Ahmedabad
University

CSE250 Database Management System and CSE251 Database Management System Lab

4th Sem B.Tech ICT

DBMS PROJECT REPORT

ORGAN DONATION AND TRANSPLANTATION

MANAGEMENT NETWORK SYSTEM

MADE BY:-

VARSHIL SHAH-AU1841095

MEET KADIYA-AU1841099

YASH R PATEL-AU1841125

DESCRIPTION OF PROJECT

- ## Problem Statement:-

Organ Transplantation is one of the major life-saving techniques utilized by medical professionals nowadays. It is the process in which an organ is removed from one body through surgery and placed in the body of a recipient who is in need of that, to replace a damaged or a missing organ. There are two types of persons involved in it: Donors and Recipients. They may be at the same location or organs may be transported from one location to another subject to distance, time and other risks.

For all this, a proper management system is essential. Organ Donation And Transplantation Management Network System plays a vital role in today's medical institutions which are responsible for the evaluation, testing, and procurement of organs for organ transplantation. The main goal of these organizations aims to identify the best candidates for the available organs and to coordinate it with the medical institutions and hospitals to decide on each organ recipient. They also conduct awareness programs for the importance of donating organs and also keeps the track record of all the transplantations carried to date.

This management and network system is a database management system that uses database technology to keep, construct, develop, maintain and manipulate various kinds of data about a person's donation or procurement of a particular organ. It maintains a comprehensive medical history and other critical

information like blood group, age, gender, location, etc, which can affect the donation process of every person(i.e. Both donor and recipient in the database design. In short, it contains all the data and statistical information regarding the network of organ donation and procurement of different patients and donors.

• Motivation:-

The motivation behind this project is a severe issue to be handled by the country. The situation of organ wastage is a major issue in our country and mostly in the case of heart. For this, it becomes utmost essential to have a centrally administered proper database of all patients, donors, and doctors in a well-formed way that can be processed easily. So, we aim to create a solution that effectively deals with the problems of finding donors and also providing statistical data of the transplants that can help the government to form better rules and regulations.

• Flow and System Requirement Specification:-

The flow first starts with the user. The user needs to register his data for either donating or receiving. Records of donors and patients are created when a person donates or procures an organ through a medical institution like hospitals. Personal Records may include some of the following information:-

1.Personal Information

2.Medical History

3.Allergies to any medicine (If any)

4.Medical Insurance (If any)

5.The need for an organ presently or donating after some duration

6.Medical Insurance of any private or government insurers specific.

7.Address of the user, City, State

8.User ID of Donor, Patient, Doctor

This record will help serve a variety of purposes and is critical to justify the proper functioning of the Organ Donation and Procurement Management Network System, especially in today's complicated health care environment in which we can see the world suffering through such global pandemics. These records provide statistical information regarding the number of organs needed and available at a particular point in time.it is essential for planning, evaluating and coordinating organ donation and procurement. Every user has an account that can only be registered by a government certified hospital, which will keep all the information as defined in the problem statement. Only Hospitals are eligible to request a donation or procurement transaction. Government organizations will keep a watch on the pairing of donors and patients and can approve a transplantation operation if all the rules are satisfied. Collecting Statistical Data through the history of Transplantation Transaction.

• Technologies Used:-

1. **JAVA**
2. **ORACLE**
3. **SWING GUI**
4. **NETBEANS**

ENTITY-RELATIONSHIP DIAGRAM

- ## ER ANALYSIS:-Identifying Entity Sets and Relationship Sets:-

A. ENTITY SETS:-

1) **User1**
   - ➔ User ID
   - ➔ Name
   - ➔ Date Of Birth
   - ➔ Medical Insurance
   - ➔ Medical History
   - ➔ Street
   - ➔ City
   - ➔ State
2) **User Phone No.**
   - ➔ User ID
   - ➔ Phone No.
3) **Organization**
   - ➔ Organization ID
   - ➔ Organization Name
   - ➔ Location
   - ➔ Government Approval
4) **Doctor**
   - ➔ Doctor ID

➔ Doctor Name

➔ Department Name

➔ Organization ID

5) Patient

➔ Patient ID

➔ Organ Required

➔ Reason Of Procurement

➔ Doctor ID

➔ User ID

6) Donor

➔ Donor ID

➔ Organ Donated

➔ Reason/Cause Of Donation

➔ Organization ID

➔ User ID

➔ Temporary Booked Organ

7) Organ Available

➔ Organization ID

➔ Organization Name

➔ Donor ID

8) Transaction

➔ Patient_ID

➔ Organ_ID

➔ Donor_ID

➔ Date Of Transaction

➔ Status

9) **Organization Phone No.**

➔ Organization ID

➔ Phone No.

10) **Doctor Phone No.**

➔ Doctor ID

➔ Phone No

11) **Organization Head**

➔ Organization ID

➔ Employee ID

➔ Name

➔ Date Of Joining

➔ Term Length

12) **Log**

➔ Query Time

➔ Comments

➔ Newly Added

13) **Login**

➔ Username

➔ Password

14) **Death Records**

➔ Name

➔ Reason Of Death

➔ Medical History

➔ Medical Insurance

➔ City

➔ Street

➔ State

➔ Date Of Birth

➔ Donation Status

➔ Perspective

15) Recovery

➔ Timestamp

➔ Table Name

➔ Data

16) Register_Table

➔ Name

➔ Phone Number

➔ Date Of Birth

➔ Medical Insurance

➔ Medical History

➔ Street

➔ City

➔ State

➔ Adhar Number

➔ E-mail ID

➔ Blood Group

B. Relationship Sets:-

1. **Donates -** The act of donation of an organ from a donor
   ➔ **Date -** Date Of Donation
2. **Procures -** The act of procuring/receiving an organ by the patient.
3. **Transaction**
   ➔ Date Of Transaction
   ➔ Status - Whether the surgery was successful or not.
4. **Organ Donated -** The organ donated by a donor, which is then stored in Organ available table.
5. **Attended By -** The transplantation performed by doctor - procuring an organ from a donor and transplanting it to the patient by surgery.
6. **Registers -** Donor is registered in which organization.
7. **Works In -** The organization where the doctor is working.
8. **Headed By -**The organization is headed by which person.

Tables and their Functional Dependencies :-

(Primary Key is highlighted in bold letters and Foreign Key is highlighted in Yellow**)**

1) User1(User_ID, Name, Date _of_birth, Medical_Insurance, Medical_History, Street, City, State)

FD={User_ID → Name, Date _of_birth, Medical Insurance, Medical History, Street, City, State**}**

2) User_phone_no(User_ID, phone_no**)**

FD={User_ID -> phone_no**}**

{User_ID} is foreign key constraint

3) Patient(Patient_ID, organ_req, reason_of_procurement, **Doctor_ID,**

User_ID)

FD={Patient_ID, organ_req -> reason_of_procurement,

Doctor_ID, User_ID**}**

{User_ID, Doctor_ID} are foreign key constraints

4) Donor(Donor_ID, organ_donated, reason_of_donation,

Organization_ID, **User_ID)**

FD={Donor_ID, organ_donated -> reason_of_donation,

Organization_ID, User_ID**}**

{User_ID, Organization_ID} are foreign key constraints

5) Organ Available(Organ_ID,Organ_name, **Donor_ID)**

FD={Organ_ID -> Organ_name,**Donor_ID}**

{Donor_ID} is foreign key constraint

6) Transaction(Patient_ID, Organ_ID, **Donor_ID**,

Date_of_transaction,

Status**)**

FD={Patient_ID, Organ_ID -> Donor_ID,Date_of_transaction,

Status}

{Patient_ID, Donor_ID} are foreign key constraints

7) Organization(Organization_ID, Organization_name, Location,

Government_approved**)**

FD={Organization_ID -> Organization_name, Location,

Government_approved**}**

8) Organization_phone_no(Organization_ID, phone_no)

FD={Organization_ID -> phone_no}

{Organization_ID} are foreign key constraints

9) Doctor(Doctor_ID, Doctor_name, Department_name, Organization_id**)**

FD={Doctor_ID -> Doctor_name, Organization_id**}**

{Organization_ID} is foreign key constraint

10) Doctor_phone_no(Doctor_ID, phone_no**)**

FD={Doctor_ID -> phone_no**}**

{Doctor_ID} is foreign key constraint

11) Organization_head(Organization_ID, Employee_ID, Name,

Date_of_joining, Term_length**)**

FD={Organization_ID, Employee_ID -> Name,
Date_of_joining,

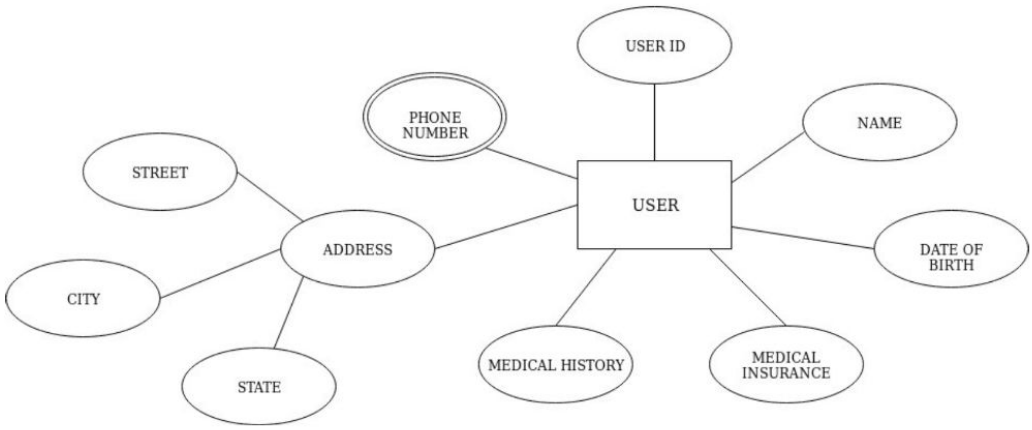
Term_length**}**

12)Add_death_user(death_id,name,reason_of_death,medical_history,medical_insurance,street,city,state,date_of_birth,donation_s tatus,perspective**)**

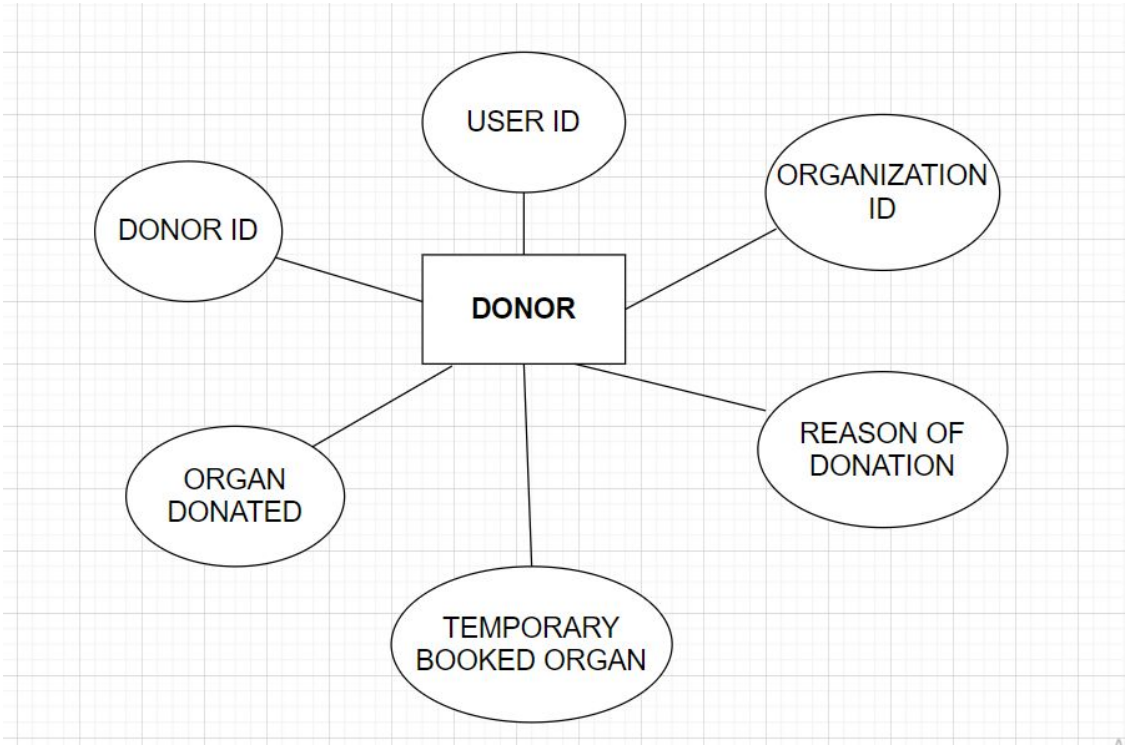
FD={death_id ->
name,reason_of_death,medical_history,medical_insurance,street,city,state,date_of_birth,donation_status,perspective**}**

EACH ENTITY SETS:-

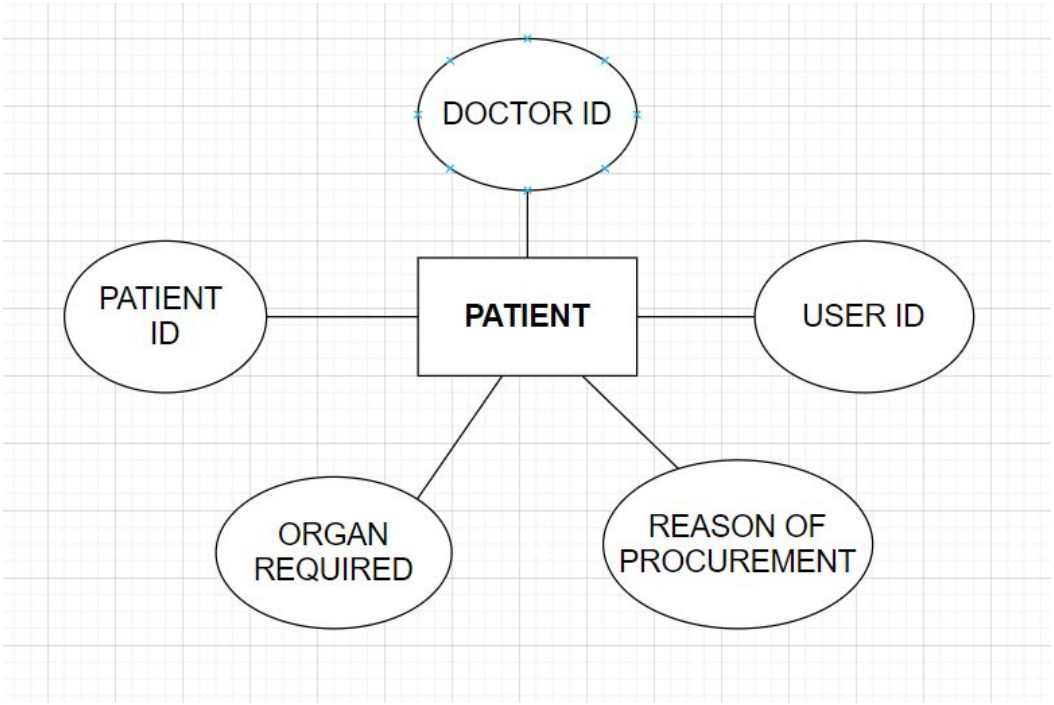
1) User -



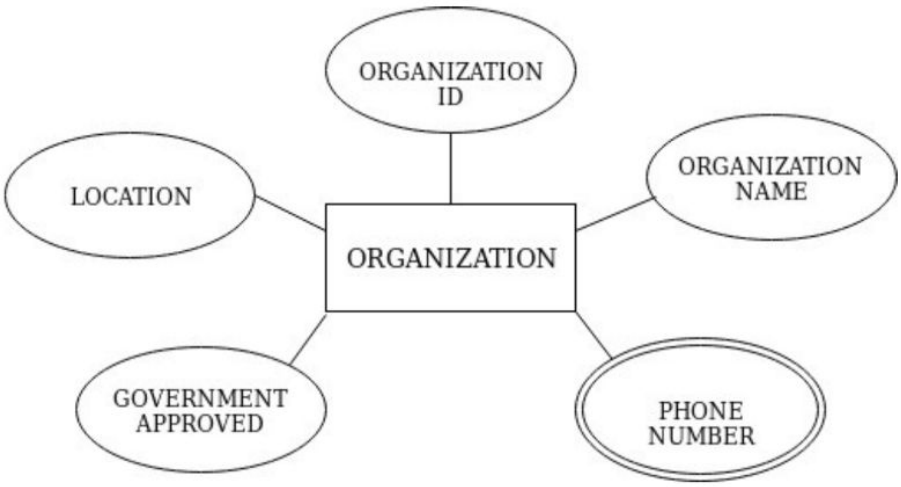
2) Donor



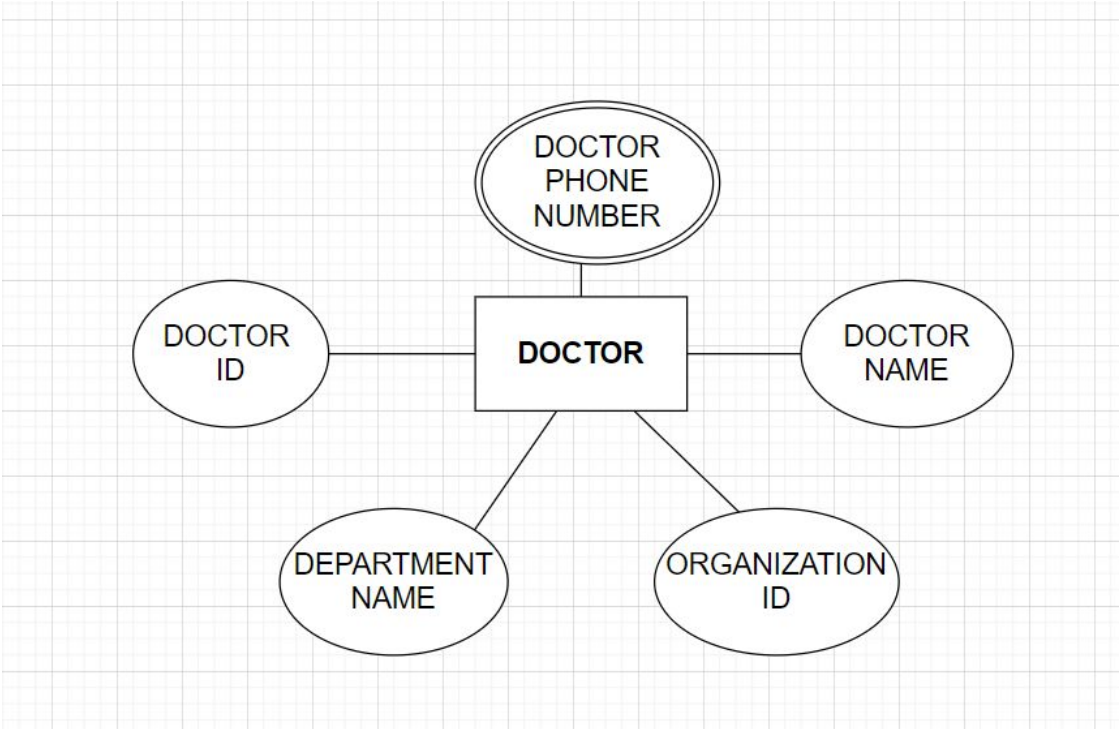
3) Patient



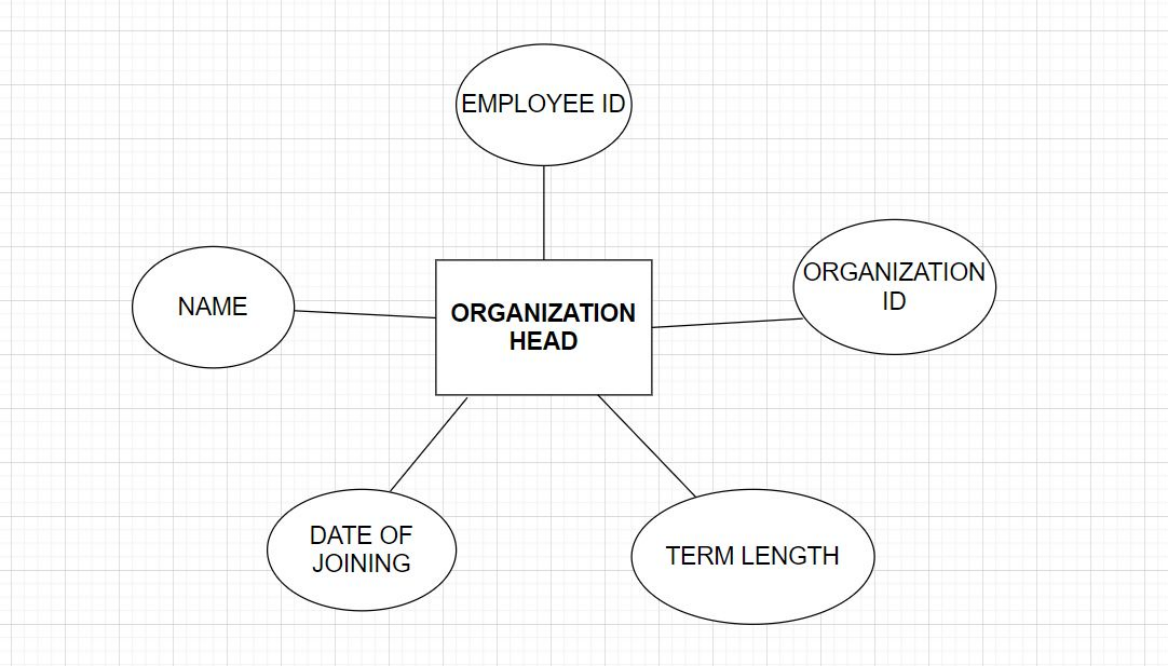
4) Organization



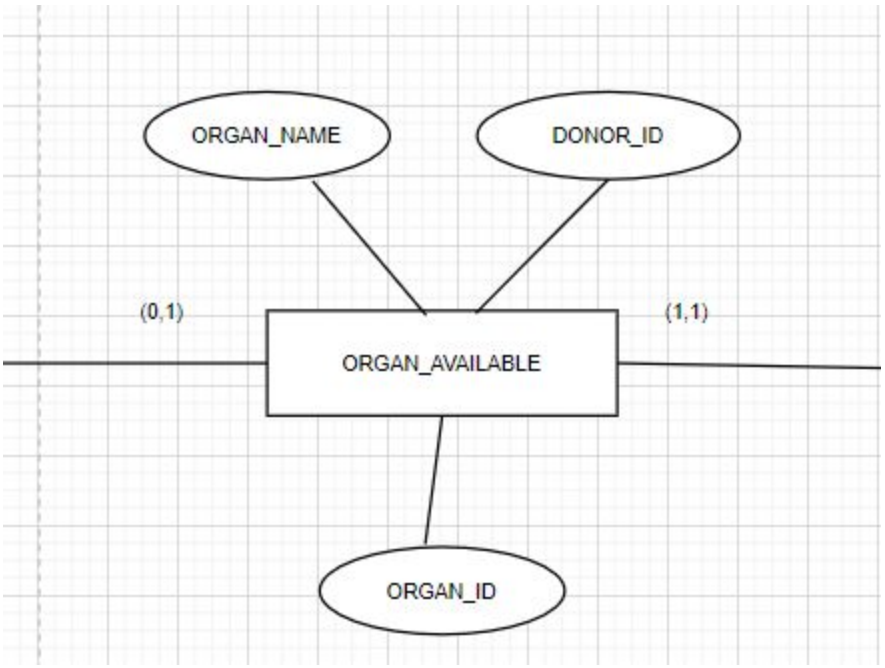
5) Doctor



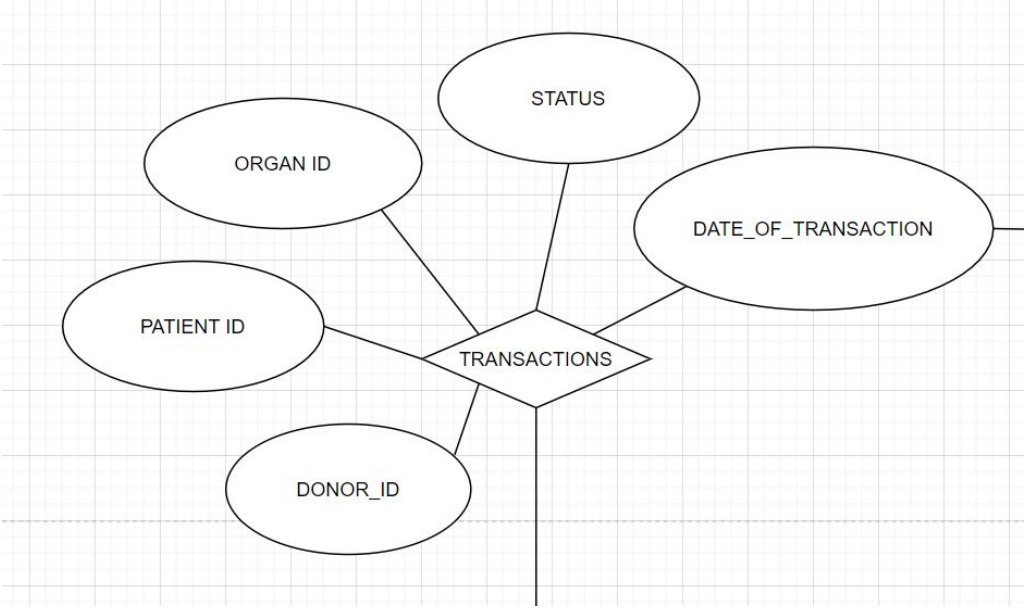
6) Organization Head



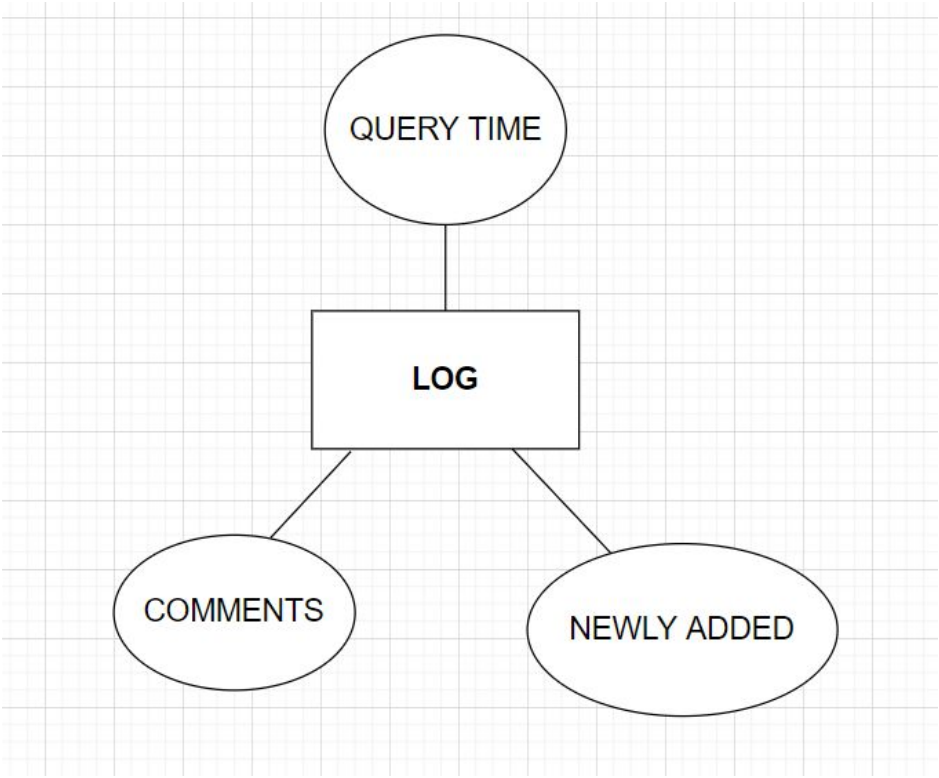
7) Organ Available



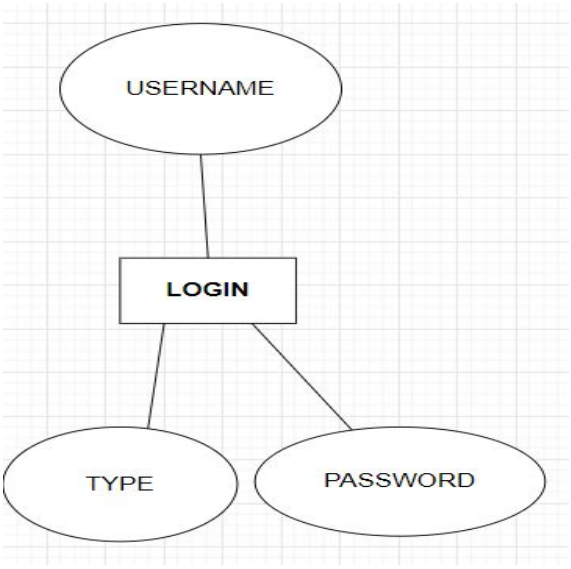
8) Transactions



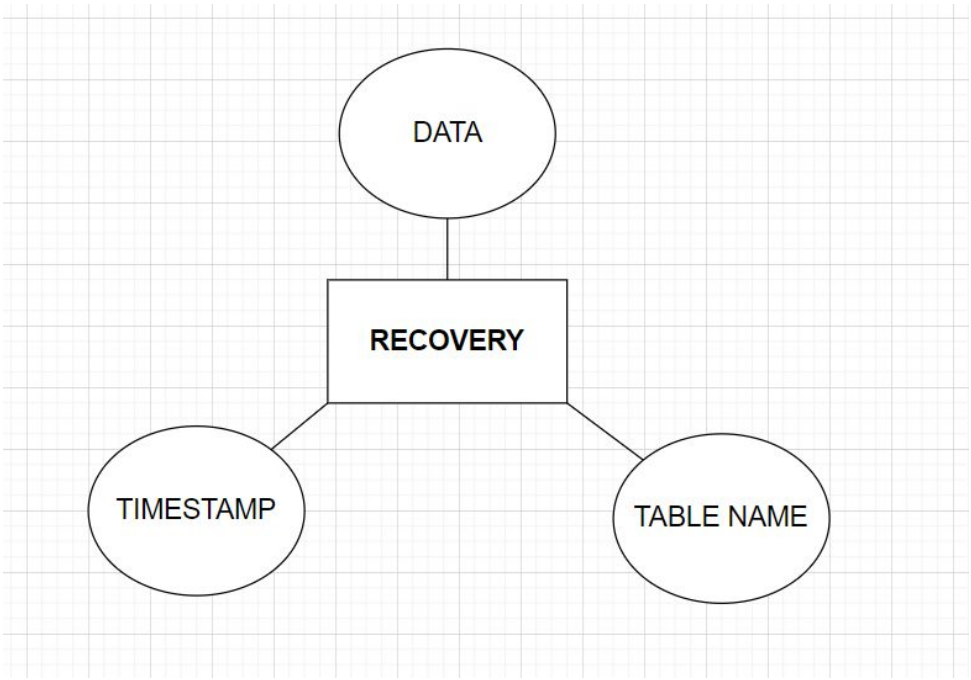
9) Log



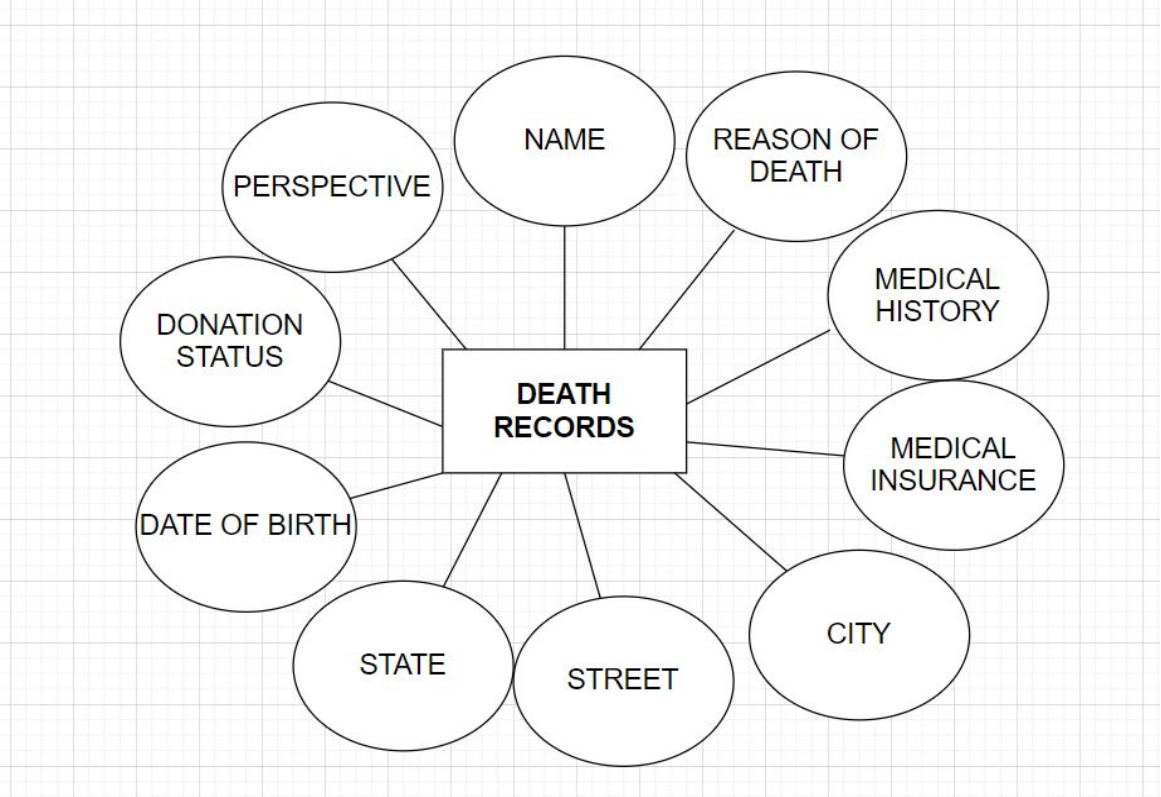
10) Login



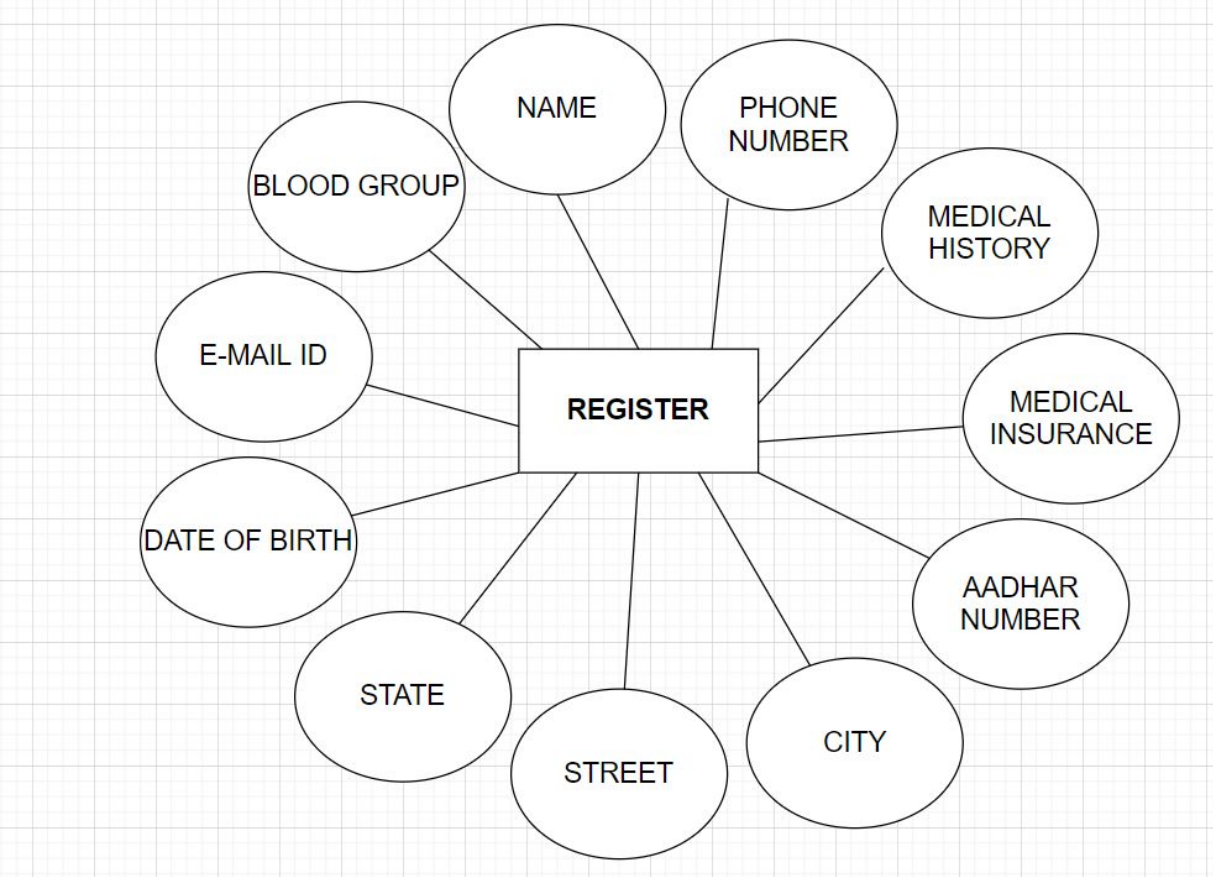
11) Recovery



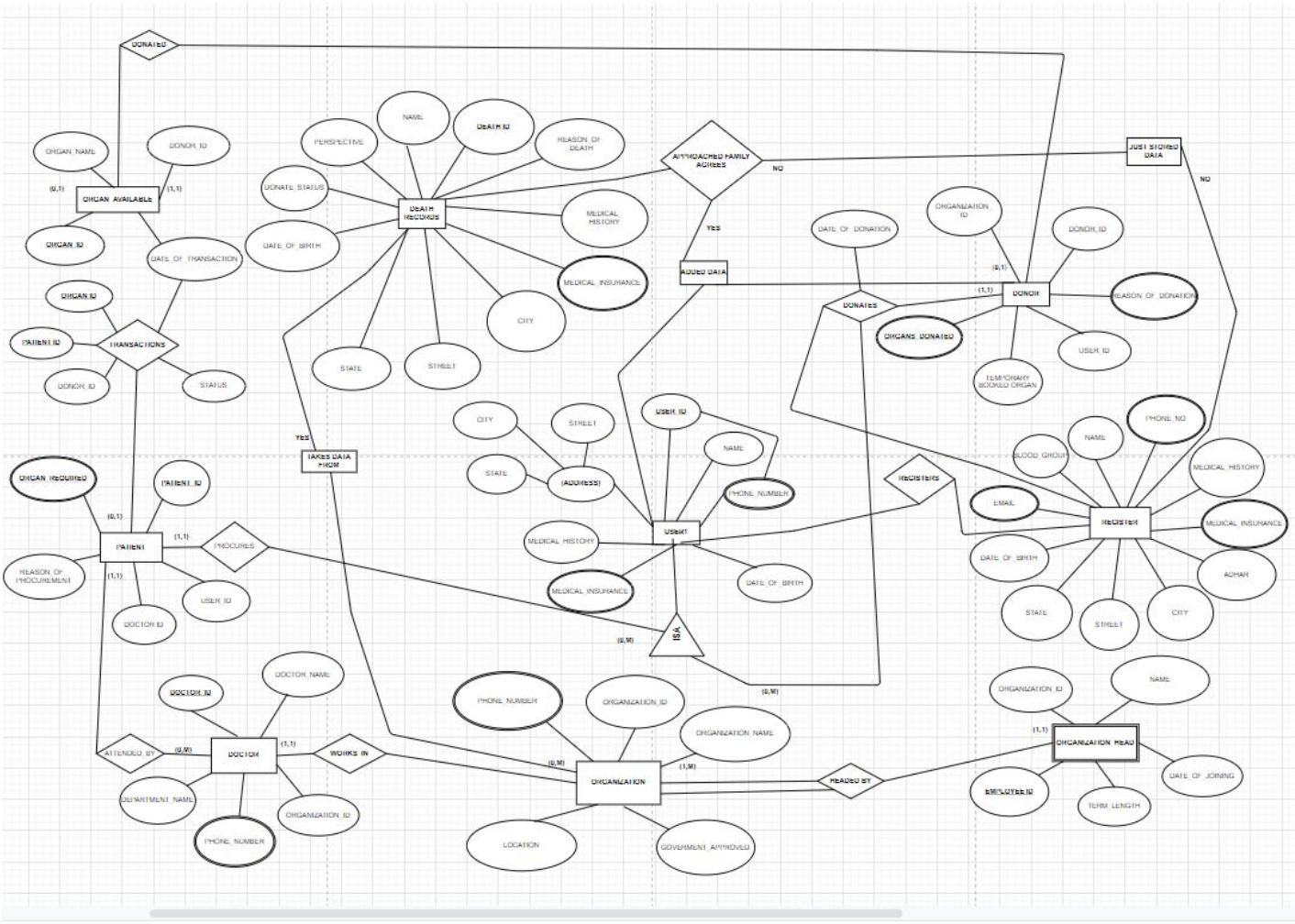
12) Death Records



13) Register


NAME
PHONE NUMBER
BLOOD GROUP
MEDICAL HISTORY
E-MAIL ID
REGISTER
MEDICAL INSURANCE
DATE OF BIRTH
AADHAR NUMBER
STATE
CITY
STREET

ER DIAGRAM



https://app.diagrams.net/#G1AEOYOp-FfB8ngms_t6ywms9kGmtFaSI-

(Here is the link to open through software/website if the above pic is not visible)

TABLE DESIGN

CREATING TABLES

1) Login

```
CREATE TABLE login(
      username VARCHAR(20) NOT NULL,
      password VARCHAR(20) NOT NULL,
      type     VARCHAR(20)
);
```

2) User

```
CREATE TABLE User1(
    User_ID int NOT NULL,
    Name varchar(20) NOT NULL,
    Date_of_Birth date NOT NULL,
    Medical_insurance int,
    Medical_history varchar(20),
    Street varchar(20),
    City varchar(20),
    State varchar(20),
    PRIMARY KEY(User_ID)
);
```

3) User Phone No

```
CREATE TABLE User_phone_no(
    User_ID int NOT NULL,
    phone_no varchar(15),
    FOREIGN KEY(User_ID) REFERENCES User1(User_ID) ON DELETE CASCADE
);
```

4) Organization

```
CREATE TABLE Organization(
  Organization_ID int NOT NULL,
  Organization_name varchar(20) NOT NULL,
  Location varchar(20),
  Government_approved int,
  PRIMARY KEY(Organization_ID)
);
```

5) Doctor

```
CREATE TABLE Doctor(
  Doctor_ID int NOT NULL,
  Doctor_Name varchar(20) NOT NULL,
  Department_Name varchar(20) NOT NULL,
  organization_ID int NOT NULL,
  FOREIGN KEY(organization_ID) REFERENCES Organization(organization_ID) ON DELETE CASCADE,
  PRIMARY KEY(Doctor_ID)
);
```

6) Patient

```
CREATE TABLE Patient(
    Patient_ID int primary key,
    organ_req varchar(20) NOT NULL,
    reason_of_procurement varchar(20),
    Doctor_ID int NOT NULL,
    User_ID int NOT NULL,
    FOREIGN KEY(User_ID) REFERENCES User1(User_ID) ON DELETE CASCADE,
    FOREIGN KEY(Doctor_ID) REFERENCES Doctor(Doctor_ID) ON DELETE CASCADE

);
```

7) Donor

```
CREATE TABLE Donor(
  Donor_ID int primary key,
  organ_donated varchar(20) ,
  reason_of_donation varchar(20),
  Organization_ID int NOT NULL,
  User_ID int NOT NULL,
  FOREIGN KEY(User_ID) REFERENCES User1(User_ID) ON DELETE CASCADE,
  CONSTRAINT donor_fk FOREIGN KEY(Organization_ID) REFERENCES Organization(Organization_ID) ON DELETE CASCADE
  );
```

8) Organ Available

```
CREATE TABLE Organ_available(
  Organ_ID int primary key ,
  Organ_name varchar(20) NOT NULL,
  Donor_ID int NOT NULL,
  FOREIGN KEY(Donor_ID) REFERENCES Donor(Donor_ID) ON DELETE CASCADE
);
```

9) Transaction

```
CREATE TABLE Transaction(
  Patient_ID int NOT NULL,
  Organ_ID int NOT NULL,
  Donor_ID int NOT NULL,
  Date_of_transaction date NOT NULL,
  Status int NOT NULL,
  FOREIGN KEY(Patient_ID) REFERENCES Patient(Patient_ID) ON DELETE CASCADE,
  FOREIGN KEY(Donor_ID) REFERENCES Donor(Donor_ID) ON DELETE CASCADE,
  PRIMARY KEY(Patient_ID,Organ_ID)
);
```

10) Organization Phone No.

```
CREATE TABLE Organization_phone_no(
  Organization_ID int NOT NULL,
  Phone_no varchar(15),
  FOREIGN KEY(Organization_ID) REFERENCES Organization(Organization_ID) ON DELETE CASCADE
);
```

11) Doctor Phone No.

```
CREATE TABLE Doctor_phone_no(
  Doctor_ID int NOT NULL,
  Phone_no varchar(15),
  FOREIGN KEY(Doctor_ID) REFERENCES Doctor(Doctor_ID) ON DELETE CASCADE
);
```

12) Organization Head

```
CREATE TABLE Organization_head(
  Organization_ID int NOT NULL,
  Employee_ID int NOT NULL,
  Name varchar(20) NOT NULL,
  Date_of_joining date NOT NULL,
  Term_length int NOT NULL,
  FOREIGN KEY(Organization_ID) REFERENCES Organization(Organization_ID) ON DELETE CASCADE,
  PRIMARY KEY(Organization_ID,Employee_ID)
);
```

13) Log

```
//created log table which will store that when
   the value was stored or deleted or manipulated
create table log(
  query_time char(30),
  comments varchar(255),
  newly_added varchar(255)
);
```

14) Death Records

```
create table death_records(death_id int NOT NULL,
                           NAME varchar(20) NOT NULL,
                           Reason_of_death varchar(20) NOT NULL,
                           Medical_history varchar(20) ,
                           medical_insurance int not null,
                           city varchar(20),
                           street varchar(20),
                           state varchar(20),
                           Date_of_birth date NOT NULL,
                           donate_status int NOT NULL,
                           perspective varchar(20) ,
                           primary key(death_id));
```

15) Recovery Table

```
create table recovery_table(  query_time char(30),
  tablename varchar(255),
  data varchar(255)
)
```

16) Register Table

```
create table register_table(
    Name varchar(20) NOT NULL,
    phone_no varchar(11) NOT NULL,
    Date_of_Birth date NOT NULL,
    Medical_insurance int NOT NULL,
    Medical_history varchar(20),
    Street varchar(20),
    City varchar(20),
    State varchar(20),
    Adhar varchar(13) NOT NULL,
    email varchar(25) NOT NULL,
    blood_group varchar(4) NOT NULL);
```

DATA DICTIONARY/ TABLE DESIGN

1) Login-

Username and Password are used to validate and authenticate the user information and type is used to check if the user is employee or admin.

Object Type **TABLE** Object **LOGIN**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
LOGIN	USERNAME	VARCHAR2	20	-	-	-	-	-	-
	PASSWORD	VARCHAR2	20	-	-	-	-	-	-
	TYPE	VARCHAR2	10	-	-	-	✓	-	-

1 - 3

COLUMN NAME	**DATA TYPE**	**CONSTRAINTS**	**FORMAT**	**DESCRIPTION**
USERNAME	VARCHAR2(20)	NOT NULL,UNIQUE	YASH	User's name
PASSWORD	VARCHAR2(20)	NOT NULL	12ABC	Password with standard encryption
TYPE	VARCHAR2(10)	NULL	ADMIN	Use to check is user is employee or admin

2) User-

This user table describes the basic user which is either donor or recipient and include their basic essential personal information which are mentioned as in the below table:-

Object Type **TABLE** Object **USER1**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
USER1	USER_ID	NUMBER	22	-	0	1	-	-	-
	NAME	VARCHAR2	20	-	-	-	-	-	-
	DATE_OF_BIRTH	DATE	7	-	-	-	-	-	-
	MEDICAL_INSURANCE	NUMBER	22	-	0	-	✓	-	-
	MEDICAL_HISTORY	VARCHAR2	20	-	-	-	✓	-	-
	STREET	VARCHAR2	20	-	-	-	✓	-	-
	CITY	VARCHAR2	20	-	-	-	✓	-	-
	STATE	VARCHAR2	20	-	-	-	✓	-	-

1 - 8

COLUMN NAME	**DATA TYPE**	**CONSTRAINTS**	**FORMAT**	**DESCRIPTION**
USER_ID	NUMBER(22)	NOT NULL,UNIQUE	92	User id of user
NAME	VARCHAR2(20)	NOT NULL	HARSHIL	Name of user
DATE_OF_ BIRTH	DATE(7)	NOT NULL	02/03/2000	Date Of Birth Of User
MEDICAL_ INSURANCE	NUMBER(22)	NULL	1 or 0	User detail of having medical insurance or not as 1, 0 respectively.
MEDICAL_ HISTORY	VARCHAR2(20)	NULL	Corona, Heart Disease,etc.	Details of Medical History
STREET	VARCHAR2(20)	NULL	Commerce Six Road	Address- Street Of User
CITY	VARCHAR2(20)	NULL	Ahmedabad	Address- City Of User
STATE	VARCHAR2(20)	NULL	Gujarat	Address-State Of User

3) User Phone No.

This table is used for storing the phone number with reference to the User ID.

Object Type **TABLE** Object **USER_PHONE_NO**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
USER_PHONE_NO	USER_ID	NUMBER	22	-	0	-	-	-	-
	PHONE_NO	VARCHAR2	15	-	-	-	✓	-	-
									1 - 2

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
USER_ID	NUMBER(22)	NOT NULL	12	User ID of user
PHONE_NO	VARCHAR2(15)	UNIQUE	9475138213	Phone No. of user

4) Organization

The organization table gives information about the basic data of the organization involved in this process of organ transplantation network.Here Government approved field has two values 1: government approved and 0: government not approved

Object Type **TABLE** Object **ORGANIZATION**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ORGANIZATION	ORGANIZATION_ID	NUMBER	22	-	0	1	-	-	-
	ORGANIZATION_NAME	VARCHAR2	20	-	-	-	-	-	-
	LOCATION	VARCHAR2	20	-	-	-	✓	-	-
	GOVERNMENT_APPROVED	NUMBER	22	-	0	-	✓	-	-
									1 - 4

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
ORGANIZATION_ ID	NUMBER(22)	NOT NULL,UNIQUE	11	Id of Organization
ORGANIZATION_ NAME	VARCHAR2(20)	NOT NULL	SEL	Name Of Organization
LOCATION	VARCHAR2(20)	NULL	Ahmedabad	Location Of Organization
GOVERNMENT_+ APPROVED	NUMBER(22)	NULL	1 or 0	Status of Government Approved Organization or Not

5) Doctor

The doctor table gives basic personal information regarding the doctors involved in this system and their relation with the organization.

Object Type **TABLE** Object **DOCTOR**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DOCTOR	DOCTOR_ID	NUMBER	22	-	0	1	-	-	-
	DOCTOR_NAME	VARCHAR2	20	-	-	-	-	-	-
	DEPARTMENT_NAME	VARCHAR2	20	-	-	-	-	-	-
	ORGANIZATION_ID	NUMBER	22	-	0	-	-	-	-

1 - 4

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
DOCTOR_ID	NUMBER(22)	NOT NULL,UNIQUE	12	Id Of Doctor
DOCTOR_NAME	VARCHAR2(20)	NOT NULL	SATYAM	Name of the Doctor
DEPARTMENT_ NAME	VARCHAR2(20)	NOT NULL,UNIQUE	CARDIO	Name of the Department
ORGANIZATION	NUMBER(22)	NOT	12	Id Of

_ID		NULL,UNIQUE		Organization

6) Patient

The Patient table is for storing the basic essential information of the Patient for performing any medical operations in the future and also contains important information related to organ donation and their authenticity of identification.

Object Type **TABLE** Object **PATIENT**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PATIENT	PATIENT_ID	NUMBER	22	-	0	1	-	-	-
	ORGAN_REQ	VARCHAR2	20	-	-	-	-	-	-
	REASON_OF_PROCUREMENT	VARCHAR2	20	-	-	-	✓	-	-
	DOCTOR_ID	NUMBER	22	-	0	-	-	-	-
	USER_ID	NUMBER	22	-	0	-	-	-	-

1 - 5

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
PATIENT_ID	NUMBER(22)	NOT NULL, UNIQUE	24	Id Of Patient
ORGAN_REQ	VARCHAR2(20)	NOT NULL	Heart	Requirement Of Organ
REASON_OF_ PROCUREMENT	VARCHAR2(20)	NULL	Failure Of Heart	Reason for procurement of Organ
DOCTOR_ID	NUMBER(22)	NOT NULL, UNIQUE	28	Id of the Doctor
USER_ID	NUMBER(22)	NOT NULL, UNIQUE	12	Id of the User

7) Donor

The Donor table contains information about the donor and about his/her organs and its specification along with authentication for unique identification.

Object Type **TABLE** Object **DONOR**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DONOR	DONOR_ID	NUMBER	22	-	0	1	-	-	-
	ORGAN_DONATED	VARCHAR2	20	-	-	-	✓	-	-
	REASON_OF_DONATION	VARCHAR2	20	-	-	-	✓	-	-
	ORGANIZATION_ID	NUMBER	22	-	0	-	-	-	-
	USER_ID	NUMBER	22	-	0	-	-	-	-

1 - 5

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
DONOR_ID	NUMBER(22)	NOT NULL,UNIQUE	23	Id of the Donor
ORGAN_ DONATED	VARCHAR2(20)	NULL	Lungs	Detail of Organ Donated
REASON_OF_ DONATION	VARCHAR2(20)	NULL	To save life	Detail of Reason Of Donation
ORGANIZATIO N_ID	NUMBER(22)	NOT NULL,UNIQUE	23	Id of the Organization
USER_ID	NUMBER(22)	NOT NULL,UNIQUE	56	Id of the User

8) Organ Available

The Organ available table is used for storing information regarding organ identification, number and the donor whose organ it is.

Object Type **TABLE** Object **ORGAN_AVAILABLE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ORGAN_AVAILABLE	ORGAN_ID	NUMBER	22	-	0	1	-	-	-
	ORGAN_NAME	VARCHAR2	20	-	-	-	-	-	-
	DONOR_ID	NUMBER	22	-	0	-	-	-	-

1 - 3

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
ORGAN_ID	NUMBER(22)	NOT NULL,UNIQUE	23	Id of the Organ
ORGAN_NAME	VARCHAR2(20)	NOT NULL	Kidneys	Name of the Organ
DONOR_ID	NUMBER(22)	NOT NULL,UNIQUE	87	Id of the Donor

9) Transaction

The transaction table is for maintaining the record of the operations of the patient and history of recorded organ and operations along with date and status.

Object Type **TABLE** Object **TRANSACTION**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
TRANSACTION	PATIENT_ID	NUMBER	22	-	0	1	-	-	-
	ORGAN_ID	NUMBER	22	-	0	2	-	-	-
	DONOR_ID	NUMBER	22	-	0	-	-	-	-
	DATE_OF_TRANSACTION	DATE	7	-	-	-	-	-	-
	STATUS	NUMBER	22	-	0	-	-	-	-

1 - 5

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
PATIENT_ID	NUMBER(22)	NOT NULL,UNIQUE	23	Id of the Patient
ORGAN_ID	NUMBER(22)	NOT NULL,UNIQUE	34	Id of the Organ
DONOR_ID	NUMBER(22)	NOT NULL,UNIQUE	45	Id of the Donor
DATE_OF_ TRANSACTION	DATE(7)	NOT NULL,	09/08/2020	Details of the date of transaction.
STATUS	NUMBER(22)	NOT NULL		Status of the Transaction

10) Organization Phone No.

This table is for storing the contact information of the organization.

Object Type **TABLE** Object **ORGANIZATION_PHONE_NO**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ORGANIZATION_PHONE_NO	ORGANIZATION_ID	NUMBER	22	-	0	-	-	-	-
	PHONE_NO	VARCHAR2	15	-	-	-	✓	-	-

1 - 2

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
ORGANIZATION_ID	NUMBER(22)	NOT NULL,UNIQUE	34	Id of the Organization
PHONE_NO	VARCHAR2(15)	UNIQUE	8746193214	Phone Number Details

11) Doctor Phone No.

This table is for storing the contact information of the Doctor working at a particular organization.

Object Type **TABLE** Object **DOCTOR_PHONE_NO**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DOCTOR_PHONE_NO	DOCTOR_ID	NUMBER	22	-	0	-	-	-	-
	PHONE_NO	VARCHAR2	15	-	-	-	✓	-	-
									1 - 2

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
DOCTOR_ID	NUMBER(22)	NOT NULL,UNIQUE	76	Id of the Doctor
PHONE_NO	VARCHAR2(15)	UNIQUE	8745136932	Phone Number details of the Doctor

12) Organization Head

This table is for storing the information of organization head and their history related with their job and profession in that particular organization.

Object Type **TABLE** Object **ORGANIZATION_HEAD**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ORGANIZATION_HEAD	ORGANIZATION_ID	NUMBER	22	-	0	1	-	-	-
	EMPLOYEE_ID	NUMBER	22	-	0	2	-	-	-
	NAME	VARCHAR2	20	-	-	-	-	-	-
	DATE_OF_JOINING	DATE	7	-	-	-	-	-	-
	TERM_LENGTH	NUMBER	22	-	0	-	-	-	-
									1 - 5

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
ORGANIZATION_ID	NUMBER(22)	NOT NULL,UNIQUE	34	Id of the Organization
EMPLOYEE_ID	NUMBER(22)	NOT NULL,UNIQUE	45	Id of the Employee
NAME	VARCHAR2(20)	NOT NULL	TEJAS	Details of Name
DATE_OF_JOINING	DATE(7)	NOT NULL	19/4/2010	Joining Date Details
TERM_LENGTH	NUMBER(22)	NOT NULL	30	Details of term length

13) Death Records

This table is used to take the data from the police stations and hospitals about the records of deaths to approach their family for donating the organs . If they agree the data is added to the user and donor group.

Object Type **TABLE** Object **DEATH_RECORDS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEATH_RECORDS	DEATH_ID	NUMBER	22	-	0	1	-	-	-
	NAME	VARCHAR2	20	-	-	-	-	-	-
	REASON_OF_DEATH	VARCHAR2	20	-	-	-	-	-	-
	MEDICAL_HISTORY	VARCHAR2	20	-	-	-	✓	-	-
	MEDICAL_INSURANCE	NUMBER	22	-	0	-	-	-	-
	CITY	VARCHAR2	20	-	-	-	✓	-	-
	STREET	VARCHAR2	20	-	-	-	✓	-	-
	STATE	VARCHAR2	20	-	-	-	✓	-	-
	DATE_OF_BIRTH	DATE	7	-	-	-	-	-	-
	DONATE_STATUS	NUMBER	22	-	0	-	-	-	-
	PERSPECTIVE	VARCHAR2	20	-	-	-	✓	-	-

1 - 11

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
DEATH_ID	NUMBER(22)	NOT NULL,UNIQUE	34	Id of Death Details
NAME	VARCHAR2(20)	NOT NULL	STANLEE	Name of the person dead
REASON_OF_ DEATH	VARCHAR2(20)	NOT NULL	Heart Failure	Details of reason of death
MEDCIAL_ HISTORY	VARCHAR2(20)	NULL	N/A	Details Of Medical History
MEDICAL_ INSURANCE	NUMBER(22)	NOT NULL	1	Details of Medical Insurance Status as Yes or No for 1 or 0 respectively.
CITY	VARCHAR2(20)	NULL	NewYork	Delhi
STREET	VARCHAR2(20)	NULL	Navrangpura	Chandni Chowk
STATE	VARCHAR2(20)	NULL	Ontario	Delhi
DATE_OF_ BIRTH	DATE(7)	NOT NULL	21/02/1987	28/04/1957
DONATE_ STATUS	NUMBER(22)	NOT NULL	1	Details of Donation Status
PERSPECTIVE	VARCHAR2(20)	NULL	YES	Perspective for donating the organ or not

14) Log

This table is used to store the backend data of what and when happened when the data is manipulated .

Object Type **TABLE** Object **LOG**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
LOG	QUERY_TIME	CHAR	30	-	-	-	✓	-	-
	COMMENTS	VARCHAR2	255	-	-	-	✓	-	-
	NEWLY_ADDED	VARCHAR2	255	-	-	-	✓	-	-

1 - 3

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
QUERY_TIME	CHAR(30)	NOT NULL, UNIQUE	26-MAR-2020 04:41:29	Details of the Query Time
COMMENTS	VARCHAR2(255)	NOT NULL	Inserted new Donor :	Details of the Comments
NEWLY_ADDED	VARCHAR2(255)	NOT NULL	38	Details of newly added

15) Recovery Table

This table is for recovering the data from the database in case if it is deleted or lost by mistake. This is extremely useful feature as it can recover and maintain the data of the organization for long run and can maintain the integrity and history of the organization

Object Type **TABLE** Object **RECOVERY_TABLE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
RECOVERY_TABLE	QUERY_TIME	CHAR	30	-	-	-	✓	-	-
	TABLENAME	VARCHAR2	255	-	-	-	✓	-	-
	DATA	VARCHAR2	255	-	-	-	✓	-	-

1 - 3

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
QUERY_TIME	CHAR(30)	NULL, UNIQUE	19-APR-2020 03:57:01	Details of the Query Time
TABLENAME	VARCHAR2(255)	NULL	Donor	Details of the tablename
DATA	VARCHAR2(255)	NULL	201,Lungs,new,2,100	Details of the data

16) Register Table

This table is used to store the data of the person who wishes to donate the organ after death.

Object Type **TABLE** Object **REGISTER_TABLE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
REGISTER_TABLE	NAME	VARCHAR2	20	-	-	-	-	-	-
	PHONE_NO	VARCHAR2	11	-	-	-	-	-	-
	DATE_OF_BIRTH	DATE	7	-	-	-	-	-	-
	MEDICAL_INSURANCE	NUMBER	22	-	0	-	-	-	-
	MEDICAL_HISTORY	VARCHAR2	20	-	-	-	✓	-	-
	STREET	VARCHAR2	20	-	-	-	✓	-	-
	CITY	VARCHAR2	20	-	-	-	✓	-	-
	STATE	VARCHAR2	20	-	-	-	✓	-	-
	ADHAR	VARCHAR2	13	-	-	-	-	-	-
	EMAIL	VARCHAR2	25	-	-	-	-	-	-
	BLOOD_GROUP	VARCHAR2	4	-	-	-	-	-	-

1 - 11

COLUMN NAME	DATA TYPE	CONSTRAINTS	FORMAT	DESCRIPTION
NAME	VARCHAR2(20)	NOT NULL	VIJAY	Details of the name of the user.
PHONE_NO	VARCHAR2(11)	NOT NULL,UNIQUE	8745696471	Details of the phone number of the user.
DATE_OF_ BIRTH	DATE(7)	NOT NULL	03/02/1974	Details of the date of birth of the user
MEDICAL_ INSURANCE	NUMBER(22)	NOT NULL	1	Details of the medical insurance status as 1 and 0 for Yes and No respectively.
MEDICAL_ ISTORY	VARCHAR2(20)	NULL	Diabetes	Details of the medical history of the user
STREET	VARCHAR2(20)	NULL	Diamond Road	Address-Street of Residence of the User
CITY	VARCHAR2(20)	NULL	Surat	Address-City of Residence of the User
STATE	VARCHAR2(20)	NULL	Gujarat	Address-State of Residence of the User
ADHAR	VARCHAR2(13)	NOT NULL,UNIQUE	164784578713	Details of the Adhar Card Number of the User

EMAIL	VARCHAR2(25)	NOT NULL,UNIQUE	abc123@gmail.com	Details of Email ID of the User
BLOOD_ GROUP	VARCHAR2(4)	NOT NULL	O+	Details of Blood Group of the User

STORED PROCEDURES, FUNCTIONS, AND TRIGGERS

PROCEDURES

1) PROCEDURE TO GET DOCTOR TRANSACTION COUNT

```
create or replace procedure get_doctor_transaction_count
as cursor c_doctor is  select distinct(patient.doctor_id),
doctor.doctor_name,doctor.department_name,
doctor.organization_id from patient,
doctor where patient.doctor_id=doctor.doctor_id;
cursor c_organ is select distinct(organ_donated) from donor;
cursor c_details is select doctor_id,organ_req,patient_id from patient
where exists(select patient_id from transaction
where patient.patient_id=transaction.patient_id);
r_doctor c_doctor%ROWTYPE;
r_organ c_organ%ROWTYPE;
r_details c_details%ROWTYPE;
count_organ int:=0;
total_count int:=0;
```

```
begin
OPEN c_doctor;
LOOP
Fetch c_doctor into r_doctor;
EXIT WHEN c_doctor%NOTFOUND;
dbms_output.put_line(r_doctor.doctor_id);
dbms_output.put_line(r_doctor.doctor_name);
dbms_output.put_line(r_doctor.department_name);
dbms_output.put_line(r_doctor.organization_id);
count_organ:=0;

        OPEN c_organ;
        LOOP
        Fetch c_organ into r_organ;
        EXIT WHEN c_organ%NOTFOUND;

EXIT WHEN c_doctor%NOTFOUND;
dbms_output.put_line(r_doctor.doctor_id);
dbms_output.put_line(r_doctor.doctor_name);
dbms_output.put_line(r_doctor.department_name);
dbms_output.put_line(r_doctor.organization_id);
count_organ:=0;

        OPEN c_organ;
        LOOP
        Fetch c_organ into r_organ;
        EXIT WHEN c_organ%NOTFOUND;
        count_organ:=0;

                OPEN c_details;
                LOOP
```

```
                Fetch c_details into r_details;
                EXIT WHEN c_details%NOTFOUND;
                if(r_organ.organ_donated=r_details.organ_req
and r_details.doctor_id=r_doctor.doctor_id) then
                count_organ:=count_organ+1;
total_count:=total_count+1;
                end if;
                end loop;
                close c_details;
dbms_output.put_line(count_organ);
        end loop;
        close c_organ;

end loop;
close c_doctor;
dbms_output.put_line(total_count);
end;

declare
begin
 get_doctor_transaction_count ;
end;

select * from patient where user_id=89
dbms_output.put_line(r_organ.organ_donated);
```

2) PROCEDURE TO GET DOCTOR TRANSACTION BY INSERTING DOCTOR NAME

```
create or replace procedure get_doctor_transaction_count_p(doctor_name varchar)
as cursor c_doctor is  select distinct(patient.doctor_id),doctor.doctor_name,
doctor.department_name,doctor.organization_id from patient,
doctor where patient.doctor_id=doctor.doctor_id;
cursor c_organ is select distinct(organ_donated) from donor;
cursor c_details is select doctor_id,organ_req,patient_id from patient
where exists(select patient_id from transaction
where patient.patient_id=transaction.patient_id);
r_doctor c_doctor%ROWTYPE;
r_organ c_organ%ROWTYPE;
r_details c_details%ROWTYPE;
count_organ int:=0;
total_count int:=0;

begin
OPEN c_doctor;
LOOP
Fetch c_doctor into r_doctor;
EXIT WHEN c_doctor%NOTFOUND;
if(r_doctor.doctor_name=doctor_name) then
dbms_output.put_line(r_doctor.doctor_id);
dbms_output.put_line(r_doctor.doctor_name);
dbms_output.put_line(r_doctor.department_name);
dbms_output.put_line(r_doctor.organization_id);
count_organ:=0;

        OPEN c_organ;
        LOOP
```

```
        Fetch c_organ into r_organ;
        EXIT WHEN c_organ%NOTFOUND;
        count_organ:=0;

                OPEN c_details;
                LOOP
                Fetch c_details into r_details;
                EXIT WHEN c_details%NOTFOUND;
                if(r_organ.organ_donated=r_details.organ_req
and r_details.doctor_id=r_doctor.doctor_id) then
                count_organ:=count_organ+1;
total_count:=total_count+1;
                end if;
                end loop;
                 close c_details;
dbms_output.put_line(count_organ);
       end loop;
       close c_organ;
end if;
end loop;
close c_doctor;
dbms_output.put_line(total_count);
end;

declare
begin
 get_doctor_transaction_count_p('Doctor-1');
end;
```

3) PROCEDURE TO GET ORGAN AVAILABLE COUNT CITY BASED

```
create or replace procedure get_organ_available_count (city varchar)
as cursor c_organ is  select distinct(organ_donated) from donor;
cursor c_details is select donor.donor_id,donor.organ_donated,
donor.user_id,user1.city from donor,user1 where not exists
(select donor_id from transaction where donor.donor_id=transaction.donor_id)
and donor.user_id=user1.user_id;
r_organ c_organ%ROWTYPE;
r_details c_details%ROWTYPE;
count_temp int:=0;

begin
OPEN c_organ;
LOOP
Fetch c_organ into r_organ;

EXIT WHEN c_organ%NOTFOUND;
count_temp:=0;

            OPEN c_details;
            LOOP
            Fetch c_details into r_details;
            EXIT WHEN c_details%NOTFOUND;
            if(r_details.organ_donated=r_organ.organ_donated
and r_details.city=city) then
            count_temp:=count_temp+1;
            end if;
            end loop;
            close c_details;
 dbms_output.put_line(count_temp);
```

```
end loop;
close c_organ;
end;

declare

begin
 get_organ_available_count('Ahmedabad');
end;
```

4) PROCEDURE TO GET FAILURE ORGANWISE

```
create or replace procedure GET_FAILURE_ORGANWISE as
cursor c_details is select transaction.PATIENT_ID,
transaction.ORGAN_ID,transaction.DONOR_ID,transaction.DATE_OF_TRANSACTION,
transaction.STATUS,donor.organ_donated from transaction,donor
where donor.donor_id=transaction.donor_id and status=0;
r_details c_details%ROWTYPE;
cursor c_organ is select distinct(organ_donated) from donor;
r_organ c_organ%ROWTYPE;

count_total int:=0;
count_temp int:=0;
begin
OPEN c_organ;
LOOP
```

```
FETCH c_organ INTO r_organ;
EXIT WHEN c_organ%NOTFOUND;
count_temp:=0;
            OPEN c_details;
            LOOP
            FETCH c_details INTO r_details;
            EXIT WHEN c_details%NOTFOUND;
            if(r_details.organ_donated=r_organ.organ_donated) then
            count_temp:=count_temp+1;
            count_total:=count_total+1;
            end if;
            end loop;
            close c_details;
dbms_output.put_line(count_temp);
end loop;

close c_organ;
dbms_output.put_line(count_total);
end;


declare
begin
GET_FAILURE_ORGANWISE;
end;
```

5) PROCEDURE TO GET PATIENT CITY BASED

```
create or replace procedure get_patient_city_based(user_city varchar,organ_required varchar)
as cursor c_user is  select user1.user_id,user1.name,user1.date_of_birth,
user1.medical_insurance,user1.medical_history,user1.street,
user1.city,user1.state,patient.organ_req from user1,patient
where user1.user_id=patient.user_id and
patient.organ_req=organ_required and user1.city=user_city;
r_user c_user%ROWTYPE;
count_done int:=0;

begin
OPEN c_user;
LOOP
Fetch c_user into r_user;
EXIT WHEN c_user%NOTFOUND;
```

```
dbms_output.put_line(r_user.user_id||','||r_user.name||',
'||r_user.date_of_birth||','||r_user.medical_insurance||',
'||r_user.medical_history||','||r_user.street||','||r_user.city||',
'||r_user.state||','||r_user.organ_req);
count_done:=count_done+1;
end loop;
close c_user;
 dbms_output.put_line('Total Pending patients: '||count_done);
end;
```

```
declare
user_city varchar(20):=:user_city;
organ_req varchar(20):=:organ_required;
```

```
begin
get_patient_city_based(user_city,organ_req);
end;
```

6) PROCEDURE TO GET SUCCESS ORGANWISE

```
create or replace procedure GET_SUCCESS_ORGANWISE as
cursor c_details is select transaction.PATIENT_ID,
transaction.ORGAN_ID,transaction.DONOR_ID,
transaction.DATE_OF_TRANSACTION,transaction.STATUS,
donor.organ_donated from transaction,donor
where donor.donor_id=transaction.donor_id and status=1;
r_details c_details%ROWTYPE;
cursor c_organ is select distinct(organ_donated) from donor;
r_organ c_organ%ROWTYPE;

count_total int:=0;
count_temp int:=0;
begin
OPEN c_organ;
LOOP

FETCH c_organ INTO r_organ;
EXIT WHEN c_organ%NOTFOUND;
count_temp:=0;

            OPEN c_details;
            LOOP
            FETCH c_details INTO r_details;
            EXIT WHEN c_details%NOTFOUND;
            if(r_details.organ_donated=r_organ.organ_donated) then
            count_temp:=count_temp+1;
            count_total:=count_total+1;
            end if;
            end loop;
            close c_details;
```

```
dbms_output.put_line(count_temp);
end loop;
close c_organ;
dbms_output.put_line(count_total);
end;


declare
begin
GET_SUCCESS_ORGANWISE;
end;
```

7) PROCEDURE TO GET USER CITY BASED

```
create or replace procedure get_user_city_based(user_city varchar)
as cursor c_user is  select user_id,name,date_of_birth,
medical_insurance,medical_history,street,city,
state from user1 where city=user_city;
r_user c_user%ROWTYPE;
count_done int:=0;

begin
OPEN c_user;
LOOP
Fetch c_user into r_user;
EXIT WHEN c_user%NOTFOUND;
```

```
dbms_output.put_line(r_user.user_id||','||r_user.name||',
'||r_user.date_of_birth||','||r_user.medical_insurance||',
'||r_user.medical_history||','||r_user.street||',
'||r_user.city||','||r_user.state);
count_done:=count_done+1;
end loop;
close c_user;
 dbms_output.put_line('Total Pending patients: '||count_done);
end;

declare
user_city varchar(20):=:user_city;
begin
get_user_city_based(user_city);
end;
```

8) PROCEDURE TO GET WAITING LIST

```
create or replace procedure get_waiting_patients
as cursor c_patient is  select patient.patient_id,
patient.organ_req,patient.reason_of_procurement,
patient.doctor_id,patient.user_id,user1.city from patient,
user1 where patient_id NOT IN (select patient_id from transaction)
and user1.user_id=patient.user_id;
r_patient c_patient%ROWTYPE;
count_pending int:=0;
count_done int:=0;

begin
OPEN c_patient;
LOOP
Fetch c_patient into r_patient;
EXIT WHEN c_patient%NOTFOUND;
```

```
dbms_output.put_line(r_patient.patient_id||',
'||r_patient.ORGAN_REQ||','||r_patient.REASON_OF_PROCUREMENT||',
'||r_patient.DOCTOR_ID||','||r_patient.USER_ID||','||r_patient.city);
count_pending:=count_pending+1;
end loop;
close c_patient;
 dbms_output.put_line('Total Pending patients: '||count_pending);
end;


declare
begin
get_waiting_patients;
end;
```

9) PROCEDURE TO GET WAITING LIST CITY BASED

```
create or replace procedure get_city_waiting_patients (city varchar)
as cursor c_patient is  select patient.patient_id,
patient.organ_req,patient.reason_of_procurement,
patient.doctor_id,patient.user_id,user1.city from patient,user1
where patient_id NOT IN (select patient_id from transaction)
and user1.user_id=patient.user_id;
r_patient c_patient%ROWTYPE;
count_pending int:=0;
count_done int:=0;

begin
OPEN c_patient;
LOOP
Fetch c_patient into r_patient;
EXIT WHEN c patient%NOTFOUND;
```

```
if(r_patient.city=city) then
dbms_output.put_line(r_patient.patient_id||',
'||r_patient.ORGAN_REQ||','||r_patient.REASON_OF_PROCUREMENT||',
'||r_patient.DOCTOR_ID||','||r_patient.USER_ID||','||r_patient.city);
count_pending:=count_pending+1;
end if;
end loop;
close c_patient;
 dbms_output.put_line('Total Pending patients: '||count_pending);
end;

declare
begin
get_city_waiting_patients('Ahmedabad');
end;
```

10) PROCEDURE TO ADD DEATH OF USER

```
create or replace procedure ADD_DEATH_TO_USER(death_id int,
organ_donated varchar,reason_of_donation varchar,organization_id int)
as cursor c_death is select * from death_records;
r_death c_death%ROWTYPE;
count_done int:=0;
user_id int:=0;
begin
      open c_death;
      loop
            fetch c_death into r_death;
            EXIT WHEN c_death%NOTFOUND;
         if(r_death.death_id=death_id) then
         insert into user1 values((select count(*)+1 from user1),
r_death.name,r_death.Date_of_birth,r_death.medical_insurance,
r_death.medical_history,r_death.street,r_death.city,r_death.state);
```

```
        insert into donor values((select count(*)+1 from donor),
organ_donated,reason_of_donation,organization_id,(select count(*) from user1));
        count_done:=1;
        end if;
end loop;
close c_death;
if(count_done=1) then
dbms_output.put_line('STORED VALUE');
end if;
if(count_done=0) then
dbms_output.put_line('DATA NOT FOUND PLEASE CHECK YOUR INPUT AGAIN');
end if;
end;


declare
death_id int:=:death_id;
organ_donated varchar(20):=:organ_donated;
reason_of_donation varchar(20):=:reason_donation;
organization_id int:=:organization_id;
begin
ADD_DEATH_TO_USER(death_id,organ_donated,reason_of_donation,organization_id);
end;
```

11) PROCEDURE TO GET SUCCESS RATE ORGANIZATION-WISE

```
//11- Procedure to get success rate organization wise

create or replace procedure GET_SUCCESS_ORGANIZATIONWISE as
cursor c_details is select organization.organization_name
as name,count(*) as c from transaction,patient,doctor,organization
where transaction.patient_id=patient.patient_id and
patient.doctor_id=doctor.doctor_id and
doctor.organization_id=organization.organization_id and
transaction.status=1 group by organization.organization_name
order by c desc;
r_details c_details%ROWTYPE;

begin
OPEN c_details;
LOOP
```

```
FETCH c_details INTO r_details;
EXIT WHEN c_details%NOTFOUND;
dbms_output.put_line(r_details.name||','||r_details.c);
end loop;
close c_details;
end;

declare
begin
GET_SUCCESS_ORGANIZATIONWISE;
end;
```

12) PROCEDURE TO GET FAILURE RATE ORGANIZATION-WISE

```
//12 - PROCEDURE TO GET FAILURE RATE ORGANIZATION-WISE

create or replace procedure GET_FAILURE_ORGANIZATIONWISE as
cursor c_details is select organization.organization_name
as name,count(*) as c from transaction,patient,doctor,organization
here transaction.patient_id=patient.patient_id and
patient.doctor_id=doctor.doctor_id and
doctor.organization_id=organization.organization_id and
transaction.status=0 group by organization.organization_name
order by c desc;
r_details c_details%ROWTYPE;
```

```
begin
OPEN c_details;
LOOP
FETCH c_details INTO r_details;
EXIT WHEN c_details%NOTFOUND;
dbms_output.put_line(r_details.name||','||r_details.c);
end loop;
close c_details;
end;

declare
begin
GET_FAILURE_ORGANIZATIONWISE;
end;
```

13) PROCEDURE TO ADD REGISTER TO DEATH

```
//13 - Procedure to Add Register To Death

create or replace procedure ADD_REGISTER_TO_DEATH(adhar1 varchar2)
as cursor c_death is select * from register_table;
r_death c_death%ROWTYPE;
count_done int:=0;
death_id int:=0;
begin
      open c_death;
      loop
             fetch c_death into r_death;
             EXIT WHEN c_death%NOTFOUND;
         if(r_death.adhar=adhar1)
     then
```

```
         insert into death_records values((select count(*)+1
      from death_records),r_death.name,'NIL',r_death.medical_history,
          r_death.medical_insurance,r_death.city,r_death.street,
          r_death.state,r_death.date_of_birth,0,'');
         count_done:=1;
         end if;
end loop;

close c_death;
  if(count_done=1)
    then
  dbms_output.put_line('STORED VALUE');
delete from register_table where adhar=adhar1;
  end if;


  if(count_done=0)
    then
  dbms_output.put_line('DATA NOT FOUND PLEASE CHECK YOUR INPUT AGAIN');
 end if;
end;

declare
  adhar varchar2(13):=:adhar;
begin
  ADD_REGISTER_TO_DEATH(adhar);
end;
```

FUNCTIONS

1) Function for finding the probability of success rate of operation organwise.

```
//1-Function for finding the probability of
    success rate of operation organised.
create or replace function organ_success return varchar as
cursor c_total is select distinct(organ_donated),count(*)
as c from transaction,donor where
donor.donor_id=transaction.donor_id group by organ_donated ;
cursor c_suc is select distinct(organ_donated),count(*)as
c from transaction,donor where
donor.donor_id=transaction.donor_id and
transaction.status=1 group by organ_donated ;
r_total c_total%ROWTYPE;
r_suc c_suc%ROWTYPE;
cou int:=1;

o1 int;
o2 int;
o3 int;
o4 int;
o5 int;

po1 int;
po2 int;
po3 int;
po4 int;
po5 int;
```

```
p1 float(5);
p2 float(5);
p3 float(5);
p4 float(5);
p5 float(5);

begin
for r_total in c_total loop

if(cou=1) then
o1:=r_total.c;
elsif(cou=2) then
o2:=r_total.c;
elsif(cou=3) then

o3:=r_total.c;
elsif(cou=4) then
o4:=r_total.c;
else
o5:=r_total.c;
end if;
cou:=cou+1;
end loop;

cou:=1;

for r_suc in c_suc loop

if(cou=1) then
```

```
po1:=r_suc.c;
elsif(cou=2) then
po2:=r_suc.c;
elsif(cou=3) then
po3:=r_suc.c;
elsif(cou=4) then
po4:=r_suc.c;
else
po5:=r_suc.c;
end if;
cou:=cou+1;
end loop;
p1:=po1/o1;
p2:=po2/o2;

p3:=po3/o3;
p4:=po4/o4;
p5:=po5/o5;
return '0'||to_char(p1)||',0'||to_char(p2)||',
0'||to_char(p3)||',0'||to_char(p4)||',0'||to_char(p5);
end;

declare
z varchar(100);
begin
z:=organ_success;
dbms_output.put_line(z);
end;
```

2) Function for finding the probability of failure rate of operation organwise.

```
//2 - Function for finding the probability of
      the failure rate of operation organ-wise.

create or replace function organ_failure return varchar as
cursor c_total is select distinct(organ_donated),count(*)
as c from transaction,donor where
donor.donor_id=transaction.donor_id group by organ_donated ;
cursor c_suc is select distinct(organ_donated),count(*)as
c from transaction,donor where
donor.donor_id=transaction.donor_id and
transaction.status=0 group by organ_donated ;
r_total c_total%ROWTYPE;
r_suc c_suc%ROWTYPE;
cou int:=1;


o1 int;
o2 int;
o3 int;
o4 int;
o5 int;

po1 int;
po2 int;
po3 int;
po4 int;
po5 int;
```

```
p1 float(5);
p2 float(5);
p3 float(5);
p4 float(5);
p5 float(5);

begin
for r_total in c_total loop

if(cou=1) then
o1:=r_total.c;
elsif(cou=2) then
o2:=r_total.c;
elsif(cou=3) then

o3:=r_total.c;
elsif(cou=4) then
o4:=r_total.c;
else
o5:=r_total.c;
end if;
cou:=cou+1;
end loop;

cou:=1;

for r_suc in c_suc loop
```

```
if(cou=1) then
po1:=r_suc.c;
elsif(cou=2) then
po2:=r_suc.c;
elsif(cou=3) then
po3:=r_suc.c;
elsif(cou=4) then
po4:=r_suc.c;
else
po5:=r_suc.c;
end if;
cou:=cou+1;
end loop;
p1:=po1/o1;

p2:=po2/o2;
p3:=po3/o3;
p4:=po4/o4;
p5:=po5/o5;
return '0'||to_char(p1)||',0'||to_char(p2)||',
0'||to_char(p3)||',0'||to_char(p4)||',0'||to_char(p5);
end;

declare
z varchar(100);
begin
z:=organ_failure;
dbms_output.put_line(z);
end;
```

3) Function to get Supply Organwise

```
//3 - Function to get supply organ-wise

create or replace function organ_supply return varchar as
cursor c_suc is select distinct(organ_donated),count(*)
as c from donor where donor_id NOT IN (select donor_id
from transaction) group by organ_donated;
r_suc c_suc%ROWTYPE;
cou int:=1;
o1 int;
o2 int;
o3 int;
o4 int;
o5 int;


begin
for r_suc in c_suc loop

          if(cou=1) then
                o1:=r_suc.c;
          elsif(cou=2) then
                o2:=r_suc.c;
          elsif(cou=3) then
                o3:=r_suc.c;
          elsif(cou=4) then
                o4:=r_suc.c;
          else
                o5:=r_suc.c;
   end if;
```

```
cou:=cou+1;
end loop;
return to_char(o1)||','||to_char(o2)||',
'||to_char(o3)||','||to_char(o4)||',
'||to_char(o5);
end;

declare
z varchar(100);
begin
z:=organ_supply;
dbms_output.put_line(z);
end;
```

4) Function to get Demand organwise

```
//4 - Function to get Demand Organ-wise

create or replace function organ_demand return varchar as
cursor c_suc is select distinct(organ_req),count(*)
as c from patient where patient_id
NOT IN (select patient_id from transaction)
group by organ_req;
r_suc c_suc%ROWTYPE;
cou int:=1;
o1 int;
o2 int;
o3 int;
o4 int;
o5 int;
```

```
begin
for r_suc in c_suc loop

     if(cou=1) then
            o1:=r_suc.c;
     elsif(cou=2) then
            o2:=r_suc.c;
     elsif(cou=3) then
            o3:=r_suc.c;
     elsif(cou=4) then
            o4:=r_suc.c;
     else
            o5:=r_suc.c;
     end if;

cou:=cou+1;
end loop;
return to_char(o1)||','||to_char(o2)||',
'||to_char(o3)||','||to_char(o4)||',
'||to_char(o5);
end;

declare
z varchar(100);
begin
z:=organ_demand;
dbms_output.put_line(z);
end;
```

TRIGGERS

1) Trigger for Adding Donor Information to Log Table

```
//1 Trigger for adding Donor information to Log table.
create or replace trigger ADD_DONOR_LOG after insert on Donor
for each row
begin
insert into log values(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI: SS'),
'Inserted new Donor : ', :new.Donor_Id );
end;
```

2) Trigger for Adding Update Action Information in Log Table

```
//2  Trigger for adding "Update" action information in Log table
create trigger UPD_DONOR_LOG
after update
on Donor
for each row
begin
insert into log values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI: SS'), 'Updated Donor Details',
: new.Donor_Id);
end
```

3) Trigger for Adding Delete Action Information in Log Table(ONLY FOR LOG TABLE)

```
//3 Trigger for adding “Delete” action information in Log table.
create trigger DEL_DONOR_LOG
after delete
on Donor
for each row
begin
insert into log values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI: SS'), 'Deleted Donor',
: old.Donor_Id );
end
```

4) Trigger for Adding Add Patient Action Information in Log Table

```
//4Trigger for adding “Add patient” action information in Log table
create trigger ADD_PATIENT_LOG
after insert
on Patient
for each row
begin
insert into log values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI: SS'), 'Inserted new Patient'
, :new.Patient_Id );
end
```

5) Trigger for Adding Update Information Action Information in Log Table

```
//5 Trigger for adding “Update information” action information in Log table
create trigger UPD_PATIENT_LOG
after update
on Patient
for each row
begin
insert into log values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI:SS'),'Updated Patient Details',:new.Patient_Id);
end
```

6) Trigger for Adding Delete Information Action Information in Log Table(UPDATING LOG TABLE)

```
//6 Trigger for adding “Delete information” action information in Log table
create trigger DEL_PATIENT_LOG
after delete
on PATIENT
for each row
begin
insert into log values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI: SS'), 'Deleted Patient ',
: old.Donor_Id );
end
```

7) Trigger for Adding Add Transaction Action Information in Log Table

```
//7 Trigger for adding "Add transaction" action information in Log table
create trigger ADD_TRASACTION_LOG
after insert
on Transaction
for each row
begin
insert into log values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI:SS'), 'Added Transaction ::
Patient ID,Donor ID : ',:new.Patient_ID || :new.Donor_ID );
end
```

8) Trigger to Insert or Update Government Approved Organization with 1 or 0.

```
//8 Trigger to insert or update government approved with 1 or 0
create or replace trigger validation_org_govt_approved
before insert or update on the organization
for each row
begin
      if inserting then
          if (:new.Government_approved=0 or :new.Government_approved=1)  then

              dbms_output.put_line('Got it');
        else
           raise_application_error(-20001,'Invalid Item Type....');
```

```
         end if;
      end if;
      if updating then
        if (:new.Government_approved=0 or :new.Government_approved=1) then
              dbms_output.put_line('Got it');
            else
           raise_application_error(-20001,'Invalid Item Type....');
end if;
      end if;
end;
insert into organization values((select count(*)+1 from organization),'NEW','ALSO',1);
```

9) Trigger to Insert or Update Status with 1 or 0

```
//9  Trigger to insert or update status with 1 or 0
create or replace trigger validation_transaction_status
before insert or update on transaction
for each row
begin
      if inserting then
          if (:new.status=0 or :new.status=1)  then

              dbms_output.put_line('Got it');
        else
           raise_application_error(-20001,'Invalid Item Type....')
```

```
         end if;
      end if;

      if updating then
        if (:new.status=0 or :new.status=1) then
               dbms_output.put_line('Got it');
             else
           raise_application_error(-20001,'Invalid Item Type....');
end if;
      end if;
end;
```

10) Trigger to Delete Transaction

```
//10 Trigger to delete transaction
create trigger DEL_TRANSACTION_LOG
after delete
on transaction
for each row
begin
insert into log values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI:SS'), 'DELETED Transaction ::
Patient ID,Donor ID : ',:old.Patient_ID || :old.Donor_ID );
end
```

11) Trigger to Add Perspective

```
//11 Trigger to add perspective
create or replace trigger ADD_DEATH_PERSPECTIVE
before insert on death_records
for each row
begin
     if (:new.donate_status=0) then
         :new.perspective:='Not approched';

     elsif (:new.donate_status=1) then
         :new.perspective:='approched-No';

     elsif (:new.donate_status=2) then

        :new.perspective:='approched-Yes';
    else
raise_application_error(-20001,'Invalid Item Type....');
end if;
end;
```

TRIGGERS FOR ADDING DATA IN RECOVERY TABLE

12) Trigger for Deletion Of User1

```
//12 Trigger for deletion of user1
create or replace trigger DEL_USER_LOG
after delete
on USER1
for each row
begin
insert into log values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI:SS'),
'Deleted user',:old.user_Id );
insert into recovery_table values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI:SS'), 'USER1',


:old.user_id||','||:old.name||','||:old.DATE_OF_BIRTH||',
'||:old.MEDICAL_INSURANCE||','||:old.MEDICAL_HISTORY||',
'||:old.street||','||:old.city||','||:old.state);
end ;
```

13) Trigger for Deletion Of Patient

```
//13 Trigger for deletion of patient
create or replace trigger DEL_PATIENT_LOG
after delete
on patient
for each row
begin
insert into log values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI:SS'),
'Deleted patient',:old.patient_Id );
insert into recovery_table values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI:SS'), 'PATIENT',

:old.patient_id||','||:old.organ_req||',
'||:old.REASON_OF_PROCUREMENT||',
'||:old.doctor_id||','||:old.user_id);
end ;
```

14) Trigger for Deletion Of Donor

```
//14 Trigger for deletion of donor
create or replace trigger DEL_DONOR_LOG
after delete
on Donor
for each row
begin
insert into log values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI:SS'),
'Deleted Donor',:old.Donor_Id );
insert into recovery_table values


(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI:SS'),
 'Donor', :old.Donor_Id||','||:old.ORGAN_DONATED||',
'||:old.REASON_OF_DONATION||',
'||:old.ORGANIZATION_ID||','||:old.USER_ID );
end ;
```

15) Trigger for Deletion Of Doctor

```
//15 Trigger for deletion of doctor
create or replace trigger DEL_DOCTOR_LOG
after delete
on DOCTOR
for each row
begin
insert into log values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI:SS'),
'Deleted DOCTOR',:old.DOCTOR_Id );
insert into recovery_table values


(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI:SS'),
'DOCTOR', :old.doctor_id||','||:old.doctor_name||',
'||:old.department_name||','||:old.organization_id);
end ;
```

16) Trigger for Deletion Of Organization

```
//16 Trigger for deletion of organization
create or replace trigger DEL_ORGANIZATION_LOG
after delete
on ORGANIZATION
for each row
begin
insert into log values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI: SS'),
'Deleted ORGANIZATION',: old.ORGANIZATION_Id );
insert into recovery_table values


(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI:SS'), 'ORGANIZATION',
:old.ORGANIZATION_id||','||:old.ORGANIZATION_name||',
'||:old.location||','||:old.GOVERNMENT_APPROVED);
end ;
```

17) Trigger for Deletion Of Organ Available

```
//17 Trigger for deletion of organ_available
create or replace trigger DEL_ORG_AVAILABLE_LOG
after delete
on ORGAN_AVAILABLE
for each row
begin
insert into log values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI: SS'),
'Deleted ORGAN_AVAILABLE',: old.ORGAN_Id );


insert into recovery_table values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI:SS'),
'ORGAN_AVAILABLE', :old.ORGAN_id||',
'||:old.ORGAN_name||','||:old.DONOR_ID);
end ;
```

18) Trigger for Deletion Of Transaction

```
//18 Trigger for deletion of transaction(ERROR)
create or replace trigger DEL_TRANSACTION_LOG
after delete
on TRANSACTION
for each row
begin
insert into log values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI:SS'),
'Deleted TRANSACTION',:old.patient_Id||','||:old.donor_id );


insert into recovery_table values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI:SS'),
'TRANSACTION', :old.patient_id||','||:old.organ_id||',
'||:old.DONOR_ID||','||:old.date_of_transaction||',
'||:old.status);
end ;
```

19) Trigger for Deletion Of Organization Head

```
//19 Trigger for deletion of organization_head
create or replace trigger DEL_ORGANIZATION_HEAD_LOG
after delete
on ORGANIZATION_HEAD
for each row
begin
insert into log values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI: SS'),
'Deleted ORGANIZATION_HEAD',:old.organization_id );


insert into recovery_table values
(TO_CHAR(CURRENT_DATE, 'DD-MON-YYYY HH:MI:SS'),
'ORGANIZATION_HEAD', :old.organization_id||',
'||:old.employee_id||','||:old.name||',
'||:old.date_of_joining||','||:old.term_length);
end ;
```

20) Trigger for Validating User's Medical Insurance

```
//20 Trigger for Validating User's Medical Insurance
create or replace trigger validation_user1_MED_INSURANCE
before insert or update on user1
for each row
begin
      if inserting then
          if (:new.MEDICAL_INSURANCE=0 or :new.MEDICAL_INSURANCE=1)
            then
               dbms_output.put_line('Got it');
        else
           raise_application_error(-20001,'Invalid Item Type....');
```

```
         end if;
      end if;

      if updating then
        if (:new.MEDICAL_INSURANCE=0 or :new.MEDICAL_INSURANCE=1) then
               dbms_output.put_line('Got it');
             else
           raise_application_error(-20001,'Invalid Item Type....');
end if;
      end if;
end;
```

TRIGGERS FOR REGISTER TABLE

21) Trigger for inserting or updating phone number in register table

```
//21 - Trigger for inserting or updating phone number in register table

create or replace trigger phone_no_reg
before insert or update
on register_table
for each row
begin
if inserting then
if(length(:new.phone_no)=10 and :new.phone_no is not null
and (:new.phone_no like('%9') or :new.phone_no like('%8')
or :new.phone_no like('%7') or :new.phone_no like('%6') ))
then
dbms_output.put_line('Yes');
else
```

```
raise_application_error(-20001,'Invalid Phone Number .
Condition to be satisfied:starting with (6-9) and only 10 numbers');
end if;
end if;

if updating then
if(length(:new.phone_no)=10 and :new.phone_no is not null
and (:new.phone_no like('%9') or :new.phone_no like('%8')
or :new.phone_no like('%7') or :new.phone_no like('%6') ))
then
dbms_output.put_line('Yes');
else
raise_application_error(-20001,'Invalid Phone Number .
Condition to be satisfied:starting with (6-9) and only 10 numbers');


end if;
end if;
end;
```

22) Trigger for inserting or updating Medical Insurance Status in Register Table

```
//22 - Trigger for inserting or updating Medical Insurance
       Status in Register Table

create or replace trigger MED_INSURANCE_reg
before insert or update
on register_table
for each row
begin
      if inserting then
          if (:new.MEDICAL_INSURANCE=0 or :new.MEDICAL_INSURANCE=1)
      then
              dbms_output.put_line('Got it');
        else
           raise_application_error(-20002,'Invalid Item Type....');
```

```
          end if;
       end if;

       if updating then
         if (:new.MEDICAL_INSURANCE=0 or :new.MEDICAL_INSURANCE=1) then
                dbms_output.put_line('Got it');
              else
            raise_application_error(-20002,'Invalid Item Type....');

end if;
       end if;
end;
```

23) Trigger for inserting or updating Email-Id in register table

```
//23 - Trigger for inserting or updating email-Id in register table

create or replace trigger email_reg
before insert or update
on register_table
for each row
begin
if inserting then
if(:new.email like '%_@___%.__%' )
then
dbms_output.put_line('Yes');
else
raise_application_error(-20003,'Invalid Email_id ');
end if;
end if;
```

```
if updating then
if(:new.email like '%_@___%.__%' ) then
dbms_output.put_line('Yes');
else
raise_application_error(-20003,'Invalid Email_id ');
end if;
end if;
end;
```

24) Trigger for inserting or updating Aadhar Registration in Register Table

```
//24 - Trigger for inserting or updating
      Aadhar Registration in Register Table

create or replace trigger aadhar_reg
before insert or update
on register_table
for each row
begin
if inserting then
   if(length(:new.adhar)=12 ) then
        dbms_output.put_line('Yes');
   else
        raise_application_error(-20004,'Invalid Email_id ');
   end if;
end if;

if updating then
     if(length(:new.adhar)=12 ) then
          dbms_output.put_line('Yes');
     else
          raise_application_error(-20004,'Invalid Email_id ');
end if;
end if;
end;
```

25) Trigger for inserting or updating Blood Group Type in Register Table

```
//25 - Trigger for inserting or updating
       Blood Group Type in Register Table

create or replace trigger blood_group_reg
before insert or update
on register_table
for each row
begin
if inserting then
     if(:new.blood_group='A+' or :new.blood_group='O+'
         or :new.blood_group='B+' or :new.blood_group='AB+'
         or :new.blood_group='A-' or :new.blood_group='O-'
         or :new.blood_group='B-' or :new.blood_group='AB-')
     then


         dbms_output.put_line('Yes');
     else
         raise_application_error(-20005,'Invalid Email_id ');
    end if;
end if;

if updating then
     if(:new.blood_group='A+' or :new.blood_group='O+'
         or :new.blood_group='B+' or :new.blood_group='AB+'
         or :new.blood_group='A-' or :new.blood_group='O-'
         or :new.blood_group='B-' or :new.blood_group='AB-')
     then
         dbms_output.put_line('Yes');
     else


         raise_application_error(-20005,'Invalid Email_id ');
     end if;
  end if;
End;
```

TRIGGERS FOR INSERTION

26) Trigger for inserting Donor

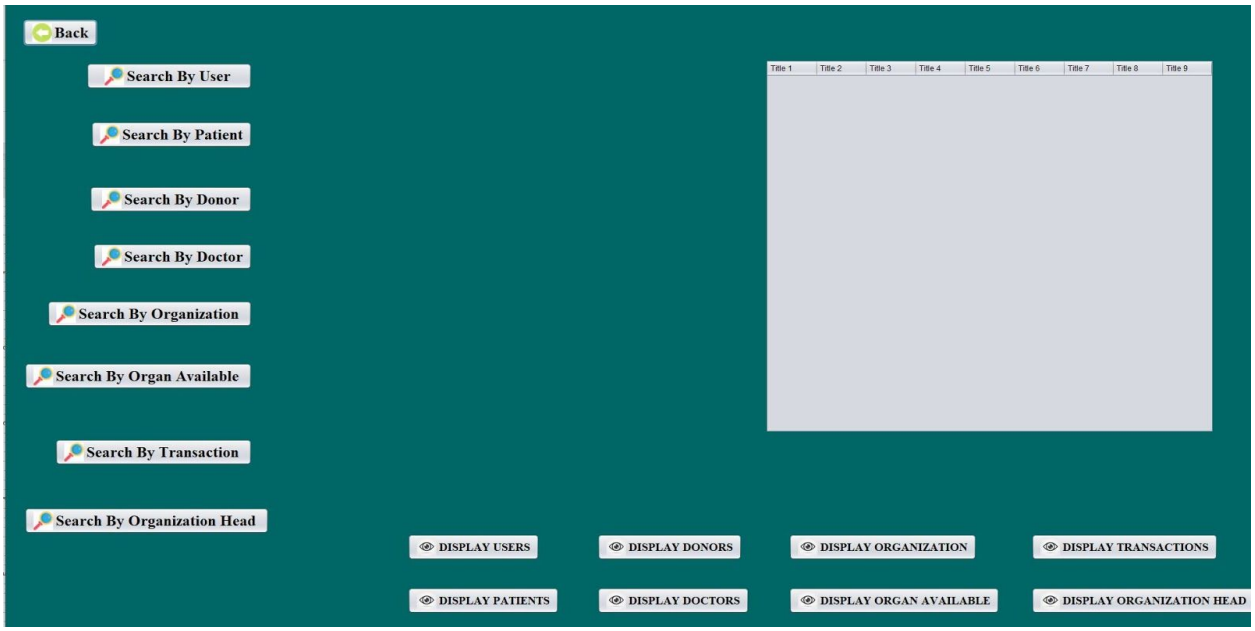
```
//26 - Trigger for inserting Donor

create or replace trigger insert_organ_donor
before insert or update
on donor
for each row
begin
if inserting then
       if(:new.organ_donated='Heart' or :new.organ_donated='Pancreas'
           or :new.organ_donated='Intestine' or :new.organ_donated='Lungs'
           or :new.organ_donated='Kidney')
       then
           dbms_output.put_line('Got it');
       else


           raise_application_error(-20002,
           'Enter the organ from this list only:
            Heart,Pancreas,Intestine,Lungs,Kidney');
        end if;
  end if;

if updating then
       if(:new.organ_donated='Heart' or :new.organ_donated='Pancreas'
           or :new.organ_donated='Intestine'
           or :new.organ_donated='Lungs'
           or :new.organ_donated='Kidney')
       then
           dbms_output.put_line('Got it');
       else


           raise_application_error(-20002,'Enter the organ
           from this list only:Heart,Pancreas,Intestine,Lungs,Kidney');
        end if;
    end if;
end;
```

27) Trigger for inserting Organ of Patient

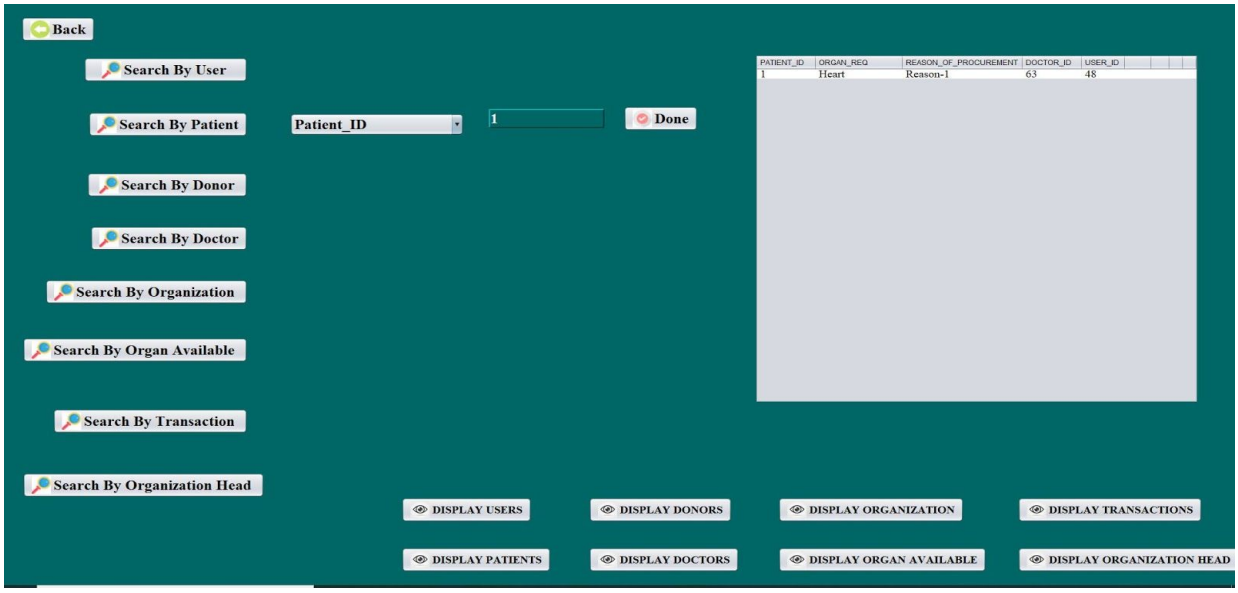
```
//27 -  Trigger for inserting Organ of Patient

create or replace trigger insert_organ_patient
before insert or update
on patient
for each row
begin
  if inserting then
        if(:new.organ_req='Heart' or :new.organ_req='Pancreas'
            or :new.organ_req='Intestine'
            or :new.organ_req='Lungs'
            or :new.organ_req='Kidney')
  then
        dbms_output.put_line('Got it');


  else
       raise_application_error(-20002,'Enter the organ
        from this list only:Heart,Pancreas,Intestine,Lungs,Kidney');
       end if;
  end if;

if updating then
     if(:new.organ_req='Heart' or :new.organ_req='Pancreas'
         or :new.organ_req='Intestine' or :new.organ_req='Lungs'
         or :new.organ_req='Kidney')
     then
         dbms_output.put_line('Got it');
     else
         raise_application_error(-20002,'Enter the organ
         from this list only:Heart,Pancreas,Intestine,Lungs,Kidney');


        end if;
   end if;
end;
```

SCREENSHOTS OF RESULTS GENERATED AFTER PROCEDURE AND FUNCTION ARE CALLED ON FRONT-END

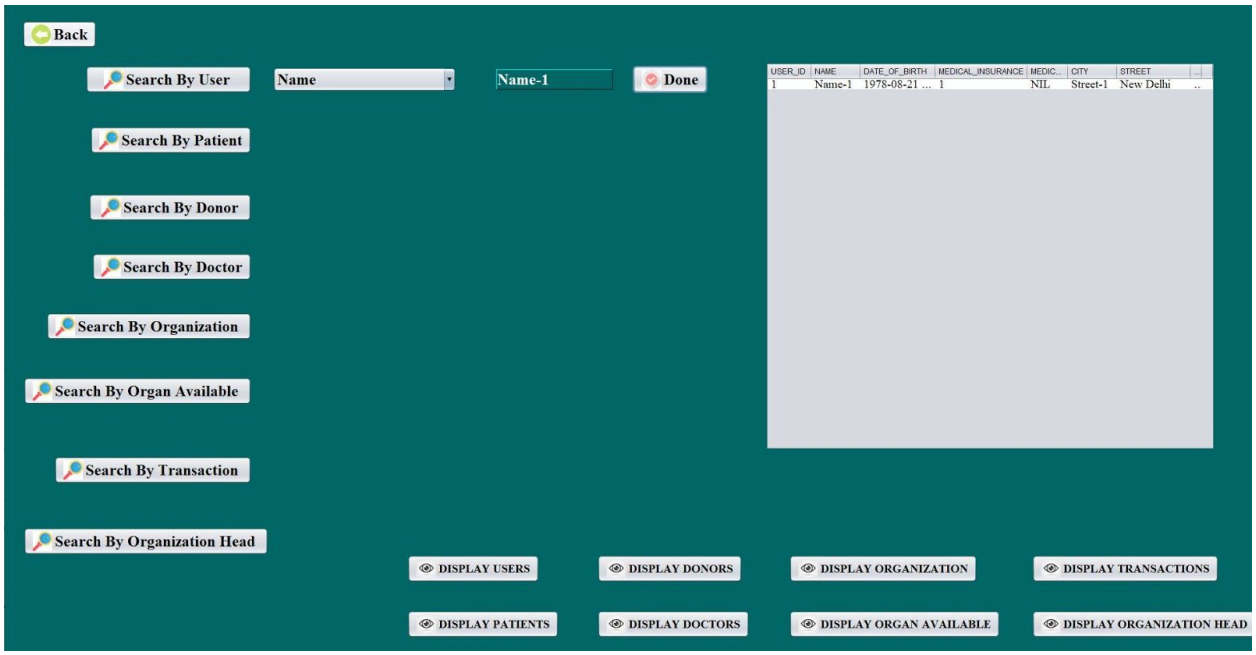
SEARCH SCREEN



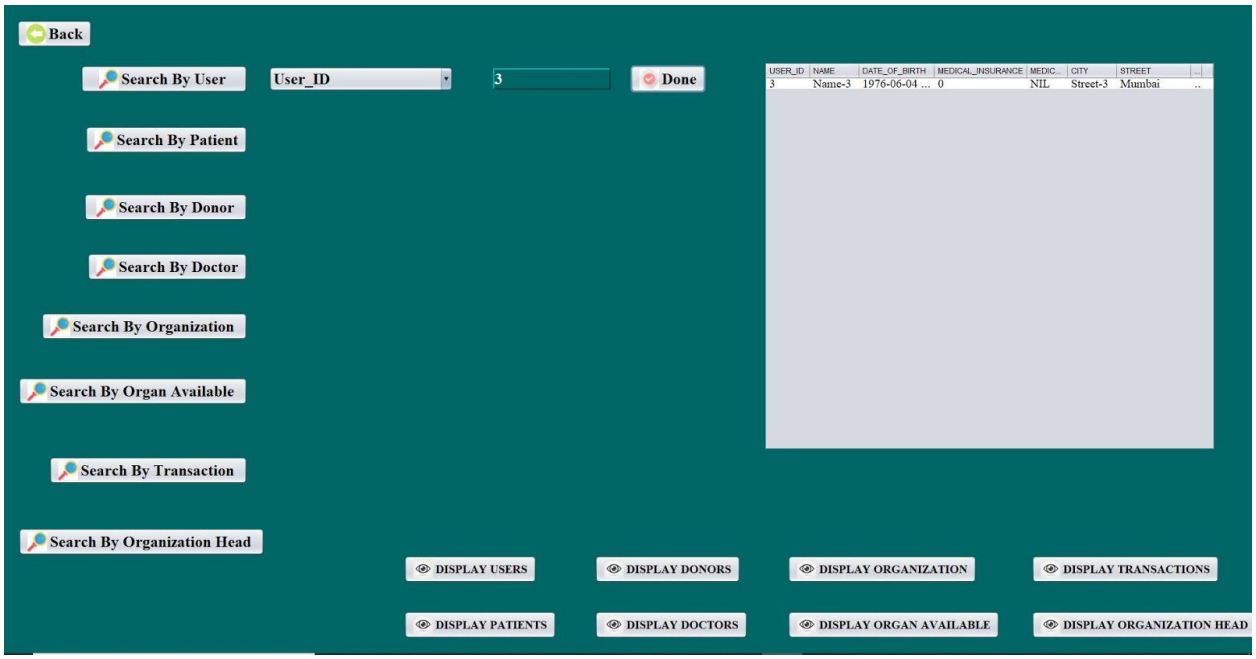
After pressing search by patient button, we see this screen i.e. SEARCH PATIENT BY ID



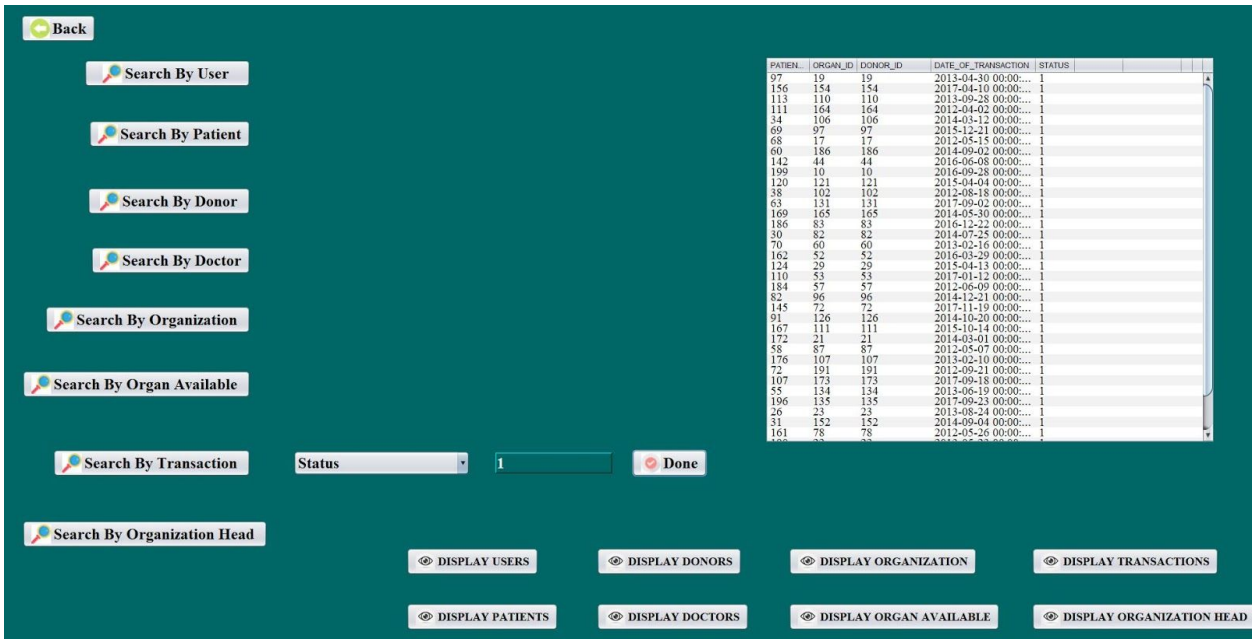
After pressing search by user button, we see this screen i.e. SEARCH USER BY NAME



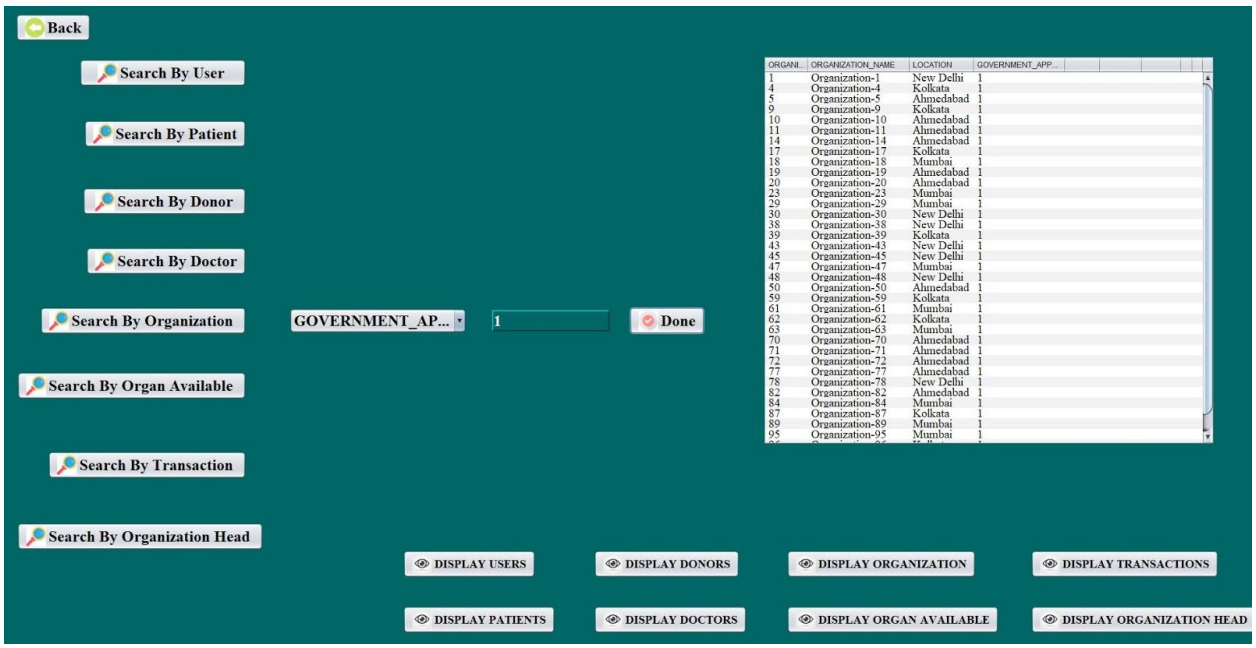
After pressing search by user ID button, we see this screen i.e. SEARCH USER BY USER ID



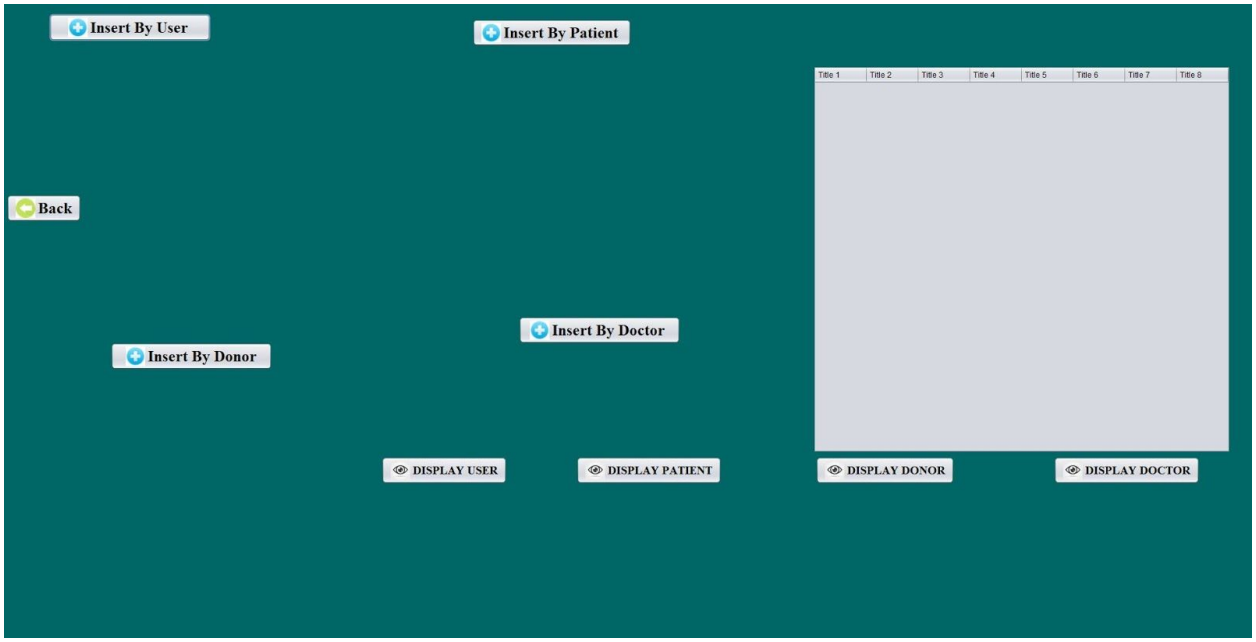
After pressing Search By Transaction and selecting status from the combo box and entering value 1, we get this screen i.e. SEARCH SUCCESSFUL TRANSACTION

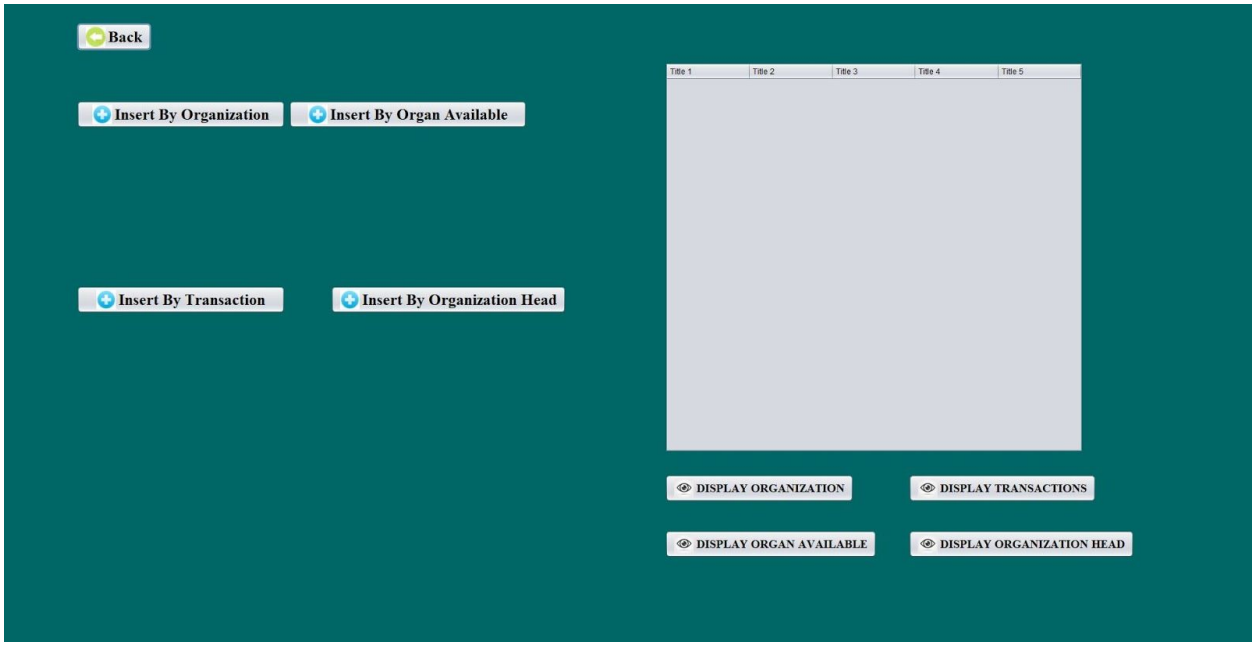


After clicking Search By Organization,selecting Government Approved from the combo box and entering value 1, we get this screen i.e SEARCH ORGANIZATION GOVERNMENT APPROVED



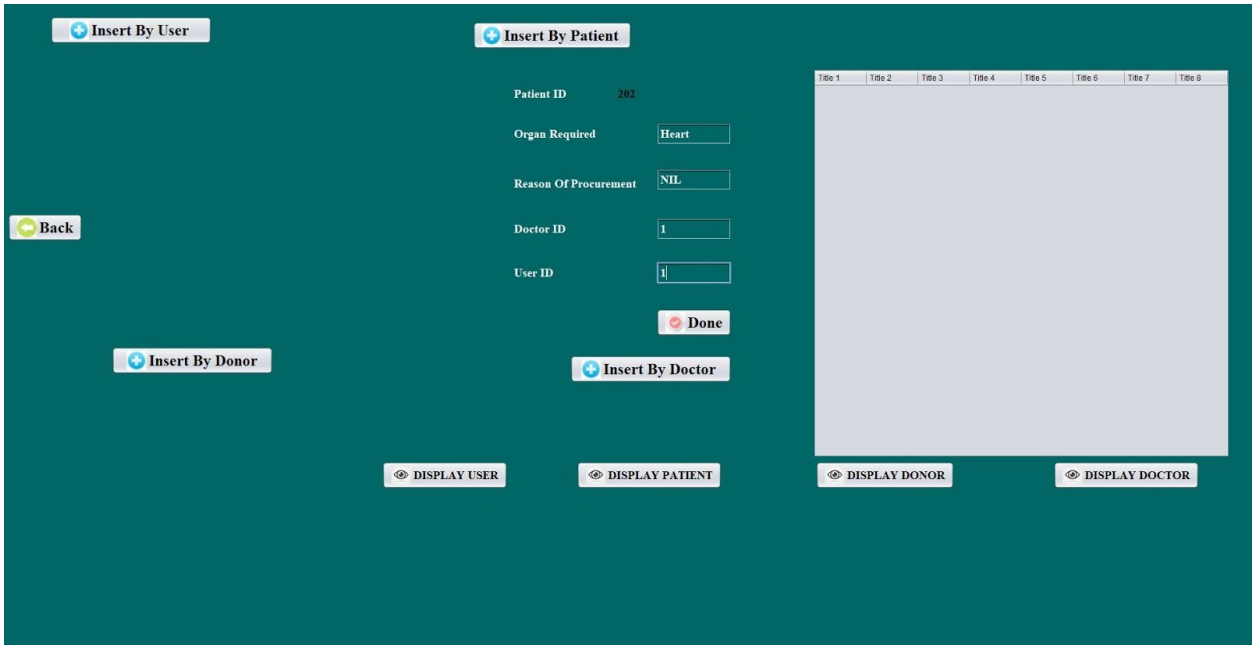
INSERT SCREEN 1 & 2



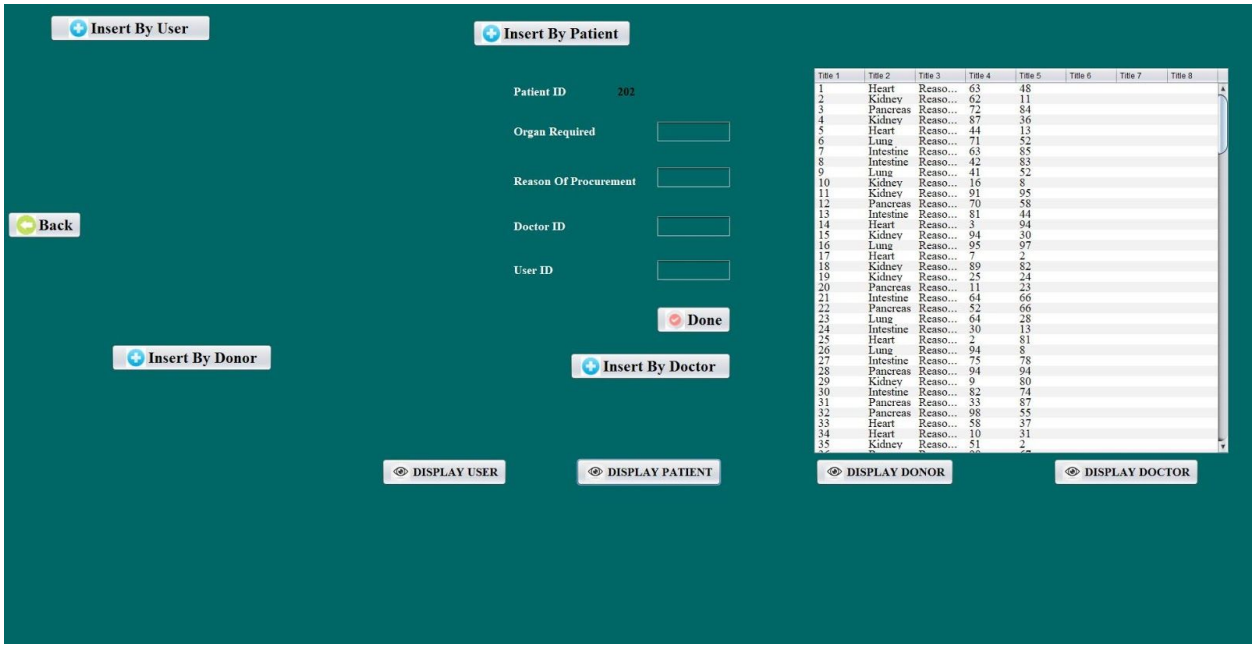


INSERT PATIENT BEFORE DONE

After pressing DONE button, the values will be inserted in the Patient Table.

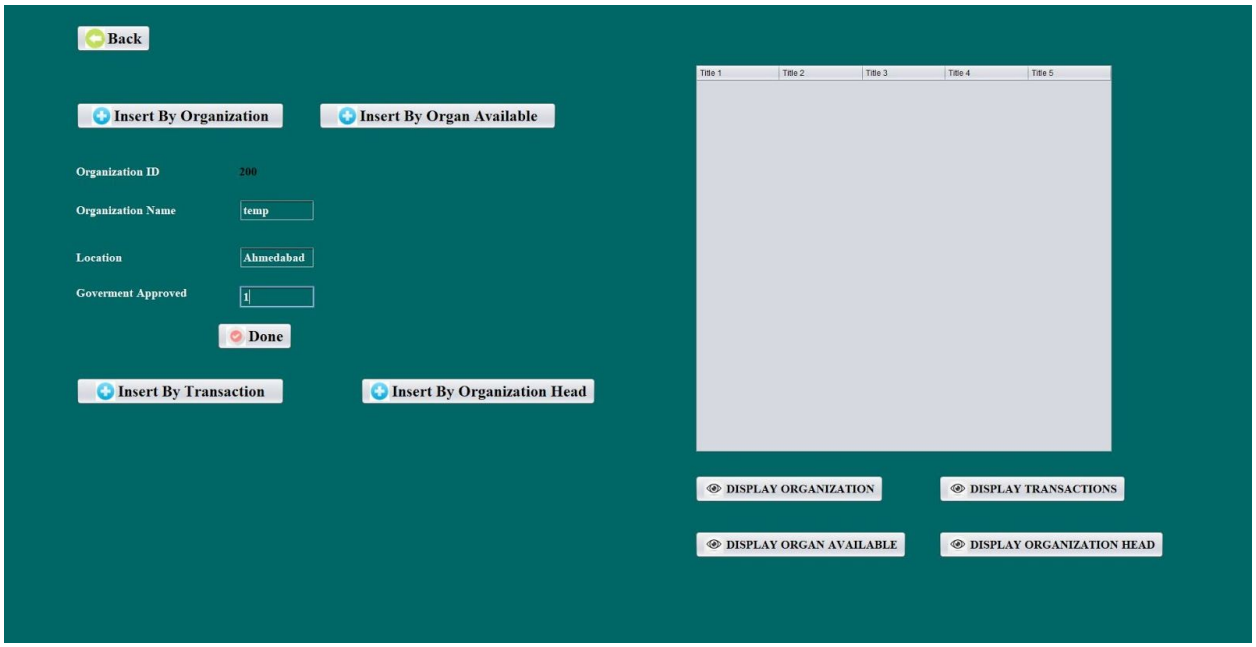


INSERT PATIENT DATA DISPLAY (NEW DATA INSERTED PATIENT ID:200)

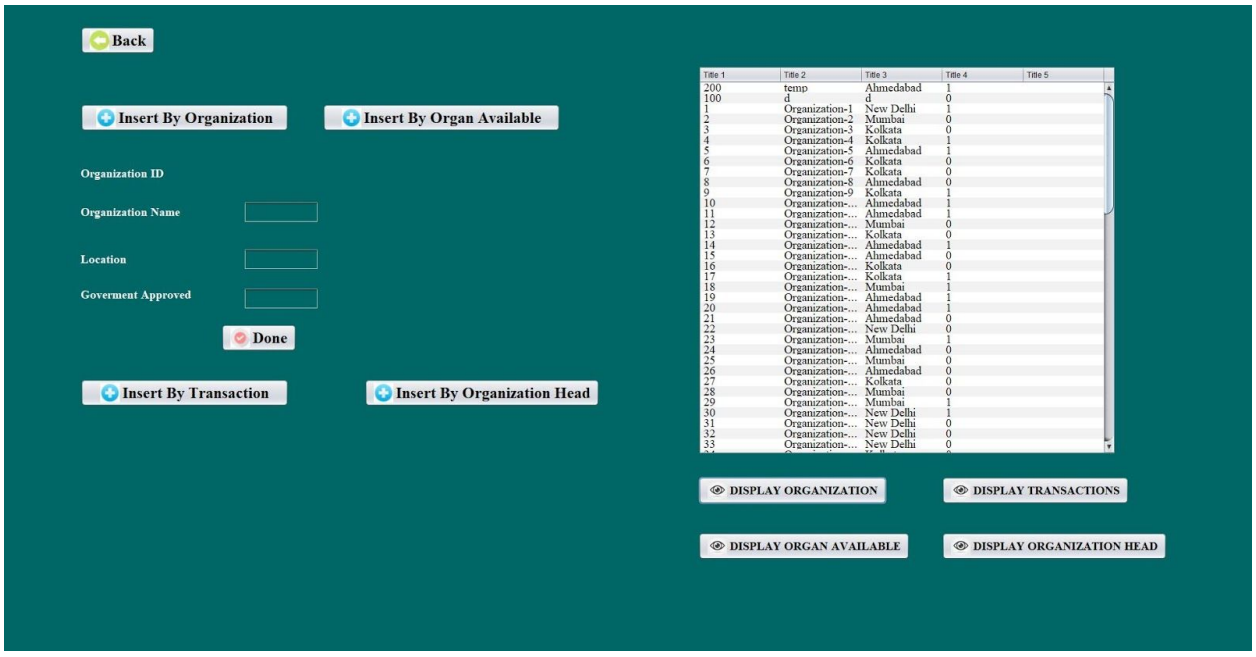


Press INSERT BY ORGANIZATION BUTTON so organization textfields will be visible.

After pressing DONE button, the values will be inserted in the ORGANIZATION Table.



VALUES INSERTED IN ORGANIZATION TABLE



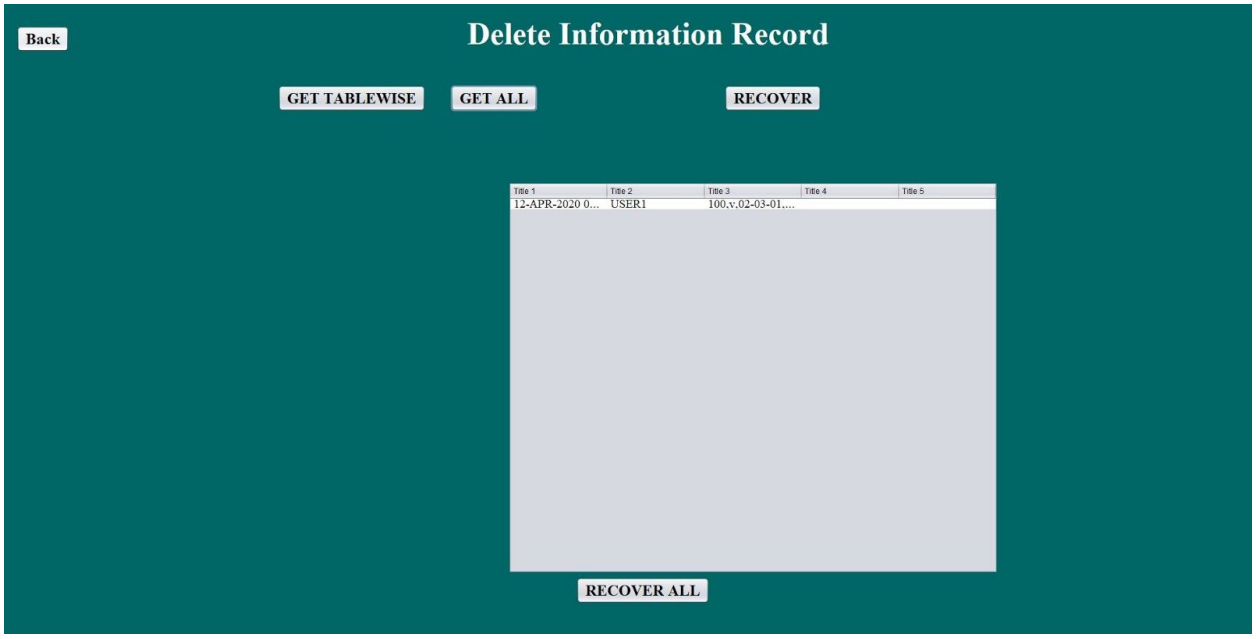
DELETE SCREEN:

PRESS DELETE BY USER BUTTON , the combo box of column names and textfield will be visible.

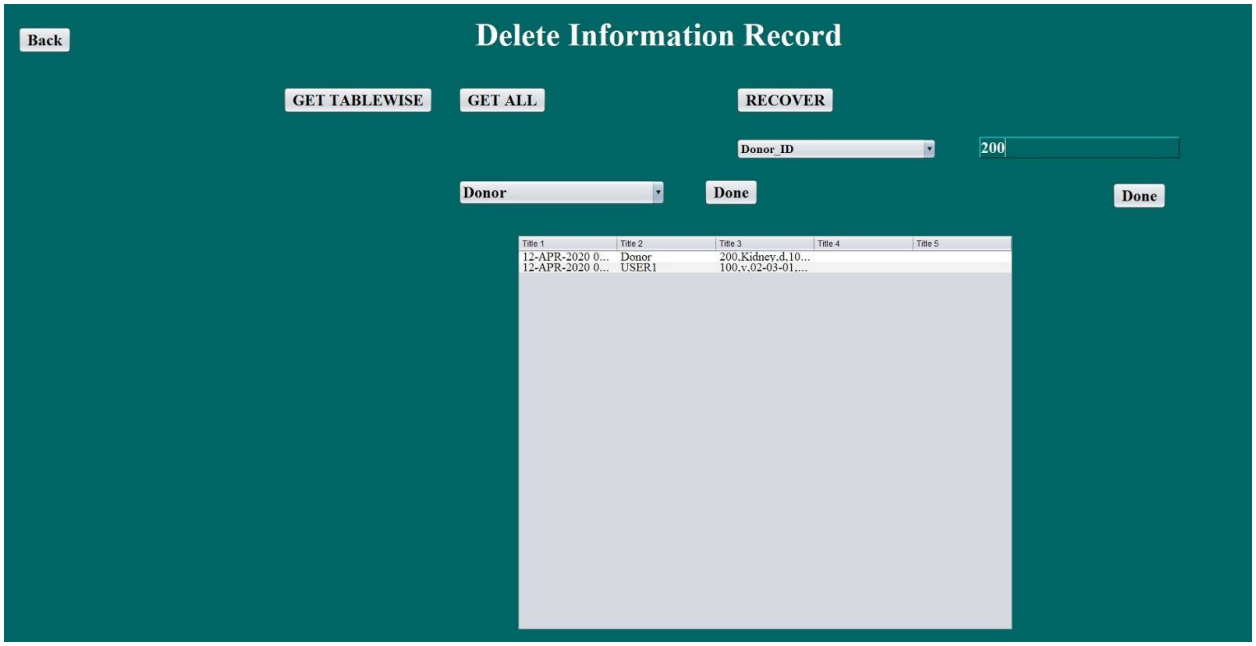
ENTER THE VALUE OF USER_ID WHICH YOU WANT TO DELETE, AND PRESS DONE BUTTON. AND THE VALUE HAVING USER_ID=100 WILL BE DELETED.

RECOVERY TABLE:

PRESS GET ALL BUTTON TO GET ALL THE DATA DELETED WITH RESPECTIVE TABLE NAME.



For recovering data, press recover button, then select table name,after that select column name and enter the information about data which you want to recover.



HENCE after pressing recover button, your data is recovered in donor table.

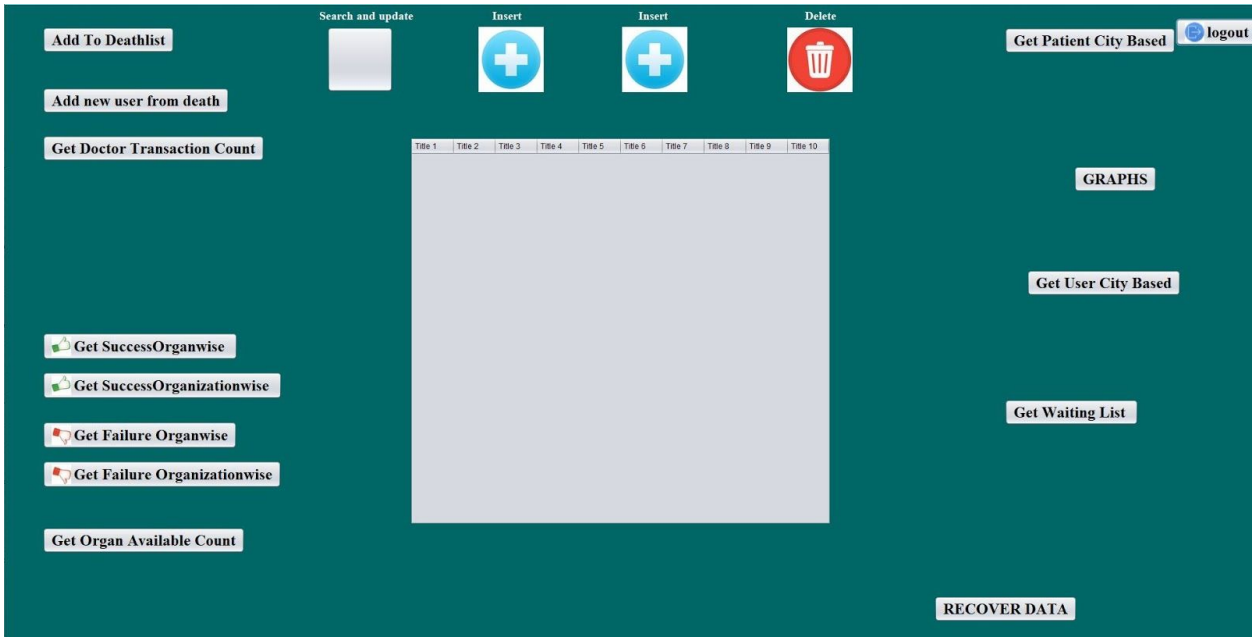
```
select * from donor where donor_id=200
```

Results Explain Describe Saved SQL History

DONOR_ID	ORGAN_DONATED	REASON_OF_DONATION	ORGANIZATION_ID	USER_ID
200	Kidney	d	100	2

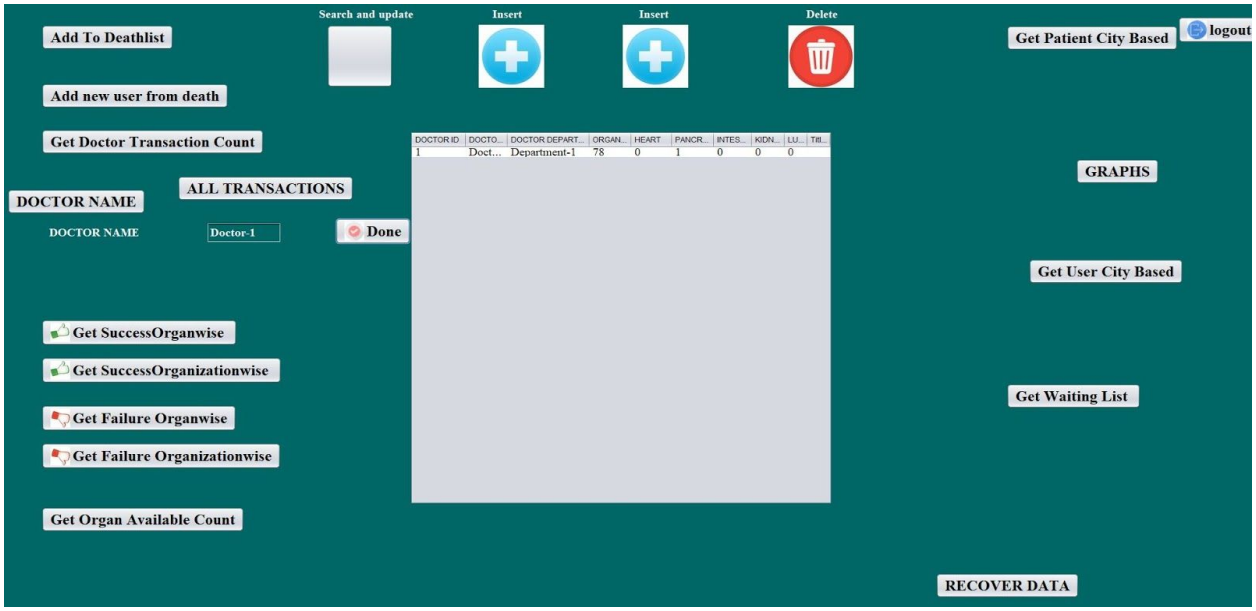
1 rows returned in 0.00 seconds Download

PROCEDURE SCREEN GUI:



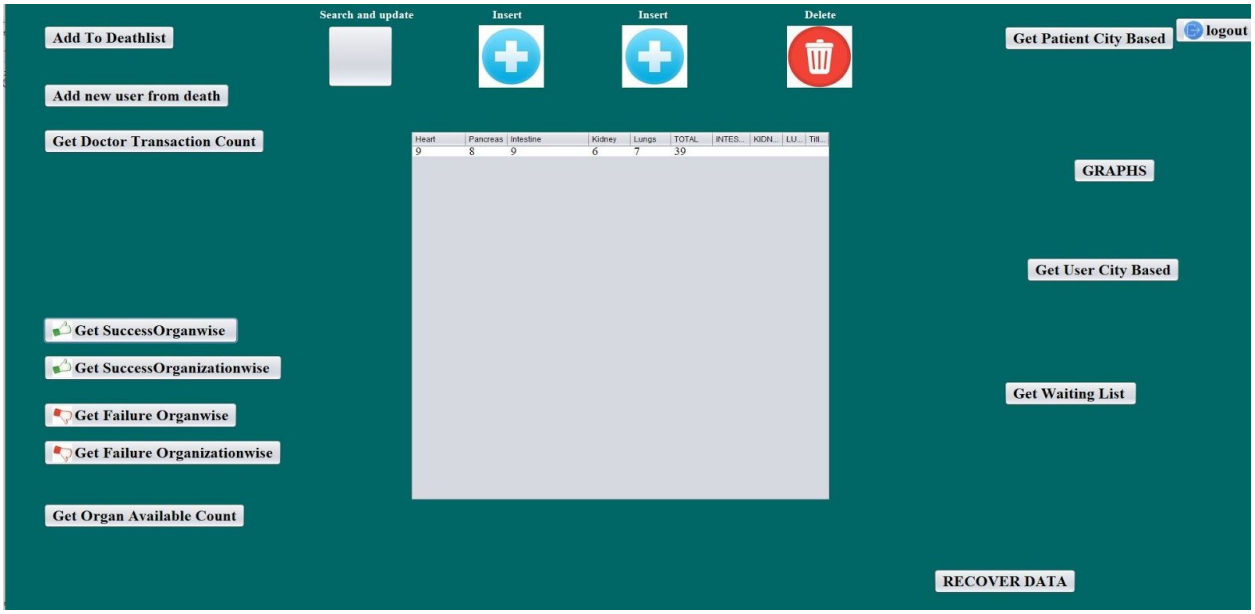
PROCEDURE TO GET DOCTOR TRANSACTION:

Press GET DOCTOR TRANSACTION COUNT BUTTON and then press the doctor button so you will be asked to enter the doctor name. After entering the press done button and you will get all transaction of doctor-1 organwise.



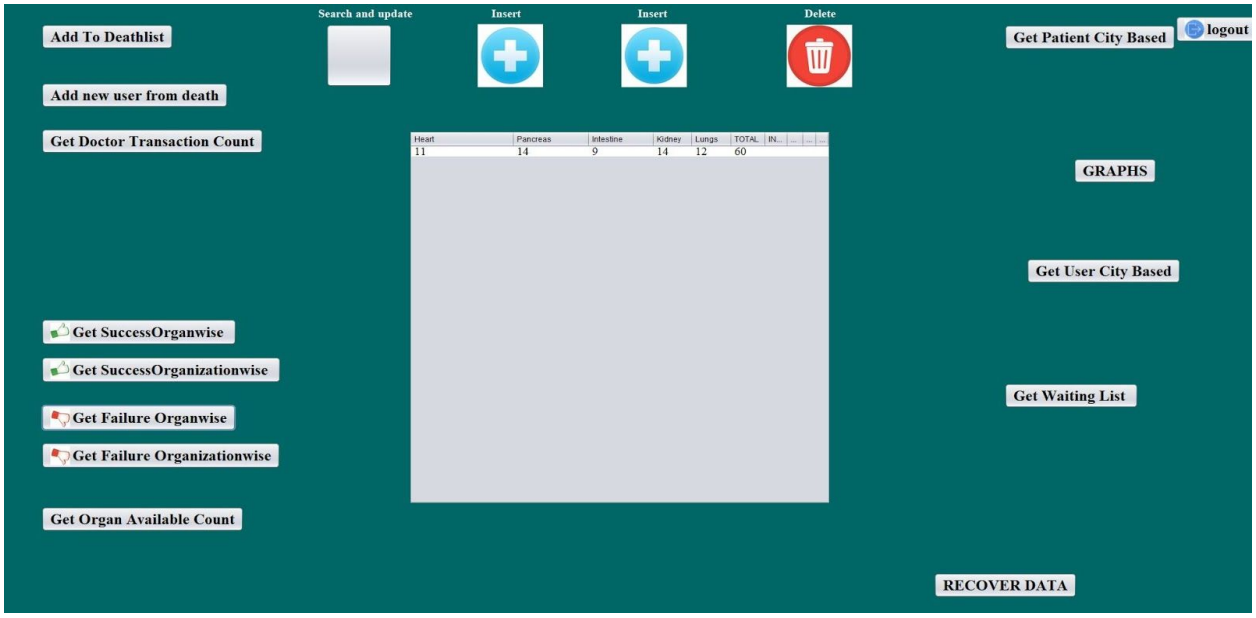
Get Success Organwise:

Press GET SUCCESS ORGANWISE BUTTON and you will get output of successful transactions organwise.



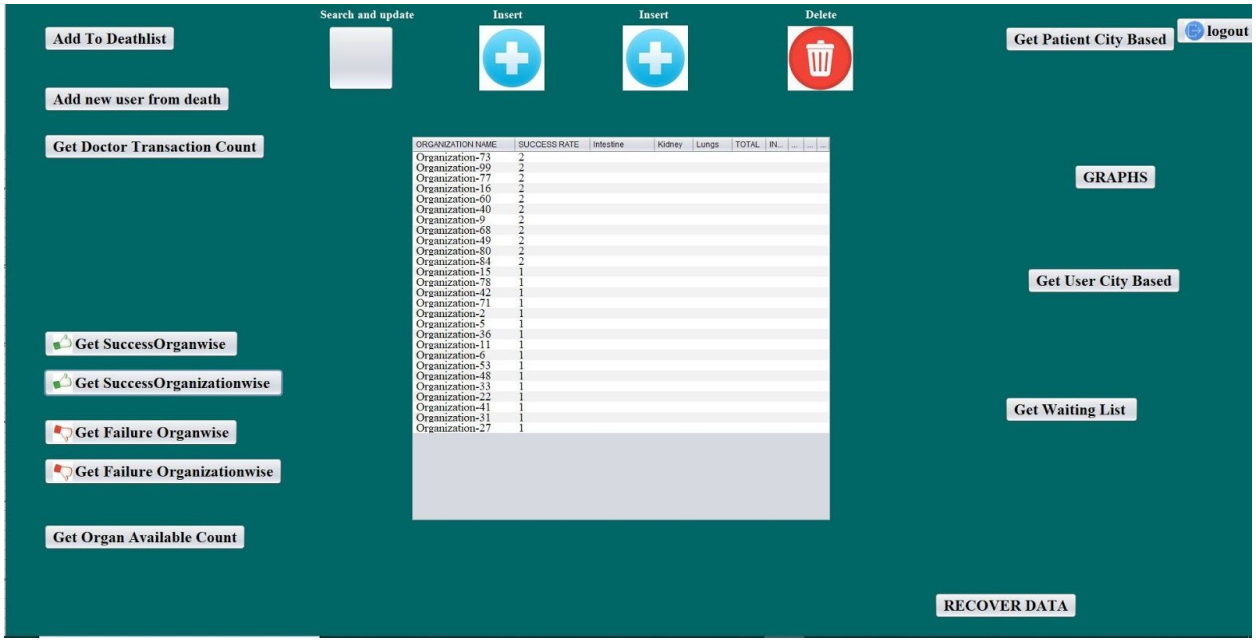
Get Failure Organwise:

Press GET FAILURE ORGANWISE BUTTON and you will get output of failed transactions organwise.



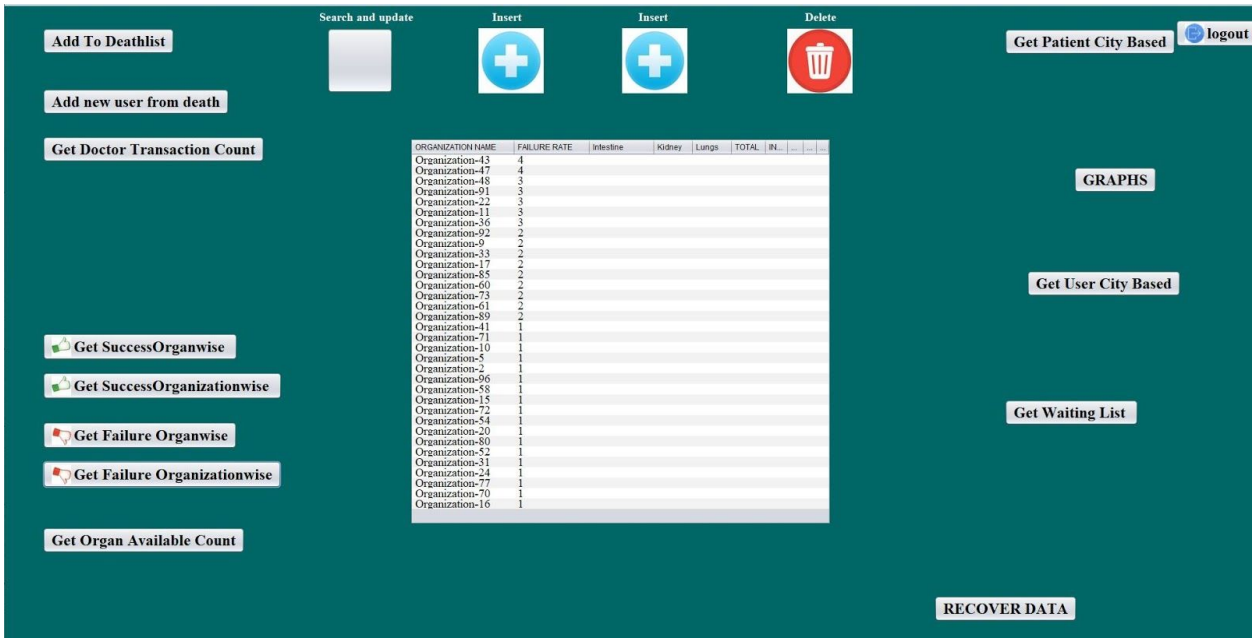
Get Success Organizationwise:

Press GET SUCCESS ORGANIZATIONWISE BUTTON and you will get output of successful transactions organorganization.



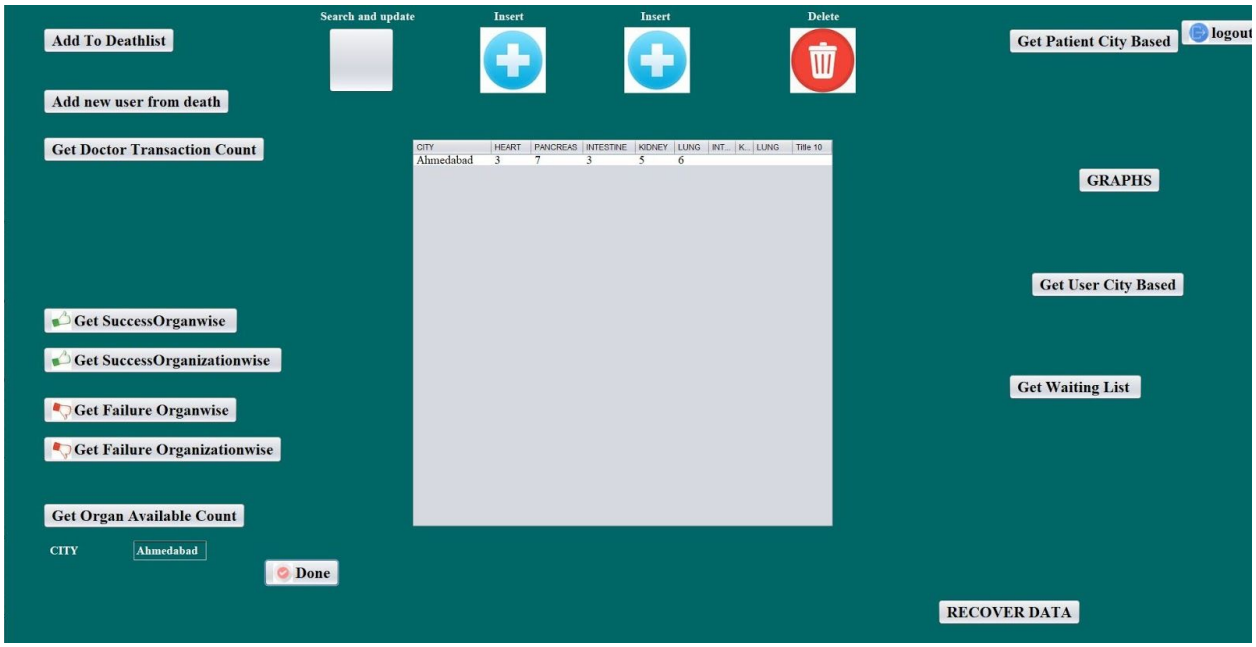
Get Failure Organizationwise:

Press GET Failure ORGANIZATIONWISE BUTTON and you will get output of failed transactions organorganization.



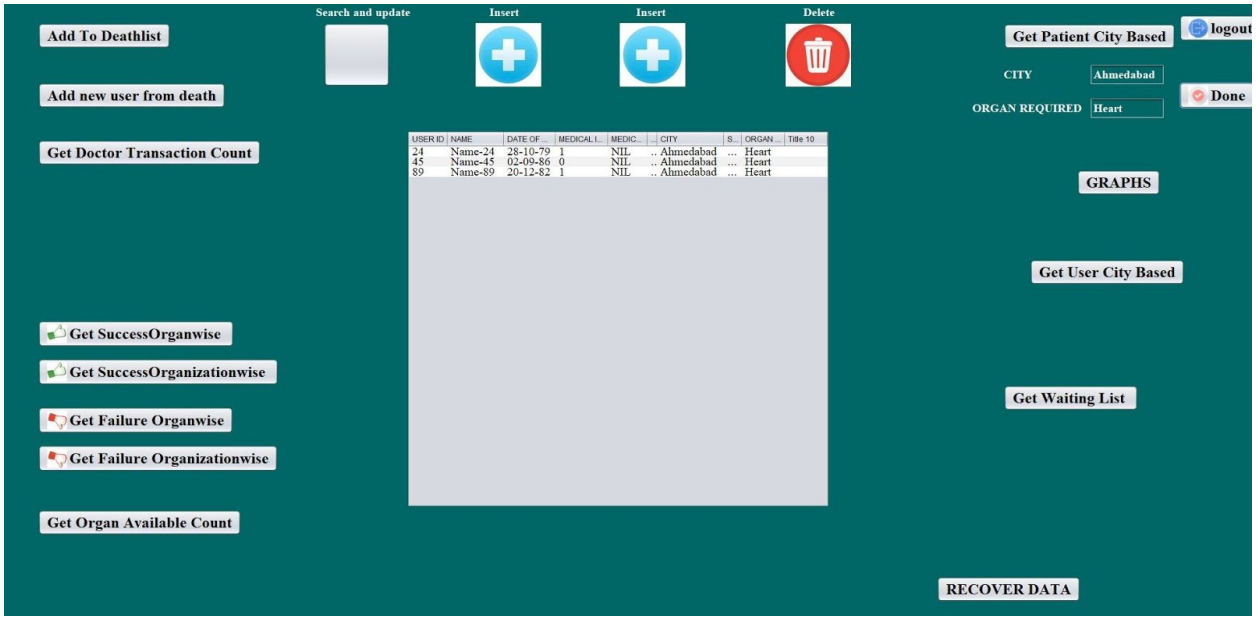
Get Organ Available City Wise:

Press GET ORGAN AVAILABLE BUTTON and you will be asked to enter the city. After that press done,output will be given as number of organs available in respective entered city.



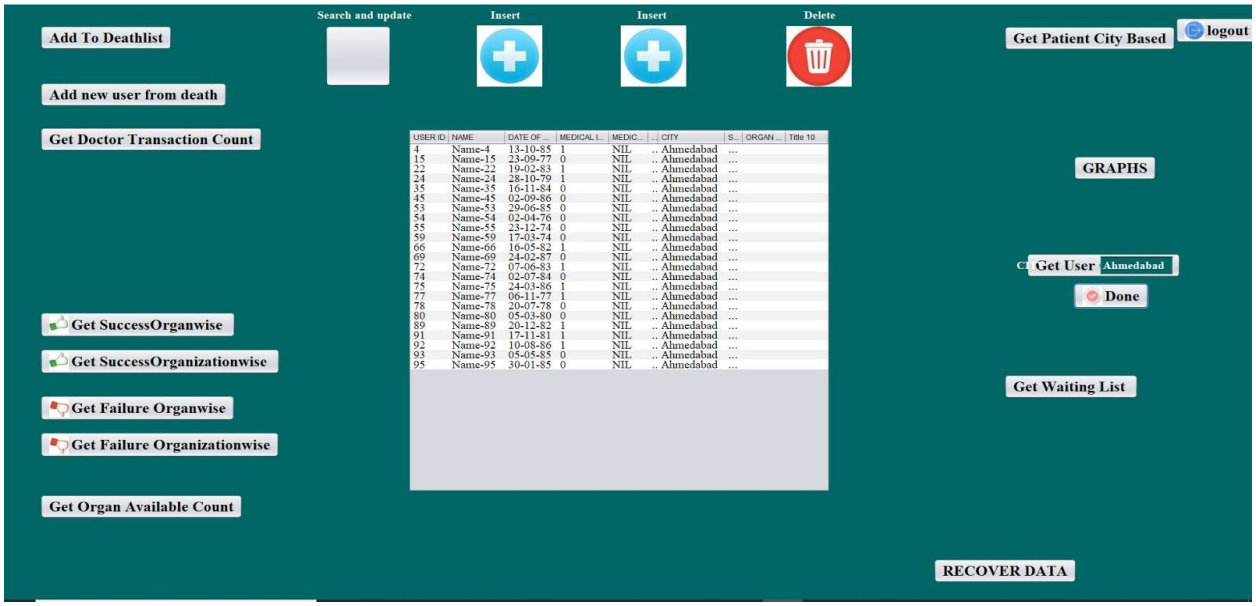
Get Patient City and Organ Wise:

Press GET PATIENT CITY AND ORGAN WISE BUTTON and you will be asked to enter city and organ required.Now press done button and all respective organs available in respective city will be displayed.



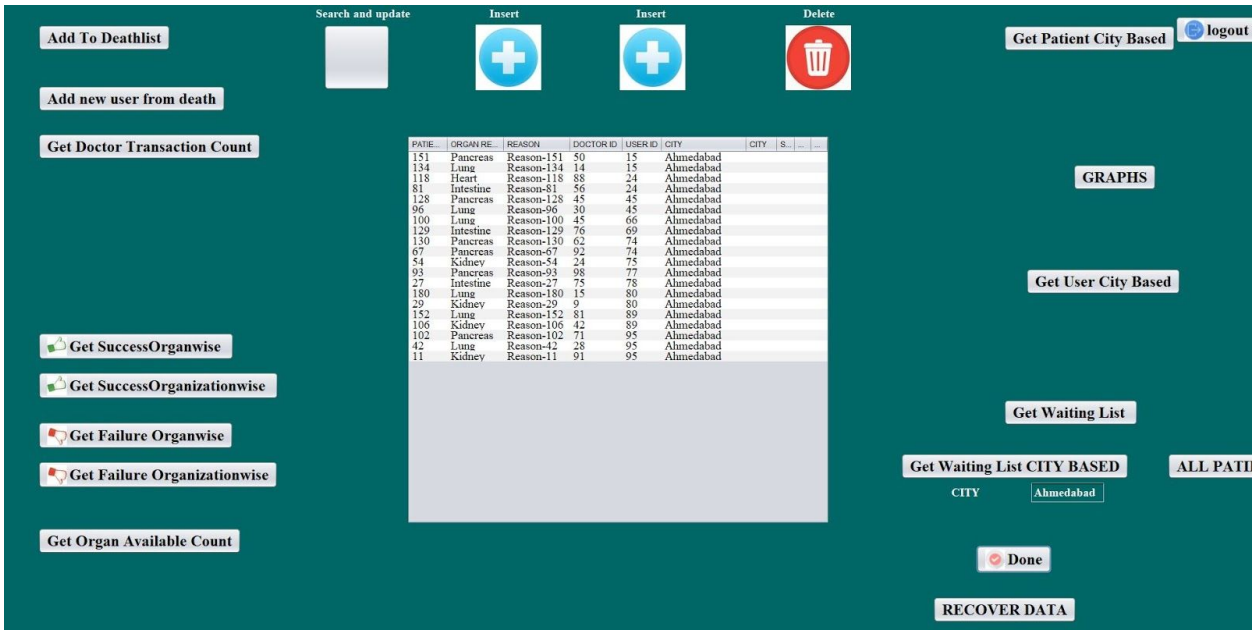
Get User City Based:

Press GET USER CITY BASED BUTTON and you will be asked to entered the city ,then press done button , now output will be displayed as all users of respective city.



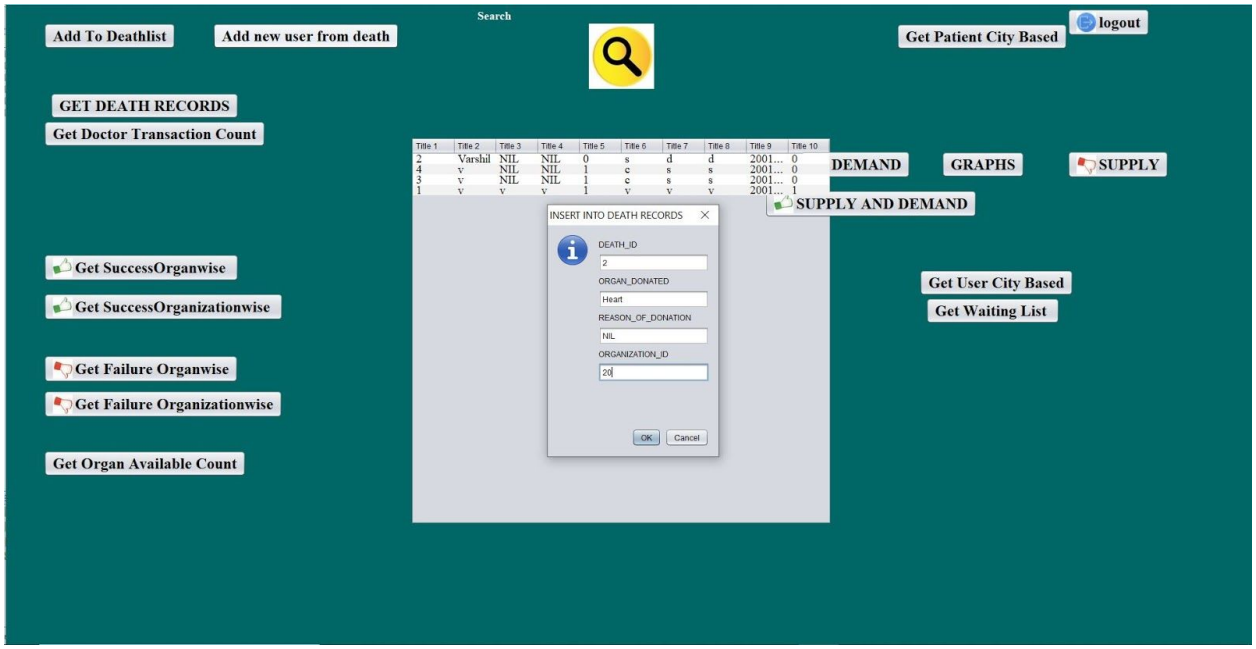
Get Waiting List City Based:

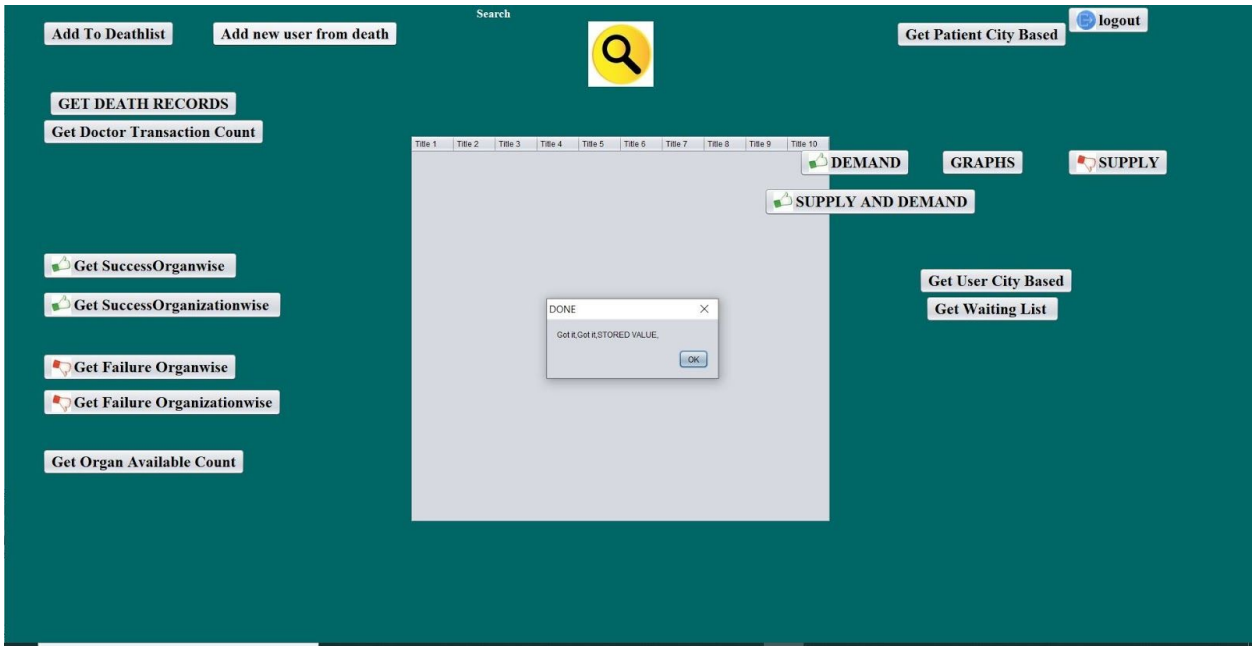
Press GET WAITING LIST BUTTON, then PRESS GET WAITING LIST CITYWISE BUTTON and you will be asked to enter the city. After that, Press Done button and output will be displayed as waiting list city wise.



ADD DATA FROM DEATH RECORD TABLE TO USER,DONOR:

This procedure is for ensuring the organ donation acceptance from the user after his/her death and adding it to the user and donor data thereafter.As shown in the image enter the death_id from the death_record table and press OK.It will store this data from death_record to user and donor table.





PROOF IN USER TABLE :

96	Name-96	02/04/1987	1	NIL	Street-96	Kolkata	West Bengal
97	Name-97	12/22/1985	1	NIL	Street-97	Kolkata	West Bengal
98	Name-98	08/06/1981	1	NIL	Street-98	Mumbai	Maharashtra
99	Name-99	01/15/1978	1	NIL	Street-99	Kolkata	West Bengal
100	Varshil	03/02/2001	0	NIL	d	s	d

PROOF IN DONOR TABLE:

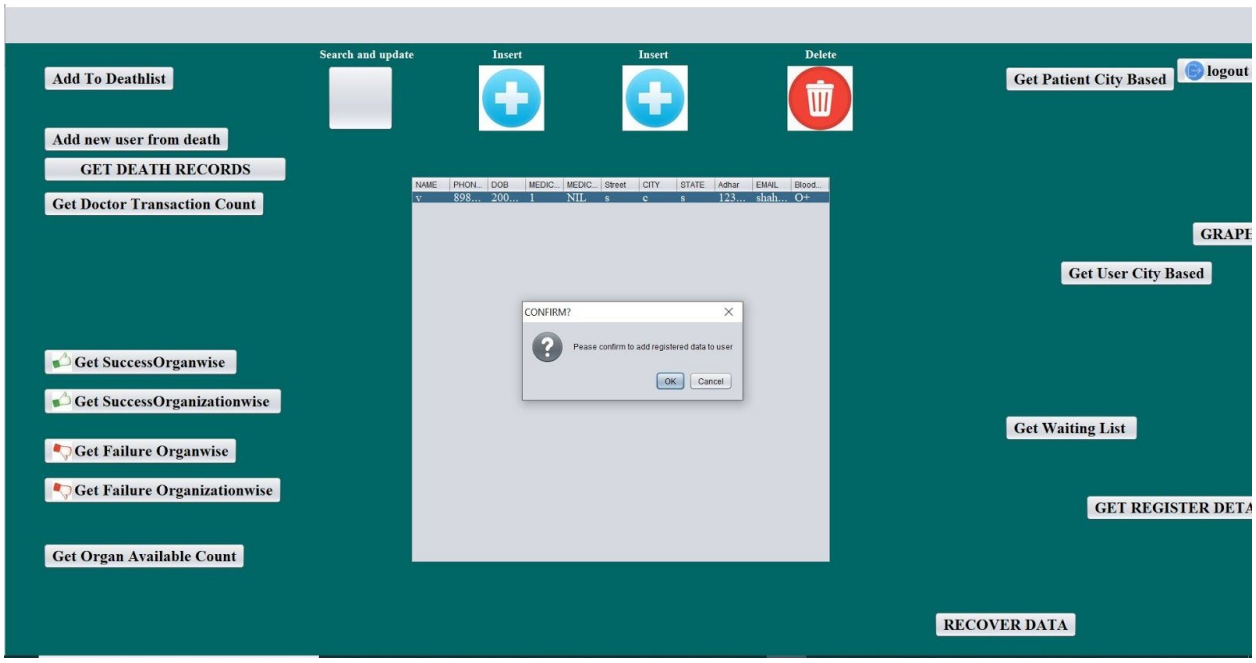
DONOR_ID	ORGAN_DONATED	REASON_OF_DONATION	ORGANIZATION_ID	USER_ID
201	Heart	NIL	20	100

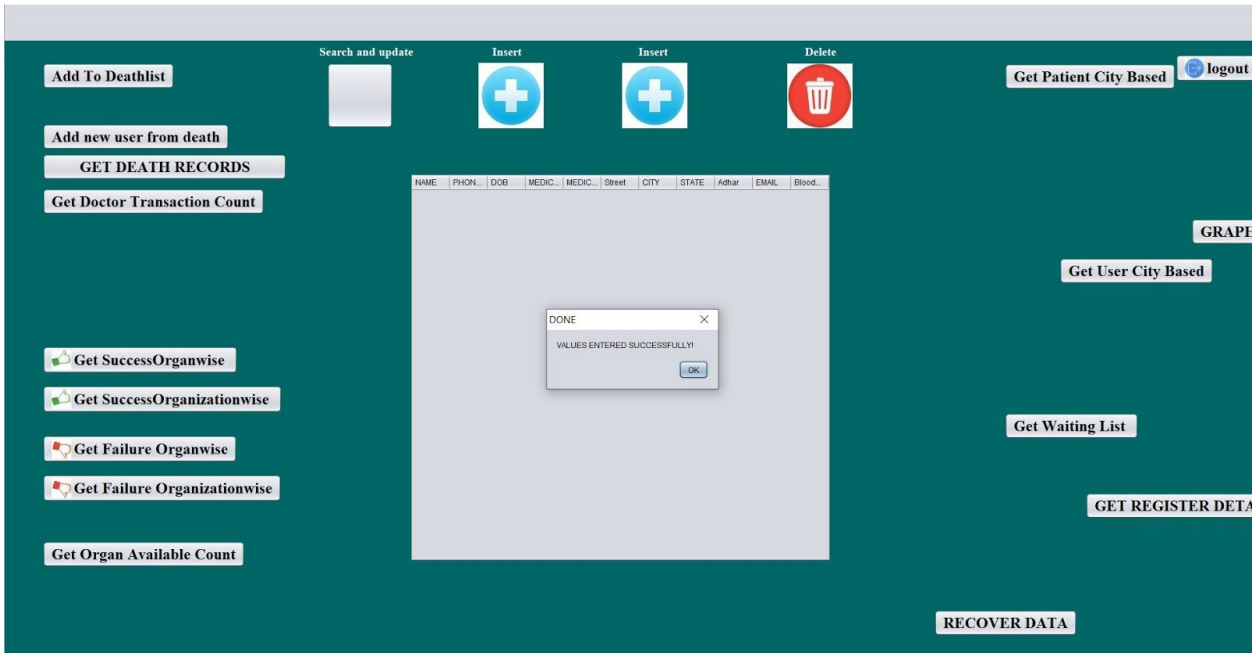
1 rows returned in 0.01 seconds Download

ADD FROM REGISTER TABLE TO DEATH RECORDS:

First of all, we store the data of the person into the register table who wish to donate his/her organs in future after his/her death. Then after his/her death, the data is to be stored in death_records table. First click on get register table button.

In that, it will show all the data of the registered users.Now, select the data which you want to transfer into death record and delete it from the register table. Example: If a person A wishes to donate any organs, so he need to register himself and his data gets stored in the register table. Now, after his death, we add his/her data to the death records table using this procedure and then approach his/her family to ask for permission of the organ donation.Hence, if permission is granted, then his/her record is stored to donor table.





Proof of death_record table(from register table):

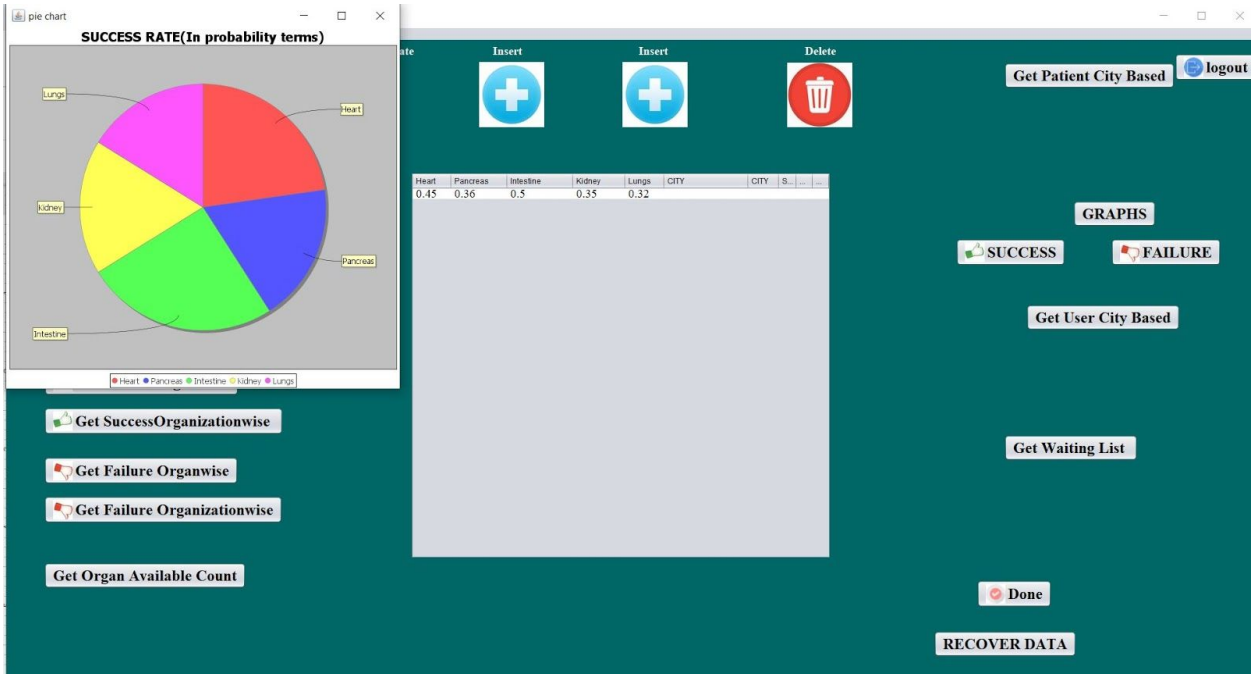
DEATH_ID	NAME	REASON_OF_DEATH	MEDICAL_HISTORY	MEDICAL_INSURANCE	CITY	STREET	STATE	DATE_OF_BIRTH	DONATE_STATUS	PERSPECTIVE
2	Varshil	NIL	NIL	0	s	d	d	03/02/2001	0	Not approched
4	v	NIL	NIL	1	c	s	s	03/02/2001	0	Not approched
3	v	NIL	NIL	1	c	s	s	02/03/2001	0	Not approched
1	v	v	v	1	v	v	v	03/02/2001	1	approched-No

4 rows returned in 0.00 seconds Download

FUNCTION

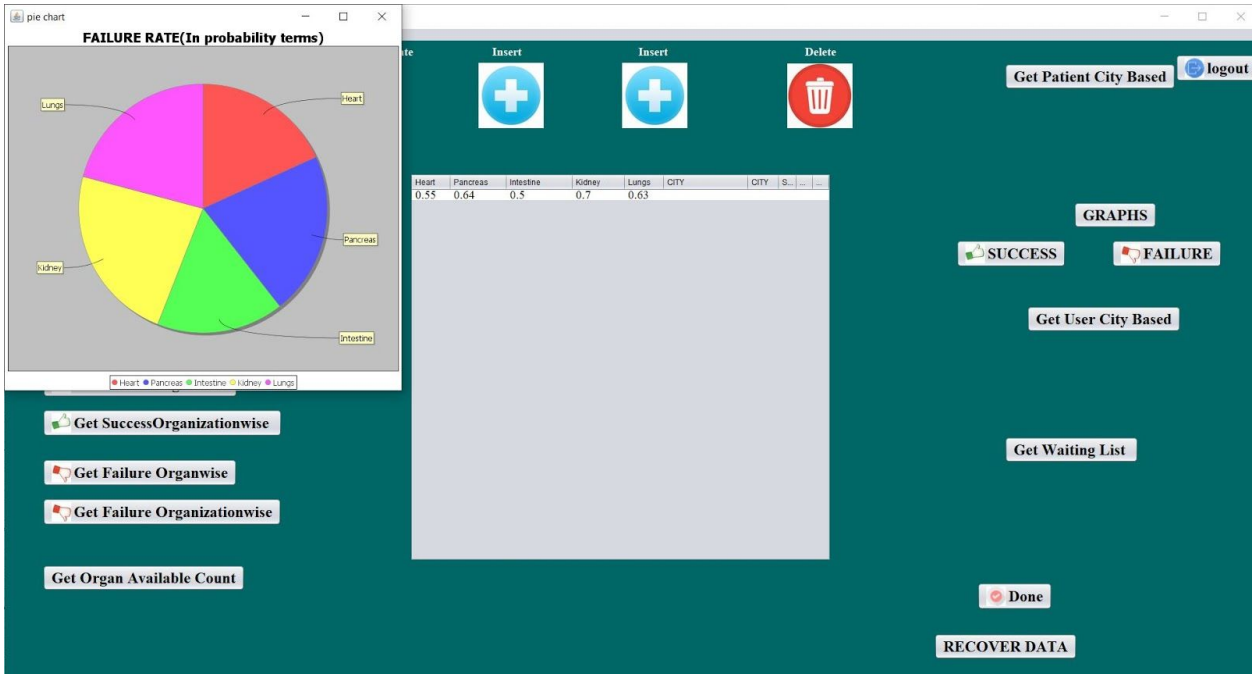
SUCCESS GRAPH:

Here, we make a function which helps us to plot the graph of the probability of the database being successful for the operations organwise in dynamic form i.e. every insertion,updation or deletion of the data will be shown in the graph with due changes.

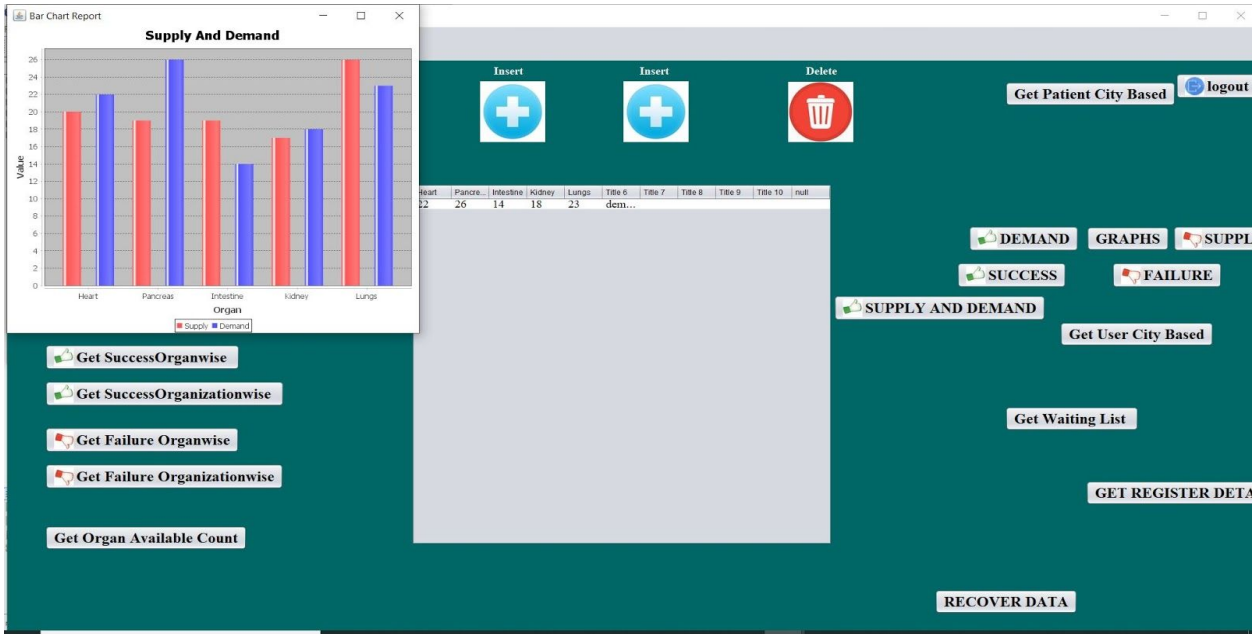


FAILURE GRAPH:

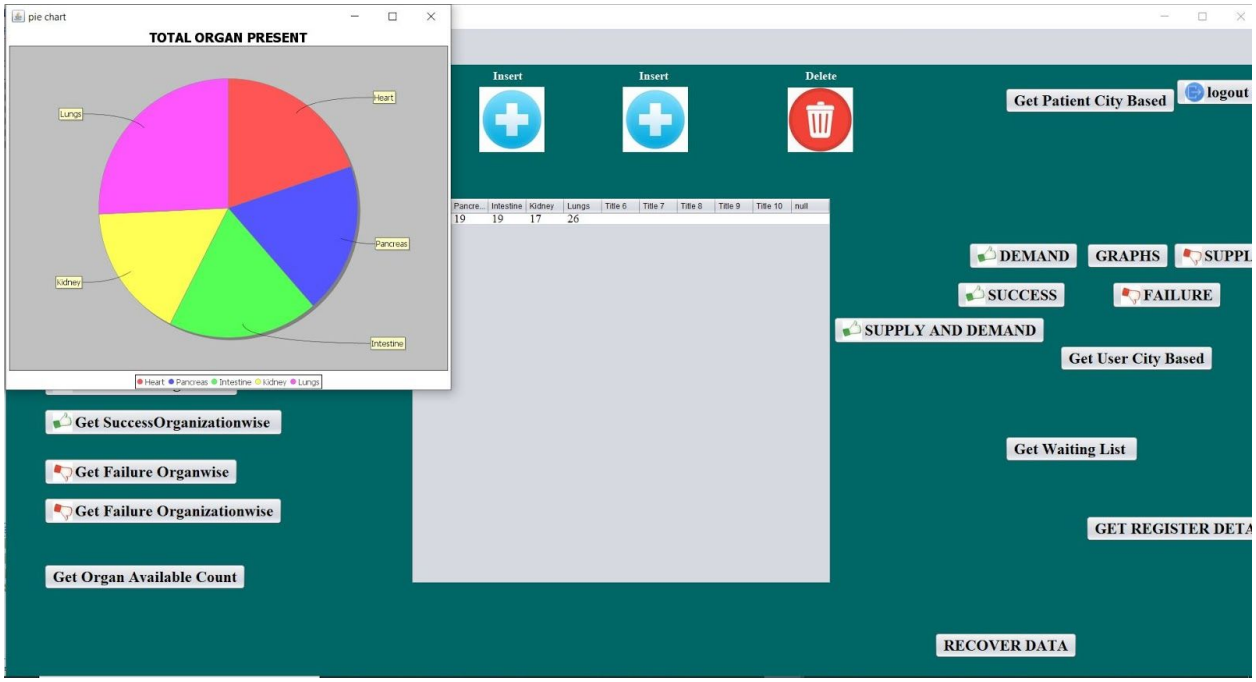
Similarly, we also make a function which helps us to plot the graph of the probability of the database being failure for the operations organwise in dynamic form i.e. every insertion,updation or deletion of the data will be shown in the graph with due changes.



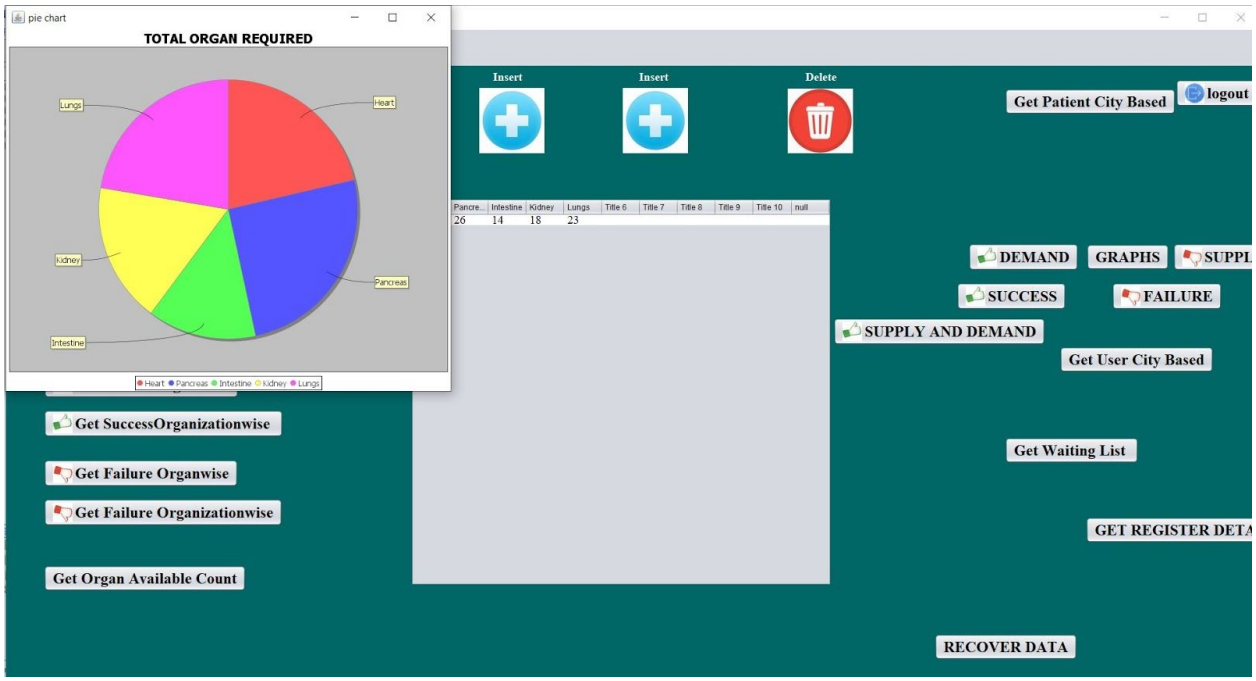
SUPPLY AND DEMAND GRAPH:-The following graph below shows the supply and demand of different organ based on dynamic database of the organization. Red bar shows amount of supply and Blue bar shows amount of demand of organs.



SUPPLY:- The following graph below shows the supply of organs in the form of pie graph visual representation.



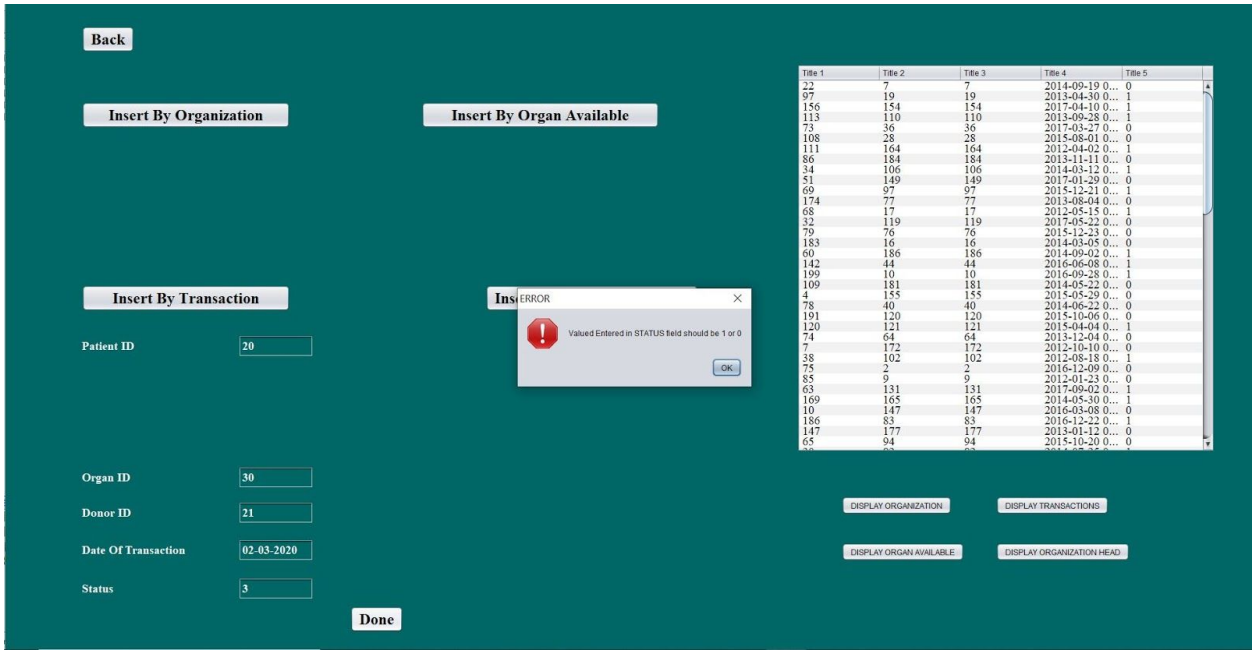
DEMAND:-The following graph below shows the demand of organs in the form of pie graph visual representation.



SCREENSHOTS OF ERRORS GENERATED ON FRONT-END WHEN TRIGGER IS FIRED

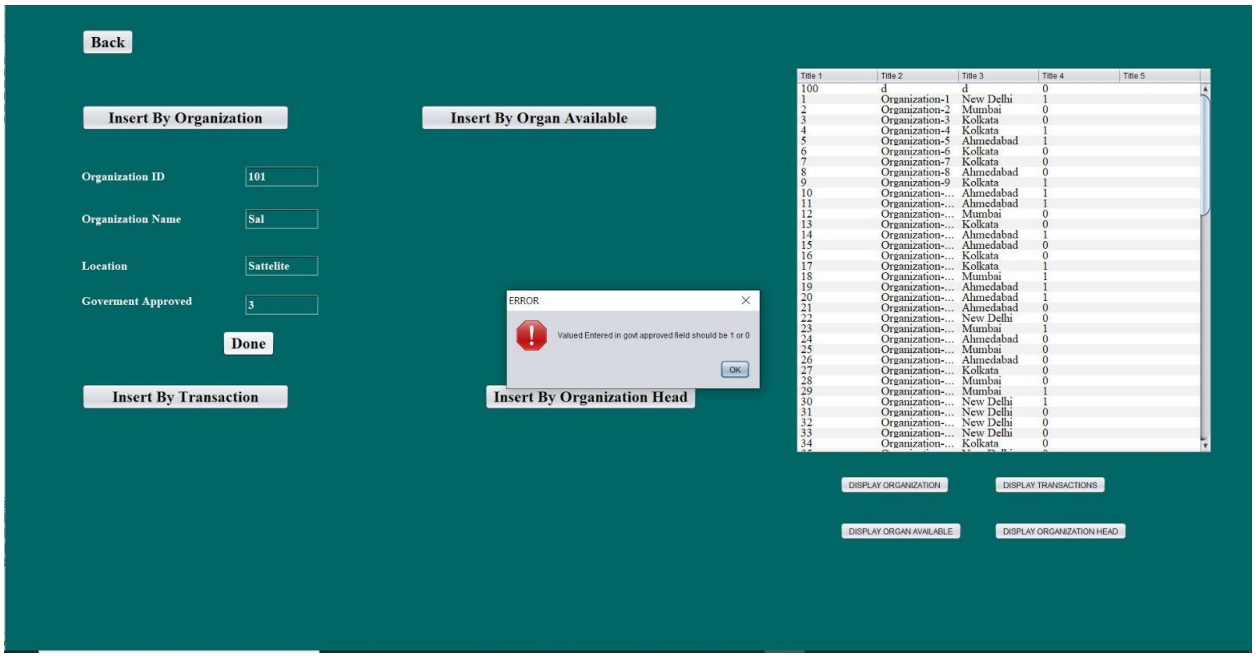
INSERT INTO TRANSACTION STATUS TRIGGER:

When you will enter the value of status in transaction other than 0 or 1 , the trigger will raise the error. (Status:1=Transaction Successful Status,0=Transaction Unsuccessful)



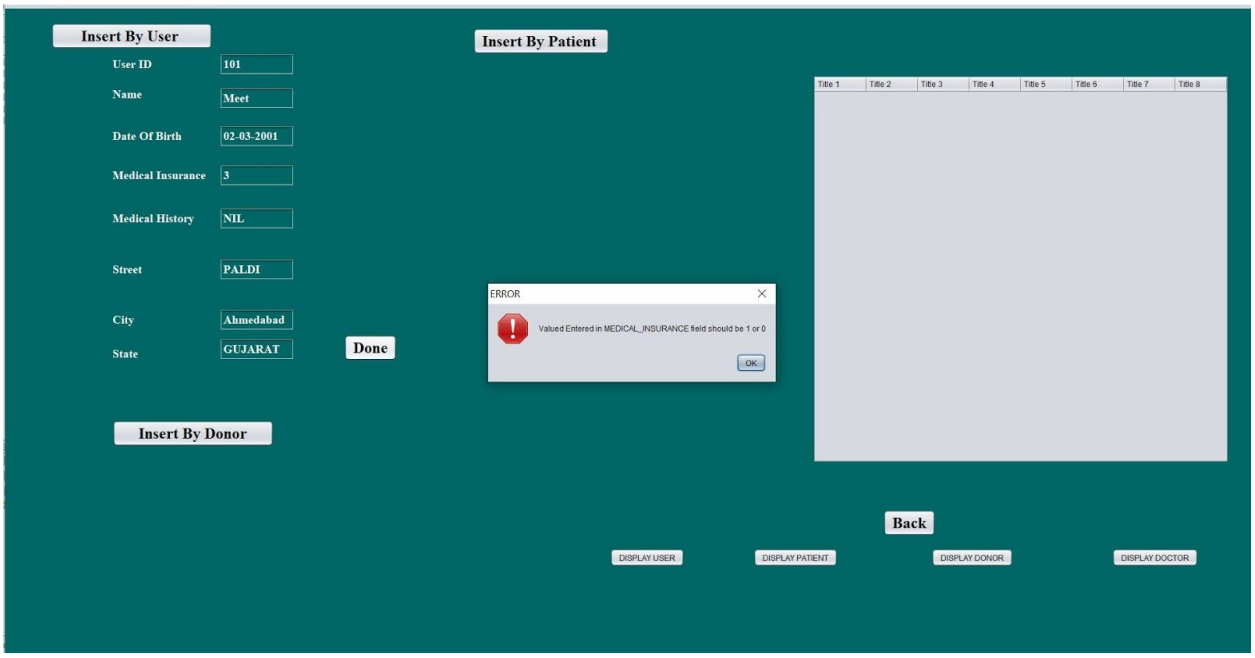
INSERT INTO ORGANIZATION GOVERNMENT APPROVED TRIGGER:

When you enter the value of a government approved field in organization table other than 0 or 1 , the trigger will raise the error. (Government Approved:1=government approved : 0=not government approved)



INSERT USER MEDICAL INSURANCE TRIGGER ERROR:

When you enter the value of a medical insurance field in user table other than 0 or 1 , the trigger will raise the error. (Medical Insurance:1=user is insured : 0=user is not insured)

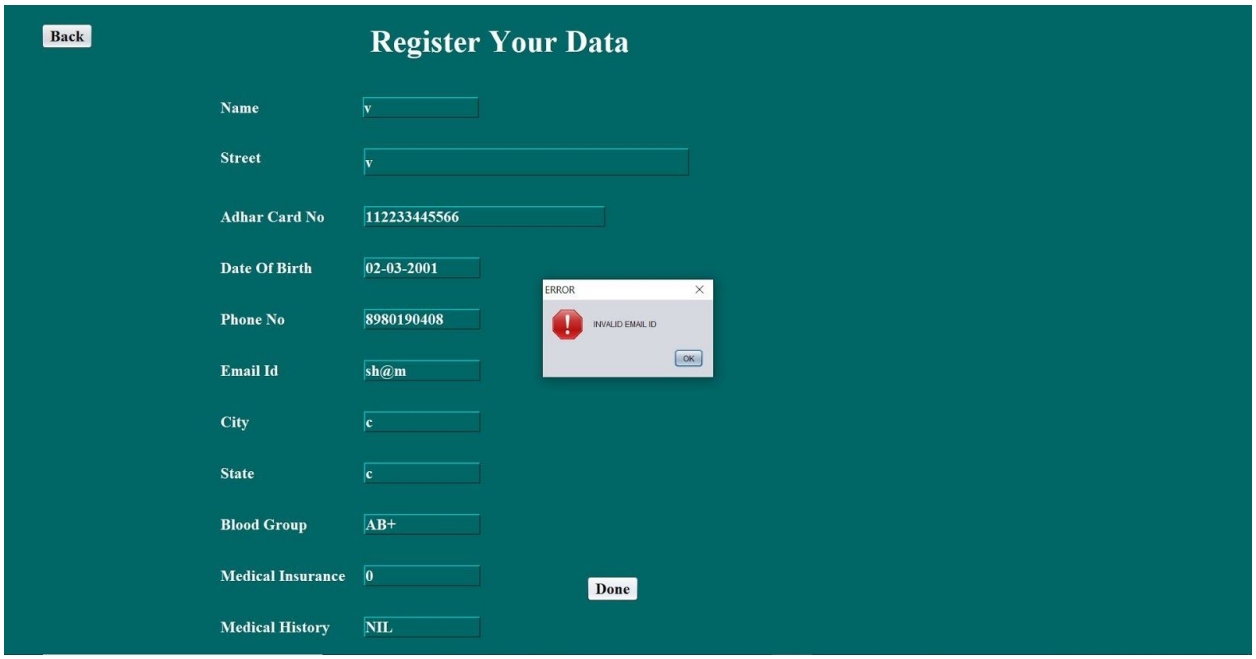


INSERT USER PHONE NUMBER TRIGGER ERROR:

When you enter the value of a Phone Number field in user table other than 10 numbers or with numbers not starting from (6-9), the trigger will raise the error. (ERROR IN Value Entered in Phone Number)

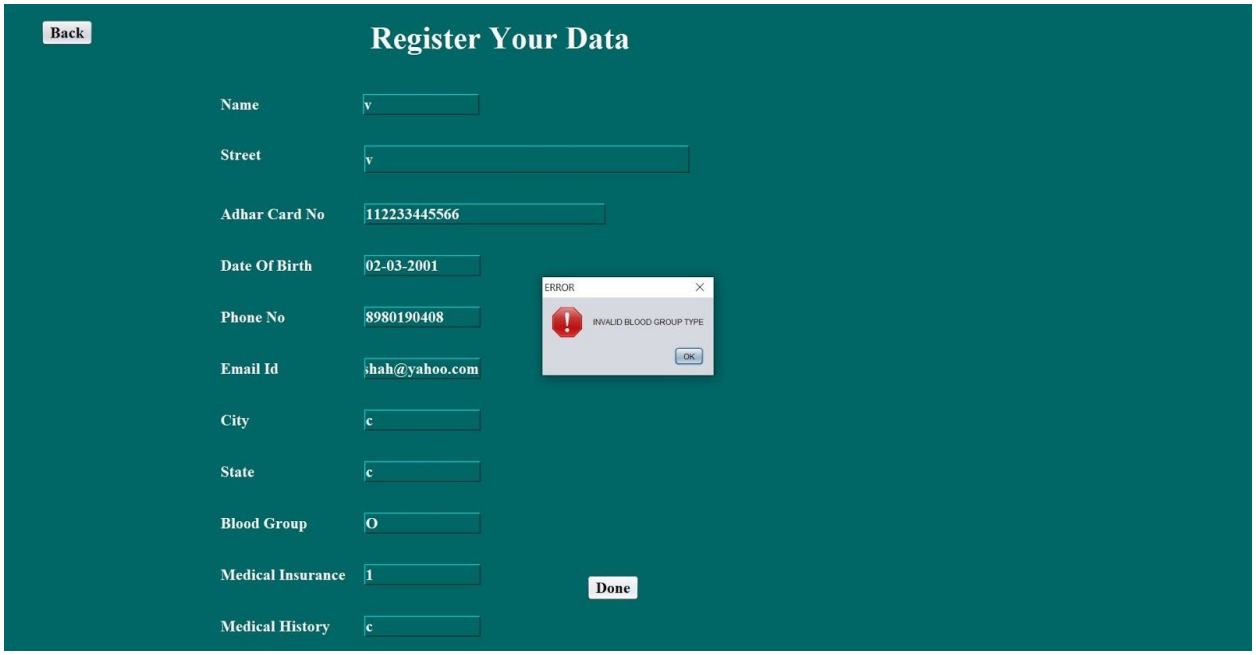
INSERT USER Email Id TRIGGER ERROR:

When you enter the value of a Email Id field in register table with format other than this ‘abc@g.xyz’, the trigger will raise the error. (ERROR IN Value Entered in email)



INSERT USER BLOOD GROUP TRIGGER ERROR:

When you enter the value of a Blood Group field in user table other than AB+,B+,O+,A+,AB-,B-,A-,,O-, the trigger will raise the error. (ERROR IN Value Entered in BLOOD GROUP)



NEW FUNCTIONALITIES WE HAVE ADDED:-

1) **Created Recovery Table which is used to recover the delete data from the database. The user can recover this data from this table which just works like a recycle bin for the computer or like recently deleted item in the mobile phone. Through this, Data Integrity can be maintained and data loss prevention can be done for a long period of time in the future which is actually fetched or used from the past.**

2) **Created Log Table which stores the manipulation of the user or the steps which user has done like insert,add,update,etc. in the backend with respect to the tables just like storing the history of what the user has done**

3) **We do validations for authenticating and transparency of this system for different occasions like checking the authentication of government approved organizations for organ donation, checking user and doctor's validity through their phone number, checking approved medical insurance availability and transaction success through these triggers.**

4) **Providing search and update options based on name,id,city,state,organs,etc.**

5) **We do store and display the total transaction of different surgeries either successful or unsuccessful of doctors as well as a particular organ.**

6) **The front end part is made user friendly so as to effectively use even through basic understanding using minimal screens including all buttons at the same place.**

7) **Added the functionality of death of user by notifying it to the organizations so that they can be approached and asked to donate organs. This is because most of the people have very less idea of donation of organs and no body gets any idea at the instant of time when the death has occured. So, approaching them directly will help the organizations and hospitals to collect more organs and help others. If the approached user agrees for donation, they are marked as 1 in the backend and if they are approached and they disagree for donation, they are marked as 0.**

8) **Made functions which returns the statistics about the whole database in form of the graph in jFrame for proper visual representation to the user. Also, the graph of the database is dynamic i.e. insertion,deletion and updation of data is shown through the graph everytime.**