Blood Line

REPORT OF MINOR PROJECT SUBMITTED FOR PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF COMPUTER APPLICATION

By

JEET PAL
REGISTRATION NO – 223771010014
UNIVERSITY ROLL NO – 37701222032

SUJAL PURBEY
REGISTRATION NO – 223771010036
UNIVERSITY ROLL NO – 37701222004

BIJOY GHOSH
REGISTRATION NO – 223771010011
UNIVERSITY ROLL NO – 37701222008

ANKITA MODAK
REGISTRATION NO – 223771010005
UNIVERSITY ROLL NO – 37701222005

UNDER THE SUPERVISION OF

Mrs. Sudipta Dey
Department of CS (Non AICTE)

AT

RCC INSTITUTE OF INFORMATION TECHNOLOGY (NON-AICTE)
Affiliated to Maulana Abul Kalam Azad University of Technology
CANAL SOUTH ROAD, BELIAGHATA, KOLKATA – 700 015

2024 - 2025

RCC INSTITUTE OF INFORMATION TECHNOLOGY (Non-AICTE)
KOLKATA – 7OOO15, INDIA

CERTIFICATE

The report of the Project titled BloodLine submitted by JEET PAL, SUJAL PURBEY, BIJOY GHOSH, ANKITA MODAK (Roll No.: BCA2022007, BCA202228, BCA202216, BCA202212 of BCA 5th Semester of 2024) has been prepared under our supervision for the partial fulfillment of the requirements for BCA degree in the MAKAUT. The report is hereby forwarded.

Mrs. SUDIPTA DEY
Dept. of CS(Non AICTE)
RCCIIT, K o l k a t a
(Internal Supervisor)

…….
ARINDAM MONDAL
Department of Computer Science (Non-AICTE)
RCC Institute of Information Technology, Kolkata – 700 015, India

ACKNOWLEDGEMENT

We are also indebted to Dr. Anirban Mukherjee, Principal (*In-Charge)* RCCIIT and Dr. Arindam Mandal (HOD, Dept. of CS) for their unconditional help and inspiration.

We express my sincere gratitude to Mrs. Sudipta Dey (Mentor) of Department of Computer Science (Non-AICTE), RCCIIT and for extending valuable times for me to take up this problem as a Project.

Last but not the least we would like to express my gratitude to Mr. Swarup Kumar Paul, Mr. Falguni Adhikary, Mrs. Priyanka Banerjee of our department who helped me in their own way whenever needed.

Date: **16/01/2025**

JEET PAL
223771010014
37701222032
BCA – 5th Semester,2024, RCCIIT

SUJAL PURBEY
223771010036
37701222004
BCA – 5th Semester, 2024, RCCIIT

BIJOY GHOSH
223771010011
37701222008
BCA – 5th Semester, 2024, RCCIIT

ANKITA MODAK
223771010005
37701222005
BCA – 5th Semester, 2024, RCCIIT

RCC INSTITUTE OF INFORMATION TECHNOLOGY (Non-AICTE)
KOLKATA – 7OOO15, INDIA

CERTIFICATE of ACCEPTANCE

The report of the Project titled BLOODLINE submitted by JEET PAL, SUJAL PURBEY, BIJOY GHOSH, ANKITA MODAK (Roll No.: BCA2022007, BCA202228, BCA202216, BCA202212 of BCA 5th Semester of 2024) is hereby recommended to be accepted for the partial fulfillment of the requirements for BCA/ B.Sc. CS degree in the MAKAUT.

Name of the Examiner	**Signature with Date**
1. …….	…….
2. ……..	……..
3. ……	……
4. …….	……

TABLE OF CONTENTS

Introduction

BloodLine is an innovative web application created to bridge the gap between blood donors and those in urgent need of blood. In many emergency situations, the delay in locating suitable donors or accessing the required blood type can mean the difference between life and death. Blood Line is designed to address this critical challenge by providing an efficient, user-friendly platform that ensures a smooth and reliable connection between blood donors and requesters.

This platform leverages modern technology to streamline the process of blood donation and request management. By incorporating location-based features, it enables requesters to identify and connect with donors in their vicinity, ensuring timely assistance. The application also prioritizes security by incorporating user authentication and data privacy measures, giving all participants peace of mind

❖ The main goals of Blood Line include:

1. Efficient Donor-Requester Connection: Facilitating quick and seamless communication between donors and requesters based on location and blood type.
2. Timely Availability: Addressing delays by ensuring real-time notifications and updates to donors about nearby requests.
3. Streamlined Processes: Simplifying the registration, request, and donation workflows to make blood donation more accessible for all.
4. Enhanced Transparency: Keeping users informed about the status of their requests and availability of resources in nearby blood banks.

Blood Line also tackles common challenges faced in the blood donation ecosystem, such as verifying the eligibility of donors, ensuring the availability of specific blood types, and maintaining an organized record of blood requests and donations. By bringing all these features into a single platform, Blood Line empowers communities to respond swiftly to emergencies, promote voluntary blood donations, and create a culture of helping others in need.

Problem Analysis

Blood donation systems in many regions face significant challenges that hinder their effectiveness. These inefficiencies can lead to delays in providing critical support to those in need. Below are some key issues commonly observed in existing blood donation systems:

Key Problems

1. **Lack of Centralized Platforms**

   Many systems operate in isolation, with no unified platform to connect donors, requesters, and blood banks. This fragmentation makes it difficult for individuals in need of blood to locate suitable donors or check the availability of resources in nearby blood banks.

2. **Delayed Communication**

   In emergencies, time is of the essence. Traditional methods of finding donors—such as contacting local hospitals, spreading word-of-mouth messages, or relying on outdated databases—are often time-consuming, leading to critical delays in fulfilling blood requests.

3. **Limited Coordination with Local Blood Banks**

   Existing systems often fail to integrate with blood banks, leading to a lack of visibility into their current stock levels or requirements. This lack of coordination can result in duplicated efforts or wasted resources.

❖ How Blood Line Solves These Issues

To address these challenges, **Blood Line** offers innovative solutions that focus on streamlining processes, improving communication, and leveraging technology to bridge gaps in the system. Key features include:

1. Digital Platform for Donor and Requester Registrations

- Blood Line provides a centralized web-based platform where donors and requesters can easily register and manage their profiles.
- Donors can specify their availability, location, and blood type, while requesters can post urgent blood requirements.
- The platform ensures that all data is organized and accessible, reducing the need for manual coordination.

2. Location-Based Matching System

- Blood Line uses geolocation technology to match requesters with donors in their vicinity.
- This feature minimizes delays by enabling quick identification of nearby donors who can fulfill the request promptly.
- Real-time mapping functionality enhances the system's responsiveness, providing requesters with clear guidance on donor locations and designated blood banks.

3. Verified Interactions Through Email Notifications

- The platform ensures secure and verified communication by sending email notifications to both donors and requesters.
- Donors are notified of new requests with options to accept or decline, while requesters receive updates about donor responses and request statuses.
- Email verification during registration prevents misuse of the platform and maintains trust between users.

By addressing these key pain points, Blood Line not only simplifies the blood donation process but also ensures that lifesaving resources are accessible to those who need them most. Through its innovative and user-friendly approach, Blood Line transforms the traditional blood donation model into a highly efficient and effective system.

Review of Literature

An analysis of existing blood donation systems, such as the Red Cross Blood Donor App and BloodBanker, highlights certain limitations that restrict their effectiveness in addressing critical blood donation challenges. These systems, while offering notable features, fail to fully bridge the gap between donors, requesters, and local blood banks. This section reviews their functionality and identifies areas for improvement.

Existing Systems and Their Limitations

1. Red Cross Blood Donor App

- Strengths:

  The app allows users to schedule donations, track donation history, and access health tips related to blood donation. It also offers a reward system to encourage repeat donations.

- **Limitations**:

  - Lack of Real-Time Notifications: Notifications are not tailored for immediate updates regarding urgent blood requests or availability.
  - No City-Specific Coordination: The app does not focus on hyperlocal requirements, making it less effective for regional emergencies.
  - Minimal Integration with Blood Banks: There is limited visibility into blood bank stock levels or real-time updates about availability.

2. BloodBanker

- Strengths:
  BloodBanker provides a directory of blood banks and donor listings, making it easier to find potential donors.
- **Limitations**:
  - Static Listings: The platform primarily relies on static data, which is often outdated and does not reflect real-time availability.
  - No Dynamic Availability Tracking: Users cannot access live updates on donor availability or blood bank inventory.
  - Limited User Experience: The interface and interaction design are outdated, making it less user-friendly.

❖ How Blood Line Enhances Existing Models

Blood Line addresses these limitations by incorporating advanced features and modern technologies that improve the overall efficiency, usability, and responsiveness of the system. Below are the key enhancements:

1. City-Specific Coordination with Blood Banks
   - Blood Line focuses on hyperlocal operations by integrating with blood banks at the city level.
   - This coordination ensures that blood banks are aware of ongoing requests and can prioritize resource allocation accordingly.
   - Donors and requesters are matched based on their proximity, reducing delays and optimizing logistics.

2. Dynamic Availability of Blood Stock Information

- Unlike existing systems that rely on static listings, Blood Line updates blood stock levels in real-time.
- Blood banks can input their inventory data directly into the system, allowing requesters to view available blood units instantly.
- This feature minimizes wastage and ensures better resource management.

3. Leveraging Modern Web Technologies for Seamless User Experience

- Blood Line utilizes a modern tech stack (e.g., React for the frontend, Node.js and MySQL for the backend) to provide a responsive and intuitive user interface.
- Features such as real-time notifications, email alerts, and geolocation-based matching enhance user engagement and operational efficiency.

The platform is designed to be mobile-friendly, ensuring accessibility across devices.

Formulation / Algorithm for Blood Line

BloodLine employs a well-structured system to ensure efficient donor-requester matching and seamless communication using an automated email notification workflow. Below is a detailed breakdown of the formulation and algorithms utilized within the platform.

1. **Donor-Requester Matching Algorithm**

   The donor-requester matching algorithm is central to the Blood Line platform, ensuring quick and precise pairing between donors and requesters. The process is as follows:

   - Step-by-Step Algorithm

     - **Requester Submits a Blood Request**
       - The requester provides details such as blood group, required units, location (state and city), and urgency through a dedicated form on the platform.
       - The data is validated and stored in the system database.

     - **System Filters Potential Donors**

       - Criteria 1: Location Matching
         Donors are filtered based on the state and city specified in the blood request to ensure proximity and feasibility of donation.
       - Criteria 2: Availability Status
         Donors with their last donation date more than 3 months ago compared to request time are selected for further notification.
       - Criteria 3: Blood Group Matching
         The system cross-checks the donor's registered blood group with the requested blood group to ensure compatibility.

- **Email Notifications to Selected Donors**

  - The system sends personalized emails to the filtered list of donors, providing key details such as the requester's blood group, urgency, and a call to action to respond (accept/decline).

- **Donor Responses are Updated**
  - Donors respond by clicking on the provided options in the email (e.g., "Yes, I can donate" or "No, I cannot donate").
  - The system records these responses, updating the donor's status and the request's progress in the database.

- **Connection at the Designated Blood Bank**

  - If a donor accepts the request, the system notifies both the donor and the requester with the details of the designated blood bank.
  - The requester and donor proceed to the specified blood bank to complete the donation process.

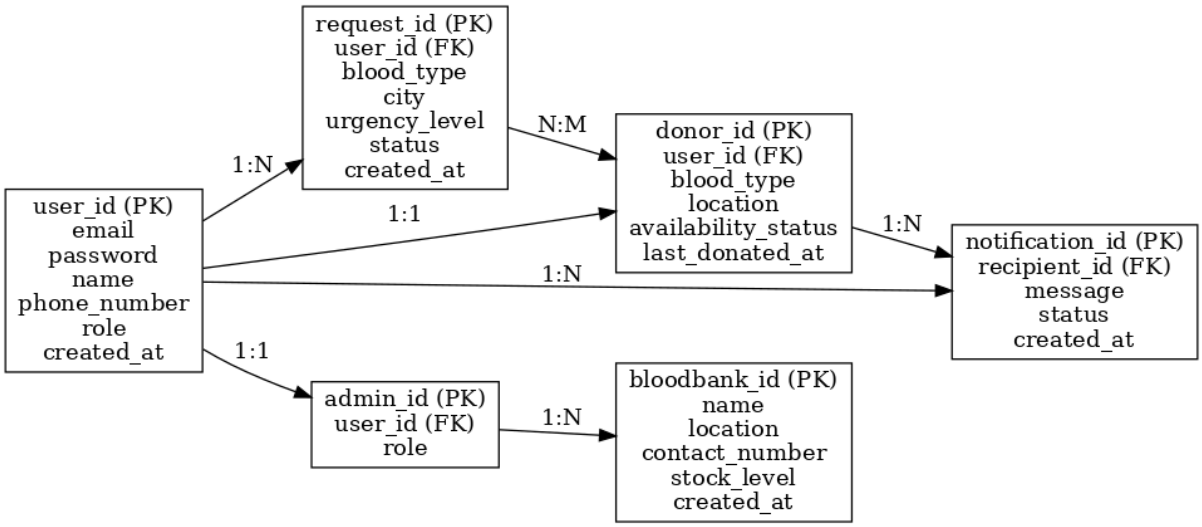
Key Advantages of the Algorithm and Workflow

- Efficiency: Automated matching reduces delays in locating eligible donors, ensuring timely support for requesters.
- Transparency: Both donors and requesters receive real-time updates about the process, fostering trust in the platform.
- Location-Based Accuracy: By focusing on state and city filters, the algorithm optimizes logistical feasibility and response rates.

Scalability: The algorithm can handle a large number of requests and donors simultaneously, making it suitable for regional and national implementation

E R Diagram of BLOODLINE:

Entity	Attributes
Users	UserID (PK), FullName, Email, Password, PhoneNumber, Role (Donor/Requester/Admin), City, State, Status
Donors	DonorID (PK, FK), BloodGroup, AvailabilityStatus, LastDonationDate
Requests	RequestID (PK), RequesterID (FK), BloodGroup, City, State, RequestStatus, CreatedAt
BloodBanks	BloodBankID (PK), Name, City, State, ContactNumber, Email, StockAvailability
Admins	AdminID (PK), FullName, Email, Password
Notifications	NotificationID (PK), RecipientID (FK), Message, NotificationDate, Status
DonorResponses	ResponseID (PK), DonorID (FK), RequestID (FK), ResponseStatus (Yes/No), ResponseDate



Problem Discussion

Challenges Faced

1. **Efficiently Filtering Donors by Location and Availability**

   The primary challenge was to identify and match donors based on their proximity to the requester and their availability status. This required a system capable of handling complex filtering criteria without compromising performance, especially as the database grows over time.

   - **Key Issues:**
     - Handling multiple filtering criteria (state, city, blood group, and availability).
     - Ensuring real-time performance when processing numerous donor entries.
     - Avoiding mismatches or missed matches due to poorly optimized queries.

2. **Ensuring Secure Email Delivery Using Nodemailer**

   Email communication is crucial for Blood Line's functionality, as it is used for confirmations, notifications, and donor responses. Challenges included ensuring reliable email delivery, preventing emails from being marked as spam, and safeguarding sensitive data.

   - **Key Issues**:
     - Avoiding delivery failures caused by invalid email addresses.
     - Mitigating risks of emails being flagged as spam.
     - Maintaining user data security during email exchanges.

3. **Managing Large Datasets in the Database Efficiently**

As the number of registered donors and requests grows, the system must manage and retrieve large volumes of data without delays. The challenge was to design a database that could handle scalability while maintaining fast query execution.

- Key Issues:
  - Optimizing database schema to reduce redundancy.
  - Ensuring fast query performance with increasing data size.
  - Implementing effective indexing strategies for frequent queries.

By addressing these challenges through optimized SQL queries, robust email handling with Nodemailer, and a user-friendly frontend, Blood Line successfully overcame the hurdles of donor-requester matching, secure communication, and scalable database management.

Implementation Details

1. **Frontend Technologies**:
    - **Vite React**: For fast development and modular code structure.
    - **Tailwind CSS**: For responsive and consistent UI design.
    - **Axios**: For seamless API integration.

2. **Backend Technologies**:
    - **Node.js**: For scalable and asynchronous server-side scripting.
    - **Express.js**: For managing routes and APIs.
    - **MySQL**: For robust database management.

3. **Email Interaction**:
    - **Nodemailer**: For sending automated and secure emails.

Implementation of Problem

The implementation of Blood Line addresses the challenges in connecting blood donors and requesters by incorporating core features that enhance functionality, security, and user interaction. The following elaborates on the implementation of these features:

1. Registration and Login

To ensure a seamless and secure onboarding process, Blood Line provides separate registration and login interfaces for donors and requesters.

Features:

- User Segmentation:
  - Donors register by providing their name, blood type, location (state and city), and availability status.
  - Requesters register with basic details like name, contact information, and location.
- Email Verification:
  - Implementation:
    - Upon registration, users receive a verification email through Nodemailer containing Welcome Message.
- Security:
  - Verification links are time-sensitive and one-time use.
  - Encrypted tokens in links ensure data integrity.
- Login Functionality:
  - Secure authentication with password hashing using bcrypt.js.
  - Sessions or JWT tokens for maintaining login status.

2. Blood Request Handling

The blood request system is the core functionality of Blood Line, designed for efficiency and accuracy.

Features:

- Requester Submission:
  - Requesters provide details such as required blood type, location, and urgency.
  - Input validation ensures all necessary details are submitted.
- Donor Matching:
  - The backend filters donors based on:
    - Location: Matching donors within the requester's state and city.
    - Availability Status: Ensuring only active and available donors are notified.
  - Matching logic is implemented through optimized SQL queries.
- Donor Notification:
  - Email Triggers: Matching donors receive automated emails with request details.
    - Emails include options to accept or decline the request directly.
  - Real-time updates are provided via WebSocket or Socket.IO (for future integration).

3. Donor-Requester Interaction

This feature ensures smooth communication and timely blood request fulfillment.

Features:

- Donor Response:
  - Donors can respond with "Yes" or "No" through email or a dedicated portal.
  - Accepted responses are logged in the database, and other notified donors are informed of request closure.
- Requester Notification:
  - Once a donor accepts, the requester receives an email confirmation with the donor's details.
  - Requesters are directed to visit the designated blood bank for the blood collection.
- Blood Bank Coordination:
  - Connections are facilitated at nearby registered blood banks to ensure secure and regulated transactions.

4. Admin Panel

The admin panel plays a crucial role in managing backend operations and ensuring the system runs smoothly.

Features:

- Blood Bank Management:
  - Admins can:
    - Add and update blood bank details.
    - Manage blood stock availability.
    - Monitor stock levels for different blood types and update requirements.

- Request Monitoring:

- Admins can view active requests and ensure timely fulfillment by coordinating with donors and blood banks.
- Automated alerts are sent to admins for unfulfilled requests.

- Reports and Analytics:

- Admins access data on donor participation, blood requests, and fulfillment rates.
- Insights help optimize donor engagement and blood stock management.

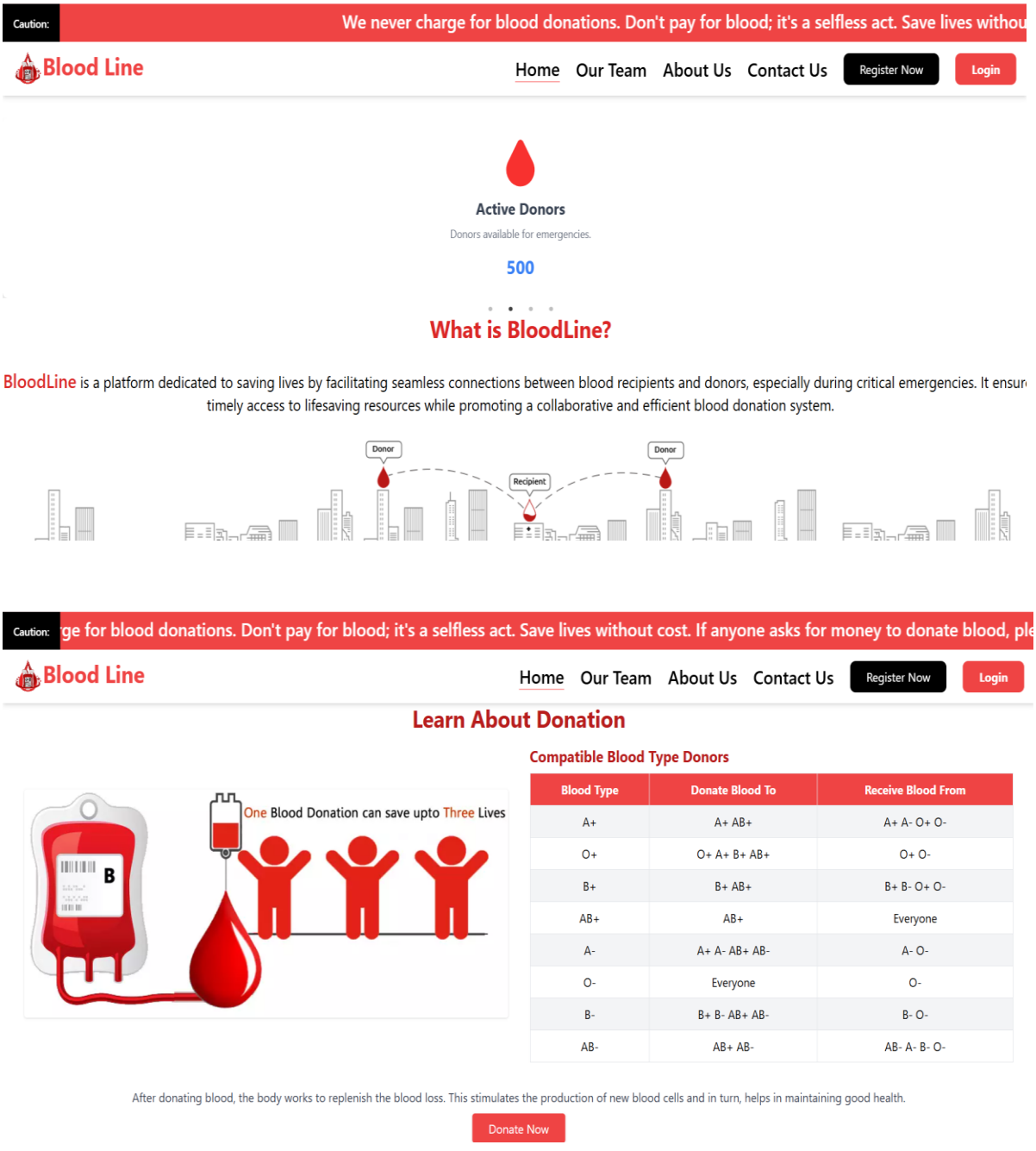
- User Management:

- Admins can manage user accounts, deactivate inactive donors, and resolve disputes or issues.

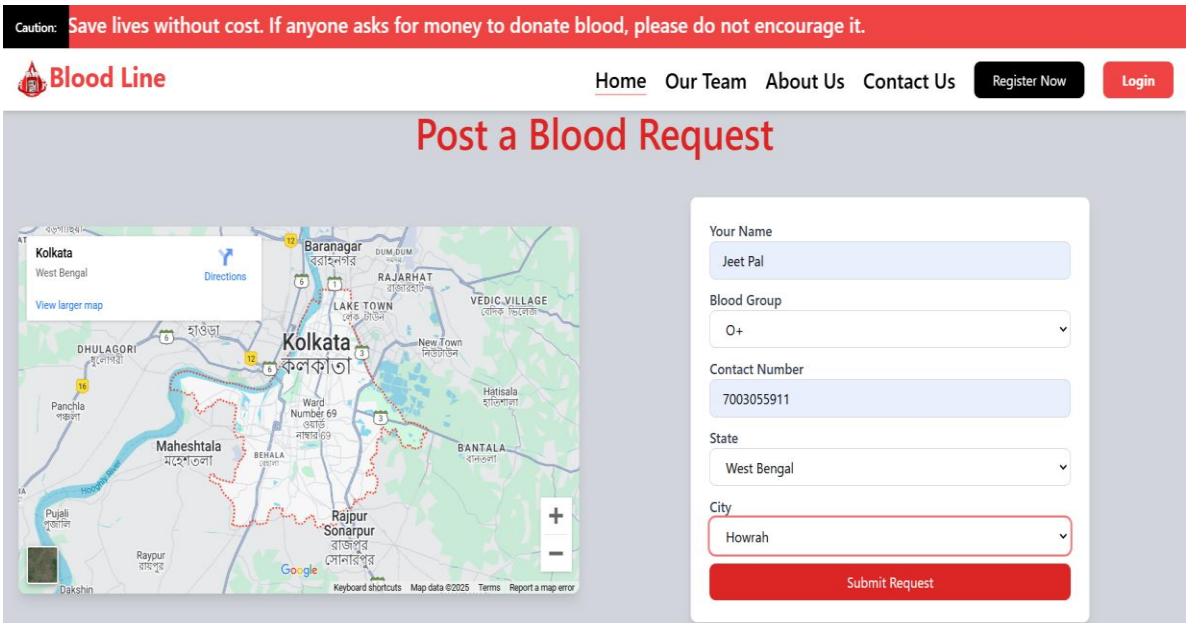
Sample Output:

Landing Page:

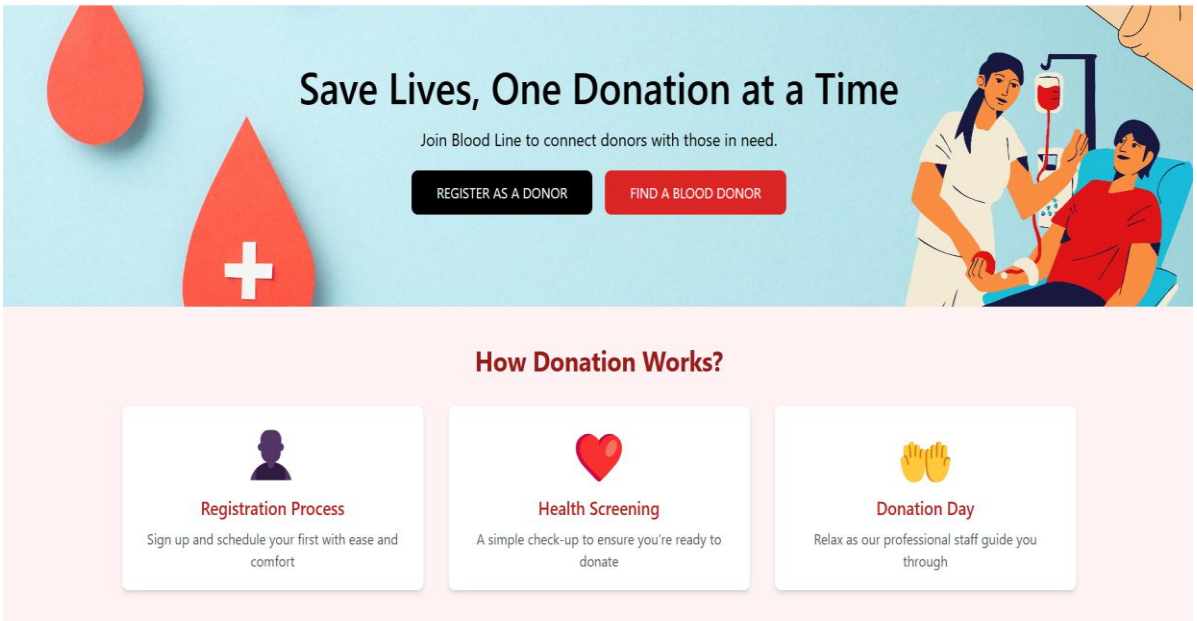
- Displays application information.

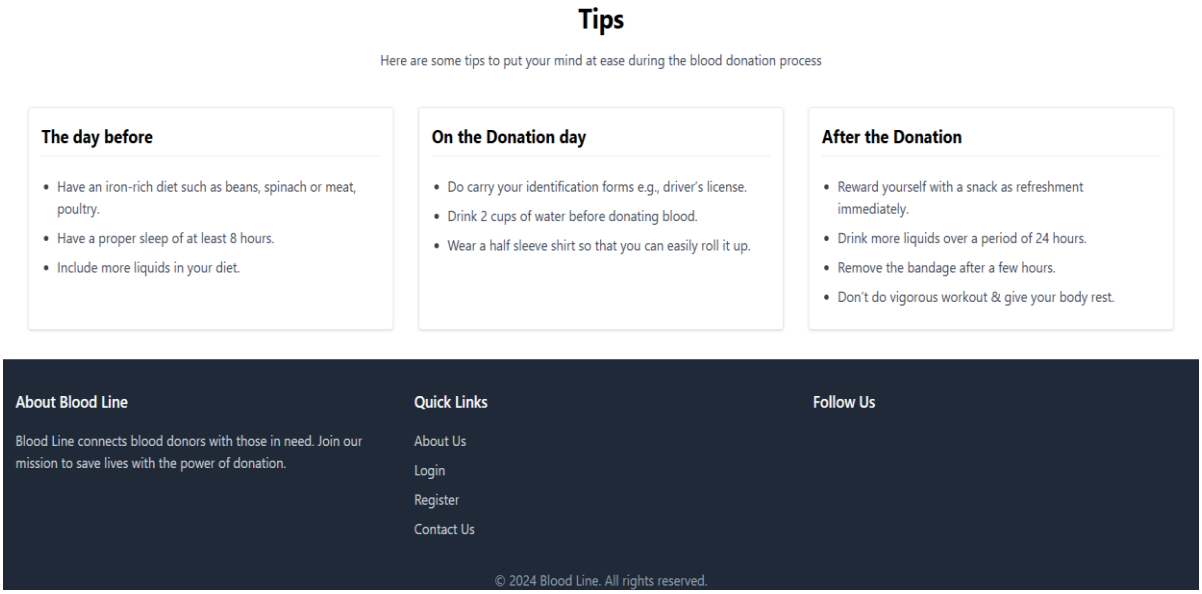


- Includes a blood request form and map guidance.



- Some tips for donors and description of donation process.

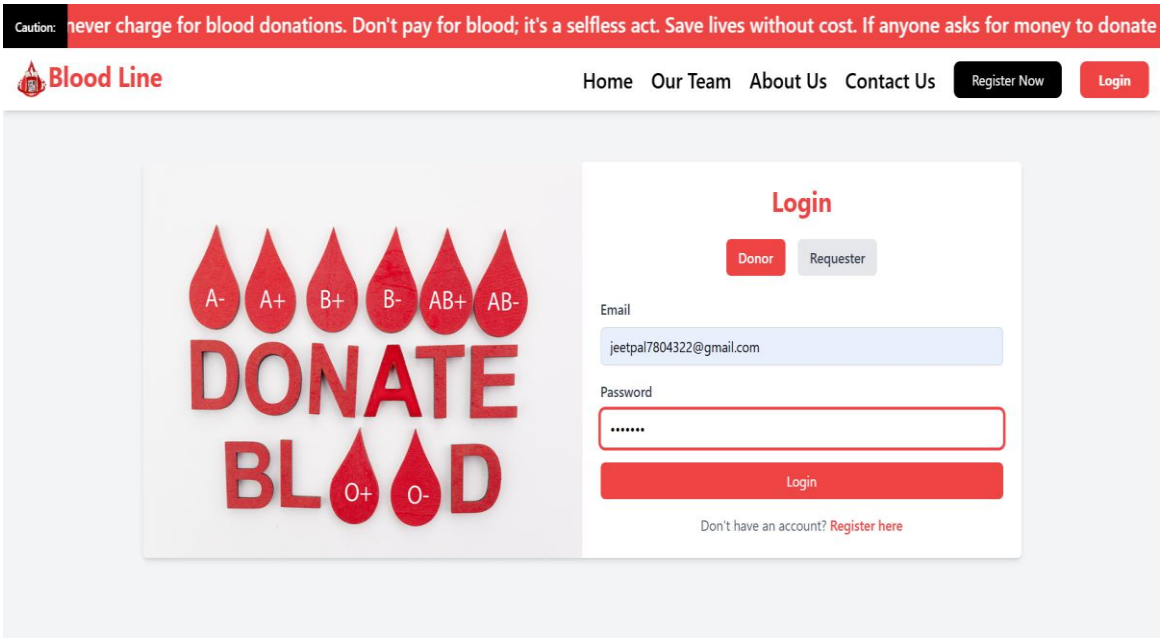




Registration Page:

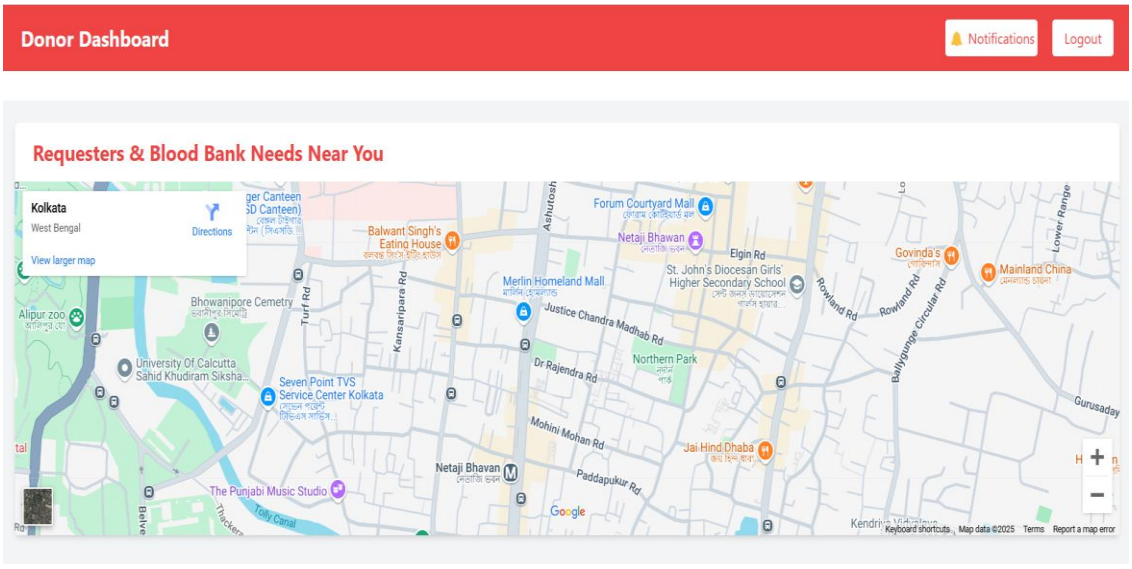
- Common registration page for donor and requester

Login Page:

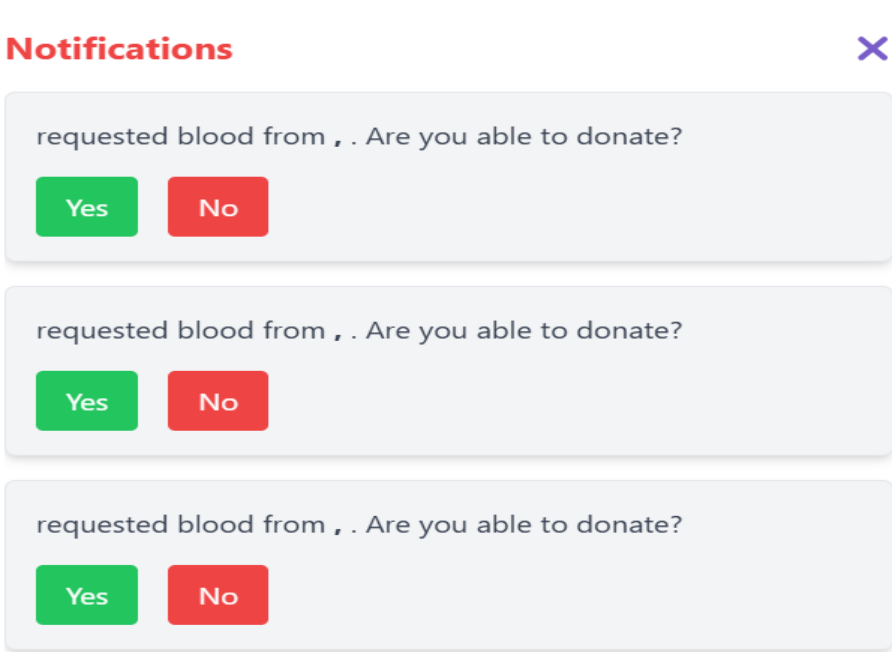


Donor Dashboard:

- All requesters and blood banks.

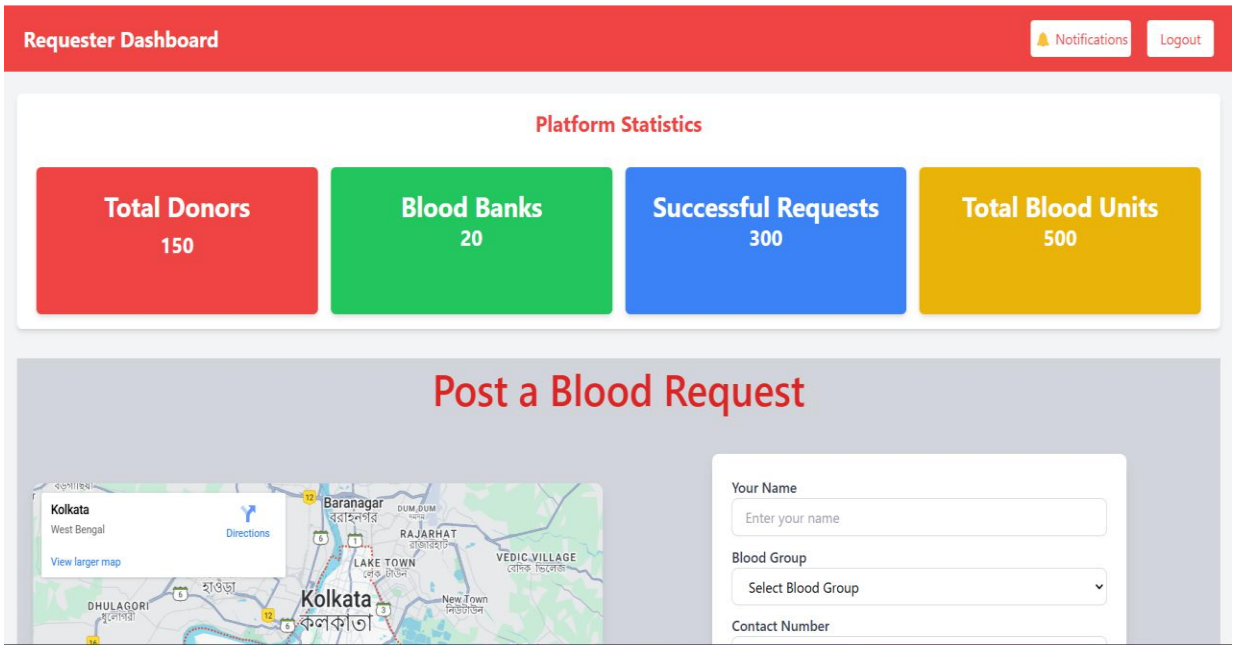


- Options to accept or decline requests.

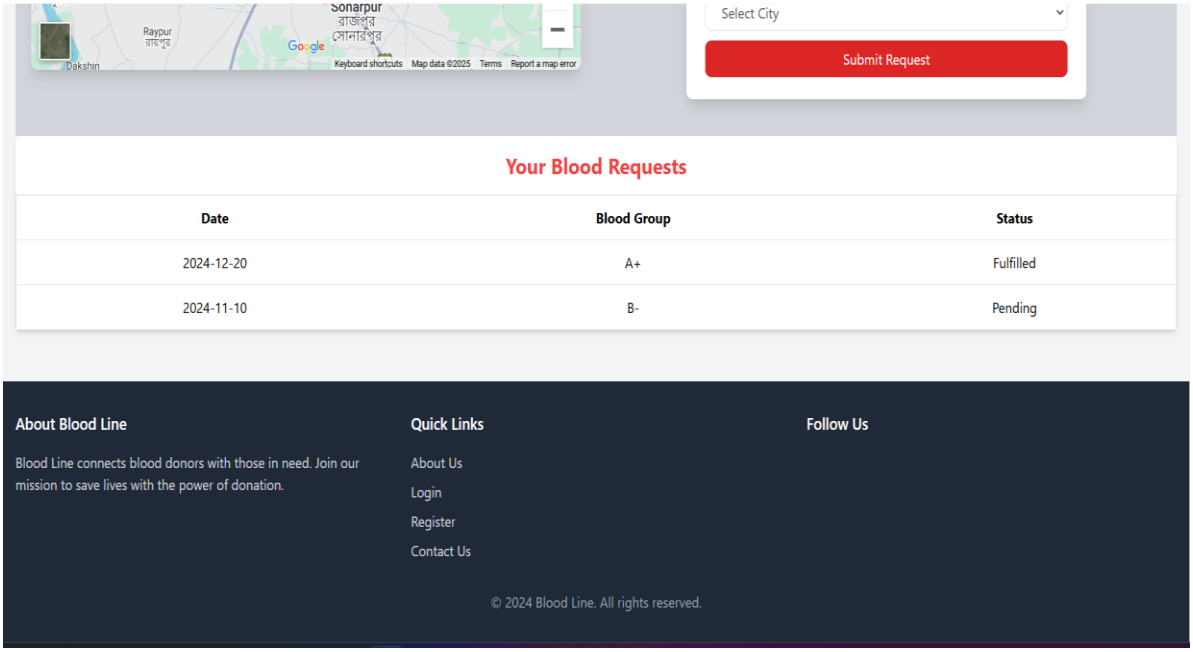


Requester Dashboard:

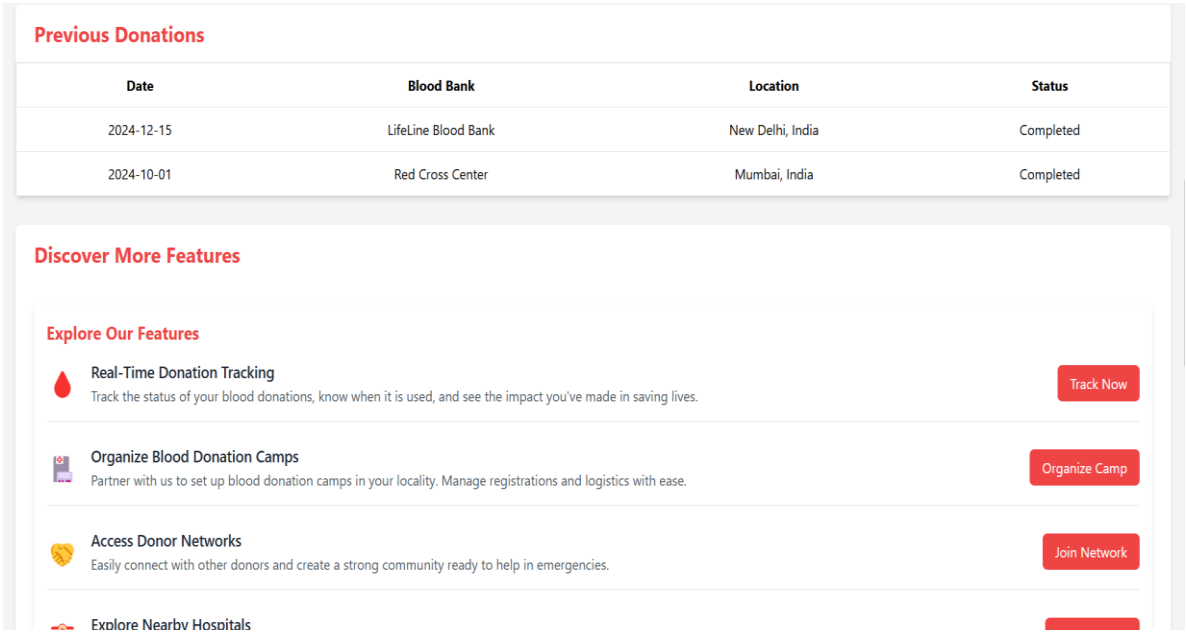
- All stats of website.



- Blood request history

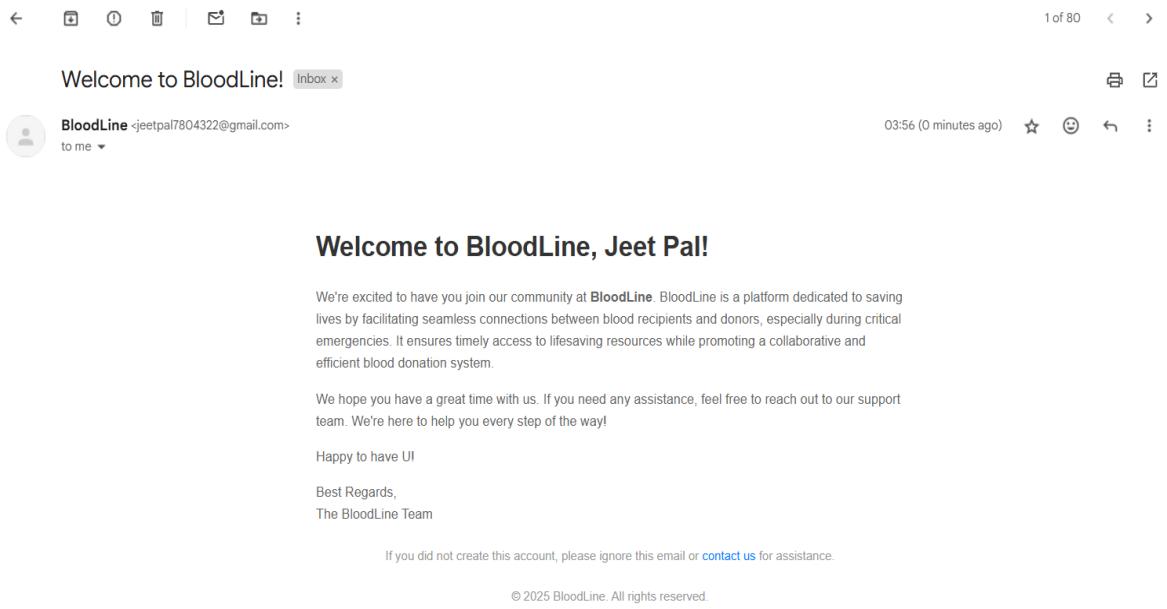


- Previous donations and some activities

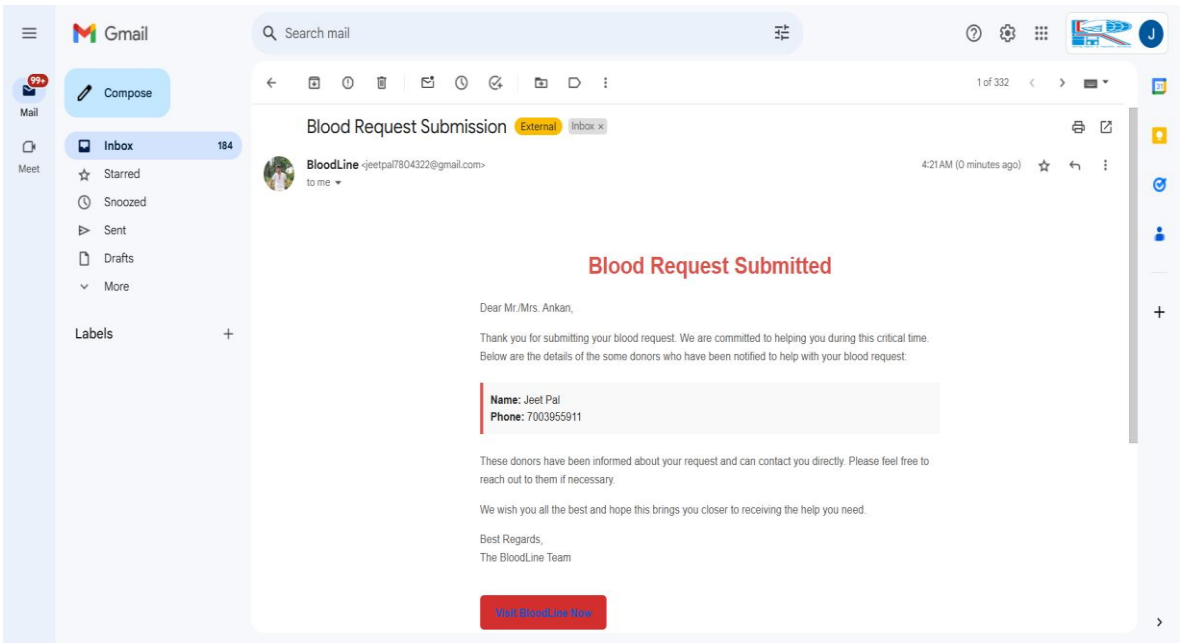


Email used:

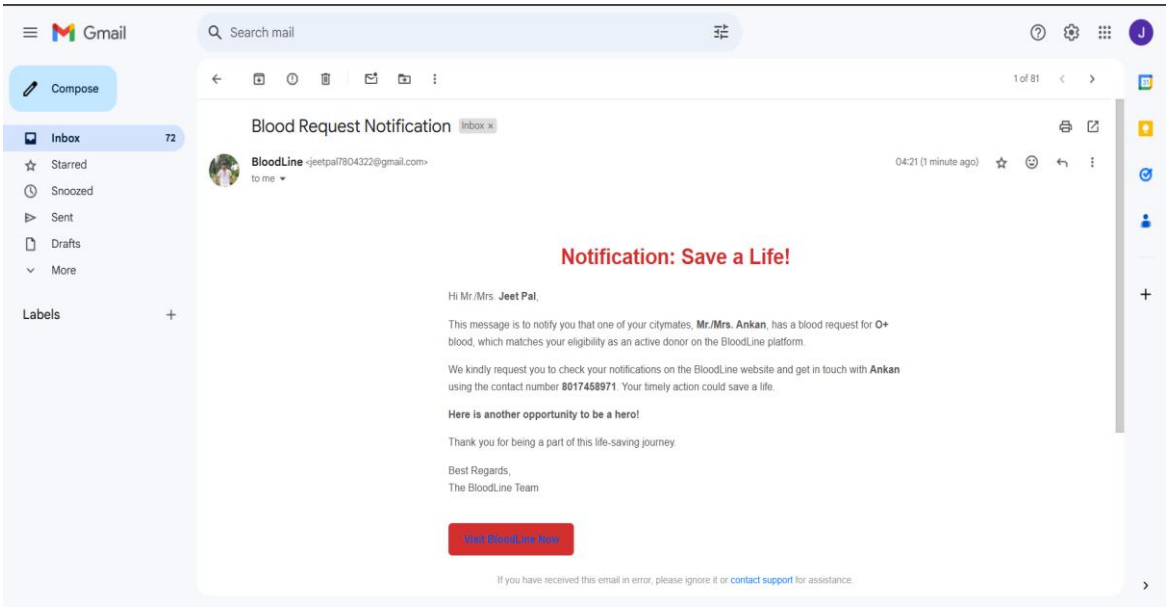
- Welcome Email



- Blood request submission email



- Blood request notification mail to donors



Conclusion

BloodLine successfully bridges the gap between blood donors and requesters by leveraging a streamlined, technology-driven approach. The platform efficiently connects users through its intuitive interface, real-time notifications, and robust database management. By facilitating swift and reliable communication between donors, requesters, and blood banks, BloodLine enhances the blood donation process, making it more efficient and impactful.

Future Scope of Work

To further improve the system and expand its reach, the following developments are planned:

- **Enhancing the Chatbot**: Incorporating advanced AI capabilities to provide 24/7 support for user queries and guidance.
- **Advanced Analytics**: Implementing predictive models to forecast blood demand based on historical data and trends.
- **Mobile Platform Integration**: Expanding the system to include iOS and Android apps for better accessibility and convenience.
- **Real-Time Communication**: Adding features for instant messaging or calls between donors and requesters to improve coordination.

Reference

- **Official Documentation of Vite React**: Comprehensive guidelines and tools for building fast and modular React applications using Vite. [https://vite.dev/]
- **Tailwind CSS Documentation**: A detailed resource for creating custom, responsive, and utility-first designs with Tailwind CSS. [https://tailwindcss.com/]
- **Node.js Official Docs**: Authoritative documentation for building scalable and efficient server-side applications with Node.js. [https://nodejs.org/en].
- **MySQL Documentation**: Comprehensive references for managing relational databases, SQL queries, and database optimization techniques. [https://dev.mysql.com/doc/]
- **Nodemailer**: Official guidance for implementing secure and reliable email functionality in Node.js applications. [https://nodemailer.com/]

- Applications:
  - **Red Cross Blood Donation**: Inspiration and information sourced from the official Red Cross platform for blood donation initiatives. [https://nodemailer.com/]
  - **U Blood**: Insights gained from this platform, focusing on real-time donor connections and blood donation awareness. [https://ublood.com/].

Appendix (Program Code)

- Important snippets include:
  - **Donor-requester matching logic**

```
const addBloodRequest = async
(name,email,requesterId,bloodGroup,state,city,contact,status) => {
   const conn = await db.getConnection();
   try {
      await conn.beginTransaction();
      const [result] = await conn.execute(
         `INSERT INTO bloodrequests (requester_id, blood_group, state, city,
contact,status)
         VALUES (?,?,?,?,?,?)`,
         [requesterId,bloodGroup,state,city,contact,status]
      );
      console.log("Blood Request added successfully!");

      const [validDonors] = await conn.execute(
         `SELECT
            d.donor_id,
            u.name,
            u.email,
            u.phone,
            d.blood_group,
            u.state,
            u.city,
            d.last_donation_date,
            d.total_donations
         FROM
            donors d
         JOIN
            users u ON d.user_id = u.user_id
         WHERE
            d.blood_group = ? -- Replace ? with the required blood group
            AND u.state = ?   -- Replace ? with the required state
            AND u.city = ?    -- Replace ? with the required city
            AND DATE(d.last_donation_date) <= DATE_SUB(CURDATE(),
INTERVAL 3 MONTH);
`,
         [bloodGroup,state,city]
      );
      console.log(validDonors);

      const notificationQuery = `
         INSERT INTO notifications (request_id, donor_id, status, notified_at)
         VALUES (?, ?, 'pending', NOW());
      `;
```

```
        for (const donor of validDonors) {
            await conn.execute(notificationQuery, [result.insertId,
donor.donor_id]);
        }
        await conn.commit();
        console.log("Notification added successfully!");
        return {
            id: result.insertId,
            requesterId: requesterId,
            name: name,
            email: email,
            bloodGroup: bloodGroup,
            state: state,
            city: city,
            contact: contact,
            status: status,
            donors: validDonors.map((donor) => ({ donorId: donor.donor_id,name:
donor.name,phone: donor.phone, email: donor.email, status: 'pending',
notifiedAt: new Date() })),

        };
    } catch (error) {
        await conn.rollback();
        console.error("Error adding request:", error);
        throw new AppError("Error adding request:", 400);
    }
    finally {
        conn.release();
    }
}
```

- **Email notification system using Nodemailer.**

```
export const sendNotificationEmail = async (email, name, requesterName,
bloodGroup, contact) => {
    const url = process.env.CLIENT_URL || "http://localhost:5173"; // replace
with your client URL
    const organizationName = process.env.ORGANIZATION_NAME ||
"BloodLine";
    const transporter = nodemailer.createTransport({
        service: "gmail",
        auth: {
            user: process.env.EMAIL,
            pass: process.env.PASSWORD
        }
    });

    const mailOptions = {
        from: `"${organizationName}" <${process.env.EMAIL}>`,
```

```
        to: email,
        subject: "Blood Request Notification",
        html: Notification_Email_Template.replace("[Donor Name]",
name).replace("[Requester Name]", requesterName).replace("[accepter]",
requesterName).replace("[Blood Group]", bloodGroup).replace("[Requester
Number]", contact).replace("[BloodLine Website URL]", url)
    }

    transporter.sendMail(mailOptions, (error,info) => {
        if(error) {
            console.error("Error sending Notification email", error);
            throw new AppError(`Error sending Notification email ${error}`)
        } else {
            console.log("Email sent successfully", info.response);
        }
    })
}
```

- **SQL schema for managing user data**

```
const [validDonors] = await conn.execute(
          `SELECT
             d.donor_id,
             u.name,
             u.email,
             u.phone,
             d.blood_group,
             u.state,
             u.city,
             d.last_donation_date,
             d.total_donations
          FROM
             donors d
          JOIN
             users u ON d.user_id = u.user_id
          WHERE
             d.blood_group = ? -- Replace ? with the required blood group
             AND u.state = ?   -- Replace ? with the required state
             AND u.city = ?    -- Replace ? with the required city
             AND   DATE(d.last_donation_date)   <=   DATE_SUB(CURDATE(),
INTERVAL 3 MONTH);
`,
          [bloodGroup,state,city]
       );
```