BLOOD BANK MANAGEMENT SYSTEM

I3306 DataBase Project

Prepared by

Hassan Alsayed Ahmad 109229

LEBANESE UNIVERSITY

FACULTY OF SCIENCES I

DEPARTMENT OF COMPUTER SCIENCES

2023 – 2024

Abstract:

Blood Bank Management System is a crucial tool in the healthcare sector designed to efficiently manage the collection, storage, and distribution of blood donations. This system streamlines the entire process, from donor registration to inventory tracking and blood transfusion requests. It ensures the availability of diverse blood types, maintains accurate donor records, and facilitates rapid response to emergency situations. Through the integration of technology, such as barcoding and database management, the Blood Bank Management System enhances the overall effectiveness of blood banks, promoting a more organized and responsive approach to blood supply management

Chap1: Architecture:

The architecture to be used is Client/Server where my system depends on, the client whether the Donor or the Patient will send a request as client to the server that will respond by connecting to database and other accessibilities. To the project we use the javafx and jdbc connection the help me to connect to the MYSQL.

Our system is made up of Three parts:

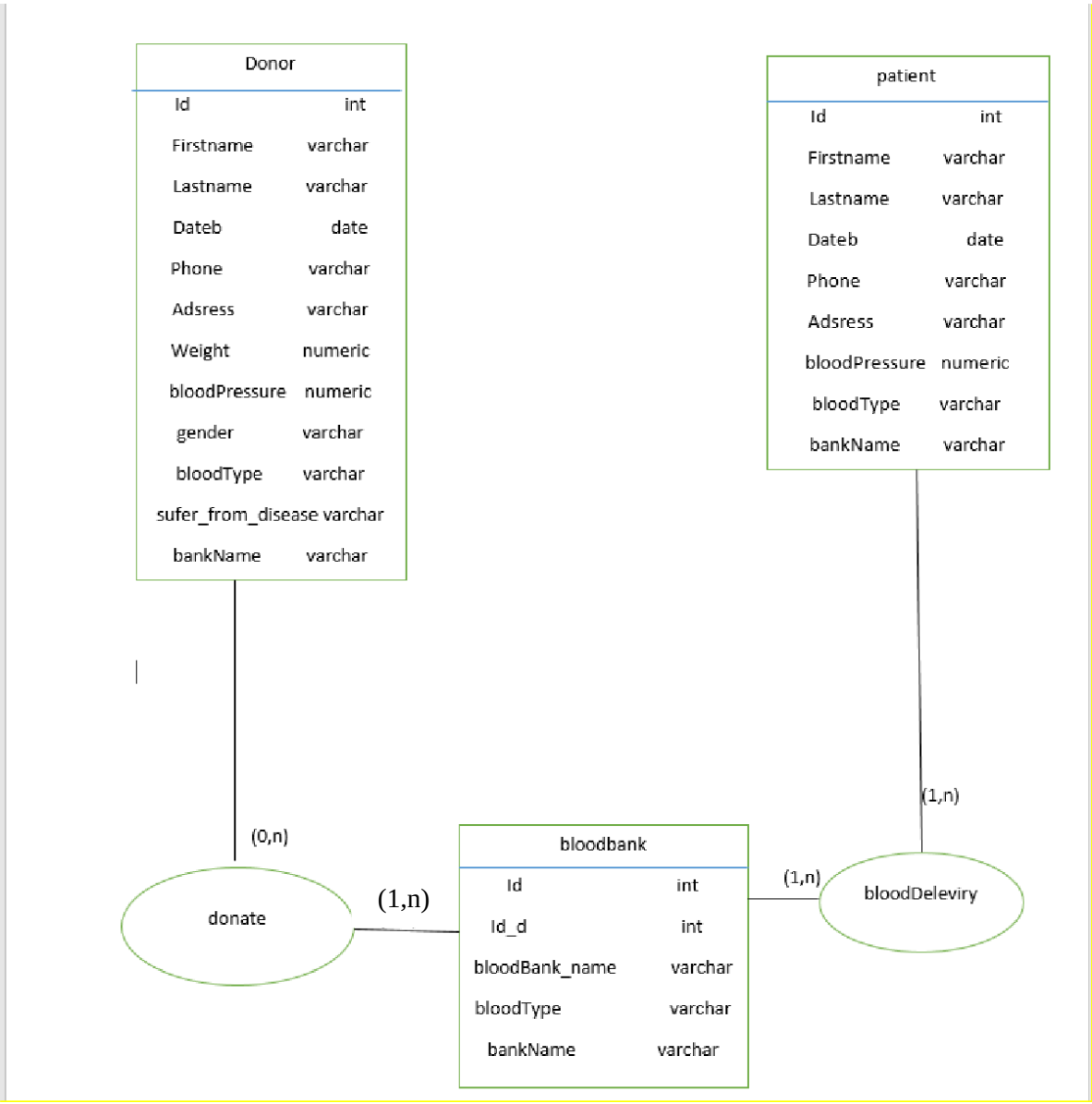
Donor: he login to system and donate his blood in the blood bank .

Patient: he login to system and deliver blood from blood bank.

Admin: who make all operation (search-delete-update).

Chap2: DataBase Modeling:

1- ER_Diagram:

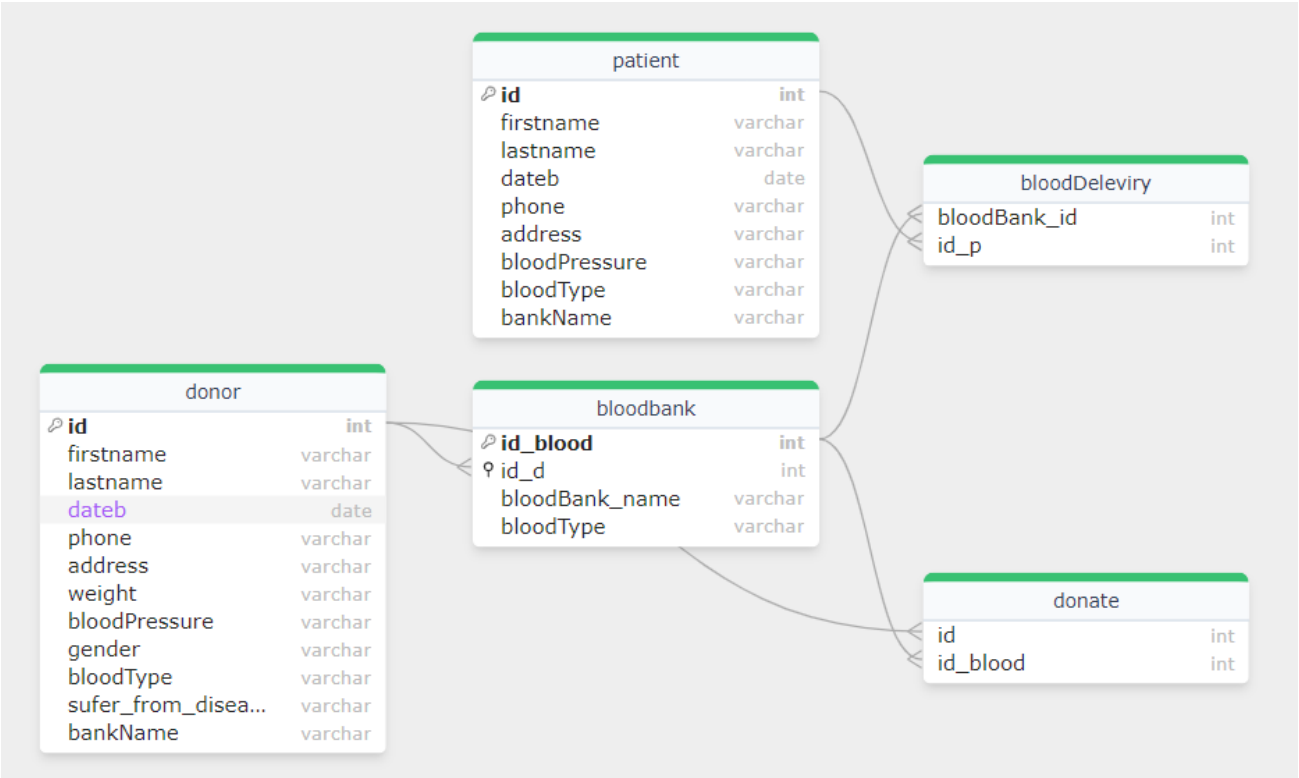


Donor : has id as a primary key also has the other attributes,and a relation donate with blood bank which is 1 to many relationship.

Patient: has id as a primary key also has the other attributes,and a relation bloodDeleviry with blood bank which is 1 to many relationship.

And on the other side 1 to many relationship.which make the bloodDeleviry as relationship betwwen them.

2- PDM:



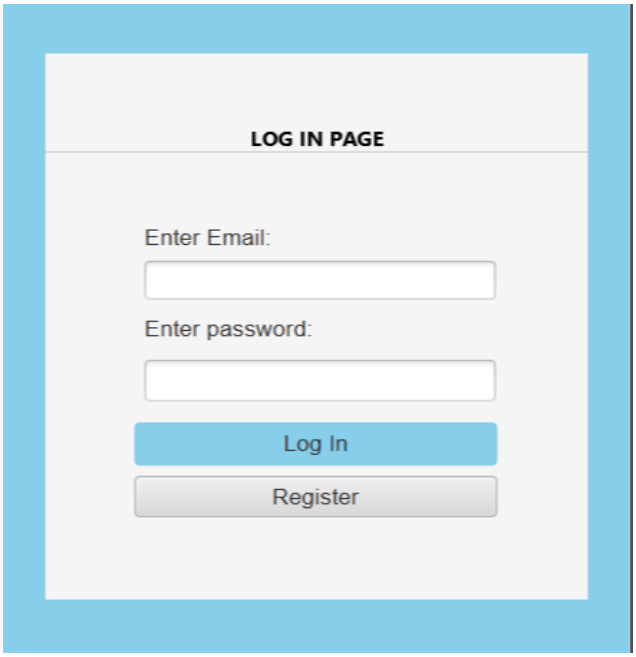
The donor will take an automatic id then donate in the blood bank.

The patient will take an automatic id and deliver from the blood bank.

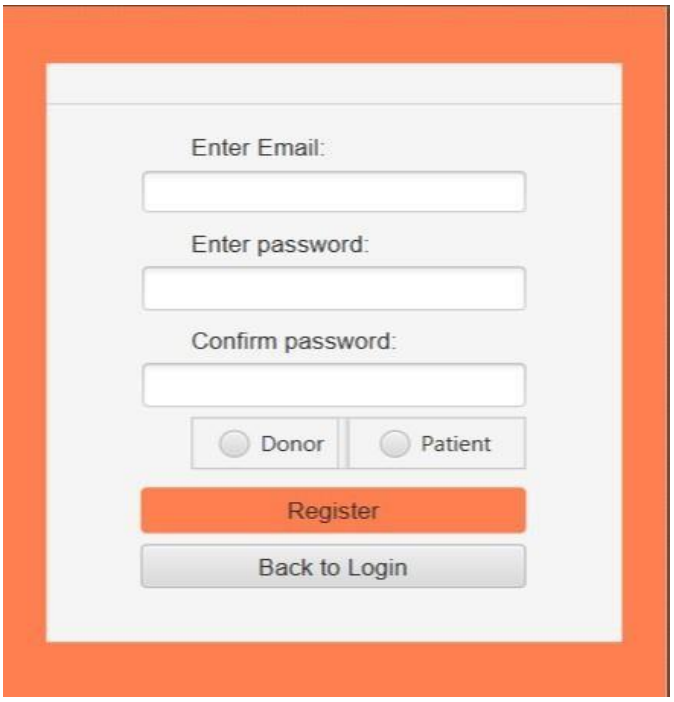
Chap3: Interface:

1-login and registration:

Login page:



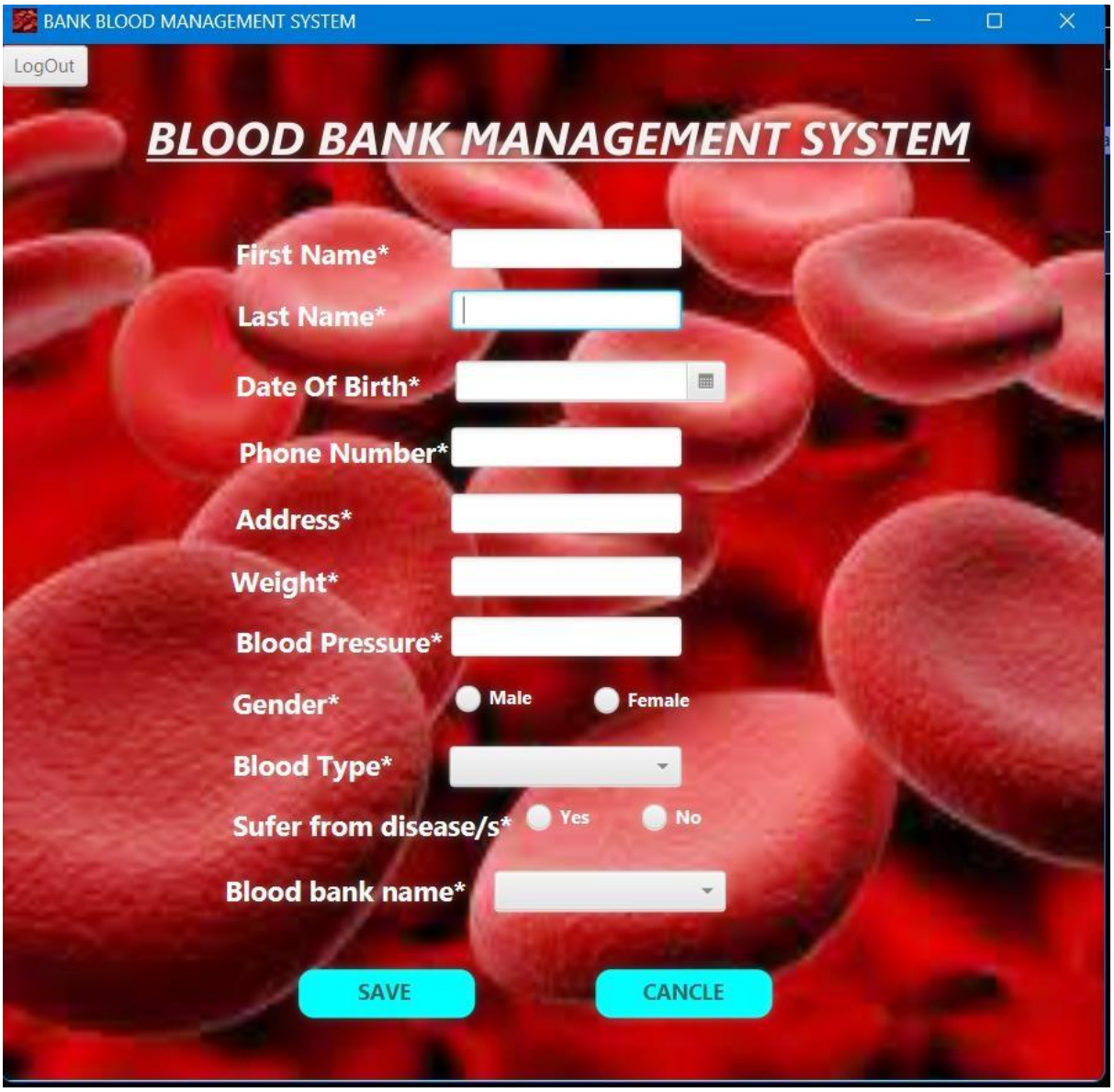
Register page:



:

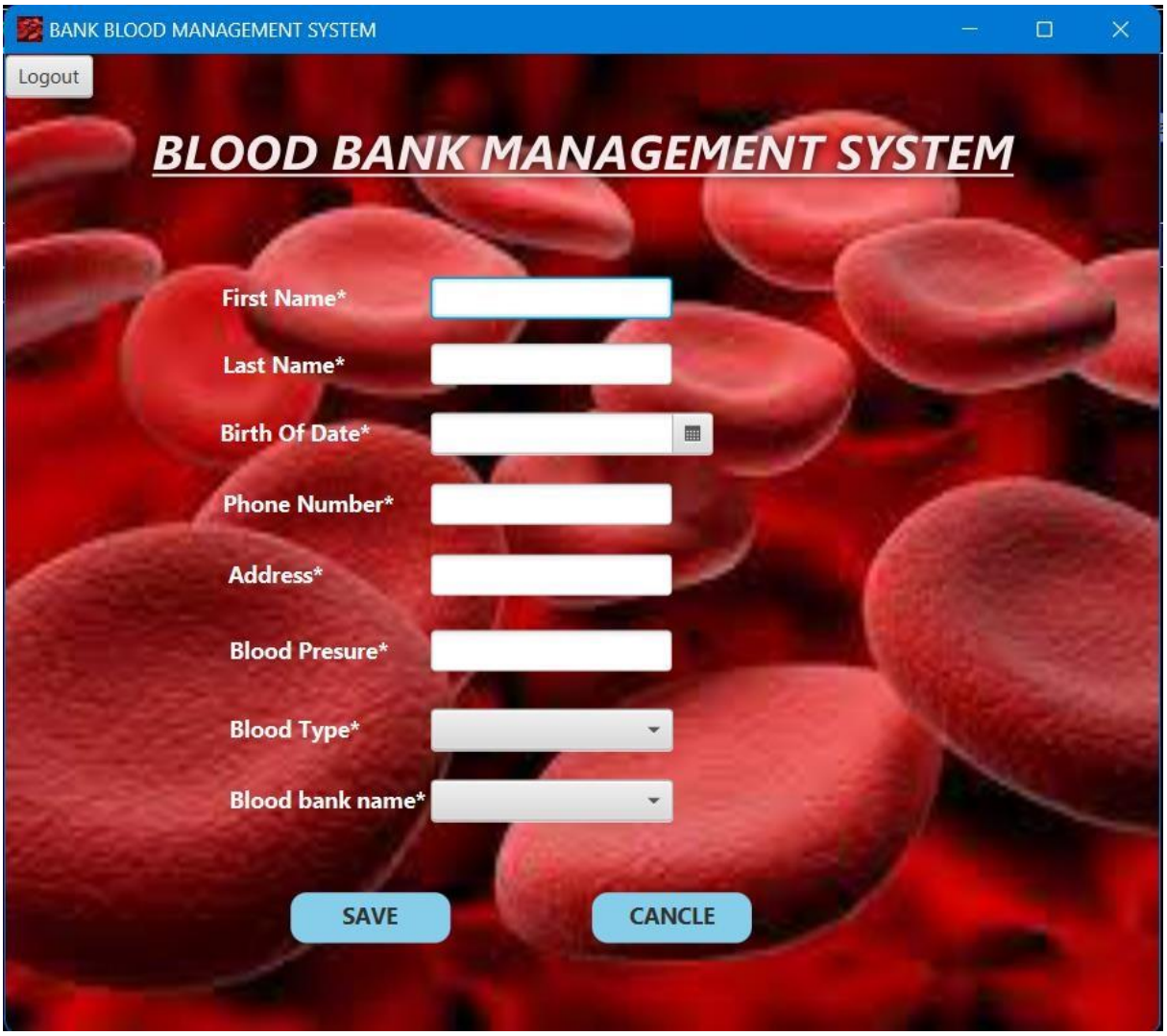
First of all the user should register before and choose if he donor or patient then login and go to appropriate page.

3- Add donor:



The donor enter his info then click to save into database or click cancle to cancled data or click to logout and go to login page.

4- Add Patient:



The patient enter his info and by clicking the save button he can reserve a blood type in any blood bank.

5- Admin page:

The admin choose the appropriate page by clicking one of them to go to donor info or patient info…

Donor info , the admin can search about specific donor by entering his id also he can delete it and show all donors who are donated in the system and upadeting his dynamic info like phone number,weight,blood pressure and bank name.

Patient info , the admin can search about specific patient by entering his id also he can delete it and show all patients who are deliverd in the system and upadeting his dynamic info like phone number,blood pressure and bank name.

Chap4: Implementation and testing:

I use javafx to implement my project with jdbc the is used to connect to MYSQL. And for the database I use the MYSQL.

1-javaFX:

- connection to database:

```
String jdbcUrl = "jdbc:mysql://127.0.0.1:3306/bloodbankmanagmentsystem";
String username = "root";
String password = "[redacted]";

try (Connection connection = DriverManager.getConnection(jdbcUrl, username, password)) {
```

- Donor model:

```
private String fn,ln,Address,Phone,bloodpressure,weight,gender,sufer_from_disease,bloodType,bankName;
3 usages
private Date Birth;
3 usages
int id;

no usages
public SelectDoner(int id,String fn, String ln, String Address, String Phone, String bloodpressure, String weight,
                   Date Birth,String gender,String sufer_from_disease,String bloodType,String bankName) {...}

no usages
public String getFn() { return fn; }

no usages
public void setFn(String fn) { this.fn = fn; }

no usages
public String getLn() { return ln; }

no usages
public void setLn(String ln) { this.ln = ln; }

no usages
public String getAddress() { return Address; }

no usages
public void setAddress(String address) { Address = address; }

no usages
public String getPhone() { return Phone; }
```

- Patient model:

```
3 usages
private String fn, ln, Address, Phone, bloodpressure, bloodType,bankName;
3 usages
private Date Birth;
3 usages
int id;

no usages
public SelectPatient(int id, String fn, String ln, String Address, String Phone, String bloodpressure,
                     Date Birth, String bloodType,String bankName) {...}

no usages
public String getFn() { return fn; }

no usages
public void setFn(String fn) { this.fn = fn; }

no usages
public String getLn() { return ln; }

no usages
public void setLn(String ln) { this.ln = ln; }

no usages
public String getAddress() { return Address; }

no usages
public void setAddress(String address) { Address = address; }
```

- Bloodbank model:

```
public class BloodBank {

    3 usages
    String bloodBank_name ,bloodType;


    no usages
    public BloodBank(String bloodBank_name, String bloodType) {
        this.bloodBank_name = bloodBank_name;
        this.bloodType = bloodType;
    }


    no usages
    public String getBloodBank_name() { return bloodBank_name; }

    no usages
    public void setBloodBank_name(String bloodBank_name) { this.bloodBank_name = bloodBank_name; }

    no usages
    public String getBloodType() { return bloodType; }

    no usages
    public void setBloodType(String bloodType) { this.bloodType = bloodType; }

}
```

2- SQL:

Create table:

```sql
-- create table donor
create table donor(
id  int not null,
firstname varchar(20),
lastname varchar(20),
dateb date,
phone varchar(20),
address varchar(20),
weight varchar(15),
bloodPressure varchar(5),
gender varchar(10),
bloodType varchar(10),
sufer_from_disease varchar(5),
bankName varchar(15),
primary key (id)
);
-- create table patient
create table patient (
id int not null,
firstname varchar(20),
lastname varchar(20),
dateb date,
```

```
23    phone varchar(20),
24    address varchar(20),
25    bloodPressure varchar(5),
26    bloodType varchar(10),
27    bankName varchar(15),
28    primary key (id)
29    );
30    -- create table bloodbank
31    CREATE TABLE bloodbank (
32    id_blood INT,
33    id_d INT,
34    bloodBank_name VARCHAR(15),
35    bloodType VARCHAR(5),
36    PRIMARY KEY (id_blood),
37    FOREIGN KEY (id_d) REFERENCES donor(id) ON DELETE SET NULL
38    );
39    -- create table bloodDeleviry
40    create table bloodDeleviry(
41    bloodBank_id INT,
42    id_p int ,
43    foreign key (id_p) references patient(id),
44    foreign key (bloodBank_id) references bloodbank(id_blood)
45    );
```

```
45    -- create table donate
46    create table donate(
47    id INT,
48    id_blood INT,
49    FOREIGN KEY (id) REFERENCES donor(id),
50    foreign key (id_blood) references bloodbank(id_blood)
51    );
52    -- create table register
53    create table register(
54    email varchar(50),
55    password varchar(20),
56    confirmPassword varchar(20),
57    role varchar(10)
58    );
59
```

Indexes:

```sql
CREATE INDEX index_name_donor ON donor (firstname);

CREATE INDEX index_name_patient ON patient (firstname) ;

CREATE INDEX index_bloodType_bloodbank ON bloodbank (bloodType);

```

Security:

```sql
create user donors;
grant insert on bloodbankmanagmentsystem.donor to donors;

create user patients;
grant insert on bloodbankmanagmentsystem.patient to patient;

create user admin;
grant update,delete,select on bloodbankmanagmentsystem.patient to admin;
grant update,delete,select on bloodbankmanagmentsystem.donor to admin;
grant select on bloodbankmanagmentsystem.bloodBank to admin;
```

Procedures:

```
String createProcedureSQL = "CREATE PROCEDURE updatePatient(IN phoneN VARCHAR(20),IN BLOODP VARCHAR(10), IN Bank VARCHAR(20))
        "BEGIN " +
        "  UPDATE patient " +
        "  SET phone=phoneN,bloodPressure=BLOODP, bankName=Bank " +
        "  WHERE id =" + ID + "; " +
        "END";
```

```
String createProcedure = "CREATE PROCEDURE searchDonor(IN idSearch INT) " +
        "BEGIN " +
        "    SELECT * FROM donor WHERE id = idSearch; " +
        "END";
```

```
String createProcedureSQL = "CREATE PROCEDURE deletepatient() " +
        "BEGIN " +
        "  DELETE FROM patient " +
        "  WHERE id=" + ID + "; " +
        "END";
```

```
String createProcedureSQL = "CREATE PROCEDURE updateDonor(IN phoneN VARCHAR(20), IN WEIGHT VARCHAR(20), IN BLOODP VARC
        "BEGIN " +
        "  UPDATE donor " +
        "  SET phone=phoneN, weight=WEIGHT, bloodPressure=BLOODP, bankName=Bank " +
        "  WHERE id=" + ID + "; " +
        "END";
```

```
String createProcedureSQL = "CREATE PROCEDURE procedurePatient(IN bank VARCHAR(10),IN type VARCHAR(10)) " +
        "BEGIN " +
        "    SELECT COUNT(bloodType) AS sum FROM bloodbank WHERE bloodBank_name = bank AND bloodType = type; " +
        "END";
```

```
String createProcedure = "CREATE PROCEDURE searchPatient(IN idSearch INT) " +
        "BEGIN " +
        "    SELECT * FROM patient WHERE id = idSearch; " +
        "END";
```

```
String createProcedureSQL = "CREATE PROCEDURE deleteDonor() " +
        "BEGIN " +
        "DELETE FROM bloodbank"+
        "  WHERE id_d=" + ID + "; " +
        "  DELETE FROM donor " +
        "  WHERE id=" + ID + "; " +
        "END";
```

Triggers:

```
CREATE TRIGGER insertdonor
BEFORE INSERT ON donor
FOR EACH ROW
SET NEW.id = (SELECT MAX(id) + 1 FROM donor);

CREATE TRIGGER id_blood
BEFORE INSERT ON bloodbank
FOR EACH ROW
SET NEW.id_blood = (SELECT MAX(id_blood) + 1 FROM bloodbank),NEW.id_d = (SELECT MAX(id_d) + 1 FROM bloodbank);



CREATE TRIGGER insertPatient
BEFORE INSERT ON patient
FOR EACH ROW
SET NEW.id = (SELECT MAX(id) + 1 FROM patient);
```