VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnanasangama", Belagavi-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY
K.R. Road, V.V.Puram, Bangalore-560 004

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DATABASE MANAGEMENT SYSTEM LAB WITH MINI PROJECT

18CSL58

"Organ Donation and Procurement Management System"

Submitted By

ASHISH KUMAR SHUKLA
1BI20CS033

for the academic year 2022-23

Department of Computer Science & Engineering
Bangalore Institute of Technology
K.R. Road, V.V.Puram, Bangalore-560 004

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnanasangama", Belagavi-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY
K.R. Road, V.V.Puram, Bangalore-560 004

Department of Computer Science & Engineering

Certificate

This is to certify that the implementation of **DBMS MINI PROJECT** entitled "**Organ Donation and Procurement Management System**" has been successfully completed by **USN: 1BI20CS033 NAME :ASHISH KUMAR SHUKLA** of V semester B.E. for the partial fulfillment of the requirements for the Bachelor's degree in Computer Science & Engineering of the Visvesvaraya Technological University during the academic year 2022-2023.

Lab Incharge Faculty :

T.P. Mansa
Asst. Professor
Dept. of CS&E
Bangalore Institute of Technology
Bangalore -04

Dr. Girija J
Professor and Head
Department of CS&E
Bangalore Institute of Technology
Bangalore-04

Examiners: 1) 2)

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance crowned our efforts with success.

I consider myself proud to be part of the Bangalore Institute of Technology family, the institution which stood by our way in endeavors.

I am grateful to Dr. Girija J, Professor and HOD, Dept of CSE who is a source of inspiration and of invaluable help in channelizing my efforts in the right direction.

I wish to thank my internal guide Mrs. T.P Mansa, Asst. Professor, Dept of CSE for guiding and correcting various documents of mine with attention and care. She has taken a lot of pain to go through the document and make necessary corrections as and when needed.

I would like to thank the faculty members and supporting staff of the Department of CSE, BIT for providing all the support for completing the Project work.

Finally, I am grateful to my parents and friends for their unconditional support and help during the course of my Project work

ASHISH KUMAR SHUKLA
1BI20CS033

TABLE OF CONTENTS

TABLE OF FIGURES

INTRODUCTION

1.1 Overview

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 networks of organ donation and procurement of different countries.

1.2 Problem Statement

Our aim is to create a solution that effectively deals with the problems of finding donors and also providing data of the transplants that can help the government to form better rules and regulations.

Records of donors and patients are created when a person donates or procures an organ from a Medical Institution.

1.3 **Objectives**

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

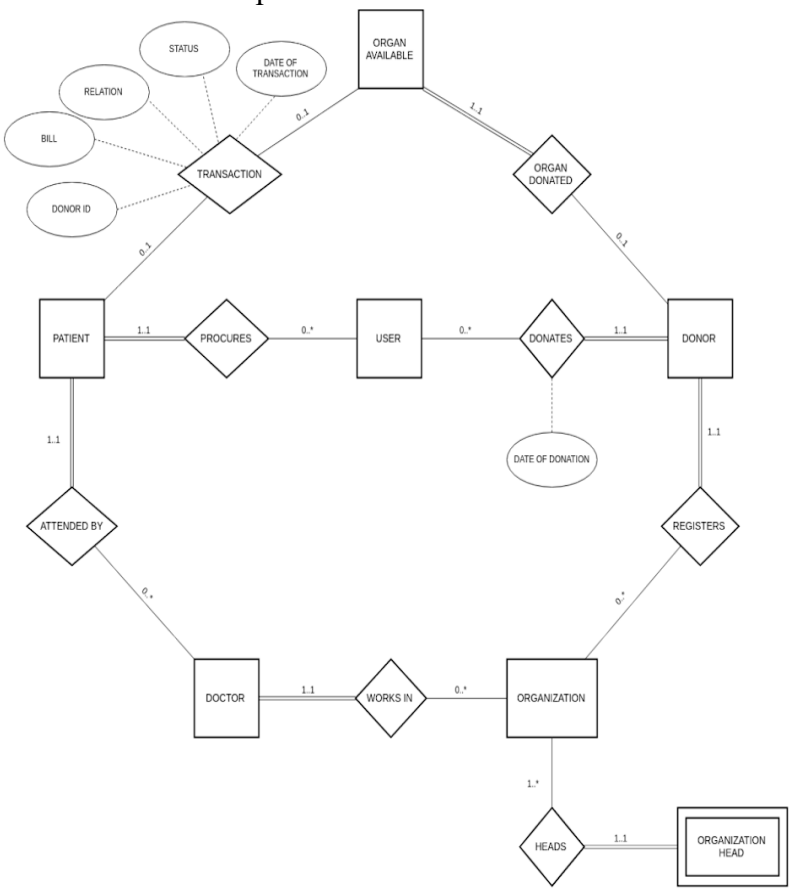
- Records of donors and patients are created when a person donates or procures an organ from a Medical Institution. Records may include the following information:-
  1. Personal Information
  2. Medical History
  3. Medical insurance, if any
  4. Allergies to any medicine, if any
  5. The need for an organ presently
  6. Medical Insurance provided by any private or government insurers.
  7. Address

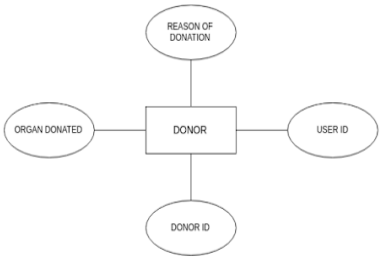
- This record serves a variety of purposes and is critical to the proper functioning of Organ Donation and Procurement Network, especially in today's complicated health care environment. These records provide statistical information regarding the number of organs needed and available at a particular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

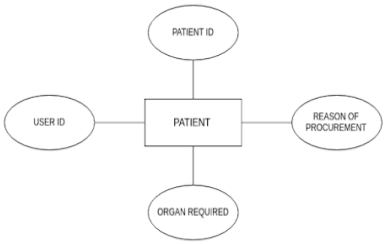
BACK END DESIGN

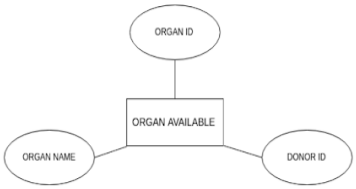
2.1 Conceptual Database Design

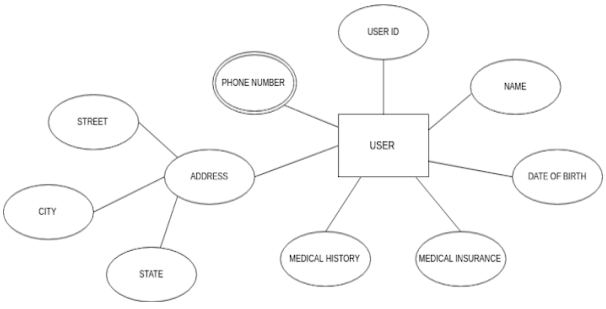
An **Entity–relationship model (ER model)** describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of the E-R model are: entity set and relationship set.

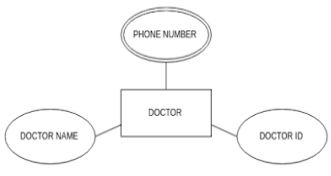


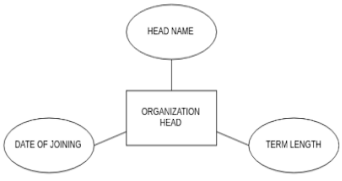












2.2 Logical Database Design

A schema diagram can display only some aspects of a schema like the name of record type, data type, and constraints. Other aspects can't be specified through the schema diagram.

ER SCHEMA DIAGRAM FOR ORGAN DONATION AND PROCUREMENT PROJECT

User

User_ID	Name	Date_of_Birth	Medical_insurance	Medical_history	Street	City	State

User_phone_no

User_ID	phone_no

Organization

Organization_ID	Organization_name	Location	Government_approved

Doctor

Doctor_ID	Doctor_Name	Department_Name	organization_ID

Patient

Patient_ID	organ_req	reason_of_procurement	Doctor_ID	User_ID

Donor

Donor_ID	organ_donated	reason_of_donation	Organization_ID	User_ID

Organ_available

Organ_ID	Organ_name	Donor_ID

Transaction

Patient_ID	Organ_ID	Donor_ID	Date_of_transaction	Status

Organization_phone_no

Organization_ID	Phone_no

Doctor_phone_no

Doctor_ID	Phone_no

Organization_head

Organization_ID	Employee_ID	Name	Date_of_joining	Term_length

2.3 Normalization

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form by removing duplicated data from the relation tables.

Normalization is used for mainly two purposes,
• Eliminating redundant(useless) data.
• Ensuring data dependencies make sense i.e., data is logically stored.

2.3.1 First Normal Form (1NF):

As per First Normal Form
a) There are no duplicate rows in the table.
b) Each cell is single valued or atomic.

2.3.2 Second Normal Form (2NF):

As per Second Normal Form, a table is in 2NF if every non-prime attribute is not partially dependent on any key of the table.

2.3.3 Third Normal Form (3NF):

Third Normal Form applies that every non-prime attribute of a table must be dependent on the primary key, or we can say that, there should not be the case that a non-prime attribute is determined by another non-prime attribute. So, this transitive functional dependency should be removed from the table and also the table must be in the Second Normal Form.

User:

User_ID	Name	Date_of_ Birth	Medical_ insurance	Medical _history	Street	City	State

1 NF: The table satisfies INF as all attributes have atomic values.
2 NF: The table satisfies 2NF as there are no partial dependencies.
3 NF: The table satisfies 3 NF as there are no transitive dependencies.

User_phone_no:

User_ID	phone_no

1 NF: The table satisfies INF as all attributes have atomic values.
2 NF: The table satisfies 2NF as there are no partial dependencies.
3 NF: The table satisfies 3 NF as there are no transitive dependencies.

Organization:

Organization_ID	Name	Organization_name	Location	Government_approved

1 NF: The table satisfies INF as all attributes have atomic values.
2 NF: The table satisfies 2NF as there are no partial dependencies.
3 NF: The table satisfies 3 NF as there are no transitive dependencies.

Doctor:

Doctor_ID	Doctor_Name	Department_Name	organization_ID

1 NF: The table satisfies INF as all attributes have atomic values.
2 NF: The table satisfies 2NF as there are no partial dependencies.
3 NF: The table satisfies 3 NF as there are no transitive dependencies.

Patient:

Patient_ID	**organ_req**	reason_of_procurement	Doctor_ID	User_ID

1 NF: The table satisfies INF as all attributes have atomic values.
2 NF: The table satisfies 2NF as there are no partial dependencies.
3 NF: The table satisfies 3 NF as there are no transitive dependencies.

Donor:

Donor_ID	**organ_donated**	reason_of_donation	Organization_ID	User_ID

1 NF: The table satisfies INF as all attributes have atomic values.
2 NF: The table satisfies 2NF as there are no partial dependencies.
3 NF: The table satisfies 3 NF as there are no transitive dependencies.

Organ_available:

Organ_ID	**organ_name**	Donor_ID

1 NF: The table satisfies INF as all attributes have atomic values.
2 NF: The table satisfies 2NF as there are no partial dependencies.
3 NF: The table satisfies 3 NF as there are no transitive dependencies.

Transaction:

Patient_ID	**Organ_ID**	Donor_ID	Date_of_transaction	Status

1 NF: The table satisfies INF as all attributes have atomic values.
2 NF: The table satisfies 2NF as there are no partial dependencies.
3 NF: The table satisfies 3 NF as there are no transitive dependencies.

Organization_phone_no:

Organization_ID	Phone_no

1 NF: The table satisfies INF as all attributes have atomic values.
2 NF: The table satisfies 2NF as there are no partial dependencies.
3 NF: The table satisfies 3 NF as there are no transitive dependencies.

Doctor_phone_no:

Doctor_ID	Phone_no

1 NF: The table satisfies INF as all attributes have atomic values.
2 NF: The table satisfies 2NF as there are no partial dependencies.
3 NF: The table satisfies 3 NF as there are no transitive dependencies.

Organization_head:

Organization_ID	**Employee_ID**	Name	Date_of_joining	Term_length

1 NF: The table satisfies INF as all attributes have atomic values.
2 NF: The table satisfies 2NF as there are no partial dependencies.
3 NF: The table satisfies 3 NF as there are no transitive dependencies.

FRONT END DESIGN

3.1 HTML

Hypertext Markup Language (HTML) is the main markup language for creating web pages and other information that can be displayed in a web browser

HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>), within the web page content. HTML tags most commonly come in pairs like <h1> and </h1> although some tags represent empty elements and so are unpaired, for example <img>

The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page.

3.2 Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

3.3 MYSQL

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

MySQL is primarily an RDBMS and ships with no GUI tools to administer MySQL databases or manage data contained within the databases. Users may use the included command line tools, [citation needed] or download MySQL front-ends from various parties that have developed desktop software and web applications to manage MySQL databases, build database structures, and work with data records

3.4 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a markup language. The most common application is to style web pages written in HTML and XHTML, but the language can also be applied to any kind of XML document. CSS is designed primarily to enable the separation of document content (written in HTML or a similar markup language) from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content (such as by allowing for tableless web design).

MODULES

4.1 Login Module

This module is used by the existing admin to login into the website. It takes the username and password as input from the user and compares the data with the database. If the data matches, it takes the user to the home page, else it displays the wrong credential alert to the user.

4.2 Homepage Module

This module is the home page of the user. It has some description about the website and contains buttons to apply all other queries.

4.3 User Module

This module provides functionality to see, update and delete user details.

4.4 Search Module

This module lists out the details of the particular artwork if present in the gallery along with the options to delete, add and search an artwork in the gallery.

4.5 Add Module

This module provides functionality to add new data to the database.

4.6 Update Module

This module provides functionality to update the data about the entities in the database.

4.6 Remove Module

The module provides functionality to remove particular data from the database.

IMPLEMENTATION

5.1 MYSQL

```
CREATE DATABASE DBMS_PRO;
USE DBMS_PRO;
--for login values
CREATE TABLE login(
   username VARCHAR(20) NOT NULL,
   password VARCHAR(20) NOT NULL
);

INSERT INTO login VALUES ('admin','admin');

#table 1
CREATE TABLE User(
   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)
);

#table 2
CREATE TABLE User_phone_no(
   User_ID int NOT NULL,
   phone_no varchar(15),
   FOREIGN KEY(User_ID) REFERENCES User(User_ID) ON DELETE CASCADE
);

#table 3
CREATE TABLE Organization(
  Organization_ID int NOT NULL,
  Organization_name varchar(20) NOT NULL,
  Location varchar(20),
  Government_approved int, # 0 or 1
  PRIMARY KEY(Organization_ID)
);

#table 4
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)
);

#table 5
CREATE TABLE Patient(
    Patient_ID int NOT NULL,
    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 User(User_ID) ON DELETE CASCADE,
    FOREIGN KEY(Doctor_ID) REFERENCES Doctor(Doctor_ID) ON DELETE
CASCADE,
    PRIMARY KEY(Patient_Id, organ_req)
);

#table 6
CREATE TABLE Donor(
  Donor_ID int NOT NULL,
  organ_donated varchar(20) NOT NULL,
  reason_of_donation varchar(20),
  Organization_ID int NOT NULL,
  User_ID int NOT NULL,
  FOREIGN KEY(User_ID) REFERENCES User(User_ID) ON DELETE CASCADE,
  FOREIGN KEY(Organization_ID) REFERENCES Organization(Organization_ID) ON
DELETE CASCADE,
  PRIMARY KEY(Donor_ID, organ_donated)
);

#table 7
CREATE TABLE Organ_available(
  Organ_ID int NOT NULL AUTO_INCREMENT,
  Organ_name varchar(20) NOT NULL,
  Donor_ID int NOT NULL,
  FOREIGN KEY(Donor_ID) REFERENCES Donor(Donor_ID) ON DELETE
CASCADE,
  PRIMARY KEY(Organ_ID)
);

#table 8
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, #0 or 1
  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)
);

#table 9
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
);

#table 10
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
);

#table 11
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)
);
--
-- delimiter //
-- create trigger ADD_DONOR
-- after insert
-- on Donor
-- for each row
-- begin
-- insert into Organ_available(Organ_name, Donor_ID)
-- values (new.organ_donated, new.Donor_ID);
-- end//
-- delimiter ;
--
-- delimiter //
```

```
-- create trigger REMOVE_ORGAN
-- after insert
-- on Transaction
-- for each row
-- begin
-- delete from Organ_available
-- where Organ_ID = new.Organ_ID;
-- end//
-- delimiter ;

create table log (
  querytime datetime,
  comment varchar(255)
);

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 //

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 //
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 //

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 //

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 //

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 //

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 //
```

5.2 Flask

```
from flask import Flask,render_template,session,request,redirect,url_for,flash
import mysql.connector,hashlib
import matplotlib.pyplot as plt
import numpy as np

mydb = mysql.connector.connect(
  host='localhost',
  user='root',
  password='Brandy@17',
  database = 'DBMS_PRO'
)
mycursor = mydb.cursor(buffered=True)

app = Flask(__name__)

@app.route("/",methods = ['POST', 'GET'])
@app.route("/home",methods = ['POST','GET'])
def home():
    if not session.get('login'):
        return render_template('login.html'),401
    else:
        if session.get('isAdmin') :
            return render_template('home.html',username=session.get('username'))
        else :
            return home_student()

@app.route("/login",methods = ['GET','POST'])
def login():
    if request.method=='POST' :
        query = """SELECT * FROM login WHERE username = '%s'"""
%(request.form['username'])
        mycursor.execute(query)
        res = mycursor.fetchall()
        if mycursor.rowcount == 0:
            return home()
        if request.form['password'] != res[0][1]:
            return render_template('login.html')
        else:
            session['login'] = True
            session['username'] = request.form['username']
            session['password'] = request.form['password']
            session['isAdmin'] = (request.form['username']=='admin')
            return home()
    return render_template('login.html')
```

```
@app.route("/show_update_detail",methods=['POST','GET'])
def show_update_detail():
    if not session.get('login'):
        return redirect( url_for('home') )
    if request.method=='POST':
        if request.form['User_ID'] ==":
            return render_template("search_detail.html")
        qry = "Select * from User where User.User_ID = %s" %(request.form['User_ID'])
        qry1 = "Select * from User_phone_no where User_ID = %s"
%(request.form['User_ID'])
        mycursor.execute(qry)
        not_found=False
        res=()
        if(mycursor.rowcount > 0):
            res = mycursor.fetchone()
        else:
            not_found=True
        fields = mycursor.column_names
        qry_upd = "Select * from User where User_ID = %s" %(request.form['User_ID'])
        mycursor.execute(qry_upd)
        upd_res = ()
        if(mycursor.rowcount > 0):
            upd_res = mycursor.fetchone()
        fields_upd = mycursor.column_names
        mycursor.execute(qry1)
        phone_no = mycursor.fetchall()
        qry_pat = "select Patient_ID, organ_req, reason_of_procurement, Doctor_name
from Patient inner join Doctor on Doctor.Doctor_ID = Patient.Doctor_ID and User_ID =
%s" %(request.form['User_ID'])
        qry_don = "select Donor_ID, organ_donated, reason_of_donation,
Organization_name from Donor inner join Organization on
Organization.Organization_ID = Donor.Organization_ID and User_ID = %s"
%(request.form['User_ID'])
        qry_trans = "select distinct Transaction.Patient_ID, Transaction.Donor_ID,
Organ_ID, Date_of_transaction, Status from Transaction, Patient, Donor where
(Patient.User_ID = %s and Patient.Patient_ID = Transaction.Patient_ID) or
(Donor.User_Id= %s and Donor.Donor_ID = Transaction.Donor_ID)"
%((request.form['User_ID']),(request.form['User_ID']))
        #
        res_pat = ()
        res_dnr = ()
        res_trans = ()
        mycursor.execute(qry_pat)
        if(mycursor.rowcount > 0):
            res_pat = mycursor.fetchall()
        fields_pat = mycursor.column_names
        #
        mycursor.execute(qry_don)
```

```
        if(mycursor.rowcount > 0):
            res_dnr = mycursor.fetchall()
        fields_dnr = mycursor.column_names
        #
        mycursor.execute(qry_trans)
        if(mycursor.rowcount > 0):
            res_trans = mycursor.fetchall()
        fields_trans = mycursor.column_names
        print(res_trans)
        if("show" in request.form):
            return render_template('show_detail_2.html',res = res,fields = fields,
not_found=not_found, phone_no = phone_no, res_dnr = res_dnr, res_pat =
res_pat,res_trans = res_trans,fields_trans = fields_trans, fields_dnr = fields_dnr,
fields_pat = fields_pat)
        if("update" in request.form):
            return render_template('update_detail.html',res = upd_res,fields = fields_upd,
not_found=not_found)
        if "delete" in request.form:
            if not_found:
                return render_template('show_detail_2.html',res = res,fields = fields,
not_found=not_found, phone_no = phone_no,  res_dnr = res_dnr, res_pat =
res_pat,res_trans = res_trans,fields_trans = fields_trans, fields_dnr = fields_dnr,
fields_pat = fields_pat)
            else:
                qry2 = "DELETE FROM User where User_ID = %s"
%(request.form['User_ID'])
                mycursor.execute(qry2)
                mydb.commit()
                return render_template("home.html")

@app.route("/search_detail",methods = ['POST','GET'])
def search_detail():
    if not session.get('login'):
        return redirect( url_for('home') )
    return render_template('search_detail.html')

#--------------Adding Information----------------------------

@app.route("/add_<id>_page",methods = ['POST','GET'])
def add_page(id):
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from " + id.capitalize()
    mycursor.execute(qry)
    fields = mycursor.column_names

    return render_template('add_page.html',success=request.args.get('success'),
error=request.args.get('error'), fields = fields, id= id)
```

```
@app.route("/add_User", methods=['POST','GET'])
def add_User():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from User"
    mycursor.execute(qry)
    fields = mycursor.column_names

    val = ()

    for field in fields:
        temp = request.form.get(field)
        if field not in ['User_ID','Medical_insurance'] and temp != '':
            temp = "\'"+temp+"\'"
        if temp == '':
            temp = 'NULL'
        val = val + (temp,)

    qry = "INSERT INTO User Values (%s,%s,%s,%s,%s,%s,%s,%s)"%val
    print(qry)
    success = True
    error = False
    try:
        mycursor.execute(qry)
    except:
        print("Error : User not Inserted")
        error = True
        success = False
    mydb.commit()

    return redirect(url_for('add_page', id='User', error=error,success=success))

@app.route("/add_User_phone_no", methods=['POST','GET'])
def add_User_phone_no():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from User_phone_no"
    mycursor.execute(qry)
    fields = mycursor.column_names

    val = ()

    for field in fields:
        temp = request.form.get(field)
        if field not in ['User_ID','Phone_no'] and temp != '':
            temp = "\'"+temp+"\'"
        if temp == '':
```

```
            temp = 'NULL'
        val = val + (temp,)

    qry = "INSERT INTO User_phone_no Values (%s,%s)"%val
    print(qry)
    success = True
    error = False
    try:
        mycursor.execute(qry)
    except:
        print("Error : User not Inserted")
        error = True
        success = False
    mydb.commit()

    return redirect(url_for('add_page', id='User_phone_no', error=error,success=success))

@app.route("/add_Patient", methods=['POST','GET'])
def add_Patient():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Patient"
    mycursor.execute(qry)
    fields = mycursor.column_names

    val = ()

    for field in fields:
        temp = request.form.get(field)
        if field not in ['Patient_ID','User_ID','Doctor_ID'] and temp != '':
            temp = "\'"+temp+"\'"
        if temp == '':
            temp = 'NULL'
        val = val + (temp,)

    qry = "INSERT INTO Patient Values (%s,%s,%s,%s,%s)"%val
    print(qry)
    success = True
    error = False
    try:
        mycursor.execute(qry)
    except:
        print("Error : User not Inserted")
        error = True
        success = False
    mydb.commit()

    return redirect(url_for('add_page', id='Patient', error=error,success=success))
```

```
@app.route("/add_Donor", methods=['POST','GET'])
def add_Donor():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Donor"
    mycursor.execute(qry)
    fields = mycursor.column_names
    val = ()
    for field in fields:
        temp = request.form.get(field)
        if field not in ['Donor_ID','User_ID','Organization_ID'] and temp != '':
            temp = "\'"+temp+"\'"
        if temp == '':
            temp = 'NULL'
        val = val + (temp,)
    mycursor.execute( "START TRANSACTION;" )
    qry = "INSERT INTO Donor Values (%s,%s,%s,%s,%s)"%val
    print(qry)
    success = True
    error = False
    try:
        mycursor.execute(qry)
    except:
        print("Error : User not Inserted")
        error = True
        success = False

    qry_insert = "insert into Organ_available (Organ_name, Donor_ID) Values (%s,%s)
"%(val[1],val[0])

    mycursor.execute(qry_insert)

    mycursor.execute("COMMIT;")

    mydb.commit()

    return redirect(url_for('add_page', id='Donor', error=error,success=success))

@app.route("/add_Doctor", methods=['POST','GET'])
def add_Doctor():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Doctor"
    mycursor.execute(qry)
    fields = mycursor.column_names

    val = ()
```

```
    for field in fields:
        temp = request.form.get(field)
        if field not in ['Doctor_ID','Organization_ID'] and temp != '':
            temp = "\'"+temp+"\'"
        if temp == '':
            temp = 'NULL'
        val = val + (temp,)

    qry = "INSERT INTO Doctor Values (%s,%s,%s,%s)"%val
    print(qry)
    success = True
    error = False
    try:
        mycursor.execute(qry)
    except:
        print("Error : User not Inserted")
        error = True
        success = False
    mydb.commit()

    return redirect(url_for('add_page', id='Doctor', error=error,success=success))
```

SNAPSHOTS

Fig 6.1:Login Page

Fig 6.2:Home Page

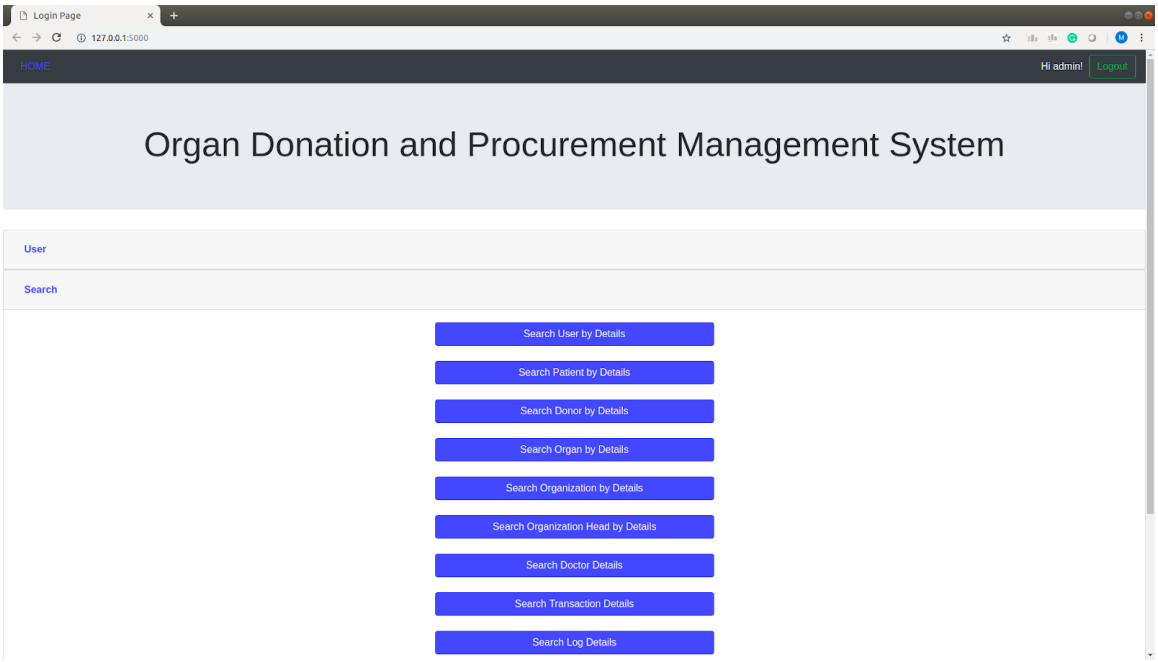
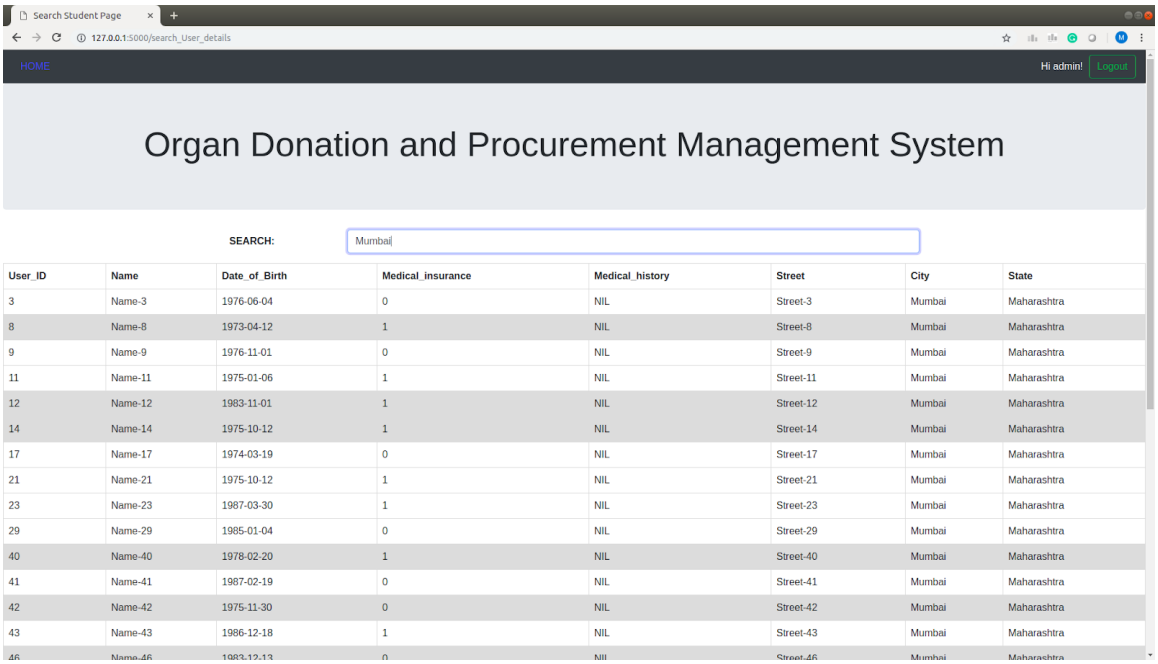


Fig 6.3:Main Page – Drop Down Menu



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

Fig 6.4:Searching Option

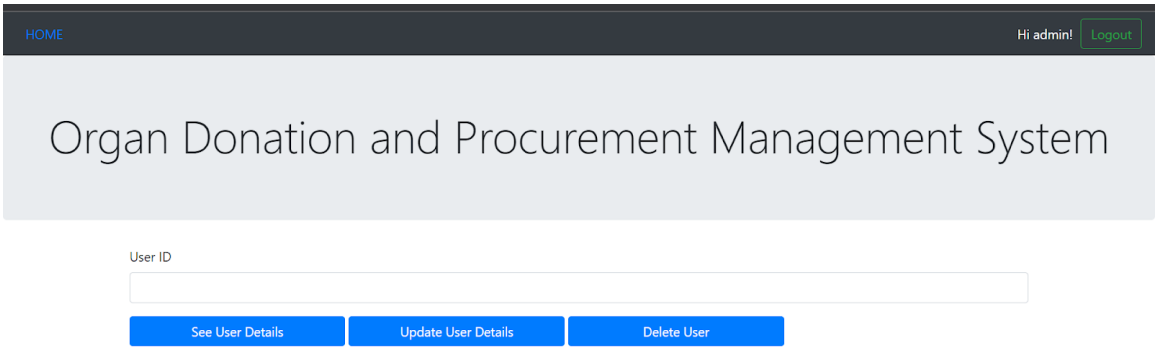


Fig 6.4:User Menu

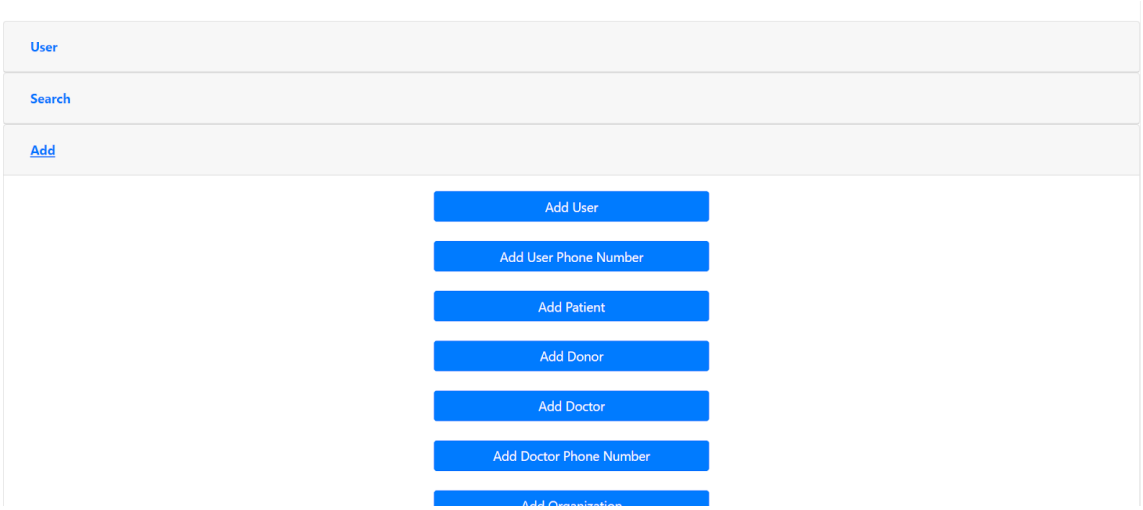


Fig 6.4: Add Option

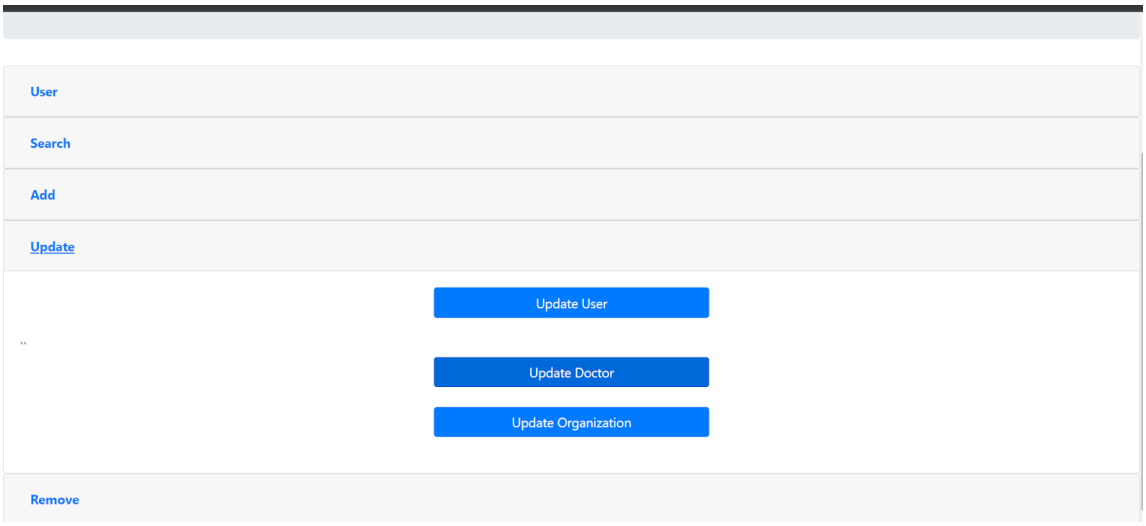


Fig 6.4:Update Option

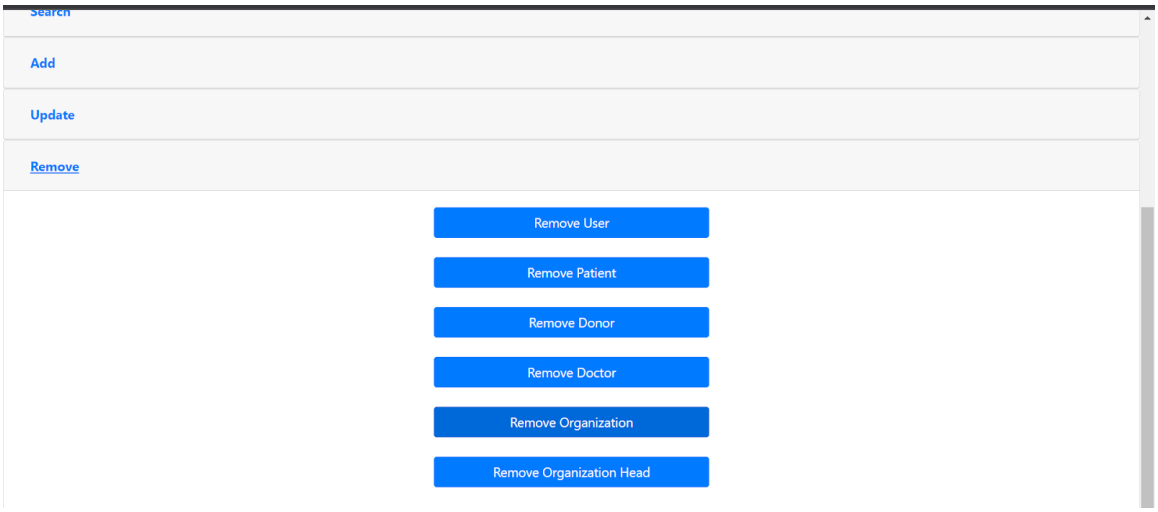


Fig 6.4:Remove Option

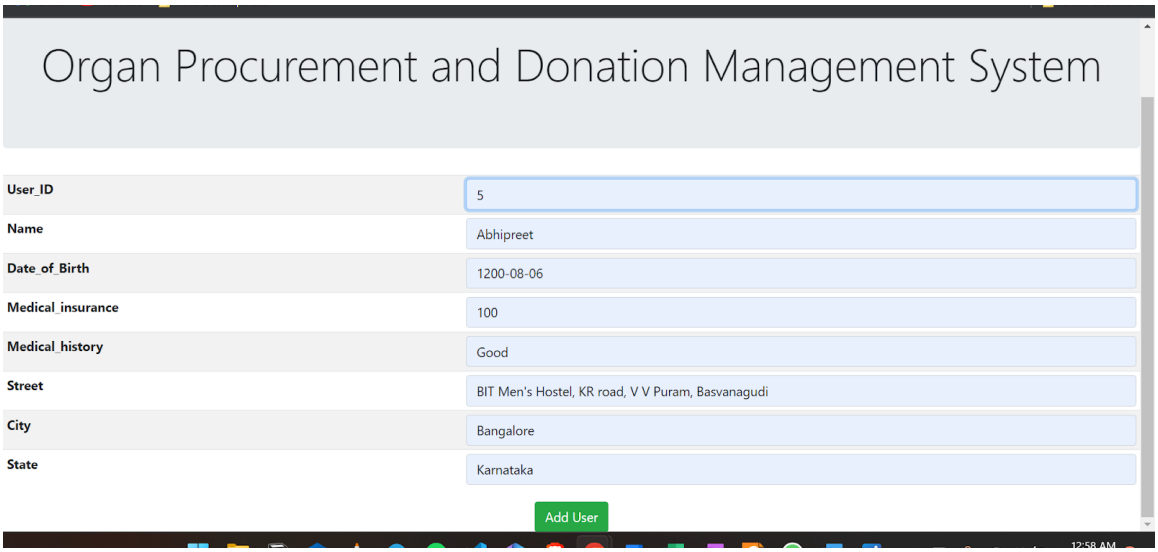


Fig 6.4:Adding User Demo

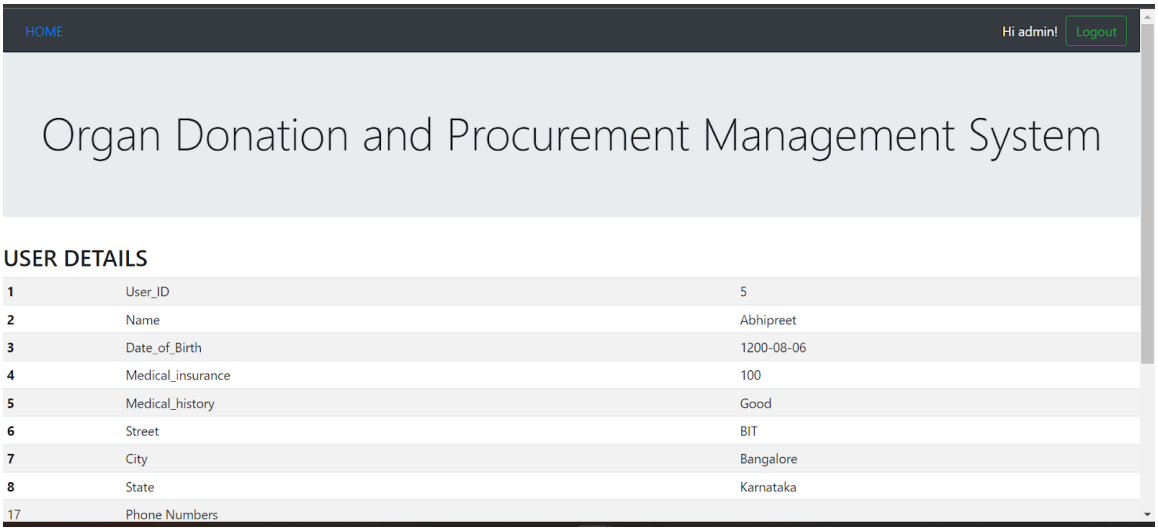


Fig 6.4:Displaying User Details

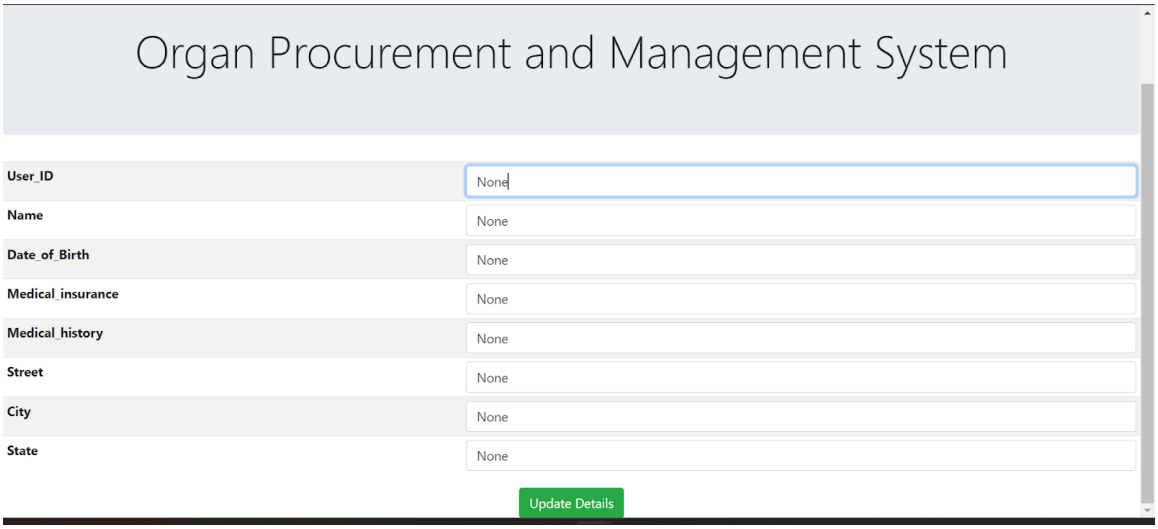


Fig 6.4:Updation Panel

APPLICATIONS

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 networks 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 patients and Donors in a well-formed way that can be processed easily. Records of donors 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.

CONCLUSION

8.1 Conclusion

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.

8.2 References

- https://www.w3schools.com
- https://www.stackoverflow.com
- Flask and MySQL Web Development - By Corey Schafer
- https://www.youtube.com