**NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES - FAST**



# Project Report CS-2005

# Database System

**Teacher: Miss Filza Akhlaq**

**Section: BS(CS)-5G**

**Semester: 7th semester Fall 2024**

**Blood Bank Management System**

**Member 1: 21K-3263 Uzair Shahzad**

**Member 2: 21K-3267 Muhammad Saad**

# Introduction

In an era where technological advancements play a pivotal role in transforming healthcare systems, we have tried to enhance the efficiency and accessibility of blood donation services. Built on the powerful trio of Node.js, HTML, CSS, and JavaScript, this innovative system redefines the way we approach blood donation, effortlessly connecting donors, recipients, and administrators in a user-friendly digital environment.

# Problem

Despite the critical importance of blood donation, the existing blood bank management systems has a loop-hole. Traditional approach is usually slow to address the dynamic nature of blood supply and demand. The absence of a unified platform often results in communication gaps between donors, recipients, and administrators, leading to potential delays in critical situations. The need of for centralized blood bank is evident keeping in mind the loop-holes of traditional blood bank system. Providing user friendly interface which interacts with user taking input for donor, recipient and admin.

# Objectives

1. **System overview:** Blood bank contains admin who will monitor donor and recipient.
2. **Technological frameworks:** Node.js, HTML, CSS, and JavaScript is used in creating a robust and user-friendly system.
3. **User roles and features:** Donor will register as a donor and recipient who is need of blood will register as a recipient and admin will accept or reject accordingly.
4. **Data management and security:**  Ensures robust measure to secure sensitive information of donor and recipient which is only visible to admin.
5. **User interface and experience:**  An intuitive interface for seamless user interaction and smooth data entry.
6. **Insightful reporting:** Holds a record and maintain logs which contains all the transactions.

# Methodologies

We designed a user-friendly interface that accommodates the diverse needs of donors, recipients, and administrators. This approach aimed to make interactions intuitive and efficient, promoting widespread system adoption. The project integrated Agile methodologies for feasible development Usability testing played a crucial role in refining the user interface and overall system functionality, we as a team member reviewed it as a user and worked on the improvement.

# System and Environment:

1. **Database Interface**

The system interfaces with a MySQL database server via XAMPP, hosted locally. The database server manages donor recipient and admin and is facilitated through SQL queries, with communication streamlined using the Node.js MySQL library.

1. **Frontend Interface**

The frontend of the system is crafted using HTML, CSS, and JavaScript, providing an intuitive user interface. This interface communicates with the backend through API calls, utilizing JavaScript's fetch API for seamless HTTP requests. The system ensures compatibility with contemporary web standards, leveraging HTML5, CSS3 and

JavaScript.

1. **Backend interface**

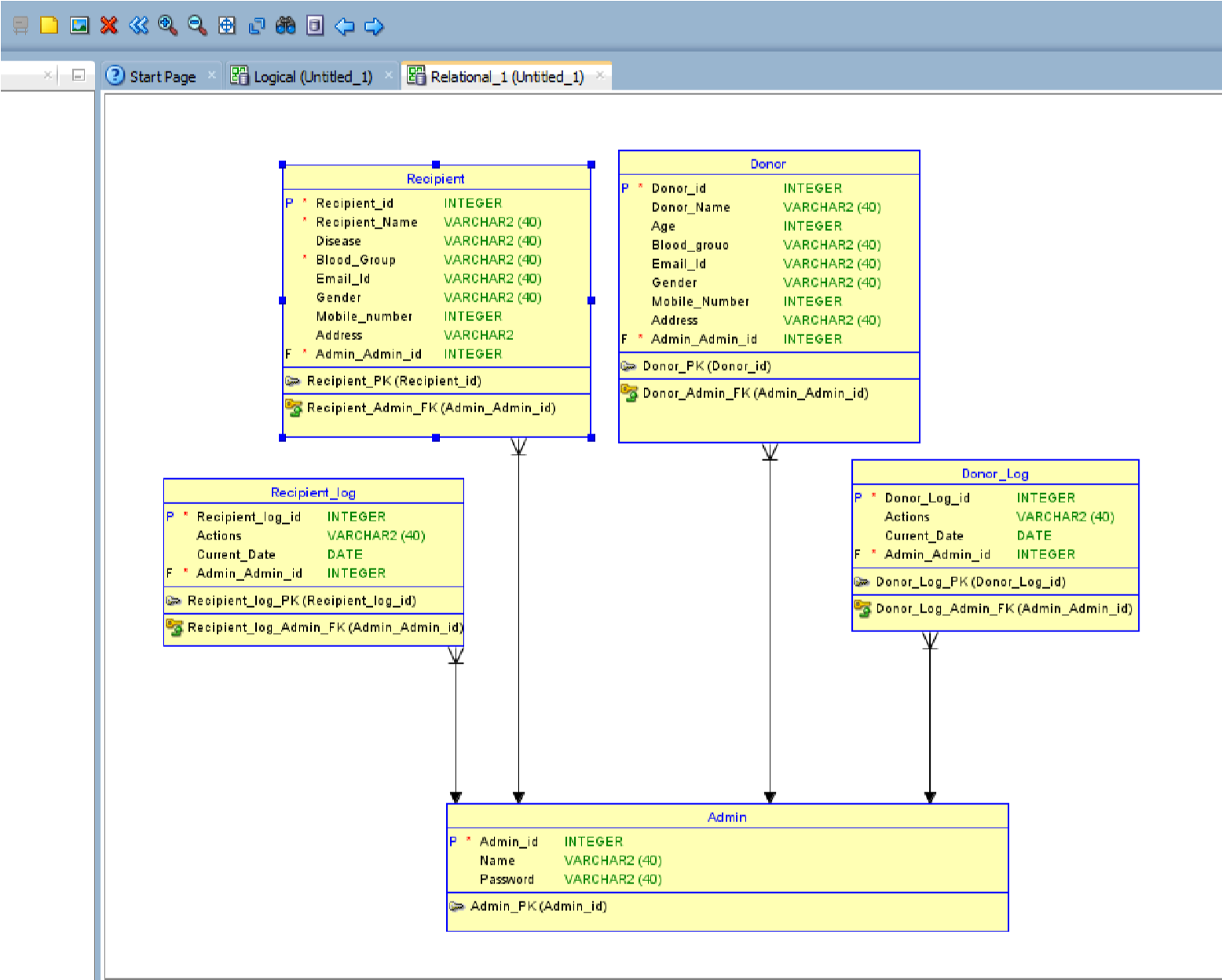
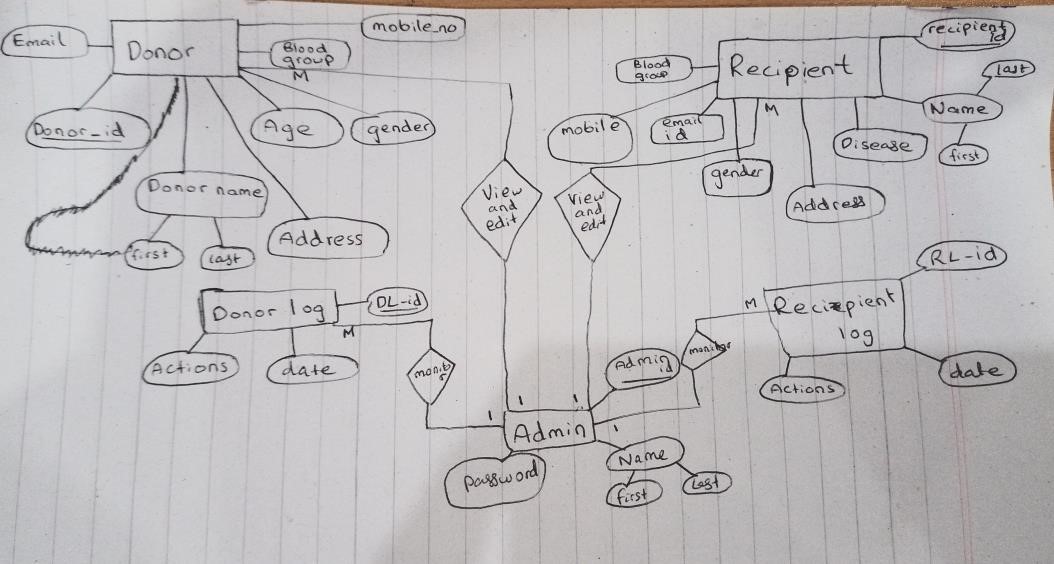
The backend infrastructure is developed using JavaScript and Node.js, a runtime for server-side JavaScript. The system exposes RESTful APIs that the frontend utilizes for various functionalities like data retrieval, donor registration, recipient registration, updating (accept or reject).

# Communications Interfaces

1. **API Communication:** Communication between the frontend and backend of the Blood Bank Management System occurs through RESTful APIs. The system exchanges data in JSON format, adhering to REST principles to ensure stateless communication. This approach not only enhances scalability but also promotes maintainability, allowing for efficient interaction between different systems components.
2. **Security measures:** To safeguard the confidentiality and integrity of data during communication, the Blood Bank Management System implements HTTPS for all interactions between the client and server. SSL/TLS protocols are employed to provide robust encryption, ensuring the secure transmission of sensitive information. This security measure is fundamental in maintaining the trustworthiness of the blood bank system.

1. **Data transferring rates:** The system is optimized for efficient data transfer rates between the frontend and backend, ensuring a responsive user experience. Designed to handle concurrent requests and responses, the Blood Bank Management System prioritizes swift and reliable communication.

**ER Diagram**



# Normalization

**1: Functional Dependencies:**

Admin\_id ---> A\_name, A\_password

Donor\_id ---> D\_name , D\_email, D\_age, D\_gender, D\_bloodgroup, D\_mobileno, D\_address

Recipient\_id ---> R\_name , R\_email, R\_disease, R\_gender, R\_bloodgroup, R\_mobileno, R\_address

DonorLog\_id ---> DL\_action, DL\_date

RecipientLog\_id ---> RL\_action, RL\_date

**2: Primary Keys:**

Admin\_id, Donor\_id, Recipient\_id, DonorLog\_id, RecipientLog\_id

1. **Admin Table (Assuming each admin has a unique ID):**

Admin (Admin\_id, A\_name, A\_password)

This table is already in 1NF and 2NF since there are no repeating groups or partial dependencies. Admin\_id serves as the primary key.

1. **Donor Table:**

Donor (Donor\_id, D\_name, D\_email, D\_age, D\_gender, D\_bloodgroup, D\_mobileno, D\_address)

This table is already in 1NF as each attribute contains atomic values. To achieve 2NF, we need to make sure that there are no partial dependencies. The primary key is Donor\_id.

1. **Recipient Table:**

Recipient (Recipient\_id, R\_name, R\_email, R\_disease, R\_gender, R\_bloodgroup, R\_mobileno, R\_address)

This table is already in 1NF as each attribute contains atomic values. To achieve 2NF, we need to make sure that there are no partial dependencies. The primary key is Recipient\_id.

1. **DonorLog Table:**

DonorLog (DonorLog\_id, Donor\_id, DL\_action, DL\_date)

This table is already in 1NF as each attribute contains atomic values. To achieve 2NF, we need to make sure that there are no partial dependencies. The primary key is DonorLog\_id.

1. **RecipientLog Table:**

RecipientLog (RecipientLog\_id, Recipient\_id, RL\_action, RL\_date)

This table is already in 1NF as each attribute contains atomic values. To achieve 2NF, we need to make sure that there are no partial dependencies. The primary key is RecipientLog\_id.

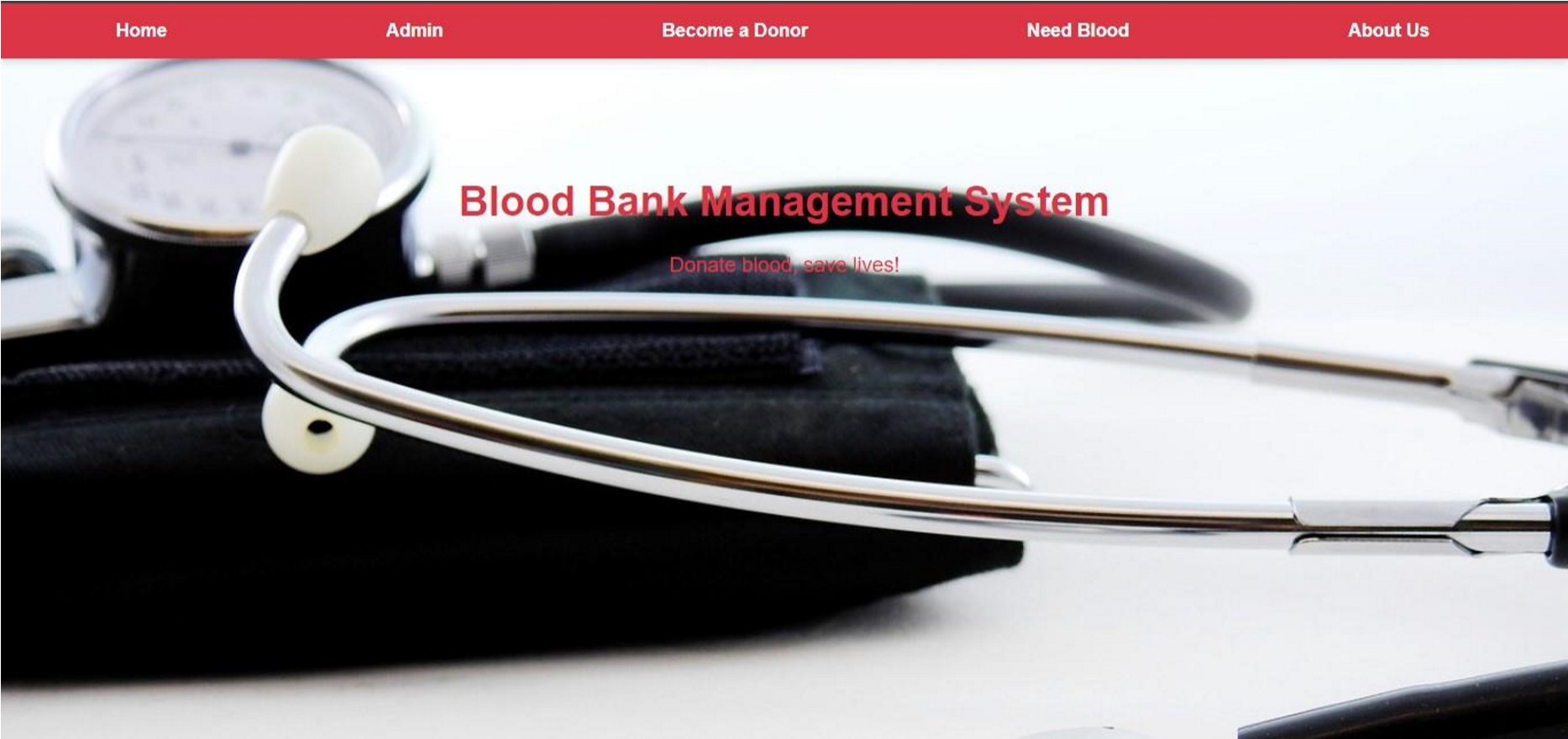
Now, let's check for 3NF:

In each table, there should be no transitive dependencies. That means every nonprime attribute must be non-transitively dependent on every superkey. No transitive dependencies are present in the current tables, so they are in 3NF.

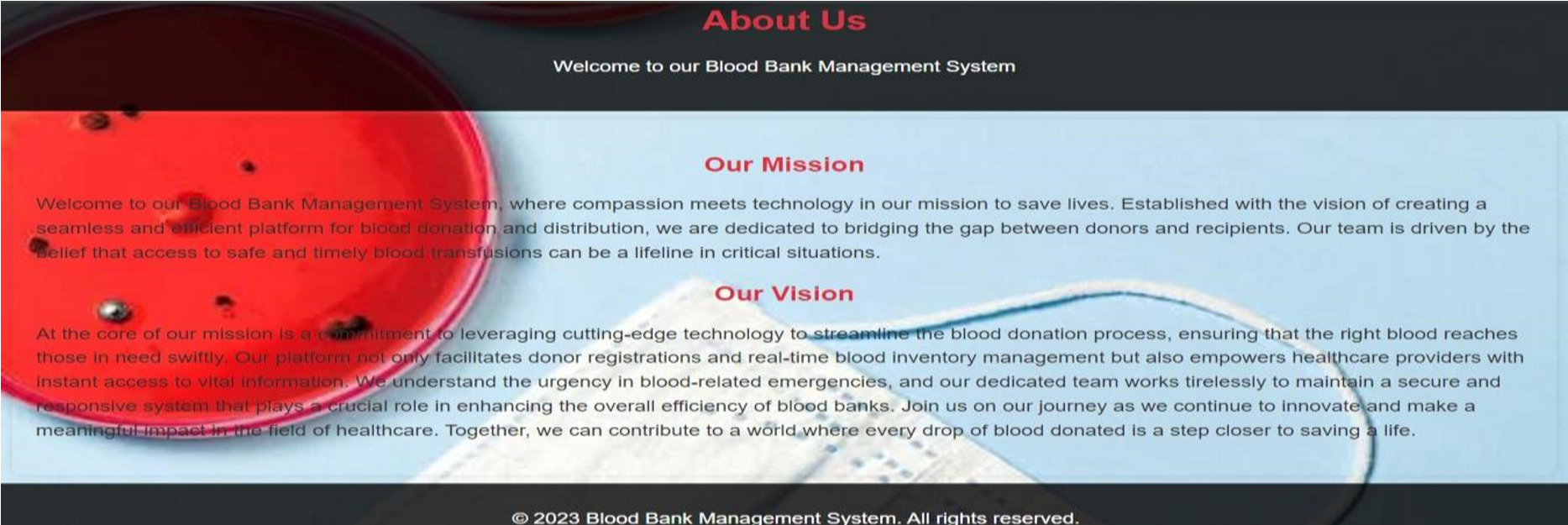
The tables are now normalized up to 3NF.

# Project Output

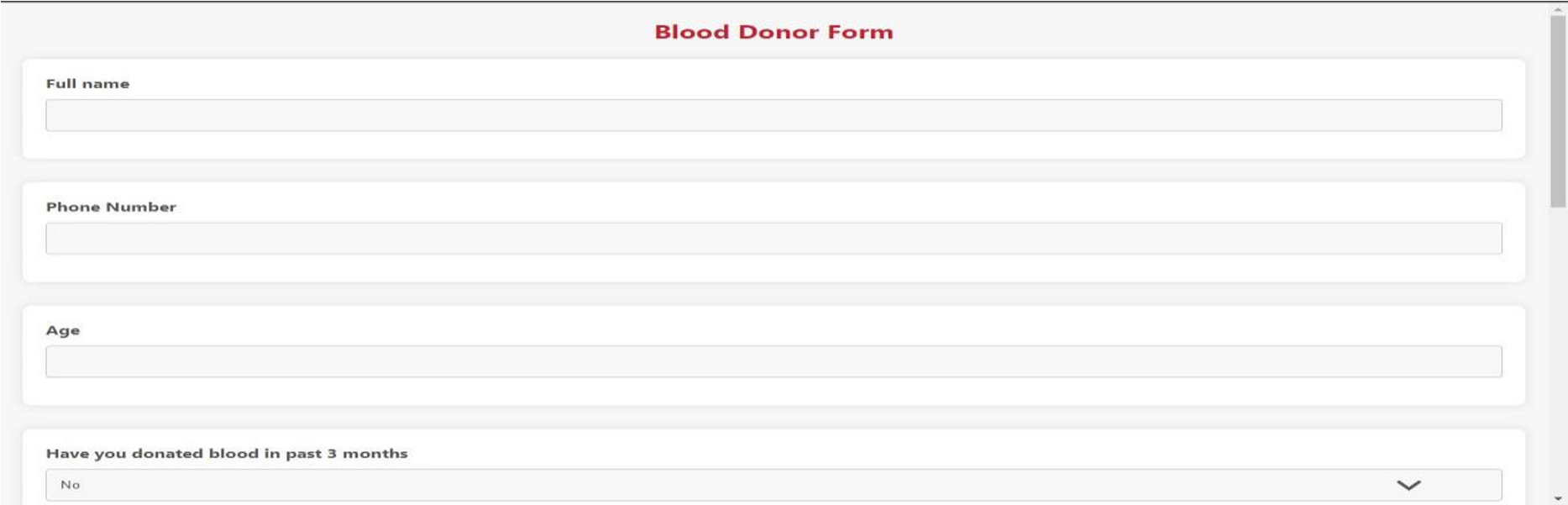
User Home Page:



About Us:



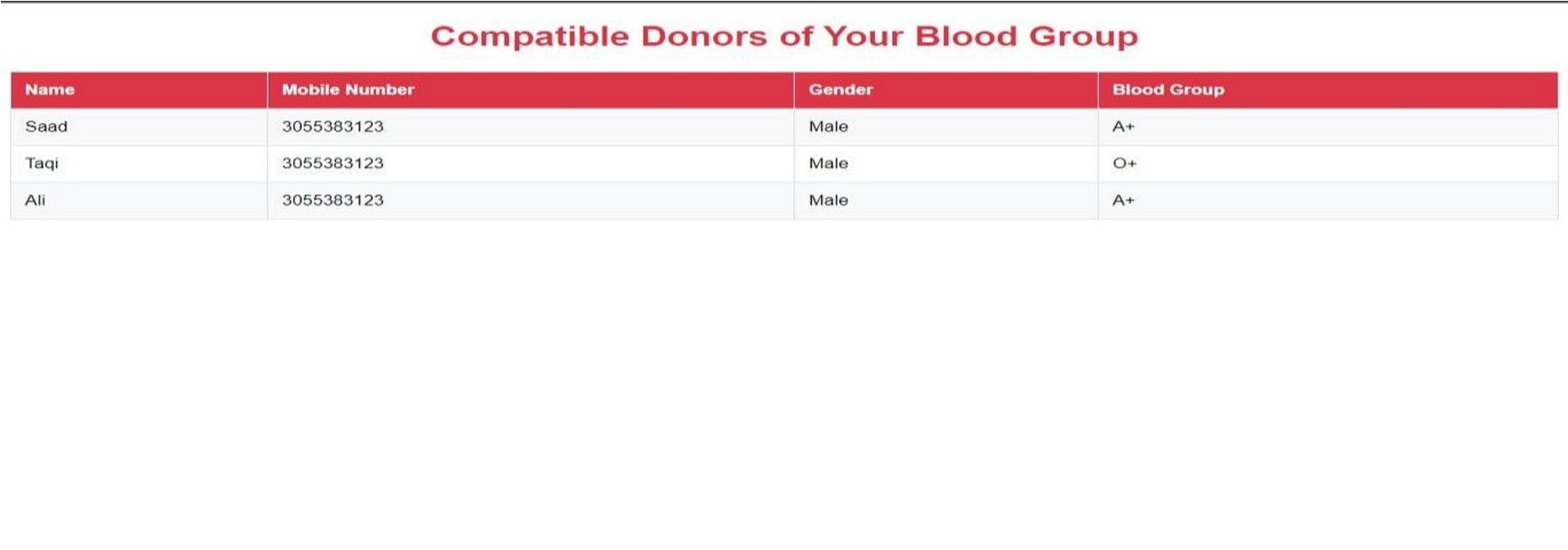
Donor Form:



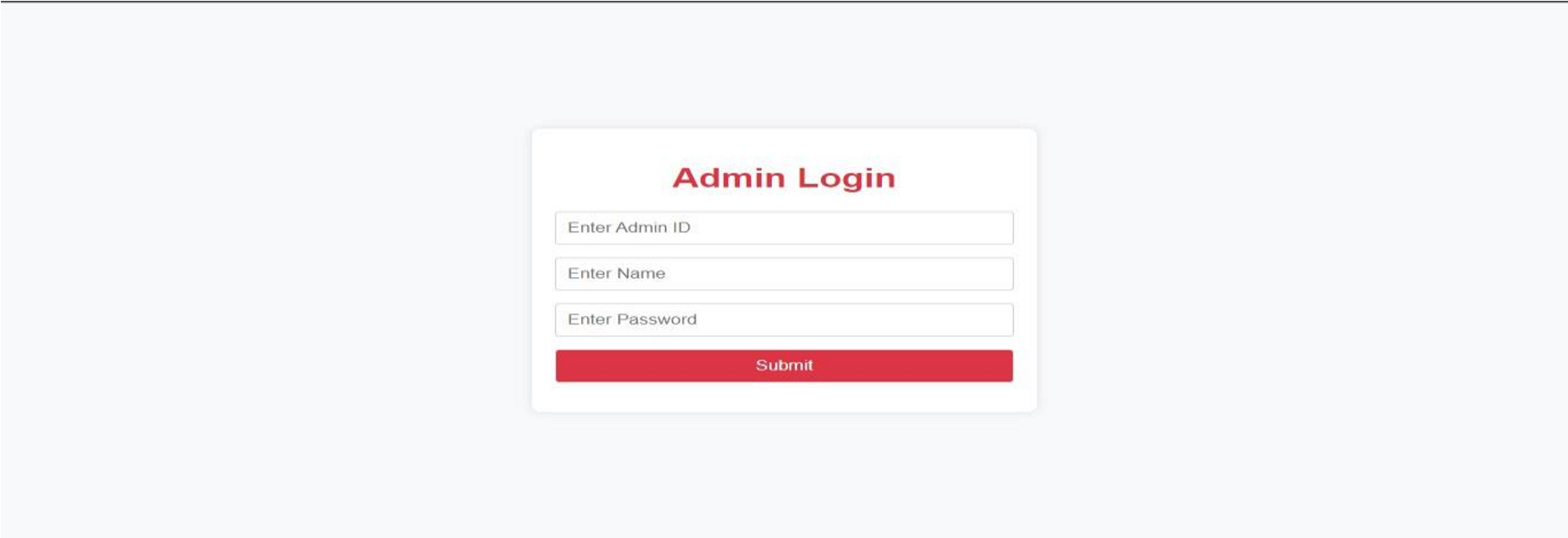
Recipient Form:



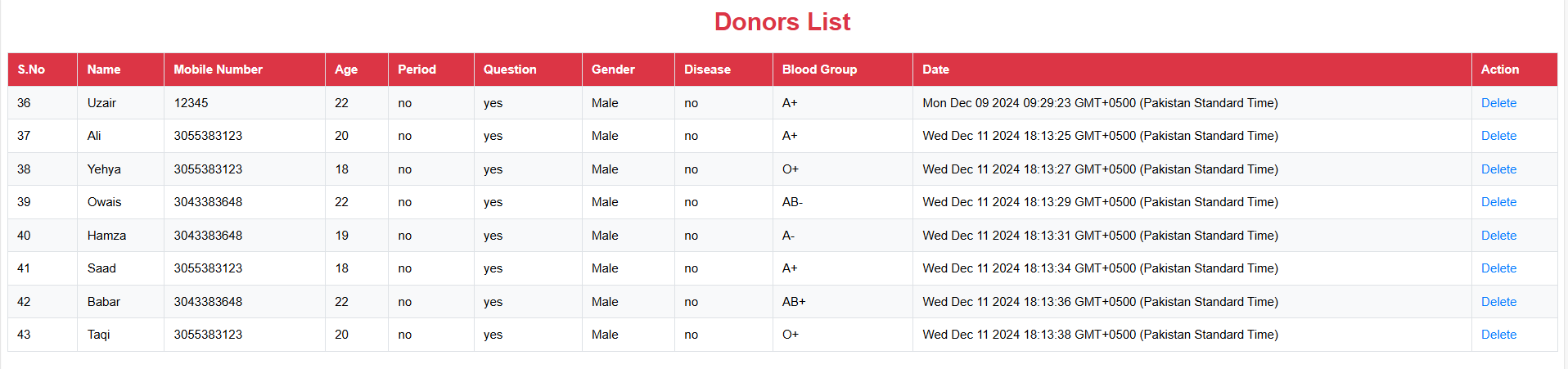
Compatible Donor of your Blood:



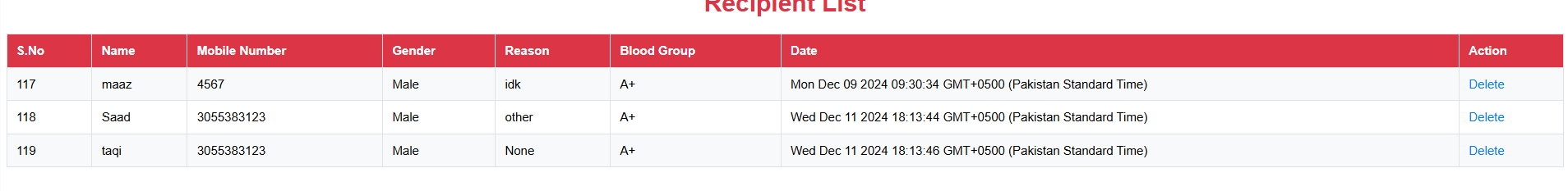
Admin Login Page:



Donor List:



Recipient List:



Recipient Logs:



Donor Logs:

