



Mini project report on
Disaster Data Management System

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering
UE23CS351A – DBMS Project

Submitted by:

SHARATH GOWDA GR	PES2UG24CS823
NANDAN D	PES2UG23CS363

under the guidance of

Prof. Shilpa S
Assistant Professor
PES University

AUG - DEC 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India



PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

CERTIFICATE

This is to certify that the mini project entitled

BLOOD & ORGONS DONATION DATABASE

is a bonafide work carried out by

SHRATH GOWDA GR

PES2UG24CS823

NANDAN D

PES2UG23CS363

In partial fulfilment for the completion of fifth semester DBMS Project (UE20CSS301) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period AUG. 2022 – DEC. 2022. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The project has been approved as it satisfies the 5th semester academic requirements in respect of project work.

Signature

Prof. Shilpa S

Assistant Professor

DECLARATION

We hereby declare that the DBMS Project entitled **Disasters Data Management System** has been carried out by us under the guidance of **Prof. Nivedita Kasturi, Assistant Professor** and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester AUG – DEC 2023.

SHARATH GOWDA GR PES2UG24CS823 <SHARTHGR>

NANDAN D PES2UG23CS823 <NANDAN D>

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. Nivedita Kasturi, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE21CS351 - DBMS Project.

I take this opportunity to thank Dr. Sandesh B J, C, Professor, ChairPerson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this DBMS Project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

The Blood and Organ Donation Management System is a comprehensive database management project that addresses critical healthcare challenges through advanced MySQL implementation. This system demonstrates core DBMS concepts including relational database design, normalization, stored procedures, user-defined functions, triggers, and transaction management. The project features an 11-table normalized schema managing donors, patients, blood banks, organ banks, and donation records with complete referential integrity. Advanced database features include automated blood compatibility scoring functions, donor risk assessment algorithms, patient priority calculations, and real-time inventory management through database triggers. The system implements ACID properties, ensures data consistency through constraints, and provides comprehensive audit trails. This project showcases practical application of database management principles in healthcare domain, demonstrating proficiency in SQL programming, database design, query optimization, and business logic implementation at the database layer.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
1.	INTRODUCTION	11
2.	PROBLEM DEFINITION	12
3.	ER MODEL	13
4.	ER TO RELATIONAL MAPPING	14
5.	DDL STATEMENTS	15
6.	DML STATEMENTS	20
7.	QUERIES (SIMPLE QUERY AND UPDATE AND DELETE OPERATION, CORRELATED QUERY AND NESTED QUERY)	23
8.	STORED PROCEDURE, FUNCTIONS AND TRIGGERS	30
9.	FRONT END DEVELOPMENT	40

INTRODUCTION

Database Management Systems form the backbone of modern healthcare operations, particularly in critical areas like blood and organ donation where data accuracy, real-time processing, and system reliability can directly impact human lives. This project implements a comprehensive DBMS solution for managing the complex workflows, relationships, and business rules inherent in donation management systems.

DBMS Concepts Demonstrated

This project serves as a practical implementation of fundamental and advanced database management concepts:

Relational Database Design: The system employs a fully normalized relational schema with 11 interconnected tables, demonstrating proper entity-relationship modeling, primary and foreign key relationships, and referential integrity constraints.

Advanced SQL Programming: Implementation includes complex stored procedures for business logic, user-defined functions for automated calculations, and database triggers for real-time data management, showcasing advanced SQL programming capabilities.

Transaction Management: The system ensures ACID properties (Atomicity, Consistency, Isolation, Durability) through proper transaction handling, particularly critical in donation operations where data integrity is paramount.

Database Security and Integrity: Implementation of comprehensive constraints, data validation rules, and audit logging mechanisms ensures data security and maintains system integrity.

Healthcare Domain Application

The healthcare domain provides an ideal context for demonstrating DBMS capabilities due to its complex data relationships, strict regulatory requirements, and need for real-time processing. Blood and organ donation management specifically involves:

Complex many-to-many relationships between donors, patients, and medical facilities

Time-sensitive operations requiring immediate data consistency

Regulatory compliance demanding comprehensive audit trails

Business rules requiring database-level implementation for reliability

System Architecture from DBMS Perspective

The database serves as the central component managing:

Data Storage: Normalized tables with optimized storage structures

Business Logic: Stored procedures and functions implementing domain rules

Data Integrity: Triggers and constraints ensuring consistency

Performance: Optimized queries and indexing strategies

Security: Access control and audit mechanisms

This project demonstrates how advanced DBMS features can be leveraged to create robust, scalable, and efficient healthcare management systems while maintaining the highest standards of data integrity and system reliability.

PROBLEM DEFINITION

Database Management Challenges in Healthcare Systems

Healthcare organizations face significant database management challenges that directly impact operational efficiency and patient care quality. This project addresses these challenges through systematic DBMS implementation.

Primary DBMS Problems

1. Data Integrity and Consistency Issues

Challenge: Healthcare data involves complex relationships where inconsistency can have life-threatening consequences

DBMS Solution: Implementation of referential integrity constraints, foreign key relationships, and database triggers to maintain automatic consistency

Project Implementation: 11-table normalized schema with comprehensive constraint definitions ensuring data accuracy

2. Complex Business Rule Implementation

Challenge: Healthcare operations involve intricate business rules that must be consistently applied across all operations

DBMS Solution: Stored procedures and user-defined functions implementing business logic at database level

Project Implementation: Blood compatibility algorithms, donor risk assessment functions, and patient priority calculations embedded in database

3. Real-time Data Processing Requirements

Challenge: Critical healthcare decisions require immediate access to current data with automatic updates

DBMS Solution: Database triggers providing real-time inventory management and automatic audit trail generation

Project Implementation: Triggers automatically updating blood/organ inventory upon donation transactions

4. Transaction Management and Concurrency Control

Challenge: Multiple users accessing and modifying critical data simultaneously without conflicts

DBMS Solution: ACID transaction properties ensuring data consistency during concurrent operations

Project Implementation: Proper transaction handling for donation processing with rollback capabilities

5. Query Performance and Optimization

Challenge: Large healthcare datasets requiring efficient data retrieval for time-critical operations

DBMS Solution: Optimized database design with proper indexing and query optimization techniques

Project Implementation: Efficient schema design with strategic indexing for frequently accessed data

6. Audit Trail and Compliance Requirements

Challenge: Healthcare regulations requiring comprehensive tracking of all data modifications

DBMS Solution: Automated audit logging through database triggers and stored procedures

Project Implementation: Complete audit trail system tracking all donation transactions and data changes

Specific Healthcare Domain Problems

1. Donor-Patient Matching Complexity

Database Challenge: Complex compatibility rules requiring consistent implementation

Solution: User-defined functions calculating compatibility scores with standardized algorithms

2. Inventory Management Accuracy

Database Challenge: Real-time inventory tracking preventing overselling or understocking

Solution: Database triggers automatically updating inventory quantities upon transactions

3. Multi-facility Data Coordination

Database Challenge: Coordinating data across multiple blood banks and organ centers

Solution: Centralized database design with facility-specific data partitioning

4. Regulatory Compliance Data Management

Database Challenge: Maintaining comprehensive records for regulatory audits

Solution: Automated audit logging and reporting through stored procedures

DBMS Project Objectives

This project demonstrates solutions to these challenges through:

Database Design Excellence

Normalized relational schema eliminating data redundancy

Comprehensive constraint implementation ensuring data integrity

Optimized table structures for performance and scalability

Advanced SQL Implementation

- Complex stored procedures for business process automation
- User-defined functions for standardized calculations
- Database triggers for real-time data management
- Performance Optimization

- Strategic indexing for query performance
- Efficient join operations across related tables
- Optimized data retrieval for critical operations
- System Reliability

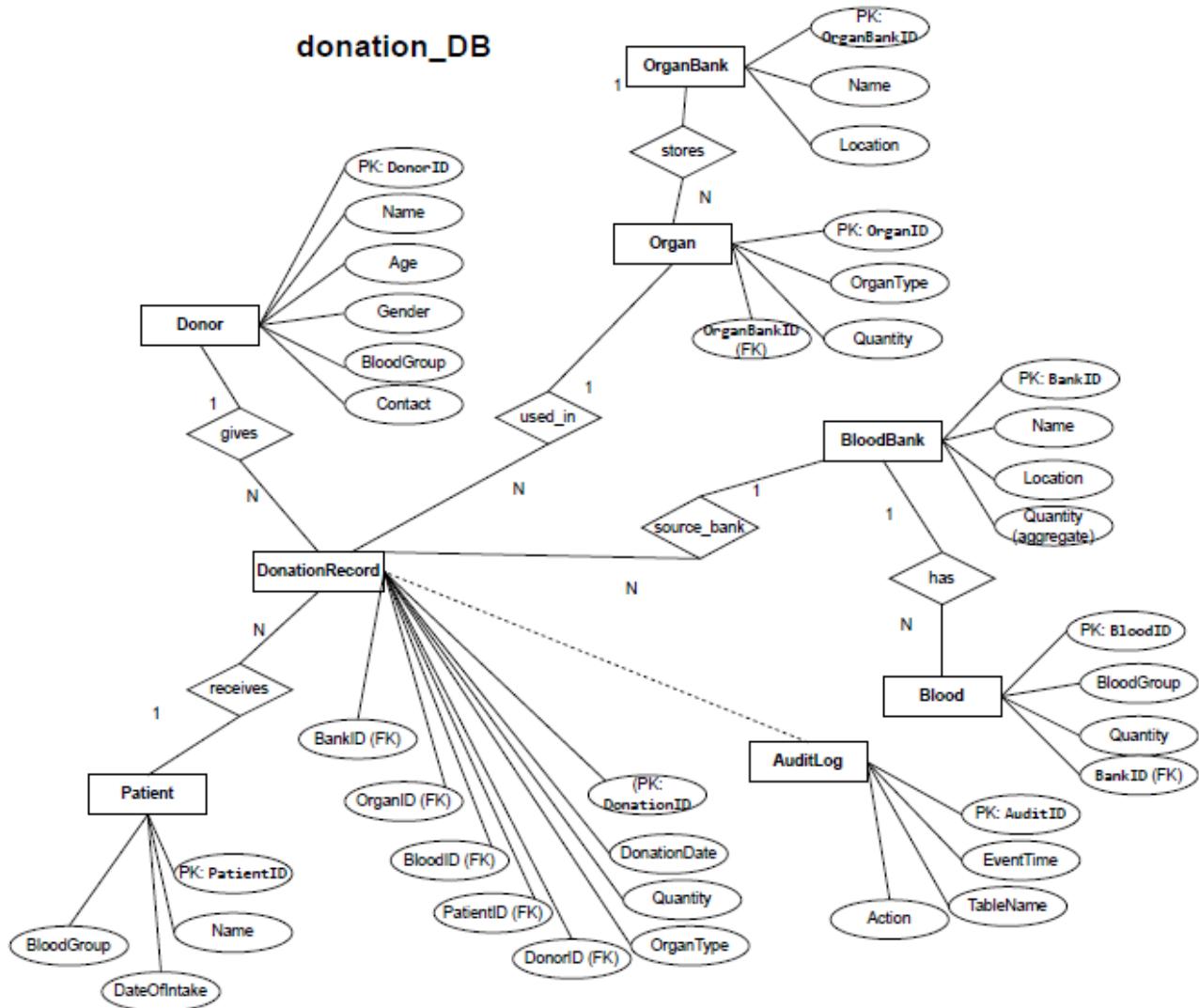
ACID transaction compliance for data consistency

Comprehensive error handling and rollback mechanisms

Robust backup and recovery procedures

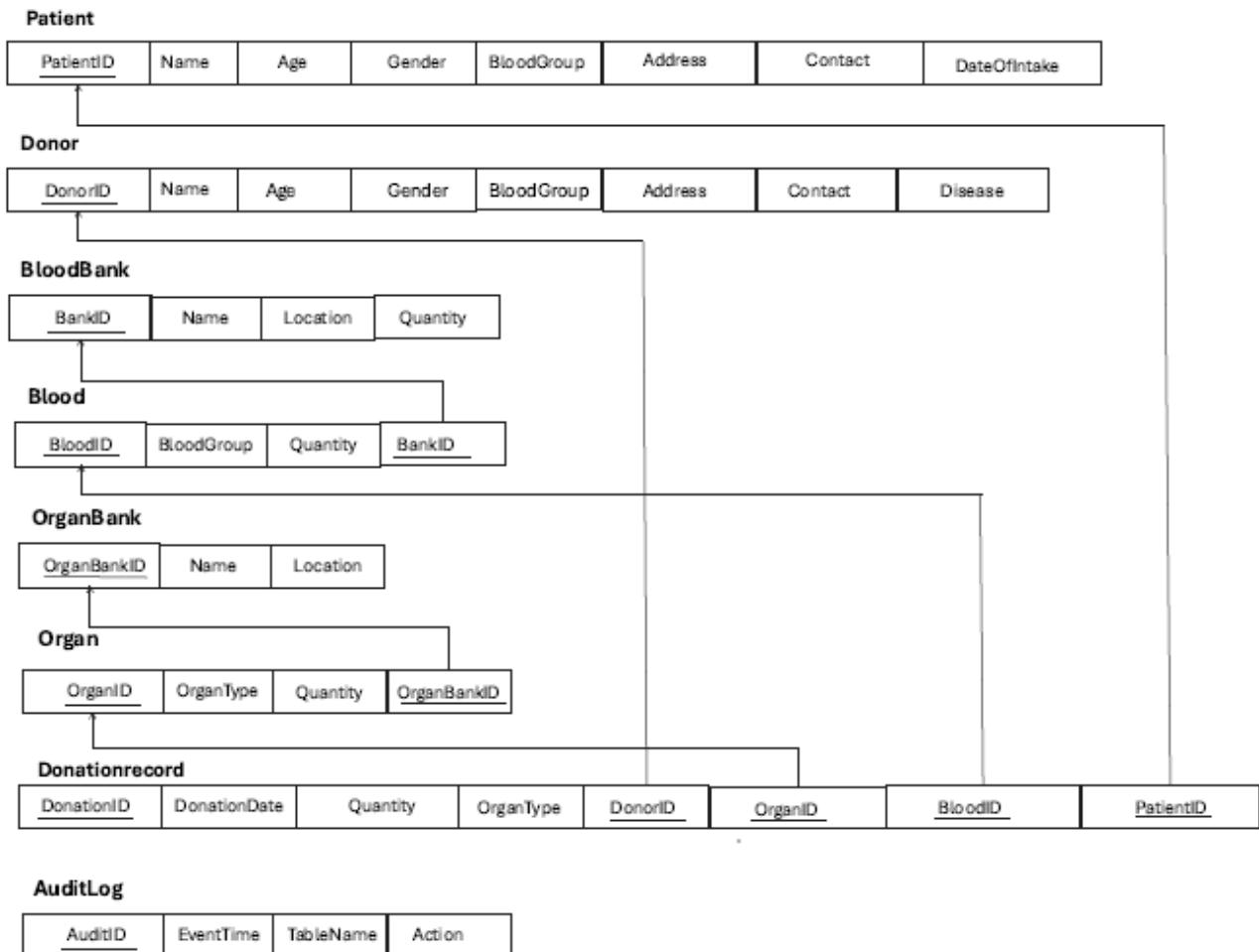
The project serves as a comprehensive demonstration of how advanced DBMS concepts can be applied to solve real-world healthcare challenges while maintaining the highest standards of data integrity, system performance, and regulatory compliance.

ER diagram



ER Scheema Model:

Blood Organ Donation Scheema Diagram



DDL:

Donor

```
CREATE TABLE Donor (
    DonorID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(120) NOT NULL,
    Age INT,
    Gender ENUM('M','F','Other'),
    BloodGroup VARCHAR(5),
    Address VARCHAR(255),
    Contact VARCHAR(20),
    Disease VARCHAR(255)
) ENGINE=InnoDB;
```

```
CREATE TABLE Patient (
    PatientID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(120) NOT NULL,
    Age INT,
    Gender ENUM('M','F','Other'),
    BloodGroup VARCHAR(5),
    Address VARCHAR(255),
    Contact VARCHAR(20),
    DateOfIntake DATE
) ENGINE=InnoDB;
```

```
CREATE TABLE BloodBank (
    BankID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(150) NOT NULL,
    Location VARCHAR(255),
    Quantity INT DEFAULT 0
) ENGINE=InnoDB;
```

-- 5) Blood (references BloodBank)

```
CREATE TABLE Blood (
    BloodID INT AUTO_INCREMENT PRIMARY KEY,
```

```
BloodGroup VARCHAR(5) NOT NULL,  
Quantity INT NOT NULL DEFAULT 0,  
BankID INT,  
CONSTRAINT fk_blood_bank FOREIGN KEY (BankID) REFERENCES BloodBank(BankID) ON  
DELETE SET NULL  
) ENGINE=InnoDB;
```

-- 6) OrganBank (parent for Organ)

```
CREATE TABLE OrganBank (  
OrganBankID INT AUTO_INCREMENT PRIMARY KEY,  
Name VARCHAR(150) NOT NULL,  
Location VARCHAR(255)  
) ENGINE=InnoDB;
```

-- 7) Organ (references OrganBank)

```
CREATE TABLE Organ (  
OrganID INT AUTO_INCREMENT PRIMARY KEY,  
OrganType VARCHAR(100) NOT NULL,  
OrganCondition VARCHAR(100),  
OrganBankID INT,  
Quantity INT DEFAULT 0,  
CONSTRAINT fk_organbank FOREIGN KEY (OrganBankID) REFERENCES OrganBank(OrganBankID)  
ON DELETE SET NULL  
) ENGINE=InnoDB;
```

-- REMOVED: Manager and Hospital tables (not used by application)

-- 8) DonationRecord (references many tables)

```
CREATE TABLE DonationRecord (  
DonationID INT AUTO_INCREMENT PRIMARY KEY,  
DonationDate DATE NOT NULL,  
Quantity INT DEFAULT 0,  
OrganType VARCHAR(100),  
DonorID INT,  
PatientID INT,  
BloodID INT,
```

```
OrganID INT,  
BankID INT,  
Notes TEXT,  
CONSTRAINT fk_donation_donor FOREIGN KEY (DonorID) REFERENCES Donor(DonorID) ON  
DELETE SET NULL,  
CONSTRAINT fk_donation_patient FOREIGN KEY (PatientID) REFERENCES Patient(PatientID) ON  
DELETE SET NULL,  
CONSTRAINT fk_donation_blood FOREIGN KEY (BloodID) REFERENCES Blood(BloodID) ON  
DELETE SET NULL,  
CONSTRAINT fk_donation_organ FOREIGN KEY (OrganID) REFERENCES Organ(OrganID) ON  
DELETE SET NULL,  
CONSTRAINT fk_donation_bank FOREIGN KEY (BankID) REFERENCES BloodBank(BankID) ON  
DELETE SET NULL  
) ENGINE=InnoDB;  
  
-- 9) AuditLog  
CREATE TABLE AuditLog (  
AuditID INT AUTO_INCREMENT PRIMARY KEY,  
EventTime DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,  
TableName VARCHAR(100),  
Action VARCHAR(20),  
KeyValue VARCHAR(255),  
UserName VARCHAR(100),  
Details TEXT  
) ENGINE=InnoDB;
```

DML:

1) INSERT Statements

Insert Donor

```
INSERT INTO Donor(Name, Age, Gender, BloodGroup, Address, Contact, Disease)
VALUES ('Rajesh Kumar', 28, 'M', 'O+', 'MG Road, Bangalore', '9876543210', 'None');
```

```
mysql> INSERT INTO Donor(Name, Age, Gender, BloodGroup, Address, Contact, Disease)
-> VALUES ('Rajesh Kumar', 28, 'M', 'O+', 'MG Road, Bangalore', '9876543210', 'None');
Query OK, 1 row affected (0.17 sec)
```

Insert Patient

```
INSERT INTO Patient(Name, Age, Gender, BloodGroup, Address, Contact, DateOfIntake)
VALUES ('Arjun Mehta', 45, 'M', 'O+', 'Richmond Road, Bangalore', '9123456780', '2025-09-01');
```

```
mysql> INSERT INTO Patient(Name, Age, Gender, BloodGroup, Address, Contact, DateOfIntake)
-> VALUES ('Arjun Mehta', 45, 'M', 'O+', 'Richmond Road, Bangalore', '9123456780', '2025-09-01');
Query OK, 1 row affected (0.04 sec)
```

Insert Blood Inventory

```
INSERT INTO Blood(BloodGroup, Quantity, BankID)
VALUES ('A+', 25, 1);
```

```
mysql> INSERT INTO Blood(BloodGroup, Quantity, BankID)
-> VALUES ('A+', 25, 1);
Query OK, 1 row affected (0.02 sec)
```

Insert Organ Inventory

```
INSERT INTO Organ(OrganType, OrganCondition, OrganBankID, Quantity)
VALUES ('Heart', 'Excellent', 1, 2);
```

```
mysql> INSERT INTO Organ(OrganType, OrganCondition, OrganBankID, Quantity)
-> VALUES ('Heart', 'Excellent', 1, 2);
Query OK, 1 row affected (0.01 sec)
```

Insert Donation Record

```
INSERT INTO DonationRecord(DonationDate, Quantity, DonorID, PatientID, BloodID, OrganID, BankID, Notes)
```

```
VALUES ('2025-10-01', 2, 1, NULL, 7, NULL, 1, 'Rajesh donated O+ blood to City Blood Bank');
```

```
mysql> INSERT INTO DonationRecord(DonationDate, Quantity, DonorID, PatientID, BloodID, OrganID, BankID, Notes)
-> VALUES ('2025-10-01', 2, 1, NULL, 7, NULL, 1, 'Rajesh donated O+ blood to City Blood Bank');
Query OK, 1 row affected (0.19 sec)
```

2) UPDATE Statements

Increase Blood Inventory During Donation

```
UPDATE Blood SET Quantity = Quantity + 3 WHERE BloodGroup = 'A+';
```

```
mysql> UPDATE Blood SET Quantity = Quantity + 3 WHERE BloodGroup = 'A+';
Query OK, 3 rows affected (0.01 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

Decrease Blood Inventory When Patient Receives Blood

```
UPDATE Blood SET Quantity = Quantity - 2 WHERE BloodGroup = 'A+';
```

```
mysql> UPDATE Blood SET Quantity = Quantity - 2 WHERE BloodGroup = 'A+';
Query OK, 3 rows affected (0.01 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

Update Organ Condition

```
UPDATE Organ SET OrganCondition = 'Good' WHERE OrganType = 'Kidney';
```

```
mysql> UPDATE Organ SET OrganCondition = 'Good' WHERE OrganType = 'Kidney';
Query OK, 1 row affected (0.01 sec)
Rows matched: 3  Changed: 1  Warnings: 0
```

3) DELETE Statements

Delete a Donor Record

```
DELETE FROM Donor WHERE DonorID = 5;
```

```
mysql> DELETE FROM Donor WHERE DonorID = 5;
Query OK, 1 row affected (0.01 sec)
```

Delete a Patient Record

```
DELETE FROM Patient WHERE PatientID = 3;
```

Delete a Donation Record

```
DELETE FROM DonationRecord WHERE DonationID = 10;
```

7. QUERIES

7.1 SIMPLE QUERY WITH GROUP BY & AGGREGATE

```
SELECT b.BankID,
       bb.Name AS BankName,
       b.BloodGroup,
       SUM(b.Quantity) AS TotalQuantity
  FROM Blood b
 JOIN BloodBank bb ON b.BankID = bb.BankID
 GROUP BY b.BankID, b.BloodGroup
 ORDER BY b.BankID, TotalQuantity DESC;
```

```
mysql> SELECT b.BankID,
      -->      bb.Name AS BankName,
      -->      b.BloodGroup,
      -->      SUM(b.Quantity) AS TotalQuantity
      --> FROM Blood b
      --> JOIN BloodBank bb ON b.BankID = bb.BankID
      --> GROUP BY b.BankID, b.BloodGroup
      --> ORDER BY b.BankID, TotalQuantity DESC;
+-----+-----+-----+-----+
| BankID | BankName          | BloodGroup | TotalQuantity |
+-----+-----+-----+-----+
| 1     | City Blood Bank    | A+         | 35           |
| 1     | City Blood Bank    | B+         | 19           |
| 1     | City Blood Bank    | O-         | 0            |
| 1     | City Blood Bank    | B-         | 0            |
| 1     | City Blood Bank    | AB-        | 0            |
| 1     | City Blood Bank    | A-         | 0            |
| 2     | Green Health Bank  | O+         | 2            |
| 2     | Green Health Bank  | O-         | 2            |
| 2     | Green Health Bank  | AB+        | 0            |
| 2     | Green Health Bank  | A-         | 0            |
| 2     | Green Health Bank  | AB-        | 0            |
+-----+-----+-----+-----+
11 rows in set (0.05 sec)
```

7.2 UPDATE OPERATION

```
UPDATE Donor
SET Disease = 'Hypertension'
WHERE Name = 'Amit Singh';

UPDATE Blood
SET Quantity = Quantity + 5
WHERE BankID = 1 AND BloodGroup = 'A+';
```

```
mysql> UPDATE Donor
-> SET Disease = 'Hypertension'
-> WHERE Name = 'Amit Singh';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0

mysql> UPDATE Blood
-> SET Quantity = Quantity + 5
-> WHERE BankID = 1 AND BloodGroup = 'A+';
Query OK, 3 rows affected (0.01 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

7.3 DELETE OPERATION

DELETE FROM AuditLog

```
WHERE EventTime < DATE_SUB(CURDATE(), INTERVAL 1 YEAR);
```

DELETE FROM DonationRecord

```
WHERE DonationDate < '2025-09-01';
```

```
mysql> DELETE FROM AuditLog
-> WHERE EventTime < DATE_SUB(CURDATE(), INTERVAL 1 YEAR);
Query OK, 0 rows affected (0.01 sec)

mysql> DELETE FROM DonationRecord
-> WHERE DonationDate < '2025-09-01';
Query OK, 1 row affected (0.01 sec)
```

7.4 CORRELATED QUERY

```
SELECT p.PatientID, p.Name, p.BloodGroup
FROM Patient p
WHERE EXISTS (
    SELECT 1
    FROM DonationRecord dr
    JOIN BloodBank bb ON dr.BankID = bb.BankID
    WHERE dr.PatientID = p.PatientID
    AND bb.Location LIKE '%MG Road%'
);
```

```

mysql> SELECT p.PatientID, p.Name, p.BloodGroup
-> FROM Patient p
-> WHERE EXISTS (
->     SELECT 1
->     FROM DonationRecord dr
->     JOIN BloodBank bb ON dr.BankID = bb.BankID
->     WHERE dr.PatientID = p.PatientID
->         AND bb.Location LIKE '%MG Road%'
-> );
+-----+-----+-----+
| PatientID | Name      | BloodGroup |
+-----+-----+-----+
|      13    | Rohan Sharma | A+        |
+-----+-----+-----+
1 row in set (0.00 sec)

```

7.5 NESTED QUERY

```

SELECT bb.BankID, bb.Name
FROM BloodBank bb
WHERE bb.BankID IN (
    SELECT BankID
    FROM Blood
    GROUP BY BankID
    HAVING SUM(Quantity) > (
        SELECT AVG(total_qty) FROM (
            SELECT SUM(Quantity) AS total_qty
            FROM Blood
            GROUP BY BankID
        ) AS bank_totals
    )
);

```

```

mysql> SELECT bb.BankID, bb.Name
-> FROM BloodBank bb
-> WHERE bb.BankID IN (
->     SELECT BankID
->     FROM Blood
->     GROUP BY BankID
->     HAVING SUM(Quantity) > (
->         SELECT AVG(total_qty) FROM (
->             SELECT SUM(Quantity) AS total_qty
->             FROM Blood
->             GROUP BY BankID
->         ) AS bank_totals
->     )
-> );
+-----+
| BankID | Name          |
+-----+
|      1 | City Blood Bank |
+-----+
1 row in set (0.01 sec)

```

8. STORED PROCEDURES, FUNCTIONS AND TRIGGERS

8.1 STORED PROCEDURES OR FUNCTIONS

1) Procedure: sp_add_donor — add a donor (simple wrapper)

```

mysql> DELIMITER //
mysql> CREATE PROCEDURE sp_add_donor(
->     IN p_name VARCHAR(100),
->     IN p_age INT,
->     IN p_gender CHAR(1),
->     IN p_bloodgroup VARCHAR(5),
->     IN p_address VARCHAR(255),
->     IN p_contact VARCHAR(30),
->     IN p_disease VARCHAR(255)
-> )
-> BEGIN
->     INSERT INTO Donor (Name, Age, Gender, BloodGroup, Address, Contact, Disease)
->     VALUES (p_name, p_age, p_gender, p_bloodgroup, p_address, p_contact, p_disease);
->
->     INSERT INTO AuditLog (EventTime, TableName, Action, Details)
->     VALUES (NOW(), 'Donor', 'INSERT',
->             CONCAT('Inserted donor: ', p_name, ' BloodGroup=', p_bloodgroup));
-> END//"
Query OK, 0 rows affected (0.05 sec)

mysql> DELIMITER ;
mysql> |

```

2) Procedure: sp_record_donation — record donation OR patient transfusion (uses triggers to update inventory)

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE sp_record_donation(
    -->      IN p_date DATE,
    -->      IN p_qty INT,
    -->      IN p_donor_id INT,
    -->      IN p_patient_id INT,
    -->      IN p_blood_id INT,
    -->      IN p_organ_id INT,
    -->      IN p_bank_id INT,
    -->      IN p_notes TEXT
    --> )
    --> BEGIN
    -->     INSERT INTO DonationRecord (DonationDate, Quantity, DonorID, PatientID, BloodID, OrganID, BankID, Notes)
    -->     VALUES (p_date, p_qty, p_donor_id, p_patient_id, p_blood_id, p_organ_id, p_bank_id, p_notes);
    -->
    -->     -- Audit entry for insertion (inventory changes handled by triggers)
    -->     INSERT INTO AuditLog (EventTime, TableName, Action, Details)
    -->     VALUES (NOW(), 'DonationRecord', 'INSERT',
    -->             CONCAT('Donation recorded: qty=', p_qty,
    -->                    ' donor=', IFNULL(CONCAT(p_donor_id), 'NULL'),
    -->                    ' patient=', IFNULL(CONCAT(p_patient_id), 'NULL'),
    -->                    ' bloodid=', IFNULL(CONCAT(p_blood_id), 'NULL'),
    -->                    ' organid=', IFNULL(CONCAT(p_organ_id), 'NULL')));
    --> END///
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
```

3) Function: fn_get_bank_blood_qty — returns quantity of a blood group in a bank

```
mysql> DELIMITER //
mysql> CREATE FUNCTION fn_get_bank_blood_qty(p_bank_id INT, p_blood_group VARCHAR(5))
    --> RETURNS INT DETERMINISTIC
    --> BEGIN
    -->     DECLARE v_qty INT DEFAULT 0;
    -->     SELECT IFNULL(SUM(Quantity), 0) INTO v_qty
    -->     FROM Blood
    -->     WHERE BankID = p_bank_id AND BloodGroup = p_blood_group;
    -->     RETURN v_qty;
    --> END///
Query OK, 0 rows affected (0.02 sec)
```

4) Function: fn_total_inventory_by_bank — returns total units across all blood groups for a bank

```

mysql> DELIMITER //
mysql> CREATE FUNCTION fn_total_inventory_by_bank(p_bank_id INT)
-> RETURNS INT DETERMINISTIC
-> BEGIN
->     DECLARE v_total INT DEFAULT 0;
->     SELECT IFNULL(SUM(Quantity),0) INTO v_total
->     FROM Blood
->     WHERE BankID = p_bank_id;
->     RETURN v_total;
-> END///
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
mysql> |

```

8.2 TRIGGERS

1) tr_after_insert_donation — AFTER INSERT on DonationRecord

```

DELIMITER //
CREATE TRIGGER tr_after_insert_donation
AFTER INSERT ON DonationRecord
FOR EACH ROW
BEGIN
    -- 1) Blood: donor -> increases inventory
    IF NEW.BloodID IS NOT NULL AND NEW.DonorID IS NOT NULL THEN
        UPDATE Blood
        SET Quantity = Quantity + NEW.Quantity
        WHERE BloodID = NEW.BloodID;

        INSERT INTO AuditLog(EventTime, TableName, Action, Details)
        VALUES (NOW(), 'Blood', 'INCREMENT',
            CONCAT('BloodID=', NEW.BloodID, ' increased by ', NEW.Quantity,
            ' via DonationRecordID=', NEW.DonationID, ' by Donor=', NEW.DonorID));
    END IF;

    -- 2) Blood: patient -> decreases inventory
    IF NEW.BloodID IS NOT NULL AND NEW.PatientID IS NOT NULL THEN
        UPDATE Blood
        SET Quantity = Quantity - NEW.Quantity
        WHERE BloodID = NEW.BloodID;

        INSERT INTO AuditLog(EventTime, TableName, Action, Details)
        VALUES (NOW(), 'Blood', 'DECREMENT',
            CONCAT('BloodID=', NEW.BloodID, ' decreased by ', NEW.Quantity,
            ' via DonationRecordID=', NEW.DonationID, ' for Patient=', NEW.PatientID));
    END IF;

    -- 3) Organ: donor -> increases organ inventory
    IF NEW.OrganID IS NOT NULL AND NEW.DonorID IS NOT NULL THEN
        UPDATE Organ
        SET Quantity = Quantity + NEW.Quantity
        WHERE OrganID = NEW.OrganID;

```

```

INSERT INTO AuditLog(EventTime, TableName, Action, Details)
VALUES (NOW(), 'Organ', 'INCREMENT',
        CONCAT('OrganID=', NEW.OrganID, ' increased by ', NEW.Quantity,
               ' via DonationRecordID=', NEW.DonationID, ' by Donor=', NEW.DonorID));
END IF;

-- 4) Organ: patient -> decreases organ inventory
IF NEW.OrganID IS NOT NULL AND NEW.PatientID IS NOT NULL THEN
    UPDATE Organ
    SET Quantity = Quantity - NEW.Quantity
    WHERE OrganID = NEW.OrganID;

INSERT INTO AuditLog(EventTime, TableName, Action, Details)
VALUES (NOW(), 'Organ', 'DECREMENT',
        CONCAT('OrganID=', NEW.OrganID, ' decreased by ', NEW.Quantity,
               ' via DonationRecordID=', NEW.DonationID, ' for Patient=', NEW.PatientID));
END IF;
END///
DELIMITER ;

```

2) tr_after_delete_donation — AFTER DELETE on DonationRecord (reverse inventory changes)

```

DELIMITER //
CREATE TRIGGER tr_after_delete_donation
AFTER DELETE ON DonationRecord
FOR EACH ROW
BEGIN
    -- If a donation record is removed, reverse its inventory effects (safety: ensure not negative)
    IF OLD.BloodID IS NOT NULL AND OLD.DonorID IS NOT NULL THEN
        UPDATE Blood
        SET Quantity = Quantity - OLD.Quantity
        WHERE BloodID = OLD.BloodID;

        INSERT INTO AuditLog(EventTime, TableName, Action, Details)
        VALUES (NOW(), 'Blood', 'REVERSE_INCREMENT',
                CONCAT('Deleted donation: decreased BloodID=', OLD.BloodID, ' by ', OLD.Quantity,
                       ' (reversing donor addition) DonationID=', OLD.DonationID));
    END IF;

    IF OLD.BloodID IS NOT NULL AND OLD.PatientID IS NOT NULL THEN
        UPDATE Blood
        SET Quantity = Quantity + OLD.Quantity
        WHERE BloodID = OLD.BloodID;

        INSERT INTO AuditLog(EventTime, TableName, Action, Details)
        VALUES (NOW(), 'Blood', 'REVERSE_DECREMENT',
                CONCAT('Deleted donation/transfusion: increased BloodID=', OLD.BloodID, ' by ', OLD.Quantity,
                       ' (reversing patient consumption) DonationID=', OLD.DonationID));
    END IF;

    -- Same for organs
    IF OLD.OrganID IS NOT NULL AND OLD.DonorID IS NOT NULL THEN

```

```

UPDATE Organ
SET Quantity = Quantity - OLD.Quantity
WHERE OrganID = OLD.OrganID;

INSERT INTO AuditLog(EventTime, TableName, Action, Details)
VALUES (NOW(), 'Organ', 'REVERSE_INCREMENT',
        CONCAT('Deleted donation: decreased OrganID=', OLD.OrganID, ' by ', OLD.Quantity,
               ' DonationID=', OLD.DonationID));
END IF;

IF OLD.OrganID IS NOT NULL AND OLD.PatientID IS NOT NULL THEN
    UPDATE Organ
    SET Quantity = Quantity + OLD.Quantity
    WHERE OrganID = OLD.OrganID;

    INSERT INTO AuditLog(EventTime, TableName, Action, Details)
    VALUES (NOW(), 'Organ', 'REVERSE_DECREMENT',
            CONCAT('Deleted organ transplant record: increased OrganID=', OLD.OrganID, ' by ',
                   OLD.Quantity,
                   ' DonationID=', OLD.DonationID));
    END IF;
END//  

DELIMITER ;

```

9. FRONT END DEVELOPMENT

OVERVIEW

The frontend application for the Blood and Organ Donation Management System is complete. It delivers a modern, responsive, and intuitive dashboard built on **React.js (18.2.0)** and styled using **Tailwind CSS**. The architecture is highly **modular** and built for **scalability** and **performance**, providing full management capabilities for healthcare staff.

❖ KEY ARCHITECTURE & FUNCTIONALITY

- **Technology Stack:** React.js, Tailwind CSS, Vite (bundler).
- **Core Modules (7):** Comprehensive interfaces for **Donors, Patients, Blood Inventory, Organ Inventory, Donations**, and dedicated testing components for **MySQL Functions and Procedures**.
- **Data Management:** Full **CRUD** (Create, Read, Update, Delete) functionality implemented across all core modules via a centralized **API Service Layer** (api.js).
- **User Experience:** Implemented a **Mobile-First Responsive Design** and utilizes visual indicators for status (e.g., **Critical Low** stock, **Organ Condition** color codes).
- **Performance:** Optimized using **React Hooks** (useMemo, useCallback), **Lazy Loading** for non-essential components, and robust **Vite** build configuration.

Donors:

Blood & Organ Donation Management

Donors Patients Blood Inventory Organ Inventory Donations Stored Procedures MySQL Functions

Donors & Real-Time Donations

+ Make Donation Add Donor

Name	Age	Gender	Blood Group	Contact	Address	Disease	Actions
Ashok	29	M	B+	9876543210	Koramangala	None	Donate Edit Delete
Amit Sharma	35	M	O-	9876543211	45 Lajpat Nagar, Delhi	none	Donate Edit Delete
Sneha Kapoor	28	F	A+	9123456788	78 Bandra West, Mumbai	none	Donate Edit Delete
Rohit Verma	42	M	B+	9234567891	12 Koramangala, Bangalore	hypertension	Donate Edit Delete
Karan Malhotra	39	M	A-	9456789013	89 Model Town, Chandigarh	diabetes	Donate Edit Delete
Pooja Agarwal	26	F	O+	9567890124	34 Hazratganj, Lucknow	none	Donate Edit Delete
Sanjay Rao	44	M	B-	9678901235	67 Banjara Hills, Hyderabad	asthma	Donate Edit Delete
Ritika Singh	33	F	AB-	9789012346	23 Salt Lake, Kolkata	none	Donate Edit Delete

Donors Patients Blood Inventory Organ Inventory Donations Stored Procedures MySQL Functions

Donors & Real-Time Donations

+ Make Donation Cancel

Edit Donor

Ashok	29
Male	B+
Koramangala	9876543210
None	

Update Donor

Patient:

Blood & Organ Donation Management

Donors Patients Blood Inventory Organ Inventory Donations Stored Procedures MySQL Functions

Patients

Add Patient

Name	Age	Gender	Blood Group	Contact	Address	Date of Intake	Actions
Suresh	45	M	B+	9123456780	Indiranagar	01/10/2025	Edit Delete
Rajesh Kumar	45	M	A+	9876543210	123 MG Road, Bangalore	15/10/2024	Edit Delete
Priya Sharma	32	F	B+	9123456789	456 Park Street, Mumbai	20/10/2024	Edit Delete
Mohammed Ali	28	M	O-	9234567890	789 Civil Lines, Delhi	25/10/2024	Edit Delete
Sunita Patel	55	F	AB+	9345678901	321 Gandhi Nagar, Ahmedabad	12/10/2024	Edit Delete
Arjun Singh	38	M	A-	9456789012	654 Residency Road, Pune	18/10/2024	Edit Delete
Kavya Reddy	29	F	B-	9567890123	987 Jubilee Hills, Hyderabad	22/10/2024	Edit Delete

[GitHub Link](#)

Blood & Organ Donation Management

Donors Patients Blood Inventory Organ Inventory Donations Stored Procedures MySQL Functions

Patients

Cancel

Edit Patient

Suresh	45
Male	B+
Indiranagar	9123456780
30-09-2025	City Blood Bank
<button>Update Patient</button>	

Blood Inventory:

Blood & Organ Donation Management

Donors Patients Blood Inventory Organ Inventory Donations Stored Procedures MySQL Functions

Blood Inventory (Real-Time)

Refresh Add Blood Record

Blood Group	Quantity	Bank Name	Location	Actions
B+	16 units	City Blood Bank	Bangalore	<button>Update</button>
A+	10 units	City Blood Bank	Bangalore	<button>Update</button>
O-	0 units	City Blood Bank	Bangalore	<button>Update</button>
AB+	0 units	Green Health Bank	Mysore	<button>Update</button>
A-	0 units	Green Health Bank	Mysore	<button>Update</button>
B-	0 units	City Blood Bank	Bangalore	<button>Update</button>
O+	2 units	Green Health Bank	Mysore	<button>Update</button>

Adding

Blood & Organ Donation Management

Donors Patients Blood Inventory Organ Inventory Donations Stored Procedures MySQL Functions

Blood Inventory (Real-Time)

Refresh Cancel

Add New Blood Record		
<input type="text" value="A+"/>	<input type="text" value="Quantity"/>	<input type="text" value="Select Blood Bank"/>
<button>Add Blood Record</button>		

[GitHub Link](#)

Organs Inventory:

Blood & Organ Donation Management

Donors Patients Blood Inventory **Organ Inventory** Donations Stored Procedures MySQL Functions

Organ Inventory (Real-Time)

Organ Type	Condition	Quantity	Bank Name	Location	Actions
Kidney	Good	2 available	Central Organ Bank	Bangalore	<button>Update</button>
Heart	Excellent	1 available	Central Organ Bank	Bangalore	<button>Update</button>
Liver	Good	0 available	Central Organ Bank	Bangalore	<button>Update</button>
Kidney	Good	1 available	Apollo Organ Bank	Chennai	<button>Update</button>
Kidney	Good	0 available	Apollo Organ Bank	Chennai	<button>Update</button>
Lung	Excellent	0 available	AIIMS Organ Center	Delhi	<button>Update</button>
Pancreas	Good	0 available	AIIMS Organ Center	Delhi	<button>Update</button>

Refresh Add Organ Record

Adding

Blood & Organ Donation Management

Donors Patients Blood Inventory **Organ Inventory** Donations Stored Procedures MySQL Functions

Organ Inventory (Real-Time)

Add New Organ Record

Organ Type	Good
Quantity	Select Organ Bank

Add Organ Record

Donations:

[GitHub Link](#)

Blood & Organ Donation Management

Donors Patients Blood Inventory Organ Inventory **Donations** Stored Procedures MySQL Functions

Donations & Triggers

Show Trigger Logs

Record Donation

Delete Feature: When you delete a donation record, the database triggers will automatically restore the inventory quantities. This demonstrates the AFTER DELETE trigger functionality.

Donor	Patient	Quantity	Blood Group	Organ Type	Bank	Date	Notes	Actions
Amit Sharma	General Pool	2	O-	N/A	Green Health Bank	04/11/2025	O- donation from Amit - donor giving	
Sneha Kapoor	General Pool	4	A+	N/A	City Blood Bank	04/11/2025	A+ donation from Sneha	
Manoj Kumar	General Pool	3	A+	N/A	City Blood Bank	04/11/2025	A+ donation from Manoj	
Ashok	General Pool	2	B+	N/A	City Blood Bank	04/11/2025	Additional B+ from Ashok	
Rohit Verma	General Pool	1	N/A	Heart	N/A	04/11/2025	Heart donation from Rohit	
Anonymous	General Pool	1	N/A	Kidney	N/A	04/11/2025	Kidney donation from Anita	

Triggers

Blood & Organ Donation Management

Donors Patients Blood Inventory Organ Inventory **Donations** Stored Procedures MySQL Functions

Donations & Triggers

Hide Audit Logs

Record Donation

Database Trigger Audit Logs

Time	Table	Action	Details
04/11/2025, 17:12:53	Blood	UPDATE	Subtracted 2 units for patient, DonationID=20
04/11/2025, 17:12:53	DonationRecord	INSERT	Created donation record on 2025-10-15 (Patient Receiving)
04/11/2025, 17:05:03	Blood	UPDATE	Reversed blood transaction by 3 due to deletion of DonationID=1
04/11/2025, 17:05:03	DonationRecord	DELETE	Deleted donation record originally on 2024-10-25
04/11/2025, 16:57:25	Blood	UPDATE	Added 2 units from donor donation, DonationID=19
04/11/2025, 16:57:25	DonationRecord	INSERT	Created donation record on 2025-10-01 (Donor Giving)

Procedures Call:

[GitHub Link](#)

Blood & Organ Donation Management

Donors Patients Blood Inventory Organ Inventory Donations **Stored Procedures** MySQL Functions

Stored Procedures Dashboard

What are Stored Procedures?

Stored procedures are pre-compiled SQL code blocks stored in the database. They provide better performance, security, and reusability compared to regular SQL queries. Click the tabs below to see different procedures in action.

Blood Compatibility

Blood Compatibility Checker

Select Patient Blood Group:

A+

Procedure: GetBloodCompatibility('A+')
Purpose: Finds all compatible blood donors and available blood units for a patient with A+ blood type.

Donor	Blood Group	Contact	Available Units	Bank	Status
-------	-------------	---------	-----------------	------	--------

Compatibility checking

Blood Compatibility

Blood Compatibility Checker

Select Patient Blood Group:

A-

Procedure: GetBloodCompatibility('A-')
Purpose: Finds all compatible blood donors and available blood units for a patient with A- blood type.

Donor	Blood Group	Contact	Available Units	Bank	Status
Divya Nair	O-	9901234568	2 units	Green Health Bank	Compatible
Amit Sharma	O-	9876543211	2 units	Green Health Bank	Compatible

Inventory report :

Blood Compatibility **Inventory Report**

Comprehensive Inventory Report

Procedure: GetInventoryReport()
Purpose: Generates a complete inventory analysis with stock levels, alerts, and recommended actions.

Type	Item	Quantity	Bank/Location	Status	Recommended Action
BLOOD	O-	0	City Blood Bank - Bangalore	CRITICAL - OUT OF STOCK	URGENT RESTOCKING NEEDED
BLOOD	AB+	0	Green Health Bank - Mysore	CRITICAL - OUT OF STOCK	URGENT RESTOCKING NEEDED
BLOOD	A-	0	Green Health Bank - Mysore	CRITICAL - OUT OF STOCK	URGENT RESTOCKING NEEDED
BLOOD	B-	0	City Blood Bank - Bangalore	CRITICAL - OUT OF STOCK	URGENT RESTOCKING NEEDED
BLOOD	AB-	0	City Blood Bank - Bangalore	CRITICAL - OUT OF STOCK	URGENT RESTOCKING NEEDED
BLOOD	A-	0	City Blood Bank - Bangalore	CRITICAL - OUT OF STOCK	URGENT RESTOCKING NEEDED
BLOOD	B-	0	City Blood Bank - Bangalore	CRITICAL - OUT OF STOCK	URGENT RESTOCKING NEEDED
BLOOD	AB+	0	Green Health Bank - Mysore	CRITICAL - OUT OF STOCK	URGENT RESTOCKING NEEDED

Donor History :

[GitHub Link](#)

[Blood Compatibility](#)[Inventory Report](#)[Donor History](#)[Critical Patients](#)

Donor History Analysis

Select Donor:

Sneha Kapoor (A+)

Procedure: GetDonorHistory(3)**Purpose:** Retrieves complete donation history and statistics for a specific donor.1
Total Donations4
Total Units0
Days Since Last04/11/2025
First Donation

Date	Type	Quantity	Recipient	Bank	Days Ago	Notes
04/11/2025	A+ Blood	4	General Pool	City Blood Bank	0 days	A+ donation from Sneha

Critical patients:

[Blood Compatibility](#)[Inventory Report](#)[Donor History](#)[Critical Patients](#)

Critical Patients Priority List

Procedure: GetCriticalPatients()**Purpose:** Identifies patients who need urgent attention based on waiting time and blood availability.

Patient	Blood Group	Contact	Days Waiting	Priority Level	Compatible Blood Available
Ravi Nair	A+	9890123456	395 days	CRITICAL - 30+ DAYS	64 units
Vikram Gupta	O+	9678901234	392 days	CRITICAL - 30+ DAYS	4 units
Sunita Patel	AB+	9345678901	388 days	CRITICAL - 30+ DAYS	80 units
Rajesh Kumar	A+	9876543210	385 days	CRITICAL - 30+ DAYS	64 units
Arjun Singh	A-	9456789012	382 days	CRITICAL - 30+ DAYS	2 units
Priya Sharma	B+	9123456789	380 days	CRITICAL - 30+ DAYS	20 units
Kavya Reddy	B-	9567890123	378 days	CRITICAL - 30+ DAYS	2 units

MySQL Functions Dashboard

Blood & Organ Donation Management

Donors Patients Blood Inventory Organ Inventory Donations Stored Procedures MySQL Functions

⚡ MySQL Functions Dashboard

What are MySQL Functions?
MySQL Functions are reusable code blocks that return a single value and can be used in SQL queries. Unlike procedures, functions can be called within SELECT statements and provide calculated results.

Compatibility Score Donor Risk Level Patient Priority Real Donor Summary Patient Urgency Blood Bank Status Donation Trends

Blood Compatibility Score Calculator

Function: CalculateBloodCompatibilityScore(donor_blood, patient_blood)
Purpose: Returns a numerical score (0-100) indicating blood compatibility between donor and patient.

Donor Blood Group: O- Patient Blood Group: A+

Calculate Compatibility Score

Compatibility Score Donor Risk Level Patient Priority Real Donor Summary Patient Urgency Blood Bank Status Donation Trends

Blood Compatibility Score Calculator

Function: CalculateBloodCompatibilityScore(donor_blood, patient_blood)
Purpose: Returns a numerical score (0-100) indicating blood compatibility between donor and patient.

Donor Blood Group: O- Patient Blood Group: A+

Calculate Compatibility Score

Result:
95/100
Compatibility between O- donor and A+ patient

Compatibility Score Donor Risk Level Patient Priority Real Donor Summary Patient Urgency Blood Bank Status Donation Trends

Donor Risk Level Assessment

Function: GetDonorRiskLevel(age, disease_history)
Purpose: Assesses donor health risk based on age and medical history for screening purposes.

Donor Age: 25 Disease History: None

Assess Risk Level

Risk Assessment Result:
LOW RISK
Age: 25 years, Disease: none

[GitHub Link](#)

Compatibility Score Donor Risk Level **Patient Priority** Real Donor Summary Patient Urgency Blood Bank Status Donation Trends

Patient Priority Score Calculator

Function: GetPatientWaitingPriority(days_waiting, blood_group, compatible_units)
Purpose: Calculates patient priority score (1-100) based on waiting time, blood rarity, and availability.

Days Waiting: Blood Group: Compatible Units Available:

Calculate Priority Score

Priority Score:
50/100
 15 days waiting • A+ blood • 5 units available

Compatibility Score Donor Risk Level **Patient Priority** **Real Donor Summary** Patient Urgency Blood Bank Status Donation Trends

Real Donor Summary (Database-Driven)

Function: GetDonorSummaryByName(donor_name)
Purpose: Generates complete donor summary by looking up actual data in database. Just provide name!

Select Donor from Database:

Generate Real Donor Summary

Database-Generated Summary:

- 👤 DONOR PROFILE: Ashok
- ID: 1
- Blood Group: B+
- Age: 29 years
- ⭐ Recognition Level: VALUED DONOR
- 之心 Total Donations: 2 times
- 献血 Total Units Donated: 4 units
- 📅 Last Donation: Today
- 🌟 Status: ACTIVE DONOR

Compatibility Score Donor Risk Level Patient Priority Real Donor Summary **Patient Urgency** Blood Bank Status Donation Trends

Patient Urgency Analysis (Database-Driven)

Function: GetPatientUrgencyByName(patient_name)
Purpose: Calculates patient urgency score by looking up actual patient data and inventory.

Select Patient from Database:

Calculate Patient Urgency

Patient Urgency Score:
50/100
 Calculated from real database data for Suresh

Compatibility Score Donor Risk Level Patient Priority Real Donor Summary Patient Urgency Blood Bank Status Donation Trends

Blood Bank Status Report (Database-Driven)

Function: GetBloodBankStatus(bank_name)
Purpose: Generates comprehensive status report for a blood bank from database.

Blood Bank Name:
City Blood Bank

Generate Bank Status Report

Compatibility Score Donor Risk Level Patient Priority Real Donor Summary Patient Urgency Blood Bank Status **Donation Trends**

Donation Trends Analysis (Database-Driven)

Function: GetDonationTrendAnalysis()
Purpose: Analyzes donation patterns from the last 30 days using real database data.

Generate Trend Analysis

30-Day Donation Trend Analysis:

DONATION TREND ANALYSIS (Last 30 Days)
 Period: 2025-10-05 to 2025-11-04
 Total Donations: 8
 Total Units Collected: 17
 Blood Donations: 5 (62.5%)
 Organ Donations: 3 (37.5%)
 Daily Average: 0.27 donations/day
 Trend Status: LOW ACTIVITY - INCREASE CAMPAIGNS