Jawaharlal Nehru University, New Delhi.

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES

ACADEMIC YEAR 2022-24

DATABASE MANAGEMENT SYSTEM

Project Name:-

BLOOD DONATION MANAGEMENT SYSTEM

GROUP MEMBERS	ENROLLMENT NO.
ABHINAV OLI	22/10/JC/062
VASTVIKTA NISHAD	22/10/JC/050
MOHD. SAIF	22/10/JC/028
PIYUSH PAWAR	22/10/JC/011

COURSE - MCA

SEMESTER - 2

SUBMITTED TO-

Dr. T.V. VIJAY KUMAR

INTRODUCTION TO PROJECT

The Blood Donation Management System is a comprehensive database management project designed to streamline the process of blood donation and enhance communication between blood banks, hospitals, donors, recording staff, and recipients. This system aims to improve efficiency and ensure the availability of blood units for patients in need.

The key entities in this system include the Blood Bank Manager, Hospital, Blood Donor, Recording Staff, Blood Specimen and Recipient. The Blood Bank Manager plays a pivotal role in initiating the process by placing orders for blood units required by hospitals. The Hospital, located within a specific city, relies on the Blood Bank Manager's orders to meet the demand for blood within their facility.

To facilitate the blood supply chain, the system also incorporates the Blood Donor entity. Donors, residing in the same city as the hospital, are registered in the database. These individuals voluntarily provide blood donations, which are crucial for maintaining an adequate supply for emergencies and medical procedures.

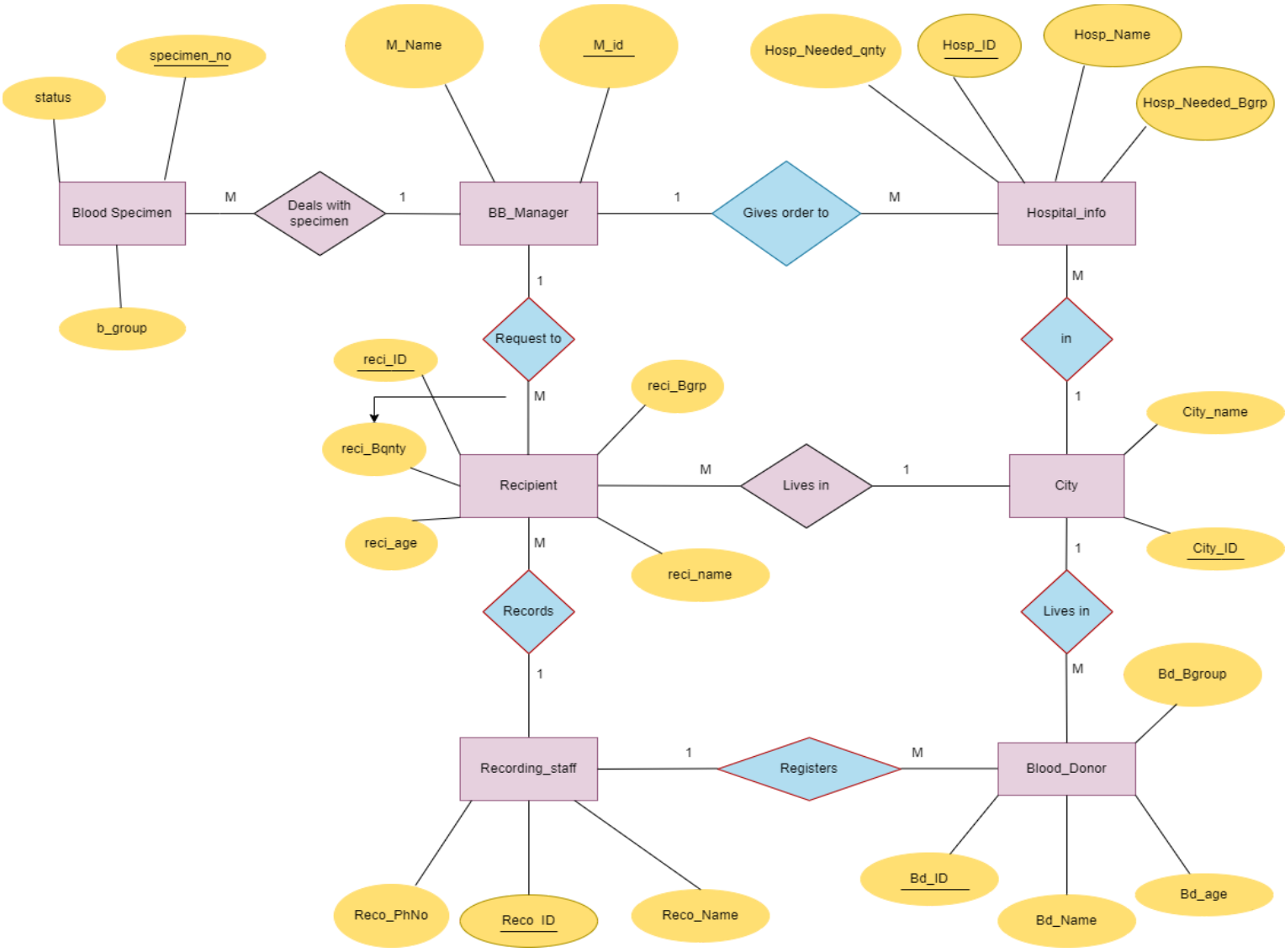
Recording Staff members are responsible for registering the details of blood donors, ensuring accurate records are maintained. They record essential information such as the donor's personal details, blood type, and medical history. This data serves as a valuable resource when matching compatible donors with recipients and maintaining a database of potential donors.

On the recipient's side, when a patient requires blood, the Recording Staff records their information in the system. This includes the recipient's personal details, required blood type, and other relevant medical information. The recipient then submits a request to the Blood Bank Manager, who coordinates with the hospital to fulfill the request promptly.

The Blood Donation Management System bridges the gap between blood donors, hospitals, and the blood bank. By maintaining an organized and accessible database, it ensures the availability of blood units, tracks blood inventory, and facilitates the timely delivery of blood to those in need.

The purpose of this project is to create a robust and user-friendly database management system that enhances the efficiency of blood donation processes, improves communication, and ultimately saves lives. By implementing this system, blood banks, hospitals, donors, recording staff, and recipients can work collaboratively to create a more effective blood donation ecosystem.

ENTITY RELATIONSHIP DIAGRAM



INFORMATION OF ENTITIES

In total we have seven entities and information of each entity is mentioned below-

1. Blood_Donor:

Attributes : (Bd_ID, Bd_Name, Bd_Bgroup,Bd_age)

The donor is the person who **donates blood**, on donation a donor id (Bd_ID) is generated and used as the primary key to identify the donor information. Other than that name, age and blood group will be stored in a database under Blood_Donor entity.

2. Recipient:

Attributes :(reci_ID, reci_name, reci_age, reci_Bgrp, reci_Bqnty)

The Recipient is the person who **receives blood** from a blood bank, when blood is given to a recipient a recipient ID (reci_ID) is generated and used as the primary key for the recipient entity to identify blood recipients information. Along with its name ,age, blood group (needed), and blood quantity(needed) are also stored in the database under the Recipient entity.

3. BB_Manager:

Attributes : (M_id, M_Name)The blood bank manager is the person who **takes care of the available blood samples in the blood bank, he is also responsible for handling blood requests from recipients and hospitals**. Blood manager has a unique identification

number (M_ID) used as primary key along with name of blood bank manager will be stored in the database under BB_Manager entity.

4. Recording_Staff :

Attributes :(Reco_ID, Reco_Name, Reco_PhNo)

The recording staff is a person who **registers the blood donor and recipients** and the Recording_Staff entity has Reco_ID which is the primary key along with recorder's name and recorder's phone number will also be stored in the database under Recording_Staff entity.

5. Hospital_Info :

Attributes : (Hosp_ID, Hosp_Name, Hosp_needed_Bgrp, hosp_needed_qnty)

In the database, **under the Hospital_Info entity we will store the information of hospitals.** In this hosp_ID and hosp_needed_Bgrp together makes the primary key. We will store the hospital name and the blood quantity required at the hospital.

6. City:

Attributes :(City_ID, City_name)

This entity will store the information of cities where donors, recipients and hospitals are present. A unique identification number (City_ID) will be used as the primary key to identify the information about the city. Along with ID city names will also be stored under this entity.

7. BloodSpecimen :

(Attributes – specimen_number, b_group , status)

In the database, under the Blood Specimen entity we will store the information of blood samples which are available in the blood bank. In this entity specimen_number and b_group together will be primary key along with status attribute which will show if the blood is contaminated on not.

RELATIONSHIP BETWEEN ENTITIES

1. **City and Hospital_Info:**

   Relationship = “in”

   Type of relation = 1 to many

   Explanation = A city can have many hospitals in it. One hospital will belong in one city.

2. **City and Blood_Donor:**

   Relationship = “lives in”

   Type of relation = 1 to many

   Explanation = In a city, many donors can live. One donor will belong to one city.

3. **City and Recipient:**

   Relationship = “lives in”

   Type of relation = 1 to many

   Explanation = In a city, many recipients can live. One recipient will belong to one city.

4. **Recording_Staff and Donor:**

   Relationship = “registers”

   Type of relation = 1 to many

   Explanation = One recording staff can register many donors. One donor will register with one recording officer.

5. **Recording_Staff and Recipient:**

   Relationship = “records”

   Type of relation = 1 to many

   Explanation = One recording staff can record many recipients. One recipient will be recorded by one recording officer.

6. **Hospital_Info and BB_Manager:**

   Relationship = “gives order to”

   Type of relation = 1 to many

   Explanation = One Blood bank manager can handle and process requests from many hospitals. One hospital will place a request on the blood bank manager.

7. **BB_Manager and Blood Specimen**:

   Relationship = “deals with specimen”

   Type of relation = 1 to many

   Explanation = One Blood bank manager can manage many blood specimens and one specimen will be managed by one manager.

8. **Recipient and BB_Manager:**

Relationship = “requests to”

Type of relation = 1 to many

Explanation = One recipient can request blood to one manager and one manager can handle requests from many recipients.

RELATIONAL SCHEMAS

1. **Donor Table:**

• The relationship with Recording staff and Donor is 1 to many. That’s why the primary key of Recording staff is used as a foreign key in Donor.

• The relationship with City and Donor is 1 to many. That’s why the primary key of City is used as a foreign key in Donor.

2. **Recipient Table:**

• The relationship with Recording staff and Blood Recipient is 1 to many. That’s why the primary key of Recording staff is used as a foreign key in Blood Recipient.

• The relationship with City and Blood Recipient is 1 to many. That’s why the primary key of City is used as a foreign key in Blood Recipient.

• The relationship with Blood Bank Manager and Blood Recipient is 1 to many. That’s why the primary key of Blood Bank Manage is used as a foreign key in Blood Recipient.

3. City Table:

• The relationship between City and Recipients, Donor,Hospital info are all of 1 to many. So that's why the primary key of City is used as a foreign key in Recipients, Donor and Hospital info.

4. Recording Staff Table:

• The relationship between Recording Staff and Blood Donor, Recipients are all of 1 to many. That's why the primary key of Recording staff is used as a foreign key in Donor and Recipient.

5. Blood Bank Manager Table:

• The relationship between Blood Bank Manager and Blood Specimen, Recipient, Hospital info are all of 1 to many. So therefore, the primary key of Blood Bank Manager is used as a foreign key in Blood Specimen, Recipient and Hospital info.

6. Hospital info Table:

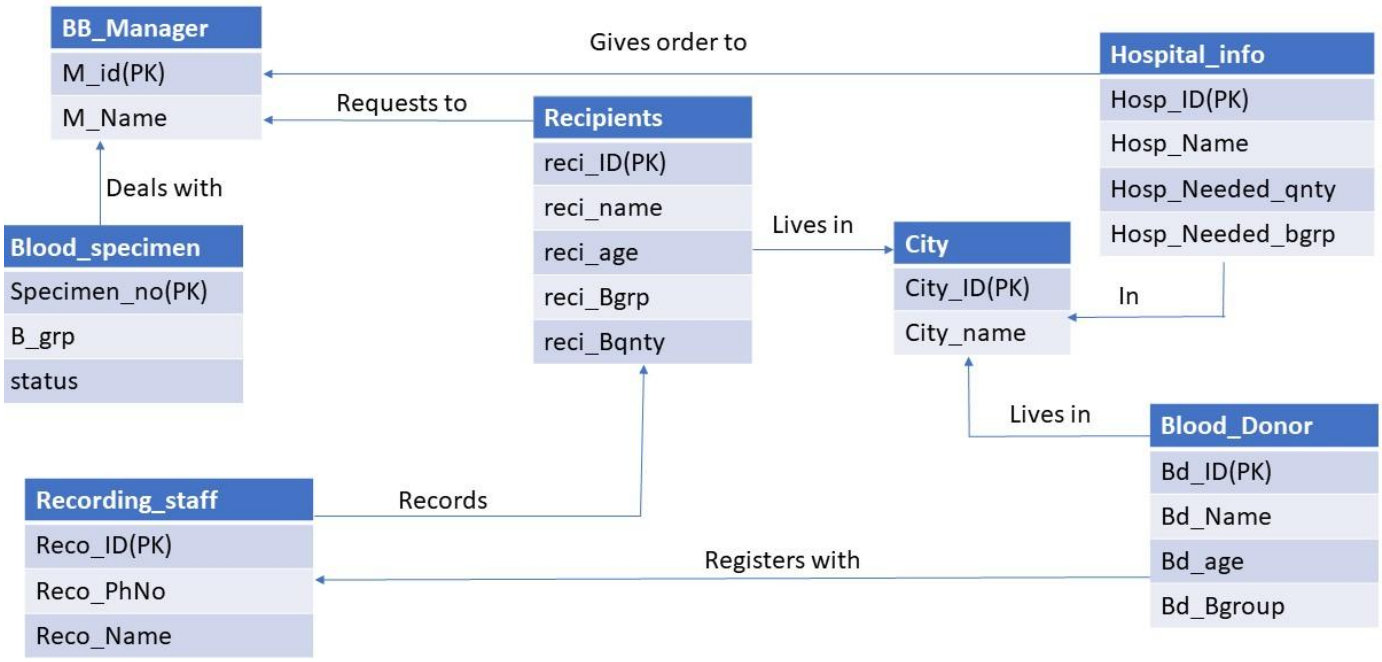
• The relationship with City and Hospital info is 1 to many. That's why the primary key of City is used as a foreign key in Hospital info.

• The relationship with Blood Bank Manager and Hospital info is 1 to many. That's why the primary key of the Blood Bank manager is used as a foreign key in Hospital info.

7. Blood Specimen Table:

The relationship with Blood Bank manager and Blood Specimen is 1 to many. That's why the primary key of Blood Bank manager is used as a foreign key in Blood Specimen.

RELATIONAL SCHEMA



NORMALIZATION RULE

Normalization rules are divided into the following normal forms:

1. First Normal Form

2. Second Normal Form

3. Third Normal Form

4. BCNF Normal Form

★ FIRST NORMAL FORM (1NF):

For a table to be in the First Normal Form, it should follow the following 4 rules:

1. It should only have single (atomic) valued attributes/columns.

2. Values stored in a column should be of the same domain.

3. All the columns in a table should have unique names.

4. And the order in which data is stored, does not matter.

★ SECOND NORMAL FORM (2NF):

For a table to be in the Second Normal Form,

1. It should be in the First Normal form.

2. And, it should not have Partial Dependency.

★ THIRD NORMAL FORM (3NF) :

A table is said to be in the Third Normal Form when,

1. It is in the Second Normal form.

2. And, it doesn't have Transitive Dependency.

★ BOYCE - CODD NORMAL FORM (BCNF) :

1. Every determinant(left side of the functional dependency) must be a candidate key.
2. There should be no non-Trivial functional dependencies where the determinant is a proper subset of a candidate key.

NORMALIZATION OF BLOOD DONOR DATABASE

1. **Blood_Donor** (Bd_ID,Bd_Name,Bd_age,Bd_Bgroup,City_ID,Reco_ID)
   a) {Bd_Id} = > {Bd_Name,Bd_Bgroup,Bd_age,Reco_ID,City_ID}
   b) {Bd_Name} => {Bd_Bgroup,Bd_age}

   The table is in its first normal form.

   The table is in its second normal form.

   The table is not in its third normal form.

   There should be no Transitive dependencies between non key attributes.To remove transitive dependencies we separate the attributes involved in transitive dependencies into a separate table.

   Blood_Donor1(Bd_ID,Bd_Name,City_ID,Reco_ID)

   a){Bd_ID}=>{Bd_Name,City_ID,Reco_ID}

   Blood_Donor2(Bd_ID,Bd_Name,Bd_Bgroup,Bd_age)

   a){Bd_Name}=>{Bd_Bgroup,Bd_age}

   The table is in 3NF as well as in BCNF.

2. City (City_id , City_name)

{City_id}= > {City_name}

The table is in its first normal form.

The table is in its second normal form.

The table is in its third normal form.

The table is in BCNF.

3. Recording_staff

(Reco_Name, Reco_ID, Reco_PhNo)

{Reco_ID} = > {Reco_Name} (functional dependency exists).

{Reco_ID} = > {Reco_PhNo} (functional dependency exists).

The table is in its first normal form.

The table is in its second normal form.

The table is in its third normal form.

The table is in BCNF.

4. Blood_recipient

(reci_ID, reci_age, reci_name, reci_Bqnty, reci_Bgrp, reco_id, city_ID, M_id)

{reci_ID} = >
{reci_age,reci_name,reci_Bqnty,reci_Bgrp,reco_ID,City_ID,M_id}

{reci_name} => {reci_age,reci_Bgrp,reci_Bqnty}

The table is in its first normal form.

The table is in its second normal form.

The table is not in its third normal form as a non-prime attribute reci_name is determining other non-prime attributes.

There should be no Transitive dependencies between non key attributes.To remove transitive dependencies we separate the attributes involved in transitive dependencies into a separate table.

Blood_recipient1(reci_ID, reci_name, reco_id, City_ID, M_id)

{reci_ID} = > {reci_name,reco_ID,City_ID,M_id}

Blood_recipient2(reci_name,reci_age ,reci_Bqnty, reci_Bgrp)

{reci_name} => {reci_age,reci_Bgrp,reci_Bqnty}

The table is in 3 NF and in BCNF.

5. BB_manager (M_id, M_Name)

{M_id} = >{M_Name} (functional dependency exists)

The table is in its first normal form.

The table is in its second normal form.

The table is in its third normal form.

The table is in BCNF.

6. Hospital_Info (Hosp_ID, Hosp_Name, Hosp_needed_Bgrp, Hosp_needed_qnty, City_ID, M_id)

{Hosp_ID}= > {Hosp_Name, City_ID, M_id}

{Hosp_ID, Hosp_needed_Bgrp } = > Hosp_needed_qnty (functional dependency exists)

The table is in its first normal form.

Since every non-primary key attribute is not fully functionally dependent on the primary key of the table, this table is not in second normal form.

Hence we have to split the table.

Hospital1 (Hosp_ID, Hosp_Name, City_ID, M_id).

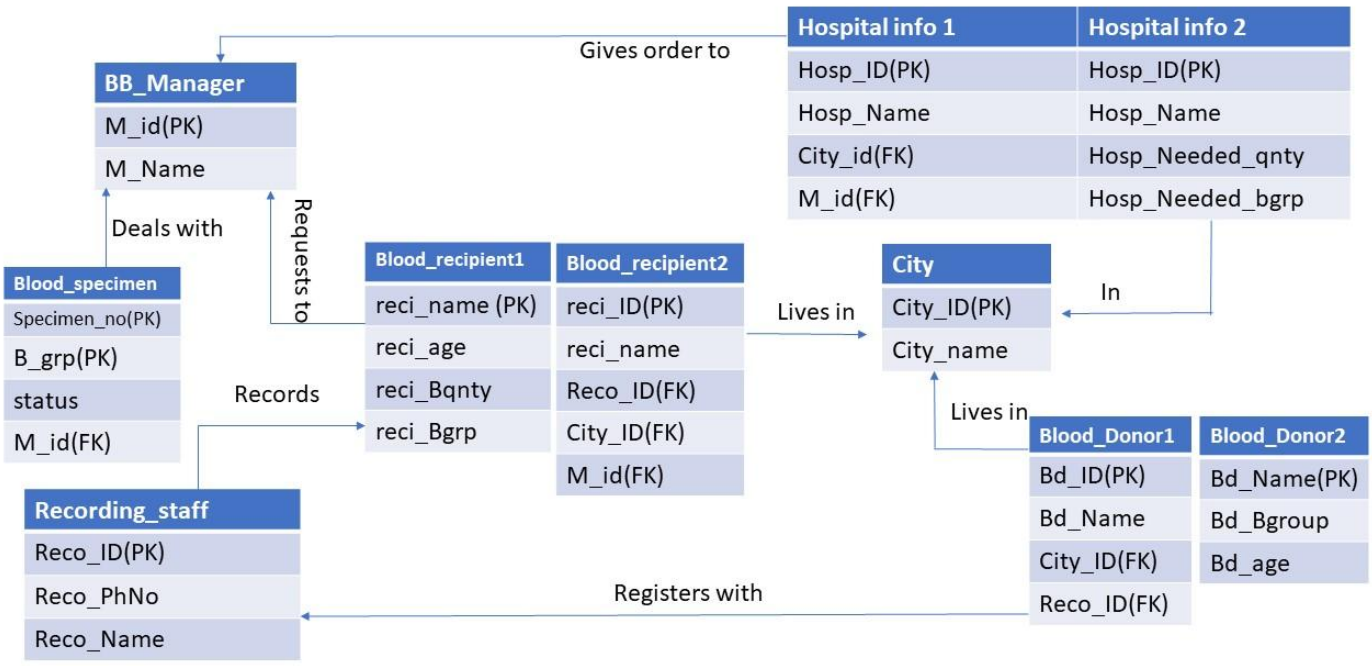
Hospital2 (Hosp_ID, Hosp_needed_Bgrp, Hosp_needed_qnty)

Now it is in its second normal form.

The table is in its third normal form.

The table is in BCNF.

RELATIONAL SCHEMA AFTER NORMALIZATION



SQL IMPLEMENTATION

★ BB_Manager:

```
CREATE TABLE BB_Manager
( M_id int NOT NULL PRIMARY KEY,
 mName varchar(100) NOT NULL
);
INSERT ALL
into BB_Manager VALUES(101,'shivank')
into BB_Manager VALUES(102,'shwetanshu')
into BB_Manager VALUES(103,'singh')
into BB_Manager VALUES(104,'yusuf')
into BB_Manager VALUES(105,'jackson')
into BB_Manager VALUES(106,'akhil')
into BB_Manager VALUES(107,'jojo')
into BB_Manager VALUES(108,'stella')
into BB_Manager VALUES(109,'monika')
into BB_Manager VALUES(110,'himanshi')
select 1 from DUAL;
```

```
1  select * from BB_Manager;
2
```

M_ID	MNAME
101	shivank
102	shwetanshu
103	singh
104	yusuf
105	jackson
106	akhil
107	jojo
108	stella
109	monika
110	himanshi

★ Blood_Donor:

```
CREATE TABLE Blood_Donor_2(
 bd_name varchar(100) NOT NULL PRIMARY KEY,
 bd_age varchar(100),
 bd_Bgroup varchar(10)
);
CREATE TABLE Blood_Donor_1
```

```
( bd_ID int NOT NULL PRIMARY KEY,
  bd_name varchar(100) NOT NULL,
 reco_ID int NOT NULL,
 City_ID int NOT NULL,
 FOREIGN KEY(reco_ID) REFERENCES Recording_Staff(reco_ID),
 FOREIGN KEY(City_ID) REFERENCES City(City_ID),
 FOREIGN KEY(bd_name) REFERENCES Blood_Donor_2(bd_name)
);

INSERT ALL
into Blood_Donor_2 VALUES('Mark',25,'O+')
into Blood_Donor_2 VALUES('Abdul',35,'A-')
into Blood_Donor_2 VALUES('Shivank',22,'AB+')
into Blood_Donor_2 VALUES('Shweta',29,'B+')
into Blood_Donor_2 VALUES('Shyam',42,'A+')
into Blood_Donor_2 VALUES('Dan',44,'AB-')
into Blood_Donor_2 VALUES('Mike',33,'B-')
into Blood_Donor_2 VALUES('Elisa',31,'O+')
into Blood_Donor_2 VALUES('Carrol',24,'AB+')
into Blood_Donor_2 VALUES('shivansh',29,'O-')
select 1 from dual;
```

```
1 select * from Blood_Donor_2;
2
```

BD_NAME	BD_AGE	BD_BGROUP
Mark	25	O+
Abdul	35	A-
Shivank	22	AB+
Shweta	29	B+
Shyam	42	A+
Dan	44	AB-
Mike	33	B-
Elisa	31	O+
Carrol	24	AB+
shivansh	29	O-

INSERT ALL

into Blood_Donor_1 VALUES(150011,'Mark',101412,1100)

into Blood_Donor_1 VALUES(150012,'Abdul',101412,1100)

into Blood_Donor_1 VALUES(150013,'Shivank',101212,1200)

into Blood_Donor_1 VALUES(150014,'Shweta',101212,1300)

into Blood_Donor_1 VALUES(150015,'Shyam',101212,1300)

into Blood_Donor_1 VALUES(150016,'Dan',101212,1200)

into Blood_Donor_1 VALUES(150017,'Mike',101312,1400)

into Blood_Donor_1 VALUES(150018,'Elisa',101312,1200)

into Blood_Donor_1 VALUES(150019,'Carrol',101312,1500)

into Blood_Donor_1 VALUES(150020,'shivansh',101212,1200)

```
select 1 from dual;
```

```
1 select * from Blood_Donor_1;
2
```

BD_ID	BD_NAME	RECO_ID	CITY_ID
150011	Mark	101412	1100
150012	Abdul	101412	1100
150013	Shivank	101212	1200
150014	Shweta	101212	1300
150015	Shyam	101212	1300
150016	Dan	101212	1200
150017	Mike	101312	1400
150018	Elisa	101312	1200
150019	Carrol	101312	1500
150020	shivansh	101212	1200

★ Hospital info:

```
CREATE TABLE Hospital_Info_1
( hosp_ID int NOT NULL PRIMARY KEY,
 hosp_name varchar(100) NOT NULL,
 City_ID int NOT NULL,
 M_id int NOT NULL,
 FOREIGN KEY(M_id) REFERENCES BB_Manager(M_id),
 FOREIGN KEY(City_ID) REFERENCES City(City_ID)
);
CREATE TABLE Hospital_Info_2
( hosp_ID int NOT NULL,
 hosp_name varchar(100) NOT NULL,
 hosp_needed_Bgrp varchar(10),
 hosp_needed_qnty int,
 primary key(hosp_ID,hosp_needed_Bgrp)
);

INSERT ALL
into Hospital_Info_2 VALUES(1,'MayoClinic','A+',20)
```

```
into Hospital_Info_2 VALUES(1,'MayoClinic','A-',0)
into Hospital_Info_2 VALUES(1,'MayoClinic','AB+',40)
into Hospital_Info_2 VALUES(1,'MayoClinic','AB-',10)
into Hospital_Info_2 VALUES(1,'MayoClinic','B-',20)
into Hospital_Info_2 VALUES(2,'Cleveland Clinic','A+',40)
into Hospital_Info_2 VALUES(2,'Cleveland Clinic','AB+',20)
into Hospital_Info_2 VALUES(2,'Cleveland Clinic','A-',10)
into Hospital_Info_2 VALUES(2,'Cleveland Clinic','B-',30)
into Hospital_Info_2 VALUES(2,'Cleveland Clinic','B+',0)
into Hospital_Info_2 VALUES(2,'Cleveland Clinic','AB-',10)
into Hospital_Info_2 VALUES(3,'NYU','A+',0)
into Hospital_Info_2 VALUES(3,'NYU','AB+',0)
into Hospital_Info_2 VALUES(3,'NYU','A-',0)
into Hospital_Info_2 VALUES(3,'NYU','B-',20)
into Hospital_Info_2 VALUES(3,'NYU','B+',10)
into Hospital_Info_2 VALUES(3,'NYU','AB-',0)
into Hospital_Info_2 VALUES(4,'Baylor','A+',10)
into Hospital_Info_2 VALUES(4,'Baylor','A-',40)
into Hospital_Info_2 VALUES(7,'Forestpark','B-',40)
into Hospital_Info_2 VALUES(8,'Parkland','B+',10)
into Hospital_Info_2 VALUES(9,'Pinecreek','AB-',20)
```

select 1 from DUAL;

HOSP_ID	HOSP_NAME	HOSP_NEEDED_BGRP	HOSP_NEEDED_QNTY
1	MayoClinic	A+	20
1	MayoClinic	A-	0
1	MayoClinic	AB+	40
1	MayoClinic	AB-	10
1	MayoClinic	B-	20
2	CleavelandClinic	A+	40
2	CleavelandClinic	AB+	20
2	CleavelandClinic	A-	10
2	CleavelandClinic	B-	30
2	CleavelandClinic	B+	0
2	CleavelandClinic	AB-	10
3	NYU	A+	0
3	NYU	AB+	0
3	NYU	A-	0
3	NYU	B-	20
3	NYU	B+	10
3	NYU	AB-	0
4	Baylor	A+	10
4	Baylor	A-	40
7	Forestpark	B-	40
8	Parkland	B+	10
9	Pinecreek	AB-	20

```
INSERT ALL
into Hospital_Info_1 VALUES(1,'MayoClinic',1100,101)
into Hospital_Info_1 VALUES(2,'Cleveland Clinic',1200,103)
into Hospital_Info_1 VALUES(3,'NYU',1300,103)
into Hospital_Info_1 VALUES(4,'Baylor',1400,104)
into Hospital_Info_1 VALUES(5,'Charlton',1800,103)
into Hospital_Info_1 VALUES(6,'Greenoaks',1300,106)
into Hospital_Info_1 VALUES(7,'Forestpark',1300,102)
into Hospital_Info_1 VALUES(8,'Parkland',1200,106)
into Hospital_Info_1 VALUES(9,'Pinecreek',1500,109)
into Hospital_Info_1 VALUES(10,'WalnutHill',1700,105)
select 1 from DUAL;
```

HOSP_ID	HOSP_NAME	CITY_ID	M_ID
1	MayoClinic	1100	101
2	CleavelandClinic	1200	103
3	NYU	1300	103
4	Baylor	1400	104
5	Charlton	1800	103
6	Greenoaks	1300	106
7	Forestpark	1300	102
8	Parkland	1200	106
9	Pinecreek	1500	109
10	WalnutHill	1700	105

★ Blood Recipient:

```
CREATE TABLE Recipient_2(
 reci_name varchar(100) NOT NULL PRIMARY KEY,
 reci_age varchar(10),
```

```
 reci_Brgp varchar(100),
 reci_Bqnty float);

CREATE TABLE Recipient_1
( reci_ID int NOT NULL PRIMARY kEY,
  reci_name varchar(100) NOT NULL,
  reco_ID int NOT NULL,
  City_ID int NOT NULL,
  M_id int NOT NULL,
  FOREIGN KEY(M_id) REFERENCES BB_Manager(M_id),
  FOREIGN KEY(City_ID) REFERENCES City(City_ID),
  FOREIGN KEY(reco_ID) REFERENCES Recording_Staff(reco_ID),
  FOREIGN KEY(reci_name) REFERENCES Recipient_2(reci_name)
);

INSERT ALL
into Recipient_2 VALUES('Peter',25,'B+',1.5)
into Recipient_2 VALUES('shivank',60,'A+',1)
into Recipient_2 VALUES('akhil',35,'AB+',0.5)
into Recipient_2 VALUES('Parker',66,'B+',1)
into Recipient_2 VALUES('jojo',53,'B-',1)
```

into Recipient_2 VALUES('Preetham',45,'O+',1.5)

into Recipient_2 VALUES('Swetha',22,'AB-',1)

into Recipient_2 VALUES('Swathi',25,'B+',2)

into Recipient_2 VALUES('Lance',30,'A+',1.5)

into Recipient_2 VALUES('Marsh',25,'AB+',3.5)

select 1 from DUAL;

```
1 select * from Recipient_2;
2
```

RECI_NAME	RECI_AGE	RECI_BRGP	RECI_BQNTY
Peter	25	B+	1.5
shivank	60	A+	1
akhil	35	AB+	.5
Parker	66	B+	1
jojo	53	B-	1
Preetham	45	O+	1.5
Swetha	22	AB-	1
Swathi	25	B+	2
Lance	30	A+	1.5
Marsh	25	AB+	3.5

INSERT ALL

into Recipient_1 VALUES(10001,'Peter',101212,1100,101)

into Recipient_1 VALUES(10002,'shivank',101312,1100,102)

into Recipient_1 VALUES(10003,'akhil',101312,1200,102)

into Recipient_1 VALUES(10004,'Parker',101212,1300,104)

into Recipient_1 VALUES(10005,'jojo',101412,1400,105)

into Recipient_1 VALUES(10006,'Preetham',101512,1500,105)

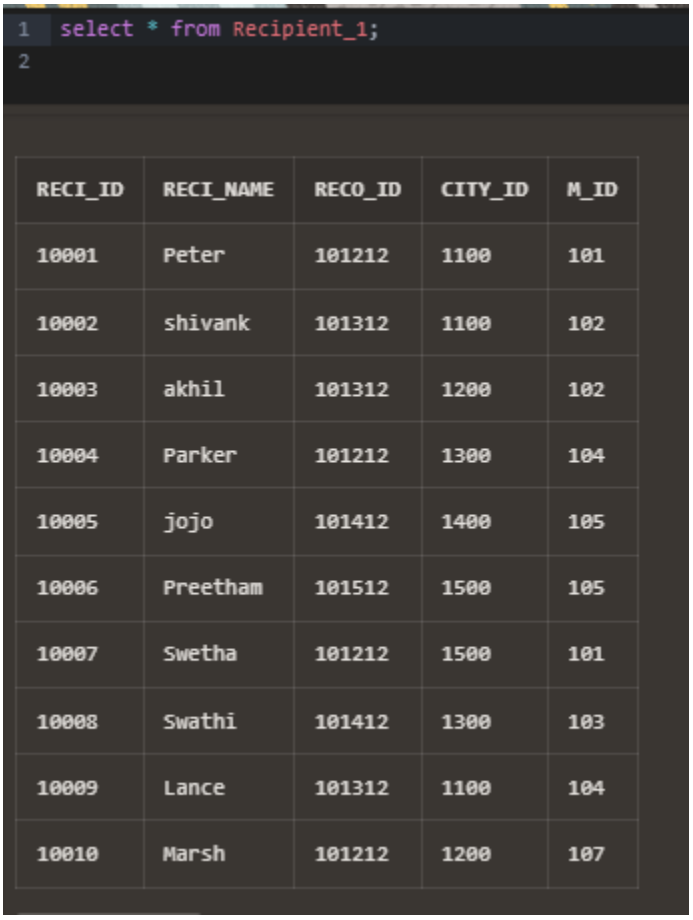
into Recipient_1 VALUES(10007,'Swetha',101212,1500,101)

into Recipient_1 VALUES(10008,'Swathi',101412,1300,103)

into Recipient_1 VALUES(10009,'Lance',101312,1100,104)

into Recipient_1 VALUES(10010,'Marsh',101212,1200,107)

select 1 from DUAL;



```
1  select * from Recipient_1;
2
```

RECI_ID	RECI_NAME	RECO_ID	CITY_ID	M_ID
10001	Peter	101212	1100	101
10002	shivank	101312	1100	102
10003	akhil	101312	1200	102
10004	Parker	101212	1300	104
10005	jojo	101412	1400	105
10006	Preetham	101512	1500	105
10007	Swetha	101212	1500	101
10008	Swathi	101412	1300	103
10009	Lance	101312	1100	104
10010	Marsh	101212	1200	107

★ Recording Staff:

```
CREATE TABLE Recording_Staff
( reco_ID int NOT NULL PRIMARY KEY,
  reco_Name varchar(100) NOT NULL,
  reco_phNo varchar(10)
);
INSERT ALL
into Recording_Staff VALUES(101012,'Lekha',4044846553)
into Recording_Staff VALUES(101112,'shivam',4045856553)
into Recording_Staff VALUES(101212,'Walcot',4045806553)
into Recording_Staff VALUES(101312,'jackson',4045806553)
into Recording_Staff VALUES(101412,'Silva',4045806553)
into Recording_Staff VALUES(101512,'Adrian',4045806553)
into Recording_Staff VALUES(101612,'shivam',4045806553)
into Recording_Staff VALUES(101712,'shyam',4045816553)
into Recording_Staff VALUES(101812,'Jerry',4045826553)
into Recording_Staff VALUES(101912,'Tim',4045836553)
select 1 from dual;
```

```
1 select * from Recording_Staff;
2
```

RECO_ID	RECO_NAME	RECO_PHNO
101012	Lekha	4044846553
101112	shivam	4045856553
101212	Walcot	4045806553
101312	jackson	4045806553
101412	Silva	4045806553
101512	Adrian	4045806553
101612	shivam	4045806553
101712	shyam	4045816553
101812	Jerry	4045826553
101912	Tim	4045836553

★ City:

```
CREATE TABLE City(
City_ID int NOT NULL PRIMARY KEY,
 City_name varchar(100) NOT NULL
);
INSERT ALL
into City VALUES(1100,'Dallas')
into City VALUES(1200,'Austin')
```

into City VALUES(1300,'Irving')

into City VALUES(1400,'Houston')

into City VALUES(1500,'Richardson')

into City VALUES(1600,'Plano')

into City VALUES(1700,'Frisco')

into City VALUES(1800,'Arlington')

into City VALUES(1900,'San Antonio')

into City VALUES(2000,'Tyler')

select 1 from dual;

```
1  select * from City;
2
```

CITY_ID	CITY_NAME
1100	Dallas
1200	Austin
1300	Irving
1400	Houston
1500	Richardson
1600	Plano
1700	Frisco
1800	Arlington
1900	San Antonio
2000	Tyler

★ **Blood Specimen**

```
CREATE TABLE BloodSpecimen (
specimen_number int NOT NULL,
b_group varchar(10) NOT NULL,
status int,
M_id int NOT NULL,
primary key (specimen_number,b_group),
FOREIGN KEY(M_id) REFERENCES BB_Manager(M_id)
);
INSERT ALL
into BloodSpecimen VALUES(1001, 'B+', 1,101)
into BloodSpecimen VALUES(1002, 'O+', 1,102)
into BloodSpecimen VALUES(1003, 'AB+', 1,102)
into BloodSpecimen VALUES (1004, 'O-', 1,103)
into BloodSpecimen VALUES(1005, 'A+', 0,101)
into BloodSpecimen VALUES(1006, 'A-', 1,104)
into BloodSpecimen VALUES(1007, 'AB-', 1,104)
into BloodSpecimen VALUES(1008, 'AB-', 0,105)
into BloodSpecimen VALUES(1009, 'B+', 1,105)
into BloodSpecimen VALUES(1010, 'O+', 0,105)
into BloodSpecimen VALUES(1011, 'O+', 1,103)
```

into BloodSpecimen VALUES(1012, 'O-', 1,102)

into BloodSpecimen VALUES(1013, 'B-', 1,102)

into BloodSpecimen VALUES(1014, 'AB+', 0,101)

select 1 from dual;

```
1  select * from BloodSpecimen;
2
```

SPECIMEN_NUMBER	B_GROUP	STATUS	M_ID
1001	B+	1	101
1002	O+	1	102
1003	AB+	1	102
1004	O-	1	103
1005	A+	0	101
1006	A-	1	104
1007	AB-	1	104
1008	AB-	0	105
1009	B+	1	105
1010	O+	0	105

SAMPLE SQL QUERIES

QUERY 1 : List the names of all blood donors along with the corresponding recording staff names:

SELECT blood_donor_1.bd_name,recording_staff.reco_name

from(blood_donor_1 join recording_staff

on blood_donor_1.reco_id=recording_staff.reco_id);

```
1  --List the names of all blood donors along with the corresponding recording staff names:
2  SELECT blood_donor_1.bd_name,recording_staff.reco_name
3  from( blood_donor_1 join recording_staff
4  on blood_donor_1.reco_id=recording_staff.reco_id);
5
```

BD_NAME	RECO_NAME
Shivank	Walcot
Shweta	Walcot
Shyam	Walcot
Dan	Walcot
shivansh	Walcot
Mike	jackson
Elisa	jackson
Carrol	jackson
Mark	Silva
Abdul	Silva

Query 2 : Get the total quantity of blood needed by each hospital:

Select hospital_info_1.hosp_name,
SUM(hospital_info_2.hosp_needed_qnty)

from hospital_info_1 join hospital_info_2 on

hospital_info_1.hosp_id = hospital_info_2.hosp_id

group by(hospital_info_1.hosp_name);

```
1  --Get the total quantity of blood needed by each hospital:
2  Select hospital_info_1.hosp_name, SUM(hospital_info_2.hosp_needed_qnty) as Total_Quantity
3  from hospital_info_1 join hospital_info_2 on
4  hospital_info_1.hosp_id = hospital_info_2.hosp_id
5  group by(hospital_info_1.hosp_name);
6
```

HOSP_NAME	TOTAL_QUANTITY
Baylor	50
Forestpark	40
Parkland	10
NYU	30
MayoClinic	90
CleavelandClinic	110
Pinecreek	20

Query 3 : List the hospitals along with the names of their associated blood bank managers.

select hospital_info_1.hosp_name,bb_manager.mname from

bb_manager join hospital_info_1

on bb_manager.m_id = hospital_info_1.m_id;

```
1  --List the hospitals along with the names of their associated blood bank managers
2  select hospital_info_1.hosp_name,bb_manager.mname from
3  bb_manager join hospital_info_1
4  on bb_manager.m_id = hospital_info_1.m_id;
```

HOSP_NAME	MNAME
MayoClinic	shivank
CleavelandClinic	singh
NYU	singh
Baylor	yusuf
Charlton	singh
Greenoaks	akhil
Forestpark	shwetanshu
Parkland	akhil
Pinecreek	monika
WalnutHill	jackson

Query 4 : Retrieve the names of blood donors who have the blood group 'O+':

SELECT BLOOD_DONOR_1.BD_NAME

FROM blood_donor_1 JOIN blood_donor_2

ON blood_donor_1.bd_name = blood_donor_2.bd_name

WHERE blood_donor_2.bd_bgroup = 'O+';

```
1  --Retrieve the names of blood donors who have the blood group 'O+':
2  SELECT BLOOD_DONOR_1.BD_NAME,blood_donor_2.bd_bgroup
3  FROM blood_donor_1 JOIN blood_donor_2
4  ON blood_donor_1.bd_name = blood_donor_2.bd_name
5  WHERE blood_donor_2.bd_bgroup = 'O+';
6
```

BD_NAME	BD_BGROUP
Mark	O+
Elisa	O+

Query 5 : Retrieve the blood groups and the total quantity of blood needed for each blood group:

SELECT HOSPITAL_INFO_2.HOSP_NEEDED_BGRP,

SUM(HOSPITAL_INFO_2.HOSP_NEEDED_QNTY)

FROM hospital_info_2

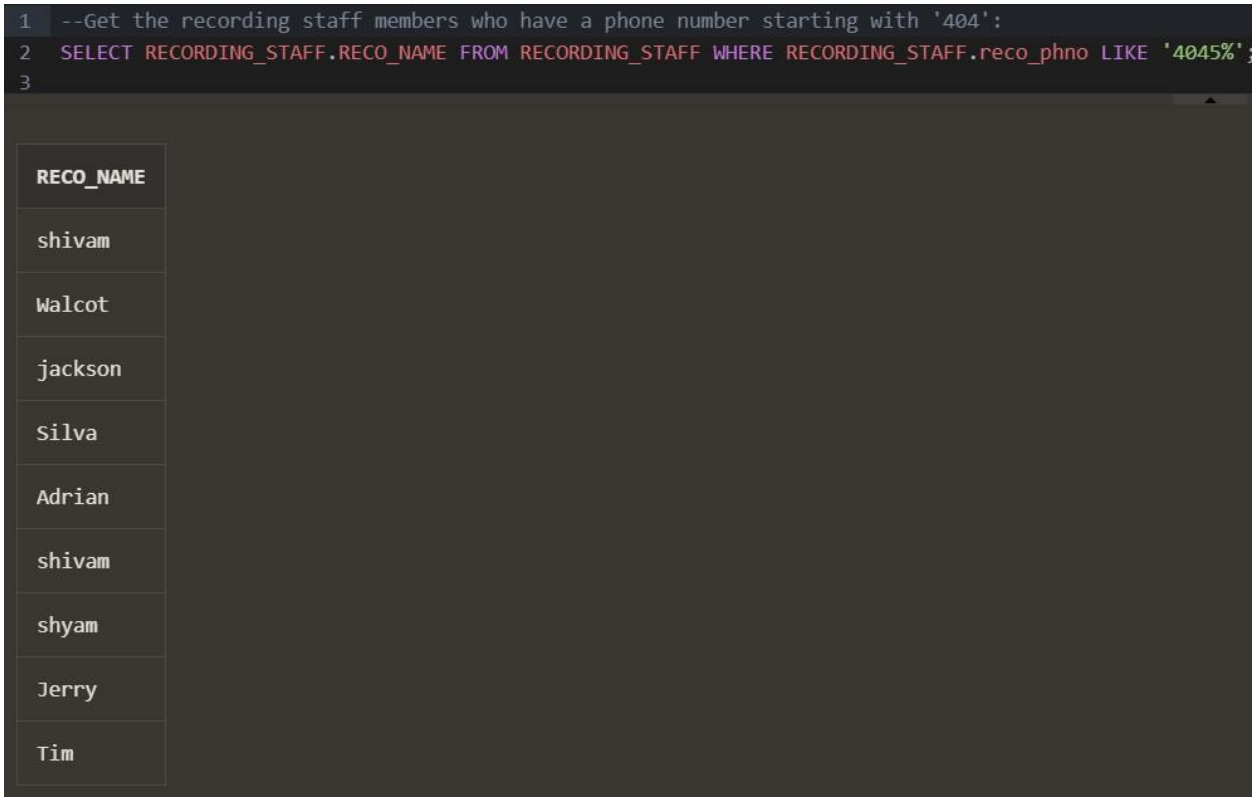
GROUP BY (HOSPITAL_INFO_2.HOSP_NEEDED_BGRP);

```
1  --Retrieve the blood groups and the total quantity of blood needed for each blood group:
2  SELECT HOSPITAL_INFO_2.HOSP_NEEDED_BGRP,SUM(HOSPITAL_INFO_2.HOSP_NEEDED_QNTY)
3  FROM hospital_info_2
4  GROUP BY (HOSPITAL_INFO_2.HOSP_NEEDED_BGRP);
5
6
```

HOSP_NEEDED_BGRP	SUM(HOSPITAL_INFO_2.HOSP_NEEDED_QNTY)
AB+	60
B+	20
A+	70
A-	50
B-	110
AB-	40

Query 6 : Get the recording staff members who have a phone number starting with '4045':

SELECT RECORDING_STAFF.RECO_NAME FROM RECORDING_STAFF WHERE RECORDING_STAFF.reco_phno LIKE '4045%';



SYSTEM SPECIFICATION

Device name:- Acer Aspire 7

Processor:- i5 9 Generation

System type:- 64 bit Operating system

RAM:- 8GB

Memory:- 512 GB (SSD)

Operating system :- Window 10

CONCLUSION

In conclusion, the blood bank management system project has been successfully implemented, providing an efficient and organized solution for managing blood donation, storage, and distribution processes.

The database design includes several essential tables, such as Recording Staff, City, Blood Donor, BB Manager, Blood Recipient, Blood Specimen and Hospital Information. These tables establish the necessary relationships between entities, ensuring data integrity and smooth operations.

Overall, the blood bank management system project provides a robust and scalable solution for the effective management of blood donations, storage, and distribution.

It can greatly enhance the efficiency of blood banks and hospitals, ensuring timely availability of blood for patients in need. With further development and integration with user interfaces, this system can be deployed in real-world settings, benefiting both medical professionals and individuals requiring blood transfusions.