CO527: Advanced Database System

Organ Transplantation Management System

Made By:

E/14/010
E/14/028
E/14/065

Title: Organ Transplantation Management System

Problem Statement:

Organ transplantation is a medical procedure in which an organ is removed from one body and placed in the body of a recipient, to replace a damaged or missing organ. The donor and recipient may be at the same location, or organs may be transported from a donor site to another location.

Organ Donation and Procurement Organizations play a pivotal role in today's medical institutions. Such organizations are responsible for the evaluation and procurement of organs for organ transplantation. These organizations represent the front-line of organ procurement, having direct contact with the hospital and the family of a recently deceased donor. The work of such organizations includes to identify the best candidates for the available organs and to coordinate with the medical institutions to decide on each organ recipient. They are also responsible for educating the public to increase the awareness of and participation in the organ donation process. Also, it keeps track of all transplantation operations carried till date.

The Organ Donation and Procurement Network Management System is a database management system that uses database technology to construct, 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, etc of every person in the database design. In short, it maintains a database containing statistical information regarding network of organ donation and procurement of different countries.

Organ Wastage is a major issue that can only be solved by having a proper database of all Patient and Donors in a well-formed way, that can be processed easily.

Records of donor and patients are created when a person donates or procures an organ from a Medical Institution. Records may include the following information: -

1. Personal Information
2. Medical History
3. Medical insurance, if any
4. Allergies to any medicine, if any
5. The need for an organ presently
6. Medical Insurance provided by any private or government insurers.
7. Address

This record serves a variety of purposes and is critical to the proper functioning of Organ Donation and Procurement Network, especially in today's complicated health care environment. These records provide statistical information regarding the number of organs needed and available at a particular point of time. It is essential for planning, evaluating and coordinating organ donation and procurement.
Our 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.

Basic Steps in Implementation:

- Every user has an account with can only be registered by a government certified hospital, which will keep all the information as defined in Problem Statement.
- Only Hospitals are eligible to request for 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:

- MYSQL
- HTML
- CSS
- Python
- Flask

ER Analysis: Identifying Entity Sets and Relationship Sets:

Entity Sets:

1. **User**
    1. User ID
    2. Name
    3. Date of birth
    4. Phone Number (multi-valued)
    5. Medical Insurance
    6. Medical History
    7. Address
2. **Patient**
    1. Patient_ID
    2. Organ Required
    3. Reason of procurement
    4. User_ID (foreign key)
3. **Donor**
    1. Donor_ID
    2. Organ Donated
    3. Reason of donation
    4. User_ID (foreign key)
4. **Organ Available**
    1. Organ_ID
    2. Organ Name
    3. Donor_ID (foreign key)
5. **Organization**
    1. Organization ID
    2. Organization Name
    3. Location
    4. Government approved organization or not
    5. Phone Number (multi-valued)
6. **Doctor**

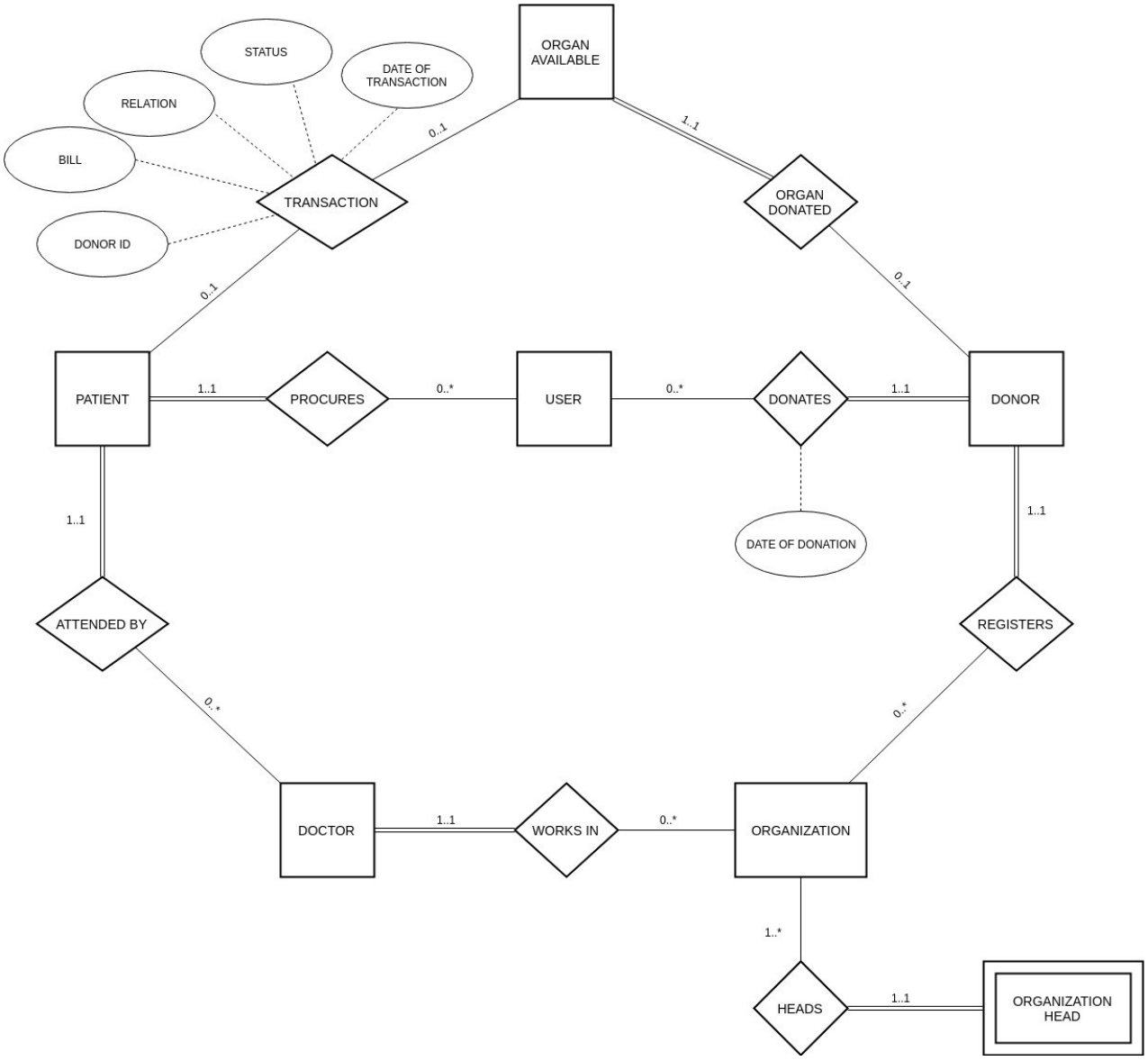
1. Doctor ID
2. Doctor Name
3. Phone Number (multi-valued)

7. **Organization Head**
    1. Head Name
    2. Date of Joining
    3. Term Length

Relationship Sets:

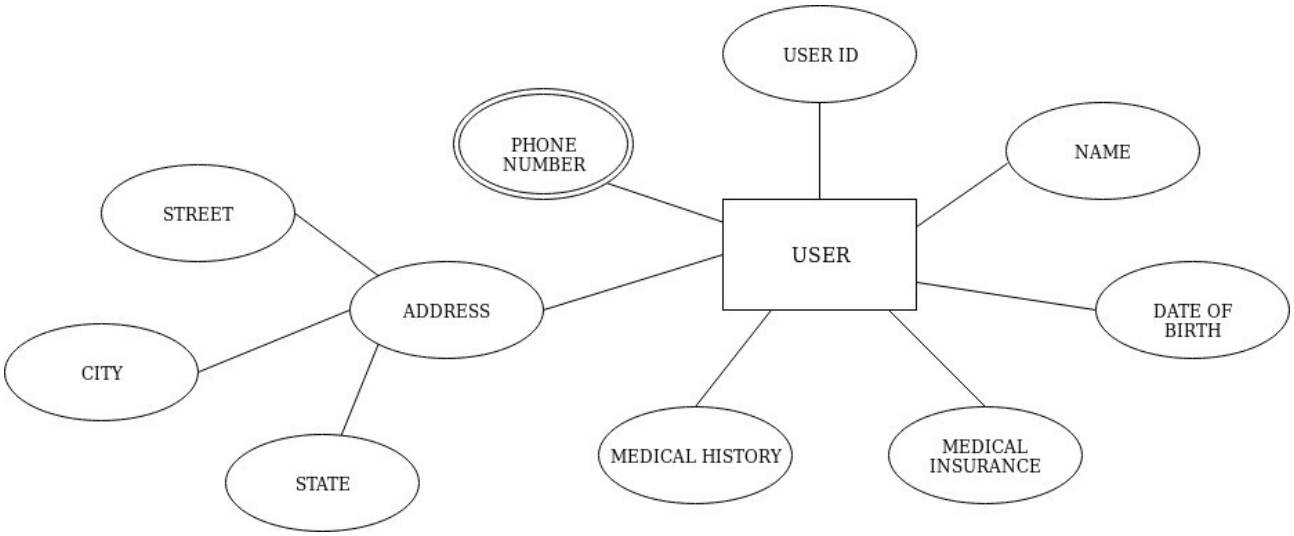
1. **Donates –** The act of donation of an organ from a donor
    1. Date – Date of donation
2. **Procures -** The act of procuring an organ by the patient
3. **Transaction**
    1. Date of transaction
    2. Status – whether the surgery was successful or not
4. **Organ Donated -**The organ donated by an 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 works.
8. **Headed By –** The organization is headed by which person

ER DIAGRAM

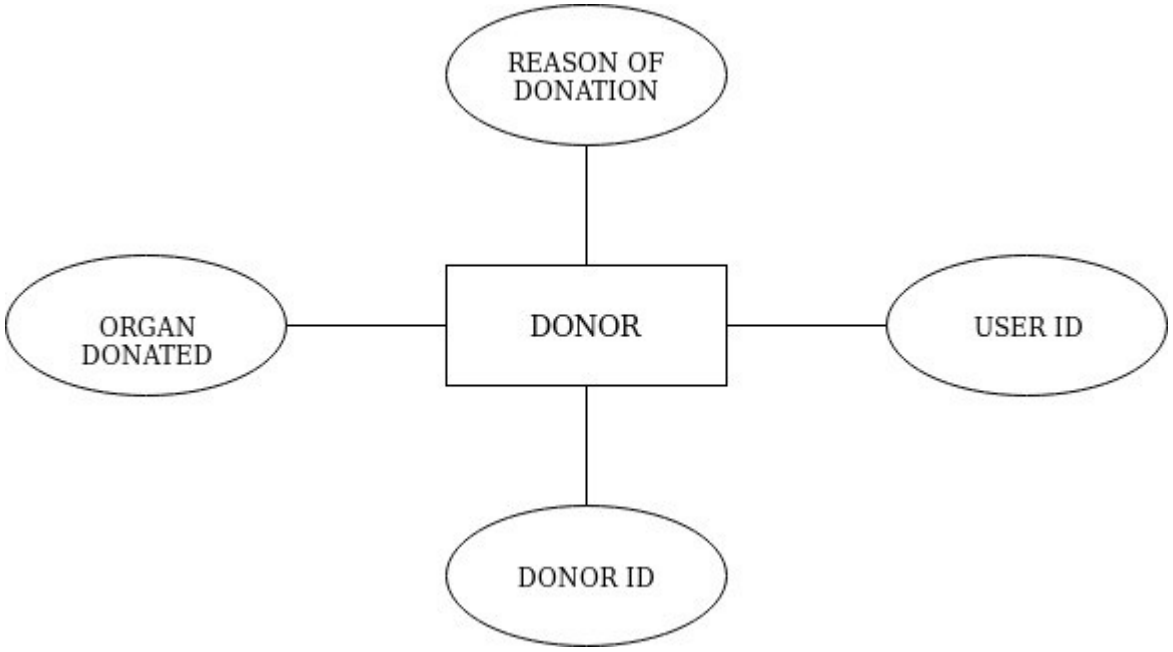

STATUS
DATE OF TRANSACTION
RELATION
BILL
DONOR ID
TRANSACTION
ORGAN AVAILABLE
ORGAN DONATED
0..1
1..1
0..1
0..1
PATIENT
1..1
PROCURES
0..*
USER
0..*
DONATES
1..1
DONOR
DATE OF DONATION
1..1
1..1
ATTENDED BY
REGISTERS
0..*
0..*
DOCTOR
1..1
WORKS IN
0..*
ORGANIZATION
1..*
HEADS
1..1
ORGANIZATION HEAD

Entity Sets

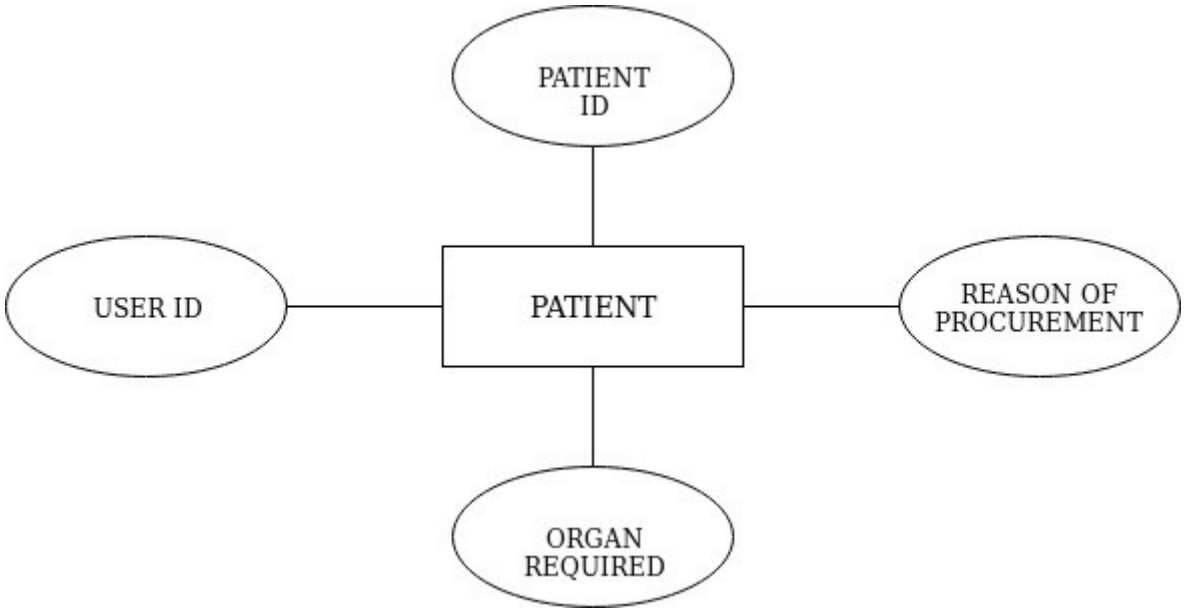
1) User -



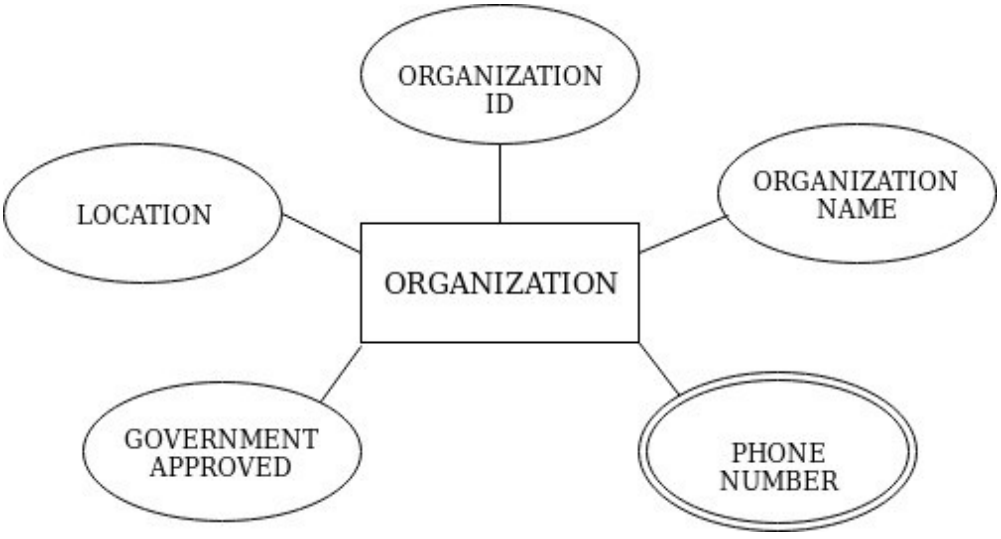
2) Donor



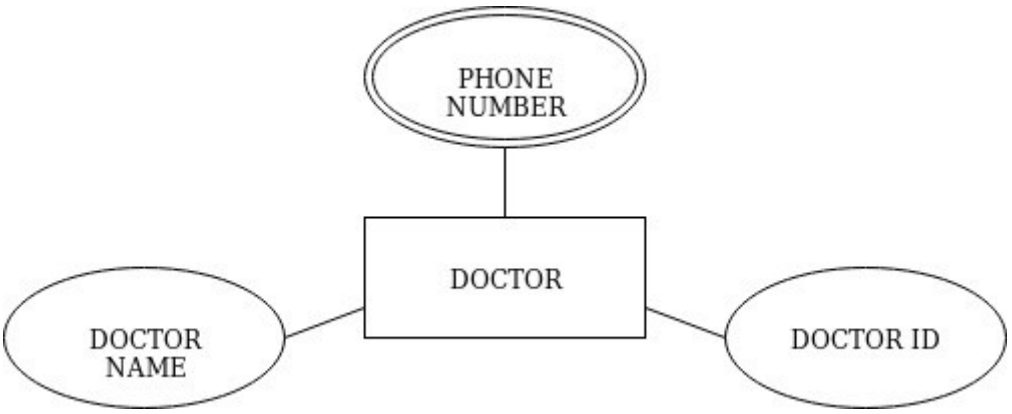
3) Patient



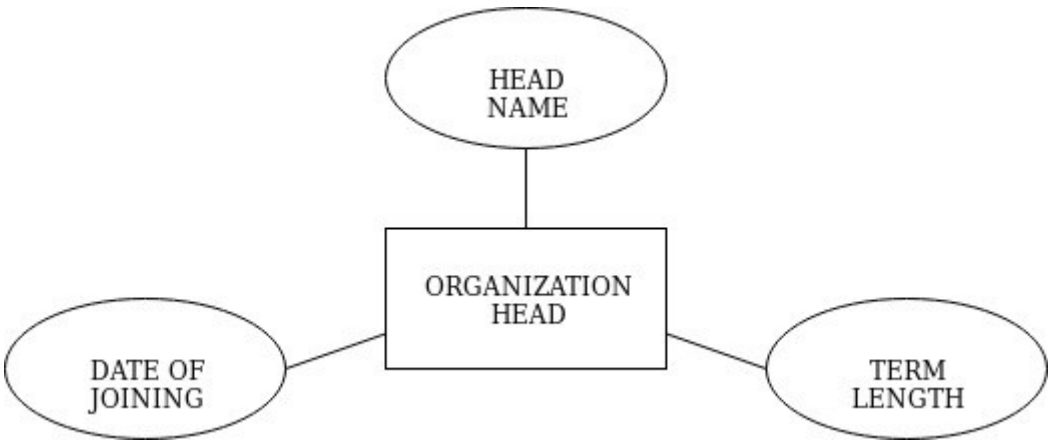
4) Organization



6) Doctor



7) Organization Head



Tables and their Functional Dependencies :-

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

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**}**

Triggers

The following triggers are added to create a log of actions done on database. The logs are added to the log table.

1) Trigger for adding Donor information to Log table.

```
delimiter //
create trigger ADD_DONOR_LOG after
insert
on Donor
for each row
begin
insert into log values
(now(), concat("Inserted new Donor",
cast(new.Donor_Id as char)));
end // delimiter
;
```

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
(now(), concat("Updated Donor Details",
cast(new.Donor_Id as char)));
end // delimiter
;
```

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
(now(), concat("Deleted Donor ",
cast(old.Donor_Id as char))); end //
delimiter ;
```

4) Trigger 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
(now(), concat("Inserted new Patient ",
cast(new.Patient_Id as char))); end //
delimiter ;
```

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
(now(), concat("Updated Patient Details ",
cast(new.Patient_Id as char)));
end // delimiter
;
```

6) Trigger for adding “Delete information” action information in Log table

```
create trigger DEL_PATIENT_LOG after
delete
on Donor
for each row
begin
insert into log values
(now(), concat("Deleted Patient ",
cast(old.Donor_Id as char)));
end // delimiter
;
```

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
(now(), concat("Added Transaction :: Patient ID : ",
cast(new.Patient_ID as char), "; Donor ID :
" ,cast(new.Donor_ID as char))); end //
delimiter ;
```

Transactions

1) Whenever a donor is added to the Donor Table, a corresponding organ must be added to the Organ_available table. So the two insert commands must be atomic. We have created the following transcation for this purpose

```
-- 1. start a new transaction START
TRANSACTION;

-- 2. insert into Donor table
INSERT INTO Donor values ( _ , _ , _ , _ , _ );

-- 3. insert into Organ_available table INSERT INTO
Organ_available ( _ , _ );

-- 4. commit changes
COMMIT;
```

2) Whenever a transaction takes place, the record corresponding to that Organ_ID must be deleted from Organ_available table. So the insert and delete commands must be atomic. We have created the following transaction for this purpose.

```
-- 1. start a new transaction START
TRANSACTION;

-- 2. insert into Donor table
INSERT INTO Transaction values ( _ , _ , _ , _ ,
_ );

-- 3. delete from Organ_available table
DELETE FROM Organ_available where Organ_ID = _;

-- 4. commit changes
COMMIT;
```

Procedure to run

Procedure to run on your computer:

The Project Uses:

1. MySQL
2. HTML 5
3. Python
4. Flask Framework
5. CSS
6. Bootstrap
7. JavaScript
8. Ajax

Steps to run:

Step 1. Making the database:
Import **otms_create_database.sql** to create the database & tables.
Then go to the database and import **otms_insert_data.sql** to it.

Step 2. Make sure do the changes (password, port…) in config.py to your MySQL.

Step 3. Run main.py.

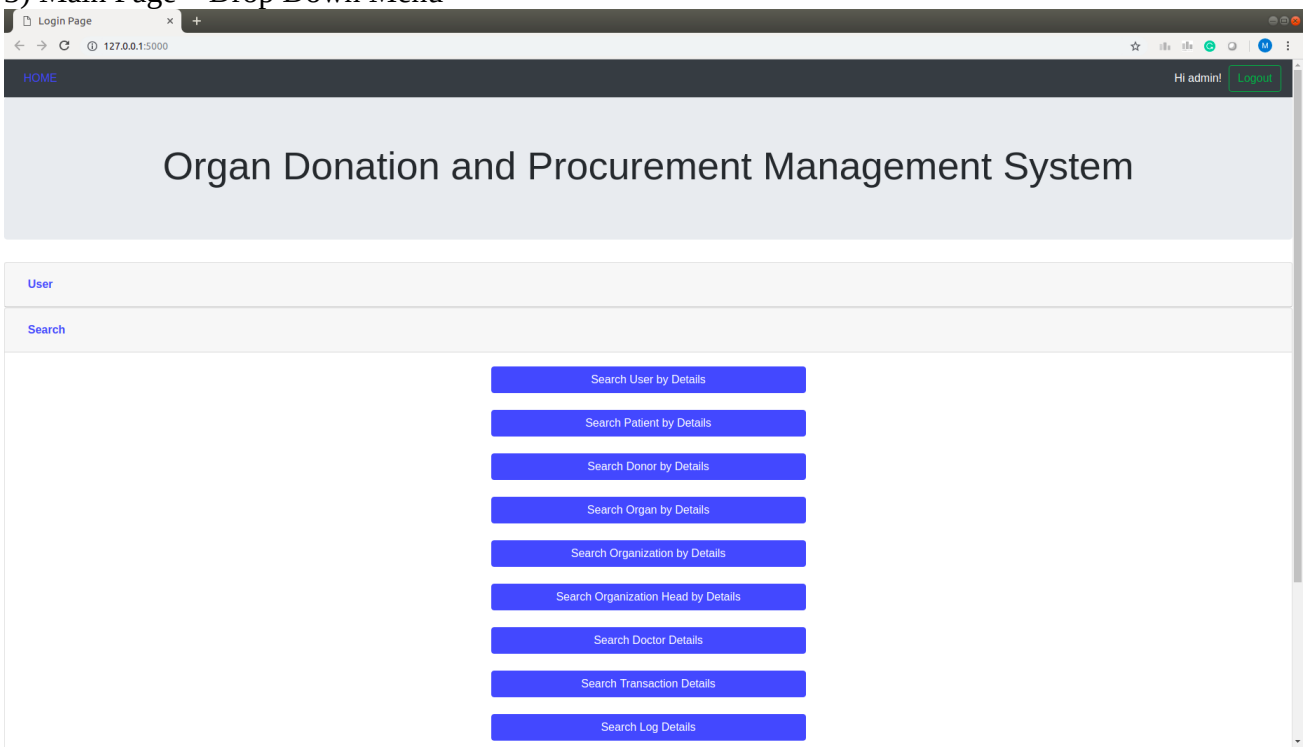
Step 4. Go to localhost:/5000 on browser.

Screenshots

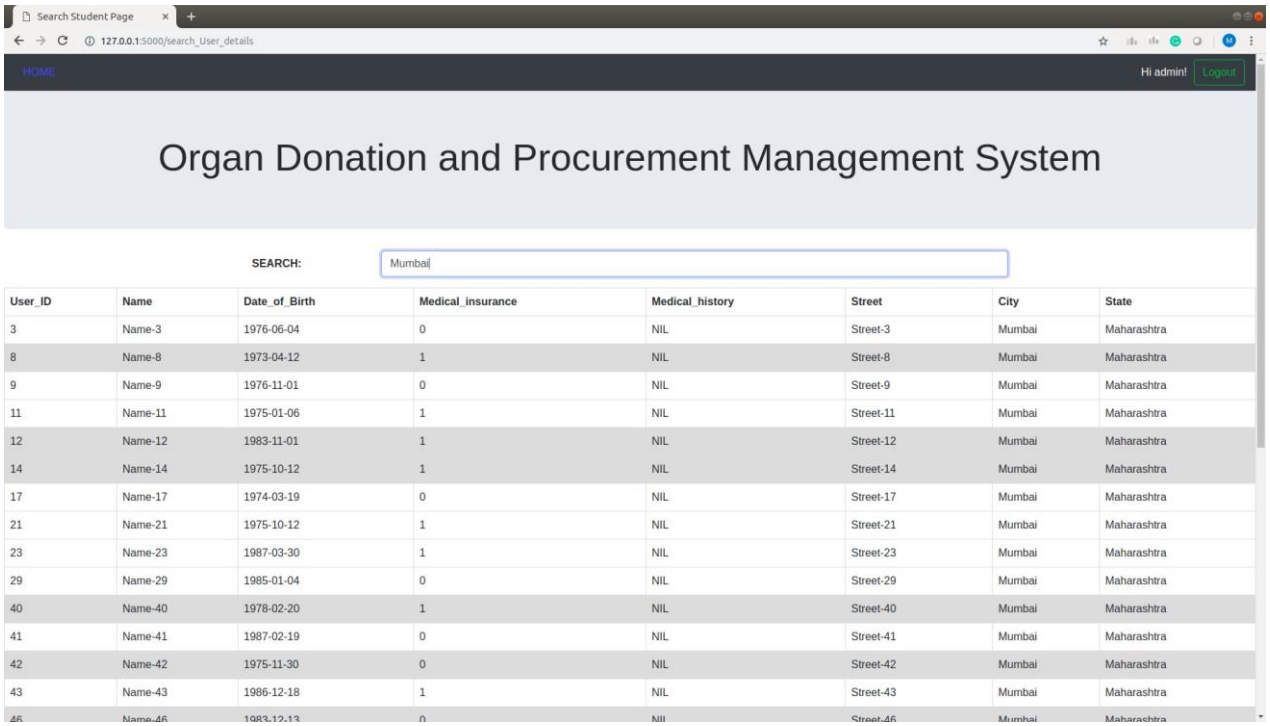
1) Login Page

2) Main Page – GUI

3) Main Page – Drop Down Menu



4) Searching Option



User_ID	Name	Date_of_Birth	Medical_insurance	Medical_history	Street	City	State
3	Name-3	1976-06-04	0	NIL	Street-3	Mumbai	Maharashtra
8	Name-8	1973-04-12	1	NIL	Street-8	Mumbai	Maharashtra
9	Name-9	1976-11-01	0	NIL	Street-9	Mumbai	Maharashtra
11	Name-11	1975-01-06	1	NIL	Street-11	Mumbai	Maharashtra
12	Name-12	1983-11-01	1	NIL	Street-12	Mumbai	Maharashtra
14	Name-14	1975-10-12	1	NIL	Street-14	Mumbai	Maharashtra
17	Name-17	1974-03-19	0	NIL	Street-17	Mumbai	Maharashtra
21	Name-21	1975-10-12	1	NIL	Street-21	Mumbai	Maharashtra
23	Name-23	1987-03-30	1	NIL	Street-23	Mumbai	Maharashtra
29	Name-29	1985-01-04	0	NIL	Street-29	Mumbai	Maharashtra
40	Name-40	1978-02-20	1	NIL	Street-40	Mumbai	Maharashtra
41	Name-41	1987-02-19	0	NIL	Street-41	Mumbai	Maharashtra
42	Name-42	1975-11-30	0	NIL	Street-42	Mumbai	Maharashtra
43	Name-43	1986-12-18	1	NIL	Street-43	Mumbai	Maharashtra
46	Name-46	1983-12-13	0	NIL	Street-46	Mumbai	Maharashtra

6) Data visulaization using matplotlib in Python.