

Blood Bank Management System

PROJECT REPORT

Submitted by

Vedika Kapoor [RA2311050010025]
Abhay Singh [RA2311050010036]

Under the Guidance of

Dr.D.Rajeswari

(Professor, Data Science And Business Systems)

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING
with specialization in (BLOCKCHAIN TECHNOLOGY)



DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR- 603 203

May 2025

BONAFIDE CERTIFICATE

Certified that the 21CSC205P Database Management System course project report titled “**Blood Bank Management System**” is the bona fide work done by **Vedika Kapoor**[RA2311050010025], **Abhay Singh** [RA2311050010036], of **II Year/IV Sem B.tech(CSE-BCT)** who carried out the mini project under my supervision.

Dr. D. Rajeswari

Professor,

Department of DSBS,

SRM Institute of Science and Technology

Kattankulathur

Dr.Kavitha V

Professor and Head

Department of DSBS,

SRM Institute of Science and Technology

Kattankulathur

TABLE OF CONTENTS

Chapter No	Chapter Name	Page No
1	Problem understanding, Identification of Entity and Relationships, Construction of DB using ER Model for the project	5
2	Design of Relational Schemas, Creation of Database Tables for the project.	13
3	Complex queries based on the concepts of constraints, sets, joins, views, Triggers and Cursors.	23
4	Analyzing the pitfalls, identifying the dependencies, and applying normalizations	31
5	Implementation of concurrency control and recovery mechanisms	37
6	Result and Discussion (Screenshots of the implementation with front end.	40
7	Conclusion	43

ABSTRACT

A **Blood Bank Management System (BBMS)** is an advanced software solution designed to streamline and optimize the management of blood donation, storage, and distribution. The system minimizes manual errors and enhances operational transparency by ensuring efficiency, accuracy, and traceability.

The BBMS facilitates **donor registration, blood inventory tracking, recipient request management, and compatibility matching**, ensuring that blood transfusions are timely and safe. It maintains **real-time records** of available blood types, monitors donor eligibility, and tracks blood unit expiration, reducing wastage and improving stock management. Automation of critical processes such as **donor screening, blood collection tracking, and expiry monitoring** enhances the efficiency of blood banks and healthcare organizations.

Furthermore, integration with **hospitals and emergency healthcare providers** ensures seamless coordination, enabling quick response times in critical situations. Advanced features such as **low-stock alerts, secure donor-recipient matching, and predictive data analytics** help in resource optimization and demand forecasting.

By implementing a **digital Blood Bank Management System**, healthcare institutions can significantly improve **efficiency, reduce wastage, enhance emergency preparedness, and save more lives** through a well-organized and automated approach to blood management.

INTRODUCTION

A Blood Bank Management System is a crucial application designed to streamline the process of blood donation, storage, and distribution. This system ensures efficient management of blood donors, recipients, and blood banks while maintaining accurate records of transactions. With increasing demand for blood in medical emergencies, hospitals, and surgeries, a well-structured system is necessary to ensure timely availability and proper utilization of blood units.

The system includes various entities such as donors, patients, blood banks, hospitals, and registration teams, each playing a significant role in the process. Donors register and provide blood, which is then analyzed and stored in blood banks. Patients in need receive blood through hospitals and registered teams, ensuring smooth coordination between different entities. Blood banks act as intermediaries, storing and managing available blood groups while maintaining updated records of inventory levels.

Key functionalities of the Blood Bank Management System include donor registration, blood collection, blood analysis, storage, and patient requests. Additionally, critical analysts ensure the quality of donated blood before storage or transfusion. The system also facilitates communication between hospitals and blood banks to manage blood supply efficiently. Managers oversee the entire process to ensure smooth operations and maintain records for future reference.

This project aims to develop a well-structured and automated blood bank management system that minimizes errors, prevents blood wastage, and enhances accessibility for those in need. By implementing an organized database system, the availability and tracking of blood units become more efficient, ultimately contributing to saving lives.

PROBLEM UNDERSTANDING

The need for an efficient Blood Bank Management System arises from the challenges associated with blood donation, storage, and distribution. In traditional systems, blood banks often struggle with improper record-keeping, leading to issues such as blood shortages, wastage, and difficulty in tracking available blood groups. Patients in urgent need may face delays due to the unavailability of real-time information on blood inventory. Additionally, the manual process increases the risk of human errors in donor registration, blood analysis, and transfusion compatibility.

Hospitals and blood banks require a centralized system to manage donor records, maintain blood stock, and streamline the process of blood allocation. Without an organized system, hospitals face difficulties in ensuring a timely supply of blood for surgeries, accidents, and medical emergencies. Furthermore, improper handling of blood storage and quality analysis can lead to serious health risks for recipients.

Another major issue is the lack of proper communication between donors, blood banks, and hospitals. In many cases, patients and their families struggle to find the required blood type due to a lack of coordination. Blood donors also face challenges in tracking their donation history and scheduling future donations.

This project aims to address these problems by developing a structured Blood Bank Management System that ensures efficient donor registration, proper storage, and quick access to blood supply information. By automating the process, the system minimizes errors, reduces wastage, and improves the availability of blood, ultimately helping to save lives.

1. Inefficiency in Traditional Systems

- Issues with manual record-keeping leading to errors.
- Difficulty in tracking available blood groups.

2. Blood Shortages & Wastage

- Lack of real-time inventory updates.
- Delays in finding compatible blood for patients.

3. Hospital Challenges

- Struggle to ensure timely blood supply for emergencies.
- Risk of human errors in transfusion compatibility.

4. Poor Communication & Coordination

- Donors, hospitals, and blood banks lack a centralized system.
- Patients and families face difficulties in finding the required blood types.

5. Need for an Automated System

- Improved donor registration and record management.
- Ensures efficient storage and quick access to blood supply.
- Reduces wastage and minimizes errors, ultimately saving lives.

ENTITIES

Entities in the System In a Blood Bank Management System, entities represent real-world objects involved in blood donation, storage, and transfusion.

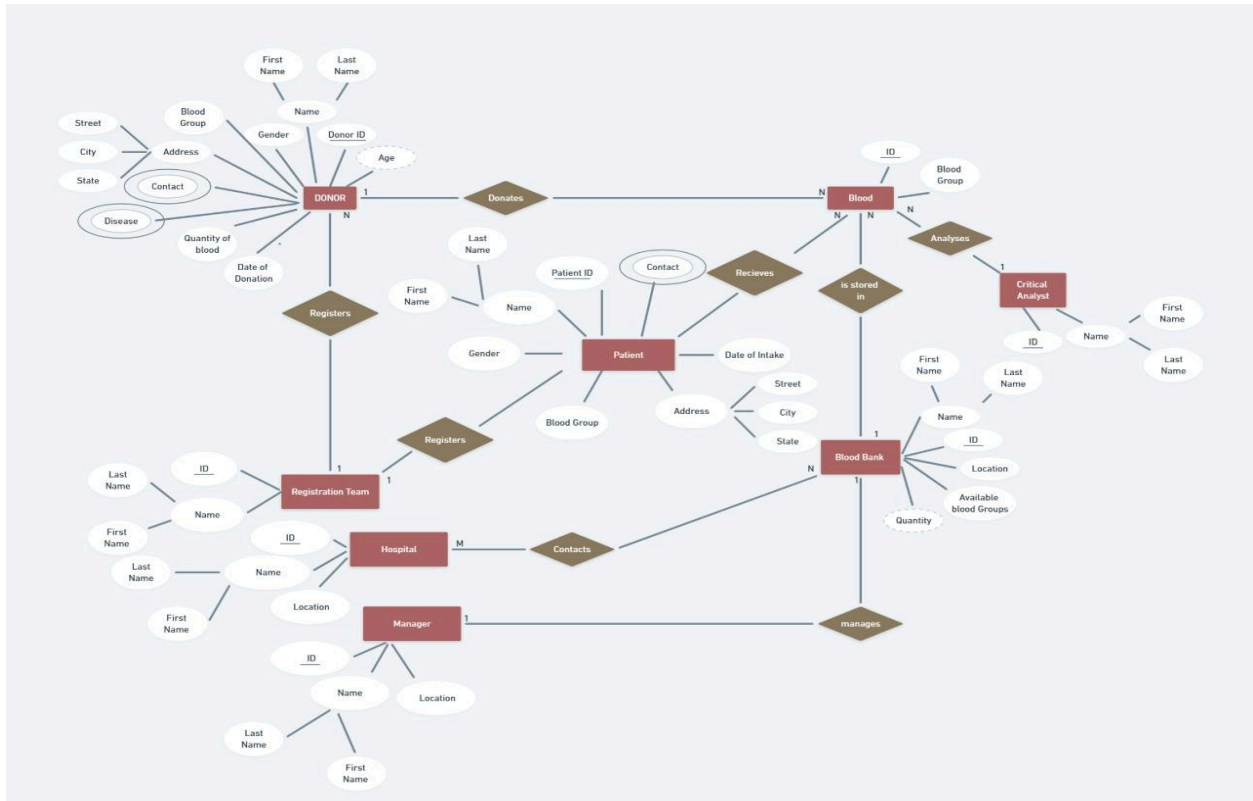
The key entities are:

Entity Name	Description
Donors	Stores details of all blood donors.
Patients	Stores details of patients requiring blood.
Blood	Stores information about donated blood.
Blood Bank	Stores details of blood banks.
Hospitals	Stores details of hospitals.
Critical Analysts	Stores details of blood analyses.
Registration Team	Stores details of the team handling registrations.
Managers	Stores details of blood bank managers.

Relationships Between Entities: Entities are connected through relationships.

Here's how they interact:

Entities Involved	Relationship Type	Description
Donor - Blood	One-to-Many (1:N)	A donor can donate multiple blood units.
Patient - Blood	One-to-Many (1:N)	A patient can receive multiple blood units.
Blood - Blood Bank	Many-to-One (N:1)	Multiple blood units are stored in a blood bank.
Blood - Critical Analyst	One-to-One (1:1)	Each blood unit is analyzed by one analyst.
Registration Team - Donor	One-to-Many (1:N)	A team registers multiple donors.
Registration Team - Patient	One-to-Many (1:N)	A team registers multiple patients.
Blood Bank - Hospital	Many-to-Many (M:N)	Hospitals contact multiple blood banks.
Blood Bank Manager	One-to-One (1:1)	Each blood bank is managed by one manager.



Entity-Relationship (ER) Diagram Overview A simple representation of relationships:

Donors (Donor ID, Name, Age, Gender, Blood Group, Address, Contact)

└─ 1:N ──> Blood (Blood ID, Blood Group)

└─ 1:N ──> Registration Team (Team ID, Name)

Patients (Patient ID, Name, Gender, Blood Group, Address, Contact)

└─ 1:N ──> Blood

└─ 1:N ──> Registration Team

Blood Bank (Bank ID, Name, Location, Available Blood Groups, Quantity)

└─ N:1 ──> Blood

└─ 1:1 ──> Manager (Manager ID, Name)

└─ M:N ──> Hospital (Hospital ID, Name, Location)

Critical Analyst (Analyst ID, Name)

└─ 1:1 ──> Blood

Primary and Foreign Keys in Relationships

Table	Primary Key (PK)	Foreign Key (FK)
Donors	Donor_ID (PK)	None
Patients	Patient_ID (PK)	None
Blood	Blood_ID (PK)	Donor_ID (FK) (from Donors), Patient_ID (FK) (from Patients)
Blood Bank	Bank_ID (PK)	None
Hospitals	Hospital_ID (PK)	None
Critical Analysts	Analyst_ID (PK)	None
Registration Team	Team_ID (PK)	None
Managers	Manager_ID (PK)	Bank_ID (FK) (from Blood Bank)

Cardinality Explanation

- **One-to-Many (1:N)**

- A donor can donate multiple blood units, but each blood unit belongs to one donor.
- A patient can receive multiple blood units, but each blood unit is assigned to one patient.
- A registration team can register multiple donors and patients.

- **One-to-One (1:1)**

- A blood bank is managed by one manager.
- Each blood unit is analyzed by one critical analyst.

- **Many-to-Many (M:N)**

- Hospitals contact multiple blood banks, and each blood bank can serve multiple hospitals.

TUPLE RELATIONAL CALCULUS (TRC)

Find all blood donations where the quantity is more than 500ml

$\{ t \mid t \in \text{Donation} \wedge t.\text{quantity} > 500 \}$

List only donor IDs and their blood groups

$\{ t.\text{donor_id}, t.\text{blood_group} \mid t \in \text{Donor} \}$

Find all donors living in 'Delhi'

$\{ t \mid t \in \text{Donor} \wedge t.\text{address} = \text{'Delhi'} \}$

Retrieve all unique blood groups available in the blood bank

$\{ t.\text{blood_group} \mid t \in \text{BloodStock} \}$

Find all blood donations with quantities between 300ml and 700ml

$\{ t \mid t \in \text{Donation} \wedge 300 \leq t.\text{quantity} \leq 700 \}$

Find recipient names along with their blood request details

$\{ t.\text{recipient_name}, r.\text{request_id}, r.\text{blood_group} \mid t \in \text{Recipient}, r \in \text{BloodRequest}, (t.\text{recipient_id} = r.\text{recipient_id}) \}$

DOMAIN RELATIONAL CALCULUS (DRC)

Find all blood donations where the quantity is more than 500ml

$\{ \langle D_ID, D_Date, Quantity \rangle \mid \exists D_ID, D_Date, Quantity (D_ID, D_Date, Quantity) \in \text{Donation} \wedge Quantity > 500 \}$

List only donor IDs and their blood groups

$\{ \langle D_ID, B_Group \rangle \mid \exists D_ID, B_Group (D_ID, B_Group) \in \text{Donor} \}$

Find all donors living in 'Delhi'

$\{ \langle D_ID, D_Name, Address, Phone \rangle \mid (D_ID, D_Name, Address, Phone) \in \text{Donor} \wedge Address = 'Delhi' \}$

Retrieve all unique blood groups available in the blood bank

$\{ \langle B_Group \rangle \mid \exists \text{Stock_ID} (\text{Stock_ID}, B_Group, Quantity) \in \text{BloodStock} \}$

Find all blood donations with quantities between 300ml and 700ml

$\{ \langle D_ID, Recipient_ID, Quantity \rangle \mid (D_ID, Recipient_ID, Quantity) \in \text{Donation} \wedge 300 \leq Quantity \leq 700 \}$

RELATIONAL ALGEBRA QUERIES

Find donors who live in Mumbai.'

σ Address = 'Mumbai' (Donor)

Find blood donations where the quantity is more than 500ml

σ Quantity > 500 (Donation)

List only blood groups and their stock quantities

π Blood_Group, Quantity (BloodStock)

List only blood request IDs and the required blood groups

π Request_ID, Blood_Group (BloodRequest)

Find all individual and organizational blood donors

π Donor_ID, Name, Phone, Address (Donor) \cup π Donor_ID, Name, Phone, Address (Organization_Donor)

Find recipients who have requested blood but have never received it

π Recipient_ID (BloodRequest) $- \pi$ Recipient_ID (BloodDonation)

Find all possible combinations of donors and recipients

Donor \times Recipient

Find recipients along with their blood request details

Recipient \bowtie Recipient.Recipient_ID = BloodRequest.Recipient_ID BloodRequest

Find donors who have donated in different cities.

Donor \bowtie Donor.Donor_ID = Donation.Donor_ID Donation \bowtie Donation.City_ID =
City.City_ID City

Find donations linked to blood stock details

Donation \bowtie Donation.Blood_Group = BloodStock.Blood_Group BloodStock

MYSQL QUERIES FOR BLOOD BANK MANAGEMENT

SYSTEM

1. CREATE

```
CREATE TABLE admin (  
    username VARCHAR(50) PRIMARY KEY,  
    password VARCHAR(50) NOT NULL  
);
```

```
CREATE TABLE blood_stock (  
    blood_group VARCHAR(5) NOT NULL,  
    quantity INT NOT NULL  
);
```

```
CREATE TABLE donors (  
    id INT PRIMARY KEY,  
    name VARCHAR(100) NOT NULL,  
    blood_group VARCHAR(5) NOT NULL,  
    contact VARCHAR(15) UNIQUE NOT NULL,  
    address VARCHAR(255) NOT NULL  
);
```

```
CREATE TABLE patients (  
    id INT PRIMARY KEY,  
    name VARCHAR(100) NOT NULL,  
    blood_group VARCHAR(5) NOT NULL,  
    contact VARCHAR(15) UNIQUE NOT NULL,  
    address VARCHAR(255) NOT NULL  
);
```

```
CREATE TABLE transactions (  
    id INT PRIMARY KEY,  
    patient_id INT NOT NULL,
```

```
blood_group VARCHAR(5) NOT NULL,  
quantity INT NOT NULL,  
FOREIGN KEY (patient_id) REFERENCES patients(id) ON DELETE  
CASCADE  
);
```

	Tables_in_blood_bank
►	admin
	availableblood
	blood_stock
	donors
	patients
	transactions

2. INSERT

INSERT INTO admin (username, password) VALUES

('admin1', 'pass123'),
('admin2', 'admin@456'),
('admin3', 'secure789'),
('admin4', 'admin!001'),
('admin5', 'pass555'),
('admin6', 'qwerty99'),
('admin7', 'letmein22'),
('admin8', 'root321'),
('admin9', 'superadmin'),
('admin10', 'adminpass');

	username	password
▶	admin1	pass123
	admin10	adminpass
	admin2	admin@456
	admin3	secure789
	admin4	admin!001
	admin5	pass555
	admin6	qwerty99
	admin7	letmein22
	admin8	root321
	admin9	superadmin

-- Insert data into blood_stock table

INSERT INTO blood_stock (blood_group, quantity) VALUES

('A+', 15),
('A-', 10),
('B+', 18),
('B-', 7),
('AB+', 12),
('AB-', 5),
('O+', 25),
('O-', 8),
('A+', 10),

('B+', 9);

	blood_group	quantity
▶	A-	10
	A+	10
	AB-	5
	AB+	12
	B-	7
	B+	18
	O-	8
	O+	25

-- Insert data into donors table

INSERT INTO donors (id, name, blood_group, contact, address) VALUES

(1, 'Ravi Kumar', 'A+', '9876543210', 'Delhi'),
(2, 'Sneha Sharma', 'B+', '8765432109', 'Mumbai'),
(3, 'Aman Verma', 'O+', '7654321098', 'Kolkata'),
(4, 'Priya Singh', 'AB+', '6543210987', 'Chennai'),
(5, 'Anil Mehta', 'O-', '9123456780', 'Bangalore'),
(6, 'Neha Jain', 'A-', '9988776655', 'Hyderabad'),
(7, 'Manoj Rathi', 'B-', '8899776655', 'Jaipur'),
(8, 'Pooja Gupta', 'AB-', '7788665544', 'Pune'),
(9, 'Karan Thakur', 'A+', '6677554433', 'Ahmedabad'),
(10, 'Deepa Rao', 'B+', '5566443322', 'Surat');

	id	name	blood_group	contact	address	age
▶	1	Ravi Kumar	A+	9876543210	Delhi	18
	2	Sneha Sharma	B+	8765432109	Mumbai	18
	3	Aman Verma	O+	7654321098	Kolkata	18
	4	Priya Singh	AB+	6543210987	Chennai	18
	5	Anil Mehta	O-	9123456780	Bangalore	18
	6	Neha Jain	A-	9988776655	Hyderabad	18
	7	Manoj Rathi	B-	8899776655	Jaipur	18
	8	Pooja Gupta	AB-	7788665544	Pune	18
	9	Karan Thakur	A+	6677554433	Ahmedabad	18
	10	Deepa Rao	B+	5566443322	Surat	18

-- Insert data into patients table

```
INSERT INTO patients (id, name, blood_group, contact, address) VALUES
(1, 'Rahul Yadav', 'A+', '9123412345', 'Delhi'),
(2, 'Meena Joshi', 'B+', '8123456789', 'Mumbai'),
(3, 'Alok Ranjan', 'O+', '7123456789', 'Lucknow'),
(4, 'Kavita Das', 'AB+', '6123456789', 'Chennai'),
(5, 'Sunil Kumar', 'O-', '5123456789', 'Bangalore'),
(6, 'Sita Rani', 'A-', '4123456789', 'Patna'),
(7, 'Vinay Rao', 'B-', '3123456789', 'Indore'),
(8, 'Gita Shah', 'AB-', '2123456789', 'Pune'),
(9, 'Dilip Mehta', 'A+', '1123456789', 'Nagpur'),
(10, 'Anita Nair', 'B+', '9999999999', 'Kochi');
```

	id	name	blood_group	contact	address
▶	1	Rahul Yadav	A+	9123412345	Delhi
	2	Meena Joshi	B+	8123456789	Mumbai
	3	Alok Ranjan	O+	7123456789	Lucknow
	4	Kavita Das	AB+	6123456789	Chennai
	5	Sunil Kumar	O-	5123456789	Bangalore
	6	Sita Rani	A-	4123456789	Patna
	7	Vinay Rao	B-	3123456789	Indore
	8	Gita Shah	AB-	2123456789	Pune
	9	Dilip Mehta	A+	1123456789	Nagpur
	10	Anita Nair	B+	9999999999	Kochi

-- Insert data into transactions table

```
INSERT INTO transactions (id, patient_id, blood_group, quantity) VALUES
(1, 1, 'A+', 2),
(2, 2, 'B+', 1),
(3, 3, 'O+', 3),
(4, 4, 'AB+', 1),
(5, 5, 'O-', 2),
(6, 6, 'A-', 1),
(7, 7, 'B-', 2),
(8, 8, 'AB-', 1),
(9, 9, 'A+', 3),
```

(10, 10, 'B+', 2);

	id	patient_id	blood_group	quantity
▶	1	1	A+	2
	2	2	B+	1
	3	3	O+	3
	4	4	AB+	1
	5	5	O-	2
	6	6	A-	1
	7	7	B-	2
	8	8	AB-	1
	9	9	A+	3
	10	10	B+	2

3. ALTER

ALTER TABLE donors ADD age INT DEFAULT 18;

	id	name	blood_group	contact	address	age
▶	1	Ravi Kumar	A+	9876543210	Delhi	18
	2	Sneha Sharma	B+	8765432109	Mumbai	18
	3	Aman Verma	O+	7654321098	Kolkata	18
	4	Priya Singh	AB+	6543210987	Chennai	18
	5	Anil Mehta	O-	9123456780	Bangalore	18
	6	Neha Jain	A-	9988776655	Hyderabad	18
	7	Manoj Rathi	B-	8899776655	Jaipur	18
	8	Pooja Gupta	AB-	7788665544	Pune	18
	9	Karan Thakur	A+	6677554433	Ahmedabad	18
	10	Deepa Rao	B+	5566443322	Surat	18
•	NULL	NULL	NULL	NULL	NULL	NULL

4. DROP

DROP TABLE admin;

5. TRUNCATE

TRUNCATE TABLE blood_stock;

6. SELECT and ALIASING

SELECT * FROM donors;

SELECT name AS donor_name, blood_group AS bg FROM donors;

	donor_name	bg
▶	Ravi Kumar	A+
	Sneha Sharma	B+
	Aman Verma	O+
	Priya Singh	AB+
	Anil Mehta	O-
	Neha Jain	A-
	Manoj Rathi	B-
	Pooja Gupta	AB-
	Karan Thakur	A+
	Deepa Rao	B+

7. WHERE and PATTERN MATCHING

SELECT * FROM donors WHERE blood_group = 'A+';

	id	name	blood_group	contact	address	age
▶	1	Ravi Kumar	A+	9876543210	Delhi	18
	9	Karan Thakur	A+	6677554433	Ahmedabad	18

SELECT * FROM donors WHERE name LIKE 'P%';

	id	name	blood_group	contact	address	age
▶	4	Priya Singh	AB+	6543210987	Chennai	18
	8	Pooja Gupta	AB-	7788665544	Pune	18

8. ORDER BY and GROUP BY

SELECT * FROM donors ORDER BY name ASC;

	id	name	blood_group	contact	address	age
▶	3	Aman Verma	O+	7654321098	Kolkata	18
	5	Anil Mehta	O-	9123456780	Bangalore	18
	10	Deepa Rao	B+	5566443322	Surat	18
	9	Karan Thakur	A+	6677554433	Ahmedabad	18
	7	Manoj Rathi	B-	8899776655	Jaipur	18
	6	Neha Jain	A-	9988776655	Hyderabad	18
	8	Pooja Gupta	AB-	7788665544	Pune	18
	4	Priya Singh	AB+	6543210987	Chennai	18
	1	Ravi Kumar	A+	9876543210	Delhi	18
	2	Sneha Sharma	B+	8765432109	Mumbai	18

SELECT blood_group, COUNT(*) AS count FROM donors GROUP BY blood_group;

	blood_group	count
▶	A+	2
	B+	2
	O+	1
	AB+	1
	O-	1
	A-	1
	B-	1
	AB-	1

9. JOIN ON MULTIPLE TABLES

```
SELECT p.name AS patient_name, t.blood_group, t.quantity  
FROM transactions t  
JOIN patients p ON t.patient_id = p.id;
```

	patient_name	blood_group	quantity
▶	Rahul Yadav	A+	2
	Meena Joshi	B+	1
	Alok Ranjan	O+	3
	Kavita Das	AB+	1
	Sunil Kumar	O-	2
	Sita Rani	A-	1
	Vinay Rao	B-	2
	Gita Shah	AB-	1
	Dilip Mehta	A+	3
	Anita Nair	B+	2

10. NESTED QUERIES

```
SELECT * FROM donors  
WHERE id IN (SELECT patient_id FROM transactions WHERE quantity > 2);
```

	id	name	blood_group	contact	address	age
▶	3	Aman Verma	O+	7654321098	Kolkata	18
	9	Karan Thakur	A+	6677554433	Ahmedabad	18

11. SET OPERATIONS

-- UNION

SELECT name FROM donors

UNION

SELECT name FROM patients;

	name
►	Ravi Kumar
	Sneha Sharma
	Aman Verma
	Priya Singh
	Anil Mehta
	Neha Jain
	Manoj Rath
	Pooja Gupta
	Karan Thakur
	Deepa Rao
	Rahul Yadav
	Meena Joshi
	Alok Ranjan
	Kavita Das
	Sunil Kumar

12. INTERSECT (Simulated using JOIN)

SELECT d.name

FROM donors d

JOIN patients p ON d.name = p.name;

Or

SELECT name FROM donors

INTERSECT

SELECT name FROM patients;

13. EXCEPT (Simulated using NOT IN)

SELECT name FROM donors

WHERE name NOT IN (SELECT name FROM patients);

14. STRING FUNCTIONS

SELECT name, UPPER(name) AS uppercase, LENGTH(name) AS name_length
FROM donors;

	name	uppercase	name_length
▶	Ravi Kumar	RAVI KUMAR	10
	Sneha Sharma	SNEHA SHARMA	12
	Aman Verma	AMAN VERMA	10
	Priya Singh	PRIYA SINGH	11
	Anil Mehta	ANIL MEHTA	10
	Neha Jain	NEHA JAIN	9
	Manoj Rathi	MANOJ RATHI	11
	Pooja Gupta	POOJA GUPTA	11
	Karan Thakur	KARAN THAKUR	12
	Deepa Rao	DEEPA RAO	9

15. AGGREGATE FUNCTIONS

SELECT COUNT(*) AS total_donors FROM donors;

	total_donors
▶	10

SELECT AVG(quantity) AS avg_quantity FROM transactions;

	avg_quantity
▶	1.8000

SELECT MAX(quantity), MIN(quantity) FROM transactions;

	MAX(quantity)	MIN(quantity)
▶	3	1

16. COMMIT, TRANSACTIONS, AND ROLLBACK

```
START TRANSACTION;  
INSERT INTO donors (id, name, blood_group, contact, address)  
VALUES (11, 'Test User', 'O+', '0000000000', 'TestCity');  
ROLLBACK  
COMMIT;
```

17. TRIGGERS

```
DELIMITER //  
CREATE TRIGGER check_quantity  
BEFORE INSERT ON transactions  
FOR EACH ROW  
BEGIN  
    IF NEW.quantity <= 0 THEN  
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Quantity must be  
greater than zero';  
    END IF;  
END //  
DELIMITER ;
```

✓ 50 06:01:07 CREATE TRIGGER check_quantity BEFORE INSERT ON transactions FOR EACH ROW BEGIN IF NEW.q... 0 row(s) affected

18. PROCEDURE (PL/SQL Concept)

```
DELIMITER //  
CREATE PROCEDURE GetPatientDetails(IN pid INT)  
BEGIN  
    SELECT * FROM patients WHERE id = pid;  
END //  
DELIMITER ;
```

```
-- CALL PROCEDURE  
CALL GetPatientDetails(1);
```

	id	name	blood_group	contact	address
▶	1	Rahul Yadav	A+	9123412345	Delhi

19. SAVEPOINTS

```
START TRANSACTION;  
SAVEPOINT sp1;  
INSERT INTO donors (id, name, blood_group, contact, address)  
VALUES (12, 'Rollback User', 'B+', '0000000001', 'RollbackCity');  
ROLLBACK TO sp1;  
COMMIT;
```

20. VIEW

```
CREATE VIEW AvailableBlood AS  
SELECT blood_group, SUM(quantity) AS total_quantity  
FROM blood_stock  
GROUP BY blood_group;  
SELECT * FROM AvailableBlood;
```

	blood_group	total_quantity
▶	A-	10
	A+	10
	AB-	5
	AB+	12
	B-	7
	B+	18
	O-	8
	O+	25

NORMALIZATION FOR BLOOD BANK MANAGEMENT SYSTEM

1. ADMIN Table

	username	password
▶	admin1	pass123
	admin10	adminpass
	admin2	admin@456
	admin3	secure789
	admin4	admin!001
	admin5	pass555
	admin6	qwerty99
	admin7	letmein22
	admin8	root321
	admin9	superadmin

Normal Form	Status	Reason
1NF	Yes	Atomic values, no repeating groups
2NF	Yes	Fully functionally dependent on the primary key
3NF	Yes	No transitive dependencies
4NF	Yes	No multivalued dependencies
5NF	Yes	No join dependencies

2. BLOOD_STOCK Table

	blood_group	quantity
▶	A-	10
	A+	10
	AB-	5
	AB+	12
	B-	7
	B+	18
	O-	8
	O+	25

Normal Form	Status	Reason
1NF	Yes	Atomic values
2NF	Yes	Fully dependent on the primary key
3NF	Yes	No transitive dependencies
4NF	Yes	No multivalued dependencies
5NF	Yes	No join dependencies

3. DONORS Table

	id	name	blood_group	contact	address	age
▶	1	Ravi Kumar	A+	9876543210	Delhi	18
	2	Sneha Sharma	B+	8765432109	Mumbai	18
	3	Aman Verma	O+	7654321098	Kolkata	18
	4	Priya Singh	AB+	6543210987	Chennai	18
	5	Anil Mehta	O-	9123456780	Bangalore	18
	6	Neha Jain	A-	9988776655	Hyderabad	18
	7	Manoj Rathi	B-	8899776655	Jaipur	18
	8	Pooja Gupta	AB-	7788665544	Pune	18
	9	Karan Thakur	A+	6677554433	Ahmedabad	18
	10	Deepa Rao	B+	5566443322	Surat	18

Normal Form	Status	Reason
1NF	Yes	All values are atomic
2NF	Yes	No partial dependencies
3NF	Yes	No transitive dependencies
4NF	Yes	No multivalued dependencies
5NF	Yes	No join dependencies

4. PATIENTS Table

	id	name	blood_group	contact	address
▶	1	Rahul Yadav	A+	9123412345	Delhi
	2	Meena Joshi	B+	8123456789	Mumbai
	3	Alok Ranjan	O+	7123456789	Lucknow
	4	Kavita Das	AB+	6123456789	Chennai
	5	Sunil Kumar	O-	5123456789	Bangalore
	6	Sita Rani	A-	4123456789	Patna
	7	Vinay Rao	B-	3123456789	Indore
	8	Gita Shah	AB-	2123456789	Pune
	9	Dilip Mehta	A+	1123456789	Nagpur
	10	Anita Nair	B+	9999999999	Kochi

Normal Form	Status	Reason
1NF	Yes	Atomic attributes
2NF	Yes	Fully functionally dependent on the primary key
3NF	Yes	No transitive dependencies
4NF	Yes	No multivalued dependencies
5NF	Yes	No join dependencies

5. TRANSACTIONS Table

	id	patient_id	blood_group	quantity
▶	1	1	A+	2
	2	2	B+	1
	3	3	O+	3
	4	4	AB+	1
	5	5	O-	2
	6	6	A-	1
	7	7	B-	2
	8	8	AB-	1
	9	9	A+	3
	10	10	B+	2

Normal Form	Status	Reason
1NF	Yes	Atomic values
2NF	Yes	Fully dependent on the composite primary key
3NF	Yes	No transitive dependencies
4NF	Yes	No multivalued dependencies
5NF	Yes	No join dependencies

Final Summary:

Table Name	1NF	2NF	3NF	4NF	5NF	Reason
Admin	✓	✓	✓	✓	✓	Atomic values, single attribute PK, no transitive or join dependencies
Blood_Stock	✓	✓	✓	✓	✓	All fields dependent on blood_group; no multivalued fields
Donors	✓	✓	✓	✓	✓	Atomic fields, full functional dependency on PK
Patients	✓	✓	✓	✓	✓	Same structure as Donors; no partial or transitive dependencies
Transactions	✓	✓	✓	✓	✓	Full dependency on composite key; no multivalued or join dependencies

CONCURRENCY AND CONSISTENCY

In a Blood Bank system, multiple users (admins, doctors, nurses) might be:

- Adding new donors.
- Updating blood stock.
- Processing blood requests for patients.

1. Transactions

Wrap critical operations (like issuing blood) in a transaction to ensure atomicity:

START TRANSACTION;

UPDATE blood_stock

SET quantity = quantity - 2

WHERE blood_group = 'A+' AND quantity >= 2;

INSERT INTO transactions (id, patient_id, blood_group, quantity)

VALUES (11, 3, 'A+', 2);

2. Locking Mechanism

SELECT quantity FROM blood_stock WHERE blood_group = 'A+' FOR
UPDATE;

3. Isolation Levels

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

READ COMMITTED: Avoid dirty reads.

REPEATABLE READ: Prevent non-repeatable reads.

SERIALIZABLE: Highest isolation; slow but safest.

Data Consistency

1. Foreign Keys

Maintain relational integrity:

```
ALTER TABLE transactions
```

```
ADD CONSTRAINT fk_patient
```

```
FOREIGN KEY (patient_id) REFERENCES patients(id);
```

2. Check Constraints

Validate data at the time of entry:

```
ALTER TABLE blood_stock
```

```
ADD CONSTRAINT chk_quantity CHECK (quantity >= 0);
```

3. Triggers

Ensure consistency rules automatically:

```
CREATE TRIGGER before_blood_issue
```

```
BEFORE INSERT ON transactions
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    DECLARE available_qty INT;
```

```
    SELECT quantity INTO available_qty FROM blood_stock WHERE blood_group  
= NEW.blood_group;
```

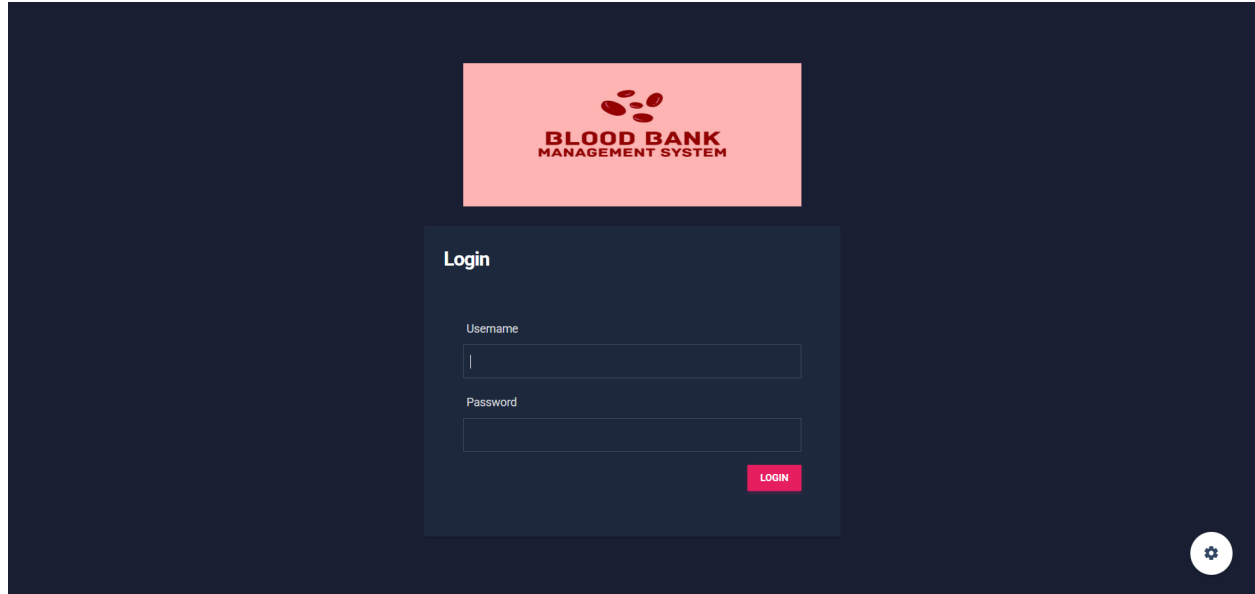
```
    IF available_qty < NEW.quantity THEN
```

```
        SIGNAL SQLSTATE '45000'
```

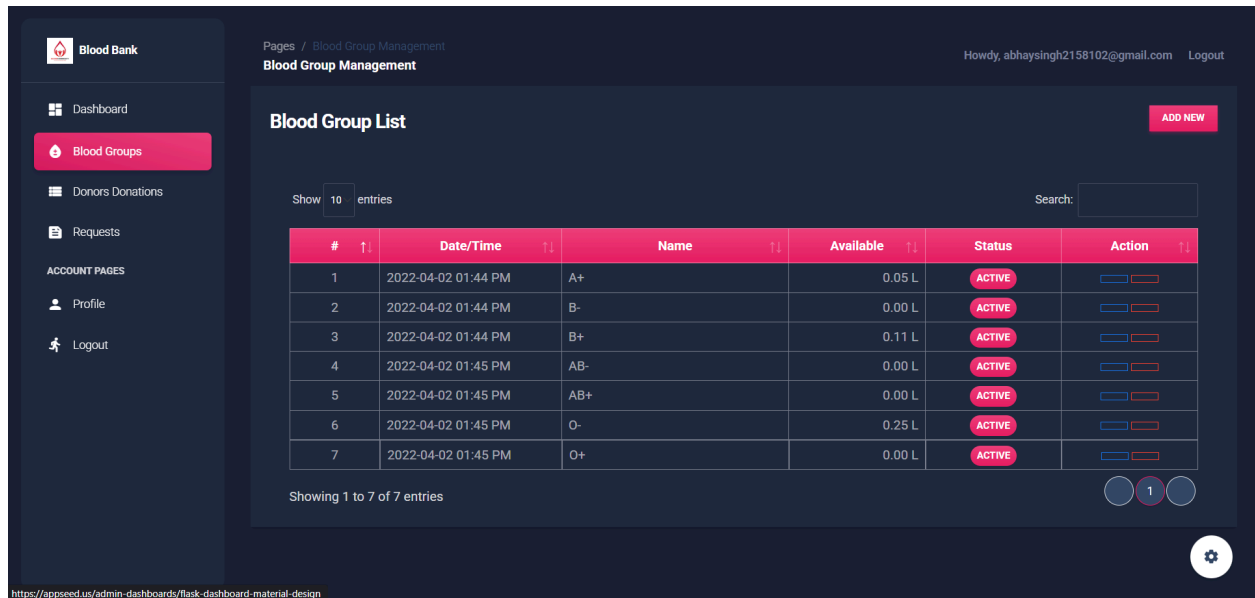
```
    SET MESSAGE_TEXT = 'Insufficient blood stock';  
END IF;  
END;
```

SCREENSHOTS


Login Page:



Admin Dashboard:



Blood Request Page:

 Blood Bank

Dashboard

Blood Groups

Donors Donations

Requests

ACCOUNT PAGES

Profile

Logout

Pages / Blood Requests Management

Blood Requests Management

Howdy, abhaysingh2158102@gmail.com Logout

Requests List

ADD NEW

Show 10 entries

Search:


#	DateTime	Patient	Physician	Blood Type (Volume)	Action
1	2022-05-05	Rufas	Dr.sabbir	A+ (200.00 ml)	ACTION
2	2025-04-23	abhay	gupta	B+ (-107.00 ml)	ACTION

Showing 1 to 2 of 2 entries

1

127.0.0.18000

Donors Page:

 Blood Bank

Dashboard

Blood Groups

Donors Donations

Requests

ACCOUNT PAGES

Profile

Logout

Pages / Donors Blood Donations Management

Donors Blood Donations Management

Howdy, abhaysingh2158102@gmail.com Logout

Donors Donation List

ADD NEW

Show 10 entries

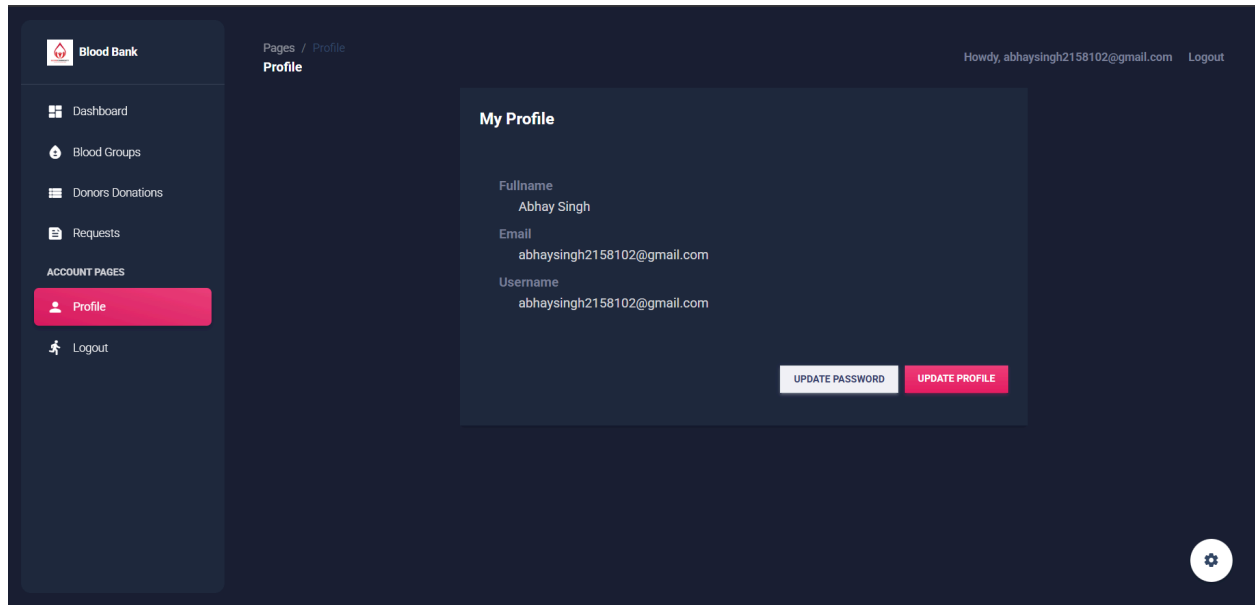
Search:

#	Transfusion DateTime	Donor	Contact	Blood Type (Volume)	Action
1	2022-05-05	person 1	123456789	A+ (250.00 ml)	ACTION
2	2022-05-05	person 2	12345678	O- (250.00 ml)	ACTION

Showing 1 to 2 of 2 entries

1

User Profile:



CONCLUSION

The Blood Bank Management System has been designed and implemented to streamline the critical operations of blood collection, storage, and distribution. By integrating multiple modules such as donor registration, patient requests, blood stock tracking, and transaction history, the system ensures efficient and transparent management of blood resources.

Through the application of data normalization (up to 5NF), the database structure minimizes redundancy and enhances data integrity. Features such as foreign key constraints, check conditions, triggers, and transactions have been incorporated to maintain data consistency and support data concurrency, ensuring that multiple users can safely interact with the system in real-time without conflict.

This project not only enhances operational efficiency but also ensures the accuracy and security of sensitive medical data. The system serves as a scalable and reliable solution for healthcare institutions to manage their blood bank operations and can be further enhanced by integrating features like automated notifications, blood compatibility checks, and online donor-patient matching in future versions.

In conclusion, this project lays a strong foundation for digitizing blood bank operations and contributing to faster, safer, and more accessible healthcare services.