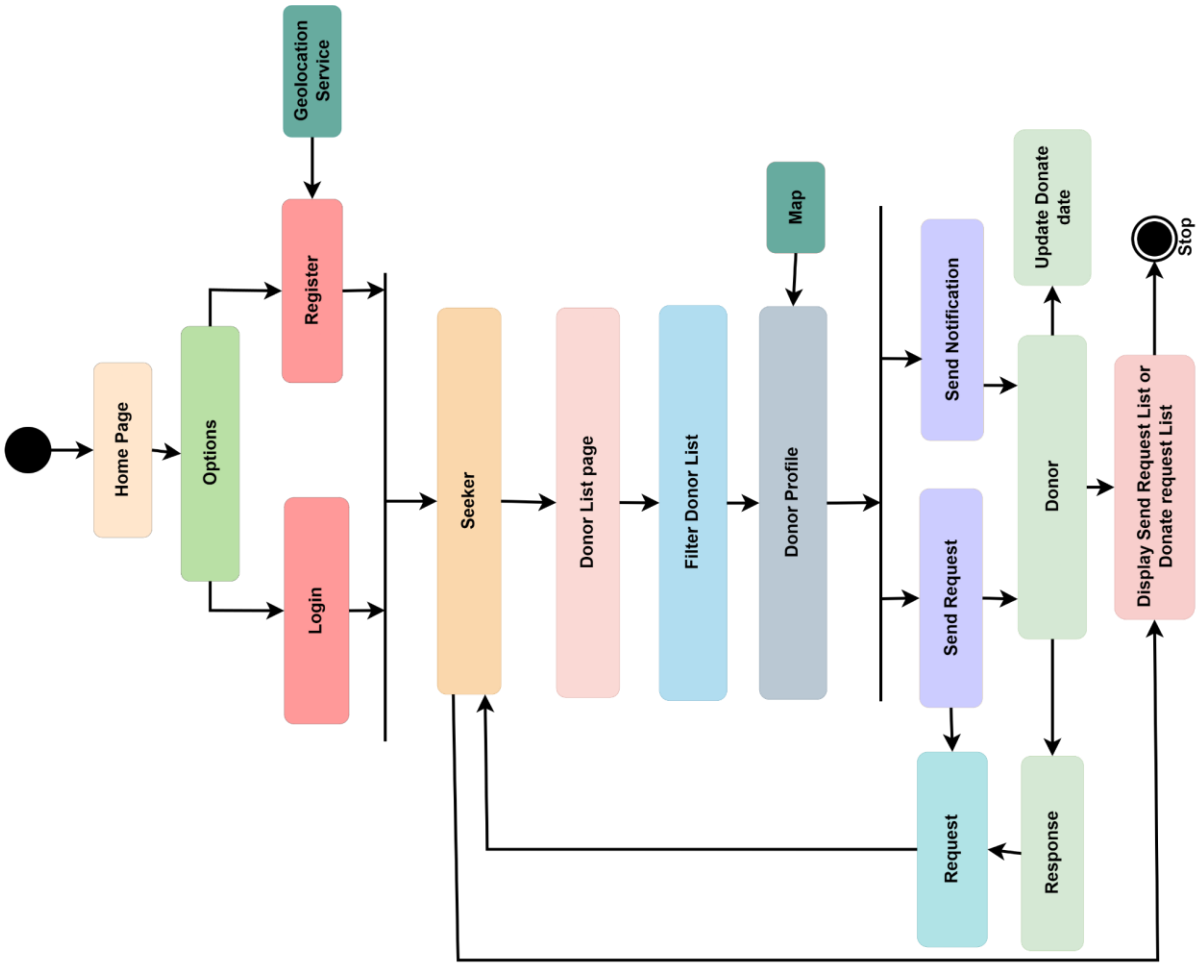

Fig.17 Block diagram of Blood Donation Web Based Application

The block diagram in Fig.17 represents the proposed system, which has a real-time web push notification service, real-time request and response updates to the database, a filter method to filter donor lists, and a System to track donation dates and intervals of 90 days. We have integrated different service APIs into the system. The geolocation service activates while registering so that it can take the latitude and longitude of the user if the user gives permission. It will give the seeker trust to send the request to the donor. The user interface of the application is very user-friendly and easy to interact with. It has a user profile, where user can update their last donation date. It has two separate pages for too see the sent request list and received request list.

2.4 Multiple Design Approach Analysis

2.4.1 Qualitative Assessment

Qualitative assessment is performed on non-numerical data, which attempts to study concepts, opinions, or experiences. It deals with the descriptive data extracted by means of instruments such as interviews, observations, or open-ended surveys. The analysis gives an insight into the subject, trying to explain why people behave, feel, or perceive certain things in a certain way.

TABLE II. Qualitative Assessment of the Three Design Approaches

Topics	Existing Approach 1	Existing Approach 2	Proposed Approach
Direct Interaction	(✗)	(✗)	(✓)
End-to-End Service	(✗)	(✗)	(✓)
Interaction Via Else	(✓)	(✓)	(✗)
Notification	(✗)	(✗)	(✓)
Geolocation	(✗)	(✗)	(✓)
Interval of Donor	(✗)	(✗)	(✓)
User-Friendly UI	(✗)	(✗)	(✓)
Google Map	(✗)	(✗)	(✓)
Appointment via Call Center	(✗)	(✓)	(✗)

After analyzing the existing design approaches with our proposed design approach the qualitative analysis is represented in TABLE II. Here (✗) represents that the feature or functionality is not present in the approach and (✓) represents that it is present in the approach. In our proposed design approach:

1. A seeker can directly contact the donor by sending a request to the donor and accepting the request from the seeker. Filter method is also implemented.
2. There is also a 90-day interval period for donors after blood donation. At this time request sending option is disabled and after 90 days it will reopen again.
3. Geolocation service has also been introduced to this system. When a user tries to register to this system it will ask permission for location, then latitude and longitude will be stored in the database.

2.5 Algorithm For The Proposed Design

A refined and easier way to understand the algorithm of "A Sophisticated Platform of Blood Donation with Advanced Web Technologies" is given below.

Step1: Start

Step 2: Interaction of User (Donor, Seeker):

- **Input**: Track user activity (login, register, donate, request, view, navigate, etc.).
- **Decision**:
    - If the user action is valid, proceed to the next step.
    - If the user activity is not valid, display an error message and end the loop.

Step 3: Authentication of User(Conditional):

- **Check**: Is the action requiring authentication (e.g., donate, view donor, register, send request)?
    - **Yes**:
        - Validate user credentials with the help of Firebase Authentication.
        - If valid, update the state with user information and proceed and find user data from userCollection.
        - If invalid, redirect the user to the login page.
    - **No**: Skip to Routing.

Step 4: Routing:

- **Process**: Track the user request to extract the URL path.
- **Decision**:
    - Render the component if the related component is located.
    - If not found, display an Error Page and option to redirect to the Home Page.

Step 5: State Management:

- Update state based on user actions and interactions.
- Use React's useState and useContext hooks to manage the application's state.
- Update UI elements for the changes in state.

Step 6: API Interaction (Conditional):

- **If** data needed to fetch (e.g., donor list (all user collection), requests)?
    - Send API requests based on the route to the Express server.
    - If successful, parse the data and update the state.
    - Else, handle the error and display an error message. Continue without state update if necessary.
- **Else** Skip to UI Update.

Step 7: UI Update:

- **Process**: Render updated UI components based on the current state and user interactions.

Step 8: Notification and Geolocation Handling (If applicable):

- **Notification**:
    - If the user subscribes to notifications, store subscription details and send notifications as needed.
- **Geolocation**:
    - Update the user's geolocation data if provided in the registration process, for better service personalization.

Step 9: Repeat The Loop:

- **Action**: Return to step 1 and wait for the next user interaction.

Step 10: End

2.6 Software Environment

A software environment is a collection of applications, libraries, and tools that enable users to carry out particular functions. Programmers frequently use software environments to create new apps or execute ones that already exist. For our Blood Donation Web Application, the technologies we have used are HTTP, RESTful API, React JS, NoSQL, Express JS, and MongoDB.

2.6.1 HTTP

HTTP is a protocol used for delivering data from a server to a client, or vice versa, over IP. HTTP works based on request and response. This request may be generated by the client to the server or from the client, and the response is made from the server to the client. However, it remains a stateless protocol, meaning every request is completely independent of all others. It defines some varied methods that act upon a resource: GET, POST, PUT, DELETE, HEAD, OPTIONS, and PATCH methods are supported. HTTPS assures secure interaction by data encryption in transit between a client and a server. Fig.18, is a representation if HTTP request method.

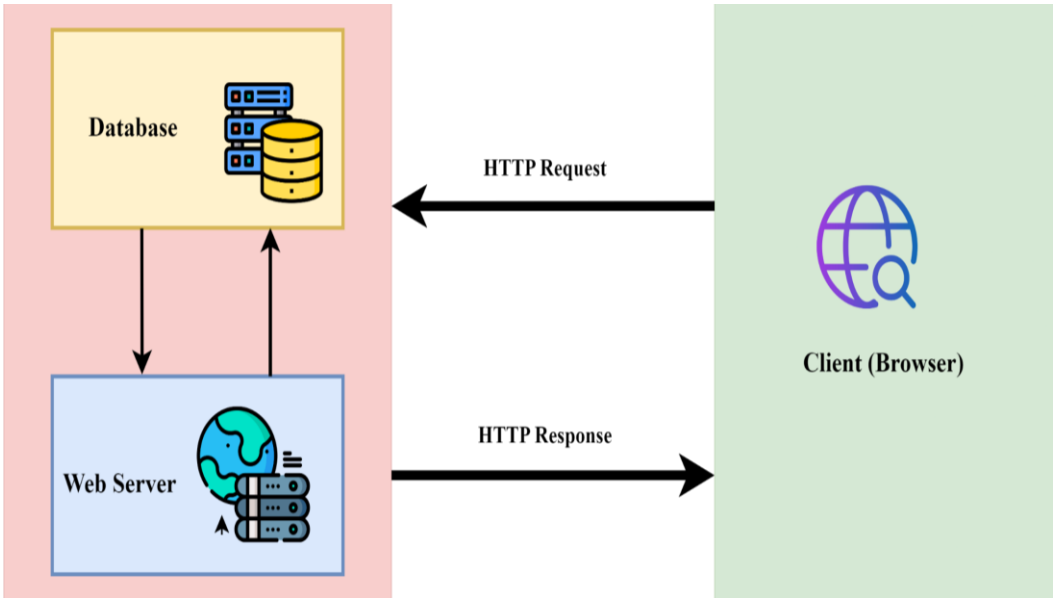

Fig.18 HTTP request method

2.6.2 RESTful API

The RESTful API is an interface through which two computers communicate with each other to convey information over the internet safely. Those aspects are among the big factors that make RESTful APIs largely adopted for modern APIs powering web and mobile apps because of their simplicity, scalability, and flexibility. This broad application makes RESTful APIs very famous for developing web services and microservices architecture.

A RESTful API is a Representational State Transfer API. It is a web service utilizing HTTP methods that perform CRUD (Create, Read, Update, and Delete)operations on resources identified by URIs. RESTful APIs provide a standard for executing CRUD on resources. Be certain that the operation to create, read, update, and delete data is standard and predictable. Their standardization improves the scalability, maintainability, and usability of web services and provides efficient ways through which clients can communicate with servers. The RESTful API operation method is depicted in Fig.19.

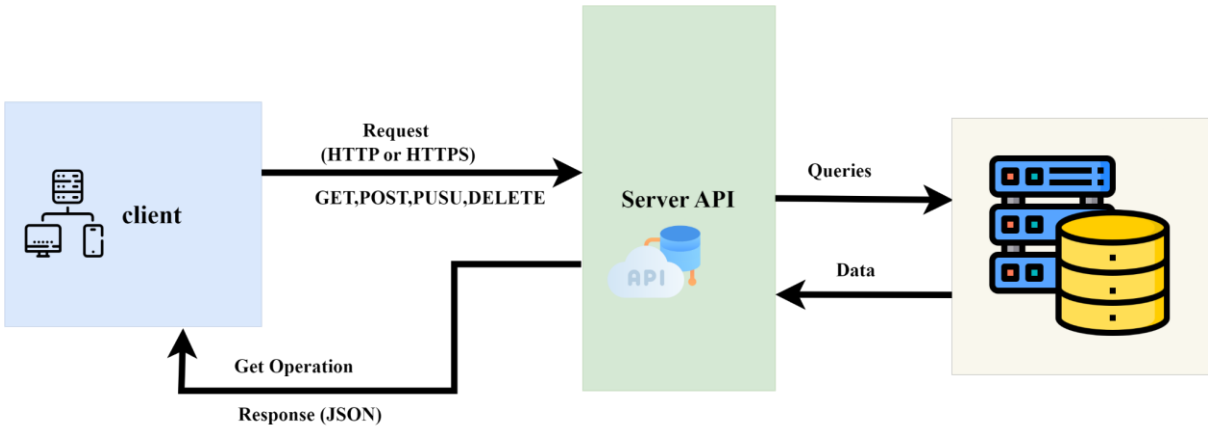

Fig.19 RESTful API operation method

2.6.3 React.Js a JavaScript Library

React.js is a frontend JS library for building single-page and multi-page interfaces. It's the world's most popular programming technology, justifying its solid position with the number of companies reaching out to it. It supports an efficient way to develop reusable components and manage states, then apply this virtual DOM to update UIs for higher performance. JSX provides an extended syntax that allows writing JavaScript and HTML together, which becomes very useful in the design and management of complex UIs. Moreover, the component-based architecture, backed by hooks, makes React powerful in today's Web development.

2.6.4 NoSQL Database

NoSQL databases, unlike the traditional relational databases using structured query language, that is, SQL, and that store their data in predefined schema tables, are much more flexible and use different data models, like key-value pairs, document databases, column families, and graphs. It is this flexibility that makes NoSQL databases efficient at handling data that doesn't lend itself so easily to tables, such as text, images, or highly complicated relationships.

NoSQL databases provide several key facilities that increase their flexibility and performance, basically by supporting flexible data models in the form of key-value pairs, documents, column families, or graphs, which support adaptable schemas and dynamic data structures. This flexibility is good for managing different and unstructured data. It provides high speed in data accessing and processing, scalability, and a flexible data model. On the whole, NoSQL databases are supported by a wealth of tools and libraries; most of them have strong backing from the community, and integration with some programming languages and platforms is well implemented.

2.6.5 Express.js a Node.js Framework

Express.JS is a Node.JS web application framework that is used in developing web and mobile applications. Hence, it would be primarily used in the development of single-page, multi-page, and hybrid web applications. This tends to save much of the coding time to nearly half while the web and mobile applications remain efficient. Besides, Express.JS is a minimal and flexible Node.js framework that enables rapid development of web applications and APIs with ease.
It provides an easier way of routing and middleware management, hence easily handling HTTP requests and responses. Because it is flexible and modular, Express.js is suitable for both small projects and large-scale applications, with the benefit of an active community and integration with many tools and libraries.

2.6.6 MongoDB a NoSQL Database

One of the most popular NoSQL databases is MongoDB, offering flexibility, very high scalability, and high performance. This document database allows for structured or unstructured data in the easiest possible manner. In its JSON-like hosting of documents, MongoDB gives dynamic schemas that easily fit into the changing structures of data. Horizontal scaling in sharding—a process of distributing the data across several servers—eases a Mongo DB from handling huge volumes and high traffic. On the other hand, replica sets are quite useful for high availability due to data redundancy and fault tolerance. Rich sets of indexing and aggregation features allow MongoDB to be pretty effective at running complex queries and processing data. Generally, this is how MongoDB handles functionality for unstructured and semi-structured data, scaling features, and support for modern application development.

2.7 Conclusion

Our blood donation web app system environment is the complete set of technologies and tools that support the development and run-time of our software. The environment will include HTTP as the communication protocol, RESTful APIs for structured data exchange, React JS at the front-end interface, and NoSQL databases like MongoDB at the back-end for efficient storage and retrieval of data. Express JS will provide the back-end framework, which allows robust server-side logic and API management. The combination of these technologies provides a reliable, scalable, and efficient base under the functionalities of the application to ensure an ideal experience for end-users.

Chapter 3: Software Description

3.1 Introduction

The Blood Donation Web Application was developed to increase the efficiency of blood donation such that it will become easy for users to contribute to it and receive timely assistance. By using state-of-the-art web technologies and best practices, it shall provide a reliable and scalable solution, making it easier for donors and seekers to communicate effectively.

It details all phases of system architecture, design, and implementation. It means it captures, at the system level, all the components of a system and interactions among them with the rationale behind the design choices. Besides, it gives deployment strategies and technologies used to guarantee the reliability and performance of the system.

3.2 Frontend Design

Donors and those looking for blood interact with the system through the frontend of the blood donation web application. The user experience on various devices and screen sizes is guaranteed by this interface's intuitive, responsive, and accessible design. Best practices for user interface and user experience design are followed in the implementation of the design, which makes use of modern web technologies.

3.2.1 Purpose and Functionality

The frontend, which is the application's user interface component, is in charge of information display, user input collection, and enabling user-system interactions. Key functionalities include:

1. **User Registration and Login:** Allows users to create an account, authenticate, and access the system's features.
2. **User Profile:** Displays detailed profiles, including personal information, and blood type, and updates the last blood donation date.
3. **Donor List:** Displays all the donors registered in the system with sort profile information.
4. **Filter Donor List:** Enables users to filter donors by blood type and area.
5. **Sent and Received Request Page:** Pages to manage the request that is received or sent.
6. **Donor Profile:** Contains all the information of the donor and the location of the donor when registered in the application.

3.2.2 Design Considerations

The frontend design prioritizes the following aspects:

1. **User-Centric Design:** While developing the draft of the design user experience was always prioritized with ease of use. To reduce user effort and increase satisfaction, the design adheres to accepted UX principles.

2. **Responsiveness:** Trying to make the application responsive leads to adapting to both desktop and mobile screen sizes and devices, providing an optimal viewing experience. This responsiveness is achieved using Tailwind CSS.
3. **Performance Optimization:** The frontend is optimized for performance to ensure fast loading times and smooth interactions and minimize HTTP requests.
4. **Security:** Implementations include input validation, secure handling of user data, and alerts for any wrong input or mismatch which was achieved with Sweet Alert a javascript library.

3.2.3 Technologies Used

1. **HTML and CSS:** To structure and style the application HTML and CSS were used. Tailwind CSS is also used for responsive design.
2. **React JS:** The UI of the blood donation web application is built using React JS, which is a JavaScript library to create dynamic user interfaces. It provides a component-based architecture that makes the code to be reusable.
3. **Google Maps API:** Integrated for geolocation services, allowing users to view donors' registered location on the donor profile page.

3.2.4 Elements in Frontend Interface

1. **Navbar:** Provides easy access to various sections of the application, such as Home, Look For Donor, User Profile, Sent Request, and Received Request.
2. **Forms:** Used for user registration, and login, with validation to ensure accurate data entry.
3. **Cards and Lists:** Used to display donor lists and their information and filtered results in an organized and visually appealing manner.
4. **Modals and Alerts:** To give feedback to users, such as confirmation dialogs, and error messages.

3.3 Backend Design

Node JS and Express JS are used in the backend of our blood donation online application to provide a stable and scalable foundation for server-side functionality, data management, and API endpoints. The backend architecture is made up of several distinct components and services that work together to ensure efficient data processing, secure communication, and seamless connection with external systems.

3.3.1 Server and Routing

1. **Node JS and Express JS:** The server of the blood donation web application is built using Node JS, and Express JS. Node JS is a run time environment of JavaScript and Express JS is a framework of Node JS. Express JS is responsible for routing, middleware, and request handling, making it easier to build a structured and maintainable application.

2. **API Endpoints:** The backend provides RESTful API endpoints for various functionalities, including blood donation requests, user profiles, registration, and more. These endpoints follow REST principles, ensuring clear and predictable communication between the client and server.

3.3.2 API Design

The API design follows RESTful principles, ensuring a standardized, stateless interaction model between the frontend and backend. Key aspects include:

1. **Base URL**: The API is accessible at https://donate-red-server.vercel.app/.
2. **Endpoints:**
    a. **User Management**
        - POST **/users**: Store data of a newly registered user.
        - GET **/singleUsers/:email**: Retrieve single user by email.
        - PATCH **/user/:id**: Update the user's last donate date.
    b. **Blood Donation Requests**
        - POST **/request**: Create a new donation request.
        - GET **/request/**: Retrieve details of all donation requests.
        - GET **/request/:id**: Retrieve details of a specific donation request.
        - DELETE **/requests/:id**: Cancel a donation request.
    c. **Donor Management**
        - GET **/users**: Retrieve a list of all donors by sorting all last donate dates in descending order, with optional filtering by blood type and area.
        - GET **/users/:id**: Retrieve details of a specific donor.
        - PATCH **/request/:id/state**: Update donation request status.
    d. **Notifications and Miscellaneous**
        - POST **/notify**: Send a notification to users based on certain criteria.
        - POST **/subscribe**: Subscribe a user to notifications.
        - GET/**totalUsers**: Counts total documents for pagination.

3.3.3 Database Management:

1. **MongoDB:** The database used in the application is MongoDB, a NoSQL database. It is very popular because of its flexibility and scalability. To facilitate more effective data handling, MongoDB stores data in BSON format. MongoDB's document model is used to store and manage data related to users, subscriptions, and requests.
2. **Aggregation Pipeline:** Complex data processing and analysis can be done with MongoDB Aggregation. This pipeline makes it easier to carry out data conversions and aggregations, including filtering donor lists based on thana and blood type. Data retrieval and manipulation are made possible and easier by the aggregation pipeline phases, which include filtering, grouping, and sorting.

3.3.4 Notification System:

1. **Web Notifications:** The Web Push API is used to construct a notification system. Real-time notifications regarding new blood donation requests are sent to donors and receivers using this system. Notifications improve user communication and increase the system's efficiency.

3.4 Database Design

The database design for the Blood Donation Web Application consists of three main collections: Users, Requests, and Subscriptions. Each collection stores different types of information essential for the system's functionality. The database is designed using NoSQL principles and is suitable for MongoDB, ensuring scalability and flexibility.

3.4.1 Users Collection Representation

In TABLE III, the user collection is represented by the field name representing the entity and the field value type for the value type of attribute. The values are mostly in string data type except for latitude and longitude which are in double-type data format.

TABLE III. User Collection Representation

Field Name	Field Value Type
_ id	ObjectId
Name	String
Email	String
Gender	String
Contact	String
District	String
Thana	String
Date of Birth	String
Blood Type	String
LastDateOfDonation	String
Photo	String
Issue	String
Latitude	Double
Longitude	Double

3.4.2 Donate Request Collection Representation

For the collection of request data, TABLE IV represents the attribute and value type of their corresponding attribute. In this collection, only the request status field value is in boolean format and the remaining are in string type data.

TABLE IV. Donate Request Collection Representation

Field Name	Field Value Type
_id	ObjectId
SeekerID	String
SeekerName	String
SeekerEmail,	String
SeekerPhoto	String
DonorID	String
DonorName	String
DonorEmail	String
DonorPhoto	String
DonorContact	String
RequestStatus	Boolean
State	String

3.4.3 Subscription Collection Representation

The MongoDB database also contains another collection for the subscription of users which stores the user's endpoint to send notifications which are depicted in TABLE V. This collection, is a complex structure, here subscription is an object type data containing endpoint and expiration time and keys as well an object type data.

TABLE V. Subscription Collection Representation

Field Name	Field Value Type
_id	ObjectId
Subscription	Object
Endpoint	String
ExpirationTime	Null

Keys	Object
Auth	String
Email	String

3.4.4 Sample Data in JSON Format for User Collection

This section defines the JSON format of the user collection defined in TABLE III. A similar data format is used to store the data in the database. Each line corresponds to two values, one is for the attribute and another one is for the value of the attribute. As an example "photo":"https://i.ibb.co/Gc5XwBg/PP-02.jpg" defines that “photo” is the attribute and the "https://i.ibb.co/Gc5XwBg/PP-02.jpg” is value of the attribute.

{

"_id":{"$oid":"66a946f53e5ed43f248c5bc8"},

"name":"Imran Hossain Joy",

"email":"imran@gmail.com",

"photo":"https://i.ibb.co/Gc5XwBg/PP-02.jpg",

"contact":"01791043734",

"gender":"Male",

"group":"B Positive",

"date":"1997-06-17",

"district":"Dhaka",

"thana":"Motijheel Thana",

"lastDate":"2023-06-09",

"issue":"No",

"latitude":{"$numberDouble":"23.8045"},

"longitude":{"$numberDouble":"90.3607"}

}

3.4.5 Sample Data in JSON Format for Request Collection

This section is also a JSON format of the request collection defined for TABLE IV. Here also each line corresponds to two values, one is for the attribute and another one is for the value of the attribute containing all the information related to when a request is sent to a donor. As an example "seekerEmail": "ridoyhasan87@gmail.com".

```
{
  "_id": { "$oid": "66a947133e5ed43f248c5bc9" },
  "seekerEmail": "ridoyhasan87@gmail.com",
  "donorID": "66472d23be6b8eb442783c10",
  "donorName": "Imran Hossain Joy",
  "statusReq": true,
  "donorPhoto": "https://i.ibb.co/Gc5XwBg/PP-02.jpg",
  "donorEmail": "imran@gmail.com",
  "state": "requested",
  "donorContact": "01791043734",
  "seekerName": "kamrul Hasan Ridoy",
  "seekerPhoto": "https://i.ibb.co/523pjpR/sagdsgd.jpg",
  "seekerID": "6676958a88cac98c2e36ea15"
}
```


3.4.6 Sample Data in JSON Format for Subscription Collection

This section is also a JSON format of the subscription collection defined for TABLE V. Here inside JSON data an attribute contains another object with attributes making a complex data structure. For example “subscription” is an attribute that contains another two attributes “expirationTime” and “keys”. Here keys also contain another attribute of “auth”, making it an object inside of another object.


```
{
  "_id": { "$oid": "66a947803e5ed43f248c5bca" },
  "subscription": {
        "endpoint":"https://fcm.googleapis.com/fcm/send/f74nr6azcvY:APA91bGqPl7AJlmY
        PPVscCJjSgrwXvHHUfb1GNTiRkCGG1311M-
        2eESWFv9AyqD9AxtQSK_3fFXdc6N-
        IweHESDEIQljjkgWl1cnBDwr8jOvrnuM4x201ER7hdG4eG3ZDcPW1x1OtJNZ",
    "expirationTime": null,
    "keys": {
      "auth": "6cGCIyHEmiHuk5X84quXQw"
    }
  },
  "email": "ayesha50-020@diu.edu.bd"
}
```

3.5 Notification System Implementation

It is the blood donation app's explicit notification system, keeping users up-to-date on requests. Web push notifications are a form of implementation that conveys a message to a user's device. It puts into play some of the major key components and considerations: service workers, the Web Push API, and VAPID keys.

1. Implementation Overview

The system implements the notification system through service workers combined with the Web Push API. Service workers will sit between the server and a user's device, ensuring that the user's device receives push notifications even when the web app has been closed. The Notifications component in the front end will initiate the service worker and subscribe the user for notifications.

2. Subscription Process

On log-in, the Notifications component will register a service worker and subscribe to the notifications using the PushManager available on the browser. After this, the subscription details, along with the user's email, are sent to the server to be stored in the database. All this setup is to allow the server to send notifications to users, targeting them by their email addresses.

3. Sending Notifications

It enables the server to utilize the Push API for sending out notifications. Now, on any event of interest, say, any new donation request, it fetches subscription details from the database and then sends a notification payload. The payload explicitly mentions the title, body, and icon of the notification, stating clear and concise information to the user.

4. Use of VAPID Keys

Very important in secure and authenticated delivery is the use of VAPID keys. The main role of the VAPID keys is two-fold:

1. **Sender Authentication**: These keys are provided to clients requesting a PushSubscription to pass on to the endpoint to which they subscribe. VAPID keys authenticate the server to the push service; be it Google Cloud Messaging or Mozilla's Push Service. This ensures that, similarly, the notification comes from a trusted source, and user endpoint receivers will be protected against messages from parties other than those allowed.
2. **Message Encryption**: VAPID keys thus help with encrypting the payload of the notification. It ensures that only the intended party can actually decrypt and read the message. This is highly crucial in the maintenance of the privacy and security of the content delivered to users.

3.6 Geolocation Implementation

The Geolocation System in the application provides location data for seekers, aiding in the identification and coordination of nearby donors. The implementation leverages the browser's built-in geolocation API, ensuring that location data is obtained efficiently and securely.

Fetching User Location:

1. The system utilizes the navigator.geolocation API to retrieve the user's current geographic location while registering into the system.
2. Upon component mount, the getLocation function is triggered, as a user to permission to for location.

Handling Geolocation Data:

1. The getCurrentPosition method is used to fetch the user's coordinates (latitude and longitude). This data is then stored in the state variables latitude and longitude.
2. If the location is successfully fetched, the system sets the loading state to false. In case of any errors (such as user denial or browser issues), appropriate error messages are logged, and the loading state is similarly updated.

3.7 Deployment and Hosting

The Blood Donation Web Application is deployed using a combination of Firebase and Vercel, which provide a robust and scalable infrastructure for both front-end and back-end components. The following TABLE VI describes our project deployment and hosting.

TABLE VI. Deployment and Hosting

Front-End Deployment	The front end of the web application is hosted on Firebase.
	It is fast and secure static content delivery, ensuring the user interface of our application is served efficiently across various devices.
Back-End Deployment	The back-end services are deployed on Vercel.
	It is a serverless function and a static site-optimized platform.

	Providing zero-config automated scaling, and auto-deployment without configuration,
	Integrates with Git repositories to further make the development and deployment process easier.
Version Control	The project's source code is managed using GitHub, which facilitates version control and collaboration.
	GitHub repositories are used to track changes, manage branches, and perform code reviews.

3.8 Conclusion

This is our final system description of the Blood Donation Web Application, which gives a broad overview of the architectural design. It comprises of all important components, combined and integrated, providing a seamless user experience from its front end developed using React JS to its back end powered by Express JS and MongoDB. It uses HTTP and RESTful APIs to make communication and data handling between the mobile application and the server fast and effective, all while ensuring safety and reliability. Geolocation services help in increasing the relevance and accessibility of the service, while notifications of different kinds are used to keep users updated in real time. Therefore, the system setup meets both the core requirements of a blood donation platform and provides flexibility and scalability for future enhancements.

Chapter 4: Software Requirement Specification

4.1 Introduction

The process of creating our web application for blood donation is covered in this chapter. An in-depth description of the actions done to guarantee a full comprehension of the project's growth is given in this chapter. It provides an overview of the strategy our team adopted to resolve the issue and provides an explanation of the theories and concepts we used for our research and computations. This chapter is broken down into the following subsections for better clarity.

In the proposed, Blood Donation Web Application with an authentication system, users (donors, seekers) will have secure access to the platform. Each user would have unique login credentials, enhancing the systems' security and ensuring that only authenticated users can interact with the functionality of updates, requests, acceptance of requests, etc. Using the Firebase authentication system, the authentication procedure would involve confirming the user's identity with a username and password combination. In this chapter, we will describe our Blood Donation web application, by using use cases, use case diagrams, activity diagrams, swimlane diagrams, ER diagrams, etc.

4.2 Software Requirements

Software requirements document the functionality, features, and constraints that a software application must possess to be able to meet users' and stakeholders' needs. These are very important to be adhered to while any development is underway to ensure that the final product is functional, reliable, and user-friendly.

4.2.1 Normal Requirements

User Authentication: Basic login, and registration with multi-factor authentication by Firebase**.**

Donor Management: Display all registered donors by pagination.

Donor Information: Display single donor information on a different page.

User Profile: Show users their registered personal information**.**

Request: Sending requests to available donors matched with the required blood type.

Response: Accept the request of the seeker in response to the request.

4.2.2 Expected Requirements

Tracking Donate Time: Real-time tracking of time from the last donation of blood donors.

Search/Filter Donor: A seeker can filter donors based on the blood type required and the area near the seeker or patient.

Notifications: Basic alerts for donors if a seeker sends a request.

Responsive Layout: Accessible through both mobile and desktop through responsive design.

Easy to use: User-friendly interface to access and use all functionality.

Request List: Different page to show the list of sent requests, and another for received requests.

Ambiguous: Donors major contact information will be hidden, and will only be shown if the donor accepts a request from the seeker.

Scalability: Ability to scale the database with the growth of the organization.

Real-Time Update: Update the database in real-time based on the actions of users made through the system.

Performance: Synchronize based on real-time updates. Ability to handle a high volume of user requests efficiently and maintain quick response time.

4.2.3 Excited Requirements

Geolocation: Will take users' latitude and longitude to give ensuring to the seeker that the donor is located in the location during registration**.**

Google Map Integration: Display the location of donors on the donor's information page.

Donation Frequency Regulations: A system to track donation dates and enforce a 90-day interval between donations to protect donor health. During this interval, seekers won't be able to send requests.

Real-Time Donate Date Update: A system to update the donate date if the donor donates and presses the button in the user profile.

User Profile Picture: User can upload their picture while registering, and pictures will be stored in a cloud.

4.3 Use Case Diagram Elements

Six visual components make up the use case diagram, which depicts the entire system.

- Actors
- Use case
- System boundary
- Include
- Extend
- Generalizations

4.4 Use Case Diagram

Use case diagrams show the interaction between the system and its actors and they provide a graphical description of functionality that a system can deliver concerning its Actors, use cases which are representations of their aims, and any dependencies among those use cases. The following Fig.20, is the use case diagram for the Blood Donation Web Application.

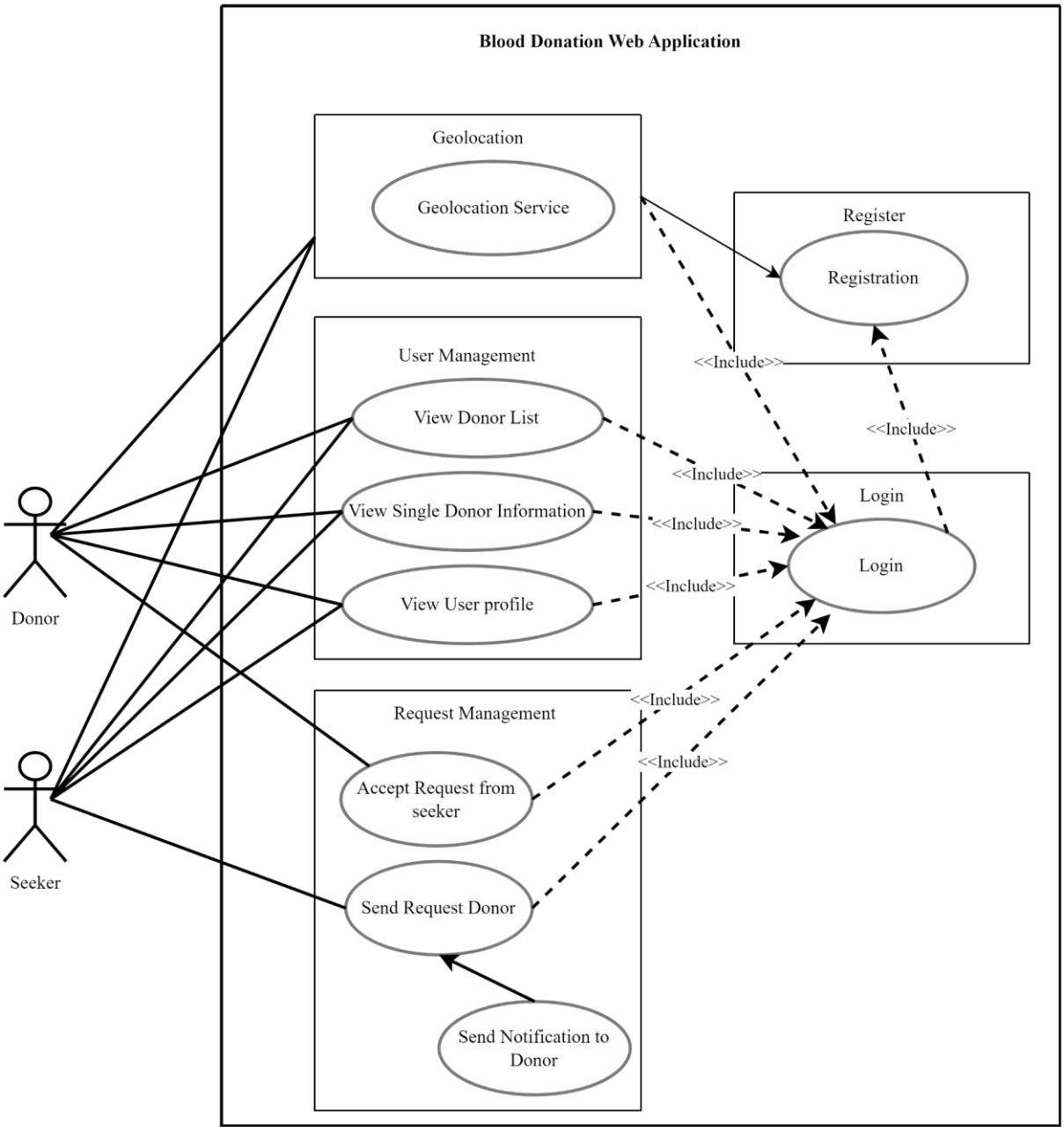

Fig.20 Use case diagram for blood donation web application

4.5 Use Cases for Blood Donation Web Application

4.5.1 Use case 1: User Registration

Table VII shows the use case description of user registration in the Blood Donation Web Application. This use case addresses the process of creating an account for a new user, elaborating on the registration procedure through which each new user will have the ability to join the site and utilize all of its functionality without any problems.

TABLE VII. Use Case For User Registration

Use-Case	**User Registration**
Primary Actor	New User
Goal in Context	To allow a new user to register on the platform
Preconditions	User has internet access
Trigger	User decides to register on the platform
Scenario	1. User navigates to the registration page. 2. User fills in the required information (name, email, password, contact information, blood group, etc.). 3. User submits the registration form. 4. System (Firebase) validates the information and creates a new user account. 5. User receives a confirmation notification.
Exceptions	1. If the email is already registered, an error message is displayed. 2. If any required information is missing, an error message is displayed.
Priority	Essential
When available	During the initial use of the platform
Frequency of use	Low, only once per user
Channel to actor	Web interface
Secondary actor	None
Channel to secondary actor	N/A
Open issues	None

4.5.2 Use case 2: Validate User Email and Password

TABLE VIII describes a use case of the system checking a user's credentials during the login process. A user inputs their email address and password on the login screen. The system then matches this with the data stored in the Firebase. If the email address and password correspond to a registered account, access is granted and a user session created. In case of wrong credentials, an error message will pop up, asking for reentry of information. This use case efficiently verifies users' credentials to ensure security in application access.

TABLE VIII. Use Case For Validation

Use-Case	**Validate User Email and Password**
Primary Actor	Registered User
Goal in Context	To authenticate a registered user and provide access to their dashboard
Preconditions	User has a registered account
Trigger	User decides to log in
Scenario	1. User navigates to the login page. 2. User enters email and password. 3. System authenticates the user. 4. User is redirected to their user profile.
Exceptions	If the email or password is incorrect, an error message is displayed.
Priority	Essential
When available	First Increment
Frequency of use	Regularly and high
Channel to actor	Web interface
Secondary actor	None
Channel to secondary actor	N/A
Open issues	None

4.5.3 Use case 3: Filter Donor List by Blood Type and Area

Table IX describes a use case for a feature which can help in fast and efficient searching for blood donors based on their blood type and thana. After accessing the donor management interface, a user will specify his criteria and obtain a list of donors matching these criteria. This ensures that users may get hold of appropriate donors quickly, more so in emergency cases

where specific blood types are required, mainly in certain locations. If no matching donors are available, then a blank page will show inside the header and footer. This feature is very important to optimize the process of donor matching and is constantly available through the application's interface.

TABLE IX. Use Case For Filtration of Donor List

Use-Case	**Filter Donor List by Blood Type and Area**
Primary Actor	Seeker
Goal in Context	To find potential blood donors based on specific blood type and geographical area.
Preconditions	Seeker is logged in.
Trigger	Seeker initiates a search for potential donors.
Scenario	1. Seeker navigates to the donor list page. 2. Seeker selects the required blood type and the desired area. 3. System processes the search criteria. 4. System filters and displays a list of potential donors that match the criteria.
Exceptions	1. No donors match the search criteria.
Priority	Essentail
When available	Any time
Frequency of use	Whenever a seeker needs to find a donor
Channel to actor	Web interface
Secondary actor	Donor
Channel to secondary actor	N/A
Open issues	Optimizing search performance for large datasets.

4.5.4 Use case 4: Blood Donation Request

The following use case, TABLE X, allows the user to create and manage blood donation requests inside the application. The request sieve is then applied to the list of donors; if the user locates a potential matching donor who is also available, he can land on the donor profile and click the 'Send request' button. One can trigger the request function in generating request information objects and thereafter send requests to the donors.

TABLE X. Use Case For Blood Donation Request

Use-Case	**Blood Donation Request**
Primary Actor	Seeker
Goal in Context	To allow a seeker to request blood from donors
Preconditions	User has a registered account and logged in
Trigger	Seeker needs blood
Scenario	1. Seeker navigates to the donor list page. 2. Seeker filters the required information (blood type, area, etc.). 3. Seeker navigates to the available donor information page 4. Seeker press the send request button 5. System sends notifications to potential donors.
Exceptions	1. If the request cannot be processed, an error message is displayed.
Priority	Essential
When available	Any time
Frequency of use	Variable
Channel to actor	Web interface
Secondary actor	Donor
Channel to secondary actor	Web notification
Open issues	User should enable notification

4.5.5 Use case 5: Web Push Notification System

TABLE XI Use case: Enables real-time user notifications on some basis to be sent, which in this case is the sending of blood donation requests. The users will have the ability to subscribe to such notifications using the web application, and these subscription details, endpoints, and keys will be stored securely. In the case of a seeker sending a request to a donor, a push notification is generated through the system using the stored subscription information. It, therefore, sends this notification to the user's device, ensuring relevant and timely updates to the user. It also manages subscriptions and error logging for effective delivery and efficient user engagement. This functionality enhances communication and allows users to be promptly updated on relevant critical updates.

TABLE XI. Use Case For Notification System

Use-Case	**Web Push Notification System**
Primary Actor	Seeker
Goal in Context	To send a push notification to potential donors when a blood donation request is made.
Preconditions	1. Seeker is logged in. 2. Potential donors have agreed to receive notifications.
Trigger	Seeker sends a blood donation request.
Scenario	1. Seeker navigates to the available donor information page. 2. Seeker press the send request button 3. System sends a push notification to the potential donors. 4. Donors receive the notification on their devices.
Exceptions	1. No potential donors are found. 2. Push notification service fails. 3. Donors have disabled notifications.
Priority	High
When available	Immediately upon deployment
Frequency of use	Whenever a new blood donation request is made
Channel to actor	Web interface
Secondary actor	Donor
Channel to secondary actor	Push notification service
Open issues	1. Ensuring donor notification settings are up-to-date.

4.5.6 Use case 6: Accept or Reject Blood Donation Request

TABLE XII: Case The application provides for the user to successfully handle new requests for blood donation. As soon as a notification goes out to the donor that a new request for donation has been placed, the application allows the donor to view details of the request on the page of received requests. Options of either accepting or rejecting, depending on the willingness of the donor to donate, will be displayed by the system. On acceptance, the donor will change the status to accepted and update both the donor and the requester on successful matching. If a donor rejects your request, the status of the request will be updated to reject and you are no longer the requester in order to look for other donors. It handles donation requests by updating all the concerned donors and recipients appropriately.

TABLE XII. Use Case For Response to Blood Donation Request

Use-Case	**Accept/Reject Blood Donation Request**
Primary Actor	Donor
Goal in Context	To allow a donor to accept or reject a blood donation request
Preconditions	Donor has received a blood donation request notification
Trigger	Donor receives a blood donation request notification
Scenario	1. Donor receives a notification about a blood donation request. 2. Donor reviews the request details. 3. Donor decides to accept or reject the request. 4. System updates the request status and notifies the seeker.
Exceptions	1. If the system fails to update the request status, an error message is displayed.
Priority	High
When available	Any time
Frequency of use	Variable
Channel to actor	Web interface
Secondary actor	Seeker
Channel to secondary actor	None
Open issues	None

4.5.7 Use case 7: Geolocation Update

This use case allows a user to change their current location in the process of registration to the blood donation web application. In updating the location, the system shall use the location services on the device to take the user's newest geolocation information upon initiation by the user. The geolocation data that was collected will be transmitted back to the server for storage within the user's profile. Hence, this information is utilized in giving a more accurate proximity matching between donors and requests, making the blood donation procedure more effective, as shown in TABLE XIII.

TABLE XIII. Use Case For Geolocation Service

Use-Case	**Geolocation Update**
Primary Actor	User

Goal in Context	To update the donor's location using geolocation API
Preconditions	Enables location services
Trigger	User navigates to registration page while registration
Scenario	1. Donor allows location access. 2. System retrieves the geolocation data (latitude and longitude). 3. System updates the donor's location in the database.
Exceptions	1. Donor denies location access. 2. System encounters an error during location retrieval or update.
Priority	Medium
When available	Immediately upon deployment
Frequency of use	Upon login or donor profile view
Channel to actor	Web interface
Secondary actor	None
Channel to secondary actor	N/A
Open issues	None

4.5.8 Use case 8: Update Last Blood Donation Date

This use case allows donors to update the last date they donated blood. After a donation, the donor can log on to the web application to check the date of donation. The date is recorded in the donor's profile in the system. The program checks this information to guarantee the ninety-day gap between any two consecutive donations, thus observing the policies related to blood donation. It avoids early contribution and ensures staggered donation by all donors through the maintenance of correct records of dates of donation. This improves overall safety and efficiency of the donation process and the procedure for this use case is shown in TABLE XIV.

TABLE XIV. Use Case For Updating Last Blood Donation

Use-Case	**Update Last Blood Donation Date**
Primary Actor	User (Donor)
Goal in Context	To update the date of the last blood donation.
Preconditions	User is logged in.

Trigger	User decides to update the last donation date after donating blood.
Scenario	1. User navigates to their profile page. 2. User press the option to update the last donation date. 3. System takes the current date as new donation date. 4. System validates the date and updates the user's profile. 5. System decibels update donation button for 90 days.
Exceptions	1. System encounters an error during the update process. 2. User mistakenly press the button
Priority	Medium
When available	Immediately upon deployment
Frequency of use	After each blood donation
Channel to actor	Web interface
Secondary actor	None
Channel to secondary actor	N/A
Open issues	None

In this project, total of 8 Use cases have been considered described organizingly in the Table VII, VIII, IX, X, ... XIV. All the 8 Use cases have been indicated in the Fig 20.

4.6 Data Flow Diagram

A data flow diagram (DFD) shows the data movement inside the system. DFD helps users with and without technical knowledge to understand a graphical representation of system data flow. Depending on a system's requirements, it may be subdivided into multiple levels; in our DFD, we have segmented it into three levels. Level 0, Level 1, and Level 2 are what they are.

Level 0 which is popularly called context diagram is the top level of a DFD that gives an outline of major data flows and processes. Level 1 is more detailed, which gives the view of subsystems breaking down from level 0. Level 2 is even more detailed than Level 1. It breaks down every process from the system described in Level 1.

4.6.1 Data Flow Diagram for Level 0

The following diagram Fig.21 is for Level 0 or context level DFD of our Blood Donation Web Application.

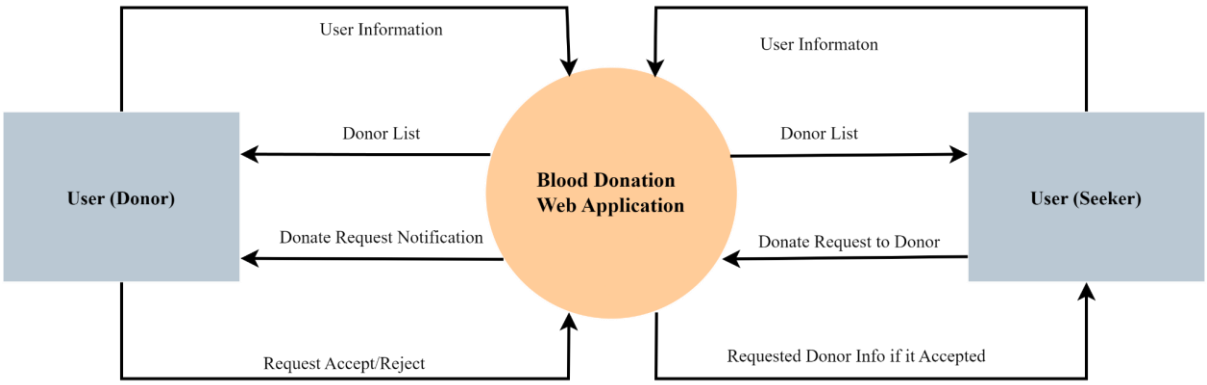

Fig.21 Data Flow Diagram for Blood Donation Web Application of Level 0

4.6.2 Data Flow Diagram for Level 1

The following diagram Fig.22 is for Level 1 DFD of our Blood Donation Web Application.

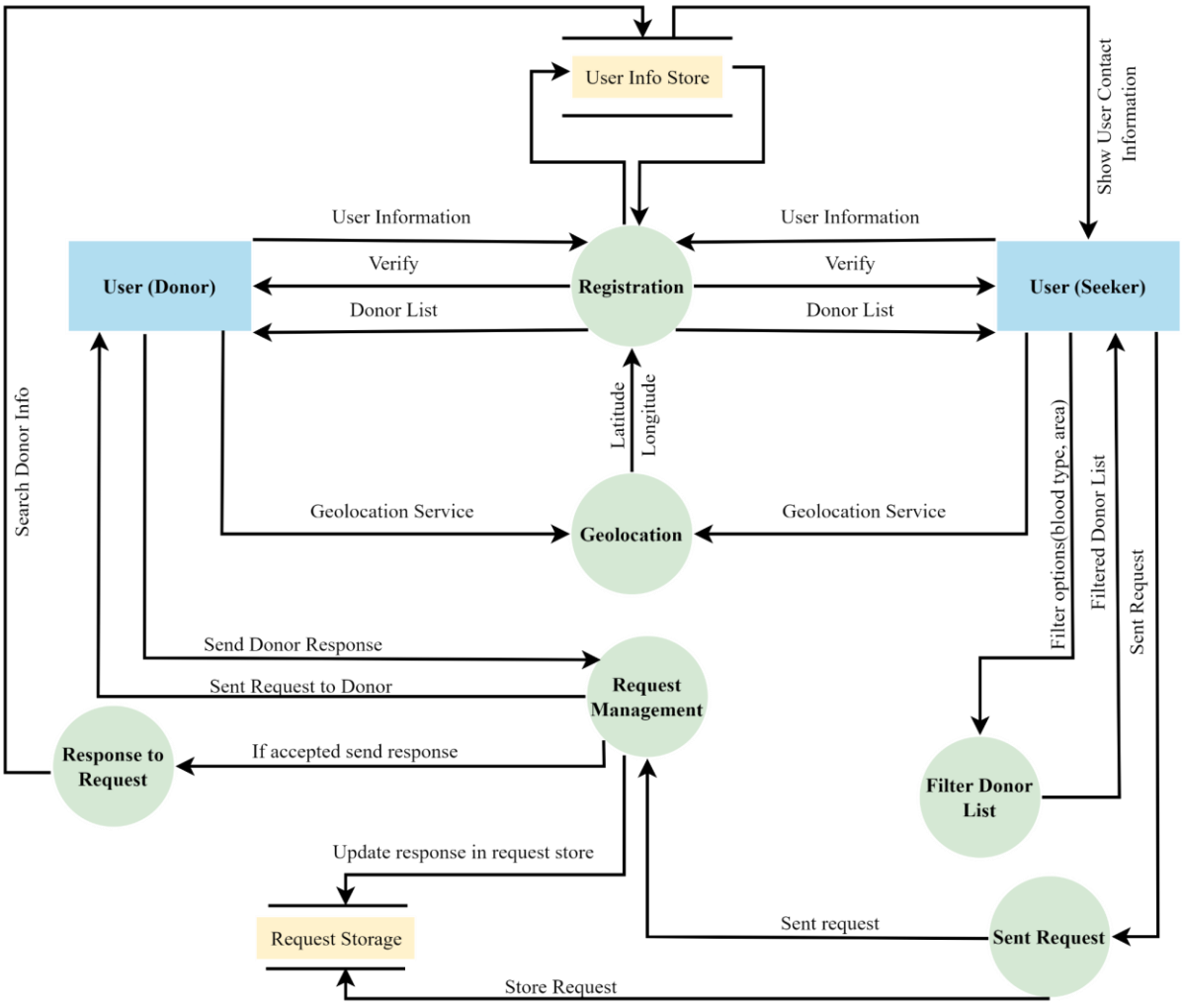

Fig.22 Data Flow Diagram for Blood Donation Web Application of Level 1

4.6.3 Data Flow Diagram for Level 2

The following diagram Fig.23 is for Level 2 DFD of our Blood Donation Web Application.

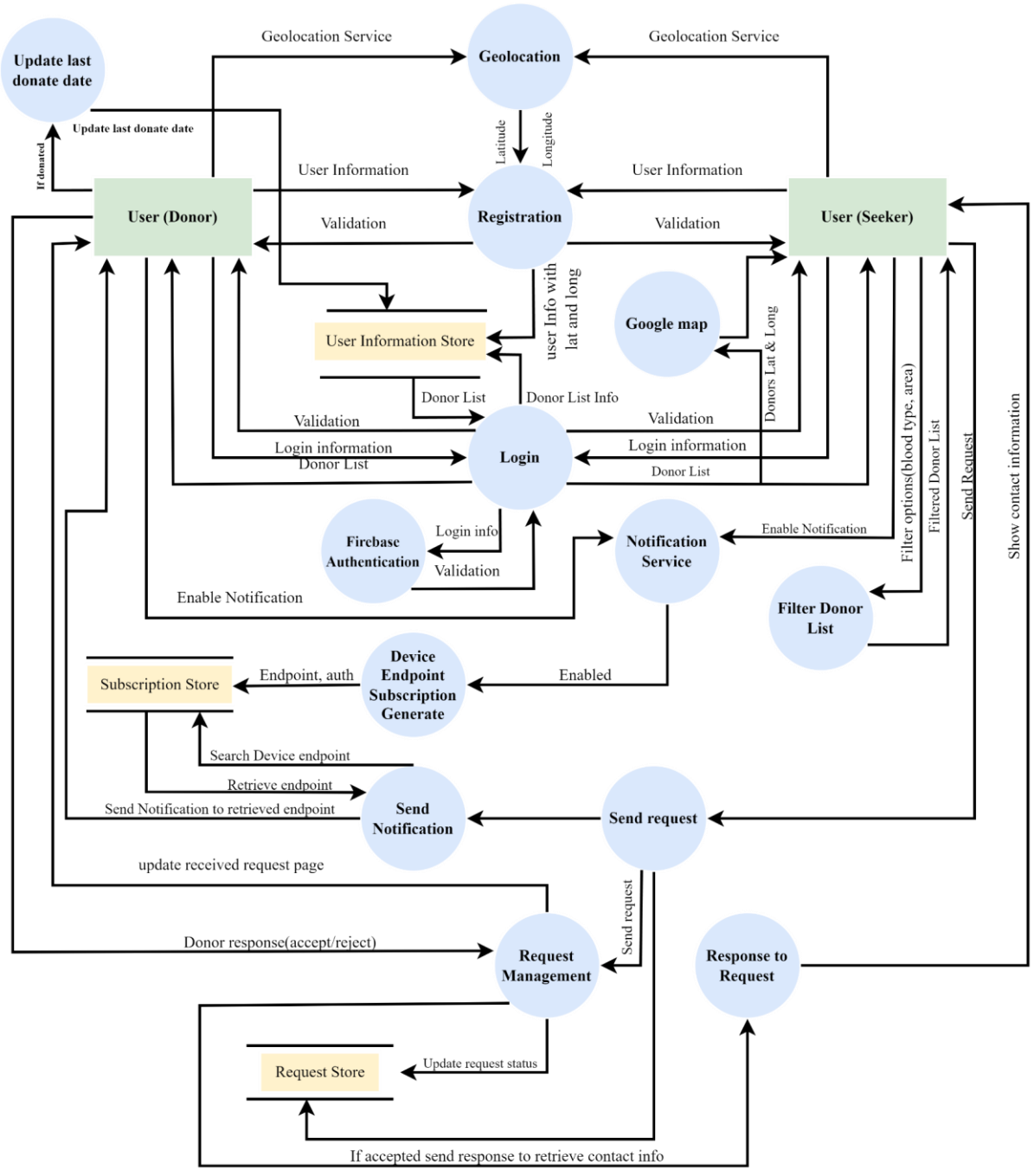

Fig.23 Data Flow Diagram for Blood Donation Web Application of Level 2

4.7 Activity Diagram

The activity diagrams provide meaningful visualization of the sequence and flow of activities, which can be very helpful in the analysis, design, and optimization of business processes and system workflows.

4.7.1 Activity Diagram for Login and Registration

The diagram in Fig.24 represents the activity flow for the login and registration system and here For the login process, the user inputs their email and password, which the system validates. If the credentials are correct, and the user is already registered the user is granted access; otherwise, an error message prompts the user to try again. In the registration process, new users provide their details, including email, password, and personal information for example blood type, date of birth etc. The system then checks if the email is already registered or not. If email is not registered system registers the user and will give access to use the functionality.

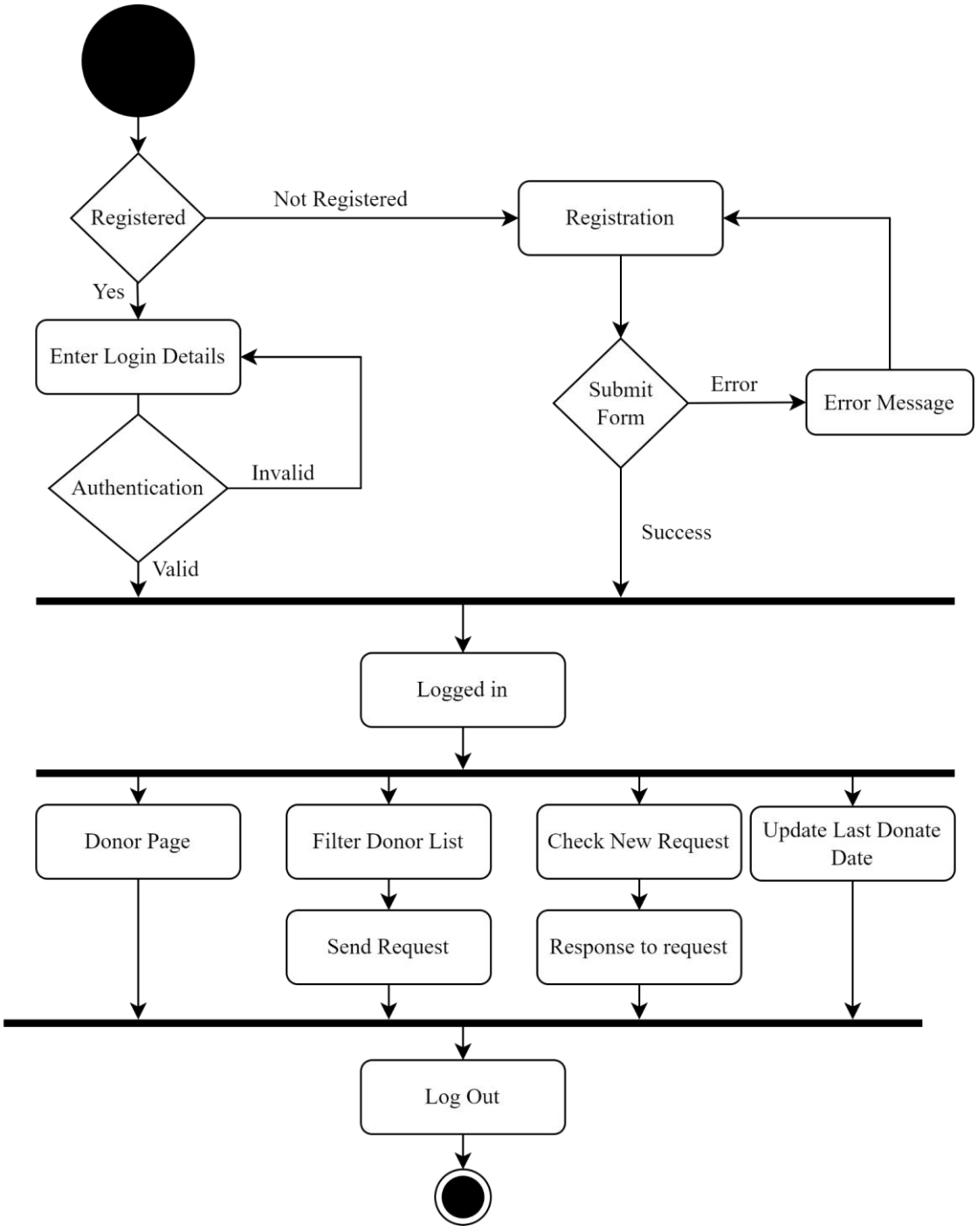

Fig.24 Activity diagram for login and registration system

4.7.2 Activity Diagram for Filtering Donor List

Fig.25 represents the activity diagram for the filtering donor list. At first, the user will log in to the system. Then will navigate to the donor list page. Then it is forked into two options. Selecting blood type will search for the matched blood type object and will display them. The selecting thana will search for matching donors of the same thana. If both are selected will filter the recently filtered data, and could be vice versa. Then it will join both options resulting in terminating the operation. If not found user can change the selection option also.

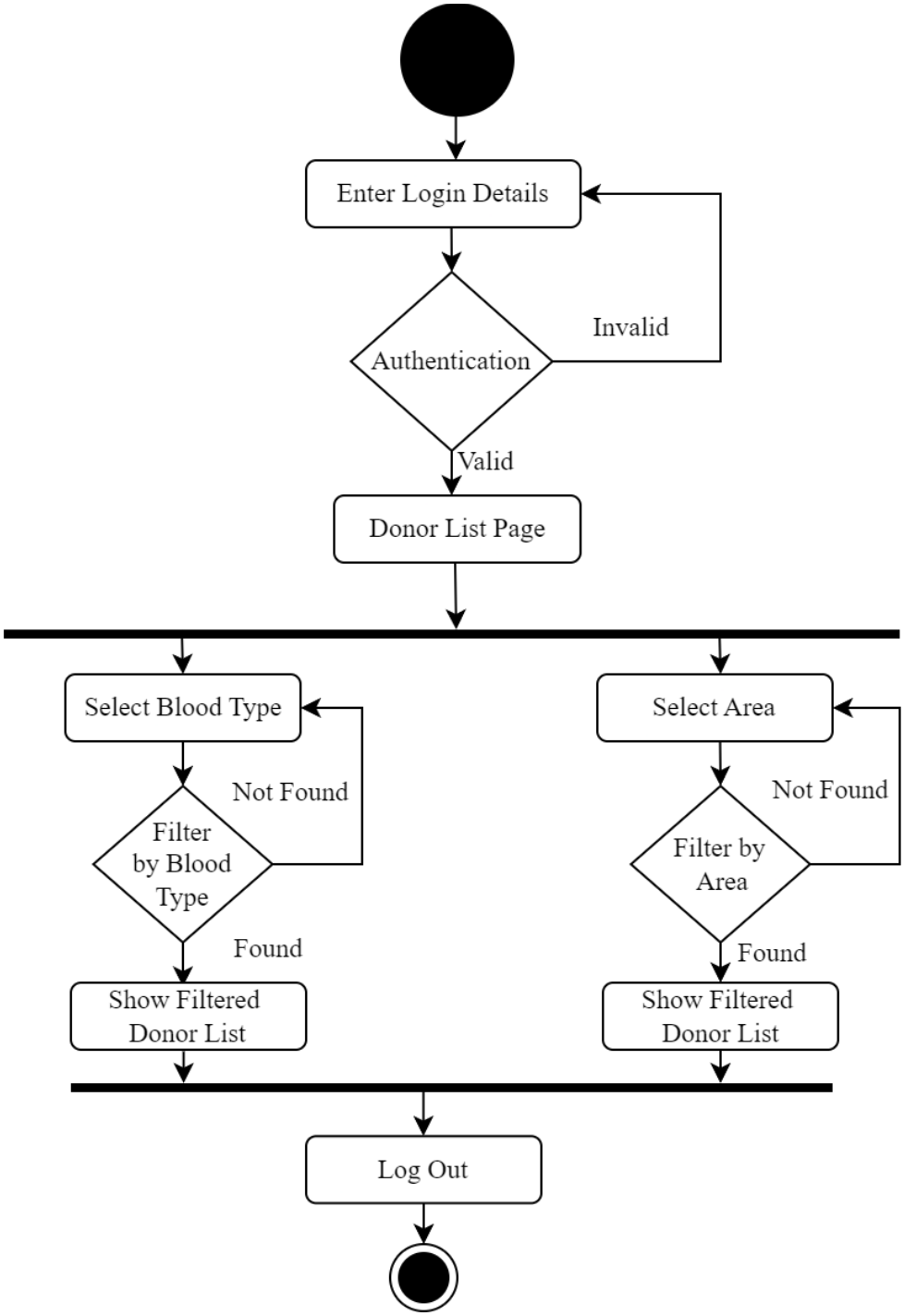

Fig.25 Activity diagram for filtering donor list

4.7.3 Activity Diagram for Sending Request to Donor

The activity diagram for the sending request to the donor is depicted in Fig.26. After authenticating user and donor will navigate to the donor list page, and two options are forked by filtering by blood type and area. Selecting the desired option will join the option and will check the availability. If not available, the user can terminate their work, else user will navigate to the matched donor profile and send the request. And send function will activate, and if any error occurs will send an error message, else if successful it will be forked into three works. Sending notifications to donors, and updating both the donation request page, and the sent request page are three jobs. After successfully finishing the tasks will join them into one and the user can log in from the application.

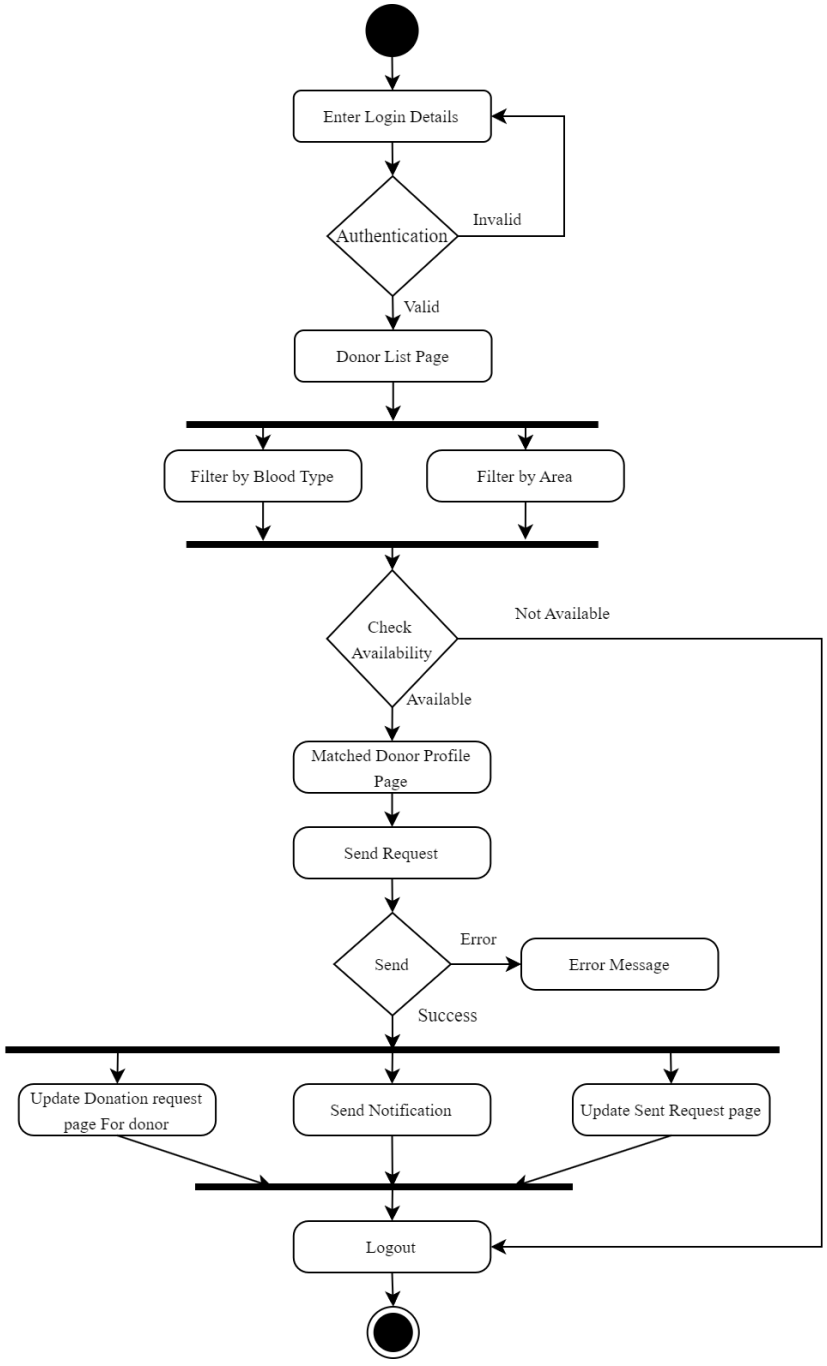

Fig.26 Activity diagram for sending a request to the donor

4.7.4 Activity Diagram for Response to Request

The activity diagram for the sending request to the donor is depicted in Fig.27. After authenticating user and donor will navigate to the donate request page. If no request is present user can log out of the application, else donor will respond to the request. If the response is accepted two forked tasks will occur. One task is to update the request status to accepted and show all the hidden information. If the response is rejected two forked tasks will occur, which is to update status to rejected and delete the request. Then both responses will join them and can terminate the operation.

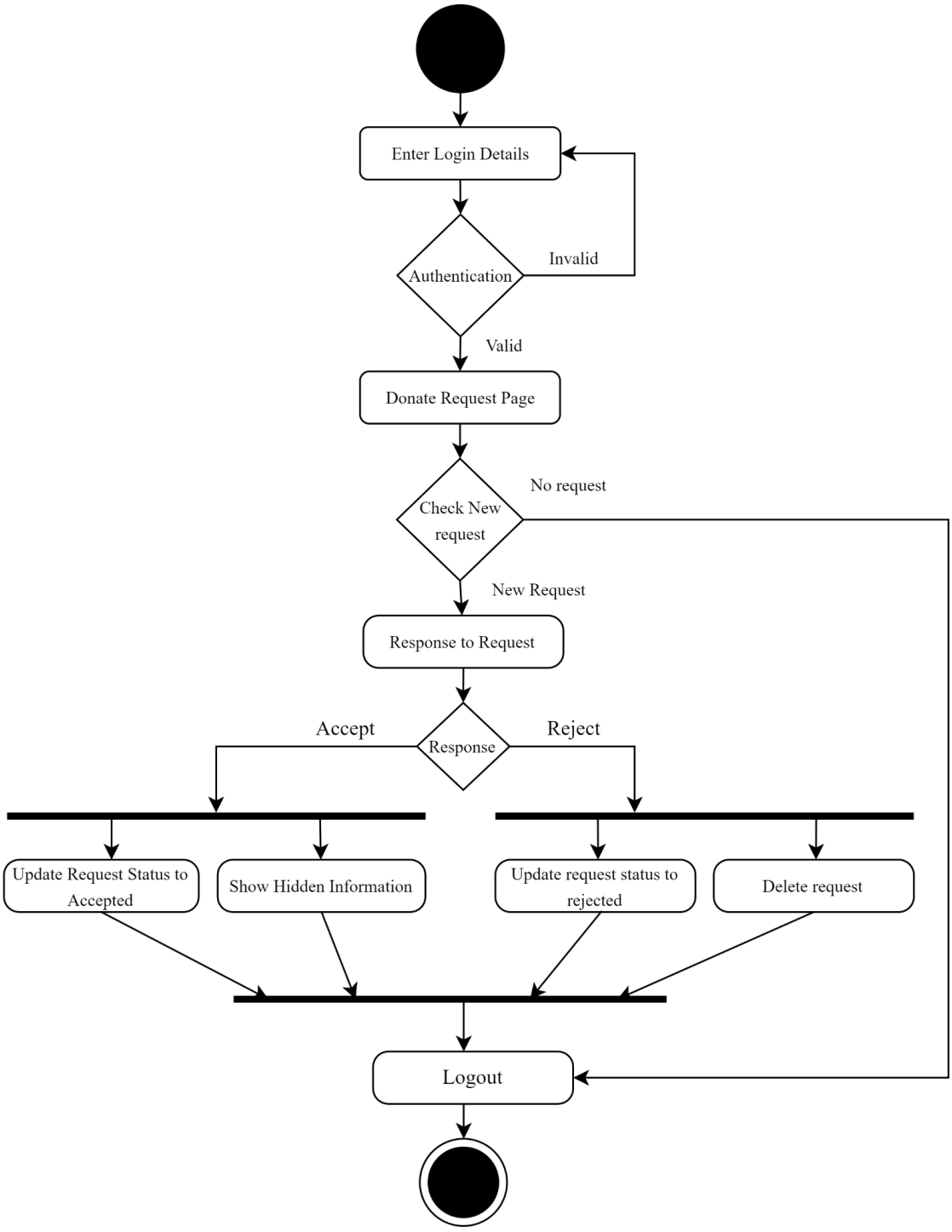

Fig.27 Activity diagram for sending a request to the donor

4.8 Sequence Diagram

A sequence diagram is a type of UML diagram that specifies how objects interact with one another in a specific order in relation to time. In order for a certain task or procedure to be finished, the messages that are transferred between objects must be arranged chronologically. These modeling diagrams can be used for debugging purposes by clearly indicating the location of message flow inside a process, or they can be used to model dynamic behavior, elaborate use case scenarios, or create system interactions.

4.8.1 Sequence Diagram for Registration

For the sequence diagram in Fig.28, the user will open the system to register, which will give a message to ask permission to register. Then the user will input all the information. Then we can see an interaction box for condition. If any error occurs or information is missing user will send a reject response and an error message. Else, the system will check the email if already registered and send validation. Here is also an interaction box inside the other condition. If the email is already in use system will send an error message and rejection. Else system will store data in the database and return success to the system. Then system will send a confirmation and a success message.

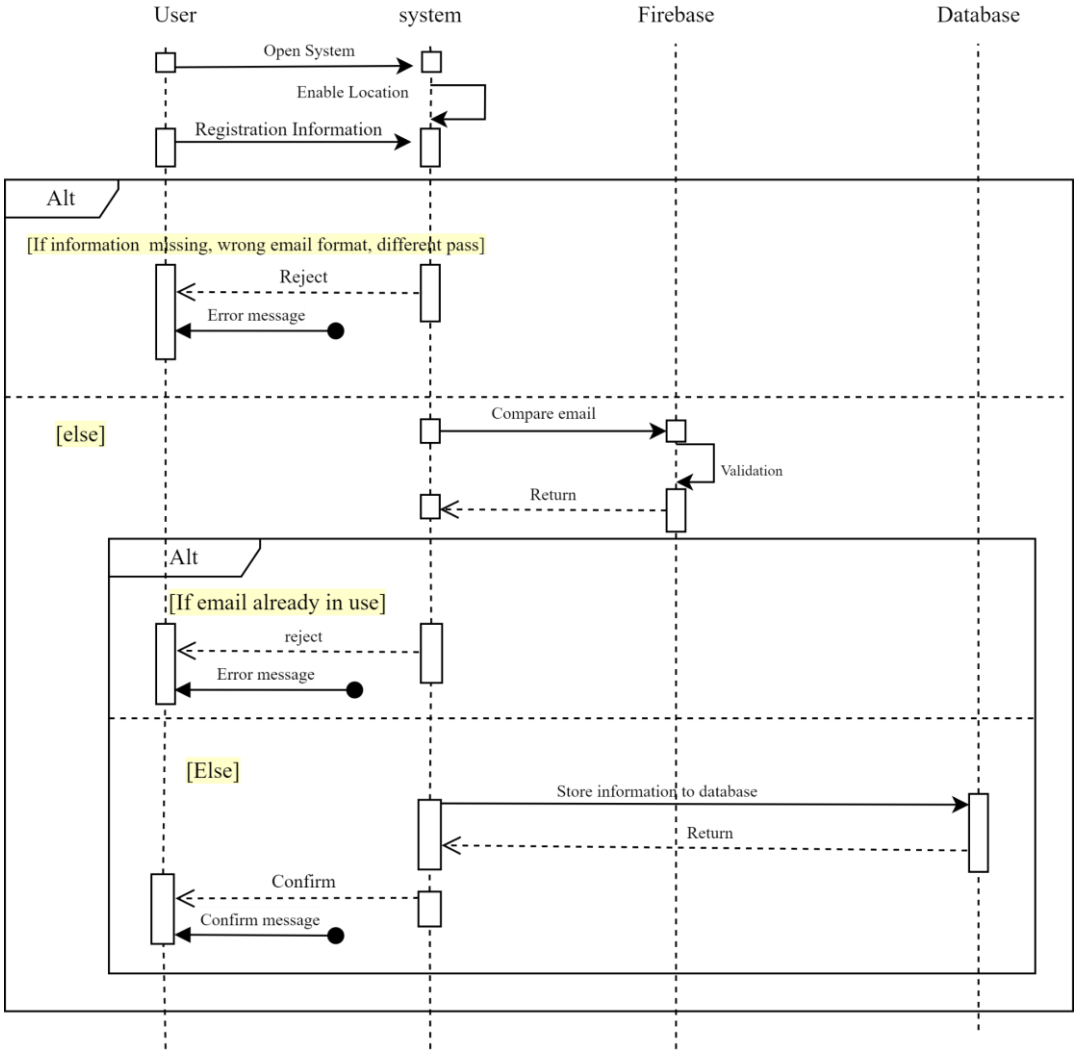

Fig.28 Sequence diagram for Registration

4.8.2 Sequence Diagram for Login

For the sequence diagram in Fig.29, the user will open the system to log in, the user will give the user email, and password, and submit. Then system will send these to Firebase to validate. After validation, Firebase will return the result. In the interaction box, if the password or email is not matched system will return reject along with an error message. Else, the system will send a request to retrieve information from the database. Then database will send a response regarding the response to the system. Then the system will send a confirm with success message to user and will redirect to the home page. Below is the sequence diagram showing all the work in sequence for the login process of the application.

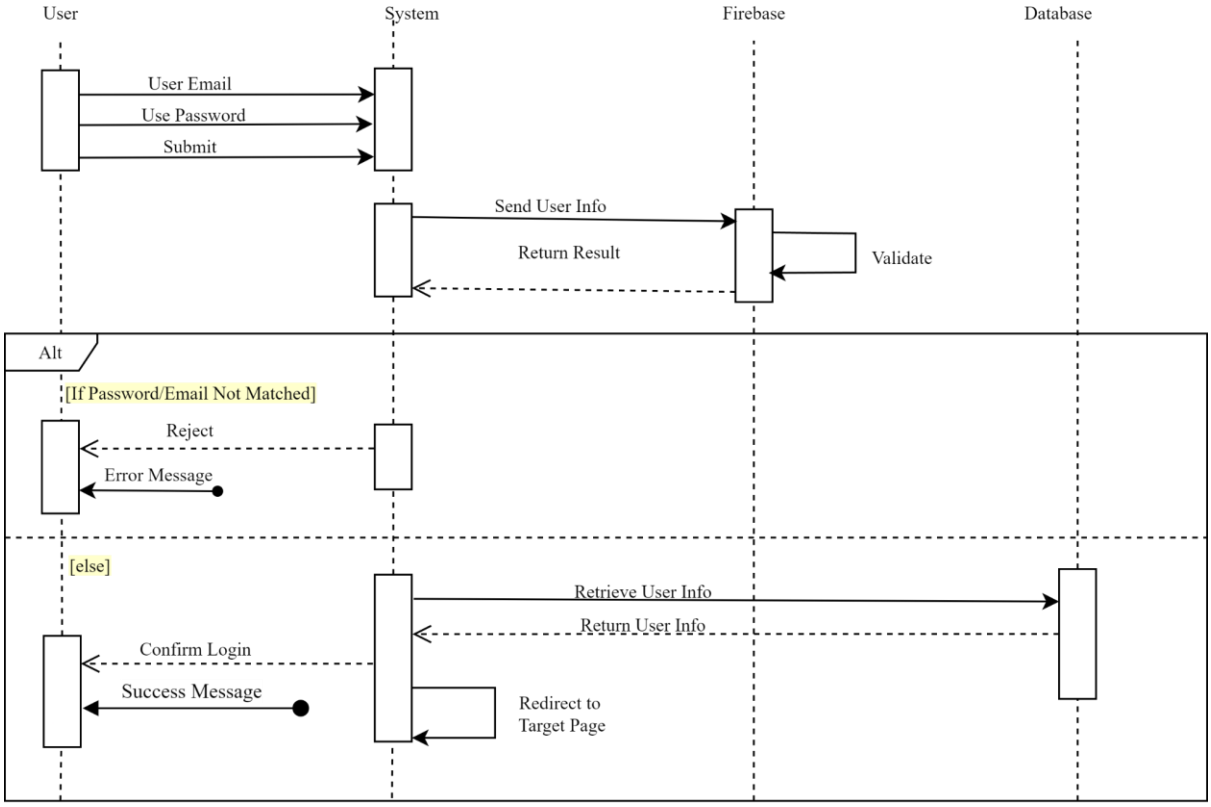

Fig.29 Sequence diagram for Login

4.8.3 Sequence Diagram for Sending Request

For the sequence diagram in Fig.30, the seeker will open the system. After validation which is described in Fig.29, the seeker will navigate to the donor list page and the system will return with the donor list of the registered users (donors). Then from the donor page user will ask for a filtered list based on selected blood type and thana. Then a response with a filtered list will be sent to the seeker. Then if available and matched with the requirements found, the seeker will open the donor profile page, and the donor profile page will be sent seeker as a response. Then seeker will send the request. Then from the sent request function a request information object will be created, and will be delivered to the database to store. Then database will send a success message with the returning response, and also sent request page will be updated with the response. Else, the seeker will just get a return response with a blank page.

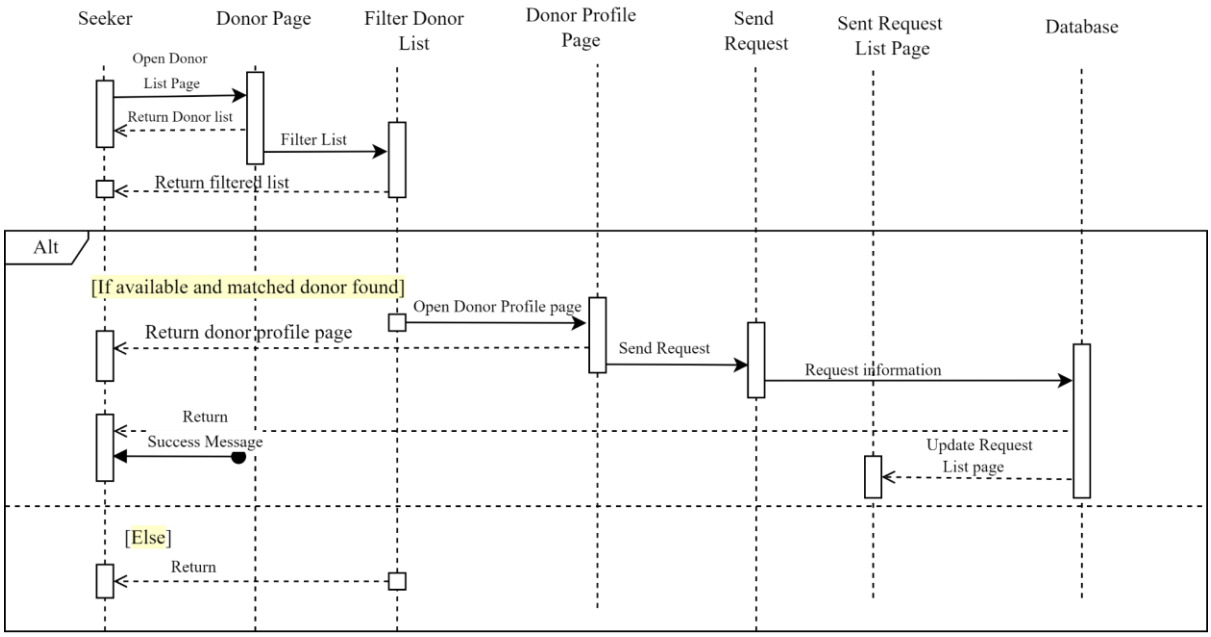

Fig.30 Sequence diagram for sending request

4.8.4 Sequence Diagram for Response to Request

For the sequence diagram in Fig.31, After validation which is described in Fig.29, the donor will navigate to the donate request page and the system will return the donate request list. If the request is available and accepted, the response function will send a request to update the request status. A confirmed response with an acceptance message will be sent to the donor. Else, donor rejects the request database will update the request status to rejected and return the response. Finally, the request will be deleted.

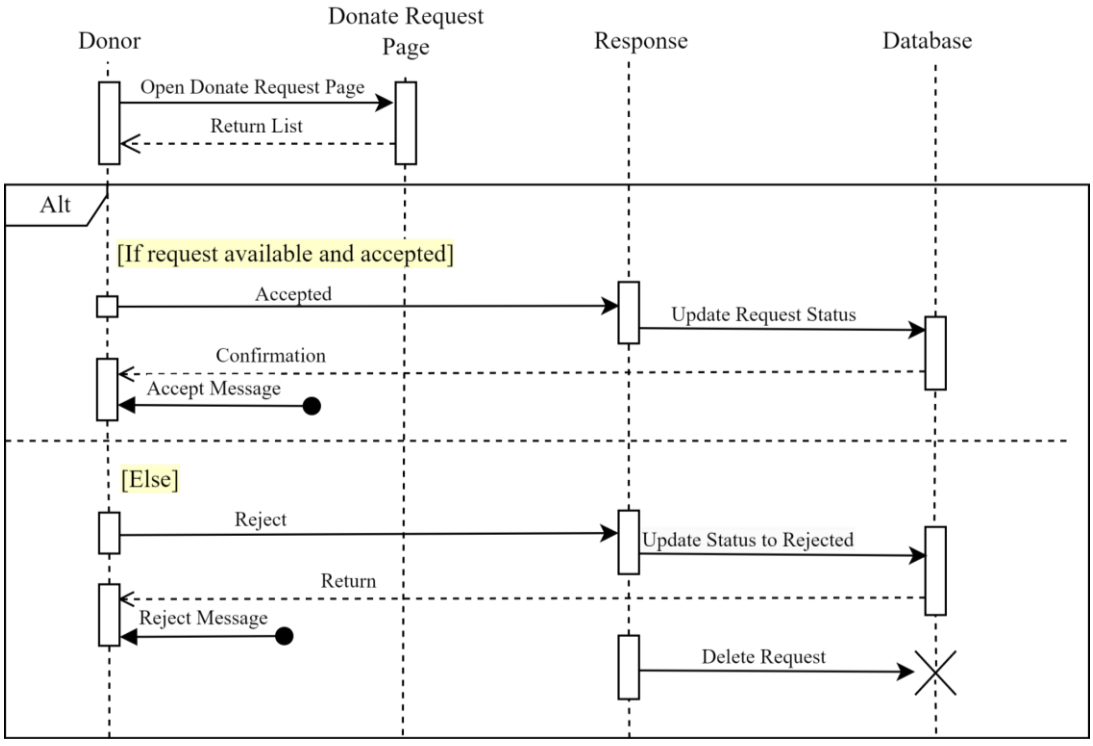

Fig.31 Sequence diagram for response to request

4.8.5 Sequence Diagram for Notification System

For the sequence diagram in Fig.32, the working of the notification system is depicted. When the user opens the web app a self message is generated asking for permission to enable the notification. In the interaction box, two conditions are performed. If notification is enabled by the user, the system generates a subscription endpoint, then generates auth by notification service, and returns them to the system. The system will send an accept message and send the information to the database to store. Else if denied to enable notification service, a reject response will be sent to the system which will send a return response with a reject message.

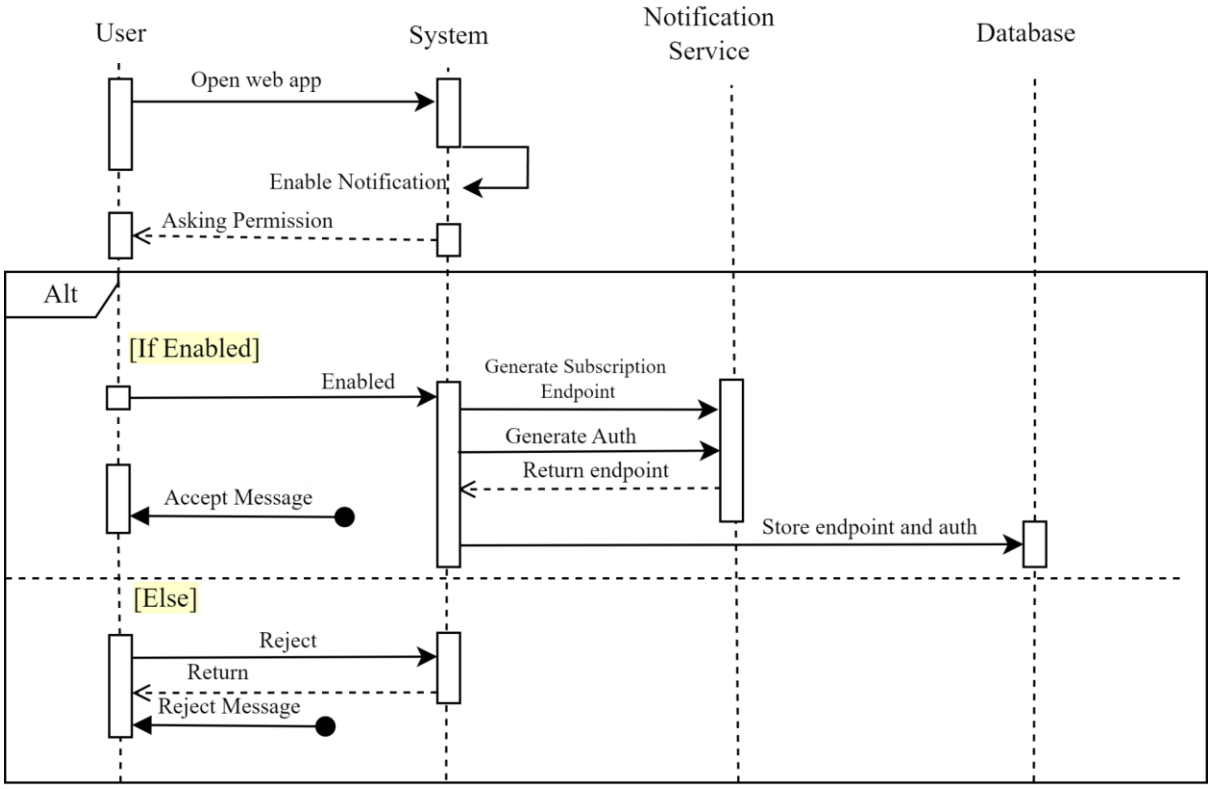

Fig.32 Sequence diagram for notification system

4.9 Swimlane Diagram

A swimlane diagram is a kind of flowchart that segregates process flow in different lanes, each reflecting the different actors departments, or systems involved in the process. This facilitates the view of who is responsible for which part of a process.

4.9.1 Swimlane Diagram for Registration

The swimlane diagram for the user registration process is in Fig.33. The user will go to the registration page. The system will get the location of the user. Then the user will fill out the registration form and submit it, which takes to the system lane to check the form. If the form is not completed it will take you back to the registration page, and show an error. Otherwise, it will check in the database lane to see if the email is already in use. If yes, then show an error message, else in the Firebase lane account will be created and return successfully to the database lane to store data while successfully returning to the system lane. Here two forked tasks will be created, which show a success message and redirect the user to the user profile.

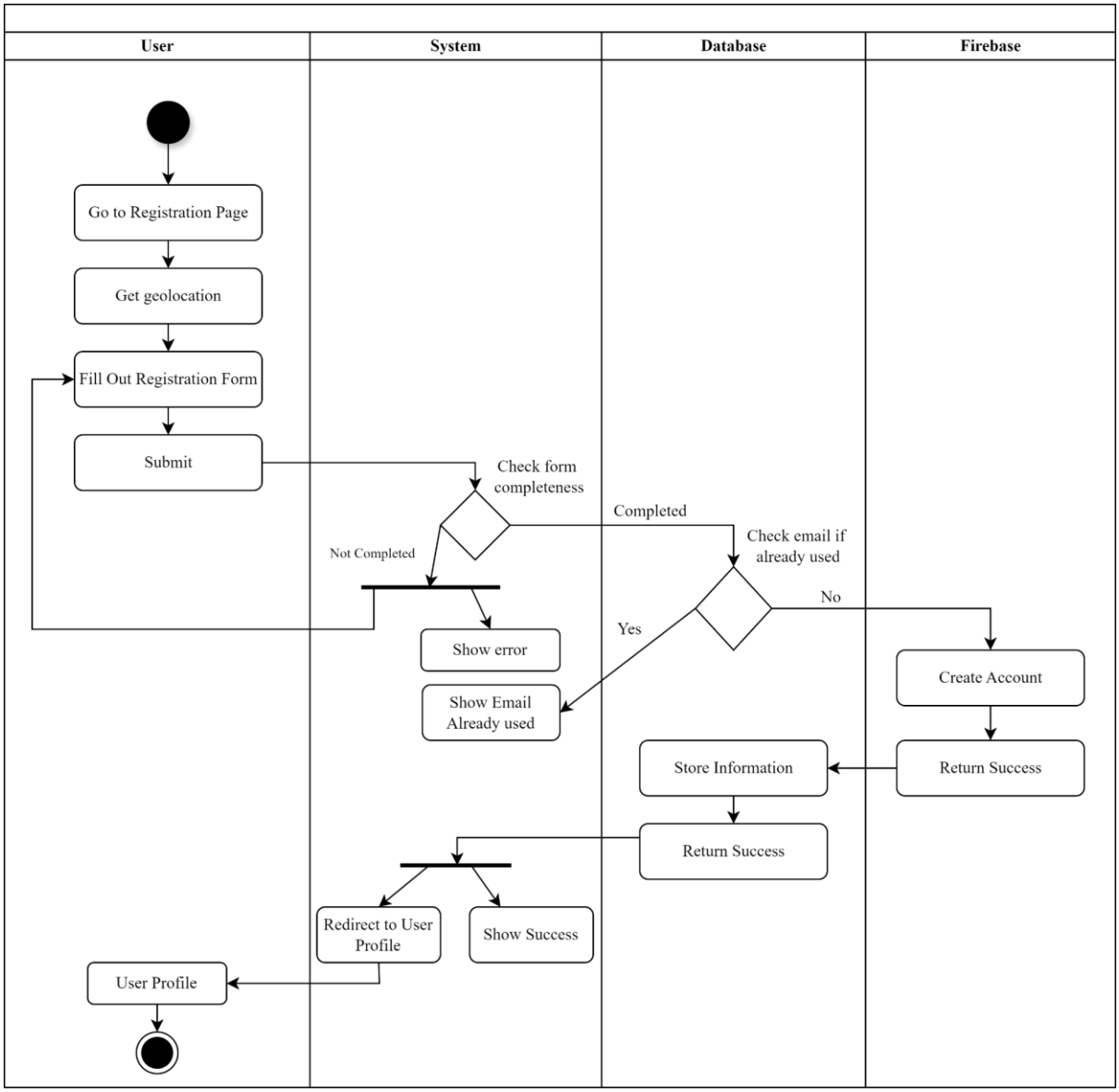

Fig.33 Swimlane diagram for registration system

4.9.2 Swimlane Diagram for Filtering Donor List

In the swimlane diagram for the filtering donor list process in Fig.34 users will enter their email and password to access into application. After entering them, in the system lane validation is done. If invalid it will again redirect to the login page, else will be redirected to the navigated page in the user lane. Then will proceed to the donor list page where the user will select the option to select blood type and area which will join and submit as criteria and will be sent to the system lane. From there criteria will be sent to the API lane. When API is hit by criteria it will fetch the information from the database lane and API after fetching them will return the fetched donor list to the system lane. In the system lane, the filtered donor list will be sent to the user to display the list. In the user lane, the user will view the list and can do other functionality.

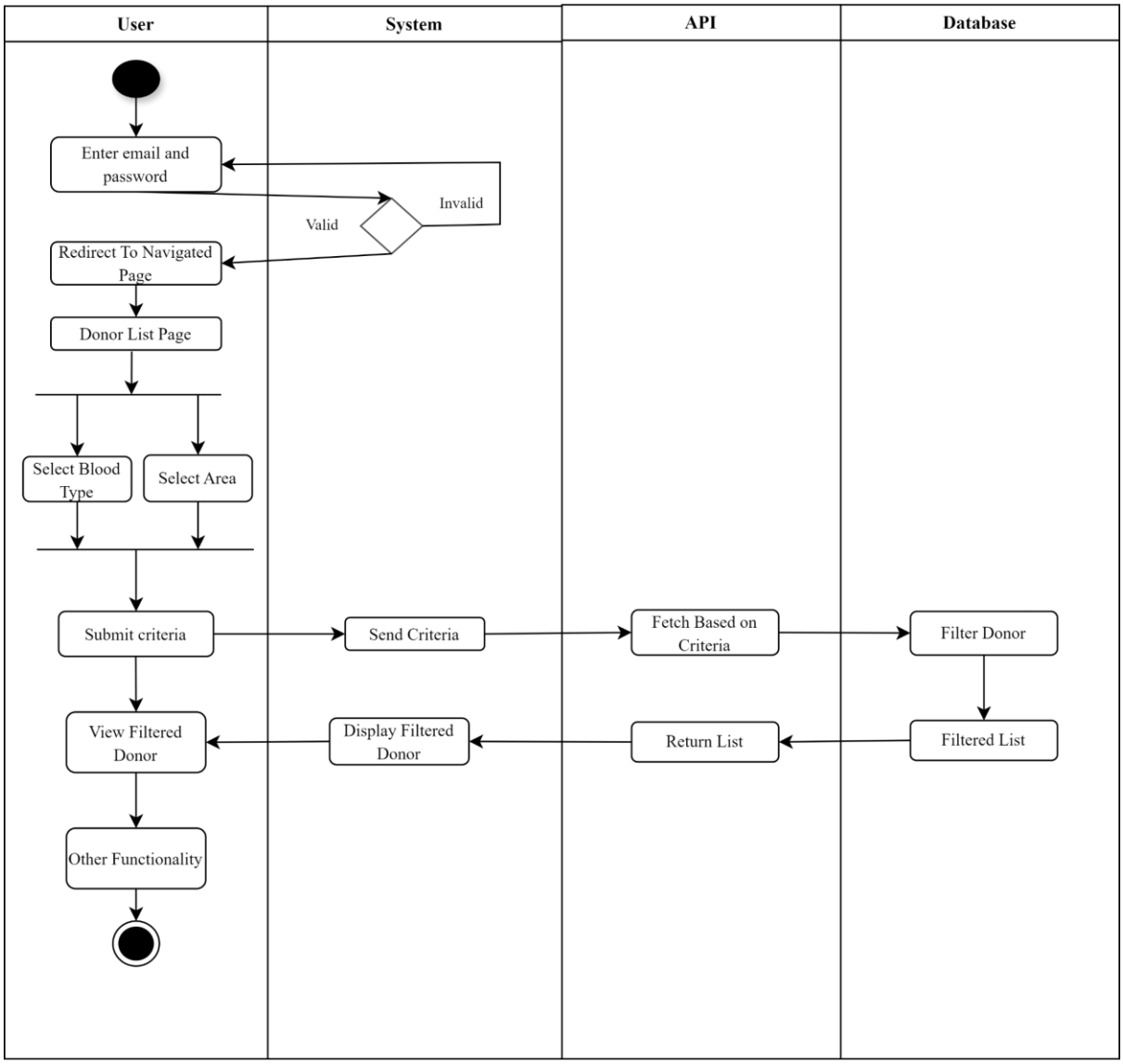

Fig.34 Swimlane diagram for filtering donor list

4.9.3 Swimlane Diagram for Sending Requests to Donors

The swimlane diagram for sending requests to donors is a complex process which is described in swimlane to be more understandable in Fig.35. The User will be validated exactly as we described in Fig.34. Then will proceed to the donor list page where the user will filter donor, we have described the process in Fig.34. In system lane, it will check donors availability. If not available it will exit function depending on user. Else, the system will send the filtered list to the user. The user will navigate to the donor profile page and send a request to the donor, resulting in two forked tasks in the system lane. Creating request data object and activating the notification process described in Fig.32. For the first task, the system will send the object to API and API will deliver them to store in the database. After storing database will return success through API, and system lane to the user. For the notification process, the system will send a request to search the subscription endpoint of the donor's device to the API, and the API will forward it database to the search endpoint. If found, the database will send it to API, and API will deliver it to the system. In the system, lane notification will be sent to the targeted endpoint and exit the function. Else not found, it will exit the function.

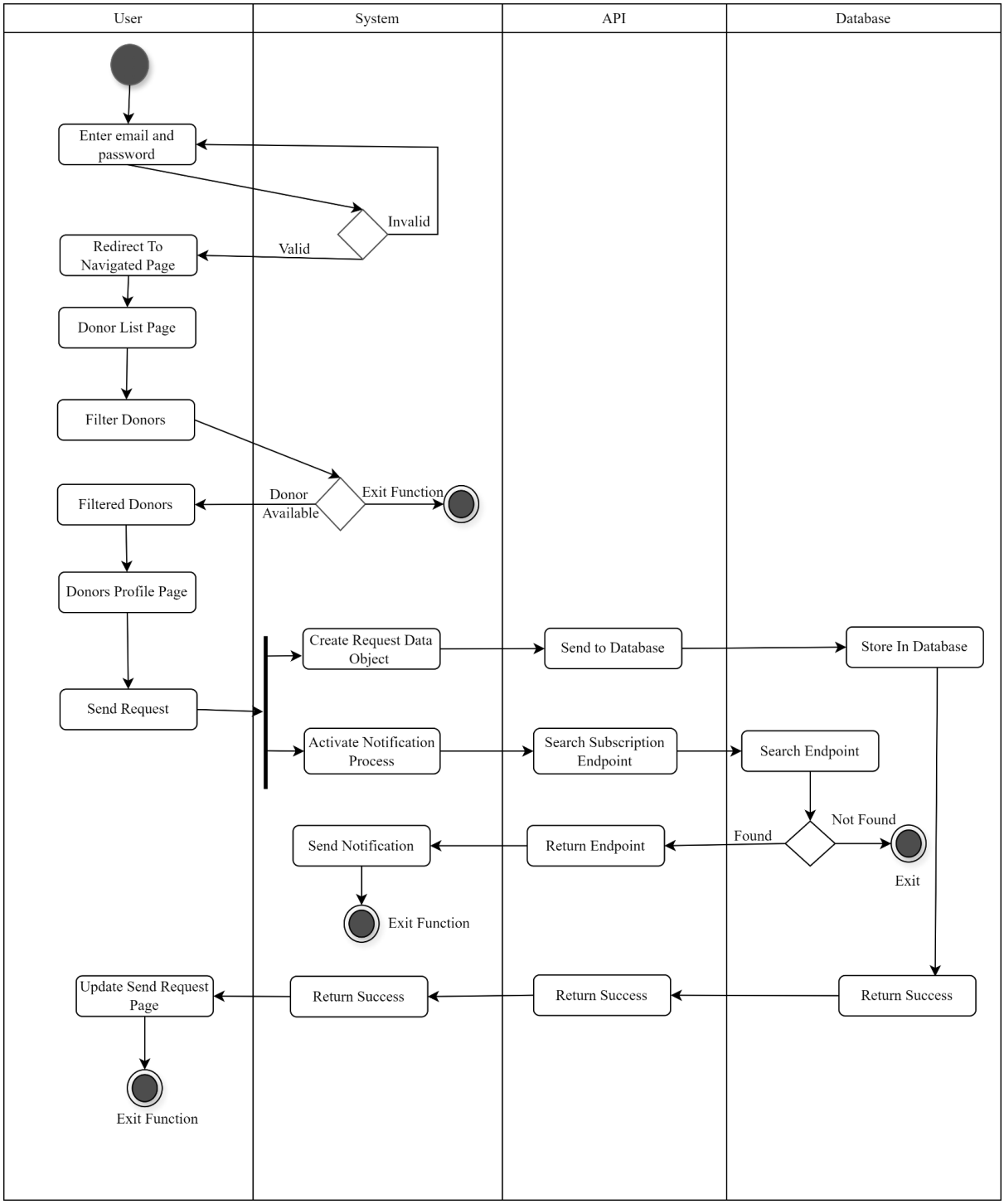

Fig.35 Swimlane diagram for sending request

4.9.4 Swimlane Diagram for Response to Request

The swimlane diagram for response to the request process which is described in swimlane is more understandable in Fig.36. The User will be validated exactly as we described in Fig.34. The User will visit to doate request page, and here request list will be checked. If no request then will exit the function. Otherwise, the user will respond to the request and send it to the system lane. The system will deliver the response to API. API will send this to the database to update the status. In the database lane, will check the response. If the response is accepted, the status will be updated to accepted and send a response to API lane, and system lane. The system

will update the request page to show hidden information. If rejected, the status will be updated to rejected and the request object will be deleted. The response will be sent to the API lane and system. The system will update the request page. Both will be joined in the user lane and an updated page will be shown.

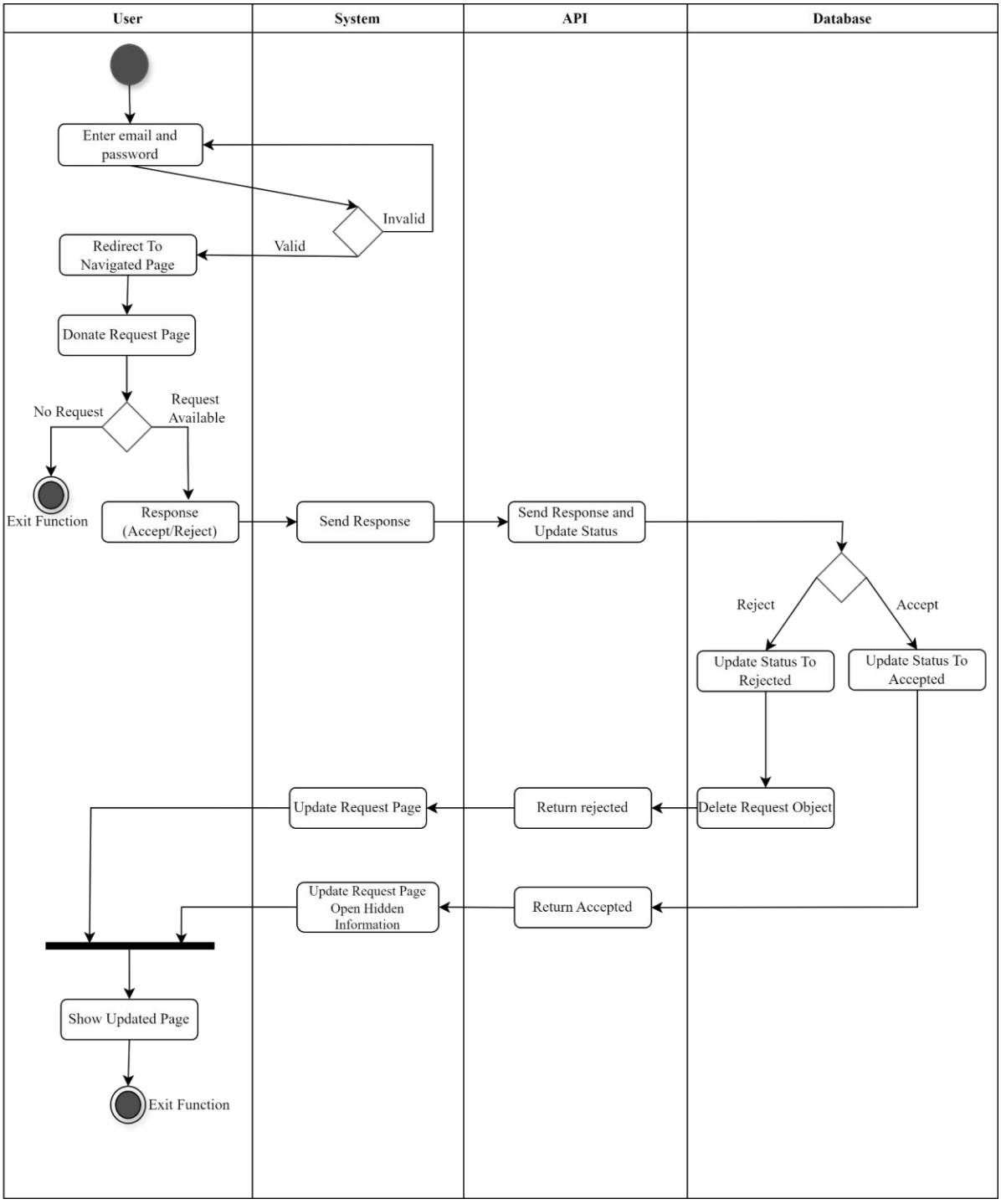

Fig.36 Swimlane diagram for response to request

4.10 Entity Relationship (ER) Diagram

ER diagrams are important in database design since they document data structures to ensure all data requirements of the business are met. In this diagram, rectangles indicate the entities, whereas ovals join the characteristics with the entities they relate to. The diamond joining the connected entities indicates the relationship.

4.10.1 ER Model Diagram for Blood Donation Web Application

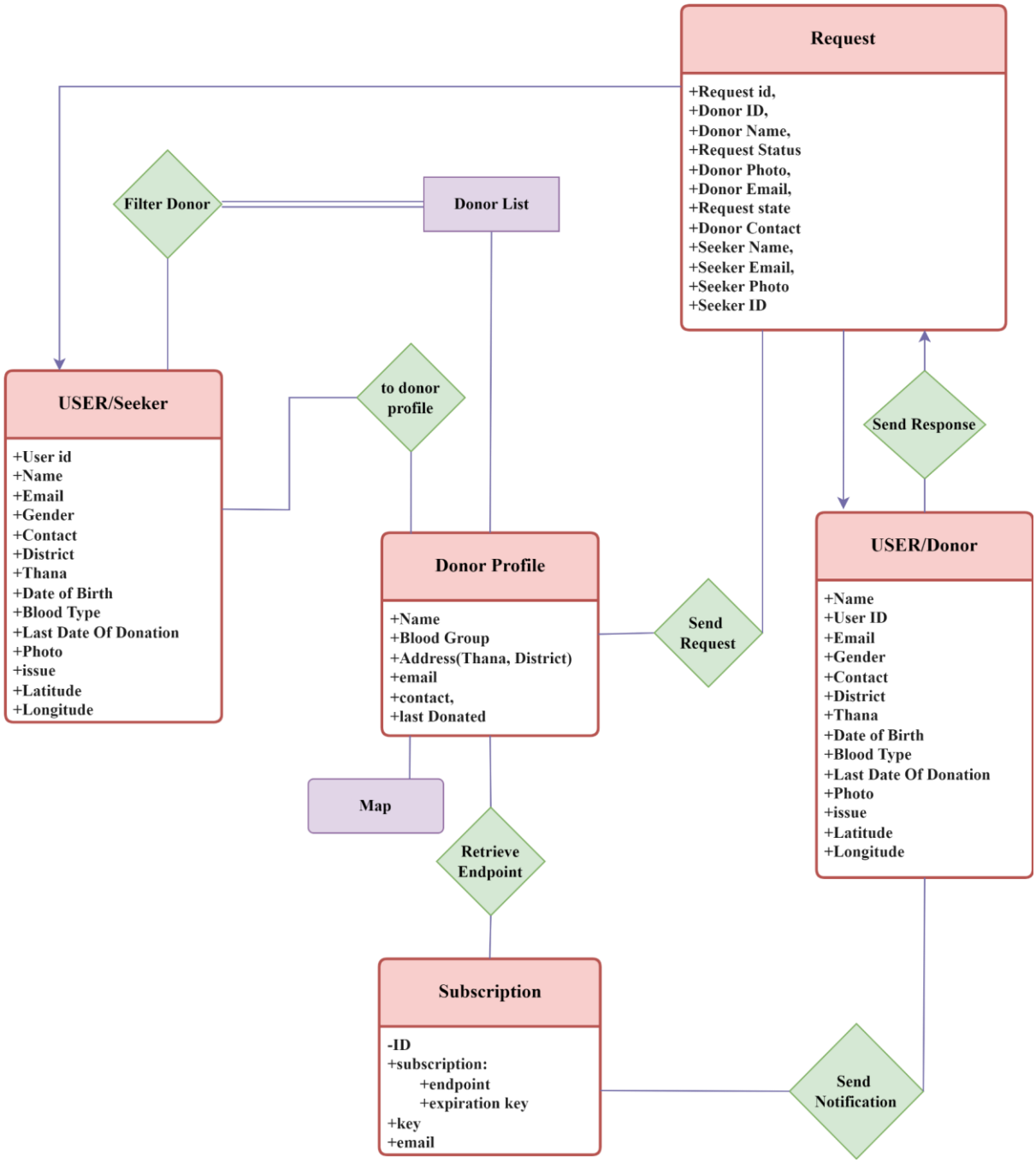

Fig.37 ER Model Diagram for Blood Donation Web Application

In Fig.37, the ER model for the application is represented where we can see all the entities' connection and their relation with other entities. As an example send request has a relationship with both donor profile and request.

4.11 Class Diagram

Class diagrams are useful in documentation and communication for software structures since they give a big picture of design. Class diagrams also form a basis that is very essential in visualizing and developing the static structure of a system, since they show the many components involved and their interaction.

4.11.1 Class Diagram for Blood Donation Web Application

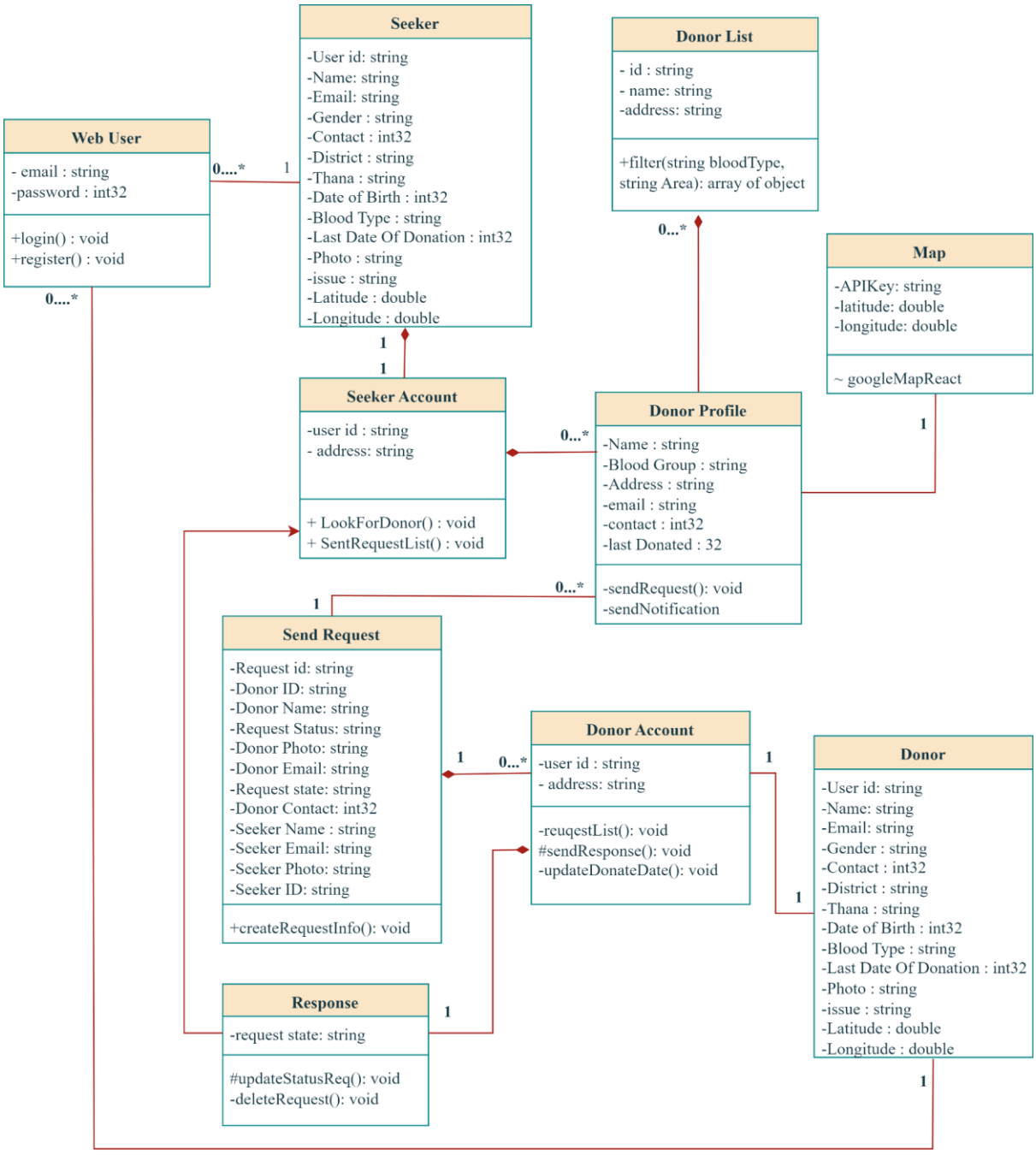

Fig.38 Class Diagram for Blood Donation Web Application

In Fig.38 class diagram for our application is represented. In the diagram, we can see that a class can contain three notations. The first one is the class name which is written on top, in the middle contains attributes representing data members of the class, and lastly, the operations the class contains.

4.12 Class, Responsibilities, and Collaborator (CRC)

CRC cards are used to define and organize class roles within an object-oriented system. Each card is representative of a class, listing the responsibilities along with other classes or objects the class will interact with. The use of CRC cards brings clarity of purpose to classes defines how classes will interact with one another and refines system design. This very simple method readily makes it possible to visualize and document class responsibilities and relationships easily.

4.12.1 CRC Cards for Blood Donation Web Application

Class : User	
Responsibilities:	Collaborators:
1. Manage user authentication (register, login)	Request
2. Update user profile information	Notification
3. Initiate and view requests	Response
4. Provide geolocation for google	Geolocation Service

Class : Geolocation Service	
Responsibilities:	Collaborators:
1. Provide location data of where registered from	User
2. Update location information for seekers	Seeker
	Donor

Class : Seeker	
Responsibilities:	Collaborators:
1. Filter donors by blood type and area	User
2. Send donation requests to donors	Donor
	Request

Class : Request	
Responsibilities:	Collaborators:
1. Create and track donation requests	User
2. Update request status (accepted, rejected)	Notification
3. Link donors and seekers	Donor
	Seeker

Class : Notification	
Responsibilities:	Collaborators:
1. Send notifications about request	User
	Request
	System

Class : Donor	
Responsibilities:	Collaborators:
1. Update last donation date	Request
2. Respond to donation requests	Response
	User

Fig.39 CRC Cards for Blood Donation Web Application

In Fig.39, the main class that is most essential is represented in the CRC card containing the responsibilities of the classes, and collaborators related to the class are depicted.

4.13 Conclusion

This Software Requirements Specification (SRS) paper concludes by outlining the technical requirements, system architecture, and important capabilities for the creation of our web application for blood donation. The application seeks to improve efficiency and efficacy in both routine blood donation activities and emergency scenarios by streamlining the process of matching blood donors and seekers. The SRS has outlined all of the necessary elements, such as notification protocols, geolocation capabilities, and user responsibilities, which when combined create a unified system intended to make it easier for donors and seekers to engage with one another. Through the utilization of contemporary online technology and strong data management procedures, the application guarantees a safe and intuitive user experience, encouraging increased blood donation rates.

Chapter 5: Software Testing And Analysis

5.1 Introduction

Software testing is the last stage of specification, design, and coding review and is a crucial component of software quality assurance. The only phase in the software engineering process that may be seen as destructive as opposed to constructive is testing. A software testing strategy incorporates software test case design techniques into a methodical sequence of actions that lead to the effective development of software. The group of tasks known as testing can be organized ahead of time and carried out methodically. Program testing's primary goal is to validate software quality using techniques that are both strategically and practically applicable to large and small-scale systems.

5.2 Implementation of Testing

5.2.1 Process of Testing

The main goal of the testing procedure for the Blood Donation Web Application is to find out the system defects and eliminate them. Testing is also important for the improvement of the application, making it less erroneous and more reliable so that it may infuse confidence in the potential users of the application. However, since a broad range of input values and user interactions happen in this case, conducting exhaustive testing will not be practical. Instead, great effort is put into evaluating the application critically, to identify any flaws and to document a view of the conditions under which failures resulted. This information is precious for improvement and rectification so that the system works as expected and serves the users effectively in the process.

Process Conducted in The Testing

Registration: First users must be registered. A user can be both a donor and a seeker. The user has to fill up the registration form completely. If any error is found such as wrong email format, mismatch of password, or incompleteness of any part will give an error. Otherwise, the donor will be registered to the application.

Login: Without authorization or in simple terms only registered users can have the facility of the application. The security features are provided by the system through email and password matching by Firebase.

Filter Method: By selecting blood type, area or both from the options user can filter the donor list. If the donor of the matched query is available will display the donors or it will be a blank page. Also, the donor list is sorted in descending order by the difference between the last donation date and the current date.

Donor Interval: An interval period for donors of 90 days is implemented. If a donor has donated blood and the time is less than or equal to 90 days seeker can not send a request as the donate request button will be disabled at this time.

Sending Request: If a donor is available, a seeker can send a request. A request object with donor and seeker information will be created. The sent request page will be updated with a list of requests. However, the major information of donors will be hidden but if the donor accepts the request it will be visible to a seeker. When a request is sent a notification will also be delivered to the donor.

Response to Request: If a donor accepts the request, the request status in the database will be updated in the database and the hidden information will be visible to a seeker. But if rejected, the status will be updated to rejected and the request will be deleted.

Update Donate Date: If a donor wants to update the donate date of his blood donation a button in the user profile is given. Clicking the button will update the date to the current date.

Geolocation and Map: While the registration system will ask permission for location, if permission is given it will take latitude and longitude from the user and update in registration form data. In the donor profile, Google Maps is integrated.

5.2.2 Report From The Testing

TABLE XV. Testing Result From the User Registration

Test: Registration of the user				
Test No.	Situations	Prospective Result	Real Outcome	Result (✓,X)
Test 1	Submitting with the incompleteness of form input.	The system doesn't allow the user to submit and shows an error message "Please fill out all the input" in the alert.	As prospective.	✓
Test 2	Submitting with the mismatched password.	The system shows the error message "Password did not match".	As prospective.	✓
Test 3	Submitting with the wrong format of email.	The system shows the wrong email format message.	As prospective.	✓
Test 4	Click submit with all the information correctly.	The system registers the user, shows the "Registration Successful" message, and redirects the user to the home page.	The system allows the user to access the features of the application and shows a success message.	✓

To test the user registration system, four types of testing were done, which are described in the above TABLE XV. In the first test, form submissions with incomplete input fields returned an error message. The second test involving mismatched passwords returned an error message as expected. The third test, involving an incorrect email format, produced an appropriate error and did not create a user. In all of these, no new user was created in Firebase and there was no storing of user data in the database. In the fourth test, upon correct filling, the form returned a successful registration message, and a new account appeared in Firebase with user information stored in the database, which evidenced the proper working of the registration process.

TABLE XVI. Testing Results From the User Login

Test: Login of the user				
Test No.	Situations	Prospective Result	Real Outcome	Result (✓,X)
Test 1	Submitting with the incompleteness of form input.	The system doesn't allow the user to submit and shows an error message "Please fill out this field" in the alert.	As prospective.	✓
Test 2	Submitting with the wrong email or password.	The system shows the error message "Email or password did not match ".	As prospective.	✓
Test 3	Click submit with all the information correctly.	The system registers the user, shows the "Login Successful" message, and redirects the user to the home page.	The system allows the user to access the features of the application and shows a success message.	✓

The user login system was also tested to find if any flaws were present in the system, three types of situations were brought out in testing, which are described in the above TABLE XVI. In the first test, form submissions with incomplete input fields returned an error message. The second test involving mismatched passwords or emails returned an error message as expected. In all of these, no new user was given access as no similar user email or password was found in Firebase. In the third test, upon correct filling, the form returned a successful login message, as Firebase authenticated the account. Finally, the user was given access to the application and used the functionality of the application.

TABLE XVII. Testing Results From the Filter Method

Test: Filter Method				
Test No.	Situations	Prospective Result	Real Outcome	Result (✓,X)
Test 1	Selecting different blood types and thana from options.	Display the matched donor in sorted order. Selecting doesn't require reloading.	Display donors by the matching query.	✓
Test 2	Selecting blood type and thana from options that have not registered.	A blank page is displayed with the navbar, footer, and filter options.	As prospective.	✓
Test 3	Selecting both blood type and thana.	Display the donor who has matched both blood type and thana.	As prospective.	✓

Three test cases were conducted for the functionality of the filter method which is depicted in TABLE XVII. First, the result shows that selecting different blood types and locations correctly shows matched donors in sorted order without reloading the page upon selection. The second test was to select a blood type and location that had no registered donors. This returned a blank page, including the navbar, footer, and filter options, which is also in line with the projected result. Lastly, the third test proved that choosing both blood type and location satisfactorily returned donors matching both criteria, meeting anticipated results. All tests passed, proving the reliability of the filter method.

TABLE XVIII. Testing Results From the Interval Period of Donor

Test: Donation Interval of Donor				
Test No.	Situations	Prospective Result	Real Outcome	Result (✓,X)
Test 1	Visiting a donor profile who is not available.	The sending request option will be labeled as “Not available” and the button won’t work.	As prospective.	✓
Test 2	Visiting a donor profile who is available.	A sending request button will be available.	As prospective.	✓

The following are some tests on the donation interval of donors to ensure that it is working as it should depicted in TABLE XVIII. For instance, in case of a visit to a donor profile marked not available, the sending request option was correctly labeled "Not available" and the button disabled, matching the anticipated result. In the visit of a donor profile marked available, there was an active sending request button as expected. Both tests passed because the system maintains accurate availability status for the donors and correspondingly enables or disables the request button based on a given donor's availability.

TABLE XIX. Testing Results From Sending Request

Test: Sending Request				
Test No.	Situations	Prospective Result	Real Outcome	Result (✓,X)
Test 1	Visiting a donor profile who is not available.	The sending request option will be labeled as “Not available” and the button won’t work.	Can not send a request as the send request button is not working.	✓
Test 2	Visiting a donor profile who is available.	A sending request button will be available and a request can be made and the button will be disabled with the label request sent. The sent request page will be updated with the request list.	Request sent and a message “Request sent successfully” shown, button disabled and labeled sent request. Sent Request list page updated.	✓
Test 3	After sending the request Sent Request page update.	The page is updated with proper information with the donor name, donor profile page link, and hidden information labeled as “Please wait till accepted”.	As prospective.	✓

The send request functionality has been tested in three different scenarios to ensure that it works as expected. In the very first test, visiting a donor's profile who was marked as not available correctly made the send request button disabled with a view "Not available", thus not allowing any request to be sent. The second test was to visit an available donor's profile, which turned on the send request button. Tapping this sent the request, circulated a "Request sent successfully" message, and changed the button label to "Request sent." This also updated the sent request page with the new request. In the third test, it was checked that the sent request page did include the name of the donor, the link to his profile page, and the message to wait until the request is accepted. All tests passed, ensuring that the send request feature was working as expected. The full testing and outcome are described in the above TABLE XIX.

Testing of the response to request functionality was done in two scenarios to prove its performance described in below TABLE XX. First, when a donor accepts a request for blood donation, "Request accepted" will appear in the message, the request status will change to "Accepted," and all hidden information will become visible to the seeker. The status had changed on the sent request page to "Accepted," and now the hidden information is accessible. In the second test, where a donor rejected a blood donation request, the "Request rejected" message was displayed and the request was removed from the accounts of both the donor and the seeker. These tests were carried out, bringing out expected results that prove the response to the requested feature works correctly.

TABLE XX. Testing Results From Response to Request

Test: Response to Request				
Test No.	Situations	Prospective Result	Real Outcome	Result (✓,X)
Test 1	A blood donation request is available on the request page. Donor accepts the request.	The donor gets a message “Request accepted ”. The request status is updated to “Accepted”. All the hidden information will be visible to the seeker.	In the sent request page, the request status is changed to accepted. The hidden informations are visible now.	✓
Test 2	A blood donation request is available on the request page. The donor rejects the request.	The donor gets a message “Request rejected ”. The request is deleted from both users.	As prospective.	✓

Sending notification functionality was tested in two ways which are depicted in TABLE XXI. In the first test, when a request was sent to a donor who did not enable this service, the system did not send the notification—clearly due to the absence of a subscription endpoint. However, the donate request page was updated, and the sent request page also got updated; moreover, the donor responded to the request. This was expected. In the second test, when a request was sent to a donor with the notification service on, the application sent a notification to the subscribed device with the message "Someone requested for blood donation" and an application logo. The notification was processed successfully. Both tests proved that the functionality works as expected.

TABLE XXI. Testing Results From Sending Notification

Test: Sending Notification

Test No.	Situations	Prospective Result	Real Outcome	Result (✓,X)
Test 1	Sending a request to a donor who did not enable notification service.	The subscription endpoint is not available so notification won't be sent to the donor. But the donate request page and sent request page will be updated. Donors can still respond to the request.	Did not get any notification. The donate request page updated with a new request for blood donation.	✓
Test 2	Sending a request to a donor who did not enable notification service.	A notification to the subscribed device will be sent with the message "Someone requested for blood donation" with the application logo.	Received notification with application logo of Donate Red and the message.	✓

Updating donate date testing was also done, which is depicted in TABLE XXII. In the first test case, it was checked that the "Update Donate Date" button was disabled for a user who was still within the 90-day interval from his or her last donation. In the second test, a user donated blood outside the interval and updated the donation date to the current date; the button was disabled as expected.

TABLE XXII. Testing Results From Update Donate Date

Test: Update Donate Date				
Test No.	Situations	Prospective Result	Real Outcome	Result (✓,X)
Test 1	The user is in 90 90-day interval period from the last donation.	Update Donate Date button in the user profile is disabled.	Can not click the button.	✓
Test 2	The user donated blood, not in intervals, and wants to update the donation date.	The donate date will be updated to the current date. The button will be disabled. Donors will be in intervals and not available.	As prospective.	✓

The geolocation service functionality was tested in two scenarios which are described in TABLE XXIII. In the first test, when a user did not accept the location service during registration, the integrated Google Maps in the donor profile displayed a blank page, as no latitude and longitude data were provided. This was the expected outcome. In the second test,

when a user accepted the location service, Google Maps successfully displayed the user's location on the donor profile page, as the system received and used the location data from registration. Both scenarios produced the expected results.

TABLE XXIII. Testing Results From Geolocation Service

Test: Geolocation Sevice				
Test No.	Situations	Prospective Result	Real Outcome	Result (✓,X)
Test 1	While registration user does not accept location service.	In donor profile integrated Google Maps will be blank as no latitude and longitude is sent.	Google Maps shows a blank page beside donor information in the donor profile.	✓
Test 2	While registration user accepts the location service.	In donor profile integrated Google Maps shows the user's location of the user which received from the registration process.	Google Maps shows donors' location on the donor profile page beside donor information.	✓

The testing was done thoroughly and got our expected result. Every module and feature of the system was tested to ensure that it worked perfectly. All parts of the system, starting from user authentication and donor management to processing blood donation requests, worked well. The testing also involved the changes made to the database and how it was being accessed. The system accounted for all changes in the data, whether new user registration updates to the donor info or any requests for a donation. Simulation tests ensured that the application not only complied with the specified requirements but also maintained the integrity and consistency of data throughout all operations.

5.3 Developed Prototype

The Blood Donation Web Application fulfills all of the project's objectives by providing a robust, user-friendly blood donation management platform. It offers an intuitive and secure user interface with efficient user authentication and detailed donor and seeker interaction abilities in the final product. Real-time information on the available donors, donation requests, and personal profile details are some of the features. We have created the prototype in accordance with our design methodology.

This is how the application's home page will seem to every user who visits it, as shown in Fig. 40. When a user enters the application, it will be shown as the home page. Both registered and nonregistered users can see the home page. On the home page, we tried to keep it minimal, and it has three sections along with a navbar and footer. The first section is to welcome the users,

the second section is a slider for giving some information about blood donation, and the last section is the pictures of some donors.

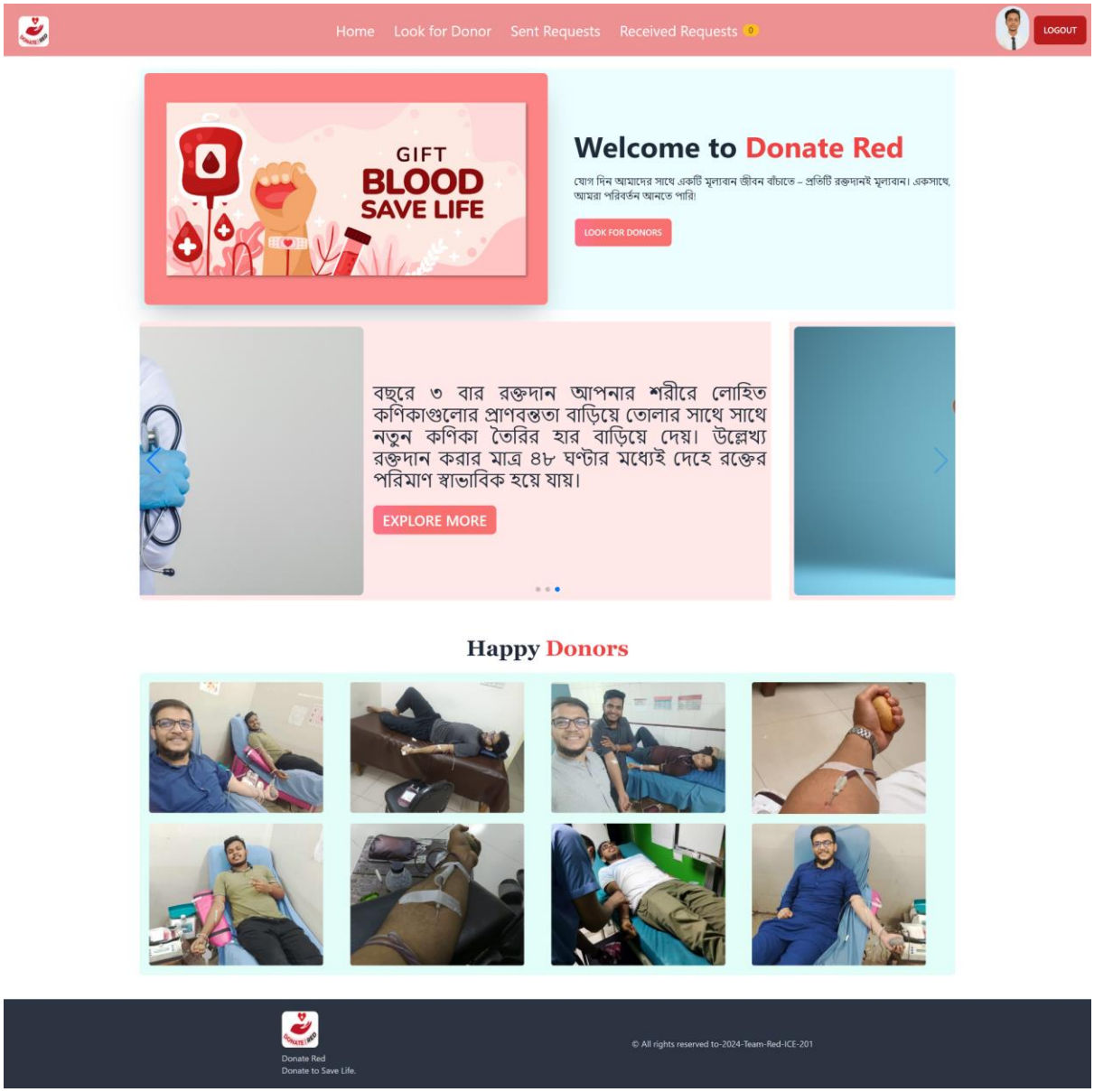

Fig.40 Front page of the web application

To use the facilities and functionality user should be registered to the application. The Fig.41. is the first page of the three pages from the registration form. On this page, the user should provide their name, email, and password, confirm the password, and upload a picture.

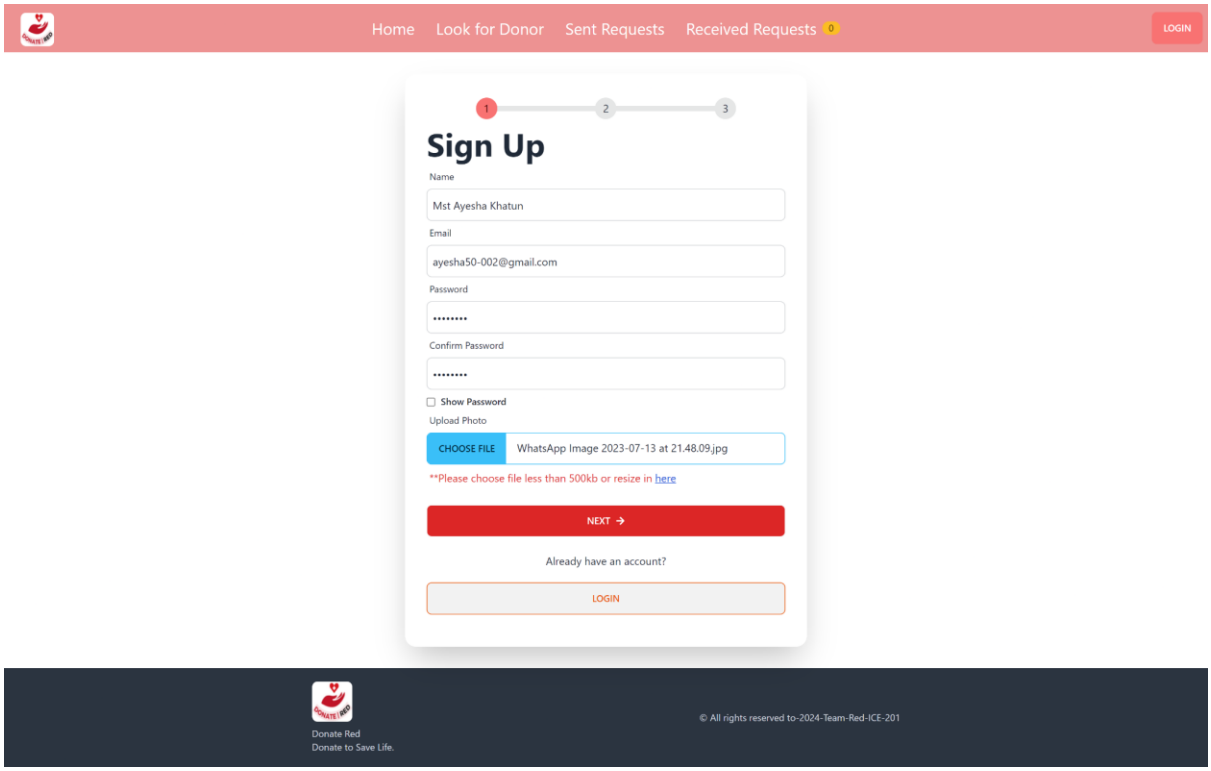

Fig.41 First page of sign up form

After clicking next, it will take the user to the second page, which is like Fig.42., here contact, gender, blood type, and date of birth should be filled in to move to the next page of the application.

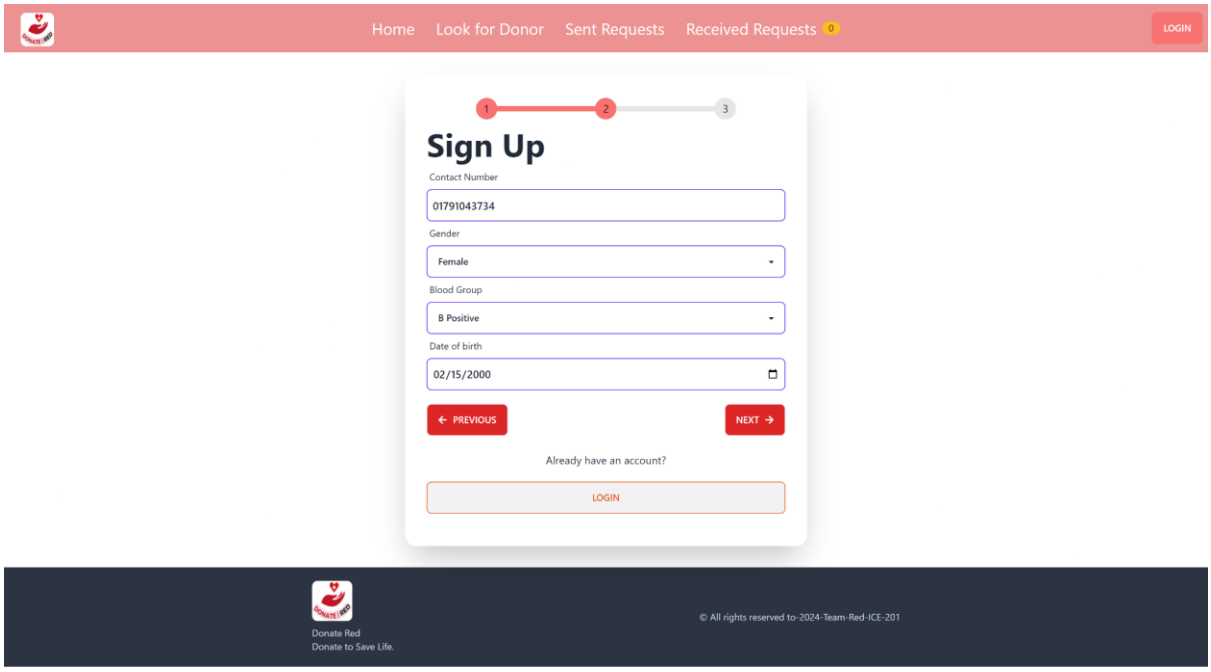

Fig.42 Second page of the web application

The next page Fig.43. is the final page of the three pages where district, thana, last date of donation, and if any issue related to donating blood should be mentioned. Upon clicking in the

submit if any option is missed it will show an error message, else it will take a few seconds to upload the picture to imgBB cloud, generate a link, and authenticate the user.

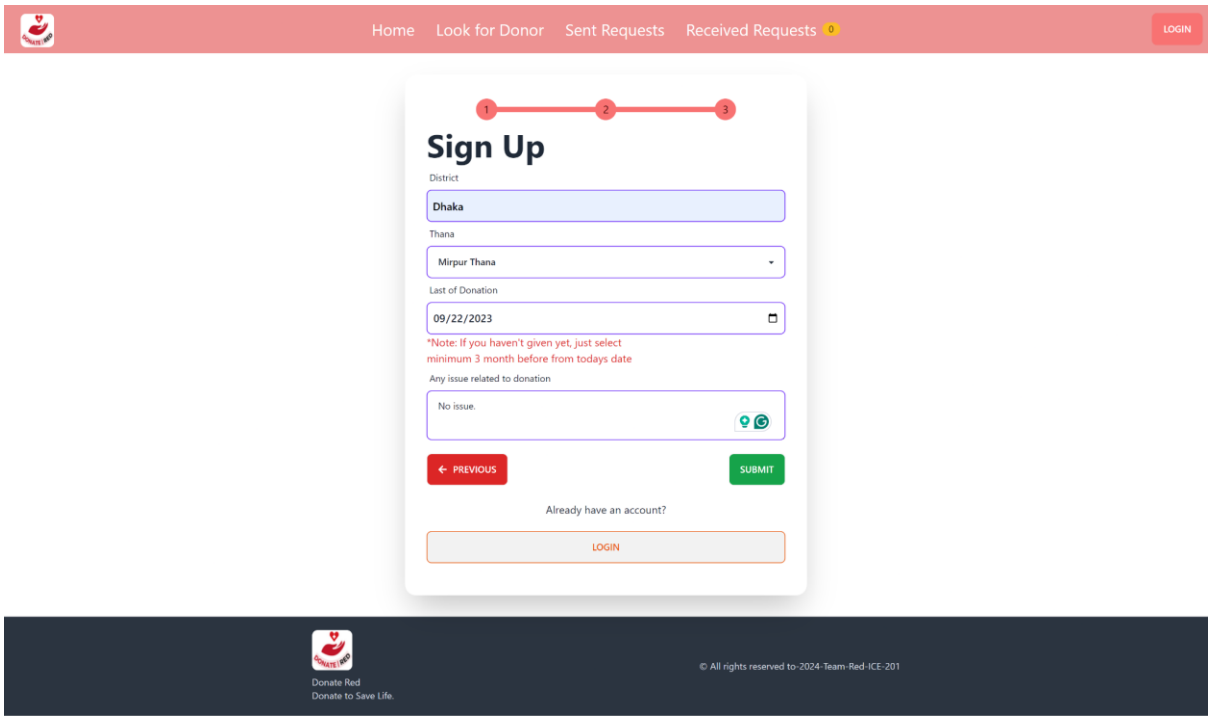

Fig.43 Final page of the web application

In Fig.44 the login page is shown. Here user will provide the email and password that he used at the time of registration.

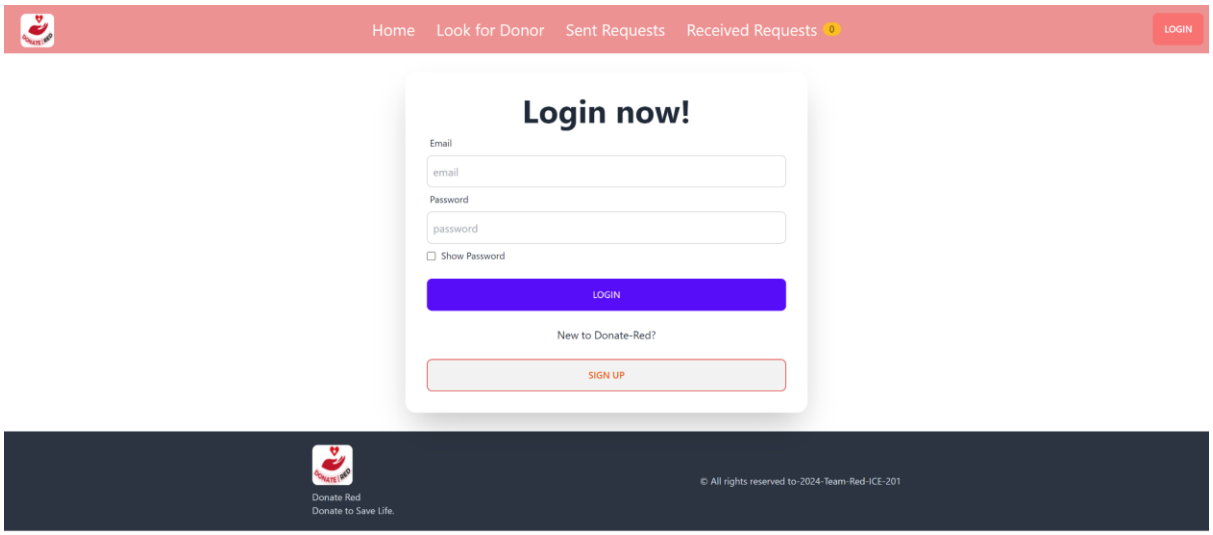

Fig.44 Login page of the web application

In Fig.45, we can see that if a user provides the wrong email or password, it will show an error message of "Email or password did not match". Then in Fig.46 user provides the correct email and password. The user was redirected to the home page and a message of successful login is displayed on the home page. A user profile for the registered user and a button to update the last donate date are shown in Fig.47.

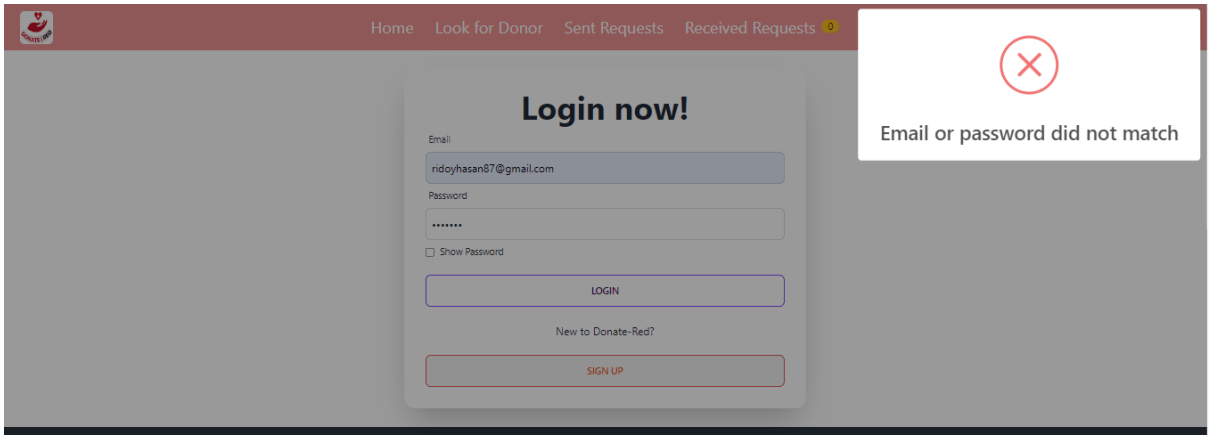

Fig.45 Error on wrong password or email of the web application

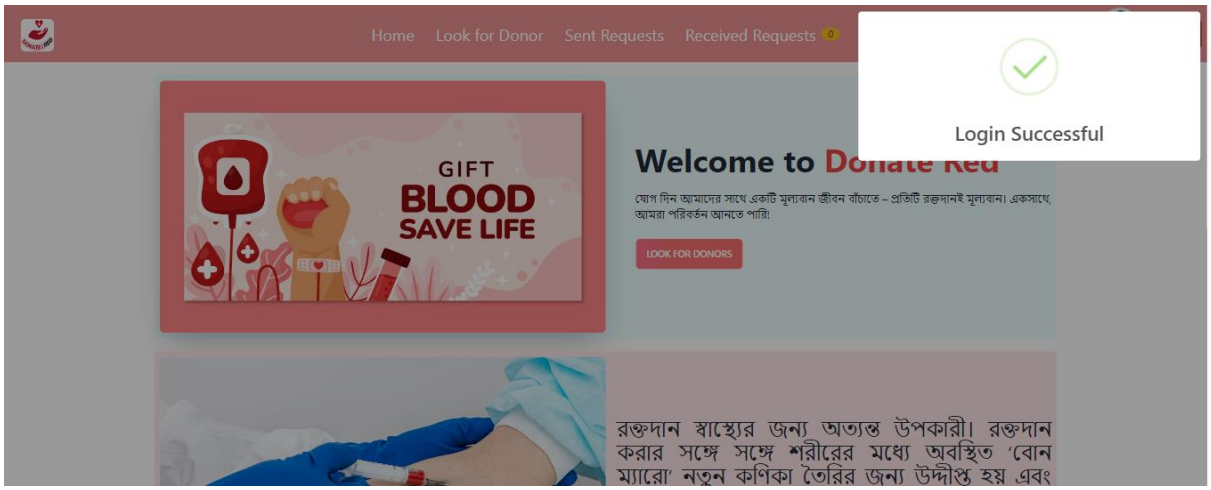

Fig.46 Successful login of the web application

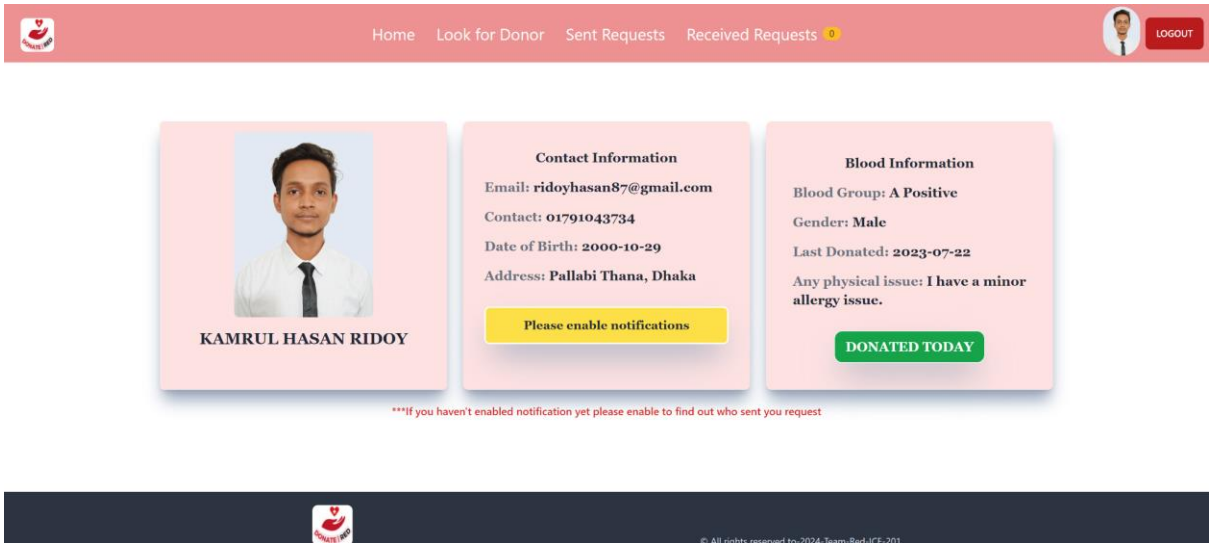

Fig.47 User profile page of the web application

Fig.48 represents the donor collection page, where all the registered users are listed. In Fig.49 a difference is shown that available and not available donors are marked with different colors.

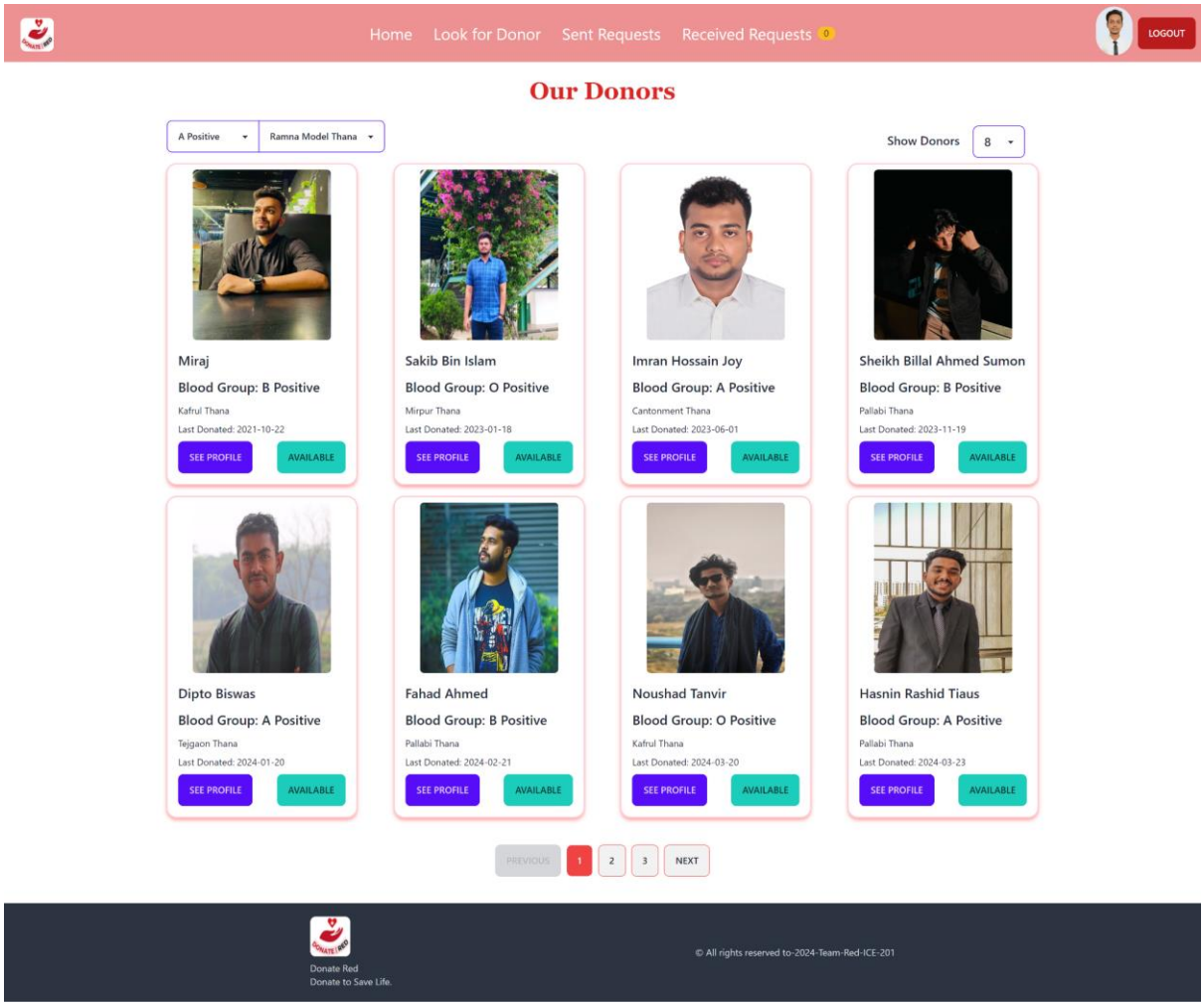

Fig.48 Registered Donors of the web application

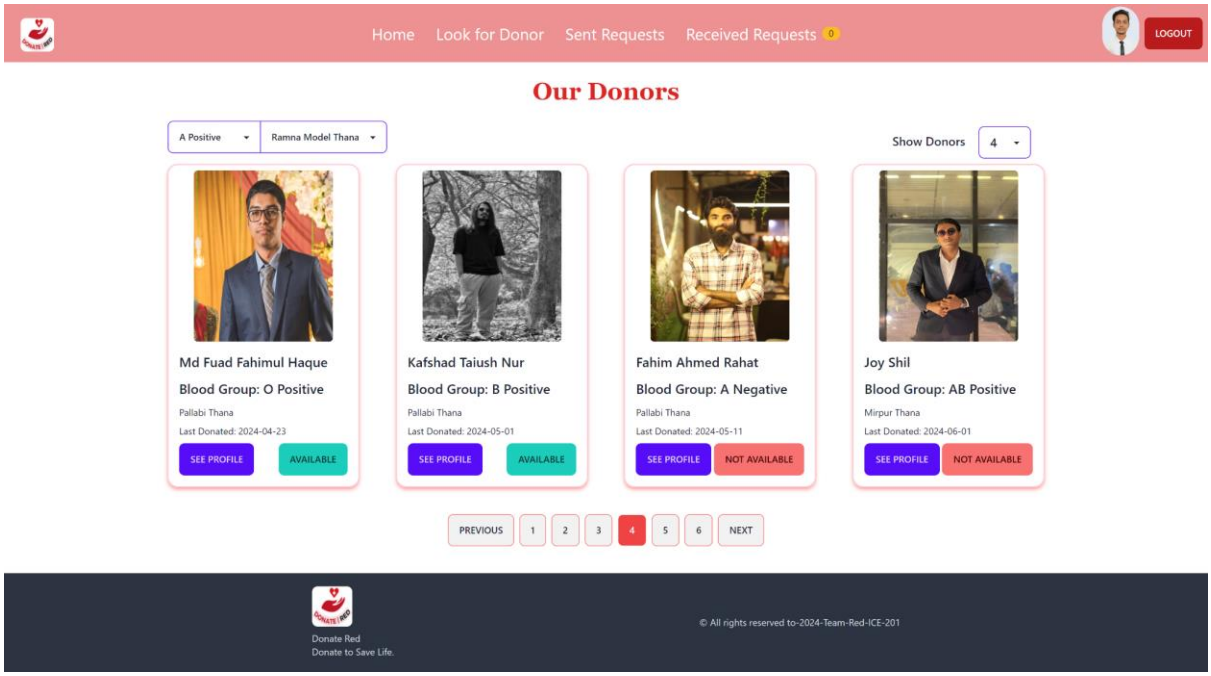

Fig.49 Donors who are available or not of the web application

We have implemented a filter method to find out the donors who have the same blood type matching with the patient and who are near the area. In Fig.50 we filtered by selecting A positive blood type and Pallabi Thana which led to three donors.

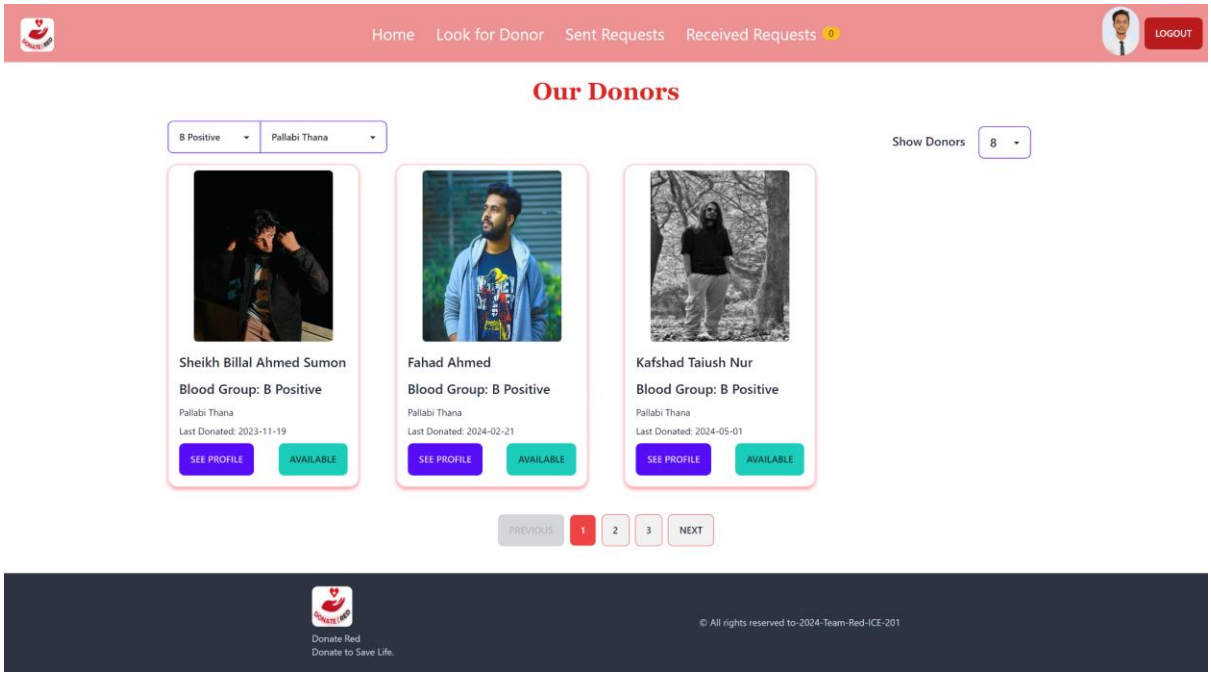

Fig.50 Filtered donor of the web application

Fig.51 shows a donor who has recently donated blood and is in a 90-day interval period. And a request can not be sent to him. Besides his profile information, our integrated Google Maps shows the donor's registered location.

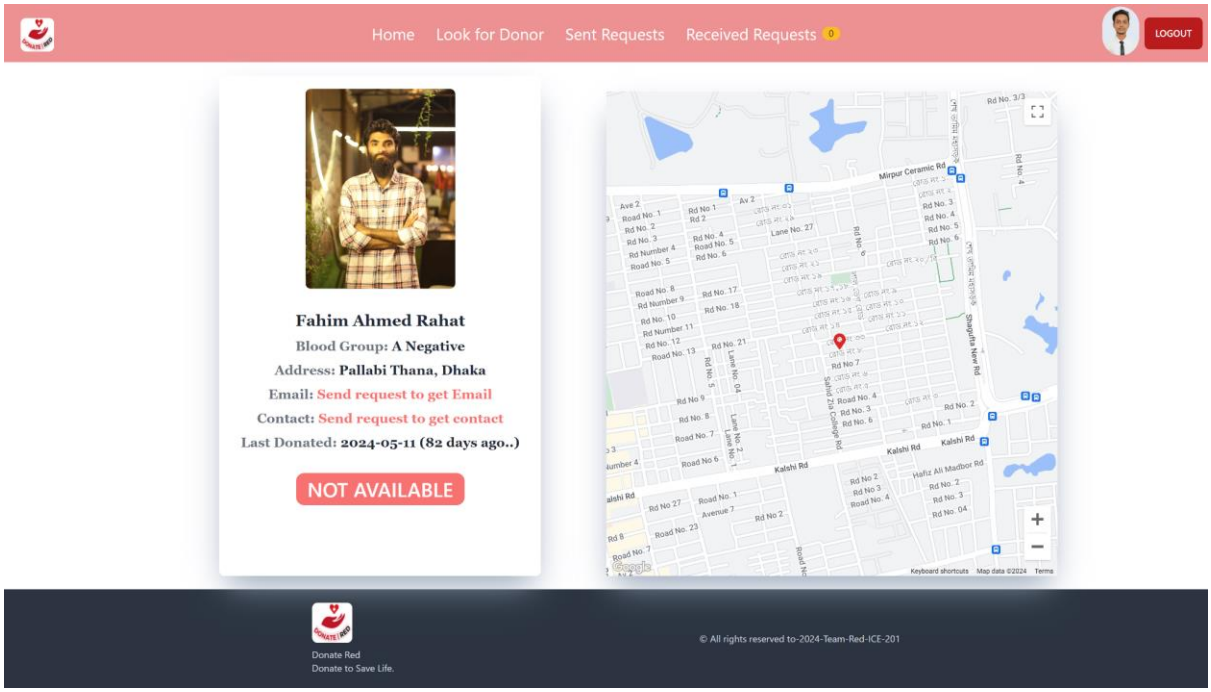

Fig.51 Donor profile who is in an interval of the web application

In Fig.52, a donor profile is shown who is available to donate blood. His contact information is hidden and asked to send a request to get the hidden information.

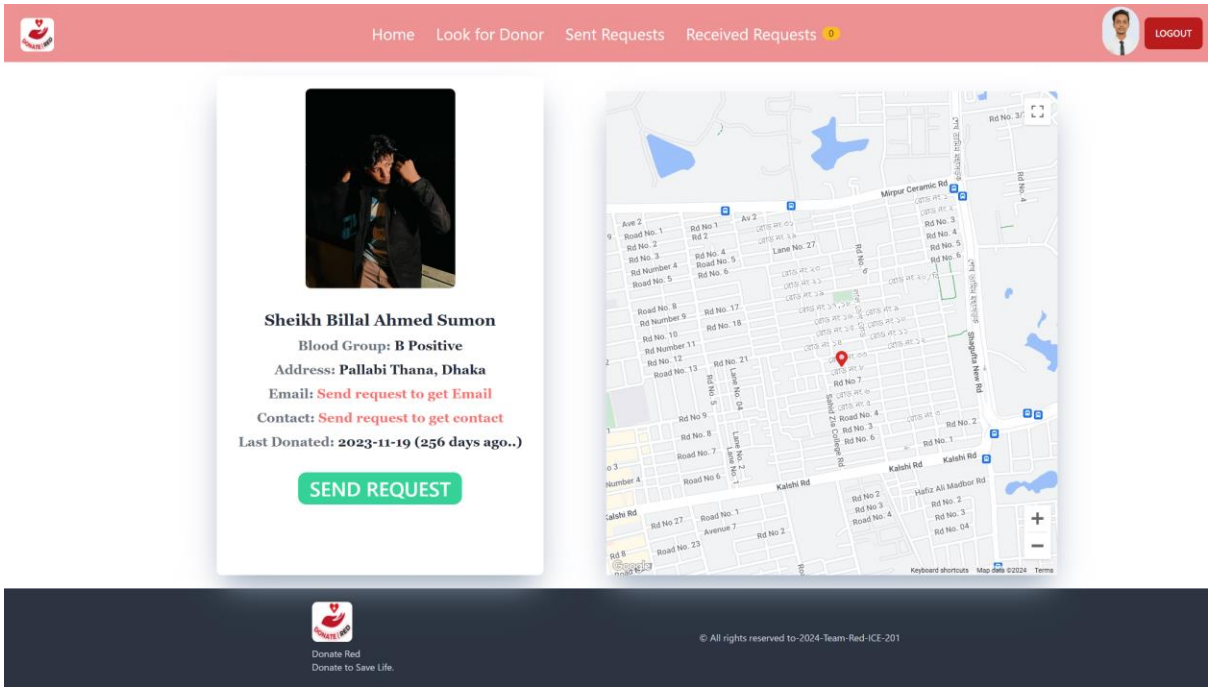

Fig.52 Donor profile who is available on the web application

We have sent a request to a donor who was available to donate blood. After sending the request we got a confirmation message alert that the request to donate was sent successfully. The send request button is changed to disabled in Fig.53.

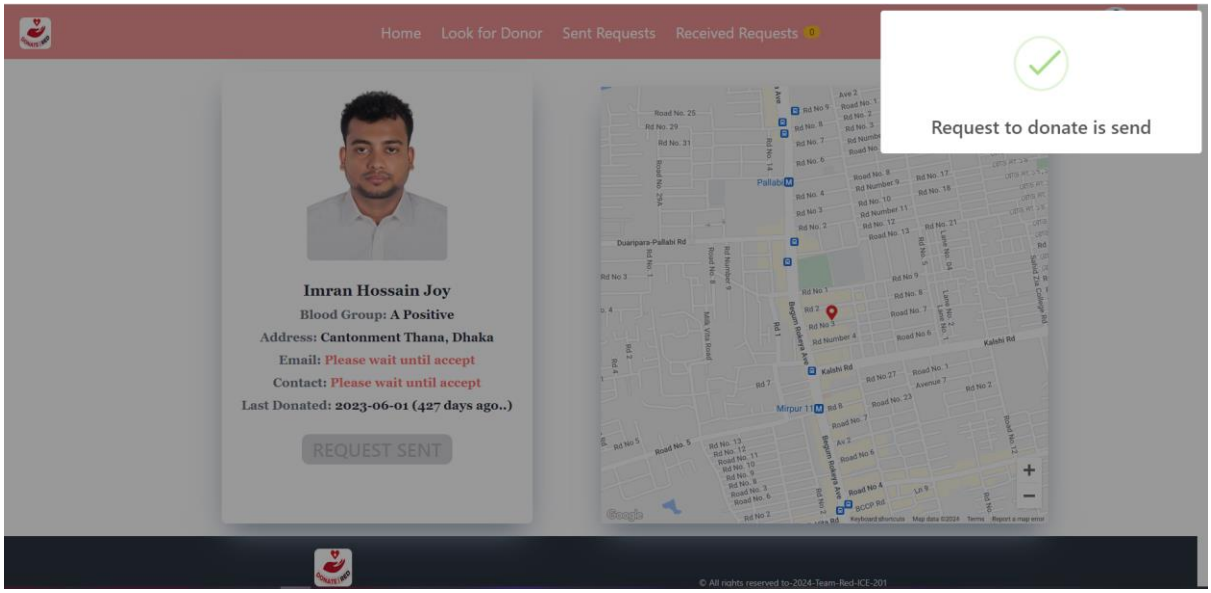

Fig.53 Request sent and information is hidden

In Fig.54 after sending a request to the donor, a notification is sent by Chrome browser. The received request page also got updated by the new request from the seeker. It has a seeker image, name, email, profile link, and two response buttons.

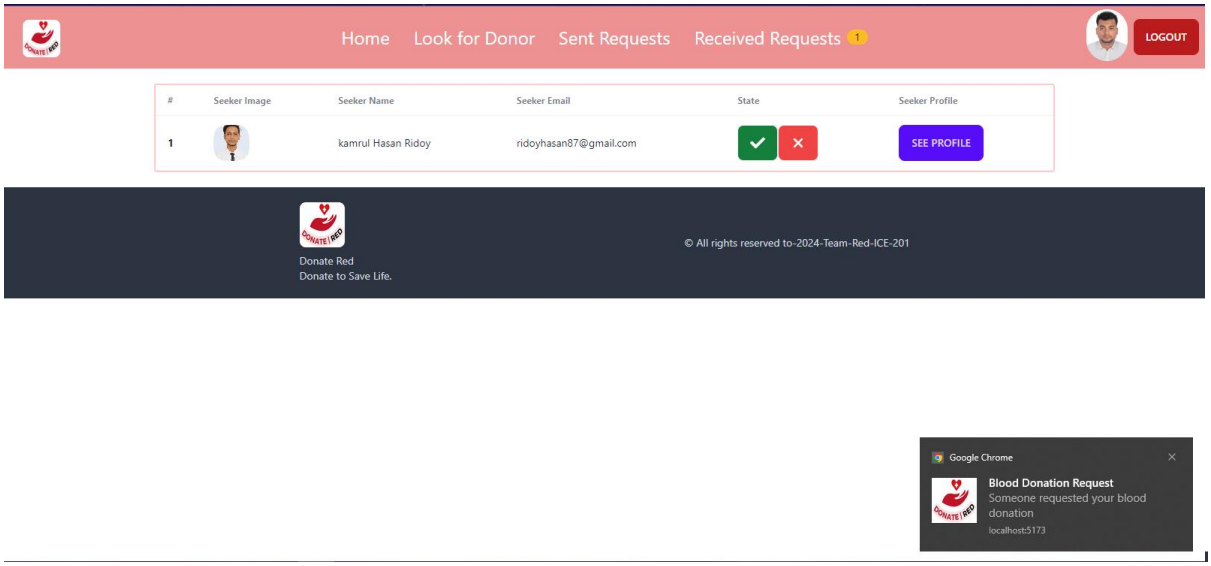

Fig.54 Notification and received request page

In Fig.55 after sending a request to the donor, the sent request page is updated with a list of requests sent to donors. However, the information of contact and email is hidden and requested to wait until the donor accepts the request.

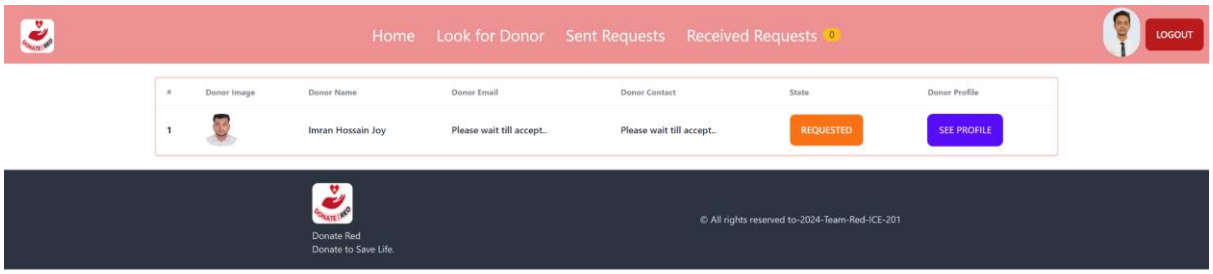

Fig.55 Sent request page of the application

When the donor accepts the request a message with “Request accepted” is displayed, then the state is changed to “ACCEPTED” which we can see in Fig.56.

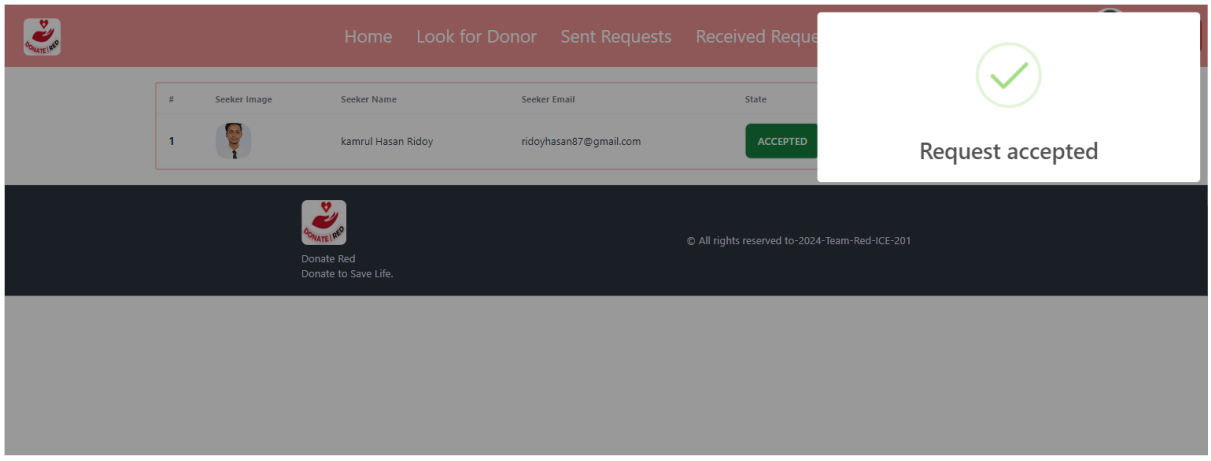

Fig.56 Acceptance of a request

In Fig.57 all the hidden information is available on the sent request page and also state has been changed. We can also visit the donor's profile by clicking the see profile button, which is linked to the requested donor’s profile page.

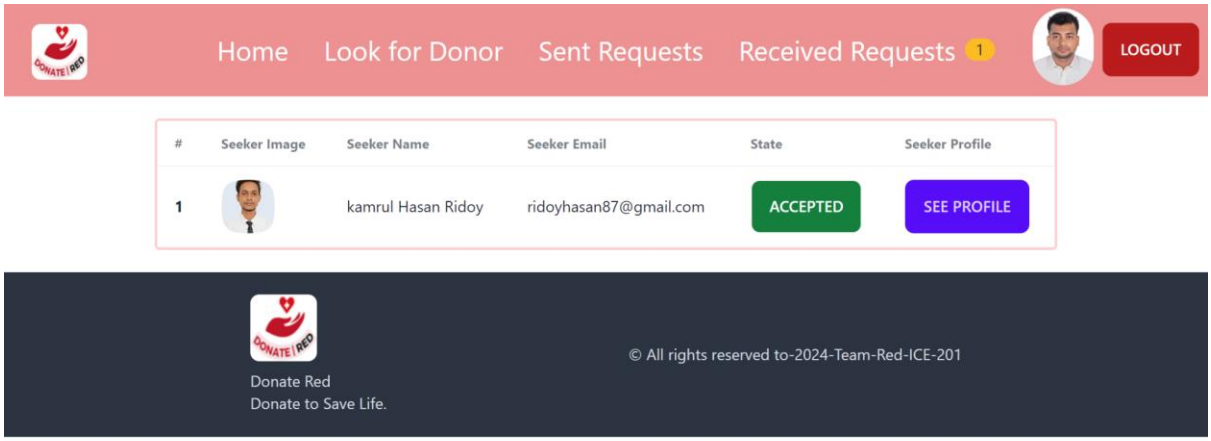

Fig.57 Hidden information is available

After clicking on the see profile link, the user was redirected to a donor profile page. This is shown in Fig.58, That information is also available in the donor’s profile page and a request can not be made again.

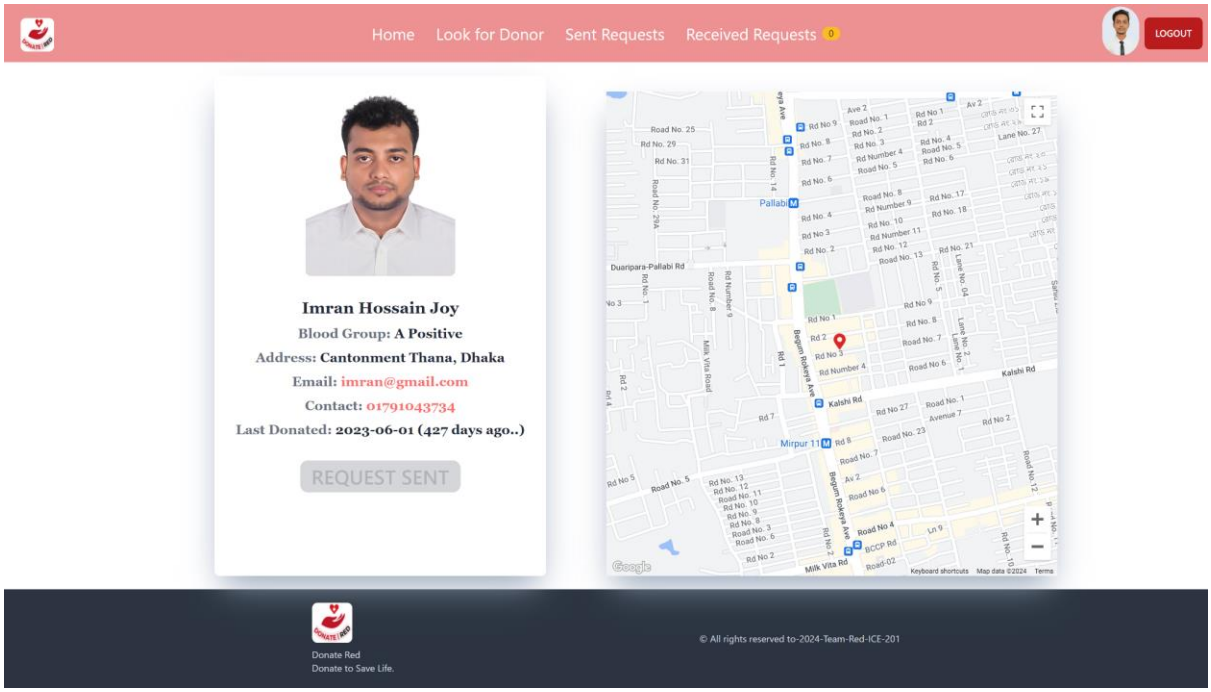

Fig.58 Request sending option disabled

The user data is registered in the mongoDB, and a part of the database is displayed showing a registered user’s information in Fig.59.

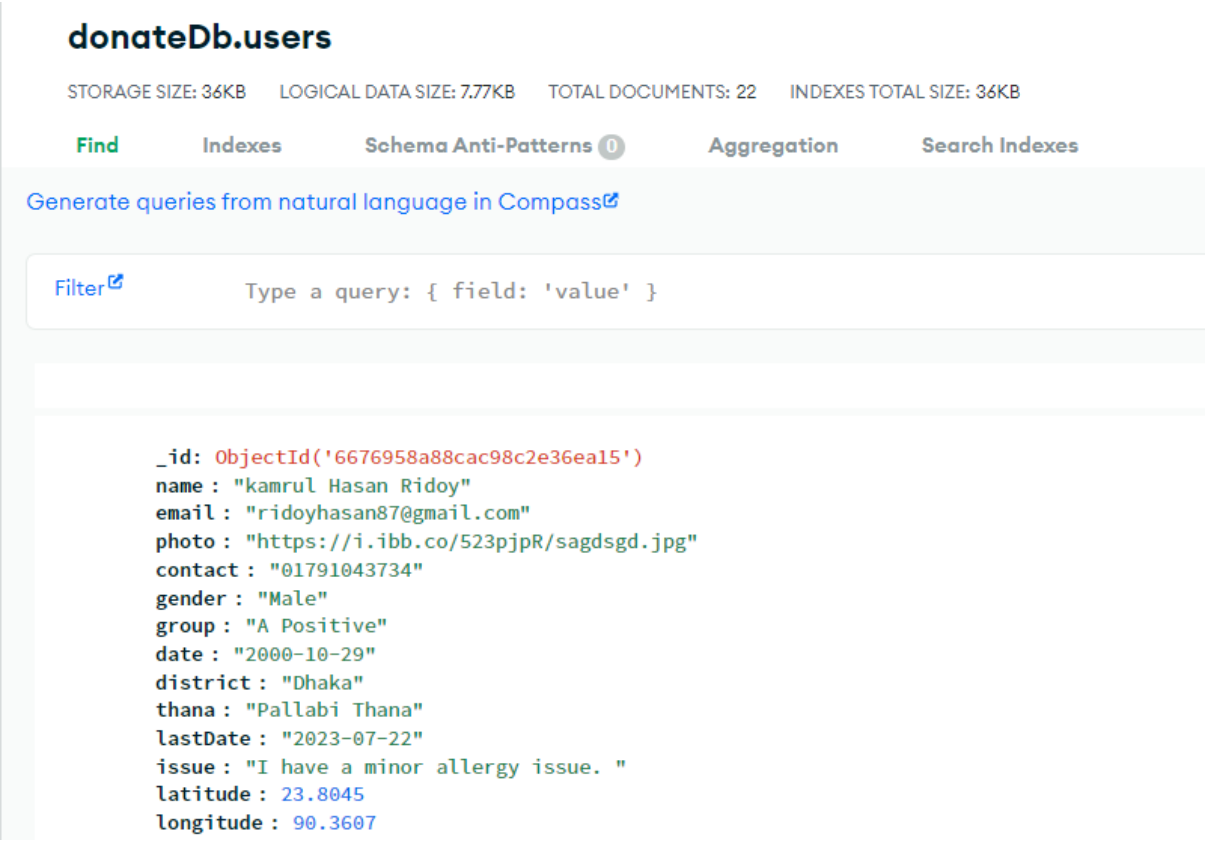

Fig.59 User collection from MongoDB

Similarly, when a request is made we create an object of information regarding the request, when sending the request to the donor the below object in Fig.60 was created and stored in the MongoDB.

Fig.60 Request collection from MongoDB

Fig.61 shows the subscription details of a user's device endpoint stored in the subscriptions collection.

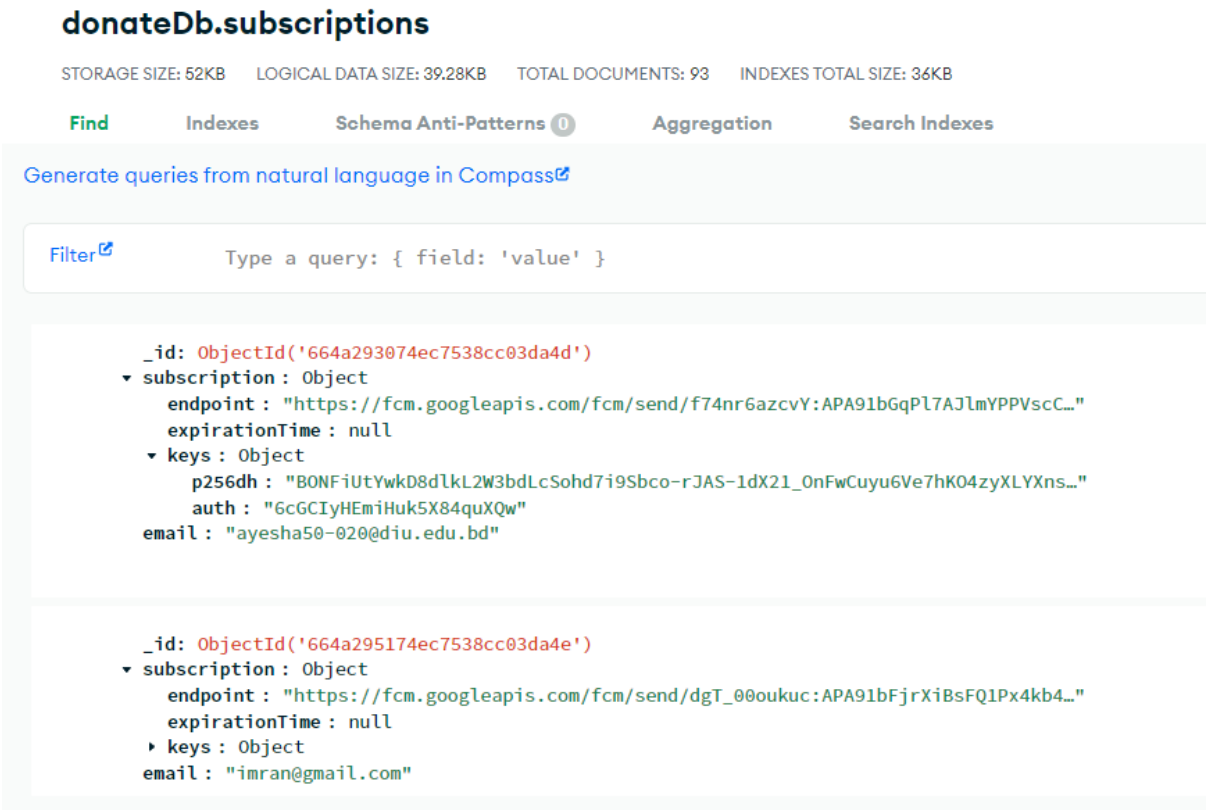

Fig.61 Subscription collection from MongoDB

5.4 Conclusion

Blood Donation Web Application testing and development were both really successful and inclusive. We have been able to detect and fix defects, validate system functionality, and ensure data accuracy through unit, integration, and system testing. The Blood Donation Web Application testing phase has been well-covered from all of its key functionalities. Iterative testing cycles and user feedback also contributed to taming the application into a well-rounded and reliable final prototype.

A prototype that has been developed is validated with detailed screenshots showing the interface of the application, the user's interaction, and messages of success or error messages. Due to rigorous testing and validation, this application is reliable and user-friendly. The final prototype portrays robust functionality, providing a solid foundation for a real-world blood donation management system. This project places great importance on careful testing and validation as a deliverable of high-quality and functional software.

Chapter 6: Project Management

6.1 Introduction

Engineering project management is a field that supervises and directs engineering projects from start to finish by fusing project management abilities with engineering knowledge. It involves planning, coordinating, and controlling various aspects of a project to ensure that it meets specified requirements within constraints such as time, cost, and quality. Project management entails organizing the several facets of web development in order to construct a web application. This includes defining project requirements, setting timelines for design and development, managing technical resources, ensuring quality assurance, and coordinating with stakeholders to deliver a successful website or web application.

6.2 Project Planning and Contribution of Team Members

6.2.1 Project Management

It is a diversified highly challenging field of planning, coordinating, and implementing complex engineering undertakings. At the very outset, engineers should plan their work on solid ground and introduce it to the working teams to achieve the "/Dk goals. Omission of this stage can lead to different kinds of problems at a later stage. Success demands technical expertise, leadership, and time and resource management abilities. Effective management of an engineering project involves the identification of the objectives of the project, the development of a comprehensive project plan, and the management of the risks involved in the project. When engineering projects became complex, and there were increasing demands on the effective usage of resources available, the term 'project management' was specifically called for to ensure the success of such projects. From the very start, our team members were aware of what their roles were, and this was the driving force toward the successful completion of our project. All in all, collaboration among members of our team was great. Our online blood donation application was developed and implemented as a genuine team effort, for it was impossible to tell where one member's contribution began or another's ended since each brought along different skills and expertise in pursuit of our common objective.

6.2.2 Planning and Proposal Preparing Phase in Project Management

Identifying a complex engineering problem was a significant challenge, as finding a web development-related issue proved difficult. However, recognizing the widespread difficulties related to timely blood collection inspired us to pursue this project. Our goal is to develop a blood donation website that facilitates connections between donors and recipients, ultimately enhancing the efficiency and effectiveness of the blood donation process. We engaged in collaborative discussions, created multiple design drafts, and provided a comprehensive project plan, complete with risk management strategies, expected outcomes, and relevant codes. Through our efforts, we proved that our chosen problem is indeed a complex engineering challenge.

6.2.3 Phase 1 Gantt Chart

Before starting Phase 1, we created a Gantt chart to systematically organize task assignments and detect potential issues that could delay project advancement.

Tasks	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
Planning a Significant Engineering project	Kamrul,	Aayesha											
Methodology and Project plan		Kamrul,	Ayesha										
Discuss it with the supervisor			Kamrul, Ayesha										
Multiple design approach				Kamrul	,	Ayesha							
Collecting survey Report					Ayesha	Kamrul							
Draft concept note						Kamrul	, Ayesha						
Progress slide							kamrul						
Make proposal slide									Kamrul,	Ayesha			
Final proposal report submission												Kamrul	, Ayesha

Fig.62 Phase 1 Gantt Chart

6.2.4 Design and Development Phase in Project Management

Our task in this step was to evaluate draft designs using various criteria and ultimately choose the best one. To create the initial frontend and backend designs, we distributed the work among the team members and used a browser and our IDE to simulate our application. Based on parameters like sending requests and accepting requests, updating donation dates, and other elements like the Firebase authentication system for user authentication, we were able to choose the best design.

6.2.5 Phase 2 Gantt Chart

Tasks	Week 1	Week 2	Week 3	week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	week 14	Week 15
Frontend Design 1	Kamrul,	Ayesha													
Backend Design 1		Kamrul		Ayesha											
Frontend Design 2				Ayesha											
Backend Design 2					Kamrul										
Identify Problems and Constraints							Kamrul	Ayesha							
Design Modification based on problems								Kamrul		Ayesha					
Writing Report											Kamrul,		Ayesha		
Collection of User Data											Ayesha				
Database Integration													Kamrul		
Optimal design selection and modification														Kamrul,	Ayesha

Fig.63 Phase 2 Gantt Chart

6.2.6 Completion and Execution Phase in Project Management

This was the last phase of our final year project, where we had to implement the most important criteria like notification system, Google map integration, and database development. We divided work among ourselves, we had to show proper teamwork, based on problems, and testing our application. After relentless work, we finally completed the project.

6.2.7 Phase 3 Gantt Chart

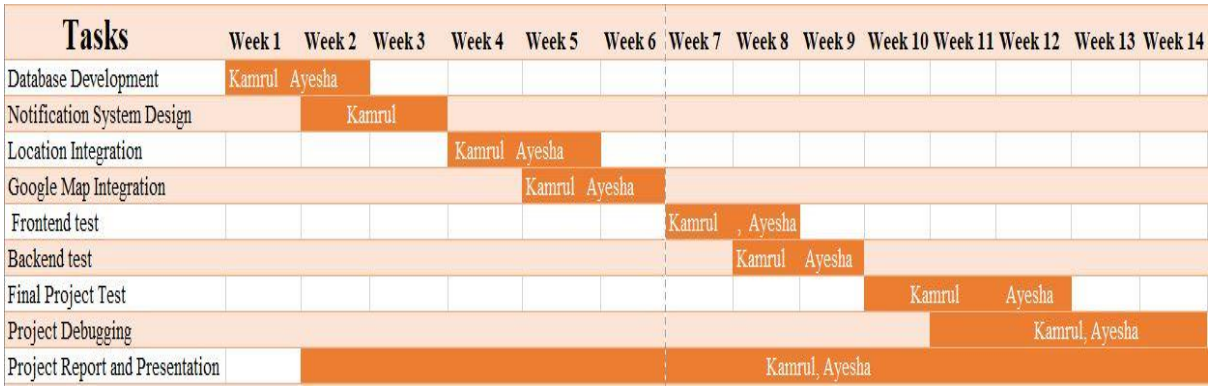

Fig.64 Phase 3 Gantt Chart

6.3 Challenges and Decision Making

During our group meetings, we meticulously addressed most of the issues that arose during our work. We facilitated open and in-depth discussions, allowing each team member to contribute their unique perspectives and ideas. Through this collaborative approach, we were able to develop innovative and well-considered solutions, thus significantly enhancing our problem-solving skills and the overall effectiveness of our project management. Throughout the project, we organized weekly meetings with our team members and supervisor. Their invaluable guidance and feedback played a crucial role in the successful completion of the project. To ensure clarity and accountability, we diligently documented all meeting discussions, which helped us work more formally and keep track of our activities. Before initiating the project, we developed a Gantt chart for each phase, ensuring timely completion of all tasks.

6.4 Conclusion

In the real world of work, engineers will encounter challenges like tight budgets and uneven task allocation. To get beyond these challenges and accomplish their objectives, engineers will need to demonstrate their situational management abilities. A key component of our capstone project's success was efficient team management. Through the utilization of individual talents and abilities, we overcame obstacles, showed creativity, and produced a sturdy and intuitive application. Our capstone project presented a number of difficulties. Nonetheless, we were able to resolve the issue through open communication, straightforward goal-setting, a friendly atmosphere that kept us motivated, and most importantly the supervision of our supervisor.

Chapter 7: Conclusion And Recommendation

7.1 Conclusion

Our objective was to develop a lean, efficient platform that makes blood donation easy, establish a 90-day period after which one can make the next donation, and facilitate donors to communicate directly with the seekers. The Blood Donation Web Application achieves this by providing a user interface for donor registration and creating donation requests, and it ensures real-time updates in the form of notifications. It ensures that donors donate in intervals as recommended and allows seekers to communicate directly with matching donors, thereby improving the whole process of donation. Overall, the Blood Donation Web Application project has been a success story in that we created a friendly platform for the maintenance of blood donation. It implements all the required functions: donor registration, donation request, geolocation, and notification nicely. It was passed through rigorous testing to make sure that each part functioned with great accuracy, while the system was able to handle varied scenarios. The prototype showed with the help of screenshots confirmed the stability and usability of the application, ensuring that it serves as a very reliable foundation for further improvements.

7.2 Future Recommendation

Integration of SMS and Email Notifications: Implement SMS and email notifications to provide donors and recipients with timely updates about donation schedules, requests, and other critical information.

Enhanced Geolocation Features: Upgrade the geolocation tracking capabilities to provide more accurate location data, possibly integrating with advanced mapping services. This enhancement will improve the matching of donors and recipients based on proximity.

Hospital Integration: Establish a connection with hospitals and medical centers to streamline the donation process. Hospitals can update donation statuses, verify donor eligibility, and manage donation days directly through the system.

Automated Donation Day Reminders: Develop a system to automatically notify donors about their eligibility to donate again based on their previous donation dates. This feature could be integrated with hospital systems to provide accurate, up-to-date information.

In summary, the Blood Donation Web Application represents a significant step forward in streamlining the blood donation process. By harnessing the latest web technologies and providing an intuitive user interface, we have created a system that not only meets current needs but also lays the foundation for future enhancements. The proposed recommendations, including integrating SMS and email notifications, enhancing geolocation features, and incorporating hospital systems, will further revolutionize the blood donation process, making it more efficient, reliable, and user-friendly. Healthcare organizations, governments, and technology providers need to collaborate and implement these solutions.

APPENDIX

Related Code/Theory

Frontend Code: https://github.com/Kamrul101/donate-red-client

Backend Code: https://github.com/Kamrul101/donate-red-server

Live Link: https://blood-donation-9245e.web.app/

Log Book

Team Log Book Summary

Capstone Project			
Members	**Member's Name & ID**	**Email**	**Contact**
Member 1	Kamrul Hasan Ridoy 201-50-002	kamrul50-002@diu.edu.bd	01791043734
Member 2	Mst Ayesha Khatun 201-50-020	ayesha50-020@diu.edu.bd	01756227022
ATC Details			
ATC 1			
Chairman	Md. Taslim Arefin	arefin@daffodilvarsity.edu.bd	
Internal Member	Prof. Dr. A.K.M. Fazlul Haque	akmfhaque@daffodilvarsity.edu.bd	
Internal Member	Ms. Tasnuva Ali	tasnuva@daffodilvarsity.edu.bd	
External Member	Dr. Engr. M. Quamruzzaman		

Date/Time/Place	Attendee	Meeting Summary	Comment
Date:10.08.2023 Time:12:30 PM Place: AB4,7th floor, sometimes Online Meeting	1. Kamrul 2. Ayesha 3. Dipto Biswas	Discuss the Project Plan	This meeting slot was selected for the whole semester for offline meetings.
17.08.2023	1. Kamrul 2. Ayesha 3. Dipto Biswas	Discuss project methodology and design	
22.08.2023	1. Kamrul 2. Ayesha 3. Dipto Biswas	Discuss the diagram and proposal report	
28.08.2023	1. Kamrul 2. Ayesha 3. Dipto Biswas	Proposal report	Proposal report approved
15.09.2023	1. Kamrul 2. Ayesha 3. Dipto Biswas	Login and Registration Page	
27.09.2023	1. Kamrul 2. Ayesha 3.Dipto Biswas	Homepage and donor page model	Agreed on that model and instructed to update some topics
10.10.2023	1. Kamrul 2. Ayesha 3. Dipto Biswas	Testing error problems and updated	Completed donor page section
12.11.2023	1. Kamrul 2. Ayesha 3. Dipto Biswas	Send request and received request section model and database integration	
25.11.2023	1. Kamrul 2. Ayesha	Testing errors and data and updating the server	Completed this section
13.01.2024	1. Kamrul 2. Ayesha 3. Dipto Biswas	Discuss about the notification and geolocation section	
16.02.2024	1. Kamrul 2. Ayesha 3. Dipto Biswas	Collecting data based on this location section, and Added chapters 3 and 4	Completed notification section

27.02.2024	1. Kamrul 2. Ayesha 3. Dipto Biswas	Finalizing code integration after combining all sections	Approved location section
24.03.2024	1. Kamrul 2. Ayesha 3. Dipto Biswas	Started writing the report, completed chapters 1,2	
19.04.2024	1. Kamrul 2. Ayesha 3. Dipto Biswas	Added chapters 5, and added references.	
23.05.2024	1. Kamrul 2. Ayesha 3. Dipto Biswas	Finalized Project report	Feedback given

REFERENCES

[1] Wikipedia contributors. (2024, January 15). *Blood donation in Bangladesh* [Online]. Available: https://en.wikipedia.org/w/index.php?title=Blood_donation_in_Bangladesh&oldid=1195766279, Accessed Date: 13, February 2024

[2] Islam, Muhammad Badrul. "Blood transfusion services in Bangladesh," *Asian Journal of Transfusion Science*, vol. 03, issue. 02, pp. 108-110, July 2009.

[3] Jiisun, Md & Rupa, Rasheda & Chowdhury, Monzur Hussain & Mushrofa, Hasina, et al., "Blood Donation Systems in Bangladesh: Problems and Remedy," *International Journal of Business and Management*, vol. 14, pp. 145, July 2019.

[4] Manika, Gollapelly & Venkata Sridhar Reddy & SaiVamshi, Kammampati , & Ms.Prashanthi,et al., "Development and Implementation of a Web-Based Blood Bank Management System for Efficient Blood Donation and Distribution." *International Journal for Research in Applied Science & Engineering Technology*, vol. 11, issue. 04, pp. 1830-1834, April 2023.

[5] M. Vinod & Keerthi Gaddam & Likhitha Inavolu, et al., "Online Blood Donation Management System," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 9, issue. 03, pp. 300–303, May 2023.

[6] B M, Shashikala & Pushpalatha, M. & Vijaya, B, et al., "Web Based Blood Donation Management System (BDMS) and Notifications", *Cognitive Computing and Information Processing*, pp. 118-129, April 2018.

[7] M. S. Abbas and S. Rizvi, "Online Blood Bank Management System", *Journal of Management and Service Science*, vol. 02, issue. 01, s. no. 007, pp. 1-6, March 2022.

[8] Phan, Hong Duc. "React framework: concept and implementation.", 2020.

[9] Bacastow, Todd. *Elements of a use case diagram | GEOG 468: GIS Analysis and Design. (n.d.)* [Online]. Available: https://www.e-education.psu.edu/geog468/l8_p4.html

[10] Pressman, Roger S. *Software Engineering: A Practitioner's Approach*, 7th ed. Boston, Mass., Mcgraw Hill, 2010.

[11] Ghezzi, Carlo, et al. *Fundamentals of Software Engineering*, 2nd ed. New Delhi, Prentice-Hall Of India, 2002.

[12] Larman, Craig. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*, 2nd ed. Upper Saddle River, NJ, Prentice Hall Ptr, 2002.

[13] GeeksforGeeks. (2024, January 16). *Sequence diagrams Unified Modeling Language (UML)* [Online]. Available: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/, Accessed Date: 13, March 2024.

[14] GeeksforGeeks. (2024, January 13). *Use case diagrams Unified Modeling Language (UML)* [Online]. Available: https://www.geeksforgeeks.org/use-case-diagram/, Accessed Date: 17, March 2024.

[15] GeeksforGeeks. (2024, January 15). *Activity Diagrams Unified Modeling Language (UML)* [Online]. Available: https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/, Accessed Date: 22, March 2024.

[16] GeeksforGeeks. (2024, February 20). *What is a Swimlane Diagram* [Online]. Available: https://www.geeksforgeeks.org/swim-lanes-in-activity-diagram/, Accessed Date: 29, March 2024.

[17] Leach, Ronald J. *Introduction to Software Engineering*. 2nd ed., Boca Raton Chapman Et Hall, CRC Press, 2020.

[18] Quatrani, Terry, and U. M. L. Evangelist. "Introduction to the Unified Modeling Language." *A technical discussion of UML*, vol. 6, issue. 11, pp. 03, June 2003.

[19] Baresi, Luciano, and Mauro Pezze. "An introduction to software testing." *Electronic Notes in Theoretical Computer Science*, vol. 148, issue. 01, pp.89-111, Feb 2006.

[20] Fielding, Roy T., and Richard N. Taylor. "Principled design of the modern web architecture." *ACM Transactions on Internet Technology*, vol. 02, issue. 02, pp. 115-150, May 2002.

[21] Casciaro, Mario, and Luciano Mammino. *Node. js Design Patterns*. Packt Publishing Ltd, July 2016.