18CSC303J- DATABASE MANAGEMENT SYSTEMS

COURSE PROJECT REPORT

Submitted by

SHAIK ABDUL JALEEL

RA2111047010026

Under the Guidance of

Dr. G.DINESH

Associate Professor, Department of COMPUTATIONAL INTELLIGENCE

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE ENGINEERING

with specialization in ARTIFICAL INTELLIGENCE

SCHOOL OF COMPUTING
COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

APRIL 2023

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this project report “Blood Bank Management System” is the bonafide work of SHAIK ABDUL JALEEL RA2111047010026” of II Year/IV Sem B.tech(AI)who carried out the mini project work under my supervision for the course18AIC207J - Database Management Systems for Artificial Intelligence.

Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

SIGNATURE
Faculty In-Charge
Dr .G.DINESH
Assistant Professor
Department of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur, Campus. Chennai

HEAD OF THE DEPARTMENT
Dr. R Annie thra
Professor and Head
Department of Computational Intelligence,
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

PROJECT TITLE:

BLOOD BANK MANAGEMENT SYSTEM

CONTENTS

ABSTRACT

The goal of this project is to create a management system for blood banks. Any clinic, hospital, laboratory, or emergency situation that requires blood units for survival can benefit from a blood bank management system.

In emergency situations, our system can locate the necessary type of blood from donors or a blood bank. In the event of an emergency, the current system relies on grapevine communication to locate blood, either from a donor or a blood bank. The purpose of introducing such a system is to eliminate the panic that occurs when blood is unavailable in an emergency. Any individual who is interested in donating blood can register themselves using this application, and any organization that wishes to register with this site can also register.

In addition, this website can also be used by general consumers who wish to request blood online. If necessary, admin is the primary authority and can perform addition, deletion, and modification

PROBLEM STATEMENT

Blood banks collect, store, and distribute the blood they collect to patients who require it. Blood donors are referred to as "donors." The blood that the banks receive is then grouped according to the blood groups. Additionally, they ensure that the blood is free of contamination.

The blood bank's primary goal is to donate blood to hospitals and other healthcare facilities in order to save a patient's life. Without sufficient and pure blood, no hospital can keep the health care system running. The quality of the blood and the people who donate it, or "donors," are the primary concerns of every blood bank. But the job is hard. The requirement to maintain high-quality blood and keep track of donors cannot be met by the current system. We developed a brand-new system known as the "Blood Donation Management System" in order to circumvent all of these limitations. The "Blood Bank Management System" enables us to monitor the quality of blood and the quantity of blood that is available upon request from the acceptor. The current manual systems take a long time and are not very effective. The blood distribution is automated by the Blood Bank Management System.

Each blood bank's thousands of records are contained in this database. Compared to the manual system, using this system makes searching for available blood simple and saves a lot of time. Information related to blood bank administration and inventory management will be stored, processed, recovered, and analysed by it. This system is designed to be manageable, efficient in terms of time and money, flexible, and does not require a lot of human labour.

OBJECTIVES

The objectives of blood bank management system are:

- In order to ensure accurate costs, records, and counts, provide a report that can be generated listing donors, seekers, and the total amount of blood units consumed.
- To assist with bringing issues to light locally about blood gift and make some blood gift occasions or lobbies for the general population.
- To make it possible for both the public and the organization to reserve a blood donation session and day online.
- To add authentic and authorized features to the current system so that authorized users can only view private and confidential data.
- To create a legitimate informational portal about blood donation and system management.
- To give the blood bank a way to promote blood donation programs.
- To make it possible for potential recipients to search for and match volunteer donors and request blood.
- To record donor and blood details in order to provide the blood bank with effective donor and blood stock management functions.
- To increase the effectiveness of blood stock management by informing staff members at blood banks when blood supplies are running low or have run out.
- To provide a donor and blood stock database that is synchronized and centralized.

T

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

1. RDBMS: Relational Database Management System
2. DBMS: Database Management System
3. SQL: Structured Query Language
4. CRUD: Create, Read, Update, and Delete
5. ACID: Atomicity, Consistency, Isolation, and Durability
6. ERD: Entity-Relationship Diagram
7. DDL: Data Definition Language
8. DML: Data Manipulation Language
9. DBA: Database Administrator

SCOPE OF PROJECT

All of the processes and activities of the blood bank management system will be maintained using the system. It offers an elegant blood management system as well as an online donor list and list of hospitals.

The purpose of this project is to link all blood banks, hospitals, and donors into a single network, validate various data, and store information about blood and each person's health. This system will be used to store data on a centralized server that houses a database and prevents third parties from accessing individual data. The system can be used to make an application for smartphones. Getting as close to the donor as possible from the emergency area.

The project provides the doctors with all necessary information about the donors. The donor can be located by doctors at any time. Because of the user friendly features it offers, it can be easily operated and save a persons life.

ARCHITECTUE AND DESIGN

ARCHITECTURE DIAGRAM:

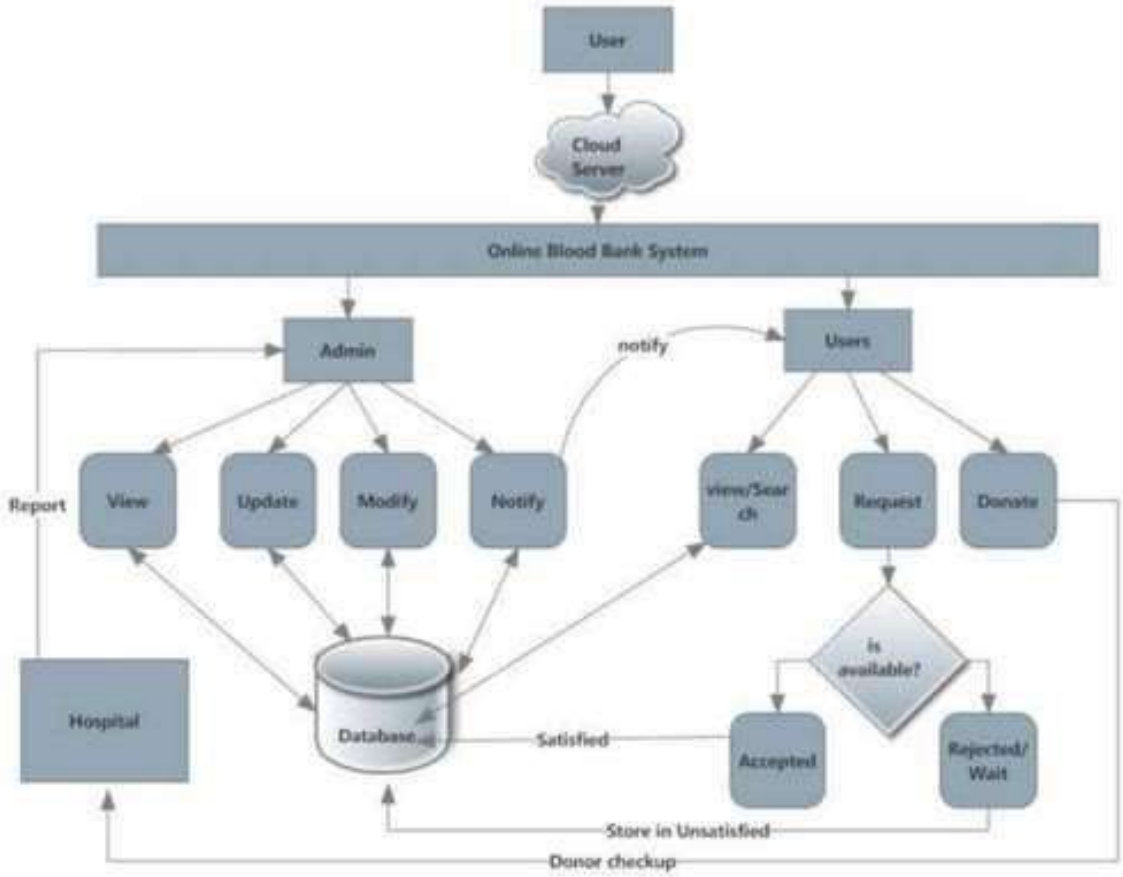


Figure 1: Architecture Diagram

ER DIAGRAM:

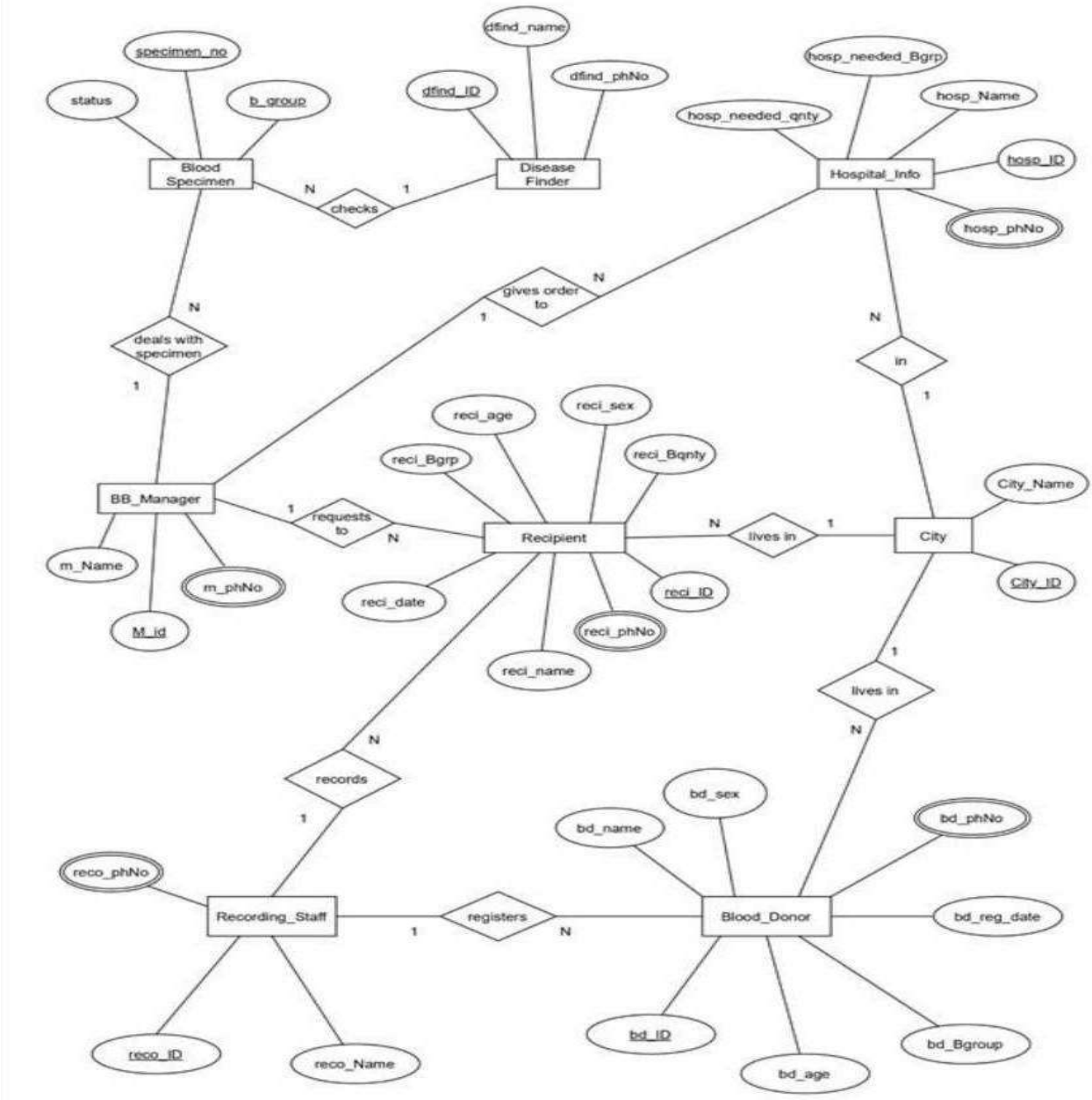


Figure 2: ER Diagram

INFORMATION OF ENTITIES:

1) **Blood_Donor:**
   (Attributes – bd_ID, bd_name, bd_sex, bd_age, bd_Bgroup, bd_reg_date, bd_phNo)

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

2) **Recipient:**
   (Attributes – reci_ID, reci_name, reci_age, reci_Bgrp,reci_Bqnty , reci_sex, reci_reg_date, reci_phNo)

   The Recipient is the person who receives blood from blood bank, when blood is given to a recipient a recipient ID (reci_ID) is generated and used as primary key for the recipient entity to identify blood recipients information. Along with it name ,age, sex, blood group (needed), blood quantity(needed) , phone number, and registration dates are also stored in the data base under recipiententity.

3) **BB_Manager:**
   (Attributes – m_ID, m_Name, m_phNo)

   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 requestsfrom recipients and hospitals. Blood manager has a unique identification number (m_ID) used as primary key along with name and phone number of blood bank manager will be stored in data base 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 enitity has reco_ID which is primary key along with recorder's name and recorder's phone number will also be stored in the database under Recording_Staff entity.

5) **BloodSpecimen :**
   (Attributes – specimen_number, b_group , status)

   In data base, under 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 statusattribute which will show if the blood is contaminated on not.

6) **DiseaseFinder :**

(Attributes - dfind_ID, dfind_name, dfind_PhNo)

In data base , under DiseaseFinder entity we will store the information of thedoctor who checks the blood for any kind of contaminations. To store that information we have unique identification number (dfind_ID) as primary key. Along with name and phone number of the doctor will also be stored under same entity.

7) Hospital_Info :

(Attributes – hosp_ID, hosp_name, hosp_needed_Bgrp,hosp_needed_Bqnty)

In the data base, under Hospital_Info entity we will store the information of hospitals. In this hosp_ID and hosp_needed_Bgrp together makes the primarykey. We will store hospital name and the blood quantity required at the hospital.

8) 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 primary key to identify the information about the city. Along with ID city names will also be stored under this entity.

RELATIONSHIP BETWEEN ENTITIES:

1) City and Hospital_Info:
Relationship = “in”

Type of relation = 1 to many

Explanation = A city can have many hospital in it. One hospital willbelong in one city.

2) City and Blood_Donor:
Relationship = “lives in”

Type of relation = 1 to many

Explanation = In a city, many donor can live. One donor will belong toone city.

3) City and Recipient:
Relationship = “lives in”

Type of relation = 1 to many

Explanation = In a city, many recipient can live. One recipient willbelong to one city.

4) Recording_Staff and Donor:
Relationship = “registers”

Type of relation = 1 to many

Explanation = One recording staff can register many donors. Onedonor 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. Onerecipient 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 request to onblood bank manager.

7) BB_Manager and Blood Specimen:
Relationship = “deales with specimen”

Type of relation = 1 to many

Explanation = One Blood bank manager can manage many bloodspecimen 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 andone manager can handle requests from many recipients.

9) Disease_finder and Blood Specimen:
Relationship = “checks”,

Type of relation = 1 to many

Explanation = A disease finder can check many blood samples. Oneblood sample is checked by one disease finder.

RELATIONAL SCHEMAS:

1) Donor Table:

- The relationship with Recording staff and Donor is 1 to many. That's why 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 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 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 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 primary key of Blood Specimen 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 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 Specimen Table:

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

6) Disease Finder Table:

- The relationship with Disease finder and Blood Specimen is of 1 to many. Therefore, the primary key of Disease finder is used as a foreignkey in Blood Specimen.

7) 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 primarykey of Blood Bank Manager is used as a foreign key in Blood Specimen,Recipient and Hospital info.

8) Hospital info Table:

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

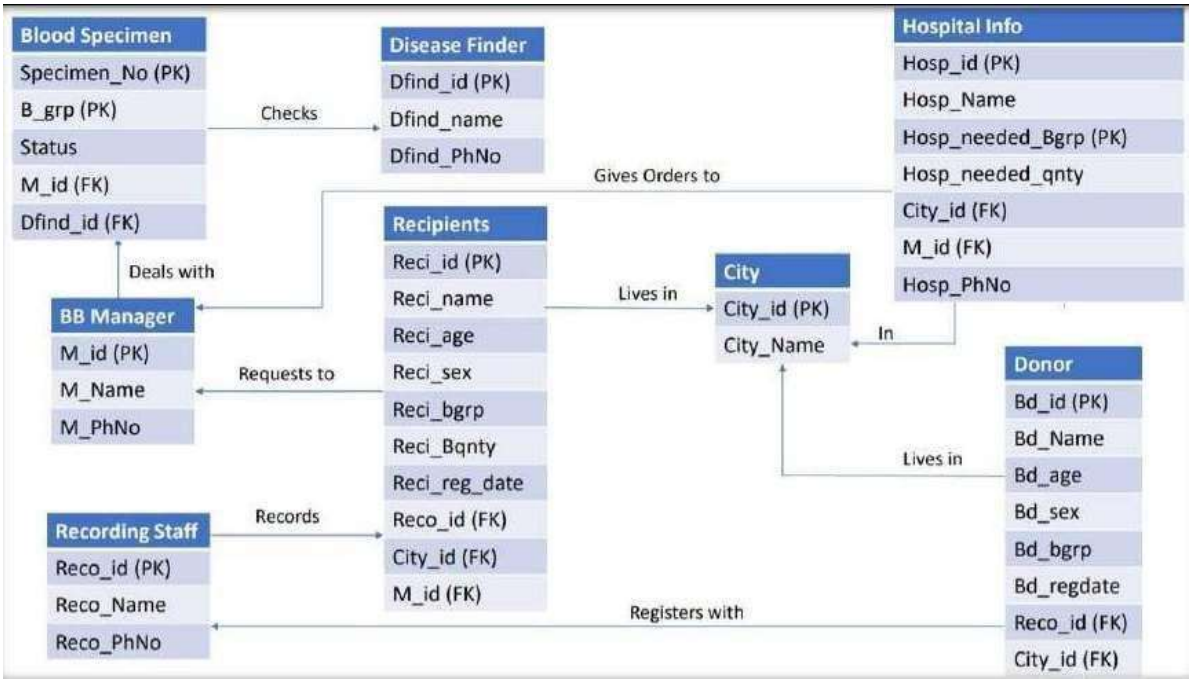


Figure 3: Relational Schema

NORMALIZATION:

Normalization Rule:

Normalization rules are divided into the following normal forms:

1) First Normal Form
2) Second Normal Form
3) Third Normal Form

First Normal Form (1NF):

For a table to be in the First Normal Form, it should follow thefollowing 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.

Normalization of Blood Bank Database:

1) **Blood_Donor** *(bd_Id, bd_name, bd_phNo bd_sex, bd_age,bd_reg_date, bd_Bgroup, reco_ID, City_ID)*

{bd_Id} = > {bd_name} (functional dependency exists, because twodifferent bd_name do not correspond to the same bd_Id).

{bd_ID} = > {bd_sex} (functional dependency exists).

{bd_ID} = > {bd_age} (functional dependency exists).

{bd_ID} = > {bd_reg_date} date (functional dependency exists).

{bd_ID} = > {reco_id} (functional dependency exists).

{bd_ID} = > {city_id} (functional dependency exists).

{bd_ID} = > {bd_Bgroup} (functional dependency exists).

As the attributes of this table does not have sub attributes, it is in first normal form. Because every non-primary key attribute is fully functionally dependent on the primary key of the table and it is already in first normal form, this table is now in second normal form.Since the table is in second normal form and no non-primary key attribute is transitively dependent on the primary key, the table is now in 3NF.

2) **City** *(city_id , city_name)*

{city_id}= > {city_name}

The table is in first normal form. The table is second normal form. The table is in third normal form.

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 first normal form. The table is second normal form. The table is in third normal form.

4) **Blood_recipient** *(reci_Id, reci_sex, reci_phNo, reci_age, reci_date, reci_name, reci_Bqnty, reci_Bgrp, reco_id, city_id, m_id)*

{reci_Id} = > {reci_sex} (functional dependency exists).

{reci_Id} = > {reci_age} (functional dependency exists).

{reci_Id} = > {reci_date} (functional dependency exists).

{reci_Id} = > {reci_name} (functional dependency exists).

{reci_Id} = > {reci_bqnty} (functional dependency exists).

{reci_Id} = > {reci_Bgrp} (functional dependency exists).

{reci_Id} = > {reco_id} (functional dependency exists).

{reci_Id} = > {city_id} (functional dependency exists).

{reci_Id} = > {m_id} (functional dependency exists).

The table is in first normal form. The table is in second normal form. The table is in third normal form.

5) **Blood Specimen** *(b_group, specimen_no, status, dfind_id, m_id)*

{b_group, specimen _no} = > {status} (functional dependency exists).

{b_group, specimen _no} = > {dfind _id} (functional dependencyexists).

{b_group, specimen _no} = > {m_id} (functional dependency exists).

The table is in first normal form. The table is in second normal form. The table is in third normal form

6) **Disease_finder** *(dfind_id, dfind_name, dfind_PhNo)*

{ dfind_id } = > { dfind_name }

{ dfind_id } = > { dfind_PhNo } (functional dependency exists).

The table is in first normal form. The table is in second normal form. The table is in third normal form.

7) **BB_manager** *(M_id, m_name, m_phNo)*

{M_id} = >{m_name}

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

The table is in first normal form. The table is in second normal form. The table is in third normal form.

8) **Hospital_Info** *(hosp_Id, hosp_Name, hosp_phNo, hosp_needed_Bgrp, hosp_needed_qty, city_id, m_id)*

{hosp_Id}= > {hosp_Name, hosp_phNo, city_id, m_id}

{hosp_Id, hosp_needed_Bgrp } = > hosp_needed_qty (functionaldependency exists)

The table is in 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 secondnormal form. Hence we have to split the table.

Hospital_1 (hosp_Id, hosp_phNo, hosp_Name, city_id, m_id).

Hospital_2 (hosp_Id, hosp_needed_Bgrp, hosp_needed_qty) Now it is in second normal form. The table is in third normal form

TABLES AFTER NORMALIZATION:

1) **BB_Manager:**

```
$sqlite3 database.sdb < main.sql
101|shivank|9693959671
102|shwetanshu|9693959672
103|singh|9693959673
104|yusuf|9693959674
105|jackson|9693959675
106|akhil|9693959676
107|jojo|9693959677
108|stella|9693959678
109|monika|9693959679
110|himanshi|9693959680
```

2) **Blood_Donor:**

Result

```
$sqlite3 database.sdb < main.sql
150011|Mark|25|M|O+|2015-07-19|101412|1100
150012|Abdul|35|M|A-|2015-12-24|101412|1100
150013|Shivank|22|M|AB+|2015-08-28|101212|1200
150014|shweta|29|M|B+|2015-12-17|101212|1300
150015|Shyam|42|M|A+|2016-11-22|101212|1300
150016|Dan|44|F|AB-|2016-02-06|101212|1200
150017|Mike|33|M|B-|2016-10-15|101312|1400
150018|Elisa|31|F|O+|2016-01-04|101312|1200
150019|Carrol|24|F|AB+|2016-09-10|101312|1500
150020|shivansh|29|M|O-|2016-12-17|101212|1200
```

3) **Blood_Specimen:**

Result

```
$sqlite3 database.sdb < main.sql
1001|B+|1|11|101
1002|O+|1|12|102
1003|AB+|1|11|102
1004|O-|1|13|103
1005|A+|0|14|101
1006|A-|1|13|104
1007|AB-|1|15|104
1008|AB-|0|11|105
1009|B+|1|13|105
1010|O+|0|12|105
1011|O+|1|13|103
1012|O-|1|14|102
1013|B-|1|14|102
1014|AB+|0|15|101
```

4) **City:**

Result

```
$sqlite3 database.sdb < main.sql
1100|Dallas
1200|Austin
1300|Irving
1400|Houston
1500|Richardson
1600|Plano
1700|Frisco
1800|Arlington
1900|San Antonio
2000|Tyler
```

5) **Disease_Finder:**

Result

```
$sqlite3 database.sdb < main.sql
11|Peter|9693959681
12|Park|9693959682
13|Jerry|9693959683
14|shivam|9693959672
15|Monika|9693959679
16|Ram|9693959684
17|Swathi|9693959685
18|Gautham|9693959686
19|Ashwin|9693959687
20|Yash|9693959688
```

6) **Hospital_Info_1:**

Result

```
$sqlite3 database.sdb < main.sql
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
```

7) **Hospital_Info_2:**

Result

```
$sqlite3 database.sdb < main.sql
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
```

8) **Recipient:**

Result

```
$sqlite3 database.sdb < main.sql
10001|Peter|25|B+|1.5|101212|1100|101|M|2015-12-17
10002|shivank|60|A+|1.0|101312|1100|102|M|2015-12-16
10003|akhil|35|AB+|0.5|101312|1200|102|M|2015-10-17
10004|Parker|66|B+|1.0|101212|1300|104|M|2016-11-17
10005|jojo|53|B-|1.0|101412|1400|105|M|2015-04-17
10006|Preetham|45|O+|1.5|101512|1500|105|M|2015-12-17
10007|Swetha|22|AB-|1.0|101212|1500|101|F|2015-05-17
10008|Swathi|25|B+|2.0|101412|1300|103|F|2015-12-14
10009|Lance|30|A+|1.5|101312|1100|104|M|2015-02-16
10010|Marsh|25|AB+|3.5|101212|1200|107|M|2016-10-17
```

9) Recording_Staff:

Result

```
$sqlite3 database.sdb < main.sql
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
```

RELATIONAL SCHEMA AFTER NORMALIZATION:

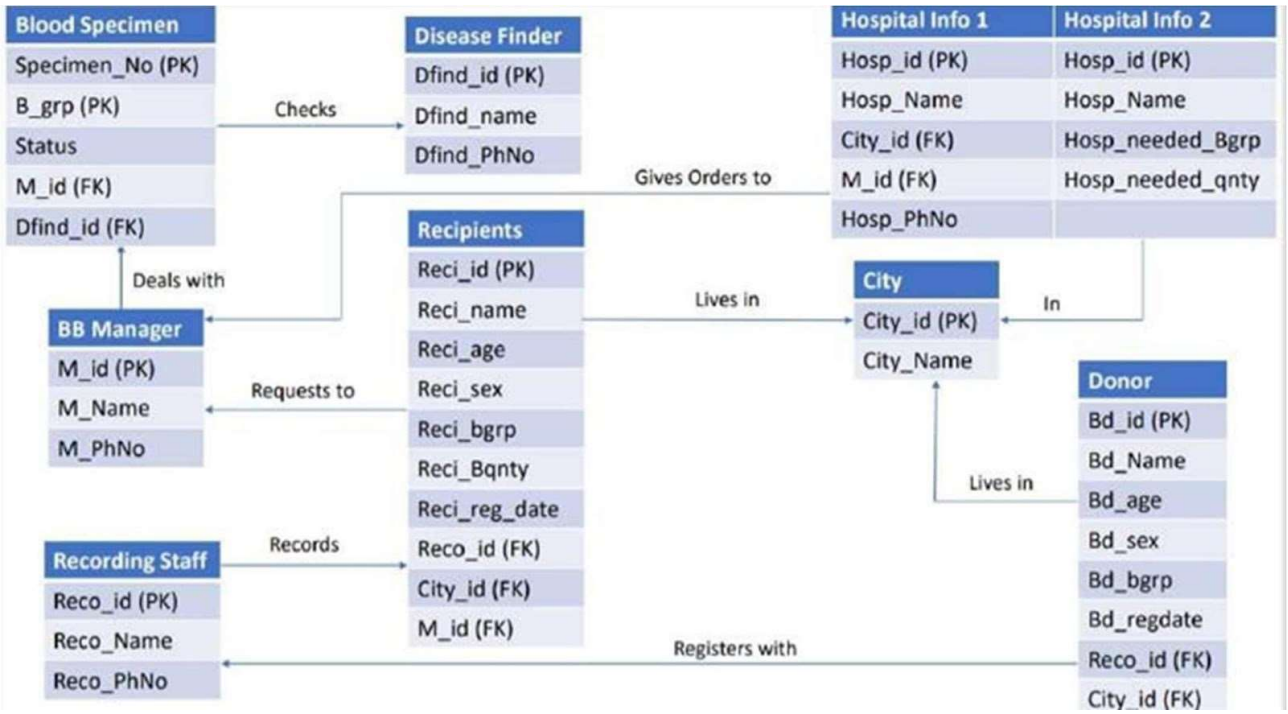


Figure 4: Relational Schema without Normalization

SQL IMPLEMENTATION:

The implementation on SQL Server is given below :-

BEGIN TRANSACTION;

```
CREATE TABLE BB_Manager
( M_id int NOT NULL PRIMARY KEY,
 mName varchar(100) NOT NULL,
 m_phNo bigint
);

INSERT into BB_Manager VALUES(101,'shivank',
9693959671),
(102,'shwetanshu', 9693959672),
(103,'singh', 9693959673),
(104,'yusuf', 9693959674),
(105,'jackson', 9693959675);
INSERT into BB_Manager
VALUES(106,'akhil', 9693959676),
(107,'jojo', 9693959677),
(108,'stella', 9693959678),
(109,'monika', 9693959679),
(110,'himanshi', 9693959680);

select * from BB_Manager;

CREATE TABLE Blood_Donor
( bd_ID int NOT NULL PRIMARY KEY,
 bd_name varchar(100) NOT NULL,bd_age
 varchar(100),
 bd_sex varchar(100), bd_Bgroup
 varchar(10),bd_reg_date date,
 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)
);
INSERT into Blood_Donor VALUES(150011,'Mark',25,'M','O+','2015-07-
19',101412,1100),
(150012,'Abdul',35,'M','A-','2015-12-24',101412,1100),
(150013,'Shivank',22,'M','AB+','2015-08-28',101212,1200),
(150014,'shweta',29,'M','B+','2015-12-17',101212,1300),
(150015,'Shyam',42,'M','A+','2016-11-22',101212,1300),
(150016,'Dan',44,'F','AB-','2016-02-06',101212,1200),
(150017,'Mike',33,'M','B-','2016-10-15',101312,1400),
(150018,'Elisa',31,'F','O+','2016-01-04',101312,1200),
```

(150019,'Carrol',24,'F','AB+','2016-09-10',101312,1500),
(150020,'shivansh',29,'M','O-','2016-12-17',101212,1200);

select * from Blood_Donor;

bd_ID	bd_name	bd_age	bd_sex	bd_Bgroup	bd_reg_date	reco_ID	City_ID
150011	Mark	25	M	O+	2015-07-19	101412	1100
150012	Abdul	35	M	A-	2015-12-24	101412	1100
150013	Shivank	22	M	AB+	2015-08-28	101212	1200
150014	shweta	29	M	B+	2015-12-17	101212	1300
150015	Shyam	42	M	A+	2016-11-22	101212	1300
150016	Dan	44	F	AB-	2016-02-06	101212	1200
150017	Mike	33	M	B-	2016-10-15	101312	1400
150018	Elisa	31	F	O+	2016-01-04	101312	1200
150019	Carrol	24	F	AB+	2016-09-10	101312	1500
150020	shivansh	29	M	O-	2016-12-17	101212	1200

CREATE TABLE BloodSpecimen
(specimen_number int NOT NULL,b_group
 varchar(10) NOT NULL, status int,
 dfind_ID int NOT NULL,M_id
 int NOT NULL,
primary key (specimen_number,b_group),
FOREIGN KEY(M_id) REFERENCES Manager(M_id),
FOREIGN KEY(dfind_ID) REFERENCES DiseaseFinder(dfind_ID)
);
INSERT into BloodSpecimen VALUES(1001,
'B+', 1,11,101),
(1002, 'O+', 1,12,102),
(1003, 'AB+', 1,11,102),
(1004, 'O-', 1,13,103),
(1005, 'A+', 0,14,101),
(1006, 'A-', 1,13,104),
(1007, 'AB-', 1,15,104),
(1008, 'AB-', 0,11,105),
(1009, 'B+', 1,13,105),
(1010, 'O+', 0,12,105),
(1011, 'O+', 1,13,103),
(1012, 'O-', 1,14,102),
(1013, 'B-', 1,14,102),
(1014, 'AB+', 0,15,101);

Select * from BloodSpecimen;

specimen_number	b_group	status	dfind_ID	M_id
1001	B+	1	11	101
1002	O+	1	12	102
1003	AB+	1	11	102
1004	O-	1	13	103
1005	A+	0	14	101
1006	A-	1	13	104
1007	AB-	1	15	104
1008	AB-	0	11	105
1009	B+	1	13	105
1010	O+	0	12	105
1011	O+	1	13	103
1012	O-	1	14	102
1013	B-	1	14	102
1014	AB+	0	15	101

```
CREATE TABLE City
( City_ID int NOT NULL PRIMARY KEY,
 City_name varchar(100) NOT NULL
);
INSERT into City
VALUES(1100,'Dallas'),
(1200,'Austin'),
(1300,'Irving'),
(1400,'Houston'),
(1500,'Richardson'); INSERT
into City
VALUES(1600,'Plano'),
(1700,'Frisco'),
(1800,'Arlington'),
(1900,'San Antonio'),
(2000,'Tyler');

select * from City;
```

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

```
CREATE TABLE DiseaseFinder
( dfind_ID int NOT NULL PRIMARY KEY,dfind_name
  varchar(100) NOT NULL, dfind_PhNo bigint
);
INSERT into DiseaseFinder
VALUES(11,'Peter',9693959681),
(12,'Park',9693959682),
(13,'Jerry',9693959683),
(14,'shivam',9693959672),
(15,'Monika',9693959679);
INSERT into DiseaseFinder
VALUES(16,'Ram',9693959684),
(17,'Swathi',9693959685),
(18,'Gautham',9693959686),
(19,'Ashwin',9693959687),
(20,'Yash',9693959688);

select * from DiseaseFinder;
```

dfind_ID	dfind_name	dfind_PhNo
11	Peter	9693959681
12	Park	9693959682
13	Jerry	9693959683
14	shivam	9693959672
15	Monika	9693959679
16	Ram	9693959684
17	Swathi	9693959685
18	Gautham	9693959686
19	Ashwin	9693959687
20	Yash	9693959688

```
CREATE TABLE Hospital_Info_1(
hosp_ID int NOT NULL,
 hosp_name varchar(100) NOT NULL,City_ID int NOT
 NULL,
 M_id int NOT NULL,
 primary key(hosp_ID),
 FOREIGN KEY(M_id) REFERENCES Manager(M_id),FOREIGN
 KEY(City_ID) REFERENCES City(City_ID)
);
INSERT into Hospital_Info_1
VALUES(1,'MayoClinic',1100,101),
(2,'CleavelandClinic',1200,103),
(3,'NYU',1300,103);
INSERT into Hospital_Info_1
VALUES(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);

select * from Hospital_Info_1;
```

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

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

(9,'Pinecreek','AB-',20); select * from
Hospital_Info_2;

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

```
CREATE TABLE Recipient
( reci_ID int NOT NULL PRIMARY kEY,reci_name
 varchar(100) NOT NULL, reci_age varchar(10),
 reci_Brgp varchar(100),
 reci_Bqnty float, reco_ID int
 NOT NULL,City_ID int NOT
 NULL, M_id int NOT NULL,
 FOREIGN KEY(M_id) REFERENCES Manager(M_id),FOREIGN
 KEY(City_ID) REFERENCES City(City_ID)
);
Alter table Recipient

ADD reci_sex varchar(100);

Alter table Recipient ADD
reci_reg_date date;INSERT into
Recipient
VALUES(10001,'Peter',25,'B+',1.5,101212,1100,101,'M','2015-12-17'),
(10002,'shivank',60,'A+',1,101312,1100,102,'M','2015-12-16'),
(10003,'akhil',35,'AB+',0.5,101312,1200,102,'M','2015-10-17'),
(10004,'Parker',66,'B+',1,101212,1300,104,'M','2016-11-17'),
(10005,'jojo',53,'B-',1,101412,1400,105,'M','2015-04-17'),
(10006,'Preetham',45,'O+',1.5,101512,1500,105,'M','2015-12-17'),
(10007,'Swetha',22,'AB-',1,101212,1500,101,'F','2015-05-17');
INSERT into Recipient
```

VALUES(10008,'Swathi',25,'B+',2,101412,1300,103,'F','2015-12-14'),
(10009,'Lance',30,'A+',1.5,101312,1100,104,'M','2015-02-16'),
(10010,'Marsh',25,'AB+',3.5,101212,1200,107,'M','2016-10-17');

select * from Recipient;

reci_ID	reci_name	reci_age	reci_Brgp	reci_Bqnty	reco_ID	City_ID	M_id	reci_sex	reci_reg_date
10001	Peter	25	B+	1.5	101212	1100	101	M	2015-12-17
10002	shivank	60	A+	1	101312	1100	102	M	2015-12-16
10003	akhil	35	AB+	0.5	101312	1200	102	M	2015-10-17
10004	Parker	66	B+	1	101212	1300	104	M	2016-11-17
10005	jojo	53	B-	1	101412	1400	105	M	2015-04-17
10006	Preetham	45	O+	1.5	101512	1500	105	M	2015-12-17
10007	Swetha	22	AB-	1	101212	1500	101	F	2015-05-17
10008	Swathi	25	B+	2	101412	1300	103	F	2015-12-14
10009	Lance	30	A+	1.5	101312	1100	104	M	2015-02-16
10010	Marsh	25	AB+	3.5	101212	1200	107	M	2016-10-17

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

select * from Recording_Staff;

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

SAMPLE SQL QUERIES:

1) Create a View of recipients and donors names having the same blood group registered on the same date andthe name of recording staff name.

-- Sample Query - 1 --

```
CREATE VIEW Blood_Recipient_SameBGrp AS
select Blood_Donor.bd_name,Recipient.reci_name,reco_Name fromRecording_Staff
inner join Blood_Donor on Recording_Staff.reco_ID =
Blood_Donor.reco_ID
inner join Recipient on Recording_Staff.reco_ID = Recipient.reco_IDwhere
Blood_Donor.bd_Bgroup = Recipient.reci_Brgp and Blood_Donor.bd_reg_date =
Recipient.reci_reg_date;
select* from Blood_Recipient_SameBGrp;
```

OUTPUT:

bd_name	reci_name	reco_Name
shweta	Peter	Walcot

2) Show the blood specimen verified by disease findershivam which are pure (status=1).

-- Sample Query - 2 --

```
Select specimen_number,b_group from BloodSpecimen,DiseaseFinderWHERE
BloodSpecimen.dfind_ID= DiseaseFinder.dfind_ID AND dfind_name='shivam' AND
status=1;
```

OUTPUT:

specimen_number	b_group
1012	O-
1013	B-

3) Show the pure blood specimen handled by BB_Managerwho also handles a recipient needing the same blood group along with the details of the BB_Manager and Recipient.

-- Sample Query - 3 --

```
select BB_Manager.M_id,mName,Recipient.reci_name,
Recipient.reci_Brgp,BloodSpecimen.b_group from
BB_Manager,Recipient,BloodSpecimen
where Recipient.M_id = BloodSpecimen.M_id and Recipient.reci_Brgp =
BloodSpecimen.b_group and Recipient.M_id = BB_Manager.M_idand status = 1;
```

OUTPUT:

M_id	mName	reci_name	reci_Brgp	b_group
101	shivank	Peter	B+	B+
102	shwetanshu	akhil	AB+	AB+

4) Show the donors having the same blood groups requiredby the recipient staying in the same city along withrecipient details.

-- Sample Query - 4 --

```
Select bd_ID,bd_name,reci_ID,reci_name FROMBlood_Donor,Recipient
WHERE bd_Bgroup=reci_Brgp AND Blood_Donor.City_ID=Recipient.City_ID;
```

OUTPUT:

bd_ID	bd_name	reci_ID	reci_name
150013	Shivank	10003	akhil
150013	Shivank	10010	Marsh
150014	shweta	10004	Parker
150014	shweta	10008	Swathi
150017	Mike	10005	jojo

5) Display the information of Hospital_Info_1 handled byBB_Manager whose ID is 103:

-- Sample Query - 5--

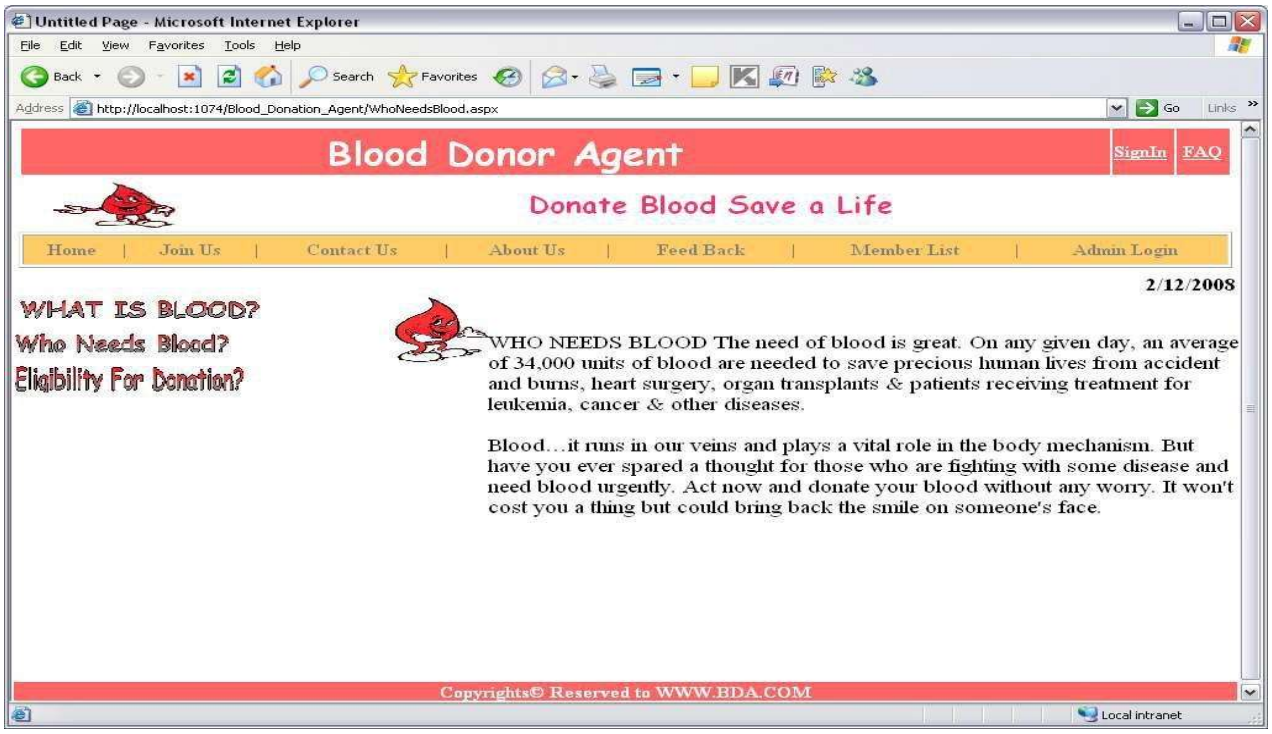
```
Select hosp_ID,hosp_name , City_ID, HOspital_Info_1.M_id from
Hospital_Info_1,BB_Manager
where BB_Manager.M_id=Hospital_Info_1.M_id andBB_Manager.M_id=103;
```

OUTPUT:

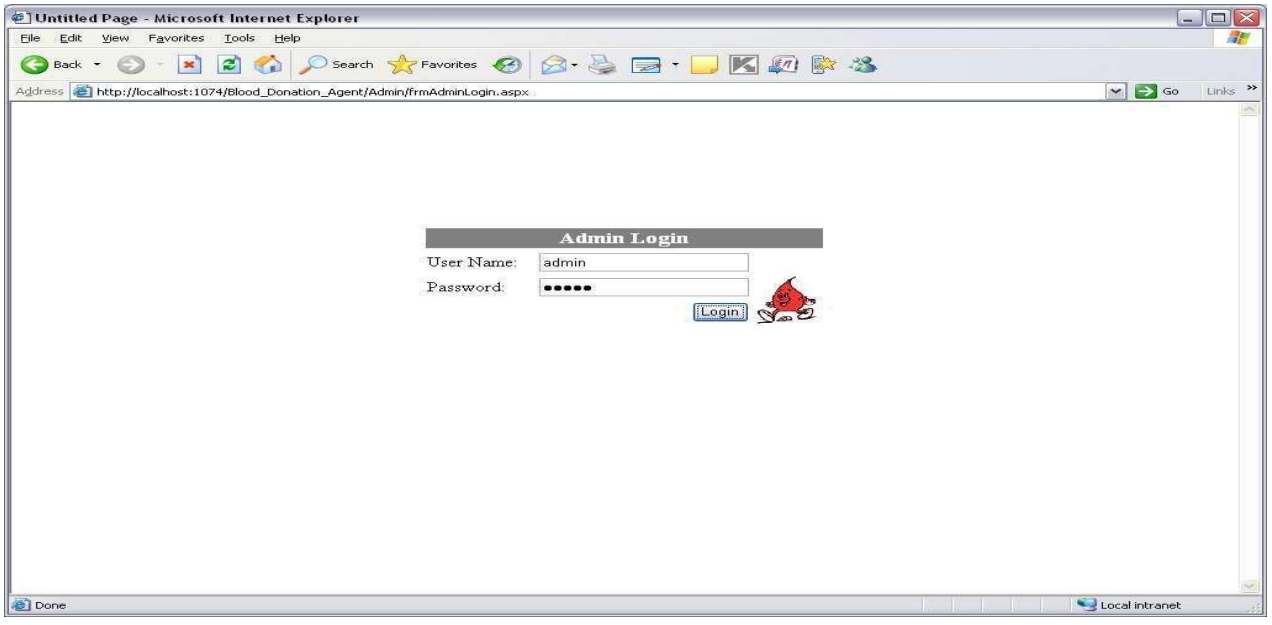
hosp_ID	hosp_name	City_ID	M_id
2	CleavelandClinic	1200	103
3	NYU	1300	103
5	Charlton	1800	103

FRONT END DESIGN

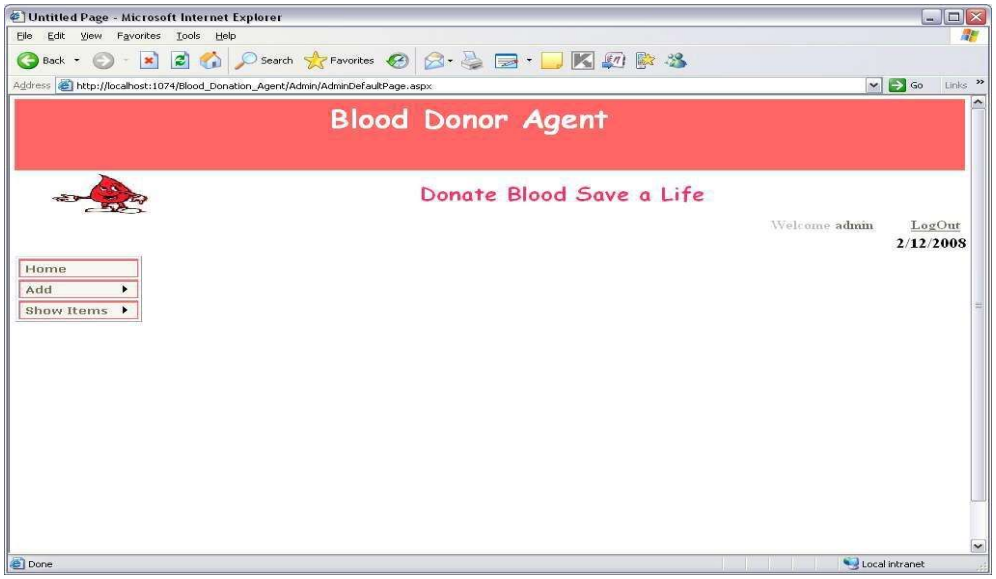
HOME PAGE:



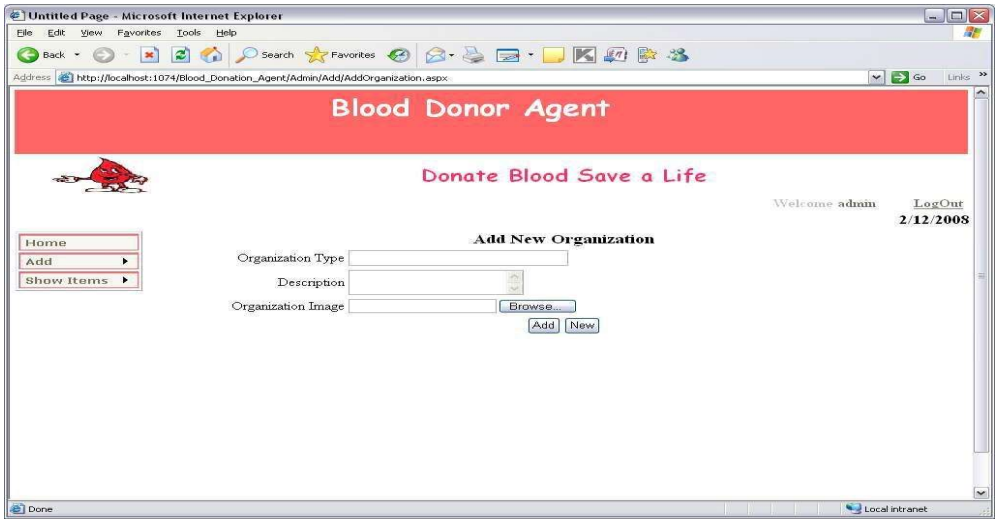
ADMIN LOGIN:



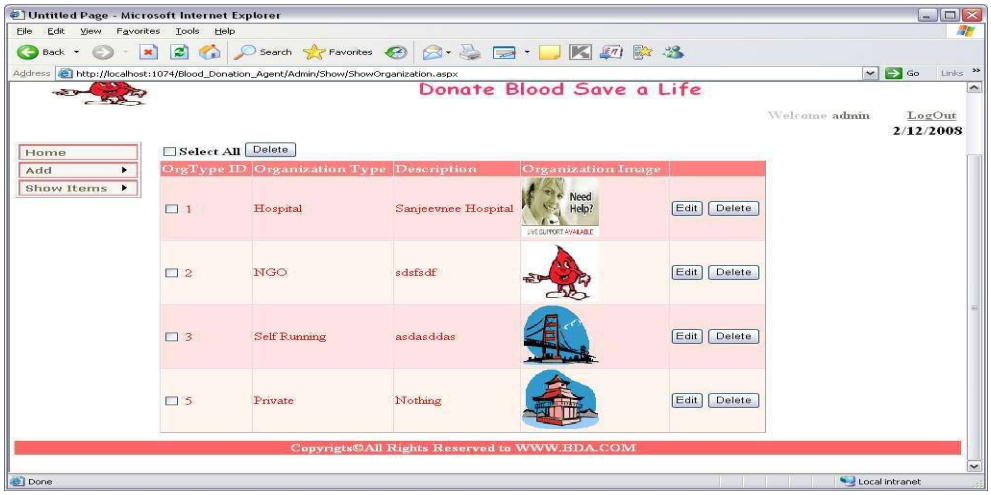
ADMIN HOME:



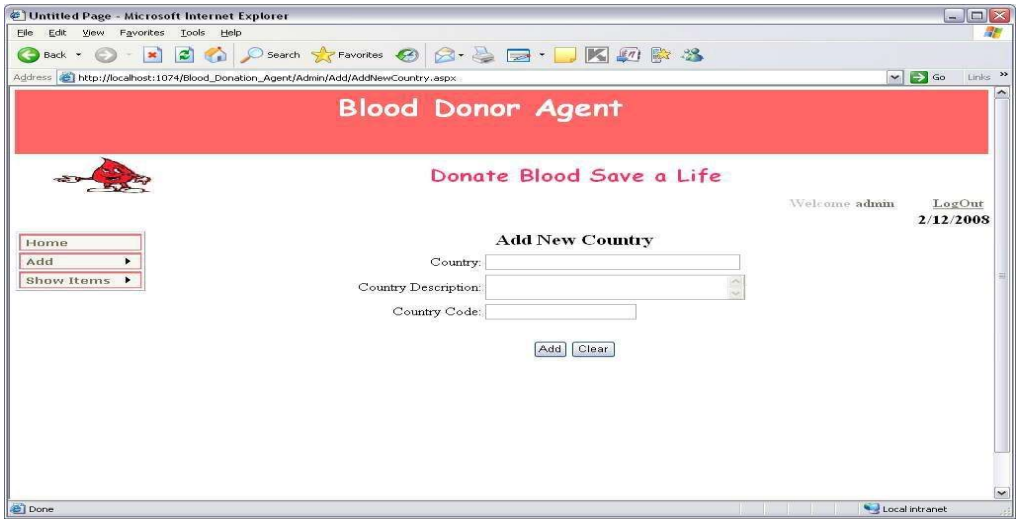
ADD NEW ORGANIZATION:



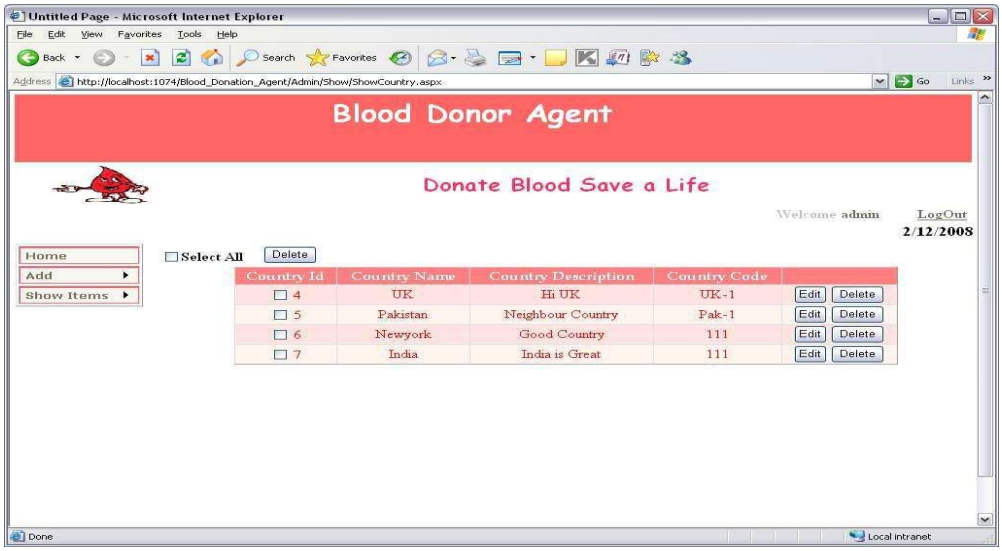
RETRIEVING ORGANIZATION INFORMATION:



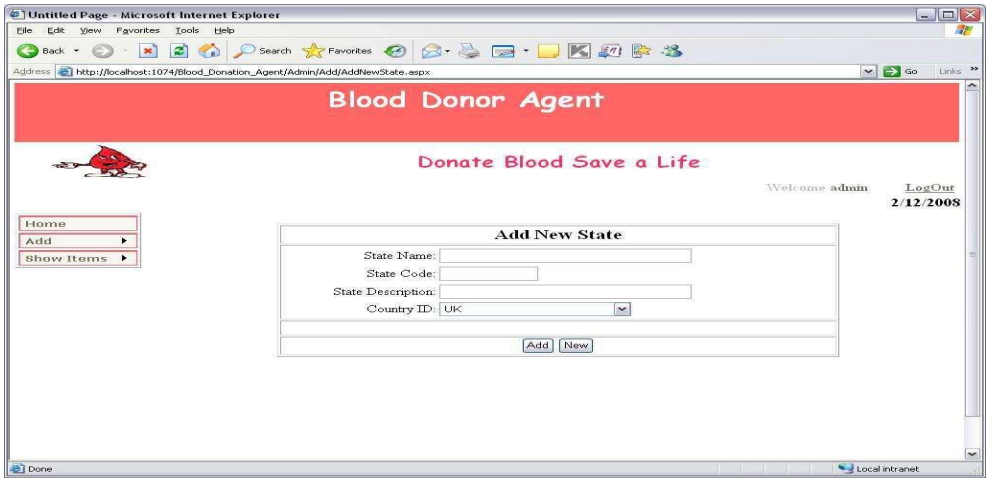
ADD NEW COUNTRY:



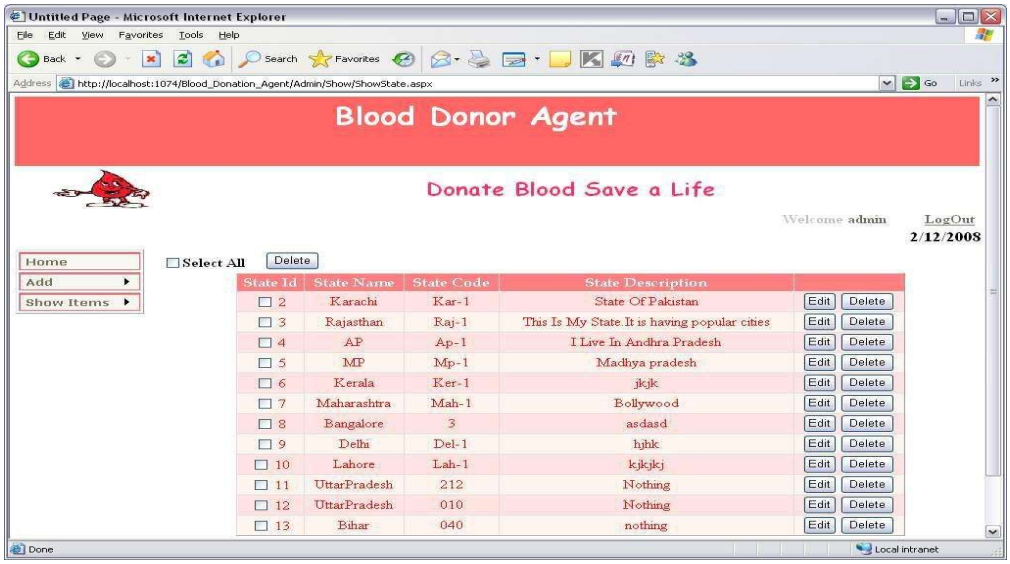
COUNTRY INFORMATION:



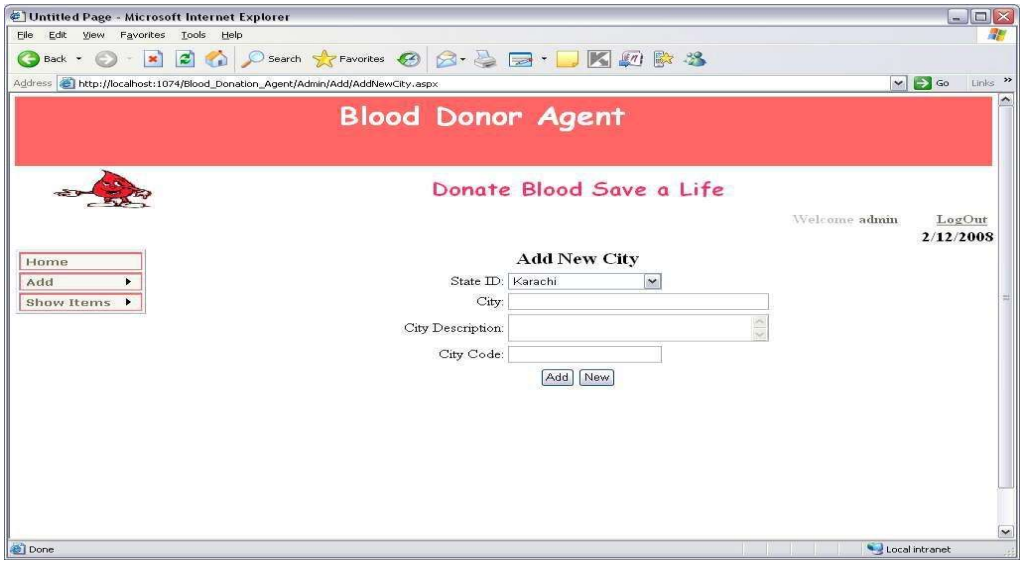
ADD NEW STATE:



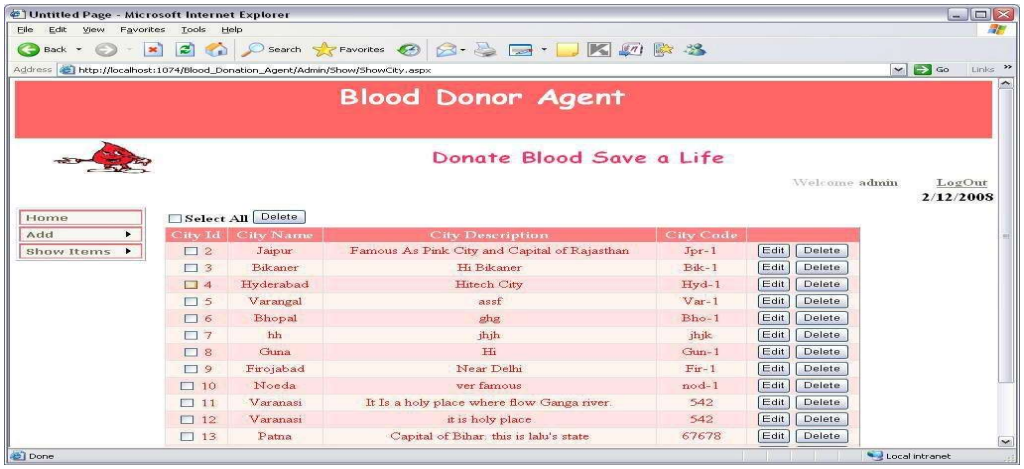
STATE INFORMATION:



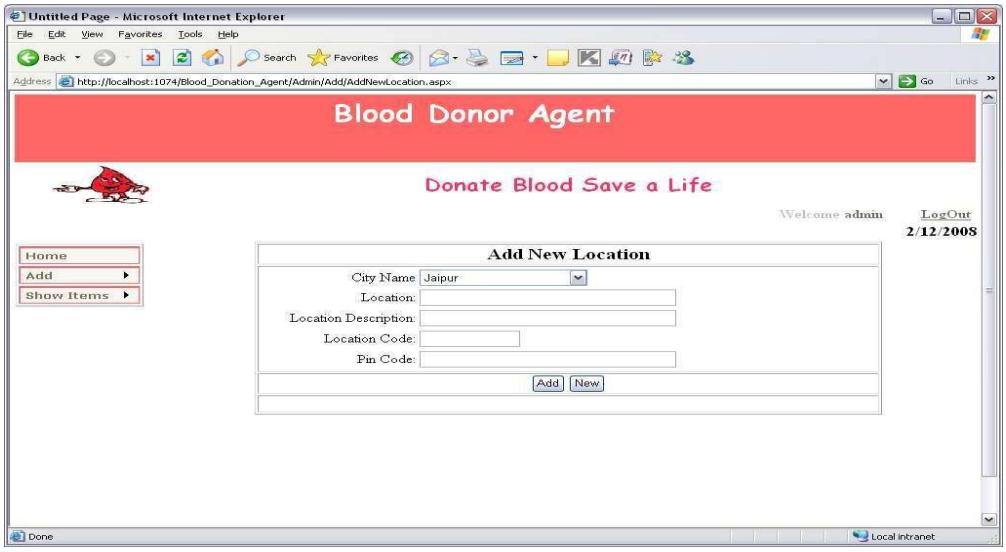
ADD NEW CITY:



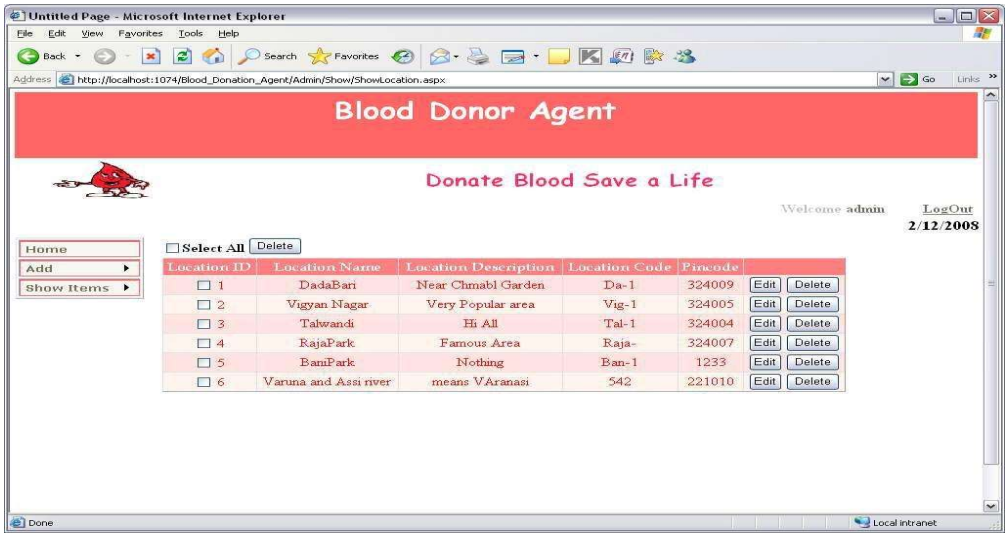
CITY INFORMATION:



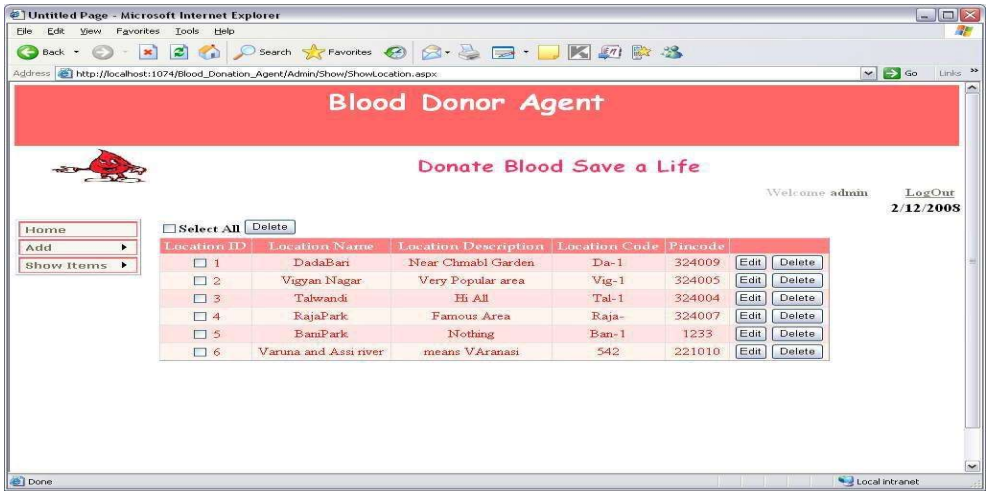
ADD NEW LOCATION:



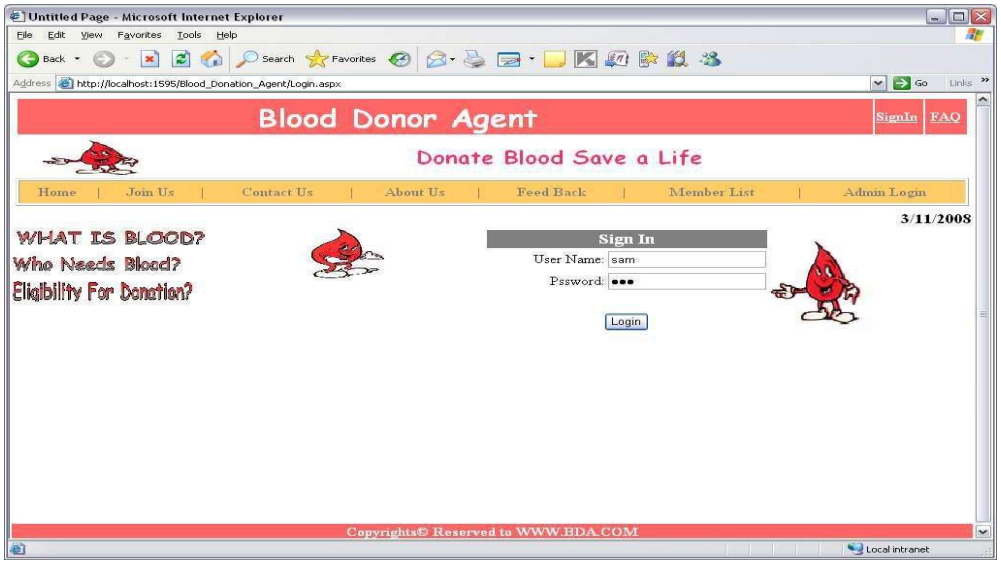
LOCATION INFORMATION:



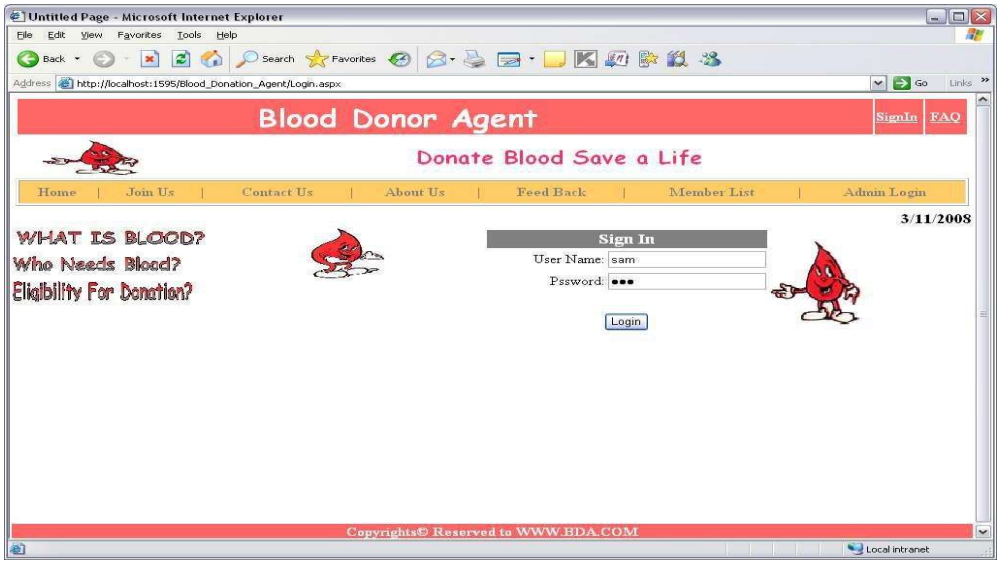
REGISTRATION FORM FOR ALL USERS:



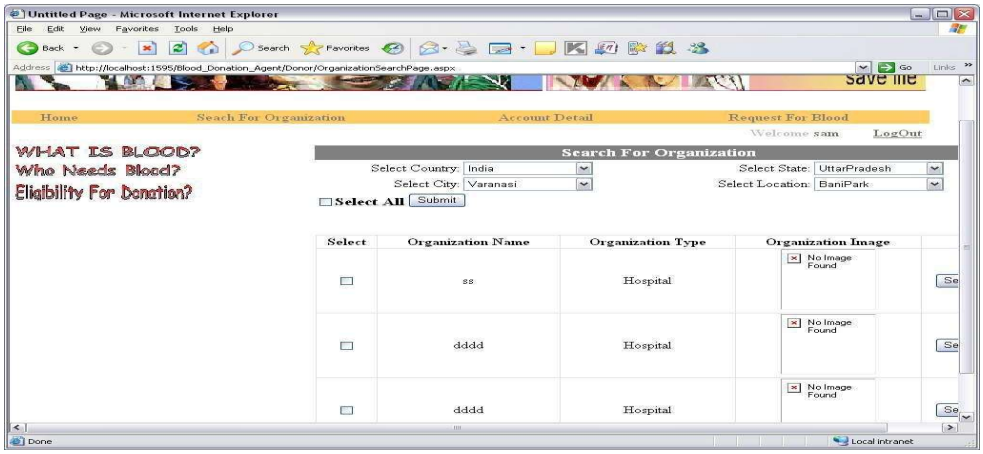
LOGIN FORM:



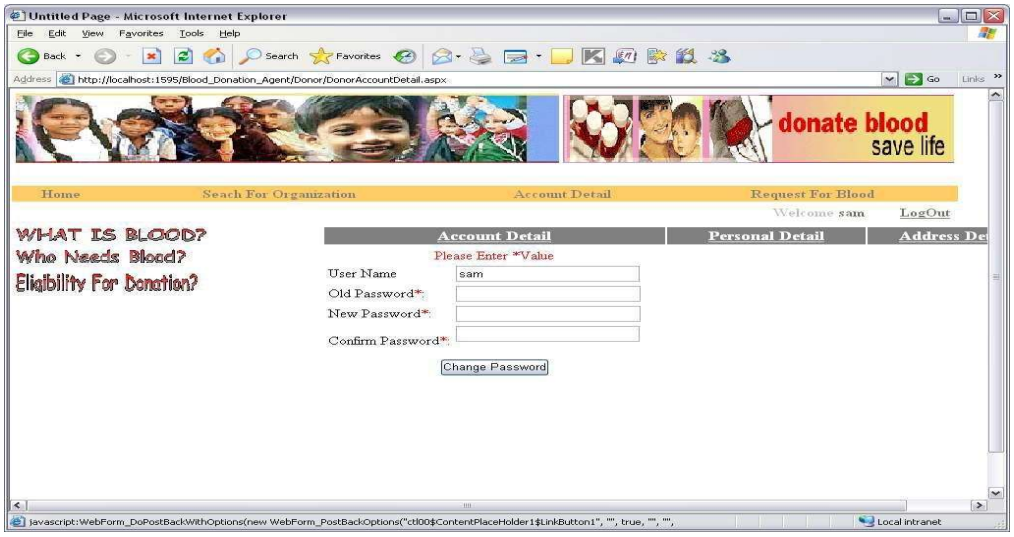
USER HOME:



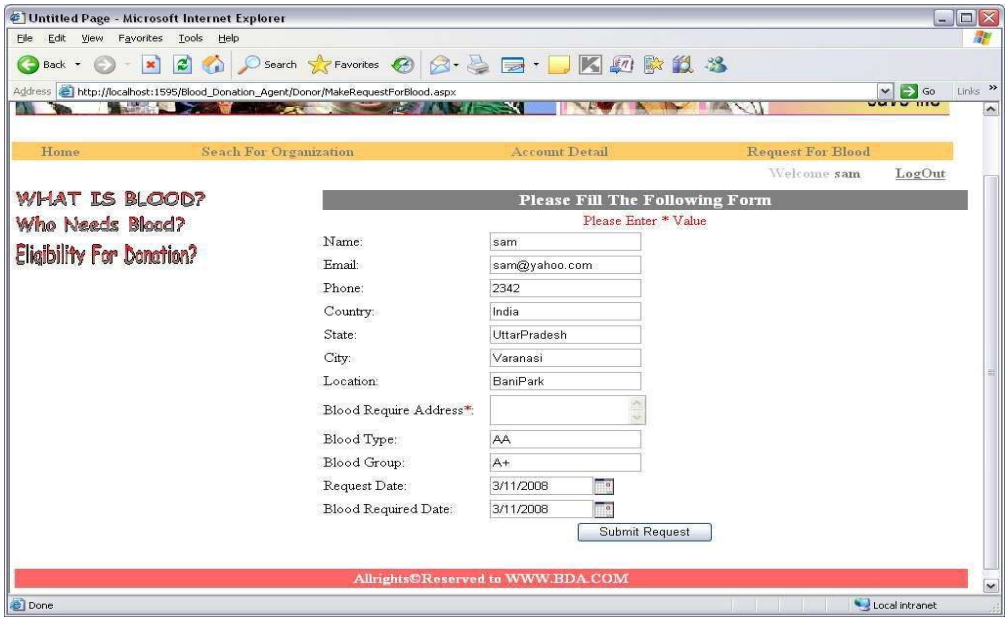
SEARCH FOR DONOR:



UPDATE ACCOUNT DETAILS:



REQUEST FOR BLOOD:



BACK END DESIGN

CODING:

WEBCONFIG FILE:

```
<?xml version="1.0"?>
<!--
    Note: As an alternative to hand editing this file you can use
the
                                    web admin tool to configure
                                    settings for your application.
                                    Use the Website->Asp.Net
                                    Configuration option in Visual
                                    Studio.
                                    A full list of settings and
-->                                 comments can be found in
                                    machine.config.comments
                                    usually located in
                                    \Windows\Microsoft.Net\Framework\v2.x\Co
                                    nfig
<configuration>
      <appSettings>
            <add key="ConnStr" value="data source=RAMYA-
2DCA5B123;database=BloodBequeathFederalAgent;integrate
d security=sspi"/>
      </appSettings>
      <connectionStrings>
  <add name="BloodDonationAgentConnectionString"
connectionString="Data Source=RAMYA-2DCA5B123;Initial
Catalog=BloodDonationAgent;integrated security=sspi"
   providerName="System.Data.SqlClient" />
 </connectionStrings>
      <system.web>
            <!--
             Set compilation debug="true" to insert debugging
             symbols into the compiled page. Because this
             affects performance, set this value to true only
             during development.
        -->
```

```
<compilation debug="true">
```

```
                <assemblies>
                    <add assembly="System.Design, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=B03F5F7F11D50A3A"/>
                    <add assembly="System.Web.Extensions,
Version=1.0.61025.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
                    <add assembly="System.Web.Extensions.Design,
Version=1.0.61025.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
<add assembly="System.Windows.Forms, Version=2.0.0.0,
Culture=neutral,
PublicKeyToken=B77A5C561934E089"/></assemblies></compilation>
          <!--
            The <authentication> section enables
            configuration of the security authentication mode
            used by ASP.NET to identify an incoming user.
        -->
          <authentication mode="Windows"/>
          <!--
            The <customErrors> section enables configuration of
            what to do if/when an unhandled error occurs during
            the execution of a request. Specifically, it
            enables developers to configure html error pages to
            be displayed in place of a error stack trace.

        <customErrors mode="RemoteOnly"
defaultRedirect="GenericErrorPage.htm">
            <error statusCode="403" redirect="NoAccess.htm" />
            <error statusCode="404" redirect="FileNotFound.htm" />
        </customErrors>
        -->
    </system.web>
</configuration>
```

USER LOGIN FORM:

```
using System;
using
System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class Login : System.Web.UI.Page
{
    CheckUser user = new CheckUser();
    UserAccountBusinessLayer account = new UserAccountBusinessLayer();
    OrganizationAccountBusinessLayer org = new
OrganizationAccountBusinessLayer();
    protected void Page_Load(object sender, EventArgs e)
    {
        txtUsername.Focus();
    }

   protected void btnLogin_Click(object sender, EventArgs e)
    {
        try
        {
            user.Username = txtUsername.Text;
            user.Password = txtPassword.Text;
            //Check User

            if (user.GetUser() == true)
            {
                account.Username = txtUsername.Text;
                DataSet ds = new DataSet();
                ds = account.GetAccountId();
```

```
                string AcId = ds.Tables[0].Rows[0][0].ToString();
                Session["username"] = txtUsername.Text;
                Session["Acid"] = AcId;
                DataSet ds1 = new DataSet();
                account.Accountid =int.Parse(AcId);
                ds1 = account.GetAddressId();
                Session["addid"] = ds1.Tables[0].Rows[0][0].ToString();
                Response.Redirect("~/Donor/DonorHome.aspx");
            }
            else
                Image2.Visible = true;
                                        lblMsg.Text = "Your Login Attempt Is Failed
                                        Plz try
Again    !";
                                        txtPassword.
                                        Text = "";
                                        txtUsername.
                                        Focus();
                                        //Checking Organization
                                        if (user.GetOrganization() == true)
                                        {
                account.Username = txtUsername.Text;
                DataSet ds = new DataSet();
                ds = account.GetAccountId();
                string AcId = ds.Tables[0].Rows[0][0].ToString();
                Session["username"] = txtUsername.Text;
                Session["Acid"] = AcId;
                DataSet ds1 = new DataSet();
                org.Orgid =int.Parse(AcId);
                ds1 = org.GetOrgAddressId();
                Session["addid"]=ds1.Tables[0].Rows[0][0].ToString();
                Response.Redirect("~/Organization/OrganizationHome.aspx");
            }
            else
                Image2.Visible = true;
            lblMsg.Text = "Your Login Atempt Is Failed Plz try Again    !";
            txtPassword.Text = "";
            txtUsername.Focus();
```

```
//Employee Checking
if (user.CheckEmployee() == true)
{
    account.Username = txtUsername.Text;
    DataSet ds = new DataSet();
    ds = account.GetAccountId();
    string AcId = ds.Tables[0].Rows[0][0].ToString();
    Session["username"] = txtUsername.Text;
    Session["Acid"] = AcId;
    Response.Redirect("~/CallCenter/CallCenterHome.aspx")
    ;
}
```

USECASE DIAGRAM

The use case diagram for blood bank management system is give below:

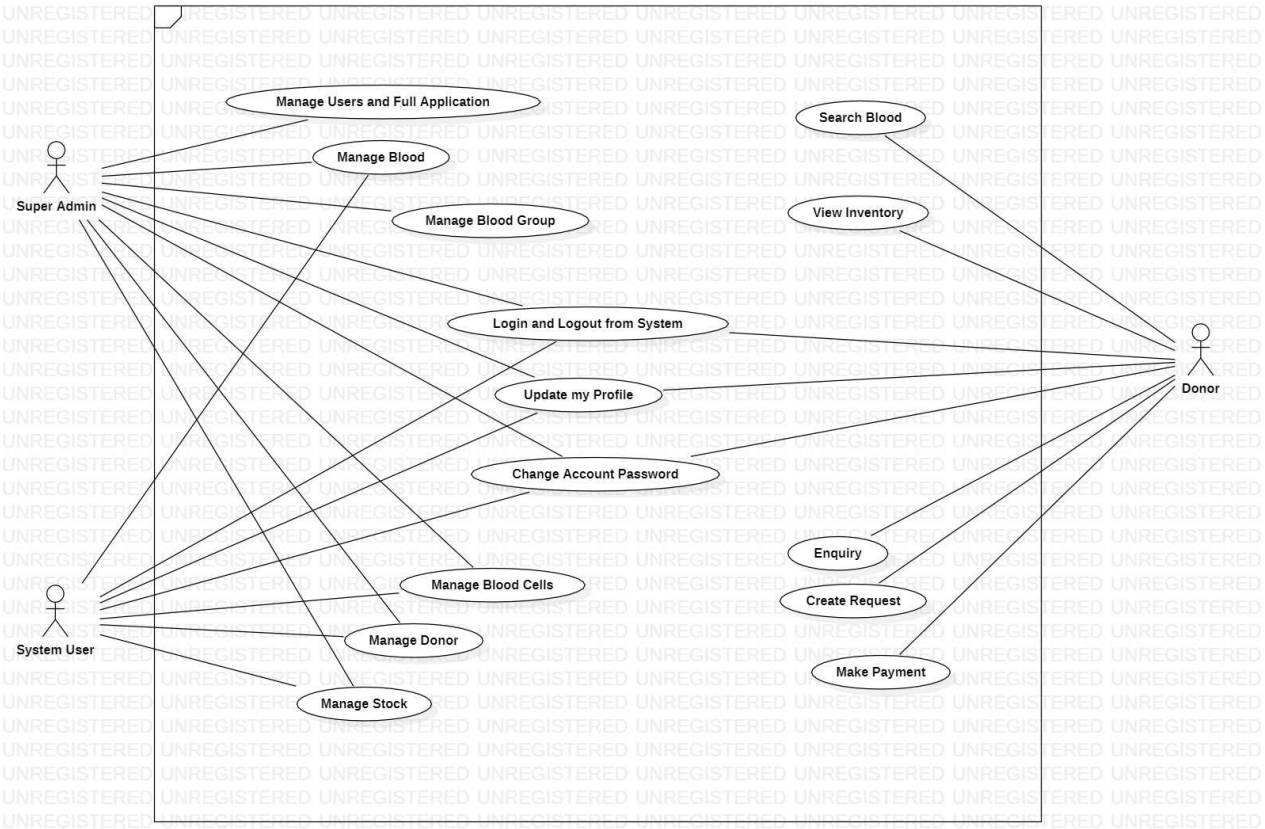


Figure 5: Use Case Diagram

CONCLUSION

Prior to this project, a comprehensive study of the blood bank management system was carried out using recent research by a number of authors. Facts were gathered, which assisted in identifying the system's shortcomings.

In order to meet the requirements of a more advanced system, a solution was developed after these issues had been thoroughly examined. The centralized blood bank repository, also known as this system, was instrumental in resolving all of the issues that plagued the previous ones. Patients, donors, hospitals, and blood banks will all be able to benefit from a plethora of features and information thanks to this system, making blood donation and reception significantly faster and easier.

In the design phase, prior to putting the database into use, we looked at a variety of blood bank characteristics and operations to determine the necessary entities, attributes, and relationships between entities for an effective Entity Relationship Diagram (ERD). We have created an ERD, converted it to a relational model, normalized the tables, and analysed all of the requirements.

The tables for our database have been created and some sample values have been inserted into the tables using SQL Server. Last but not least, we've run some sample queries on the database to see how well it works to quickly and accurately retrieve useful information.

REFERENCES

1) (PDF) blood-bank-management-system July 2021 7471122569 6713106 (researchgate.net) [Accessed 18th April,2023]

2) https://www.ijcrt.org/papers/IJCRT2105420.pdf [Accessed 18 April,2023]

3) RESEARCH PAPER BLOOD BANK MANAGEMENT SYSTEM (studylib.net) [Accessed 9th April,2023]

4) https://ieeexplore.ieee.org/document/8515789 [Accessed 18 April,2023]

5) Database management systems for blood bank applications - PubMed (nih.gov) [Accessed 22nd April,2023]

6) https://ijarsct.co.in/Paper5977.pdf [Accessed 7th April,2023]