Blood Donation Database System

Project Cover Page

Project Title: Blood Donation Database System

University: Mekelle University

College: E-iTM

School: Computing

Course Name: <u>Advanced Database</u>

Instructor:

Group Members:

- Kibrom Abebe - ugr/170206/12
- Mikiele Getachew - ugr/170207/12
- Alem Desta – ugr/178792/12
- Asmelash Hadush – fycc/170248/12
- Beti G/mskel – eitm/tur181555/16
- Hlina G/hiwet – Eitm/tur181606/16
- Angesom Zeru – UGR/172186/12

Date: [January 28,2025]

Table of Contents

1. Introduction

The **Blood Donation System** is designed to store and manage data related to blood donations, donor details, blood inventory, and staff. It aims to improve the efficiency of tracking blood donations and staff operations at healthcare facilities. This system ensures proper storage and retrieval of information while maintaining high security standards.

This document provides an overview of the **database design**, including functional and non-functional requirements, relationships between entities, the **Entity Relationship Diagram (ERD)**, relational schema, and SQL queries.

---

2. Functional Requirements

1. **Staff Management**
   - **Admin Role**: Can add, update, and delete staff members in the system.
   - **Staff Role**: Can update their personal details and view relevant information.
   - **Key Fields**: staff_id, name, role, contact, username, password.
   - **Staff Registration**: Staff can only be registered by the admin. The admin assigns a username and password for the staff to log in.
2. **Donor Management**
   - **Admin Role**: Can add new donors, update donor details, and track donation history.
   - **Key Fields**: donor_id, name, contact, blood_type, is_active.
3. **Blood Donation Management**
   - **Admin and Staff Role**: Record blood donations with details such as donor, donation date, and quantity.
   - **Key Fields**: donation_id, donor_id, staff_id, donation_date, quantity.
4. **Blood Bank Management**
   - **Admin Role**: Track and manage available blood types and quantities.
   - **Key Fields**: blood_type, quantity, expiration_date.
5. **Admin and User Authentication**
   - **Admin Role**: Full access to the system. The admin can manage all records (staff, donors, blood donations).
   - **User Role**: Limited access based on roles. For example, the Blood Accumulator can handle blood donation records, while Blood Bank Staff can manage blood inventory.
   - **Staff Authentication**: All staff can log in if they have been registered by the admin. Admin assigns usernames and passwords to staff members.

---

3. Non-Functional Requirements

Non-functional requirements define the system's operational aspects. They focus on how the system should perform, rather than the specific features it should offer.

1. **Performance Requirements**
   - **System Responsiveness**: The system should respond to user actions within 2 seconds for standard operations such as searching for blood types or updating donor records.
   - **Load Handling**: The system should be capable of handling up to 500 concurrent users without significant degradation in performance, especially during high-traffic periods such as blood donation drives.
2. **Security Requirements**
   - **Data Encryption**: All sensitive data, including staff passwords and donor details, should be encrypted both in transit (using HTTPS) and at rest (using secure encryption algorithms).
   - **Authentication**: Only authorized users (admin and registered staff) should be able to log in. The admin will assign usernames and passwords for staff members.
   - **Authorization**: Role-based access control (RBAC) should be implemented, ensuring that users can only access the features and data appropriate for their role (admin, staff).
3. **Usability Requirements**
   - **User Interface**: The system should have an intuitive and easy-to-use web interface. The navigation should be simple, with clear labels for each feature (staff management, donor management, blood donations, etc.).
   - **Mobile Responsiveness**: The interface should be responsive, working seamlessly across desktop, tablet, and mobile devices.
4. **Scalability Requirements**
   - **Database Scalability**: The database should be able to scale horizontally or vertically to accommodate growing amounts of data, especially for donor and blood donation records as the system expands.
   - **User Scalability**: The system should support scaling to a large number of concurrent users without major performance issues, including the ability to manage more staff, donors, and blood donations as the number of users grows.
5. **Availability and Reliability Requirements**
   - **System Uptime**: The system should maintain an uptime of at least 99.9% to ensure continuous availability for staff, donors, and administrators.
   - **Backup and Recovery**: Regular backups of the database should be taken to prevent data loss, and there should be a disaster recovery plan in place.
6. **Maintainability Requirements**

- **Codebase**: The codebase should follow industry standards and best practices for readability, organization, and documentation to ensure that the system can be easily maintained and updated by future developers.
- **Bug Fixes and Updates**: The system should have a clear process for tracking, prioritizing, and fixing bugs and for deploying system updates.

7. **Compliance Requirements**
    - **Data Protection**: The system should comply with relevant data protection laws (e.g., GDPR or local regulations) when handling personal data (staff and donor details).
    - **Accessibility**: The system should adhere to basic accessibility standards, ensuring that it can be used by people with disabilities (e.g., screen reader compatibility)

---

4. Database Design and Structure

The database design for the Blood Donation System follows a **relational model** and focuses on efficient storage, retrieval, and management of data related to blood donations, donor and staff management, blood inventory, and donation history.

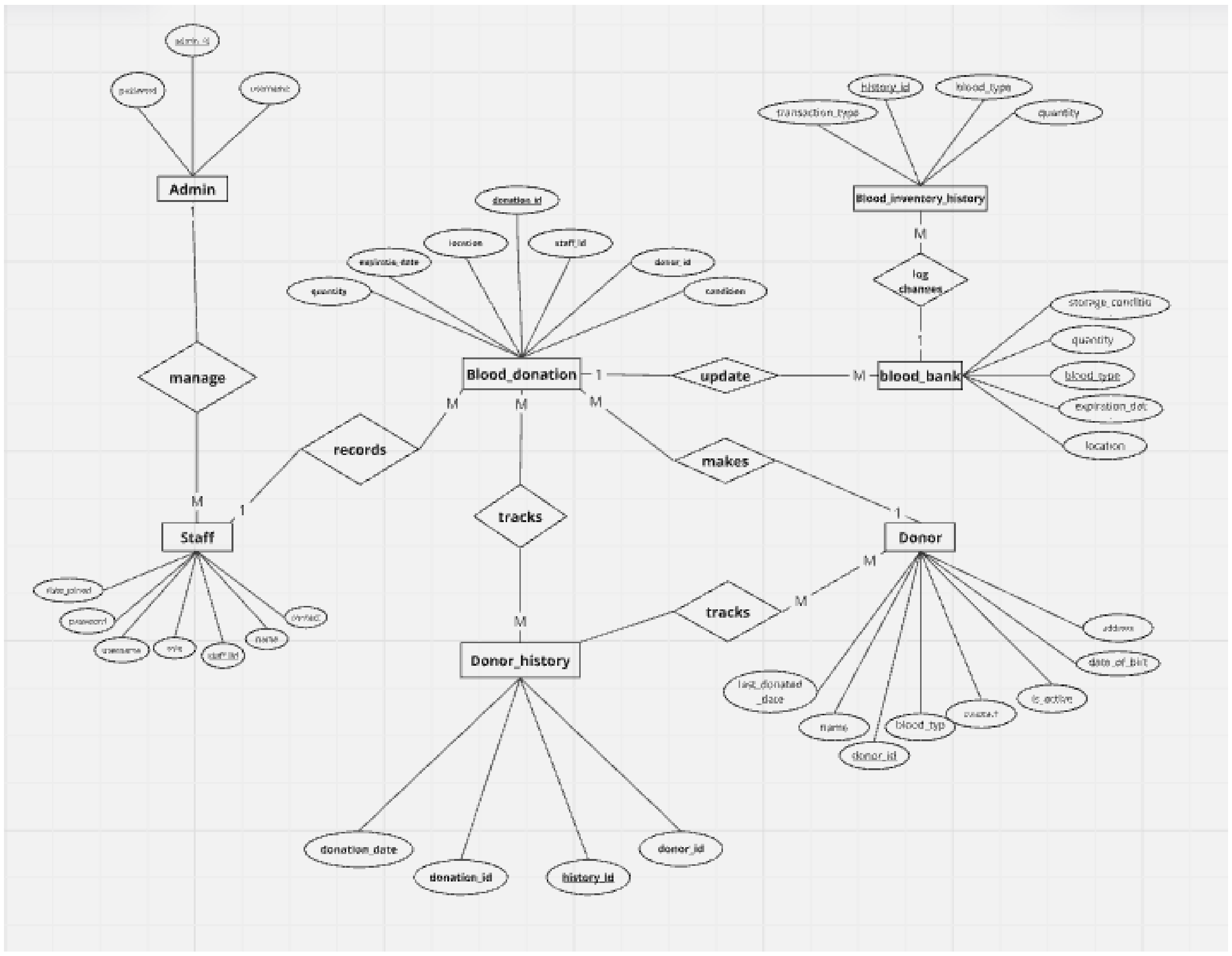
The design also ensures **data integrity**, **ease of use**, and **traceability**, supporting both **administrators** and **staff** roles within the system.

4.1 Key Concepts in Database Design

1. **Entities**: These represent real-world objects within the system (e.g., Staff, Donor, Blood Donation, Blood Bank).
2. **Relationships**: Represent the connections between entities. For example, a donor can donate blood, and each blood donation is managed by a staff member.
3. **Attributes**: Each entity has specific attributes (e.g., staff has name, role, and contact).
4. **Primary Keys**: Each table has a primary key to uniquely identify records.
5. **Foreign Keys**: Establish relationships between tables by referring to the primary key of another table.
6. **Normalization**: The database is normalized to ensure efficient data storage and avoid redundancy.

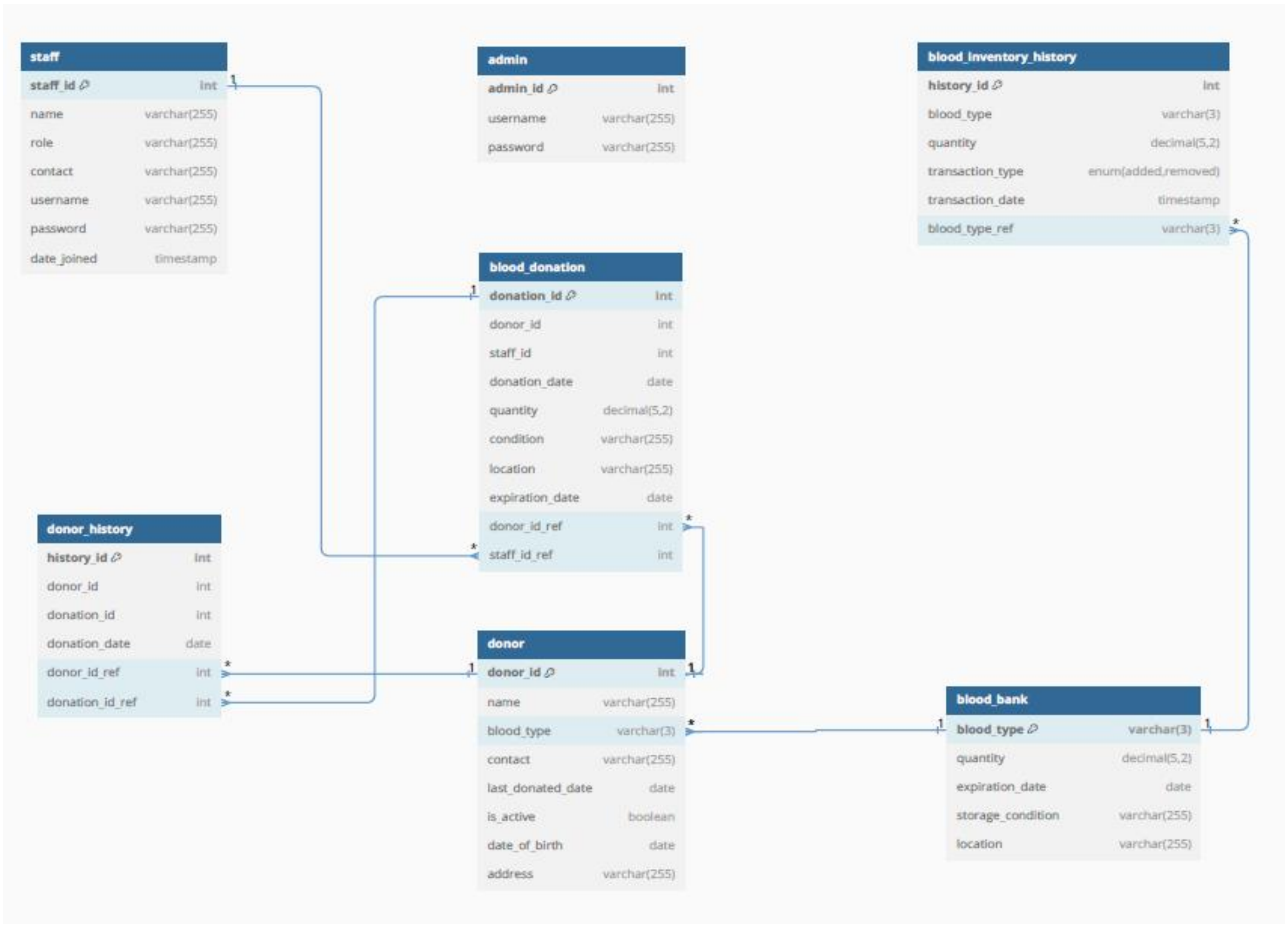
4.2 Entity Relationship Diagram (ERD)

The **ER Diagram** visually represents the relationships between the different entities in the database. For this system, we have key entities like **Staff**, **Donor**, **Blood Donation**, **Blood Bank**, and **Admin**. Additional entities like **Blood Inventory History** and **Donor History** track important historical data.



Key Entities and Their Relationships:

- **Staff**:
  - Admins can manage staff members (add, update, delete).
  - Staff members log blood donations and track blood inventory.
- **Donor**:
  - Donors give blood, and their details are recorded.
  - Donors are linked to their donation history.
- **Blood Donation**:
  - A donation involves a donor, a staff member, and a blood type.
  - Donation details are recorded for further analysis.
- **Blood Bank**:
  - Stores blood types and the quantities available.
  - Blood donations directly impact the available quantities in the bank.
- **Admin**:
  - Full access to the system, including managing staff, donors, and records.



4.3 Database Tables and Fields

The database consists of multiple tables, each serving a distinct purpose. Below is a breakdown of the tables, key fields, and their relationships.

1. Staff Table

- **Fields**:
  - staff_id: INT, Primary Key
  - name: VARCHAR
  - role: VARCHAR
  - contact: VARCHAR
  - username: VARCHAR, Unique
  - password: VARCHAR (hashed)
- **Description**: Stores staff member details. Only **Admin** can add, update, or remove staff.

---

2. Donor Table

- **Fields**:
  - donor_id: INT, Primary Key
  - name: VARCHAR
  - contact: VARCHAR
  - blood_type: VARCHAR (linked to Blood Bank)
  - is_active: BOOLEAN (Indicates if the donor is active)
  - last_donated_date: DATE
- **Description**: Records donor information. Each donor can have multiple blood donations.

---

3. Blood Donation Table

- **Fields**:
  - donation_id: INT, Primary Key
  - donor_id: INT, Foreign Key (Donor)
  - staff_id: INT, Foreign Key (Staff)
  - donation_date: DATE
  - quantity: DECIMAL (5,2)
- **Description**: Each donation is tied to a specific donor, staff member, and records the quantity of blood donated.

---

4. Blood Bank Table

- **Fields**:
  - blood_type: VARCHAR, Primary Key
  - quantity: DECIMAL (5,2) – Blood quantity available
  - expiration_date: DATE – When the blood expires
- **Description**: The Blood Bank table tracks available blood types and their quantities.

---

5. Blood Inventory History Table

- **Fields**:
  - inventory_id: INT, Primary Key
  - blood_type: VARCHAR, Foreign Key (Blood Bank)
  - quantity_changed: DECIMAL (5,2)

    - change_date: DATE
    - change_type: ENUM ('donation', 'usage', 'restock')
    - staff_id: INT, Foreign Key (Staff)
- **Description**: Keeps track of changes in the blood inventory, such as new donations, blood usage, and restocking actions.

---

6. Donor History Table

- **Fields**:
    - history_id: INT, Primary Key
    - donor_id: INT, Foreign Key (Donor)
    - donation_id: INT, Foreign Key (Blood Donation)
    - donation_date: DATE
    - quantity: DECIMAL (5,2)
    - staff_id: INT, Foreign Key (Staff)
- **Description**: Stores the donation history for each donor.

---

7. Admin Table

- **Fields**:
    - admin_id: INT, Primary Key
    - username: VARCHAR, Unique
    - password: VARCHAR (hashed)
- **Description**: Stores admin login credentials. Admins have full access to all functions in the system.

4.4 Relationships Between Tables

- **Staff to Blood Donation**: One-to-many – A staff member can record multiple blood donations.
- **Donor to Blood Donation**: One-to-many – A donor can have multiple donations.
- **Blood Donation to Blood Bank**: Many-to-one – Many donations contribute to the bSSlood bank's inventory.
- **Donor to Donor History**: One-to-many – A donor can have multiple history records for different donations.
- **Blood Inventory History to Blood Bank**: Many-to-one – Each inventory change refers to a specific blood type in the blood bank.

4.5 Database Schema and SQL Queries

```
-- Blood Donation Table
CREATE TABLE blood_donation (
  donation_id INT PRIMARY KEY AUTO_INCREMENT,
  donor_id INT,
  staff_id INT,
  donation_date DATE,
  quantity DECIMAL(5,2),
  FOREIGN KEY (donor_id) REFERENCES donor(donor_id),
  FOREIGN KEY (staff_id) REFERENCES staff(staff_id)
);

-- Blood Inventory History Table
CREATE TABLE blood_inventory_history (
  inventory_id INT PRIMARY KEY AUTO_INCREMENT,
  blood_type VARCHAR(3),
  quantity_changed DECIMAL(5,2),
  change_date DATE,
  change_type ENUM('donation', 'usage', 'restock'),
  staff_id INT,
  FOREIGN KEY (blood_type) REFERENCES blood_bank(blood_type),
  FOREIGN KEY (staff_id) REFERENCES staff(staff_id)
);

-- Donor History Table
CREATE TABLE donor_history (
  history_id INT PRIMARY KEY AUTO_INCREMENT,
  donor_id INT,
  donation_id INT,
  donation_date DATE,
  quantity DECIMAL(5,2),
  staff_id INT,
  FOREIGN KEY (donor_id) REFERENCES donor(donor_id),
  FOREIGN KEY (donation_id) REFERENCES blood_donation(donation_id),
  FOREIGN KEY (staff_id) REFERENCES staff(staff_id)
);
```

4.6 Data Integrity and Constraints

- **Foreign Keys**: Used to ensure data integrity across related tables (e.g., linking blood donations to donors and staff).

- **Enum Fields**: Used for the change_type field in the blood inventory history to ensure that only valid values are used ('donation', 'usage', 'restock').
- **Data Types**: Optimized for storing relevant information, such as DECIMAL for quantities and DATE for donation dates.

4.7 Summary of Database Structure

The database is designed to support the functionalities required for managing blood donations, donor information, and blood inventory. With the addition of **Blood Inventory History** and **Donor History**, the system can track the lifecycle of blood donations and the state of the inventory over time. This design ensures robust management, efficient querying, and traceability of all critical operations, providing both staff and admins with the necessary tools for managing blood donation activities.

---

5. User Interface and Interaction

5.1 User Interface Design Overview

The **User Interface (UI)** is a critical part of the Blood Donation System, ensuring an intuitive and user-friendly experience for both the admin, staff, and donors. The UI allows users to interact with the system seamlessly, view relevant information, and perform essential actions such as registering donations, managing staff, and tracking blood inventory.

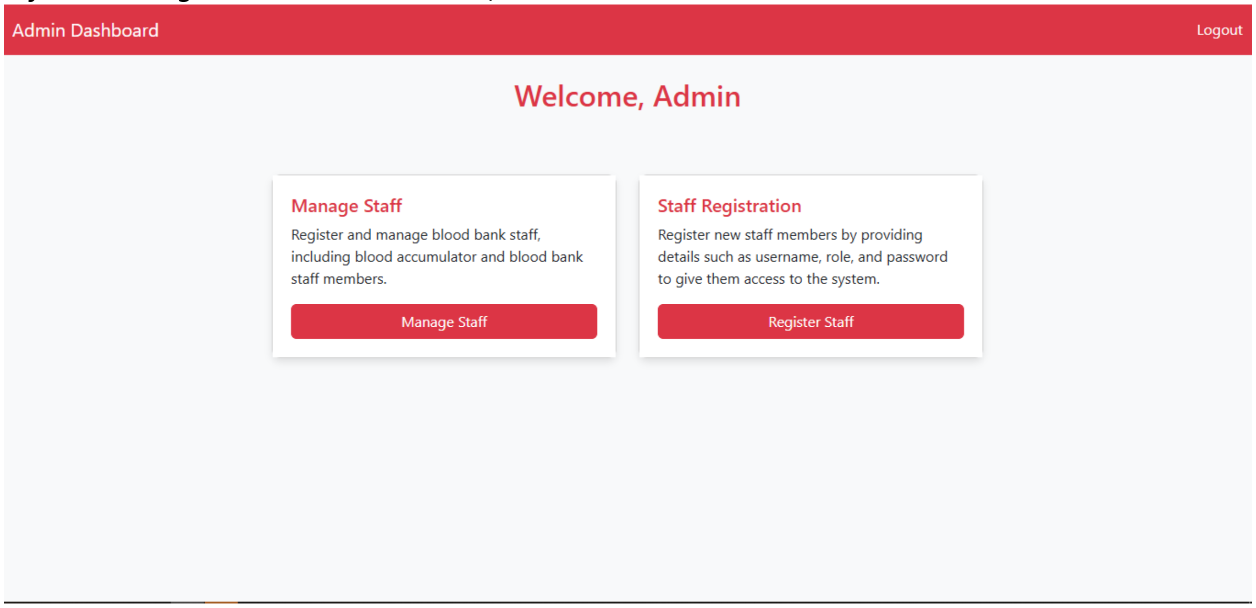
The system provides a **web-based interface** built using **HTML, CSS, JavaScript**, and **EJS** for dynamic rendering of pages. The frontend interacts with the backend (via **Express.js**) to access data from the database and display it in a user-friendly manner.

5.2 Key Features of the User Interface

1. **User Login and Authentication:**
   - **Admin and Staff:** Secure login page to allow authorized users to access the system. Admin can manage their own credentials and staff members.
   - **Donors:** If implemented, a basic donor login page for checking donation records and managing personal data.
   - This interface connects directly to the **admin** and **staff** tables for validation.

Admin Login Page

"After entering the correct credentials, admin is redirected to the admin dashboard."


Admin Dashboard

Logout

Welcome, Admin

Manage Staff

Register and manage blood bank staff, including blood accumulator and blood bank staff members.

Manage Staff

Staff Registration

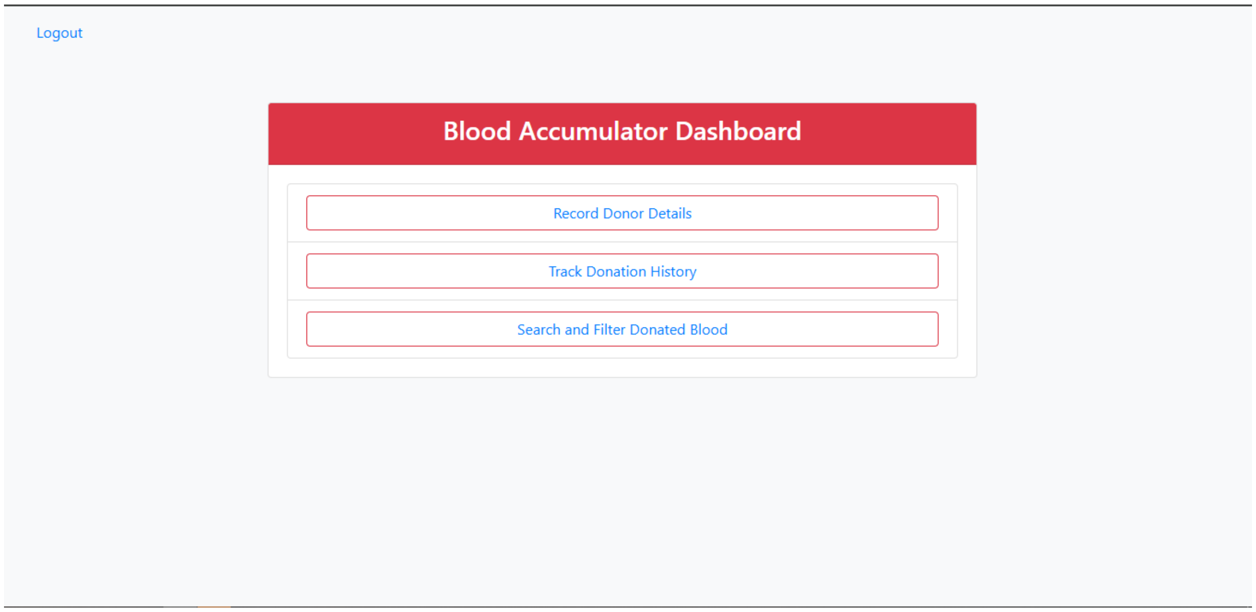
Register new staff members by providing details such as username, role, and password to give them access to the system.

Register Staff

Staff Login Page

"Staff login page where staff can log in with the credentials provided by the admin."

- If the user is blood collector direct in to this page


Logout

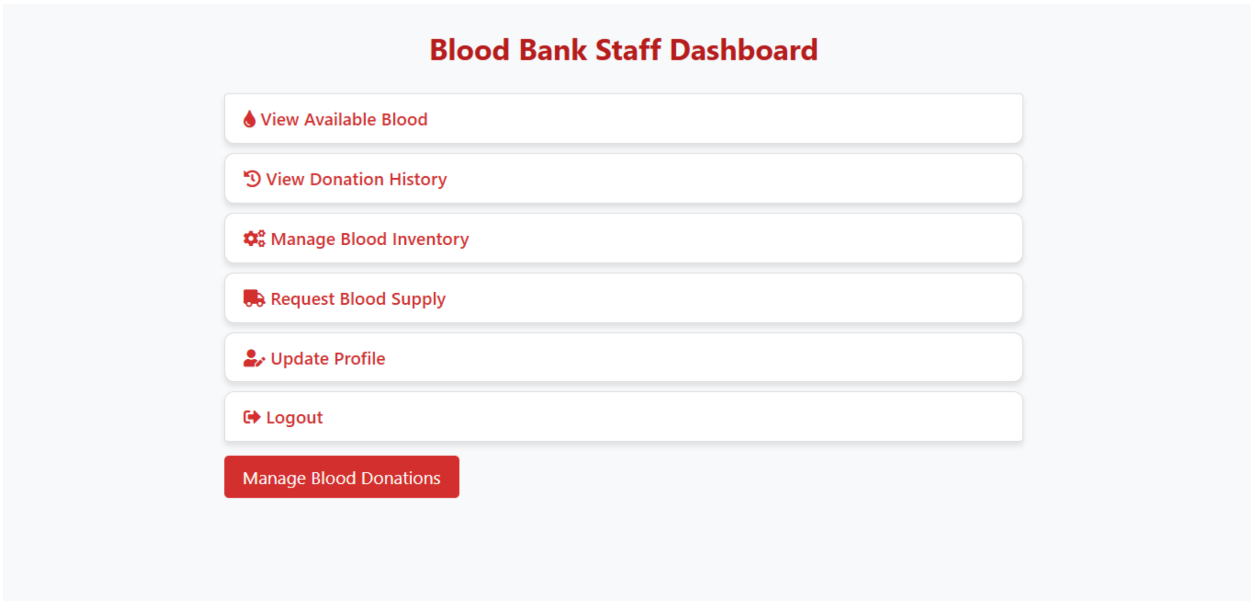
Blood Accumulator Dashboard

Record Donor Details

Track Donation History

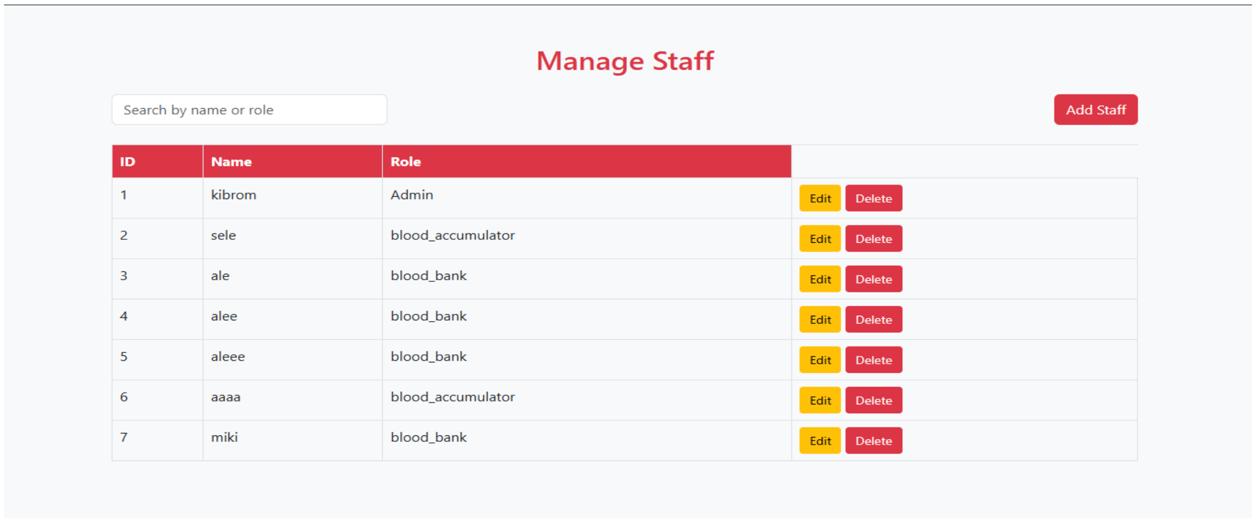
Search and Filter Donated Blood

- If the user is blood bank staff direct in to this page



2. **Admin Panel:**
    - The **Admin Panel** gives admins full control over the system, including:
        - **Managing Staff:** Admins can add, update, or delete staff members. This is linked to the **staff** table.
        - **Managing Donors:** Admins can add new donors, update existing records, and delete inactive donors.

- **Blood Inventory Management:** Admins can manage available blood types, quantities, and expiration dates. **Admin Dashboard**
  "Upon successful login, the admin is redirected to the dashboard where they can manage staff, donors, and blood inventory."
  - to manage staff



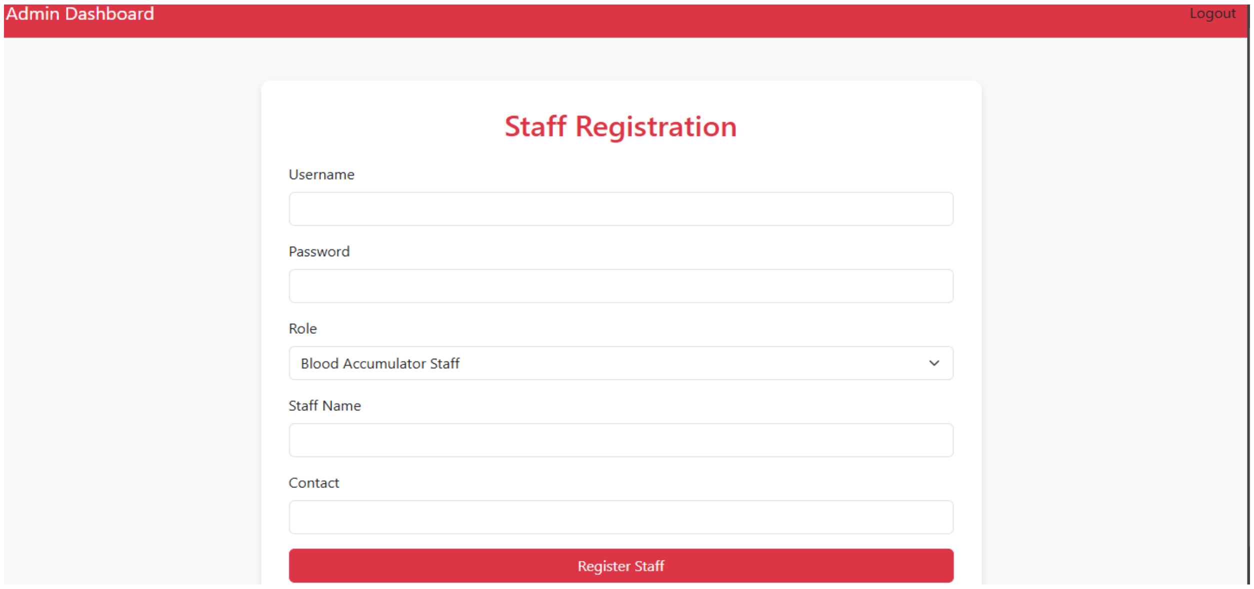
Manage Staff

Search by name or role

Add Staff

ID	Name	Role	
1	kibrom	Admin	Edit Delete
2	sele	blood_accumulator	Edit Delete
3	ale	blood_bank	Edit Delete
4	alee	blood_bank	Edit Delete
5	aleee	blood_bank	Edit Delete
6	aaaa	blood_accumulator	Edit Delete
7	miki	blood_bank	Edit Delete

- To register staff



Admin Dashboard Logout

Staff Registration

Username	
Password	
Role	Blood Accumulator Staff
Staff Name	
Contact	

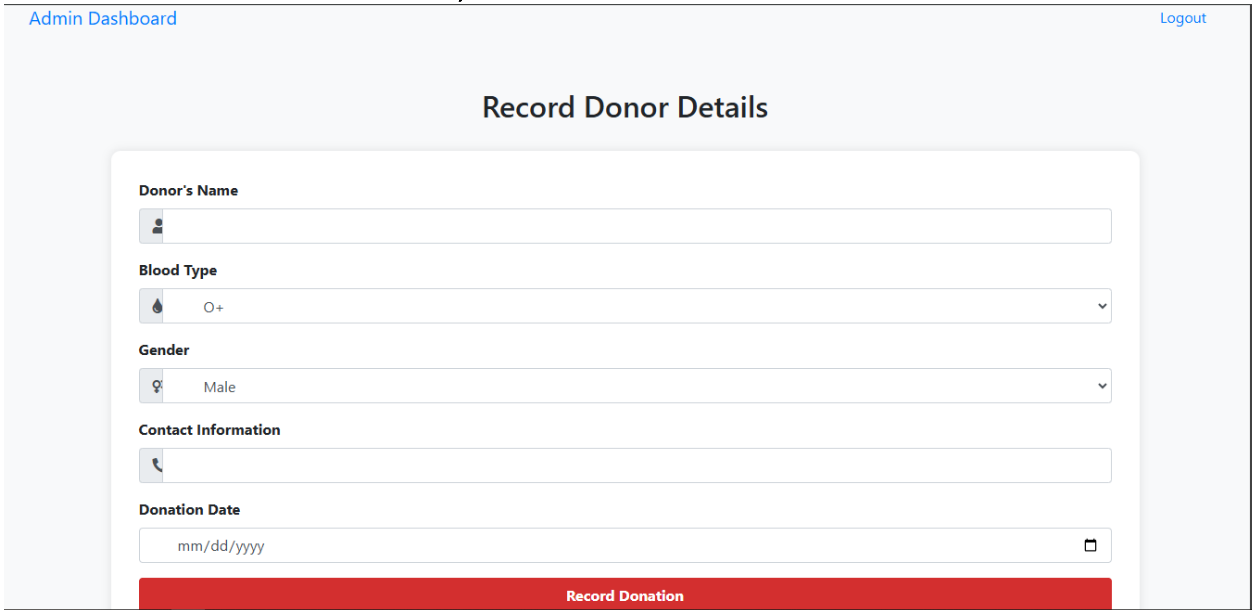
Register Staff

3. **Staff Interface:**
   - **Donation Registration:** Staff members can record new donations by inputting donor details, donation quantity, and donation date. This action updates the **blood_donation** and **donor_history** tables.
   - **View Donations:** Staff can view a list of recorded donations with associated details. Data is fetched from the **blood_donation** table.

Staff Dashboard

"After logging in, staff are redirected to their dashboard where they can record blood

donations and view donation history."


Admin Dashboard

Logout

Record Donor Details

Donor's Name

Blood Type

O+

Gender

Male

Contact Information

Donation Date

mm/dd/yyyy

Record Donation

5.3 UI and Database Interaction Flow

The UI interacts with the **database layer** through the backend API built using **Express.js**. The following interactions are typical:

1. **Login Flow:**
    - User enters credentials (username and password).
    - The backend queries the **admin** or **staff** tables (depending on the role) to verify the credentials.
    - On successful login, the user is redirected to the appropriate dashboard based on their role.

   Admin Login Flow:
   "Admin logs in and is redirected to the admin dashboard."

2. **Managing Donors:**
    - The admin adds or updates donor information via the frontend interface.
    - On submission, a request is sent to the backend, which then inserts or updates the **donor** table in the database.
    - The updated data is reflected in the UI after a successful operation.

   Donor Management Page
   "The admin can add, update, or delete donors from the system."

3. **Blood Donation Recording:**
    - Staff records a blood donation by filling out the necessary form fields.
    - The backend receives the data and inserts a new record into the **blood_donation** table.
    - The blood inventory is updated, and the **blood_inventory_history** table is modified to reflect any changes in blood stock levels.

    Blood Donation Form
    "Staff fills out the blood donation form and records donations in the system."

4. **Generating Reports:**
    - Admin or staff requests to generate reports, such as the list of donations or available blood types.
    - The backend queries the relevant tables, aggregates data, and returns the report to be displayed in the UI.

    Generated Reports Page
    "Reports generated from the system are displayed here."

5.4 User Authentication and Authorization

Even though the system doesn’t use full permission control with roles like "admin", "staff", and "donor", there are still basic levels of access provided:

1. **Admin Authentication:**
    - Admins have full access to the system, including managing staff, donors, blood donations, and inventory.
    - Admin login credentials are stored in the **admin** table and verified at login.
2. **Staff Authentication:**
    - Staff members can only access donation management, donation recording, and view blood bank inventory.
    - Staff login credentials are stored in the **staff** table, and they can only perform actions assigned to their role.
3. **Donor Authentication:**
    - Donors can optionally log in to view their donation history and manage personal details (if such functionality is implemented).
    - Donor login credentials can be stored in the **donor** table (if applicable).

5.5 Summary of User Interface and Interaction

The **UI** of the Blood Donation System is designed to be intuitive, responsive, and secure. It ensures that users—whether admins, staff, or donors—can easily interact with the system while accessing the data stored in the database. The backend serves as the core component, ensuring seamless interaction with the database and providing the necessary functionality for CRUD operations.

---

6. System Requirements

This section outlines the **hardware and software requirements** needed to **run** and **support** your Blood Donation System. It includes the **server-side**, **client-side**, and **database** requirements to ensure smooth operation and efficiency of the system. Here's a breakdown:

---

1. Hardware Requirements

The system can run on **standard hardware** for both development and deployment. Here are the **minimum hardware requirements** for a smooth user experience:

- **Server (for web hosting and database hosting)**:
  - **CPU**: Dual-core processor or better (Intel i5 or equivalent).
  - **RAM**: 4GB or more.
  - **Storage**: 20GB of free disk space (this can grow based on the number of records stored).
  - **Network**: Reliable internet connection (for cloud-based deployment or accessing from remote locations).
- **Client (for end-users accessing the system)**:
  - **CPU**: Any modern processor (Intel or AMD).
  - **RAM**: 2GB or more.
  - **Storage**: Minimal storage required (few MB for storing temporary files and browser cache).
  - **Internet Connection**: Required for accessing the system online.

---

2. Software Requirements

The software needed is divided into the **backend**, **frontend**, and **database** components. These components enable the system to run, store data, and serve pages to users.

- **Backend** (Server-side application):

    - **Operating System**: Windows, Linux, or macOS (for local development or cloud-based servers).
    - **Web Server**: Apache or Nginx (for hosting the application).
    - **Node.js**: Required to run the server-side code (Express.js).
    - **Backend Framework**: Express.js (for handling routes, middleware, etc.).
    - **Database**: MySQL or MariaDB (for relational data storage).
    - **Authentication**: Use of JWT (JSON Web Tokens) for basic login functionality.
- **Frontend** (Client-side application):
    - **Browser**: Any modern web browser (Google Chrome, Mozilla Firefox, Microsoft Edge).
    - **Frontend Framework**: Bootstrap 4/5 (for responsive design and UI components).
    - **JavaScript**: Vanilla JavaScript (or can be enhanced with frameworks like Vue.js, React.js, etc. if needed).
    - **CSS/HTML**: For styling and structuring web pages.
    - **FontAwesome**: For icons used in the user interface.

---

3. Database Requirements

- **Database Engine**: MySQL or MariaDB (for storing data securely and efficiently).
- **Database Client**: MySQL Workbench or phpMyAdmin (for easy management of the database during development and testing).
- **Storage**: Database storage needs will grow as more donors, staff, and donations are recorded. Typically, a few GB of disk space should be enough for a small to medium-sized system.

---

4. Other Requirements

- **Backup System**: Regular backups of the database to ensure data safety.
- **Security**: Basic security measures such as **SSL certificates** for encrypted connections, and hashed passwords in the database (using libraries like bcrypt).
- **Deployment**: For deployment, a cloud platform like **AWS**, **Heroku**, or **DigitalOcean** can be used to host the server and database.

---

5. Optional/Advanced Requirements

These are additional features you might want to include based on the scale or complexity of your system.

- **Role-based Access Control**: If you decide to implement roles later (admin, staff, donor), you can manage permissions at a more granular level.
- **API Integration**: For future scalability, APIs can be built to integrate with other health or donor management systems.
- **Logging and Monitoring**: Implement logging mechanisms for error tracking and monitoring the health of the system using services like **Loggly** or **Datadog**.
- **Performance Optimization**: For large systems, consider using caching systems like **Redis** for frequently accessed data, or load balancing for scaling.

7. Conclusion

This document provides a comprehensive overview of the **database design** for the Blood Donation System. The system is designed to efficiently manage blood donation records, donor details, blood bank inventory, and staff information. It ensures data consistency and security, meeting the needs of healthcare institutions in tracking blood donations.

---

9. References

- W3Schools SQL Tutorial
- MySQL Documentation
