Blood Bank Database Management

Mini Project Report -Database Lab (DSE 2241)

Department of Data Science & Computer Applications

B. Tech Data Science

4th Semester – Batch: B1/B2/B3/B4- Group:

Submitted By

Vedant Ganesh	230968340
Upashana N.Bhojake	230968354
Joyal	230968362
Manish R	230968364
Satwik Jain	230968378
Vanshika Mittal	230968382
Himanshu B.Agarwal	230968390
Trisal Palla	230968394

Mentored By

Shreenidhi Bhat
Assistant Professor
DSCA, MIT

Archana H
Assistant Professor-Senior
DSCA, MIT

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Date:

CERTIFICATE

This is to certify that the Vedant Ganesh (230968340), Upashana N.Bhojake (230968354), Joyal (230968362), Manish R (230968364), Satwik Jain (230968378), Vanshika Mittal (230968382), Himanshu B.Agarwal (230968390), Trisal Palla (230968394) have successfully executed a mini project titled "Blood Bank Database Management" rightly bringing fore the competencies and skill sets they have gained during the course- Database Lab (DSE 2241), thereby resulting in the culmination of this project.

Shreenidhi Bhat	**Archana H**
Assistant Professor	**Assistant Professor-Senior**
DSCA, MIT	**DSCA, MIT**

ABSTRACT

The Blood Bank Management System is a robust and comprehensive digital platform developed to enhance the operational efficiency of blood banks, whether they function independently, within organizations, or as part of larger healthcare networks. This system is specifically designed to eliminate the challenges and limitations associated with traditional manual processes, which often lead to errors in donor records, delays in blood supply, and inefficient inventory tracking. By integrating various critical functionalities into a centralized platform, the system facilitates the seamless management of donor information, including eligibility status, medical history, and contact details. It also maintains real-time blood stock inventory levels, ensuring that shortages are promptly identified and addressed. Additionally, the system supports patient registration, crossmatching for blood compatibility, and thorough documentation of medical tests conducted on blood samples, all of which are essential for safe and effective transfusion practices. The Blood Bank Management System significantly improves decision-making processes, enhances the safety and reliability of blood distribution, and ultimately contributes to better healthcare outcomes.

Contents

Chapter 1

Introduction

Blood banks are essential to the healthcare system, playing a vital role in saving lives by ensuring the availability of safe and compatible blood for patients during surgeries, accidents, chronic illnesses, and emergency situations. The operations involved in running a blood bank are complex and multifaceted, including donor recruitment, eligibility assessment, medical testing, inventory management, and patient-blood compatibility matching. Traditionally, many of these operations have been handled manually, which often leads to inefficiencies, human errors, delayed responses, and difficulties in maintaining accurate and up-to-date records. These limitations can have serious consequences, particularly when there is an urgent need for a specific blood type.

To address these challenges, a modern, digital solution is required—one that not only enhances accuracy and speed but also supports decision-making with real-time data. The Blood Bank Management System has been designed to fulfil this need by automating and centralizing the core processes involved in blood bank operations. The system ensures that critical data such as donor eligibility, blood stock levels, and patient requirements are managed seamlessly and securely. It streamlines workflows and reduces dependency on paper-based records or fragmented systems, thereby minimizing delays and improving operational efficiency.

This project has been developed using **Oracle SQL** and **Oracle APEX (Application Express)**, two powerful tools from Oracle that together offer a strong and scalable database-driven application framework. Oracle SQL is used for efficient and structured data storage, manipulation, and retrieval, ensuring that all information is consistently maintained and easily accessible. Oracle APEX, a low-code development platform, provides a responsive, web-based interface for users to interact with the system—whether they are blood bank staff, healthcare providers, or administrators.

Furthermore, the system leverages **Oracle APEX triggers and SQL queries** to handle various automated tasks and validations within the application. For example, triggers are used to automatically update the inventory when a blood unit is issued or received, validate donor eligibility based on last donation date and medical test results, and ensure that crossmatching procedures are carried out correctly before blood is assigned to a patient. These built-in automations enhance the integrity of the system and reduce the chances of manual oversight.

In summary, the integration of Oracle technologies in the Blood Bank Management System ensures a secure, efficient, and user-friendly platform that enhances the accuracy and responsiveness of blood bank operations. It provides real-time access to critical data, streamlines administrative tasks, and supports safer blood transfusion practices, ultimately contributing to improved patient care and more effective healthcare delivery.

Chapter 2

Synopsis

2.1 Proposed System

The proposed Blood Bank Management System is a web-based application designed to automate and streamline key blood bank operations, including donor registration, blood inventory management, patient handling, and blood crossmatching. Built using **Oracle SQL** for backend data management and **Oracle APEX** for the user interface, the system ensures real-time data access, accuracy, and efficiency.

It maintains a centralized database of donor details, eligibility status, patient records, test results, and inventory levels. **Oracle APEX triggers** are used to automate critical tasks such as inventory updates, donor eligibility validation, and compatibility checks. Role-based access ensures data security and limits system access based on user roles.

2.2 Objectives

- To develop a centralized platform for managing donor, patient, and blood inventory records efficiently.
- To track donor eligibility based on criteria such as age, last donation date, and health test results.
- To maintain accurate test records for each blood sample collected, including screening and compatibility checks.
- To implement patient registration and perform blood crossmatching to ensure transfusion safety.
- To automate inventory updates using Oracle SQL triggers during blood donation, issue, or disposal.
- To facilitate real-time retrieval of donor and patient information for quick and informed decision-making.
- To provide a user-friendly interface using Oracle APEX for seamless interaction by different user roles.
- To issue alerts or notifications for low stock levels, expired units, and upcoming donation eligibility.

- To ensure data integrity, security, and controlled access through role-based privileges and validations.

Chapter 3

Functional Requirements

3.1 Donor registration , Donation record & Blood Stock

3.1.1 New Donor registration

Table 3.1 Registration

INPUT	Name, age, gender, blood_type, contact,email, last_donation_date, other medical history
Processing	System checks for eligibilty by triggering the donor's medical history, if it checks out, the trigger checks if the last_donation_date(if not null)>3 months, only then will it make the donor eligible
OUTPUT	Donor details are registered and the donor is ready to donate if eligible

3.1.2 Donation Record

Table 3.2 Donation record

INPUT	Donor_id, blood_type, date_of_donation, infection, quantity
Processing	Updates the test status if it comes back approved, the donated blood is ready to update the blood stock
OUTPUT	The test status is updated

3.1.3 Blood Stock

Table 3.3 Blood Stock record

INPUT	Donation_ID,Location
Processing	Updates the blood stock with blood type,quantity on the basis of donation_ID and the expiry by adding 6 weeks to the date of donation.
OUTPUT	The stock is updated with the relevant details

3.2 Cross matching request

3.2.1 Cross match validation

Table 3.4 Cross match validation

INPUT	Patient_id, cross_match_id
Processing	It sorts the blood_stock on expiry date basis. Matches the request based on blood type and quantity. Assigns it, deducts the required amount of blood from the patient table and updates the stock.
OUTPUT	Cross match request results, fulfilled, partially fulfilled or Not failed

Chapter 4:Detailed Design

4.1 ER DIAGRAM

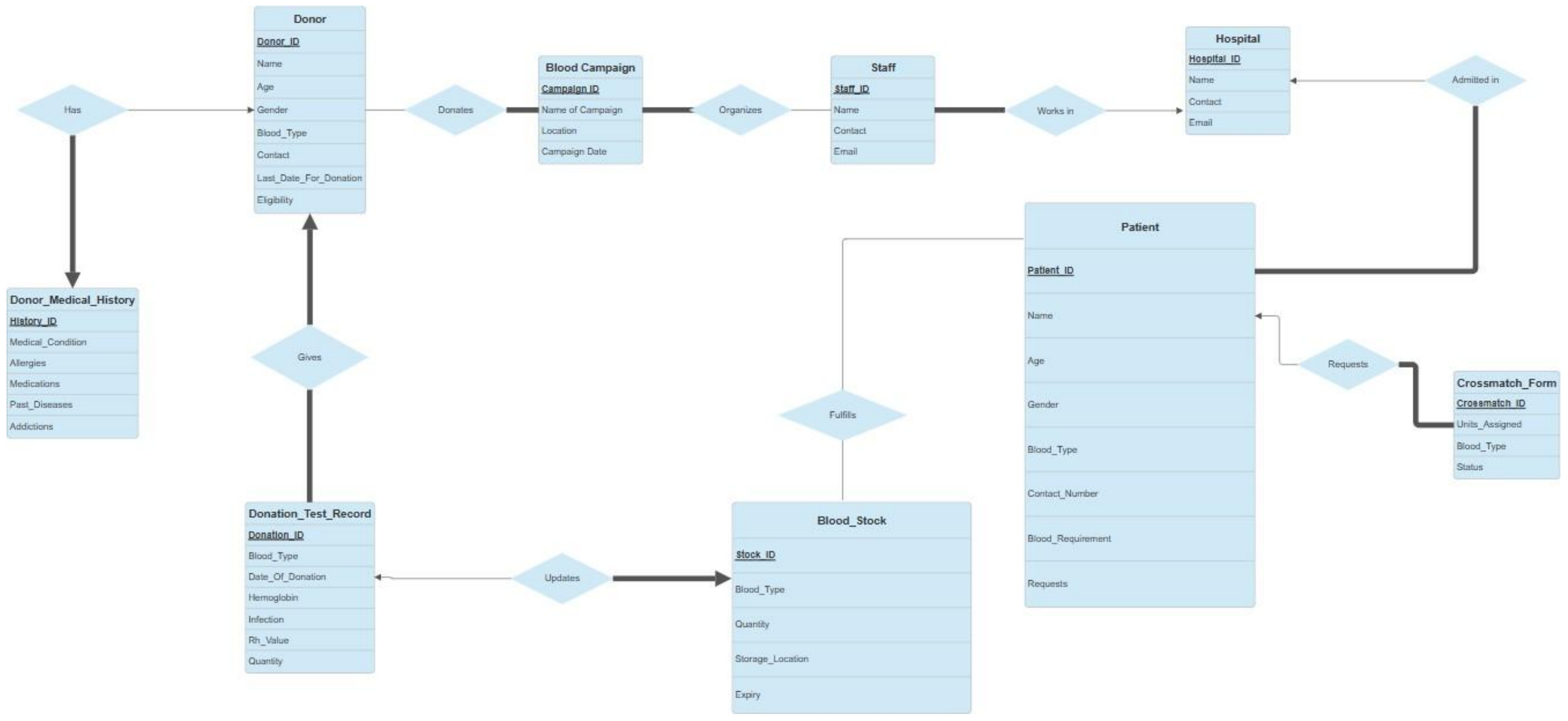


Figure 4.1 ER Diagram

4.2 Schema Diagram

Blood_Campaign(campaign_id, name_of_campaign, location, campaign_date)

Blood_Stock(stock_id, donation_id, blood_type, quantity, storage_location, expiry)

Donation_id refers to donation_test_record

Crossmatch_Form(crossmatch_id, patient_id, units_assigned, blood_type, status)

Patient_id refers to patient

Crossmatch_junction(stock_id, patient_id)

Stock_id referes to blood_stock,

Patient_id referes to patient

Donation_test_record(donation_id, donor_id,blood_type, date_of_donation, hemoglobin, infection, rh_value, test_status, quantity)

Donor_id refers to donor

Donor(donor_id,name,age,gender,blood_type,contact,last_date_for_donation,elegibility)

Donor_medical_history(history_id, donor_id, last_checkup, medical_condition, allergies, medication, past_diseases, addictions)

Donor_id refers to donor

Donor_participates(donor_id, campagin_id)

Donor_id refers donor, campaigin_id refers to blood_campaign

Hospital(hospital_id, name, contact, email)

Patient(patient_id, name, age, gender, blood_type, contact_number, blood_requirements, requests, hospital_id)

Hospital_id refers to hospital

Staff(staff_id, name, contact, email, hospital_id)

Hospital id refers to hospital

Staff_organizes(staff_id, campaign_id, role)

Campaign_id refers to campaign

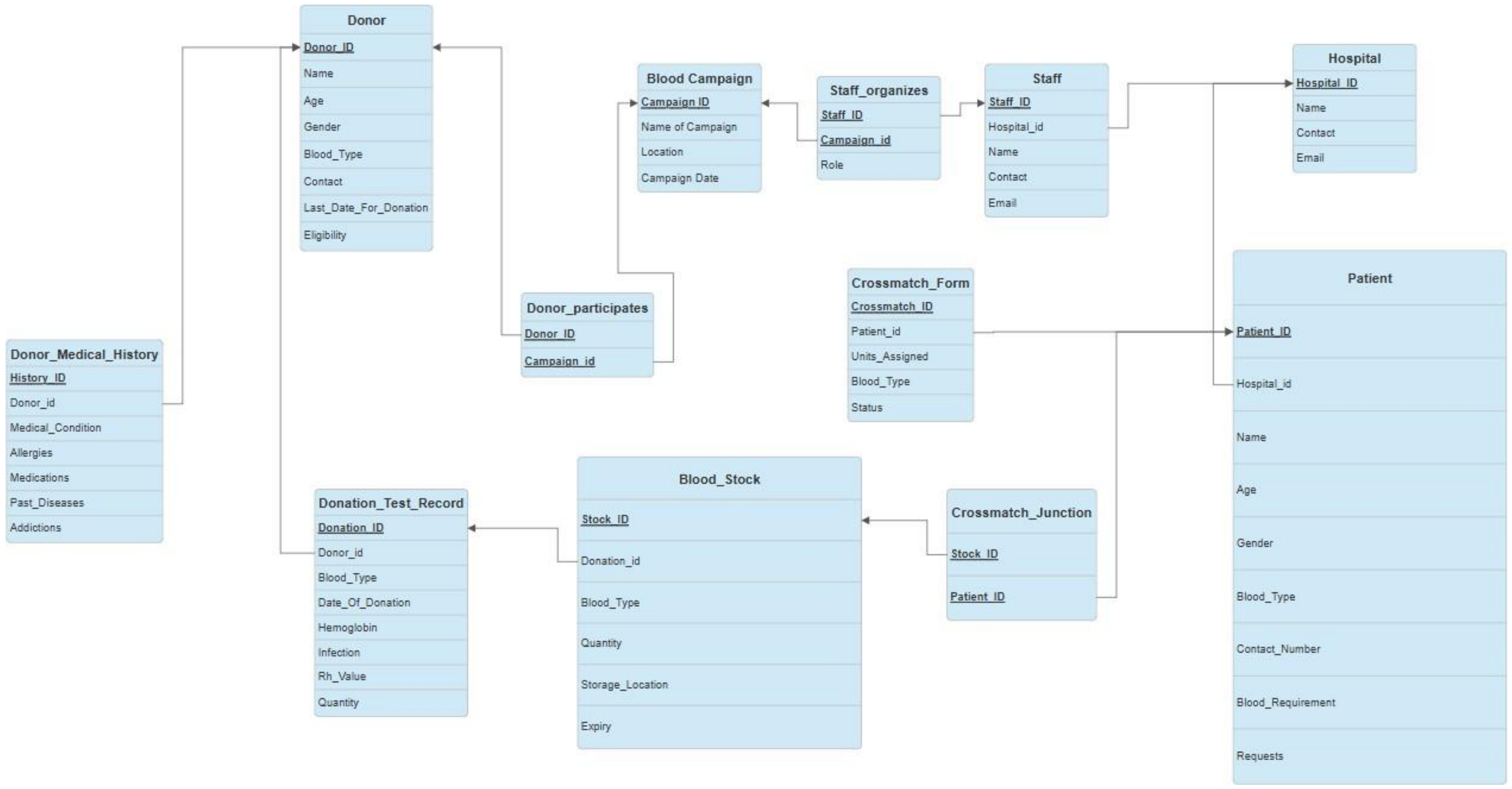


Figure 4.2 Schema Diagram

4.3 Data Dictionary

BLOOD_CAMPAIGN

Column	Data type (size)	Constraint
CAMPAIGN_ID	VARCHAR2(20 BYTE)	Primary Key
NAME_OF_CAMPAIGN	VARCHAR2(20 BYTE)	
LOCATION	VARCHAR2(100 BYTE)	
CAMPAIGN_DATE	DATE	

BLOOD_STOCK

Column	Data type (size)	Constraint
STOCK_ID	VARCHAR2(20 BYTE)	Primary Key
DONATION_ID	VARCHAR2(20 BYTE)	Foreign Key referencing Donation_Test_Record
BLOOD_TYPE	VARCHAR2(3 BYTE)	Check(Blood_Type IN ('A+', 'A–', 'B+', 'B–', 'AB+', 'AB–', 'O+', 'O–')
QUANTITY	NUMBER(3,2)	Not null
STORAGE_LOCATION	VARCHAR2(100 BYTE)	Not null
EXPIRY	DATE	Not null

CROSSMATCH_FORM

Column	Data type (size)	Constraint
CROSSMATCH_ID	VARCHAR2(5 BYTE)	Primary Key
PATIENT_ID	VARCHAR2(50 BYTE)	Foreign Key referencing Patient
UNITS_ASSIGNED	NUMBER(3,2)	
BLOOD_TYPE	VARCHAR2(5 BYTE)	
STATUS	VARCHAR2(20 BYTE)	Status in('Fulfilled','Partially Fulfilled','Failed')

CROSSMATCH_JUNCTION

Column	Data type (size)	Constraint
STOCK_ID	VARCHAR2(5 BYTE)	Foreign Key referencing Blood_Stock, ON DELETE CASCADE
PATIENT_ID	VARCHAR2(50 BYTE)	Foreign Key referencing Patient,ON DELETE CASCADE

DONATION_TEST_RECORD

Column	Data type (size)	Constraint
DONATION_ID	VARCHAR2(20 BYTE)	Primary Key
DONOR_ID	VARCHAR2(20 BYTE)	Foreign Key referencing Donor
BLOOD_TYPE	VARCHAR2(5 BYTE)	Check(Blood_Type IN ('A+', 'A−', 'B+', 'B−', 'AB+', 'AB−', 'O+', 'O−')
DATE_OF_DONATION	DATE	Not null
HEMOGLOBIN	NUMBER(4,2)	Not null,Check(Hemoglobin between 13 and 17)
INFECTION	VARCHAR2(20 BYTE)	Not null,Infection IN ('infected', 'not_infected')
RH_VALUE	VARCHAR2(8 BYTE)	Not null, Rh_Value IN ('positive', 'negative')
TEST_STATUS	VARCHAR2(30 BYTE)	
QUANTITY	NUMBER(3,2)	Not null

DONOR

Column	Data type (size)	Constraint
DONOR_ID	VARCHAR2(20 BYTE)	Primary Key
NAME	VARCHAR2(100 BYTE)	Not null
AGE	NUMBER	Not null, check(Age>18 and Age<=70)
GENDER	VARCHAR2(1 BYTE)	Not null, check(Gender in(‘M’,’F’)
BLOOD_TYPE	VARCHAR2(4 BYTE)	Check(Blood_Type IN ('A+', 'A−', 'B+', 'B−', 'AB+', 'AB−', 'O+', 'O−')
CONTACT	VARCHAR2(10 BYTE)	Unique
LAST_DATE_FOR_DONATION	DATE	
ELIGIBILITY	VARCHAR2(20 BYTE)	Eligibility IN ('Eligible', 'Not Eligible')

DONOR_MEDICAL_HISTORY

Column	Data type (size)	Constraint
HISTORY_ID	VARCHAR2(20 BYTE)	Primary Key
DONOR_ID	VARCHAR2(20 BYTE)	Foreign Key referencing Donor
MEDICAL_CONDITION	VARCHAR2(20 BYTE)	Medical_Condition IN ('medically_fit', 'medically_unfit')
ALLERGIES	VARCHAR2(255 BYTE)	
MEDICATION	VARCHAR2(255 BYTE)	
PAST_DISEASES	VARCHAR2(255 BYTE)	
ADDICTIONS	VARCHAR2(3 BYTE)	CHECK(Addictions IN ('yes', 'no'))

DONOR_PARTICIPATES

Column	Data type (size)	Constraint
DONOR_ID	VARCHAR2(20 BYTE)	Foreign Key referencing Donor,primary key
CAMPAIGN_ID	VARCHAR2(20 BYTE)	Foreign Key referencing Blood_Campaign,primary key

HOSPITAL

Column	Data type (size)	Constraint
HOSPITAL_ID	VARCHAR2(5 BYTE)	Primary Key
NAME	VARCHAR2(20 BYTE)	
CONTACT	VARCHAR2(10 BYTE)	Unique
EMAIL	VARCHAR2(50 BYTE)	

PATIENT

Column	Data type (size)	Constraint
PATIENT_ID	VARCHAR2(50 BYTE)	Primary Key
NAME	VARCHAR2(100 BYTE)	Not null
AGE	NUMBER	Not null,age between 1 and 120
GENDER	VARCHAR2(1 BYTE)	Gender IN(‘M’,’F’)
BLOOD_TYPE	VARCHAR2(5 BYTE)	Blood_type IN ('A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-'),not null
CONTACT_NUMBER	VARCHAR2(15 BYTE)	Unique
BLOOD_REQUIREMENT	VARCHAR2(20 BYTE)	Blood_requirement IN ('Required', 'Not Required')
REQUESTS	NUMBER	
HOSPITAL_ID	VARCHAR2(5 BYTE)	Foreign Key referencing Hospital ID

STAFF

Column	Data type (size)	Constraint
STAFF_ID	VARCHAR2(10 BYTE)	Primary Key
NAME	VARCHAR2(10 BYTE)	
CONTACT	VARCHAR2(10 BYTE)	unique
EMAIL	VARCHAR2(50 BYTE)	
HOSPITAL_ID	VARCHAR2(5 BYTE)	Foreign Key referencing Hospital

STAFF_ORGANIZES

Column	Data type (size)	Constraint
STAFF_ID	VARCHAR2(10 BYTE)	Foreign Key referencing Staff
CAMPAIGN_ID	VARCHAR2(20 BYTE)	Foreign Key referencing Blood_Campaign
ROLE	VARCHAR2(10 BYTE)	Check(Role in(‘Manager’,’Medical Staff’))

4.4 Relational Model Implementation

CREATE TABLE BLOOD_STOCK (STOCK_ID VARCHAR2(20 BYTE) PRIMARY KEY, DONATION_ID VARCHAR2(20 BYTE), BLOOD_TYPE VARCHAR2(3 BYTE) CHECK (BLOOD_TYPE IN ('A+', 'A−', 'B+', 'B−', 'AB+', 'AB−', 'O+', 'O−')), QUANTITY NUMBER(3,2) NOT NULL, STORAGE_LOCATION VARCHAR2(100 BYTE) NOT NULL, EXPIRY DATE NOT NULL, FOREIGN KEY (DONATION_ID) REFERENCES DONATION_TEST_RECORD(DONATION_ID));

CREATE TABLE CROSSMATCH_JUNCTION (STOCK_ID VARCHAR2(5 BYTE), PATIENT_ID VARCHAR2(50 BYTE), CONSTRAINT fk_stock_id FOREIGN KEY (STOCK_ID) REFERENCES BLOOD_STOCK(STOCK_ID) ON DELETE CASCADE, CONSTRAINT fk_patient_id FOREIGN KEY (PATIENT_ID) REFERENCES PATIENT(PATIENT_ID) ON DELETE CASCADE, PRIMARY KEY (STOCK_ID, PATIENT_ID));

CREATE TABLE CROSSMATCH_FORM (CROSSMATCH_ID VARCHAR2(5 BYTE) PRIMARY KEY, PATIENT_ID VARCHAR2(50 BYTE), UNITS_ASSIGNED NUMBER(3,2), BLOOD_TYPE VARCHAR2(5 BYTE), STATUS VARCHAR2(20 BYTE) CHECK (STATUS IN ('Fulfilled', 'Partially Fulfilled', 'Failed')), FOREIGN KEY (PATIENT_ID) REFERENCES PATIENT(PATIENT_ID));

CREATE TABLE DONATION_TEST_RECORD (DONATION_ID VARCHAR2(20 BYTE) PRIMARY KEY, DONOR_ID VARCHAR2(20 BYTE), BLOOD_TYPE VARCHAR2(5 BYTE) CHECK (BLOOD_TYPE IN ('A+', 'A−', 'B+', 'B−', 'AB+', 'AB−', 'O+', 'O−')), DATE_OF_DONATION DATE NOT NULL, HEMOGLOBIN NUMBER(4,2) NOT NULL, INFECTION VARCHAR2(20 BYTE) NOT NULL CHECK (INFECTION IN ('infected', 'not_infected')), RH_VALUE VARCHAR2(8 BYTE) NOT NULL CHECK (RH_VALUE IN ('positive', 'negative')), TEST_STATUS VARCHAR2(30 BYTE), QUANTITY NUMBER(3,2) NOT NULL, FOREIGN KEY (DONOR_ID) REFERENCES DONOR(DONOR_ID));

CREATE TABLE DONOR (DONOR_ID VARCHAR2(20 BYTE) PRIMARY KEY, NAME VARCHAR2(100 BYTE) NOT NULL, AGE NUMBER NOT NULL CHECK (AGE BETWEEN 18 AND 70), GENDER VARCHAR2(1 BYTE) NOT NULL CHECK (GENDER IN ('M','F')), BLOOD_TYPE VARCHAR2(4 BYTE) CHECK (BLOOD_TYPE IN ('A+', 'A−', 'B+', 'B−', 'AB+', 'AB−', 'O+', 'O−')), CONTACT VARCHAR2(10 BYTE) UNIQUE,

```sql
LAST_DATE_FOR_DONATION DATE, ELIGIBILITY VARCHAR2(20 BYTE) CHECK
(ELIGIBILITY IN ('Eligible', 'Not Eligible')));

CREATE TABLE BLOOD_CAMPAIGN (CAMPAIGN_ID VARCHAR2(20 BYTE)
PRIMARY KEY, NAME_OF_CAMPAIGN VARCHAR2(20 BYTE), LOCATION
VARCHAR2(100 BYTE), CAMPAIGN_DATE DATE);

CREATE TABLE DONOR_PARTICIPATES (DONOR_ID VARCHAR2(20 BYTE),
CAMPAIGN_ID VARCHAR2(20 BYTE), PRIMARY KEY (DONOR_ID,
CAMPAIGN_ID), FOREIGN KEY (DONOR_ID) REFERENCES DONOR(DONOR_ID),
FOREIGN KEY (CAMPAIGN_ID) REFERENCES CAMPAIGN(CAMPAIGN_ID));

CREATE TABLE DONOR_MEDICAL_HISTORY (HISTORY_ID VARCHAR2(20
BYTE) PRIMARY KEY, DONOR_ID VARCHAR2(20 BYTE), MEDICAL_CONDITION
VARCHAR2(20 BYTE) CHECK (MEDICAL_CONDITION IN ('medically_fit',
'medically_unfit')), ALLERGIES VARCHAR2(255 BYTE), MEDICATION
VARCHAR2(255 BYTE), PAST_DISEASES VARCHAR2(255 BYTE), ADDICTIONS
VARCHAR2(3 BYTE) CHECK (ADDICTIONS IN ('yes', 'no')), FOREIGN KEY
(DONOR_ID) REFERENCES DONOR(DONOR_ID));

CREATE TABLE HOSPITAL (HOSPITAL_ID VARCHAR2(5 BYTE) PRIMARY KEY,
NAME VARCHAR2(20 BYTE), CONTACT VARCHAR2(10 BYTE) UNIQUE, EMAIL
VARCHAR2(50 BYTE));

CREATE TABLE PATIENT (PATIENT_ID VARCHAR2(50 BYTE) PRIMARY KEY,
NAME VARCHAR2(100 BYTE) NOT NULL, AGE NUMBER NOT NULL check(age
between 1 and 120), GENDER VARCHAR2(1 BYTE) CHECK (GENDER IN ('M','F')),
BLOOD_TYPE VARCHAR2(5 BYTE) NOT NULL CHECK (BLOOD_TYPE IN ('A+', 'A-',
'B+', 'B-', 'AB+', 'AB-', 'O+', 'O- ')), CONTACT_NUMBER VARCHAR2(15 BYTE)
UNIQUE, BLOOD_REQUIREMENT VARCHAR2(20 BYTE) CHECK
(BLOOD_REQUIREMENT IN ('Required', 'Not Required')), REQUESTS NUMBER,
HOSPITAL_ID VARCHAR2(5 BYTE), FOREIGN KEY (HOSPITAL_ID) REFERENCES
HOSPITAL(HOSPITAL_ID));

CREATE TABLE STAFF (STAFF_ID VARCHAR2(10 BYTE) PRIMARY KEY, NAME
VARCHAR2(10 BYTE), CONTACT VARCHAR2(10 BYTE) UNIQUE, EMAIL
VARCHAR2(50 BYTE), HOSPITAL_ID VARCHAR2(5 BYTE), FOREIGN KEY
(HOSPITAL_ID) REFERENCES HOSPITAL(HOSPITAL_ID));

CREATE TABLE STAFF_ORGANIZES (STAFF_ID VARCHAR2(10 BYTE),
CAMPAIGN_ID VARCHAR2(20 BYTE), ROLE VARCHAR2(10 BYTE) CHECK (ROLE
IN ('Manager','Medical Staff')), FOREIGN KEY (STAFF_ID) REFERENCES
STAFF(STAFF_ID), FOREIGN KEY (CAMPAIGN_ID) REFERENCES
CAMPAIGN(CAMPAIGN_ID));
```

5. Implementation

5.1 Queries

<u>For viewing Data</u>

```
SELECT CAMPAIGN_ID, NAME_OF_CAMPAIGN, LOCATION, CAMPAIGN_DATE
FROM CAMPAIGN;

SELECT STOCK_ID, DONATION_ID, BLOOD_TYPE, QUANTITY,
STORAGE_LOCATION, EXPIRY FROM BLOOD_STOCK;

SELECT CROSSMATCH_ID, PATIENT_ID, UNITS_ASSIGNED, BLOOD_TYPE,
STATUS FROM CROSSMATCH_FORM;

SELECT STOCK_ID, PATIENT_ID FROM CROSSMATCH_JUNCTION;

SELECT DONATION_ID, DONOR_ID, BLOOD_TYPE, DATE_OF_DONATION,
HEMOGLOBIN, INFECTION, RH_VALUE, TEST_STATUS, QUANTITY FROM
DONATION_TEST_RECORD;

SELECT DONOR_ID, NAME, AGE, GENDER, BLOOD_TYPE, CONTACT,
LAST_DATE_FOR_DONATION, ELIGIBILITY FROM DONOR;

SELECT HISTORY_ID, DONOR_ID, MEDICAL_CONDITION, ALLERGIES,
MEDICATION, PAST_DISEASES, ADDICTIONS FROM
DONOR_MEDICAL_HISTORY;

SELECT DONOR_ID, CAMPAIGN_ID FROM DONOR_PARTICIPATES;

SELECT HOSPITAL_ID, NAME, LOCATION, CONTACT, EMAIL FROM HOSPITAL;

SELECT PATIENT_ID, NAME, AGE, GENDER, BLOOD_TYPE, CONTACT_NUMBER,
BLOOD_REQUIREMENT, REQUESTS, HOSPITAL_ID FROM PATIENT;

SELECT STAFF_ID, NAME, CONTACT, EMAIL, HOSPITAL_ID FROM STAFF;

SELECT STAFF_ID, CAMPAIGN_ID, ROLE FROM STAFF_ORGANIZES;
```

5.2 Triggers

<u>TRG_DONOR_DONATION_DATE</u>

```
CREATE OR REPLACE TRIGGER TRG_DONOR_DONATION_DATE
BEFORE INSERT OR UPDATE ON DONOR
FOR EACH ROW
BEGIN
  IF :NEW.LAST_DATE_FOR_DONATION > SYSDATE THEN
```

```
    RAISE_APPLICATION_ERROR(-20001, 'Last donation date cannot be in the future.');
  END IF;
END;
/
```

DONOR_MEDICAL_HISTORY_T

```
create or replace TRIGGER DONOR_MEDICAL_HISTORY_T

BEFORE INSERT OR UPDATE ON Donor_Medical_History

FOR EACH ROW

DECLARE tempdate DATE;

BEGIN

IF :NEW.addictions = 'no' AND :NEW.allergies IS NULL AND :NEW.medication IS NULL
AND :NEW.past_diseases IS NULL THEN:NEW.medical_condition :=
'medically_fit';

SELECT last_date_for_donation
INTO tempdate
FROM donor
WHERE donor_id = :NEW.donor_id;


IF months_between(SYSDATE, tempdate) >= 3 THEN
  UPDATE donor
  SET eligibility = 'Eligible'
  WHERE donor_id = :NEW.donor_id;
ELSE
  UPDATE donor
  SET eligibility = 'Not Eligible'
  WHERE donor_id = :NEW.donor_id;
END IF;

ELSE :NEW.medical_condition := 'medically_unfit'; UPDATE donor SET eligibility = 'Not
Eligible' WHERE donor_id = :NEW.donor_id;

END IF;

END;
```

TRG_DATE_OF_DONATION_VALID

```
CREATE OR REPLACE TRIGGER TRG_DATE_OF_DONATION_VALID
BEFORE INSERT OR UPDATE ON DONATION_TEST_RECORD
FOR EACH ROW
BEGIN
  IF :NEW.DATE_OF_DONATION > SYSDATE THEN
    RAISE_APPLICATION_ERROR(-20002, 'Date of donation cannot be in the future.');
  END IF;
END;
/
```

DONATION_TEST_RECORD_T

```
create or replace TRIGGER DONATION_TEST_RECORD_T BEFORE INSERT OR UPDATE ON Donation_Test_Record

FOR EACH ROW

DECLARE v_blood_type VARCHAR2(4);

 BEGIN

SELECT Blood_Type INTO v_blood_type FROM Donor WHERE Donor_ID = :NEW.Donor_ID;

:NEW.Donation_ID := :NEW.Donation_ID
```

```
:NEW.Blood_Type := v_blood_type;

  if :NEW.Date_of_Donation is null then

  :NEW.Date_of_Donation := SYSDATE;

  end if;

  IF :NEW.Hemoglobin < 13.00 OR :NEW.Hemoglobin > 17.00 OR :NEW.Infection = 'infected'
  THEN

  :NEW.Test_Status := 'Not_Eligible_for_Updation';

  :NEW.Quantity := 0;

  ELSE :NEW.Test_Status := 'Eligible_for_Updation';

  IF :NEW.Quantity IS NULL THEN :NEW.Quantity := 0.45;

  END IF;

  END IF;

   END;

   /
```

BLOOD_STOCK_T

```
create or replace TRIGGER BLOOD_STOCK_T

BEFORE INSERT ON BLOOD_STOCK

FOR EACH ROW

DECLARE

v_blood_type VARCHAR2(5);

v_quantity NUMBER(3,2);

v_donation_date DATE;

BEGIN

SELECT Blood_Type, Quantity, Date_of_Donation INTO v_blood_type, v_quantity,
v_donation_date FROM Donation_Test_Record WHERE Test_Status =
'Eligible_for_Updation' AND Quantity > 0 AND Donation_ID = :NEW.Donation_ID;
```

:NEW.Blood_Type := v_blood_type;

:NEW.Quantity := v_quantity;

:NEW.Expiry := v_donation_date + 42;

EXCEPTION

WHEN NO_DATA_FOUND THEN RAISE_APPLICATION_ERROR(-20001, 'No eligible donation test record found to populate blood stock.');

END;

/

CROSSMATCH_FORM_T

create or replace TRIGGER CROSSMATCH_FORM_T

BEFORE INSERT ON Crossmatch_form

FOR EACH ROW

DECLARE

v_required NUMBER;

v_fulfilled NUMBER := 0;

blood_type1 VARCHAR2(10);

BEGIN

SELECT requests, blood_type INTO v_required, blood_type1 FROM patient WHERE patient_id = :NEW.patient_id;

:NEW.blood_type := blood_type1;

FOR rec IN (SELECT stock_id, quantity FROM blood_stock WHERE ((blood_type1 = 'O-' AND blood_type = 'O-') OR (blood_type1 = 'O+' AND blood_type IN ('O-', 'O+')) OR (blood_type1 = 'A-' AND blood_type IN ('A-', 'O-')) OR (blood_type1 = 'A+' AND blood_type IN ('A-', 'A+', 'O-', 'O+')) OR (blood_type1 = 'B-' AND blood_type IN ('B-', 'O-')) OR (blood_type1 = 'B+' AND blood_type IN ('B-', 'B+', 'O-', 'O+')) OR (blood_type1 = 'AB-' AND blood_type IN ('AB-', 'A-', 'B-', 'O-')) OR (blood_type1 = 'AB+' AND blood_type IN ('AB-', 'AB+', 'A-', 'A+', 'B-', 'B+', 'O-', 'O+'))) AND quantity > 0 AND expiry > SYSDATE ORDER BY expiry) LOOP

EXIT WHEN v_required = 0;

```
IF          v_required          >=          rec.quantity          THEN
  v_fulfilled       :=       v_fulfilled       +       rec.quantity;
  v_required       :=       v_required       -       rec.quantity;
  insert                into                   Crossmatch_Junction
values(rec.stock_id,:NEW.Patient_ID);
  DELETE   FROM   blood_stock   WHERE   stock_id   =   rec.stock_id;

ELSE
  UPDATE                                                  blood_stock
  SET       quantity       =       quantity       -       v_required
  WHERE             stock_id              =             rec.stock_id;
  v_fulfilled       :=       v_fulfilled       +       v_required;
  v_required                         :=                             0;
  insert                into                   Crossmatch_Junction
values(rec.stock_id,:NEW.Patient_ID);
END                                                               IF;
```

END LOOP;

IF v_required > 0 THEN

IF v_fulfilled > 0 THEN

:NEW.status := 'Partially Fulfilled';

:NEW.units_assigned := v_fulfilled;

UPDATE patient SET requests = requests - v_fulfilled WHERE patient_id = :NEW.patient_id;

ELSE

:NEW.status := 'Failed';

:NEW.units_assigned := 0;

END IF;

ELSE

:NEW.status := 'Fulfilled';

:NEW.units_assigned := v_fulfilled;

UPDATE patient SET blood_requirement = 'Not Required', requests = 0 WHERE patient_id = :NEW.patient_id;

END IF;

END;

/

trg_patient_blood_stock_link

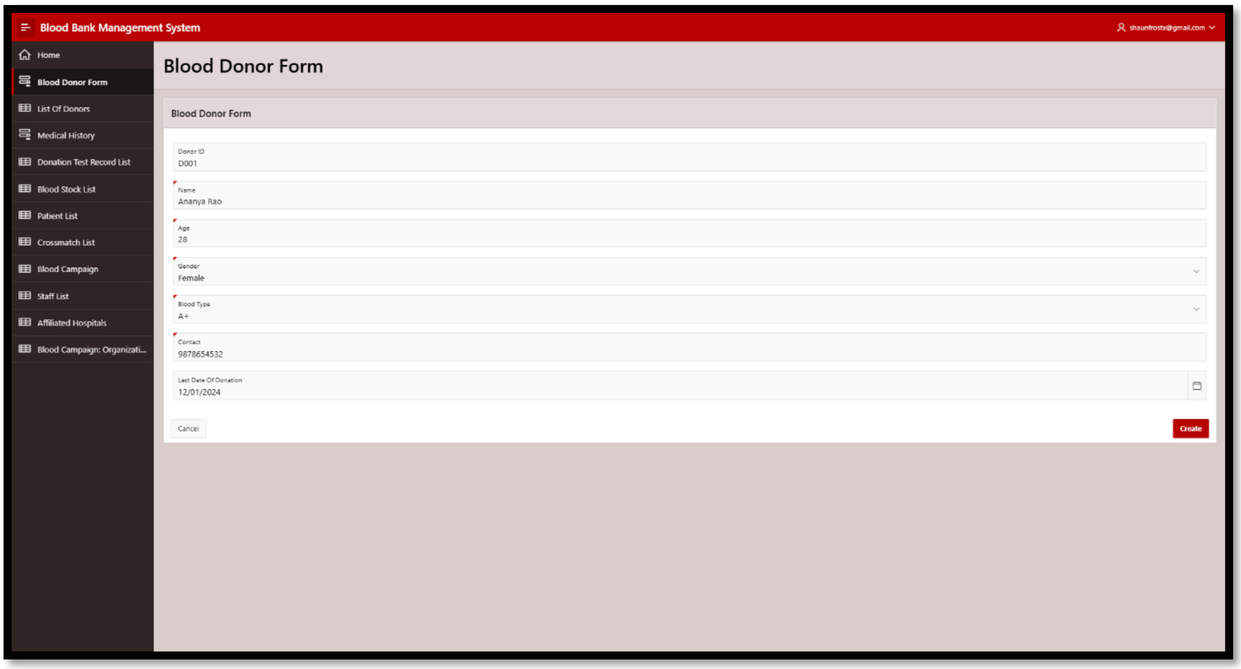
CREATE OR REPLACE TRIGGER AFTER INSERT ON PATIENT FOR EACH ROW DECLARE blood_type1 VARCHAR2(5); BEGIN IF :NEW.BLOOD_REQUIREMENT = 'Required' THEN blood_type1 := :NEW.BLOOD_TYPE;

```
FOR rec IN (
  SELECT stock_id
  FROM blood_stock
  WHERE (
    (blood_type1 = 'O-' AND blood_type = 'O-') OR
    (blood_type1 = 'O+' AND blood_type IN ('O-', 'O+')) OR
    (blood_type1 = 'A-' AND blood_type IN ('A-', 'O-')) OR
    (blood_type1 = 'A+' AND blood_type IN ('A-', 'A+', 'O-', 'O+'))
OR
    (blood_type1 = 'B-' AND blood_type IN ('B-', 'O-')) OR
    (blood_type1 = 'B+' AND blood_type IN ('B-', 'B+', 'O-', 'O+'))
OR
    (blood_type1 = 'AB-' AND blood_type IN ('AB-', 'A-', 'B-', 'O-
')) OR
    (blood_type1 = 'AB+' AND blood_type IN ('AB-', 'AB+', 'A-',
'A+', 'B-', 'B+', 'O-', 'O+'))
  )
  AND quantity > 0
  AND expiry > SYSDATE
) LOOP
  INSERT INTO CROSSMATCH_JUNCTION (STOCK_ID, PATIENT_ID)
  VALUES (rec.stock_id, :NEW.PATIENT_ID);
END LOOP;
```

END IF; END; /

6. Result

BLOOD DONOR FORM



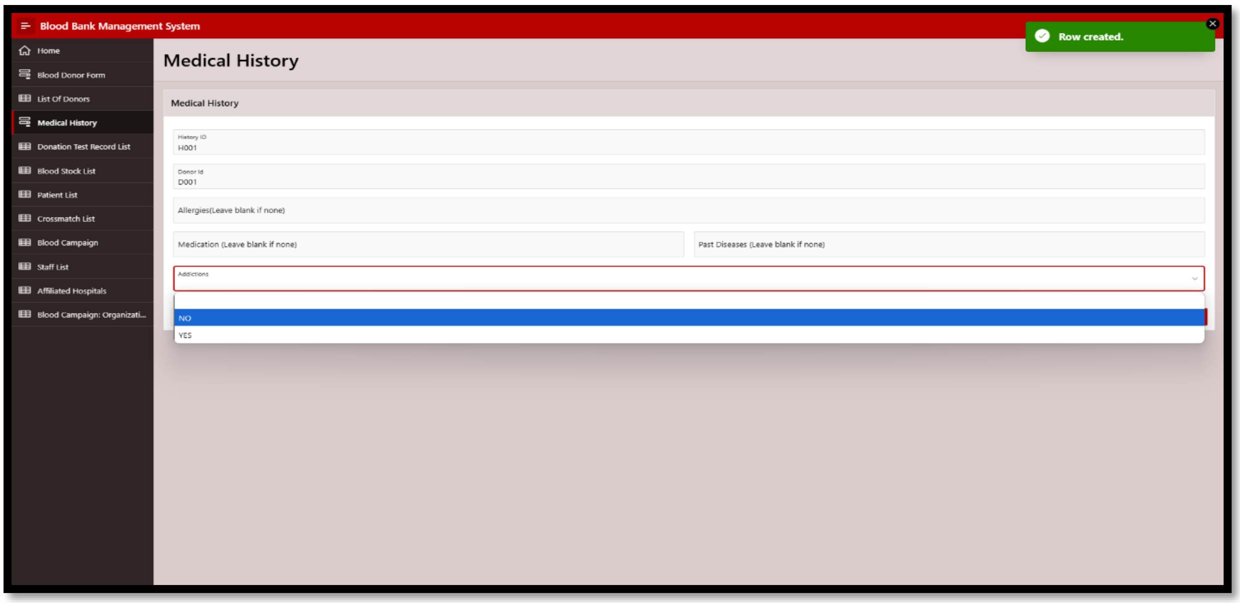
Blood Donor Form

Blood Donor Form

Donor ID	D001
Name	Ananya Rao
Age	28
Gender	Female
Blood Type	A+
Contact	9878654532
Last Date Of Donation	12/01/2024

This form requests the user to fill in the blood donor details. Upon submission, the information is inserted into the Donor table.

MEDICAL_HISTORY_RECORD



Row created.

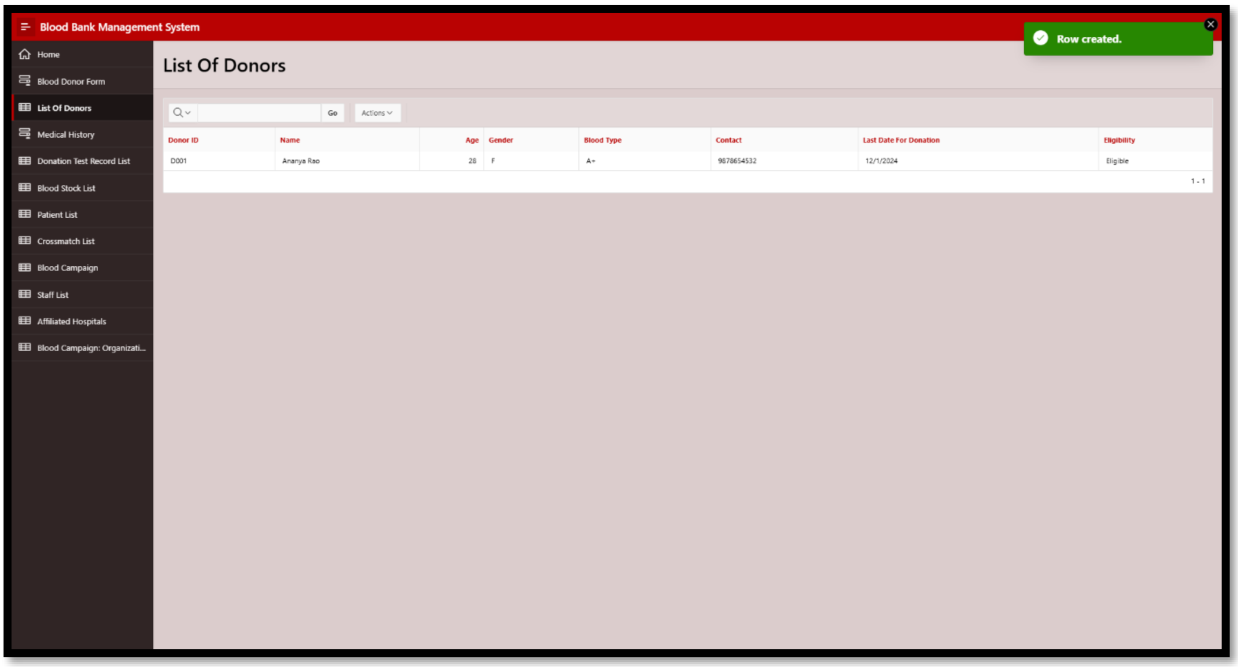
Medical History

Medical History

History ID	H001
Donor Id	D001
Allergies(Leave blank if none)	
Medication (Leave blank if none)	
Past Diseases (Leave blank if none)	
Addictions	

After submitting the Donor Form the user is redirected to the Medical History where he is prompted to enter some details on the basis of which the eligibility in the Donor form is updated.

DONOR LIST



Blood Bank Management System

Row created.

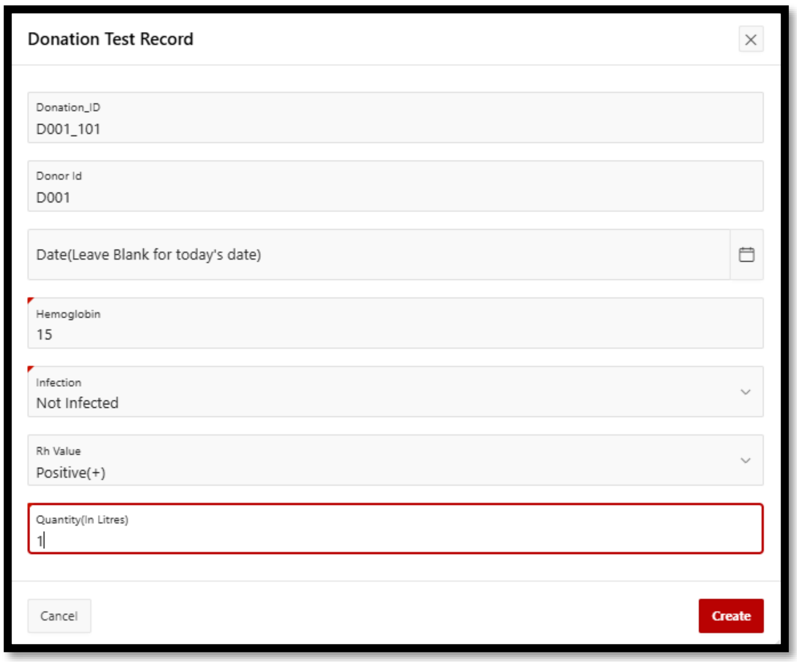
List Of Donors

Donor ID	Name	Age	Gender	Blood Type	Contact	Last Date For Donation	Eligibility
D001	Ananya Rao	28	F	A+	9878654532	12/1/2024	Eligible

1 - 1

After submitting the medical history form the page is redirected to list of donors showing whether the donor is eligible or not on the basis of their entered details.

DONATION_TEST_RECORD

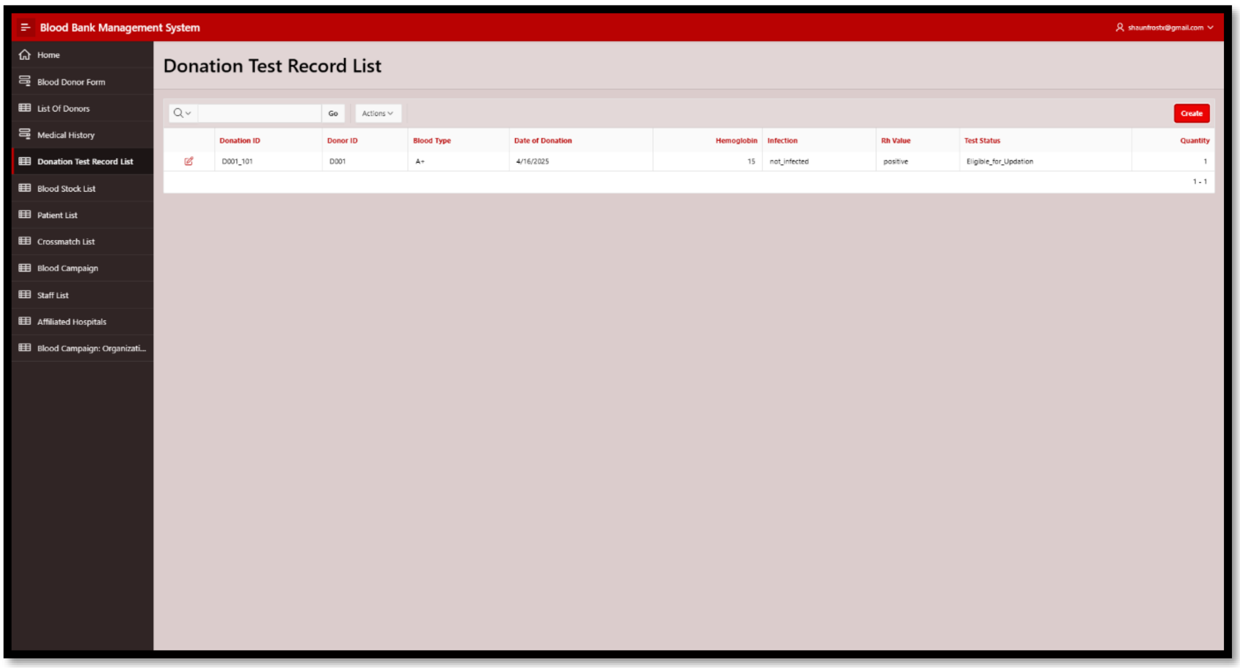


Donation Test Record

Donation_ID	D001_101
Donor Id	D001
Date(Leave Blank for today's date)	
Hemoglobin	15
Infection	Not Infected
Rh Value	Positive(+)
Quantity(In Litres)	1

On opening the Donation Test Record we can create a new record by filling in the details for donation test record which updates the eligibility status for updation.

DONATION_TEST_RECORD_LIST



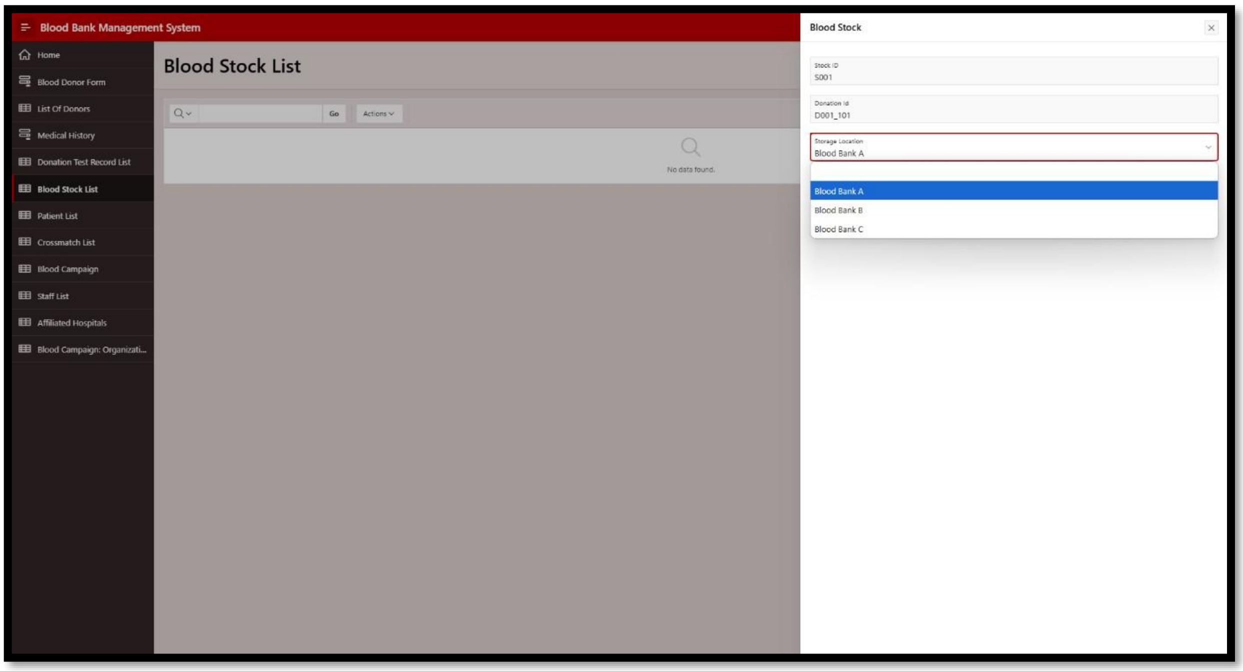
Donation Test Record List

	Donation ID	Donor ID	Blood Type	Date of Donation	Hemoglobin	Infection	Rh Value	Test Status	Quantity
	D001_101	D001	A+	4/16/2025	15	not_infected	positive	Eligible_for_Updation	1

1 - 1

After adding the donation test record the test status is updated on the basis of which it is decided whether it will be used in blood stock or not.

BLOOD_STOCK FORM



Blood Stock List

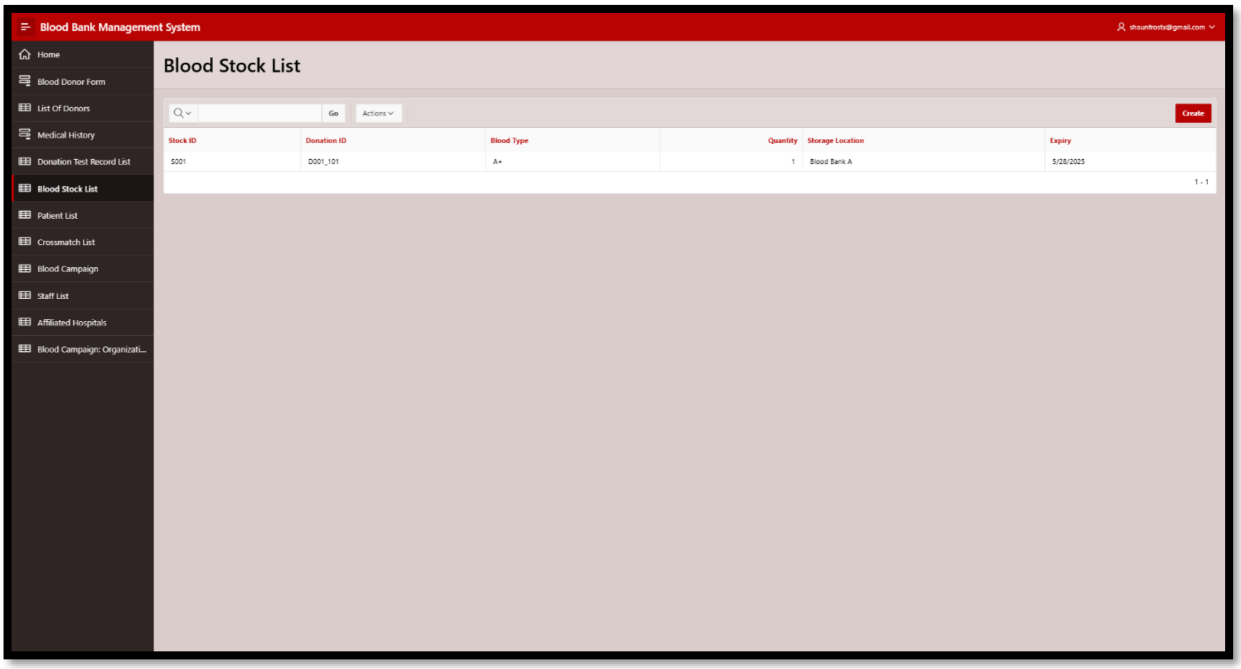
Blood Stock

Stock ID	S001
Donation Id	D001_101
Storage Location	Blood Bank A

Blood Bank A
Blood Bank B
Blood Bank C

Filling in the stock id,Donation id and selecting the storage location inserts the values into the blood stock table.

BLOOD_STOCK_LIST



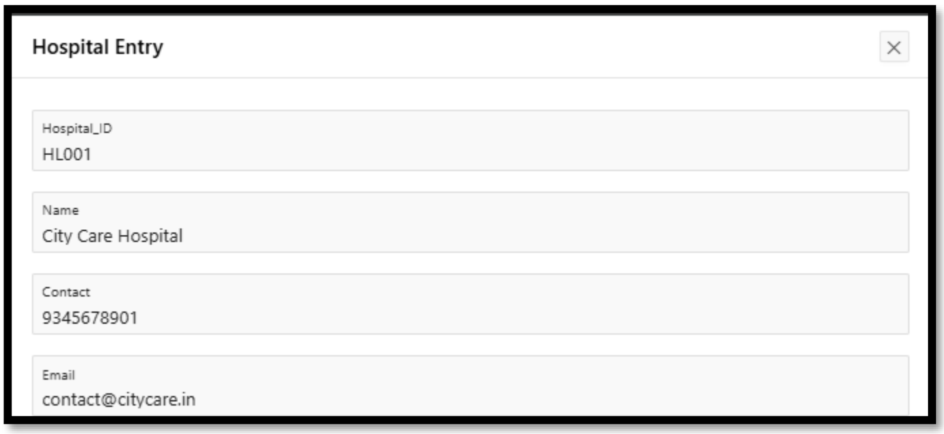
Blood Stock List

Stock ID	Donation ID	Blood Type	Quantity	Storage Location	Expiry
S001	D001_101	A+	1	Blood Bank A	5/28/2025

1 - 1

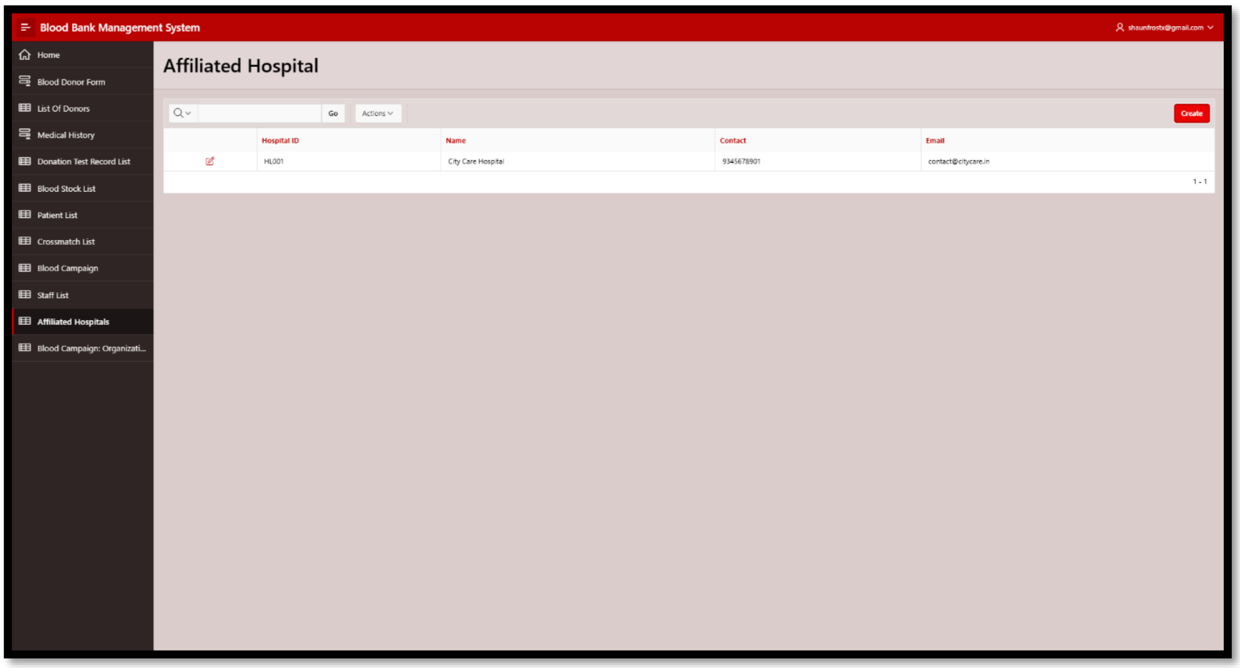
The details like blood type,quantity are updated from donation test record and the expiry date is calculated by adding 6 weeks to the date of donation.

<u>HOSPITAL FORM & LIST</u>


Hospital Entry

Hospital_ID	HL001
Name	City Care Hospital
Contact	9345678901
Email	contact@citycare.in

Form to fill in the hospital details.



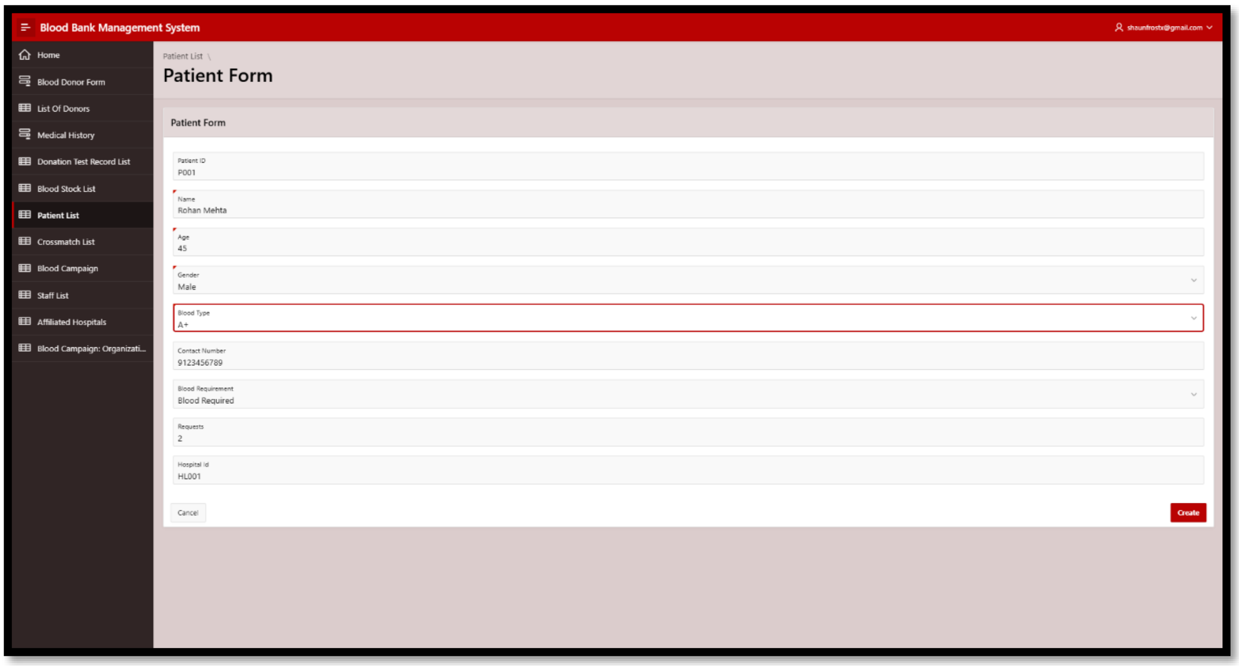
Affiliated Hospital

Hospital ID	Name	Contact	Email
HL001	City Care Hospital	9345678901	contact@citycare.in

1 - 1

Hospital List after insertion of values.

PATIENT FORM



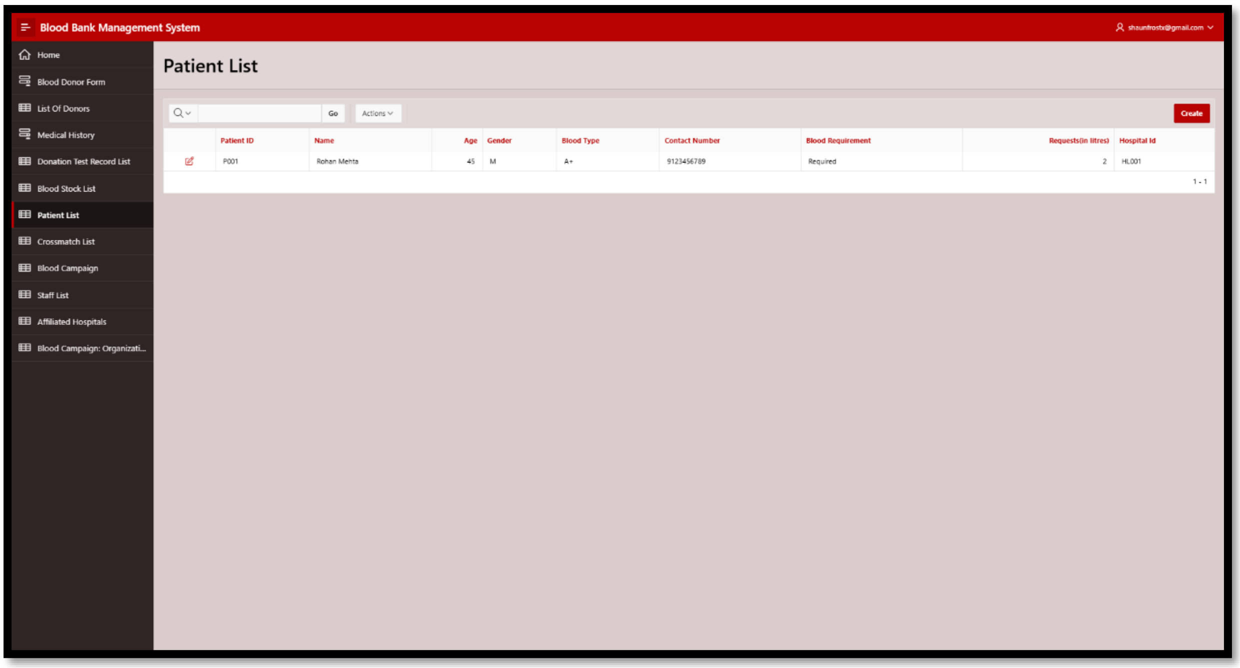
Patient Form

Patient Form

Patient ID	P001
Name	Rohan Mehta
Age	45
Gender	Male
Blood Type	A+
Contact Number	9123456789
Blood Requirement	Blood Required
Requests	2
Hospital Id	HL001

Patient Form to fill in the details of the patient along with hospital id that references hospital.

PATIENT LIST



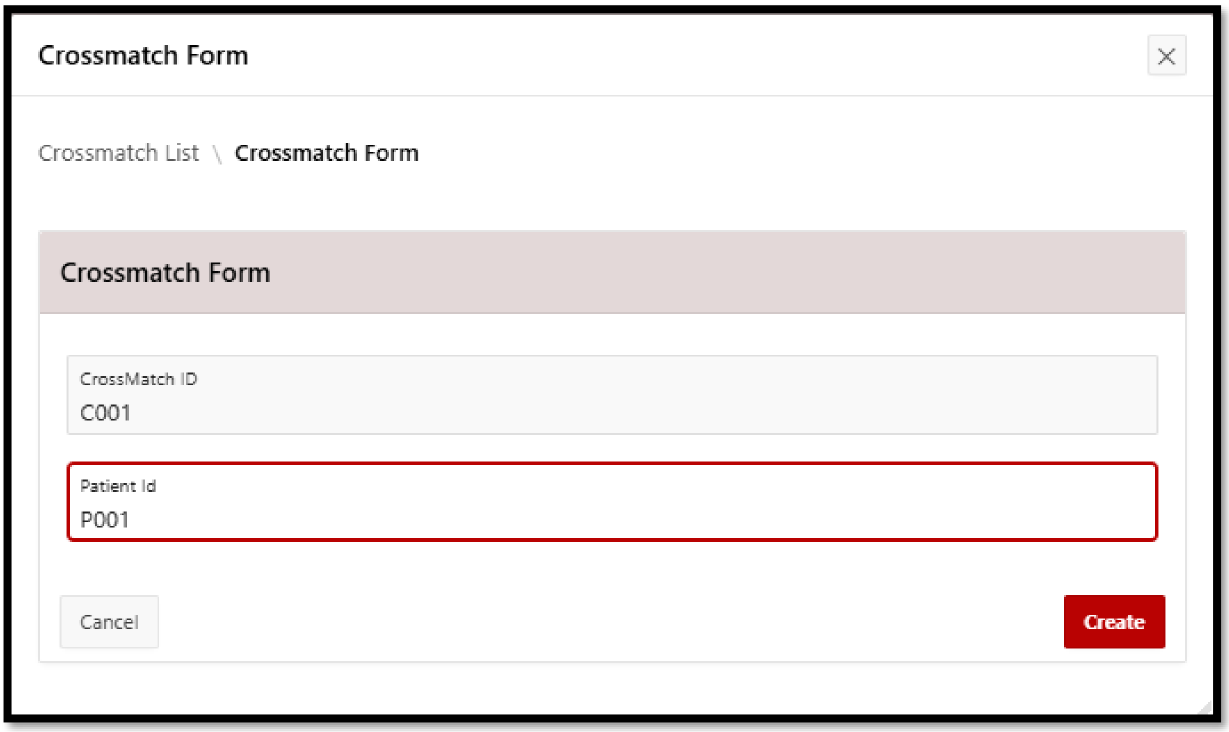
Patient List

Patient ID	Name	Age	Gender	Blood Type	Contact Number	Blood Requirement	Requests(in litres)	Hospital Id
P001	Rohan Mehta	45	M	A+	9123456789	Required	2	HL001

1 - 1

Updated list is as shown along with requirement and quantity requested.

CROSSMATCH_FORM



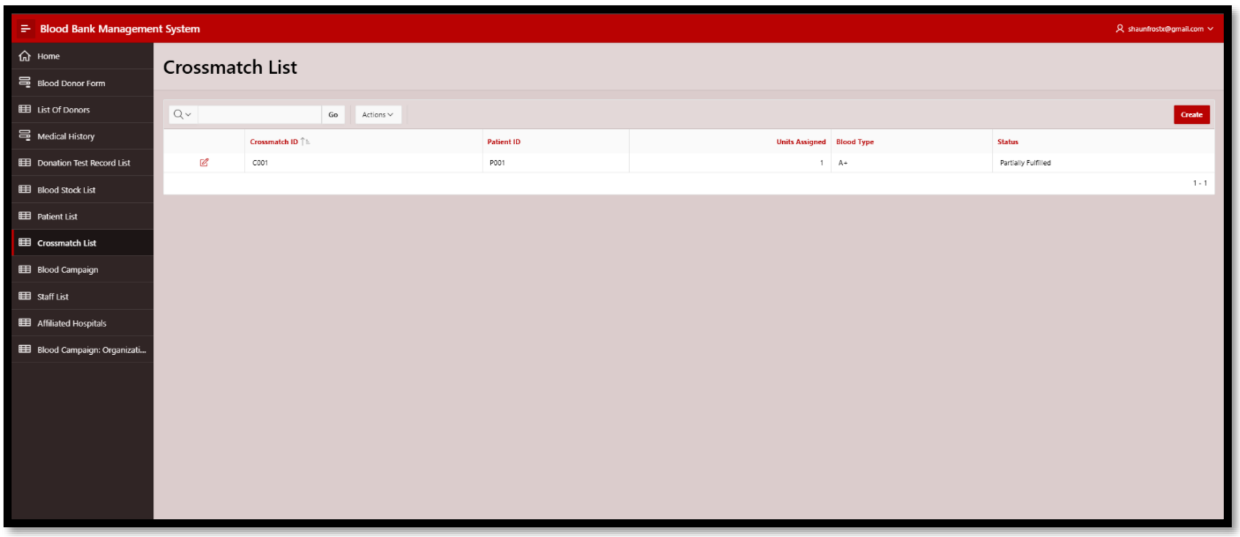
Crossmatch Form

CrossMatch ID	C001
Patient Id	P001

To fulfill the requests, we fill out the Crossmatch form with the Crossmatch ID and Patient ID. Upon clicking 'Create', the trigger Crossmatch_Form_T is fired. This trigger sorts the blood stock based on expiry date and compatible blood type. The stock with the nearest expiry date is used first to fulfill the request. If the request is not fully satisfied, the trigger moves to the next available record, continuing until the request is fulfilled.

If only a part of the required amount is fulfilled, the status is updated to ‘Partially Fulfilled’. If the full amount is provided, the status becomes ‘Fulfilled’. If no units are assigned, the status is marked as ‘Failed’. Simultaneously, the Blood Stock table and the Patient table are also updated based on the outcome.

CROSSMATCH_LIST



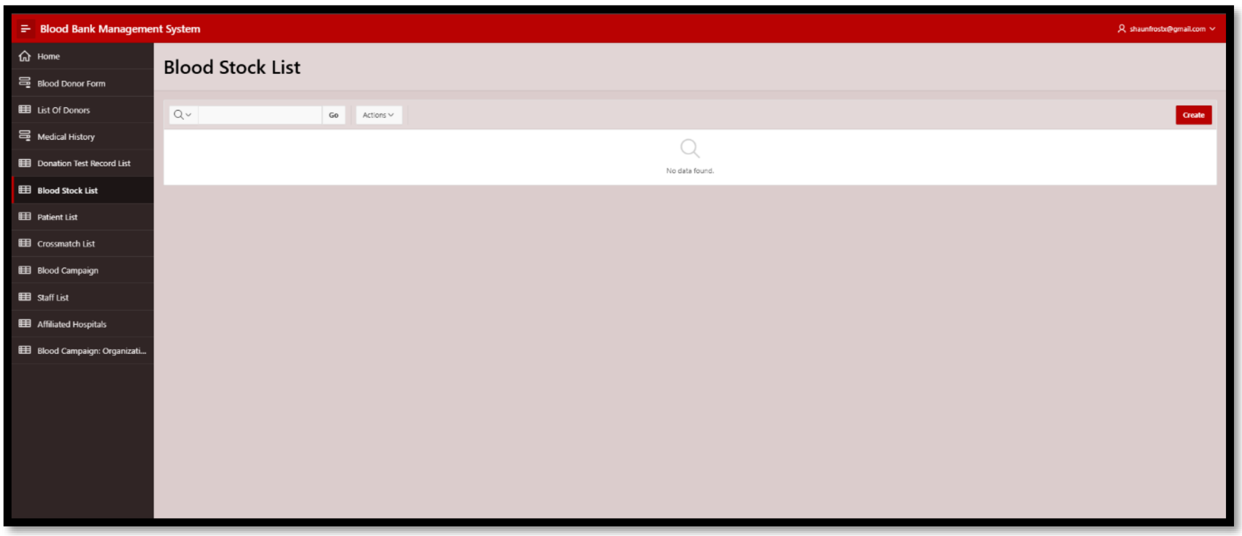
Crossmatch List

Crossmatch ID	Patient ID	Units Assigned	Blood Type	Status
C001	P001	1	A+	Partially Fulfilled

1 - 1

Since only 1 litre of A+ was available in the blood stock and the required was 2, the status was updated to partially fulfilled.

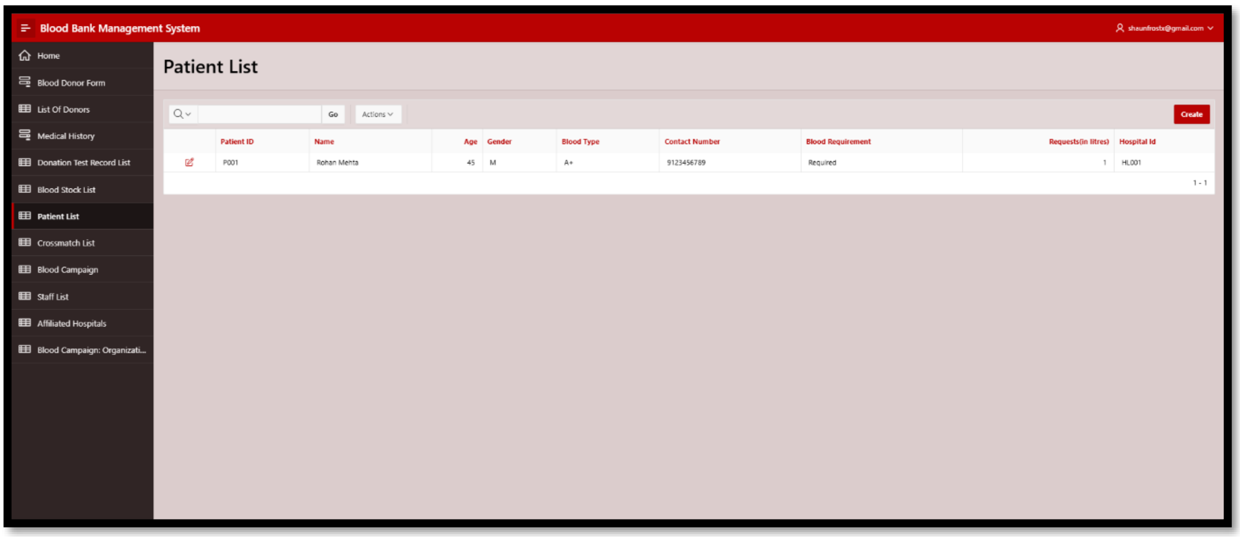
UPDATION



Blood Stock List

No data found.

Simultaneously the blood stock is updated, since the whole stock was used the record was deleted.



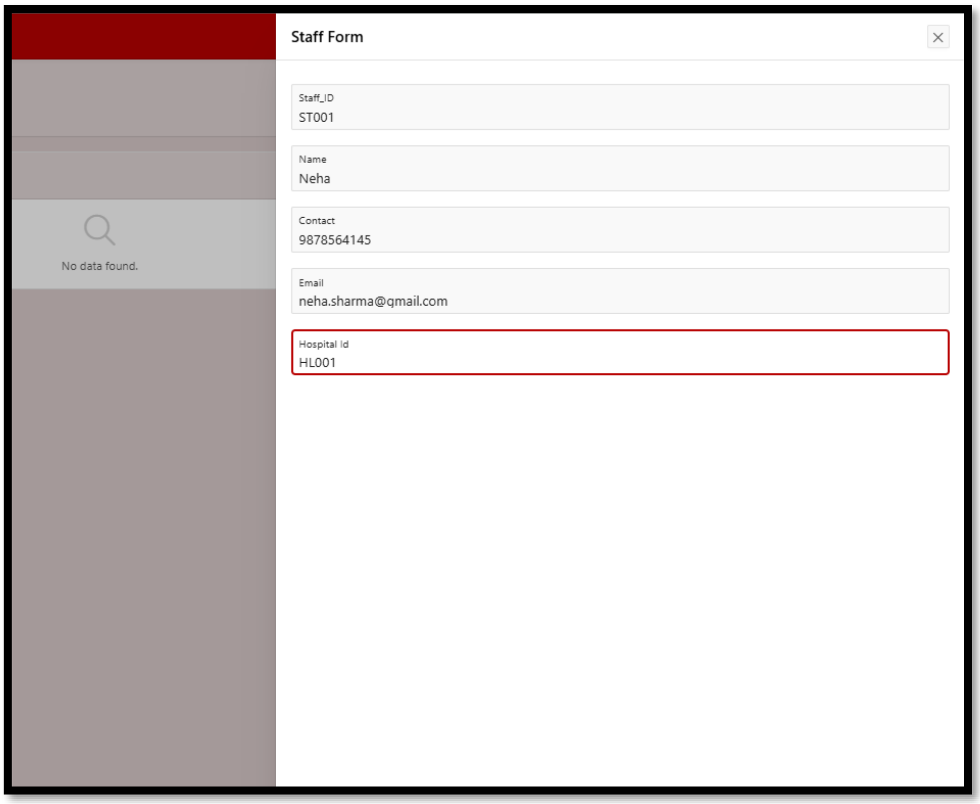
Patient List

Patient ID	Name	Age	Gender	Blood Type	Contact Number	Blood Requirement	Requests(in litres)	Hospital Id
P001	Rohan Mehta	45	M	A+	9123456789	Required	1	HL001

1 - 1

Also, the requested amount of the patient gets updated, if the request had been fulfilled then the blood_required would have been set to Not Required.

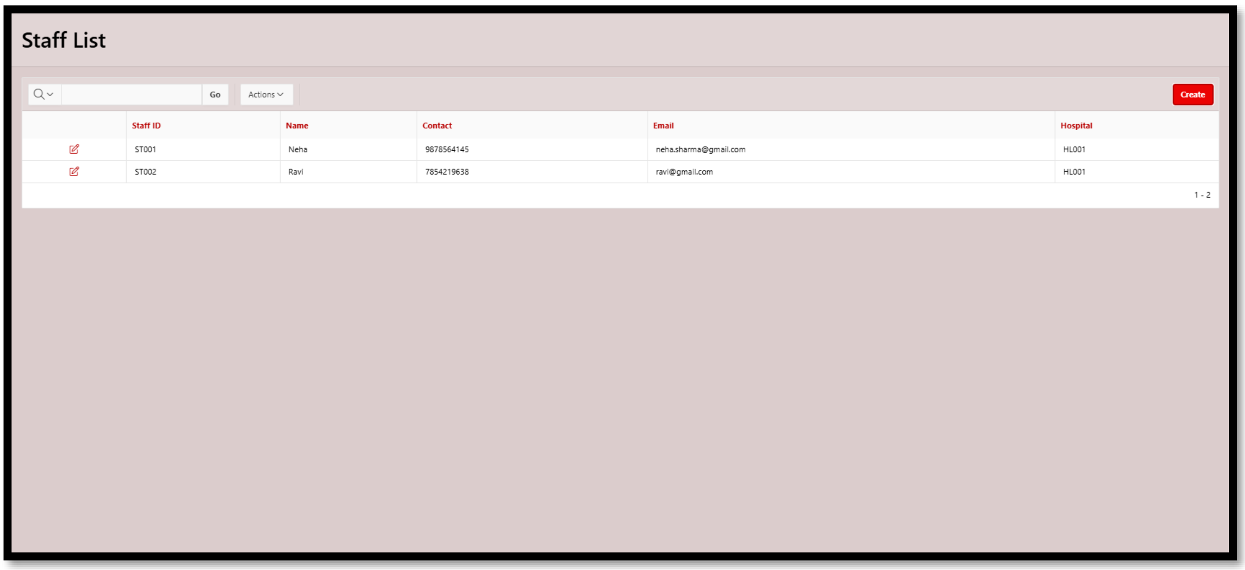
STAFF_FORM



Staff Form

Staff_ID	ST001
Name	Neha
Contact	9878564145
Email	neha.sharma@gmail.com
Hospital Id	HL001

Form to fill in the details for the staff.

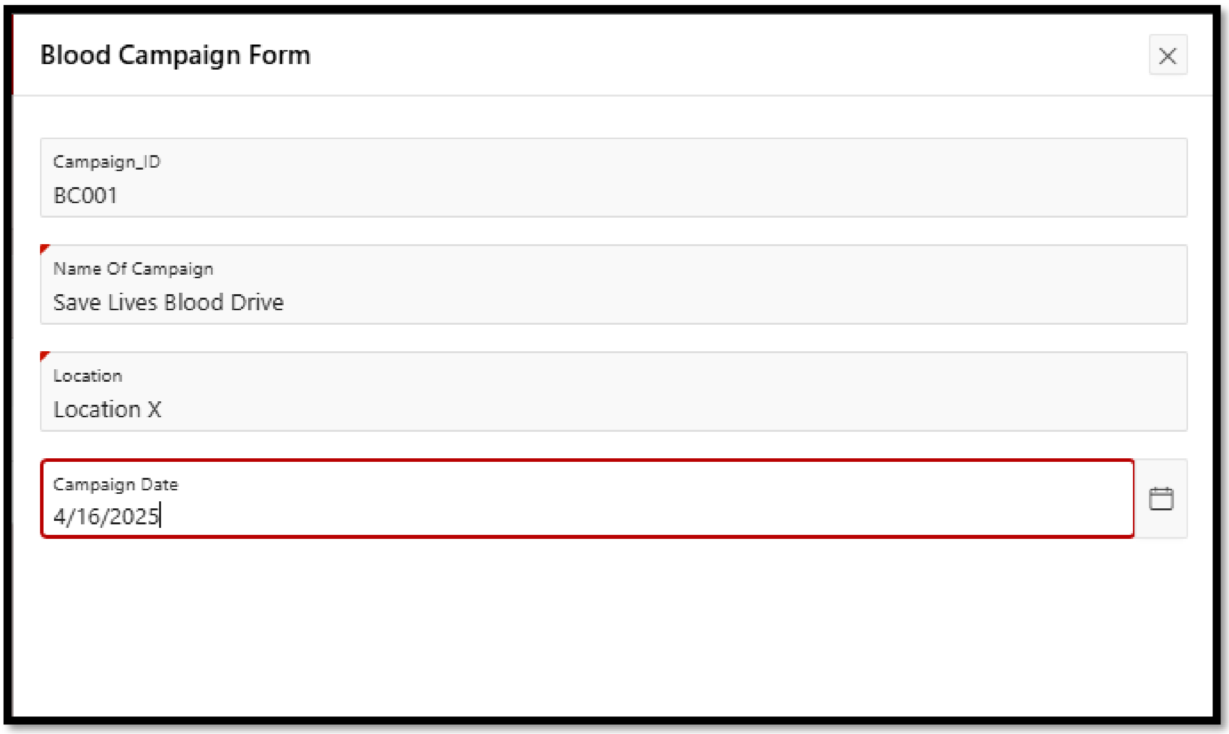

Staff List

	Staff ID	Name	Contact	Email	Hospital
	ST001	Neha	9878564145	neha.sharma@gmail.com	HL001
	ST002	Ravi	7854219638	ravi@gmail.com	HL001

1 - 2

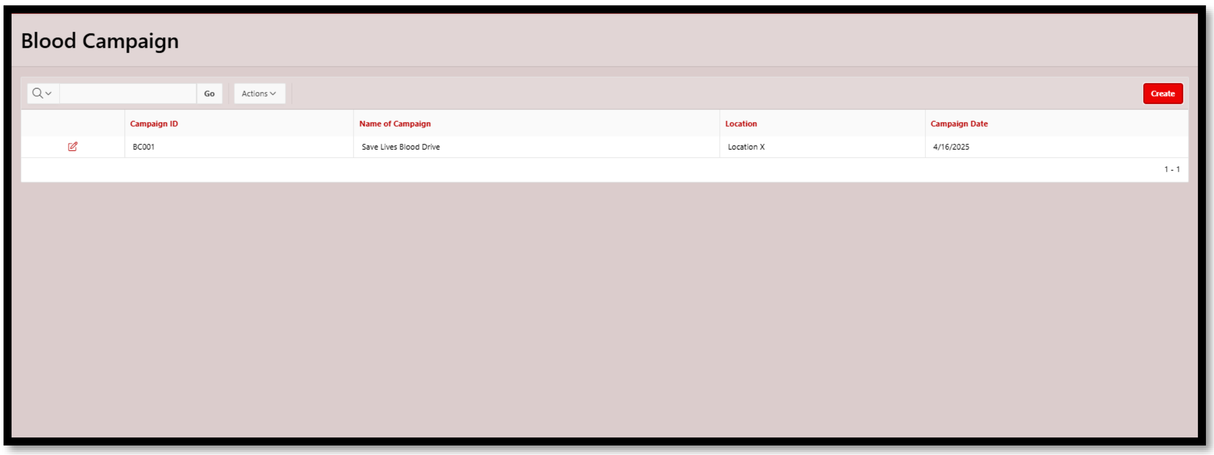
Updated list after insertion.

BLOOD_CAMPAIGN_FORM


Blood Campaign Form

Campaign_ID	BC001
Name Of Campaign	Save Lives Blood Drive
Location	Location X
Campaign Date	4/16/2025

To fill details for Blood Campaign

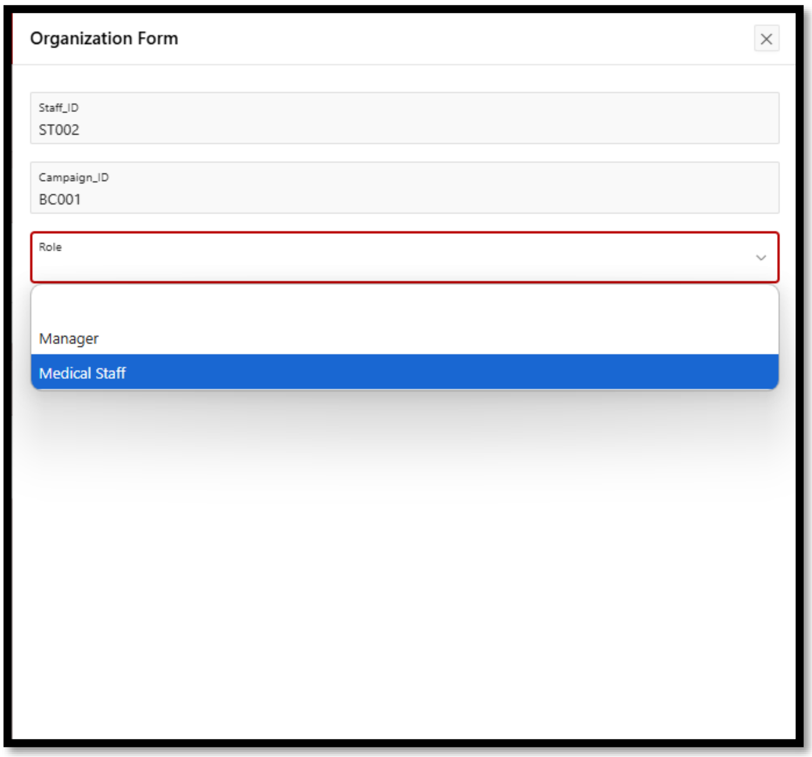

Blood Campaign

	Campaign ID	Name of Campaign	Location	Campaign Date
	BC001	Save Lives Blood Drive	Location X	4/16/2025

1 - 1

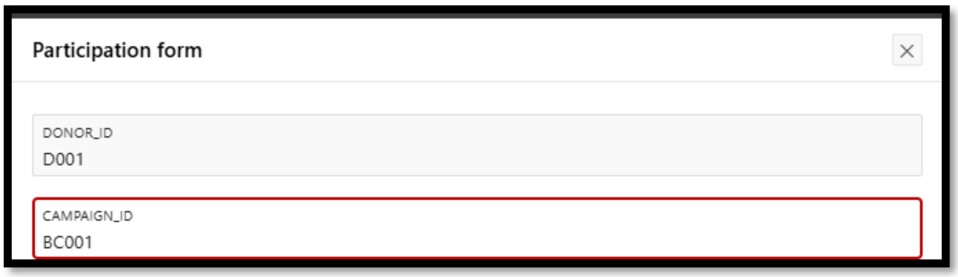
Updated list for blood campaign

<u>ORGANIZES & PARTICIPATION LIST</u>


Organization Form

Staff_ID	ST002
Campaign_ID	BC001
Role	

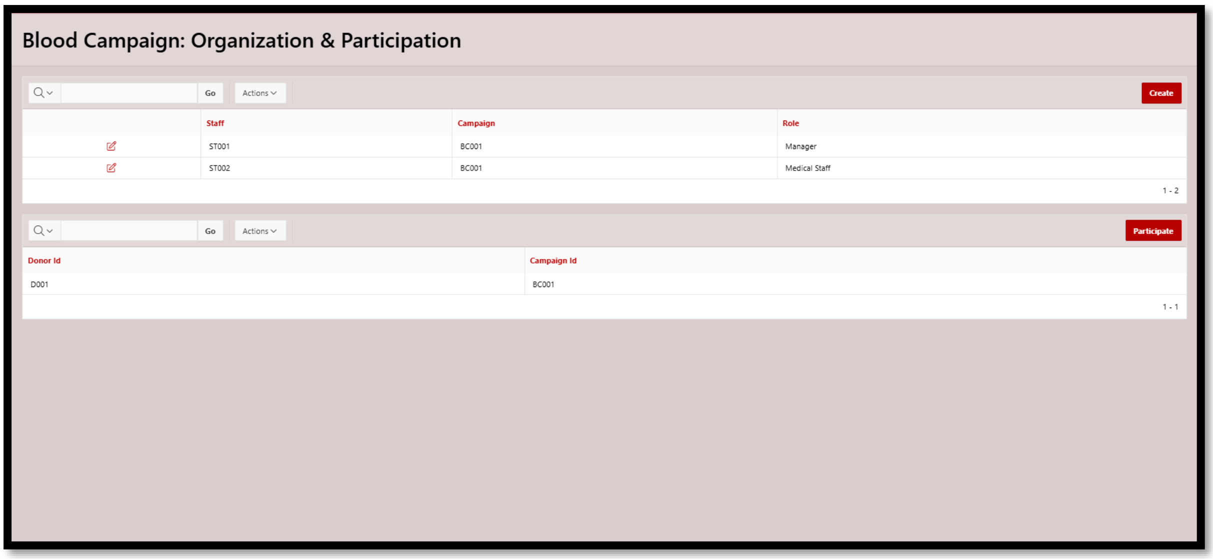
Form to fill the details of the blood campaign orgranizers.



Participation form

DONOR_ID	D001
CAMPAIGN_ID	BC001

Form to fill in the details of the participating donors.



Blood Campaign: Organization & Participation

Staff	Campaign	Role
ST001	BC001	Manager
ST002	BC001	Medical Staff

1 - 2

Donor Id	Campaign Id
D001	BC001

1 - 1

Updated List after Insertion.

7. Conclusion and Future Work

7.1 Conclusion

The Blood Bank Management System is a structured and reliable solution developed to overcome the inefficiencies of manual blood bank management. By utilizing **Oracle SQL** for backend data handling and **Oracle APEX** for creating an intuitive user interface, the system effectively automates and streamlines core operations such as donor registration, blood inventory tracking, patient details management, and crossmatching procedures.

With the use of **Oracle APEX triggers and SQL queries**, the system ensures real-time updates and accurate data management, minimizing human errors and improving overall efficiency. The centralized structure allows for quick access to essential information, helping staff respond promptly in critical situations. This system not only simplifies routine tasks but also contributes to the smooth and safe functioning of blood bank services, ultimately supporting better patient care and resource utilization.

7.2 Scope for future work

The Blood Bank Management System, while effective in automating core functionalities, offers significant opportunities for future enhancement and expansion. In the future, features such as **role-based access control** can be implemented to restrict functionalities based on user responsibilities, enhancing security and data integrity. **Automated notifications and alerts** can also be integrated to inform staff about low stock levels, upcoming donation eligibility, or expiry of blood units, enabling timely action.

Each Team Member Contribution:

Team Member	Registration number	Contribution
Vedant Ganesh	230968340	Worked on ER/Schema diagram, Backend and the Report
Upashana N.Bhojake	230968354	Worked on Abstract, Backend and the Report
Joyal	230968362	Worked on ER/Schema diagram and Backend
Manish R	230968364	Worked on ER/Schema diagram and Backend
Satwik Jain	230968378	Worked on ER/Schema diagram, Frontend/integration, Backend and the Report
Vanshika Mittal	230968382	Worked on Abstract, ER/Schema diagram, Backend and the Report
Himanshu B.Agarwal	230968390	Worked on ER/Schema diagram, Backend and the Report
Trisal Palla	230968394	Worked on ER/Schema diagram, Backend and the Report