**Database Design Document for Blood Bank Management System (BBMS)**

**1. Introduction**

This document provides a detailed database design for the Blood Bank Management System (BBMS). The database ensures efficient handling of data related to donors, patients, blood inventory, and requests while maintaining data integrity and security.

**2. Functional Requirements**

The database system must support the following functionalities:

1. **Donor Management**:
   * Add, update, and delete donor records.
   * Store donor personal and medical details for tracking eligibility.
2. **Patient Management**:
   * Maintain patient profiles.
   * Track patient blood requests and fulfillment status.
3. **Blood Inventory Management**:
   * Record and update blood stock with details such as type, quantity, and expiry date.
   * Track blood donations linked to specific donors.
4. **Blood Request Management**:
   * Allow patients to request specific blood types and quantities.
   * Manage the status of requests (e.g., Pending, Fulfilled).
5. **Admin Privileges**:
   * Allow admins to oversee and manage donors, patients, inventory, and requests.
   * Generate reports and statistics related to blood stock and donation trends.
6. **Data Security and Integrity**:
   * Secure sensitive data such as admin passwords.
   * Ensure relationships between data are maintained using foreign key constraints.

**3. Database Schema Overview**

The database schema consists of the following tables to meet the functional requirements:

**3.1 Tables Definition**

**3.1.1 Admin Table**

* **Purpose**: Stores admin details for system management.
* **Attributes and Description**:

| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| A\_ID | Integer | Primary Key | Unique identifier for each admin. |
| A\_name | Text | Not Null | Name of the admin. |
| A\_password | Text | Not Null | Password for admin login. (Hashed) |
|  |  |  |  |

**3.1.2 Donor Table**

* **Purpose**: Stores details about donors.
* **Attributes and Description**:

| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| D\_ID | Integer | Primary Key | Unique identifier for each donor. |
| D\_name | Text | Not Null | Name of the donor. |
| D\_address | Text | Not Null | Address of the donor. |
| D\_contact | Text | Not Null | Contact number of the donor. |
| D\_diseases | Text | Nullable | Pre-existing diseases, if any. |
| D\_dob | Date | Not Null | Date of birth of the donor. |
| D\_hospital | Text | Nullable | Hospital affiliated with the donor. |

**3.1.3 Patient Table**

* **Purpose**: Stores patient details and their requirements.
* **Attributes and Description**:

| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| P\_ID | Integer | Primary Key | Unique identifier for each patient. |
| P\_name | Text | Not Null | Name of the patient. |
| P\_hospital | Text | Not Null | Hospital where the patient is treated. |
| P\_units | Integer | Not Null | Number of blood units required. |

**3.1.4 Blood Inventory Table**

* **Purpose**: Tracks blood stock availability and donation details.
* **Attributes and Description**:

| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| B\_ID | Integer | Primary Key | Unique identifier for each blood record. |
| B\_type | Text | Not Null | Blood type (e.g., A+, O-, AB+). |
| B\_quantity | Integer | Not Null | Available quantity of that blood type. |
| B\_donor\_id | Integer | Foreign Key | References D\_ID in Donor Table. |
| B\_expiry\_date | Date | Not Null | Expiry date of the blood unit. |

**3.1.5 Blood Request Table**

* **Purpose**: Handles blood requests from patients.
* **Attributes and Description**:

| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| R\_ID | Integer | Primary Key | Unique identifier for each blood request. |
| R\_patient\_id | Integer | Foreign Key | References P\_ID in Patient Table. |
| R\_blood\_type | Text | Not Null | Type of blood requested (e.g., A+, O-). |
| R\_quantity | Integer | Not Null | Quantity of blood requested. |
| R\_status | Text | Not Null | Status of the request (e.g., Pending). |

**3.2 Relationships Between Tables**

* **Admin Table**: Links to the **Donor Table** and **Patient Table** for management purposes.
* **Donor Table**: Links to the **Blood Inventory Table** through B\_donor\_id for tracking donations.
* **Patient Table**: Links to the **Blood Request Table** through R\_patient\_id for managing blood requests.

**4. Data Integrity and Security**

**Data Integrity**

* **Primary Keys**: Ensure unique identification of records in each table.
* **Foreign Keys**: Maintain relationships between tables.
* **Not Null Constraints**: Ensure mandatory fields (e.g., names, quantities) are always populated.

**Security Measures**

* Passwords in the A\_password column are hashed for Admin accounts.
* Role-based access control is enforced to restrict data access based on user roles.

**5. Entity Relationship Diagram (ERD)**

A textual ERD representation:

[Admin] --- manages ---> [Donor] ---> contributes ---> [Blood Inventory]

|

+-- makes ----> [Blood Request] ---> [Patient]

**6. Implementation Considerations**

1. **Database System**: MySQL is recommended for its reliability and performance.
2. **Backup Strategy**: Implement regular backups to prevent data loss.
3. **Indexing**: Create indexes on frequently queried fields such as B\_type in the Blood Inventory Table.

**Database Implementation Timeline**

|  |  |  |
| --- | --- | --- |
| **Phase** | **Duration** | **Activities** |
| **Requirements Gathering** | 1 Week | Identify all data requirements and relationships. Collaborate with stakeholders to finalize functional requirements and schema design. |
| **Schema Design** | 1 Week | Design the database schema (tables, relationships, and constraints). Create ER diagrams and data flow diagrams for clarity. |
| **Database Creation** | 1 Week | Implement the schema in the chosen DBMS (e.g., MySQL). Create tables, set up primary/foreign keys, and constraints. |
| **Test Data Insertion** | 2 Days | Populate tables with sample data to validate the schema and test relationships. |
| **Query Testing** | 2 Days | Write and execute CRUD queries for donors, patients, blood inventory, and requests to ensure the database supports all functional requirements. |
| **Optimization** | 1 Week | Add indexes to optimize frequently queried fields, review foreign key relationships, and refine constraints for better performance. |
| **Integration Testing** | 1 Week | Test the database integration with application modules (e.g., Admin, Donor, Patient) and APIs. Resolve any issues related to data flow. |
| **Deployment** | 2 Days | Deploy the database on the production server. Ensure backups are configured, and access permissions are set based on user roles. |
| **Ongoing Maintenance** | Continuous | Monitor database performance, address issues, and update the schema as new requirements emerge. |

**7. Conclusion**

This document outlines the design of the database for the Blood Bank Management System, ensuring support for all functional requirements. The schema is optimized for scalability, security, and maintainability, making it a robust solution for managing donors, patients, and blood inventory efficiently.