**Maratha Vidya Prasarak Samaj’s**

**Karmaveer Adv. Baburao Ganpatrao Thakare College of Engineering, Nashik 422013**

Department Of Information Technology

**Semester IV [2024-25]**



A

MINI PROJECT REPORT ON

“BLOOD MANAGEMENT SYSTEM”

Submitted by

**Piyush Kulkarni**

**Vedant Jadhav**

**Nikita More**

**Nikhil Kathale**

**UNDER THE GUIDANCE OF**

**Dr.V.R.Sonawane**

# Content

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Name of the Topic** | **Page No.** |
| **1.** | **Acknowledgement Abstract** |  |
| 2. | **Introduction** |  |
| 3. | **Objective of the Project** |  |
| 4. | **Functional and Non Functional Requirement** |  |
| 5. | **Data Requirement** |  |
| 6. | **Entity Type** |  |
| 7. | **Entity Relationship (ER) Diagram** |  |
| 8. | **Database Schema (Tables & Relationships)** |  |
| 9. | **Table Creation and Data Insertion** |  |
| 10. | **Queries** |  |
| 11. | **Functional Dependencies** |  |
| 12. | **Front-End Design (Screenshots & Description)** |  |
| 13. | **Back-End Logic (Stored Procedures/PL-SQL/Code Snippets)** |  |
| 14. | **Conclusion** |  |
| 15. | **References** |  |

2

# Acknowledgement

*I take this occasion to thank God, almighty for blessing us with his grace and taking our endeavour to a successful culmination. I extend my sincere and heartfelt thanks to our esteemed guide,* ***Dr VR Sonawane****, for providing me with the right guidance and advice at the crucial junctures and for showing me the right way. I would like to thank the other faculty members also, at this occasion. Last but not the least, I would like to thank my friends for the support and encouragement they have given me during the course of our work.*

Piyush Kulkarni Vedant Jadhav Nikita More

Nikhil Kathale

3

# Abstract

Blood plays an essential role in saving lives during medical emergencies such as accidents, surgeries, childbirth complications, and critical illnesses like cancer and thalassemia. Despite numerous awareness campaigns promoting voluntary blood donation, the shortage of readily available blood continues to pose a significant challenge. The Blood Donation Management System is conceptualized and designed to address this pressing issue by providing a digital platform where willing donors and individuals in need can connect efficiently. Through an easy-to-navigate user interface, potential donors can register their personal details, including blood group and city, while recipients can search for suitable matches based on specific criteria. Unlike traditional, manually maintained blood donor records that are often outdated or inaccessible, our system ensures that the information is current, secure, and easily retrievable. Furthermore, the system automates the request and notification process, reducing delays in communication. It focuses on scalability, security, responsiveness, and user experience, ensuring that it caters to a broad audience. By developing this platform, we aim not just to save lives but also to instill a culture of regular voluntary blood donation, encouraging individuals to come forward proactively. Ultimately, the Blood Donation Management System aspires to bridge the gap between blood demand and supply, leveraging the potential of technology for the betterment of society.

**Introduction**

The necessity for a well-structured blood donation system has never been more critical than in today’s rapidly evolving world. As urbanization increases and medical treatments advance, the demand for blood components has also grown exponentially. However, the systems in place to facilitate blood donation and distribution have often been marred by inefficiencies, communication gaps, and logistical challenges. Recognizing this gap, we embarked on creating the Blood Donation Management System—an innovative, user-centric platform designed to act as a bridge between donors and recipients. This platform integrates core principles of database management systems, web development, and user experience design to provide a robust and reliable service. By allowing users to register, search, and interact through a clean, intuitive interface, it reduces the friction traditionally associated with blood donation drives. Additionally, the platform's emphasis on accurate, real-time data ensures that critical time is not lost during emergencies. By bringing this solution to life, we not only hope to save countless lives but also to contribute to creating an informed, proactive society where voluntary blood donation is normalized and encouraged. This project also serves as a practical implementation of our technical learnings, combining theory with hands-on application in database design, backend logic development, and front-end user interaction.

**Objectives**

The primary objective of the Blood Donation Management System is to develop a centralized, reliable, and secure platform that effectively connects blood donors with recipients in need, thereby significantly reducing the time taken to find suitable blood donors during critical situations. Our secondary objectives include creating a database-driven system that can manage a large volume of user data efficiently, ensuring fast search functionalities that allow users to filter donors based on blood group and city. The system also seeks to automate communication by allowing users to send connection requests directly to donors, thus eliminating the need for intermediaries and reducing the risk of information delays. Additionally, we aim to foster awareness regarding blood donation by providing a seamless platform that encourages more individuals to register as donors. Other technical objectives include maintaining high standards of security, responsiveness across devices (desktops, tablets, mobiles), scalability to accommodate increasing user bases, and robust backend architecture capable of handling real-time queries without performance degradation. Ultimately, this project endeavors to be more than just a technical exercise; it seeks to bring meaningful societal impact by facilitating faster, easier, and more organized blood donation processes across communities.

**Functional Requirements**

1. **Donor Registration**:  
   The system must allow users to register as donors by providing essential information such as full name, age, gender, blood group, contact number, address, and city. This data is stored in a secure database and made available for future search queries by recipients.
2. **Login and Authentication**:  
   Users and administrators must be able to securely log into the system using credentials. The system validates login attempts and provides access based on role—ensuring that private donor data is not accessible without authorization.
3. **Search Donor by Criteria**:  
   Recipients or users in need of blood should be able to search for donors by entering relevant criteria like blood group and location (city or region). The system filters the database and displays a list of matching, active donors.
4. **Send Blood Request to Donors**:  
   Registered users can send connection or blood requests to available donors. These requests are stored in the backend and their status (pending, accepted, declined) can be tracked by both parties.
5. **Request Management Dashboard**:  
   Users can view the list of all requests they have sent and received, along with their current status. This helps both donors and recipients manage communication efficiently.
6. **Admin Panel for User Management**:  
   An administrative backend allows system admins to manage registered users, approve or deny suspicious accounts, delete inactive donors, and monitor activity on the platform.
7. **Notification System**:  
   The system notifies donors of new requests either via email or through the platform interface. These notifications improve engagement and help ensure timely responses.

**Non-Functional Requirements**

1. **Usability and User Interface (UI/UX)**:  
   The website must be user-friendly and designed to cater to a wide range of users, including those who are not technically skilled. Interfaces must be intuitive, clean, and accessible with responsive layouts for mobile and desktop.
2. **Performance and Speed**:  
   The application should be optimized for speed. Searches, page loads, and form submissions must happen quickly (ideally under 2 seconds) even when the system scales to a large number of users and records.
3. **Scalability**:  
   The system must be built with scalability in mind—capable of supporting thousands of user accounts, multiple simultaneous queries, and expanding features such as live chat or location tracking in the future.
4. **Security and Data Protection**:  
   User information must be protected through encryption and secure login sessions. SQL injection, cross-site scripting (XSS), and other common web vulnerabilities must be prevented using best practices like prepared statements and input sanitization.
5. **Maintainability and Extensibility**:  
   The codebase must be well-documented and modular, allowing developers to add or modify features in the future without disrupting the existing functionalities.
6. **Reliability and Availability**:  
   The system should be online 24/7 and capable of handling faults gracefully. Backup mechanisms and redundant systems can ensure minimal downtime during server or database failures.
7. **Cross-Browser and Cross-Platform Compatibility**:  
   The application must function consistently across all major browsers (Chrome, Firefox, Safari, Edge) and operating systems (Windows, macOS, Linux, Android, iOS).

**Data Requirements**

Data is the backbone of the Blood Donation Management System. Accurate, well-structured, and efficiently retrievable data is crucial to ensure quick access during emergencies and effective management of donor-recipient interactions. The system requires both static and dynamic data types to support its functional and non-functional requirements.

* Primary Data Requirements:

1. Donor Information:
   * Full Name
   * Age
   * Gender
   * Blood Group (A+, A-, B+, B-, AB+, AB-, O+, O-)
   * Contact Number
   * Email ID (optional)
   * City and Address
   * Date of Last Donation
   * Donor Status (Active/Inactive) This dataset enables the platform to store and retrieve details about blood donors quickly, ensuring users can search for relevant donors based on need and location.
2. Recipient/Blood Request Data:
   * Requester’s Name
   * Blood Group Needed
   * Quantity (in units)
   * Purpose of Requirement (e.g., surgery, accident, chronic illness)
   * Preferred City/Hospital
   * Contact Number
   * Date of Requirement
   * Urgency Level (Emergency/Normal)
   * Request Status (Pending/Accepted/Declined) This information allows the system to record and track requests sent by recipients to potential donors.
3. User Login Credentials:
   * User ID or Email
   * Password (Encrypted/Hashed)
   * Role (Admin/User) These credentials are essential to authenticate users and determine access levels to different parts of the application.
4. Admin and Moderation Logs:
   * Admin Username
   * Actions Performed (e.g., User Deletion, Approval, System Update)
   * Timestamp of Actions Maintaining an audit trail of administrative actions ensures security and traceability.
5. Notifications and Messages:
   * Sender ID
   * Receiver ID
   * Message Content
   * Date & Time Sent
   * Message Status (Read/Unread) This helps in communication between users, especially when donors and recipients need to coordinate in real time.

* Secondary Data Requirements:

1. Blood Inventory Data (Optional for Expansion):
   * Blood Bank Name
   * Blood Group Availability
   * Units in Stock
   * Expiry Date
   * Location This data can help expand the project by integrating with hospitals and real blood banks in the future.
2. Feedback and Ratings:
   * User ID
   * Donor/Receiver ID
   * Rating (1 to 5 stars)
   * Review Message
   * Date of Feedback Helps build trust and transparency in the platform, especially when choosing donors.
3. Location Data and City List:
   * List of cities/states
   * Postal codes (for future integration with maps or delivery services)
   * Latitude/Longitude (for GPS mapping and distance-based searching, future feature)
4. Session and Activity Logs:
   * Login Timestamps
   * IP Address
   * Page Visited
   * Session Duration This data helps monitor usage patterns, improve system performance, and enhance security.

**Entity Types**

Entity types are the core structural components of any database model. In our Blood Donation Management System, each entity represents a real-world object or concept that needs to be stored, retrieved, and managed in the system. Below is a point-by-point breakdown of each major entity type involved in the project, along with a description of its attributes and importance.

* Primary Entity Types:

1. Donor
   * Description: Represents an individual who has volunteered to donate blood.
   * Key Attributes:
     + Donor\_ID (Primary Key)
     + Name
     + Age
     + Gender
     + Blood Group
     + Contact Number
     + Email ID
     + Address
     + City
     + Last Donation Date
     + Availability Status (Active/Unavailable)
   * Purpose: Stores detailed information to enable matching with recipients based on blood type, location, and availability.
2. Recipient / Blood Request
   * Description: Represents the person or entity requesting blood from the system.
   * Key Attributes:
     + Request\_ID (Primary Key)
     + Name
     + Blood Group Required
     + Units Needed
     + Contact Number
     + Reason for Request
     + City/Hospital
     + Date of Requirement
     + Urgency Level
     + Request Status (Pending/Accepted/Declined)
   * Purpose: To log and manage blood requests, allowing the system to match with available donors.
3. User / Login
   * Description: Represents a system user—either a donor, recipient, or administrator.
   * Key Attributes:
     + User\_ID (Primary Key)
     + Username / Email
     + Password (Encrypted)
     + Role (Admin / Donor / General User)
     + Registration Date
   * Purpose: Ensures only authorized users can access system features based on roles.
4. Admin
   * Description: A specialized user with access to manage backend processes and oversee the system.
   * Key Attributes:
     + Admin\_ID (Primary Key)
     + Name
     + Email
     + Actions Taken (Logs)
     + Timestamp of Actions
   * Purpose: To perform moderation, monitor activities, and maintain system integrity.

* Secondary / Supporting Entity Types:

1. Notification
   * Description: Represents communications triggered between system components or users.
   * Key Attributes:
     + Notification\_ID
     + Sender\_ID
     + Receiver\_ID
     + Message Content
     + Timestamp
     + Read\_Status
   * Purpose: Keeps users informed and supports communication for request acceptance or updates.
2. Feedback / Review
   * Description: Represents ratings and reviews provided by recipients for donors.
   * Key Attributes:
     + Feedback\_ID
     + User\_ID (Who is giving the feedback)
     + Donor\_ID (Who is being reviewed)
     + Rating (1–5 stars)
     + Comments
     + Date
   * Purpose: Builds user trust and improves transparency within the platform.
3. Session Log
   * Description: Tracks user sessions for security and analytics.
   * Key Attributes:
     + Session\_ID
     + User\_ID
     + IP Address
     + Login Time
     + Logout Time
   * Purpose: Maintains activity logs and helps in system performance evaluation.
4. Blood Inventory (Optional Future Entity)
   * Description: Can be integrated in future versions for hospitals or blood banks.
   * Key Attributes:
     + Inventory\_ID
     + Blood Group
     + Units Available
     + Expiry Date
     + Blood Bank Location
   * Purpose: To expand the system for real-time inventory monitoring of hospitals or blood banks.

**Database Schema**

The database schema forms the backbone of the Blood Donation Management System, defining how data is organized, stored, and related within the system. Each table corresponds to an entity, and foreign key relationships ensure data consistency and integrity. Below is a comprehensive explanation of all the major tables, their fields, data types, and relationships.

* Primary Tables:

1. Donor Table

Purpose: Stores all vital information related to blood donors.

CREATE TABLE Donor ( Donor\_ID INT PRIMARY KEY AUTO\_INCREMENT, FullName VARCHAR(100), Age INT, Gender VARCHAR(10), BloodGroup VARCHAR(5), ContactNumber VARCHAR(15), Email VARCHAR(100) UNIQUE, Address TEXT, City VARCHAR(50), LastDonationDate DATE, AvailabilityStatus ENUM('Available', 'Unavailable') );

1. Recipient Table

Purpose: Contains details of users requesting blood donations.

CREATE TABLE Recipient ( Request\_ID INT PRIMARY KEY AUTO\_INCREMENT, FullName VARCHAR(100), BloodGroupRequired VARCHAR(5), UnitsNeeded INT, ContactNumber VARCHAR(15), ReasonForRequest TEXT, City VARCHAR(50), HospitalName VARCHAR(100), DateOfRequirement DATE, UrgencyLevel ENUM('High', 'Medium', 'Low'), RequestStatus ENUM('Pending', 'Accepted', 'Declined') );

1. Users Table

Purpose: Manages login credentials and user roles.

CREATE TABLE Users ( User\_ID INT PRIMARY KEY AUTO\_INCREMENT, Email VARCHAR(100) UNIQUE, Password VARCHAR(255), Role ENUM('Admin', 'Donor', 'General'), RegistrationDate DATETIME );

1. Admin Table

Purpose: Logs activities performed by the administrator for auditing purposes.

CREATE TABLE Admin ( Admin\_ID INT PRIMARY KEY AUTO\_INCREMENT, FullName VARCHAR(100), Email VARCHAR(100) UNIQUE, ActionTaken TEXT, Timestamp DATETIME );

* Supporting Tables:

1. Notification Table

Purpose: Tracks internal communications such as blood request updates or approval notices.

CREATE TABLE Notification ( Notification\_ID INT PRIMARY KEY AUTO\_INCREMENT, Sender\_ID INT, Receiver\_ID INT, Message TEXT, SentTime DATETIME, ReadStatus ENUM('Read', 'Unread'), FOREIGN KEY (Sender\_ID) REFERENCES Users(User\_ID), FOREIGN KEY (Receiver\_ID) REFERENCES Users(User\_ID) );

1. Feedback Table

Purpose: Stores user ratings and feedback for transparency and trust-building.

CREATE TABLE Feedback ( Feedback\_ID INT PRIMARY KEY AUTO\_INCREMENT, User\_ID INT, Donor\_ID INT, Rating INT CHECK (Rating BETWEEN 1 AND 5), Comments TEXT, DateGiven DATE, FOREIGN KEY (User\_ID) REFERENCES Users(User\_ID), FOREIGN KEY (Donor\_ID) REFERENCES Donor(Donor\_ID) );

1. SessionLog Table

Purpose: Maintains logs of user logins/logouts for security and system analysis.

CREATE TABLE SessionLog ( Session\_ID INT PRIMARY KEY AUTO\_INCREMENT, User\_ID INT, IPAddress VARCHAR(50), LoginTime DATETIME, LogoutTime DATETIME, FOREIGN KEY (User\_ID) REFERENCES Users(User\_ID) );

* Optional (for future enhancements):

1. BloodInventory Table

Purpose: Tracks the availability of blood units for blood banks or hospitals.

CREATE TABLE BloodInventory ( Inventory\_ID INT PRIMARY KEY AUTO\_INCREMENT, BloodGroup VARCHAR(5), UnitsAvailable INT, ExpiryDate DATE, Location VARCHAR(100) );

* Relationships Summary:
* Donor ↔ Feedback: One donor can receive many reviews.
* Users ↔ Notification: One user can send and receive multiple notifications.
* Users ↔ SessionLog: One user can have many login sessions.
* Recipient ↔ Request: A recipient makes one or many blood requests.
* Admin ↔ Actions: Admin entries track the history of system interventions.

Table Creation

This section outlines the SQL commands used to create the necessary tables for the Blood Donation Management System. These tables are the backbone of the system and are used to manage donor information, recipient requests, user authentication, notifications, and feedback.

Table Creation Statements

1. Donor Table

CREATE TABLE Donor ( Donor\_ID INT PRIMARY KEY AUTO\_INCREMENT, FullName VARCHAR(100), Age INT, Gender VARCHAR(10), BloodGroup VARCHAR(5), ContactNumber VARCHAR(15), Email VARCHAR(100) UNIQUE, Address TEXT, City VARCHAR(50), LastDonationDate DATE, AvailabilityStatus ENUM('Available', 'Unavailable') );

1. Recipient Table

CREATE TABLE Recipient ( Request\_ID INT PRIMARY KEY AUTO\_INCREMENT, FullName VARCHAR(100), BloodGroupRequired VARCHAR(5), UnitsNeeded INT, ContactNumber VARCHAR(15), ReasonForRequest TEXT, City VARCHAR(50), HospitalName VARCHAR(100), DateOfRequirement DATE, UrgencyLevel ENUM('High', 'Medium', 'Low'), RequestStatus ENUM('Pending', 'Accepted', 'Declined') );

1. Users Table

CREATE TABLE Users ( User\_ID INT PRIMARY KEY AUTO\_INCREMENT, Email VARCHAR(100) UNIQUE, Password VARCHAR(255), Role ENUM('Admin', 'Donor', 'General'), RegistrationDate DATETIME );

1. Notification Table

CREATE TABLE Notification ( Notification\_ID INT PRIMARY KEY AUTO\_INCREMENT, Sender\_ID INT, Receiver\_ID INT, Message TEXT, SentTime DATETIME, ReadStatus ENUM('Read', 'Unread'), FOREIGN KEY (Sender\_ID) REFERENCES Users(User\_ID), FOREIGN KEY (Receiver\_ID) REFERENCES Users(User\_ID) );

1. Feedback Table

CREATE TABLE Feedback ( Feedback\_ID INT PRIMARY KEY AUTO\_INCREMENT, User\_ID INT, Donor\_ID INT, Rating INT CHECK (Rating BETWEEN 1 AND 5), Comments TEXT, DateGiven DATE, FOREIGN KEY (User\_ID) REFERENCES Users(User\_ID), FOREIGN KEY (Donor\_ID) REFERENCES Donor(Donor\_ID) );

1. SessionLog Table

CREATE TABLE SessionLog ( Session\_ID INT PRIMARY KEY AUTO\_INCREMENT, User\_ID INT, IPAddress VARCHAR(50), LoginTime DATETIME, LogoutTime DATETIME, FOREIGN KEY (User\_ID) REFERENCES Users(User\_ID) );

Optional: BloodInventory Table

CREATE TABLE BloodInventory ( Inventory\_ID INT PRIMARY KEY AUTO\_INCREMENT, BloodGroup VARCHAR(5), UnitsAvailable INT, ExpiryDate DATE, Location VARCHAR(100) );

Sample Data Insertion

Donor Table Sample Data

INSERT INTO Donor (FullName, Age, Gender, BloodGroup, ContactNumber, Email, Address, City, LastDonationDate, AvailabilityStatus) VALUES ('Ravi Kulkarni', 29, 'Male', 'O+', '9999988888', 'ravi.kulkarni@example.com', '123 MG Road', 'Pune', '2024-12-10', 'Available'), ('Sneha Sharma', 24, 'Female', 'A+', '9898989898', 'sneha.sharma@example.com', '56 Nehru Street', 'Nashik', '2025-01-15', 'Unavailable');

Users Table Sample Data

INSERT INTO Users (Email, Password, Role, RegistrationDate) VALUES ('admin@example.com', 'adminpass', 'Admin', NOW()), ('ravi.kulkarni@example.com', 'ravi123', 'Donor', NOW()), ('sneha.sharma@example.com', 'sneha123', 'Donor', NOW());

Recipient Table Sample Data

INSERT INTO Recipient (FullName, BloodGroupRequired, UnitsNeeded, ContactNumber, ReasonForRequest, City, HospitalName, DateOfRequirement, UrgencyLevel, RequestStatus) VALUES ('Anil Mehta', 'O+', 2, '9876543210', 'Accident injury', 'Mumbai', 'City Hospital', '2025-04-25', 'High', 'Pending'), ('Pooja Verma', 'B-', 1, '9765432109', 'Surgery requirement', 'Pune', 'Ruby Hall Clinic', '2025-04-28', 'Medium', 'Pending');

**Data Insertion Example**

Once the database schema and tables are created, inserting sample data is crucial to test functionality and verify that queries, relationships, and constraints work as intended. Below are example SQL statements used to populate key tables with representative data for the Blood Donation Management System. These data entries simulate real-world scenarios, such as donor registration, blood requests, user authentication, and feedback submission.

* 1. Donor Table – Sample Data

INSERT INTO Donor (FullName, Age, Gender, BloodGroup, ContactNumber, Email, Address, City, LastDonationDate, AvailabilityStatus) VALUES  
('Aarav Mehta', 30, 'Male', 'O+', '9876543210', 'aarav.mehta@example.com', '101 Gandhi Nagar', 'Mumbai', '2024-12-15', 'Available'), ('Ishita Rao', 25, 'Female', 'A+', '9765432190', 'ishita.rao@example.com', 'Block 5, Shivaji Park', 'Pune', '2025-01-10', 'Unavailable'), ('Sahil Kumar', 35, 'Male', 'B-', '9123456789', 'sahil.kumar@example.com', '42 Tagore Road', 'Nashik', '2025-02-20', 'Available');

* 1. Users Table – Sample Data

INSERT INTO Users (Email, Password, Role, RegistrationDate) VALUES  
('admin@bloodbank.com', 'adminsecure', 'Admin', NOW()), ('aarav.mehta@example.com', 'aaravpass', 'Donor', NOW()), ('pooja.verma@example.com', 'poojapass', 'General', NOW());

* 1. Recipient Table – Sample Data

INSERT INTO Recipient (FullName, BloodGroupRequired, UnitsNeeded, ContactNumber, ReasonForRequest, City, HospitalName, DateOfRequirement, UrgencyLevel, RequestStatus) VALUES  
('Pooja Verma', 'O+', 2, '8887766554', 'Emergency surgery', 'Mumbai', 'Fortis Hospital', '2025-04-25', 'High', 'Pending'), ('Manish Joshi', 'A+', 1, '8899776655', 'Accident trauma', 'Nashik', 'Wockhardt Hospital', '2025-04-27', 'Medium', 'Pending');

* 1. Notification Table – Sample Data

INSERT INTO Notification (Sender\_ID, Receiver\_ID, Message, SentTime, ReadStatus) VALUES  
(1, 2, 'New request for O+ blood near your location.', NOW(), 'Unread'), (1, 3, 'A+ blood needed urgently at Wockhardt Hospital.', NOW(), 'Unread');

* 1. Feedback Table – Sample Data

INSERT INTO Feedback (User\_ID, Donor\_ID, Rating, Comments, DateGiven) VALUES  
(3, 1, 5, 'Quick response and very helpful donor.', '2025-04-22'), (3, 2, 4, 'Polite and responsive, but delayed by an hour.', '2025-04-21');

* 1. Blood Inventory Table – Sample Data (Optional)

INSERT INTO BloodInventory (BloodGroup, UnitsAvailable, ExpiryDate, Location) VALUES  
('O+', 15, '2025-05-30', 'Mumbai Center'), ('A+', 10, '2025-05-20', 'Pune Center');

These sample data entries help establish a functional test environment and allow developers and testers to validate core features such as user interaction, donor matching, request tracking, and system notifications.

**SQL Queries**

This section outlines various SQL queries that demonstrate how the system retrieves, manipulates, and processes data in a real-world blood donation management scenario. These queries support essential operations such as donor search, recipient matching, notifications, authentication, and performance analysis.

* 1. Retrieve All Available Donors by Blood Group and City

SELECT FullName, BloodGroup, City, ContactNumber  
FROM Donor  
WHERE AvailabilityStatus = 'Available' AND BloodGroup = 'O+' AND City = 'Mumbai';

Purpose: To find all active O+ blood donors in Mumbai. This is useful when someone requests blood urgently in a specific region.

* 1. Count of Donors by Each Blood Group

SELECT BloodGroup, COUNT(\*) AS TotalDonors  
FROM Donor  
GROUP BY BloodGroup  
ORDER BY TotalDonors DESC;

Purpose: Provides an overview of blood availability trends. Helps the admin monitor shortages or surpluses of certain blood types.

* 1. List Recent Blood Donation Requests

SELECT FullName, BloodGroupRequired, UnitsNeeded, HospitalName, DateOfRequirement  
FROM Recipient  
WHERE DateOfRequirement >= CURDATE()  
ORDER BY DateOfRequirement ASC;

Purpose: Retrieves future-dated or same-day requests. Can be used to notify nearby donors.

* 1. Search Users by Role (e.g., Admins or General Users)

SELECT Email, Role, RegistrationDate  
FROM Users  
WHERE Role = 'Admin';

Purpose: Helps system administrators list and manage accounts based on user roles.

* 1. View Feedback for a Particular Donor

SELECT f.Rating, f.Comments, u.Email AS GivenBy  
FROM Feedback f  
JOIN Users u ON f.User\_ID = u.ID  
WHERE f.Donor\_ID = 1;

Purpose: Displays all feedback related to a specific donor. Useful for quality monitoring and donor recognition.

* 1. Retrieve Unread Notifications for a User

SELECT Message, SentTime  
FROM Notification  
WHERE Receiver\_ID = 2 AND ReadStatus = 'Unread'  
ORDER BY SentTime DESC;

Purpose: To show unread messages to a specific user in their dashboard.

* 1. Update Donor Availability After a Successful Donation

UPDATE Donor  
SET AvailabilityStatus = 'Unavailable', LastDonationDate = CURDATE()  
WHERE ID = 1;

Purpose: Automatically mark a donor as unavailable once they have completed a donation.

* 1. Blood Stock Summary from Inventory (Optional)

SELECT BloodGroup, SUM(UnitsAvailable) AS TotalUnits, Location  
FROM BloodInventory  
GROUP BY BloodGroup, Location;

Purpose: Provides admin with an inventory snapshot for each location.

These queries form the backbone of the application logic and are vital for user interaction, operational efficiency, and data analysis. They can be further optimized for performance using indexing and stored procedures in larger deployments.

**Functional Dependencies**

In a relational database, functional dependencies describe the relationship between attributes (columns) in a table. A functional dependency (FD) means that one attribute or a group of attributes can uniquely determine another attribute. This is critical for normalizing the database and reducing redundancy.

For the Blood Donation Bank system, functional dependencies can be defined as follows:

**1. Donor Table Functional Dependencies**

* **Donor\_ID → Donor\_Name, Donor\_Blood\_Group, Donor\_City, Donor\_Phone**
* **Explanation**: The **Donor\_ID** uniquely identifies the donor. Hence, knowing the Donor\_ID will give us the donor's name, blood group, city, and phone number.
* **Donor\_Name, Donor\_Blood\_Group → Donor\_Phone**
* **Explanation**: A combination of the donor’s **name** and **blood group** should be sufficient to determine their **phone number**. This assumes that two donors with the same name and blood group do not exist in the system.

**2. Request Table Functional Dependencies**

* **Request\_ID → Requester\_ID, Donor\_ID, Request\_Status, Request\_Date**
* **Explanation**: The **Request\_ID** uniquely determines all other attributes in the request table, including the requester’s ID, the donor’s ID, the request status (accepted, pending, etc.), and the request date.
* **Requester\_ID, Donor\_ID → Request\_Status**
* **Explanation**: A combination of **Requester\_ID** and **Donor\_ID** can be used to determine the **Request\_Status**. This ensures that for each request made by a requester to a donor, there is a status associated.

**3. Donor Registration Table Functional Dependencies**

* **Donor\_ID → Donor\_Name, Donor\_Blood\_Group, Donor\_City, Donor\_Phone**
* **Explanation**: As with the Donor table, **Donor\_ID** uniquely determines the donor's details such as name, blood group, city, and phone number.

**4. Donation Table Functional Dependencies**

* **Donation\_ID → Donor\_ID, Donation\_Date, Donation\_Quantity**
* **Explanation**: The **Donation\_ID** uniquely determines the donor’s ID, the date of donation, and the quantity of blood donated.
* **Donor\_ID, Donation\_Date → Donation\_Quantity**
* **Explanation**: A combination of **Donor\_ID** and **Donation\_Date** determines the quantity of blood donated on that particular date.
* **Examples of Functional Dependencies:**

| * **Donor\_ID** | * **Donor\_Name** | * **Donor\_Blood\_Group** | * **Donor\_City** | * **Donor\_Phone** |
| --- | --- | --- | --- | --- |
| * 1 | * John Doe | * O+ | * New York | * 1234567890 |
| * 2 | * Jane Smith | * A+ | * Los Angeles | * 0987654321 |

* **Donor\_ID → Donor\_Name, Donor\_Blood\_Group, Donor\_City, Donor\_Phone**
* This means if you know **Donor\_ID = 1**, you can determine that:
* **Donor\_Name = John Doe**
* **Donor\_Blood\_Group = O+**
* **Donor\_City = New York**
* **Donor\_Phone = 1234567890**
* **Summary:**
* The functional dependencies in this system are primarily focused on identifying the relationships between entities such as donors, requests, and donations. Each table uses a unique key (like **Donor\_ID**, **Request\_ID**, or **Donation\_ID**) to determine other attributes within the same table. Ensuring these dependencies are followed will help maintain data integrity and normalize the database structure.

**Front-End Design**

In the **Blood Donation Bank** web application, the key screens represent critical functionalities that users will frequently interact with. These screens are designed to offer smooth navigation and deliver the required services efficiently. Below are the key screens, including their primary functions, design elements, and user experience considerations.

**1. Landing Page (Home Screen)**

* **Primary Function**: The landing page serves as the first point of interaction with users. It introduces the Blood Donation Bank and provides clear calls to action (CTAs) for registering as a donor and searching for available blood donors.
* **Design Features**:
  + **Hero Section**: A large banner or image with an impactful message, such as "Your Contribution Can Save Lives!"
  + **Primary CTAs**:
    - "Become a Donor" — directs users to the registration page.
    - "Find a Donor" — redirects to the search for blood donors page.
  + **Quick Overview**: Highlights the mission of the organization and why blood donation is important.

**Example UI Design:**

* A hero image with a donation scene, CTA buttons like "Become a Donor," and "Find a Donor," and a brief description of the site's mission.

**2. Donor Registration Page**

* **Primary Function**: Allows users to sign up as blood donors by entering their personal information, such as blood group, contact details, and location.
* **Design Features**:
  + **Input Fields**: Fields to enter full name, blood group, phone number, city, and optional medical history (if applicable).
  + **Blood Group Selection**: Dropdown menu to select the user's blood group (O+, A+, B+, etc.).
  + **Submit Button**: Once users fill out the form, a clear CTA ("Register") submits the data to the database.

**Example UI Design:**

* A clean form layout with labels and fields for user input. Simple navigation with a prominent register button.

**3. Search Donors Page**

* **Primary Function**: Allows users (mainly those in need of blood) to search for nearby donors based on criteria such as blood type and city.
* **Design Features**:
  + **Search Filters**: Users can filter by blood type, city, and possibly the status of the donor (e.g., available).
  + **Results Section**: Displays a list or grid of available donors matching the search criteria, including their name, blood group, and contact information.
  + **Pagination**: If there are many donors, pagination allows users to navigate through search results more easily.

**Example UI Design:**

* A search form with dropdowns for blood group and text fields for city. A list of matching donors appears below, showing their name, blood group, and contact info.

**4. Request Notifications Page**

* **Primary Function**: Displays the status of blood donation requests, including whether a request is pending, accepted, or rejected.
* **Design Features**:
  + **Status Indicators**: Different colors or icons to indicate the status of a request (e.g., green for accepted, yellow for pending, red for rejected).
  + **Details Section**: Each request can include information about the donor, the requested blood group, and the location of the recipient.
  + **Actionable Items**: Options to either confirm the request or cancel the request (depending on the status).

**Example UI Design:**

* A card or table layout showing each request's details, with clear color-coded status indicators.

**5. Contact Us Page**

* **Primary Function**: Allows users to contact the Blood Donation Bank for inquiries, feedback, or assistance.
* **Design Features**:
  + **Form Fields**: A contact form with fields for the user’s name, email, subject, and message.
  + **Submit Button**: Users can submit the contact form to get in touch with the team.
  + **Support Information**: Display the phone number, email, and physical address of the Blood Donation Bank.

**Example UI Design:**

* Simple contact form layout with fields for name, email, subject, and message. Support info (email, phone number) listed clearly beneath the form.

**6. User Profile Page**

* **Primary Function**: Allows registered donors and users to view and edit their profile, including personal information, blood donation history, and request status.
* **Design Features**:
  + **Personal Information**: Displays name, blood group, and contact information, with the option to update them.
  + **Donation History**: Lists previous blood donations, including dates and locations.
  + **Edit Button**: Users can click an "Edit" button to update their details or change their donation preferences.

**Example UI Design:**

* A clean layout showing personal information at the top and donation history in a table or list format below.

**7. Login/Signup Page**

* **Primary Function**: Provides users with the ability to log in or create a new account to access the system’s full features.
* **Design Features**:
  + **Login Form**: Allows users to log in with their username/email and password.
  + **Signup Option**: Option for new users to create an account.
  + **Forgot Password Link**: For users who may have forgotten their login credentials.

**Example UI Design:**

* A simple login form with fields for email/username and password, along with links for account creation or password recovery.

**8. Dashboard (Admin/Donor)**

* **Primary Function**: The dashboard provides an overview of all activities, requests, and donor interactions. It may have different layouts for admin and donor users.
* **Design Features**:
  + **For Donors**: A list of their recent activities, donations made, and pending requests.
  + **For Admins**: Admins will have a more detailed view with metrics such as total donations, donor statistics, and request statuses.
  + **Action Buttons**: Quick access to important sections like "Manage Requests" for admins and "Donate Now" for donors.

**Example UI Design:**

* A card layout or statistics overview for donors, while admins see tables with actionable data like request status, donor registration, and report generation.

**9. Thank You/Confirmation Page**

* **Primary Function**: After a successful blood donation registration or request, this page thanks users for their participation and confirms their actions.
* **Design Features**:
  + **Confirmation Message**: A clear message confirming the success of the user’s action, like "Thank you for registering as a blood donor!"
  + **Next Steps**: Suggestions for what users can do next, such as sharing their donation on social media or setting a reminder for the next donation.

**Example UI Design:**

* A simple page with a "Thank You" message, success icons, and follow-up options like a CTA to the donation page or social media sharing options.

**10. Footer Design (Global)**

* **Primary Function**: The footer appears on every page and provides quick links to important sections like Privacy Policy, Terms of Service, and Social Media.
* **Design Features**:
  + **Quick Links**: Links to pages like Privacy Policy, Terms of Service, and Contact Us.
  + **Social Media Icons**: Direct links to the Blood Donation Bank’s social media pages (Facebook, Twitter, Instagram).

**Example UI Design:**

* A footer bar with concise text and icons linking to social media and important policy pages.

**Back-End Logic**

1. **User Registration Logic**:

* Validate user input (e.g., blood group, contact info).
* Check for existing donor data.
* Insert new donor details into the database if no duplicates.

1. **Blood Donor Search Logic**:

* Accept blood group and city as search criteria.
* Query database to find matching donors.
* Display matching donor details to the user.

1. **Donor Notification System**:

* Notify donors about the status of their requests (accepted or rejected).
* Update database with notification details.
* Notifications are stored and displayed to users.

1. **Admin Dashboard Logic**:

* Fetch statistics (e.g., total donors, total donations, pending requests).
* Display a summary of donation requests by blood group and city.
* Allow the admin to manage and approve/reject donation requests.

1. **User Authentication (Login/Signup)**:

* Validate user credentials during login.
* Start a session for authenticated users.
* Protect sensitive user pages with session checks.

1. **Data Security**:

* Use password hashing for secure storage.
* Prevent SQL injection with prepared statements.
* Ensure HTTPS for encrypted communication.

**Tools & Technologies Used**

1. **Programming Languages**:

* **PHP**: For back-end logic and database interaction.
* **HTML/CSS**: For front-end design and layout.
* **JavaScript**: For adding interactivity and client-side validations.

1. **Database**:

* **MySQL**: Relational database management system used to store user, donor, and request information.

1. **Web Server**:

* **Apache**: The web server used for serving PHP files and hosting the website.
* **XAMPP**: Local development environment that includes Apache, MySQL, and PHP.

1. **Front-End Frameworks**:

* **Bootstrap**: Used for responsive web design, making the website mobile-friendly and adaptable to different screen sizes.

1. **Back-End Framework**:

* **PHP**: Utilized for dynamic content generation, user registration, authentication, and request management.

1. **Version Control**:

* **Git**: For version control to track changes in the codebase and collaborate with other developers.

1. **Text Editor/IDE**:

* **Visual Studio Code**: A code editor used for writing and editing PHP, HTML, CSS, and JavaScript files.

1. **Tools for Testing**:

* **Postman**: Used for testing API endpoints and database queries.

**Conclusion**

The **Blood Donation Management System** is a comprehensive and user-friendly platform designed to facilitate the efficient management of blood donations. The project leverages modern web development technologies like PHP, MySQL, HTML, CSS, and JavaScript to create a seamless experience for users and donors alike.

Key highlights of the system include:

1. **User-Friendly Interface**: The website offers an intuitive and responsive interface, allowing users to easily register as donors, search for blood donors, and manage their donation requests.
2. **Efficient Data Management**: The system uses MySQL as the database to store critical data such as user profiles, donation requests, and donor information, ensuring smooth data management and retrieval.
3. **Real-Time Interaction**: The platform allows real-time updates of donation statuses and enables users to track the status of their requests and donations.
4. **Scalable Architecture**: Built with scalability in mind, the system can be easily expanded to incorporate additional features such as blood donation camps, alerts, and integrations with hospitals or blood banks.
5. **Technological Integration**: The use of technologies like PHP and MySQL ensures a robust back-end that efficiently handles requests, while front-end technologies like HTML, CSS, and JavaScript provide an engaging user experience.

References

1. **PHP Manual**. PHP.net. Retrieved from <https://www.php.net/manual/en/>
2. **MySQL Documentation**. MySQL.com. Retrieved from <https://dev.mysql.com/doc/>
3. W3Schools. "PHP Tutorial." W3Schools. Retrieved from <https://www.w3schools.com/php/>
4. W3Schools. "MySQL Tutorial." W3Schools. Retrieved from <https://www.w3schools.com/mysql/>
5. **HTML and CSS: Design and Build Websites** by Jon Duckett. Wiley, 2011.
6. **JavaScript and JQuery: Interactive Front-End Web Development** by Jon Duckett. Wiley, 2014.
7. **SQL for Web Developers** by Vikram Vaswani. Sams Publishing, 2007.
8. **Bootstrap Documentation**. Bootstrap. Retrieved from <https://getbootstrap.com/docs/>
9. **JavaScript: The Good Parts** by Douglas Crockford. O'Reilly Media, 2008.
10. **Database Design for Mere Mortals** by Michael J. Hernandez. Addison-Wesley, 2013.
11. **Blood Donation and Its Impact on Public Health** by National Institutes of Health. Retrieved from <https://www.nih.gov/>
12. **"The Importance of Blood Donation"**. American Red Cross. Retrieved from <https://www.redcross.org/>