ORGAN TRANSPLANTATION DATABASE SYSTEM

A PROJECT REPORT

Submitted by

SHREEMAN M [RA2211028010166]

M KRISHNA CHAITANAYA [RA2211028010165]

S CHARITH [RA2211028010197]

Under the Guidance of

Dr. Rajaram V

Assistant Professor, Department of Networking & Communications

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF NETWORKING & COMMUNICATIONS

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR– 603 203

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR–603 203

BONAFIDE CERTIFICATE

Register no. RA2211028010166, RA2211028010165 and RA2211028010197

Certified to be the bonafide work done by **Shreeman M, M Krishna Chaitanaya, S Charith** of II year B. Tech Degree Course in the Project Course – **21CSC205P Database Management Systems** in **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**, Kattankulathur for the academic year 2023-2024.

Date:

Faculty in Charge
Dr. Rajaram V
Assistant Professor
Department of Newtorking & Communications
SRMIST -KTR

HEAD OF THE DEPARTMENT
Dr. K. Ananpurarani
Head of the Department
Department of Networking & Communications
SRMIST - KTR

ABSTRACT

The Organ Transplantation Database (OTD) System stands as a critical solution for individuals suffering from end-stage organ failure, offering a chance at extended life and improved quality of life. However, the efficacy of organ transplantation relies heavily on the efficiency of the associated administrative processes, from donor registration to recipient matching and organizing associates of non-governmental organizations related to obtaining said organs. This project proposes the development of an Organ Transplantation System utilizing Database Management Systems (DBMS) to streamline and optimize the entire transplantation process. The proposed system aims to address several key challenges encountered in organ transplantation, including donor-recipient matching, organ allocation, tracking donor and recipient information, and ensuring timely communication between healthcare providers and patients. By leveraging the capabilities of a DBMS, such as data storage, retrieval, and manipulation, the system will centralize and organize vast amounts of data related to donors, recipients, organs, medical histories, and intermediaries. The OTD system integrates advanced functionalities encompassing comprehensive donor and recipient management, real-time organ inventory tracking, and robust data analytics capabilities. By centralizing relevant data and automating workflows, the system ensures timely interventions, optimal organ allocation, and seamless collaboration among healthcare and non- profit professionals. In summary, the Organ Transplantation Database represents a pivotal tool in optimizing transplant procedures, fostering collaboration, and improving patient outcomes in the field of organ transplantation.

PROBLEM STATEMENT

Organ transplantation represents a complex medical procedure that requires meticulous coordination and management of donor, recipient, and transplant-related data. However, the existing infrastructure for handling this critical information is often fragmented, inefficient, and prone to errors, leading to significant challenges throughout the transplantation process. There is an urgent need for the development of a comprehensive Organ Transplantation Database System (OTDS) to address these pressing issues. Manual data entry processes and paper-based record-keeping systems contribute to inefficiencies, delays, and errors in the transplantation workflow, potentially compromising patient outcomes. Security and Privacy Concerns: Sensitive patient information is vulnerable to security breaches, unauthorized access, and data leaks, raising concerns about patient privacy and regulatory compliance. The absence of robust data analytics tools restricts the ability to derive insights, identify trends, and optimize transplant protocols, hindering advancements in the field. Limited Analytical Capabilities: The absence of robust data analytics tools restricts the ability to derive insights, identify trends, and optimize transplant protocols, hindering advancements in the field. By addressing these challenges and incorporating these features, the Organ Transplantation Database System (OTDS) aims to revolutionize the management of organ transplantation, improve patient outcomes, and advance the field of transplant medicine.

TABLE OF CONTENTS

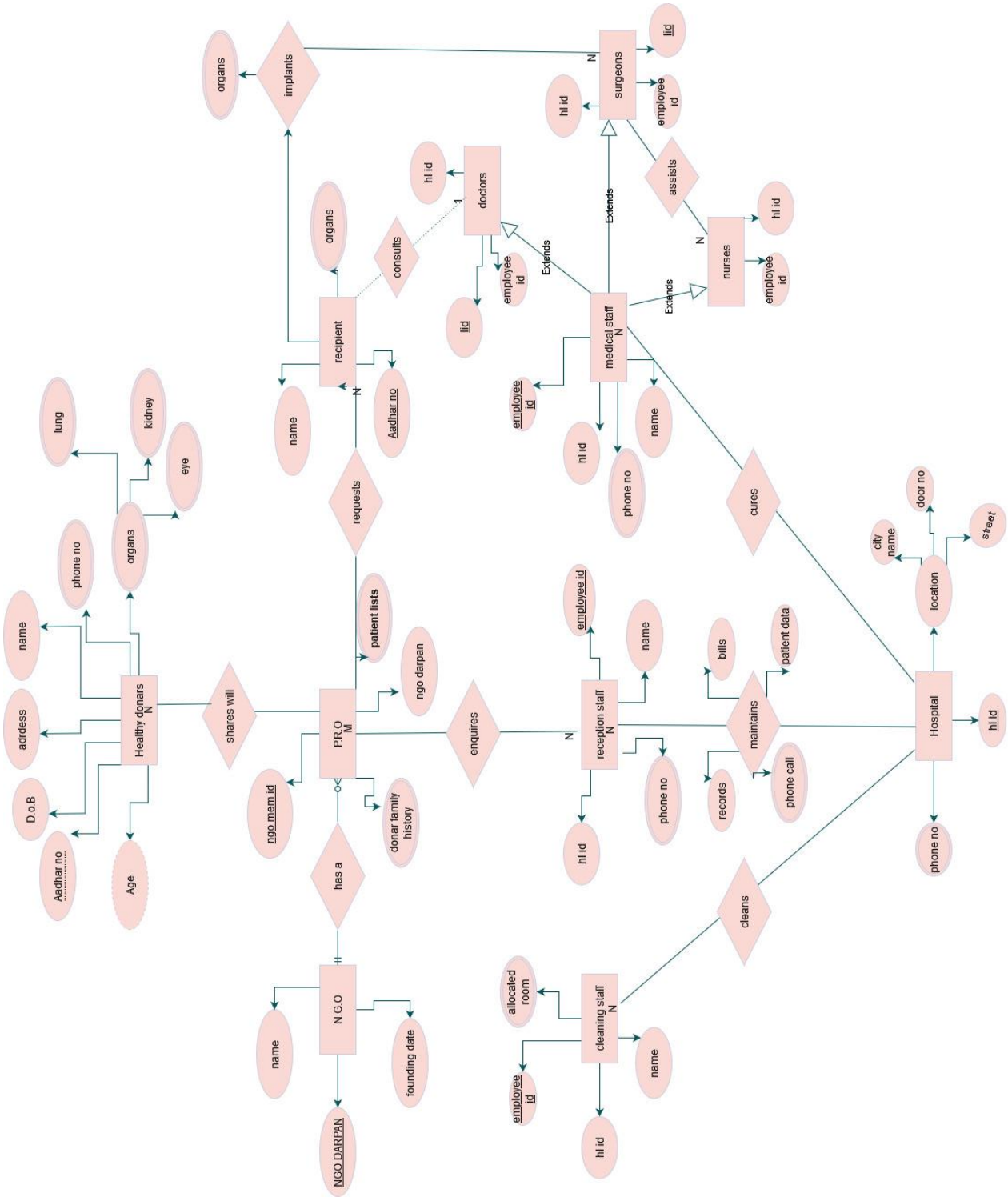
CHAPTER 1

PROBLEM UNDERSTANDING

This project aims to develop a comprehensive Organ Transplantation System using a Database Management System (DBMS). The system encompasses various functionalities including donor and recipient cataloging, staff management, storing hospital information and logging NGO information.

1. **Functionality**: Capture and track comprehensive profiles of both donors and recipients, including medical history, personal information, and the NGO's they are associated with.
2. **Database Management**: Centralize all relevant data pertaining to organ transplantation to ensure consistency, accuracy, and accessibility. Design a database architecture capable of handling large volumes of data while ensuring optimal performance. Implement mechanisms to maintain data integrity, prevent data loss, and facilitate data recovery in case of system failures.
3. **User Interface**: Design a user-friendly interface that is easy to navigate and requires minimal training for users at various levels of proficiency. Ensure accessibility so it is easy to navigate and use.
4. **Automation**: Automate repetitive tasks such as data entry, notifications, and scheduling to improve efficiency and reduce human error.
5. **Benefits**: Provide Operational Excellence by streamlining administrative processes and reduce paperwork, allowing healthcare professionals to focus more on patient care. Optimize resource utilization, reduce manual labor, and prevent costly errors, resulting in overall cost savings for healthcare institutions.

ER DIAGRAM


organs
implants
N
lid
surgeons
hi id
employee id
assists
hi id
doctors
Extends
Extends
N
nurses
hi id
employee id
organs
consults
1
lid
employee id
recipient
N
name
Aadhar no
requests
medical staff
N
employee id
hi id
phone no
name
cures
lung
kidney
eye
organs
phone no
name
adrdess
D.o.B
Aadhar no
Age
Healthy donars
N
shares will
patient lists
ngo darpan
P.R.O
M
ngo mem id
donar family history
has a
N.G.O
name
founding date
NGO DARPAN
enquires
employee id
name
reception staff
N
hi id
phone no
bills
patient data
maintains
records
phone call
door no
street
city
name
location
Hospital
hi id
phone no
cleans
allocated room
cleaning staff
N
employee id
hi id
name

CHAPTER 2

Table Creation:

```
CREATE TABLE Healthy_Donor (
    Aadhar_No. Number(12) PRIMARY KEY,
    Age Number(2),
    DOB Date,
    Name VARCHAR(20),
    Addresss VARCHAR(20),
    Phone_No Number(10),
    Organs VARCHAR(20)
    NGO_Mem_ID Number(10) FOREIGN KEY References PRO(NGO_Mem_ID)
);

CREATE TABLE Reception_Staff (
    Emp_ID Number(10) PRIMARY KEY,
    Name VARCHAR(20),
    Phone_No Number(10),
    HL_ID Number(10) FOREIGN KEY References Hospital(HLID)
);

CREATE TABLE Recipient (
    Aadhar_No Number(10) PRIMARY KEY,
```

```
    Name VARCHAR(20),
    Organs VARCHAR(20),
     NGO_Mem_ID Number(10) FOREIGN KEY References PRO(NGO_Mem_ID)
);

CREATE TABLE PRO (
    NGO_Mem_ID Number(10) PRIMARY KEY,
    Donor_Family_History VARCHAR(50),
    Patient_Lists VARCHAR(50),
    NGO_DARPAN Number(10) FOREIGN KEY References NGO(NGO_DARPAN)
);

CREATE TABLE NGO (
    NGO_Darpan Number(10) PRIMARY KEY,
    Name VARCHAR(20),
    Founding_Date Date,
    NGO_Mem_ID Number(10) FOREIGN KEY References PRO(NGO_Mem_ID)
);

CREATE TABLE Cleaning_Staff (
    Emp_ID Number(10) PRIMARY KEY,
    Name VARCHAR(20),
    Allocated_Room Number(3),
```

```
    HLID Number(10) FOREIGN KEY References Hospital(HLID)
);

CREATE TABLE Hospital (
    HLID Number(10) PRIMARY KEY,
    PH_NO Number(10),
    Location VARCHAR(50)
);

CREATE TABLE Medical_Staff (
    Emp_ID Number(10) PRIMARY KEY,
    Name VARCHAR(20),
    PH_NO Number(10),
    HLID Number(10) Number(10) FOREIGN KEY References Hospital(HLID)
);

CREATE TABLE Nurses (
    Emp_ID Number(10) PRIMARY KEY,
    HLID Number(10) FOREIGN KEY References Hospital(HLID)
);

CREATE TABLE Surgeons (
    LID Number(10) PRIMARY KEY,
```

EMP_ID Number(10),

HLID Number(10) Number(10) FOREIGN KEY References Hospital(HLID)

);

CREATE TABLE Doctors (

LID Number(10) PRIMARY KEY,

EMP_ID Number(10),

HLID Number(10) Number(10) FOREIGN KEY References Hospital(HLID)

);

```
SQL> CREATE TABLE Healthy_Donor (
  2      Aadhar_No. Number(12) PRIMARY KEY,
  3      Age Number(2),
  4      DOB Date,
  5      Name VARCHAR(20),
  6      Addresss VARCHAR(20),
  7      Phone_No Number(10),
  8      Organs VARCHAR(20)
  9      NGO_Mem_ID Number(10) FOREIGN KEY References PRO(NGO_Mem_ID)
 10  );
```

```
SQL> CREATE TABLE Reception_Staff (
  2      Emp_ID Number(10) PRIMARY KEY,
  3      Name VARCHAR(20),
  4      Phone_No Number(10),
  5      HL_ID Number(10) FOREIGN KEY References Hospital(HLID)
  6  );
```

```
SQL> CREATE TABLE Recipient (
  2      Aadhar_No Number(10) PRIMARY KEY,
  3      Name VARCHAR(20),
  4      Organs VARCHAR(20),
  5       NGO_Mem_ID Number(10) FOREIGN KEY References PRO(NGO_Mem_ID)
  6  );
```

```
SQL> CREATE TABLE PRO (
  2      NGO_Mem_ID Number(10) PRIMARY KEY,
  3      Donor_Family_History VARCHAR(50),
  4      Patient_Lists VARCHAR(50),
  5      NGO_DARPAN Number(10) FOREIGN KEY References NGO(NGO_DARPAN)
  6  );
```

```
SQL> CREATE TABLE NGO (
  2      NGO_Darpan Number(10) PRIMARY KEY,
  3      Name VARCHAR(20),
  4      Founding_Date Date,
  5      NGO_Mem_ID Number(10) FOREIGN KEY References PRO(NGO_Mem_ID)
  6  );
```

```
SQL> CREATE TABLE Cleaning_Staff (
  2      Emp_ID Number(10) PRIMARY KEY,
  3      Name VARCHAR(20),
  4      Allocated_Room Number(3),
  5      HLID Number(10) FOREIGN KEY References Hospital(HLID)
  6  );
```

```
SQL> CREATE TABLE Hospital (
  2      HLID Number(10) PRIMARY KEY,
  3      PH_NO Number(10),
  4      Location VARCHAR(50)
  5  );
```

```
SQL> CREATE TABLE Medical_Staff (
  2      Emp_ID Number(10) PRIMARY KEY,
  3      Name VARCHAR(20),
  4      PH_NO Number(10),
  5      HLID Number(10) Number(10) FOREIGN KEY References Hospital(HLID)
  6  );
```

```
SQL> CREATE TABLE Nurses (
  2      Emp_ID Number(10) PRIMARY KEY,
  3      HLID Number(10) FOREIGN KEY References Hospital(HLID)
  4  );
```

```
SQL> CREATE TABLE Surgeons (
  2      LID Number(10) PRIMARY KEY,
  3      EMP_ID Number(10),
  4      HLID Number(10) Number(10) FOREIGN KEY References Hospital(HLID)
  5  );
```

```
SQL> CREATE TABLE Doctors (
  2      LID Number(10) PRIMARY KEY,
  3      EMP_ID Number(10),
  4      HLID Number(10) Number(10) FOREIGN KEY References Hospital(HLID)
  5  );
```

Queries using Join

```
SELECT *
FROM PRO p
JOIN NGO n ON n.NGO_Mem_ID = p.NGO_Mem_ID;

SELECT *
FROM Cleaining_Staff c
JOIN Hospital h ON h.HLID = c.HLID;

SELECT *
FROM Reception_Staff r
JOIN Hospital h ON h.HLID = r.HLID

SELECT *
FROM Healthy_Donor h
JOIN PRO p ON p.NGO_Mem_ID = h.NGO_Mem_ID
JOIN Recepient r ON r.NGO_Mem_ID = h.NGO_Mem_ID;
```

```
SQL> SELECT *
  2  FROM PRO p
  3  JOIN NGO n ON n.NGO_Mem_ID = p.NGO_Mem_ID;
```

```
SQL> SELECT *
  2  FROM Cleaining_Staff c
  3  JOIN Hospital h ON h.HLID = c.HLID;
```

```
SQL> SELECT *
  2  FROM Reception_Staff r
  3  JOIN Hospital h ON h.HLID = r.HLID
  4  ;
```

```
SQL> SELECT *
  2  FROM Medical_Staff m
  3  JOIN Nurses n ON n.HLID = m.HLID
  4  JOIN Surgeons s ON  s.HLID = m.HLID
  5  JOIN Doctors d ON d.HLID = m.HLID;
```

```
SQL> SELECT *
  2  FROM Reception_Staff r
  3  JOIN Hospital h ON h.HLID = r.HLID
  4  ;
```

Queries using Trigger

```
CREATE OR REPLACE TRIGGER CheckNewHospital
BEFORE INSERT OR UPADATE ON Hospital
FOR EACH ROW
BEGIN
   IF (:new.PH_NO = NULL or :new.Location = NULL) THEN
        dbms_output.put_line(“Please enter a valid Phone No.”);
END;
```

```
CREATE OR REPLACE TRIGGER CheckRoom
BEFORE INSERT OR UPDATE ON Cleaning_Staff
FOR EACH ROW
BEGIN
   IF(:new.Allocated_Room > 810) THEN
        dbms_output.put_line(“Please enter a valid Room No.”);
END;
```

```
CREATE OR REPLACE TRIGGER SetDefaultRecipientOrgan
BEFORE INSERT ON Recipient
FOR EACH ROW
```

```
BEGIN
   :new.organ := 'Not available';
END;
```

```
CREATE OR REPLACE TRIGGER SetDefaultPROHistory
BEFORE INSERT ON PRO
FOR EACH ROW
BEGIN
   :new.Donor_Family_History := “Not Collected”;
END;
```

```
SQL> CREATE OR REPLACE TRIGGER CheckNewHospital
  2  BEFORE INSERT OR UPADATE ON Hospital
  3  FOR EACH ROW
  4  BEGIN
  5      IF (:new.PH_NO = NULL or :new.Location = NULL) THEN
  6     dbms_output.put_line(“Please enter a valid Phone No.”);
  7  END;
```

```
SQL> CREATE OR REPLACE TRIGGER SetDefaultRecipientOrgan
  2  BEFORE INSERT ON Recipient
  3  FOR EACH ROW
  4  BEGIN
  5      :new.organ := 'Not available';
  6  END;
```

```
SQL> CREATE OR REPLACE TRIGGER CheckRoom
  2  BEFORE INSERT OR UPDATE ON Cleaning_Staff
  3  FOR EACH ROW
  4  BEGIN
  5      IF(:new.Allocated_Room > 810) THEN
  6     dbms_output.put_line(“Please enter a valid Room No.”);
  7  END;
```

Cursors

```
SQL> DECLARE
  2     CURSOR c_Donor IS
  3             SELECT * FROM Healthy_Donor;
  4     CURSOR c_PRO IS
  5             SELECT * FROM PRO
  6     CURSOR c_NGO IS
  7             SELECT * FROM NGO
  8     CURSOR c_Reception_Staff IS
  9             SELECT * FROM Reception_Staff
 10     CURSOR c_Recipient IS
 11             SELECT * FROM Recipient
 12     CURSOR c_Cleaning_Staff IS
 13             SELECT * FROM Cleaning_Staff
 14     CURSOR c_Hospital IS
 15             SELECT * FROM Hospital
 16     CURSOR c_Medical_Staff IS
 17             SELECT * FROM Medical_Staff
 18     CURSOR c_Doctor IS
 19             SELECT * FROM Doctor
 20     CURSOR c_Nurse IS
 21             SELECT * FROM Nurse
 22  BEGIN
 23  FOR rec in c_Doctor
 24             dbms_output.put_line(rec.Emp_ID, rec.LID, rec.HLID);
 25  END LOOP;
 26  END;
 27  /
```

DECLARE

CURSOR c_Donor IS

SELECT * FROM Healthy_Donor;

CURSOR c_PRO IS

SELECT * FROM PRO;

CURSOR c_NGO IS

SELECT * FROM NGO;

```
    CURSOR c_Reception_Staff IS
        SELECT * FROM Reception_Staff;
    CURSOR c_Recipient IS
        SELECT * FROM Recipient;

    CURSOR c_Cleaning_Staff IS
        SELECT * FROM Cleaning_Staff;

    CURSOR c_Hospital IS
        SELECT * FROM Hospital;

    CURSOR c_Medical_Staff IS
        SELECT * FROM Medical_Staff;

    CURSOR c_Doctor IS
        SELECT * FROM Doctor;

    CURSOR c_Nurse IS
        SELECT * FROM Nurse;

BEGIN
```

```
FOR rec in c_Doctor
        dbms_output.put_line(rec.Emp_ID, rec.LID, rec.HLID);
END LOOP;


END;
```

CHAPTER 4

PITFALLS

1. Data Redundancy: Duplicate data might exist in multiple places, leading to data inconsistency and wasted storage space. For example, if organ details are repeated in both the healthy donor table and the PRO table.

2. Update Anomalies: If a piece of data is stored redundantly and needs to be updated, it must be updated in multiple places, increasing the chance of inconsistencies or errors.

3. Deletion Anomalies: Deleting a record might unintentionally remove related data. For example, deleting a book might also delete associated rental records if not handled properly.

4. Insertion Anomalies: Adding new data might not be possible or might result in incomplete records if the required information is not available.

5. Inconsistent Data: Data might be stored in different formats or with different naming conventions, leading to confusion and difficulty in querying.

6. Incomplete Information: Certain attributes might be missing from records, making it challenging to derive meaningful insights or perform certain operations.

7. Difficulty in Querying: Without proper normalization, querying the database might be complex and inefficient, requiring multiple joins and leading to slower performance.

8. Scalability Issues: As the database grows, the lack of normalization can lead to increased storage requirements and slower query performance.

9. Limited Data Integrity: Without normalization, enforcing data integrity constraints such as primary keys, foreign keys, and unique constraints might be challenging, leading to

inconsistencies and errors in the data.

Identifying and addressing these pitfalls through normalization can help improve the overall quality, efficiency, and reliability of your database management system.

FUNCTIONAL DEPENDENCIES IN THE PROJECT

- Healthy Donor:

  Aadhar No -> {Aadhar No, DOB, Address, Name, Age, Phone No, Organs, NGO Mem ID}

  {Aadhar No, Name} -> Name (Trivial Dependency)

  Aadhar No -> {Name, Age} (Non - Trivial Dependency)

  Aadhar No -> {Name, Age, Organs} (Multi-Valued Dependency)

  Aadhar No -> Name : Name -> NGO Mem ID => Aadhar No -> NGO Mem ID (Transitive Dependency)

- PRO:

  NGO_Mem_ID -> {NGO Mem ID, Patients lists, Donor Family History, NGO Darpan}

  {NGO_Mem_ID, NGO Darpan) -> NGO Darpan (Trivial Dependency)

  NGO Mem ID -> Donor Family History (Non-Trivial Dependency)

  NGO Mem ID -> {Patient List, Donor Family History (Mutli-Valued Dependency)

  NGO Mem ID -> Patient List: Patient List -> Donor Family History => NGO Mem ID -> Donor Family History (Transitive Dependency)

- NGO:

  NGO_Mem_ID -> {NGO Mem ID, Patients lists, Donor Family History, NGO Darpan}

{NGO_Mem_ID, NGO Darpan) -> NGO Darpan (Trivial Dependency)

NGO Mem ID -> Donor Family History (Non-Trivial Dependency)

NGO Mem ID -> {Patient List, Donor Family History (Mutli-Valued Dependency)

NGO Mem ID -> Patient List : Patient List -> Donor Family History => NGO Mem ID -> Donor Family History (Transitive Dependency)

- Recipient:

  Aadhar No -> {Aadhar No, Name, Organs, NGO Mem ID, LID}

  {Aadhar No, Name} -> Name (Trivial Dependency)

  Aadhar No -> Name (Non - Trivial Dependency)

  Aadhar No -> {Name, Organs} (Multi - Valued Dependency)

  Aadhar No -> Name : Name -> Organs => Aadhar No -> Organs (Transitive Dependency)

- Reception Staff:

  Employee ID -> {Employee ID, Phone No, Name, HLID}

  {Employee ID, Name} -> Name

  Employee ID -> HL ID

  Employee ID -> {Name, Phone No}

  Employee ID -> Name : Name -> HLID => Employee ID -> HLID

- Cleaning Staff:

  Employee ID -> {Employee ID, Name, Allocated Room, HLID}

  {Employee ID, Name} -> Name

  Employee ID -> Allocated Room

  Employee ID -> {Allocated Room, HLID}

  Employee ID -> Allocated Room : Allocated Room -> HLID => Employee ID -> HLID

- Doctors:

  Employee ID -> {Employee ID, LID, HLID}

  {Employee ID, LID} -> LID

  Employee ID -> LID

  Employee ID -> {LID, HLID}

- Hospital:

  HLID -> {HLID, Phone No, Location}

  {HLID, Phone No} -> Phone No

  HLID -> Location

  HLID -> {Location, Phone No}

  HLID -> Location : Location -> Phone No => HLID -> Phone No

- Medical Staff:

  Employee ID -> {Employee ID, Name, Phone No, HLID}

  {Employee ID, Name} -> Name

  Employee ID -> Phone No

  Employee ID -> {Name, Phone No}

  Employee ID -> Name : Name -> HLID => Employee ID -> HLID

1st Normal Form:

Before Normalization:

Aadhar No	DoB	Address	Name	Phone No	Organs	NGO Mem ID
XXXX	12/2/12	XX,XX,XX	Shreman	XXXX	Liver, Lungs, Kidney …	XXXX

After Normalization:

Aadhar No	DoB	Address	Name	Phone No	NGO Mem ID
XXXX	12/2/12	XX,XX,XX	Shreman	XXXX	XXXX

Aadhar No	DoB	Address	Name	Phone No	NGO Mem ID
XXXX	12/2/12	XX,XX,XX	Shreman	XXXX	XXXX

2nd Normal Form

Before Normalization:

Employee ID	Name	Allotted Room	HLID
XXXX	Krishna	304	XXXX
XXXX	Krishna	205	XXXX
YYYY	Shreeman	204	XXXX

After Normalization:

Employee ID	Name	HLID
XXXX	Shreeman	XXXXX

Employee ID	Allotted Room
XXXX	301
XXXX	202
XXXX	103

4th Normal Form

Before Normalization:

HLID	Location	Phone No
XXXX	Katankulathur	XXXXXX

After Normalization:

HLID	Location
XXXX	Kattankulathur

Location	Phone No
Kattankulathur	XXXXXXX

5th Normal Form

Before Normalization:

NGO_Mem_ID	Patients List	Donor Family History	NGO Darpan
XXXX	Shreeman	Luekemia	XXXX
XXXXX	Pranav	Cancer	XXXX

After Normalization:

NGO Mem ID	NGO Darpan	Patient Lists
XXX	XXX	Shreeman
XXX	XXX	Pranav

NGO Mem ID	NGO Darpan	Donor Family History
XXXX	XXXX	Leukemia
XXXX	XXXX	Caner

Patient Lists	Donor Family History
Shreeman	Leukemia
Pranav	Cancer

CHAPTER 5

IMPLEMENTATION OF CONCURRECY CONTROL

1. Concurrency Control:

• We'll utilize SQL transactions to group SQL statements into atomic units of work. We'll begin a transaction (BEGIN TRANSACTION), execute the required SQL statements, and commit the transaction (COMMIT) once the operations are completed successfully. We'll also rollback the transaction (ROLLBACK) in case of errors to maintain data consistency.

• We'll implement locking mechanisms using SQL commands such as SELECT ... FOR UPDATE to lock rows during read operations and UPDATE or DELETE statements to prevent concurrent modifications.

• We'll ensure that our Java code handles concurrent requests appropriately by coordinating access to shared resources and avoiding race conditions.

2. Recovery Mechanisms:

• We'll implement logging in our Java code to record changes made to the database. We can create a log table in our database to store relevant information such as timestamps, transaction IDs, and the nature of changes.

• We'll use Java's error handling mechanisms to capture and log exceptions and errors that occur during database operations.

• We'll develop scripts or functions to perform recovery actions, such as restoring the database from backups or replaying logged transactions to recover from failures.

3**. Testing and Evaluation:**

- We'll thoroughly test our Java and SQL code, focusing on concurrency scenarios such as simultaneous user requests and concurrent transactions.
- We'll perform stress testing to evaluate the performance and scalability of our system under high load conditions.
- We'll use logging and monitoring tools to analyze system behavior and identify any issues related to concurrency control and recovery.

By following these steps and integrating concurrency control and recovery mechanisms into our Java and SQL-based Organ Transplantation Database system project, we can ensure the reliability, consistency, and fault tolerance of our application.

CHAPTER 6

CODE:

```
import java.awt.*;
import javax.swing.*;
import java.sql.*;
import java.awt.event.*;

public class OrganTransplantationDatabase extends JFrame implements ActionListener {
    JButton donorButton, recipientButton, displayButton, doctorsButton, hospitalButton,
nursesButton;
    JTextArea outputArea;
    Connection con;
    Statement stmt;

    public OrganTransplantationDatabase() {
        super("Organ Transplantation Database");
        setLayout(new GridLayout(4, 1));

        donorButton = new JButton("Healthy Donor");
        donorButton.addActionListener(this);
        add(donorButton);

        recipientButton = new JButton("Recipient");
```

```
recipientButton.addActionListener(this);
add(recipientButton);

doctorsButton = new JButton("Doctors");
doctorsButton.addActionListener(this);
add(doctorsButton);

hospitalButton = new JButton("Hospital");
hospitalButton.addActionListener(this);
add(hospitalButton);

nursesButton = new JButton("Nurses");
nursesButton.addActionListener(this);
add(nursesButton);

displayButton = new JButton("Display Records");
displayButton.addActionListener(this);
add(displayButton);

outputArea = new JTextArea(10, 30);
add(new JScrollPane(outputArea), BorderLayout.PAGE_END);

try {
```

```
            Class.forName("com.mysql.jdbc.Driver");
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/Organ", "root",
"root");
            stmt = con.createStatement();
        } catch (Exception e) {
            System.out.println(e);
        }

        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent ae) {
        if (ae.getSource() == donorButton) {
            new DonorWindow();
        } else if (ae.getSource() == recipientButton) {
            new RecipientWindow();
        } else if (ae.getSource() == displayButton) {
            displayRecords();
        } else if (ae.getSource() == doctorsButton) {
            new DoctorsWindow();
        } else if (ae.getSource() == hospitalButton) {
```

```
            new HospitalWindow();
        } else if (ae.getSource() == nursesButton) {
            new NursesWindow();
        }
    }

    class DonorWindow extends JFrame implements ActionListener {
        JTextField donorAadharField, donorNameField, donorAgeField, donorDobField,
donorOrgansField, donorPhoneField, donorAddressField, donorNgoIdField;
        JButton donorInsertButton;

        public DonorWindow() {
            super("Healthy Donor");
            setLayout(new GridLayout(9, 1));

            JLabel donorAadharLabel = new JLabel("Aadhar No:");
            donorAadharField = new JTextField(20);
            add(donorAadharLabel);
            add(donorAadharField);
            JLabel donorNameLabel = new JLabel("Name:");
            donorNameField = new JTextField(20);
            add(donorNameLabel);
            add(donorNameField);
```

```
JLabel donorAgeLabel = new JLabel("Age:");
donorAgeField = new JTextField(20);
add(donorAgeLabel);
add(donorAgeField);
JLabel donorDobLabel = new JLabel("Date of Birth:");
donorDobField = new JTextField(20);
add(donorDobLabel);
add(donorDobField);
JLabel donorOrgansLabel = new JLabel("Organs:");
donorOrgansField = new JTextField(20);
add(donorOrgansLabel);
add(donorOrgansField);
JLabel donorPhoneLabel = new JLabel("Phone No:");
donorPhoneField = new JTextField(20);
add(donorPhoneLabel);
add(donorPhoneField);
JLabel donorAddressLabel = new JLabel("Address:");
donorAddressField = new JTextField(20);
add(donorAddressLabel);
add(donorAddressField);
JLabel donorNgoIdLabel = new JLabel("NGO Mem ID:");
donorNgoIdField = new JTextField(20);
add(donorNgoIdLabel);
```

```
    add(donorNgoIdField);

    donorInsertButton = new JButton("Insert Donor");
    donorInsertButton.addActionListener(this);
    add(donorInsertButton);

    setSize(400, 400);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setVisible(true);
}

public void actionPerformed(ActionEvent ae) {
    try {
        String donorAadhar = donorAadharField.getText();
        String donorName = donorNameField.getText();
        int donorAge = Integer.parseInt(donorAgeField.getText());
        String donorDob = donorDobField.getText();
        String donorOrgans = donorOrgansField.getText();
        String donorPhone = donorPhoneField.getText();
        String donorAddress = donorAddressField.getText();
        String donorNgoId = donorNgoIdField.getText();

        String query = "INSERT INTO healthy_donor(aadhar_no, name, age, dob, organs,
```

```
phone_no, address, ngo_mem_id) VALUES ('" + donorAadhar + "', '" + donorName + "', " +
donorAge + ", '" + donorDob + "', '" + donorOrgans + "', '" + donorPhone + "', '" + donorAddress
+ "', '" + donorNgoId + "')";
                stmt.executeUpdate(query);
                outputArea.append("Healthy Donor record inserted successfully.\n");
            } catch (Exception e) {
                System.out.println(e);
            }
        }
    }

    class RecipientWindow extends JFrame implements ActionListener {
        JTextField recipientAadharField, recipientNameField, recipientOrgansField,
recipientNgoIdField;
        JButton recipientInsertButton;

        public RecipientWindow() {
            super("Recipient");
            setLayout(new GridLayout(5, 1));

            JLabel recipientAadharLabel = new JLabel("Aadhar No:");
            recipientAadharField = new JTextField(20);
            add(recipientAadharLabel);
```

```
    add(recipientAadharField);
    JLabel recipientNameLabel = new JLabel("Name:");
    recipientNameField = new JTextField(20);
    add(recipientNameLabel);
    add(recipientNameField);
    JLabel recipientOrgansLabel = new JLabel("Organs:");
    recipientOrgansField = new JTextField(20);
    add(recipientOrgansLabel);
    add(recipientOrgansField);
    JLabel recipientNgoIdLabel = new JLabel("NGO Mem ID:");
    recipientNgoIdField = new JTextField(20);
    add(recipientNgoIdLabel);
    add(recipientNgoIdField);

    recipientInsertButton = new JButton("Insert Recipient");
    recipientInsertButton.addActionListener(this);
    add(recipientInsertButton);

    setSize(400, 300);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setVisible(true);
}
```

```
        public void actionPerformed(ActionEvent ae) {
            try {
                String recipientAadhar = recipientAadharField.getText();
                String recipientName = recipientNameField.getText();
                String recipientOrgans = recipientOrgansField.getText();
                String recipientNgoId = recipientNgoIdField.getText();

                String query = "INSERT INTO recipient(aadhar_no, name, organs, ngo_mem_id)
VALUES ('" + recipientAadhar + "', '" + recipientName + "', '" + recipientOrgans + "', '" +
recipientNgoId + "')";
                stmt.executeUpdate(query);
                outputArea.append("Recipient record inserted successfully.\n");
            } catch (Exception e) {
                System.out.println(e);
            }
        }
    }

    class DoctorsWindow extends JFrame implements ActionListener {
        JTextField lidField, empIdField, hlidField;
        JButton insertButton;

        public DoctorsWindow() {
```

```
super("Doctors");
setLayout(new GridLayout(4, 1));

JLabel lidLabel = new JLabel("LID:");
lidField = new JTextField(20);
add(lidLabel);
add(lidField);
JLabel empIdLabel = new JLabel("EMP_ID:");
empIdField = new JTextField(20);
add(empIdLabel);
add(empIdField);
JLabel hlidLabel = new JLabel("HLID:");
hlidField = new JTextField(20);
add(hlidLabel);
add(hlidField);

insertButton = new JButton("Insert Doctor");
insertButton.addActionListener(this);
add(insertButton);

setSize(400, 300);
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
setVisible(true);
```

```
        }

        public void actionPerformed(ActionEvent ae) {
            try {
                String lid = lidField.getText();
                String empId = empIdField.getText();
                String hlid = hlidField.getText();

                String query = "INSERT INTO doctors(lid, emp_id, hlid) VALUES ('" + lid + "', '" +
empId + "', '" + hlid + "')";
                stmt.executeUpdate(query);
                outputArea.append("Doctor record inserted successfully.\n");
            } catch (Exception e) {
                System.out.println(e);
            }
        }
    }

    class HospitalWindow extends JFrame implements ActionListener {
        JTextField hlidField, phoneField, locationField;
        JButton insertButton;

        public HospitalWindow() {
```

```
super("Hospital");
setLayout(new GridLayout(4, 1));

JLabel hlidLabel = new JLabel("HLID:");
hlidField = new JTextField(20);
add(hlidLabel);
add(hlidField);
JLabel phoneLabel = new JLabel("Phone:");
phoneField = new JTextField(20);
add(phoneLabel);
add(phoneField);
JLabel locationLabel = new JLabel("Location:");
locationField = new JTextField(20);
add(locationLabel);
add(locationField);

insertButton = new JButton("Insert Hospital");
insertButton.addActionListener(this);
add(insertButton);

setSize(400, 300);
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
setVisible(true);
```

```
        }

        public void actionPerformed(ActionEvent ae) {
            try {
                String hlid = hlidField.getText();
                String phone = phoneField.getText();
                String location = locationField.getText();

                String query = "INSERT INTO hospital(hlid, ph_no, location) VALUES ('" + hlid + "',
'" + phone + "', '" + location + "')";
                stmt.executeUpdate(query);
                outputArea.append("Hospital record inserted successfully.\n");
            } catch (Exception e) {
                System.out.println(e);
            }
        }
    }

    class NursesWindow extends JFrame implements ActionListener {
        JTextField empIdField, hlidField;
        JButton insertButton;

        public NursesWindow() {
```

```
    super("Nurses");
    setLayout(new GridLayout(3, 1));

    JLabel empIdLabel = new JLabel("Emp ID:");
    empIdField = new JTextField(20);
    add(empIdLabel);
    add(empIdField);
    JLabel hlidLabel = new JLabel("HLID:");
    hlidField = new JTextField(20);
    add(hlidLabel);
    add(hlidField);

    insertButton = new JButton("Insert Nurse");
    insertButton.addActionListener(this);
    add(insertButton);

    setSize(400, 300);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setVisible(true);
}

public void actionPerformed(ActionEvent ae) {
    try {
```

```
            String empId = empIdField.getText();
            String hlid = hlidField.getText();

            String query = "INSERT INTO nurses(emp_id, hlid) VALUES ('" + empId + "', '" +
hlid + "')";
            stmt.executeUpdate(query);
            outputArea.append("Nurse record inserted successfully.\n");
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}


public static void main(String[] args) {
    new OrganTransplantationDatabase();
}

public void stop() {
    try {
        con.close();
    } catch (SQLException e) {
        System.out.println(e);
```

```
        }
    }

    public void displayRecords() {
        outputArea.setText(""); // Clear the output area
        try {
            ResultSet rs = stmt.executeQuery("SELECT * FROM healthy_donor");
            outputArea.append("Healthy Donors:\n");
            while (rs.next()) {
                outputArea.append("Aadhar No: " + rs.getString("aadhar_no") + ", Name: " +
rs.getString("name") + ", Age: " + rs.getInt("age") + ", DoB: " + rs.getString("dob") + ", Organs:
" + rs.getString("organs") + ", Phone No: " + rs.getString("phone_no") + ", Address: " +
rs.getString("address") + ", NGO Mem ID: " + rs.getString("ngo_mem_id") + "\n");
            }

            rs = stmt.executeQuery("SELECT * FROM recipient");
            outputArea.append("\nRecipients:\n");
            while (rs.next()) {
                outputArea.append("Aadhar No: " + rs.getString("aadhar_no") + ", Name: " +
rs.getString("name") + ", Organs: " + rs.getString("organs") + ", NGO Mem ID: " +
rs.getString("ngo_mem_id") + "\n");
            }
```

```
            rs = stmt.executeQuery("SELECT * FROM doctors");
            outputArea.append("\nDoctors:\n");
            while (rs.next()) {
                outputArea.append("LID: " + rs.getString("lid") + ", EMP ID: " +
rs.getString("emp_id") + ", HLID: " + rs.getString("HLID") + "\n");
            }

            rs = stmt.executeQuery("SELECT * FROM hospital");
            outputArea.append("\nHospital:\n");
            while (rs.next()) {
                outputArea.append("HLID: " + rs.getString("hlid") + ", Phone No: " +
rs.getString("ph_no") + ", Location: " + rs.getString("location") + "\n");
            }

            rs = stmt.executeQuery("SELECT * FROM nurses");
            outputArea.append("\nNurses:\n");
            while (rs.next()) {
                outputArea.append("EMP ID: " + rs.getString("emp_id") + ", HLID: " +
rs.getString("hlid") + "\n");
            }

        } catch (SQLException e) {
            System.out.println(e)}}}
```

CHAPTER 7

RESULT:

Fig 7.1

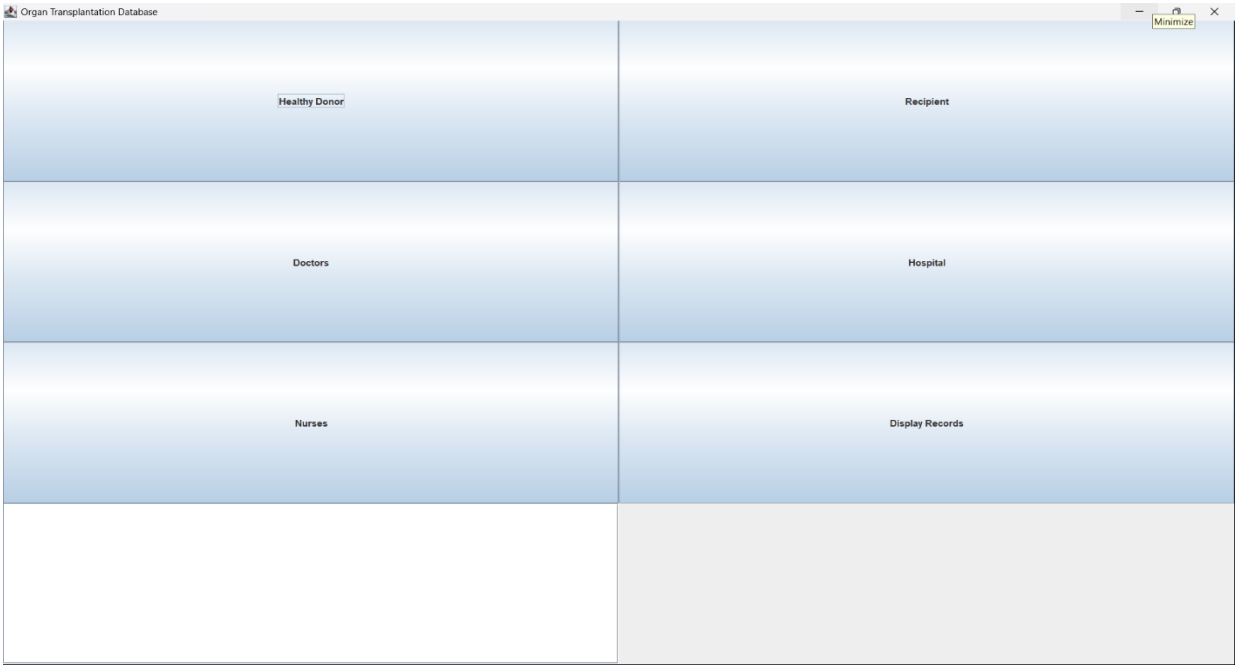


Fig 7.2

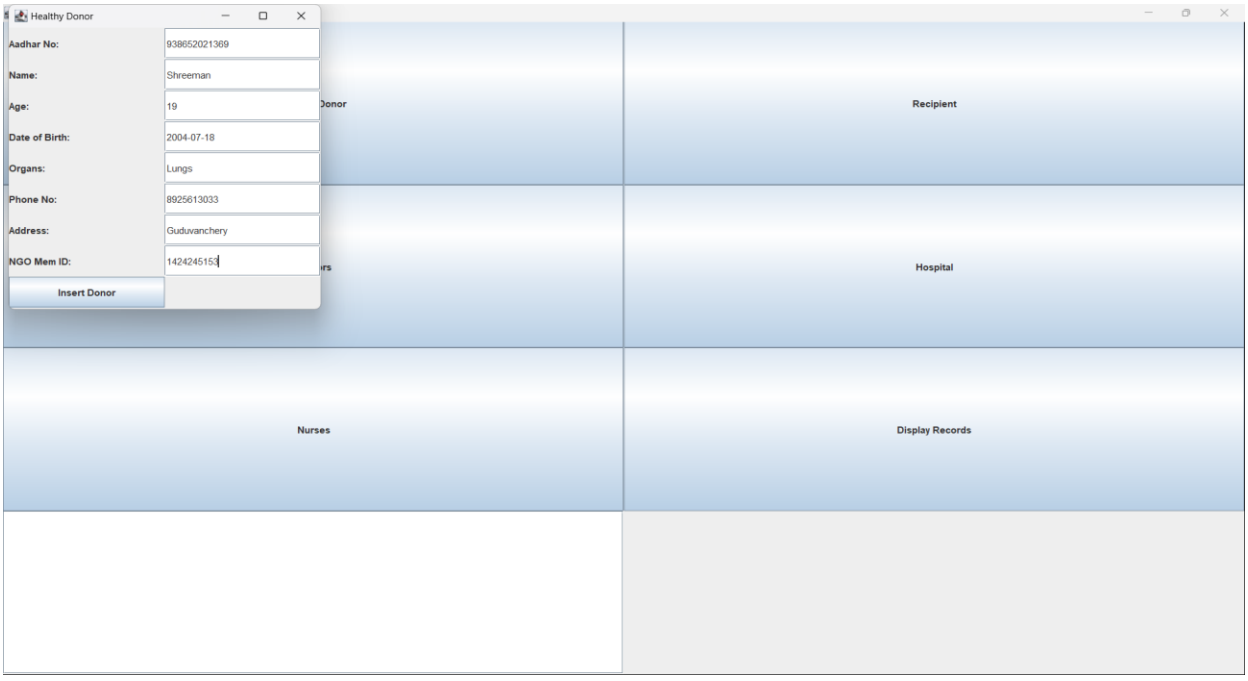


Fig 7.3

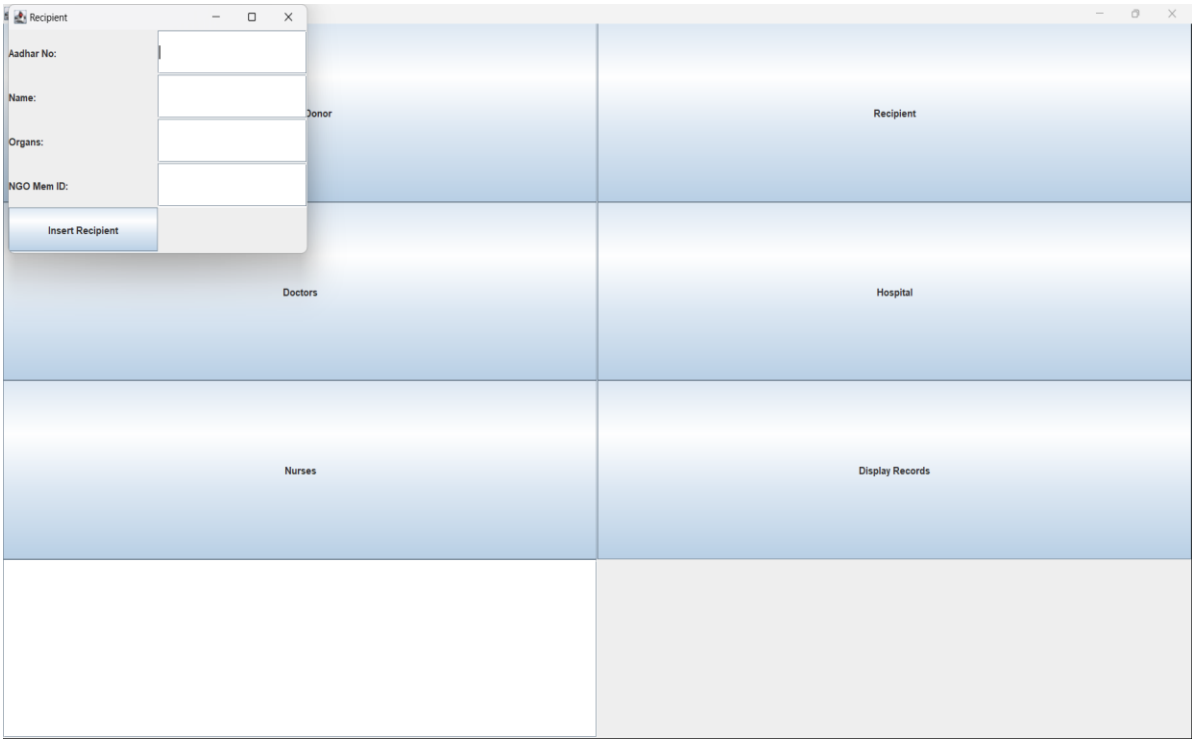

Recipient
Aadhar No:
Name:
Organs:
NGO Mem ID:
Insert Recipient
Donor
Recipient
Doctors
Hospital
Nurses
Display Records

Fig 7.4

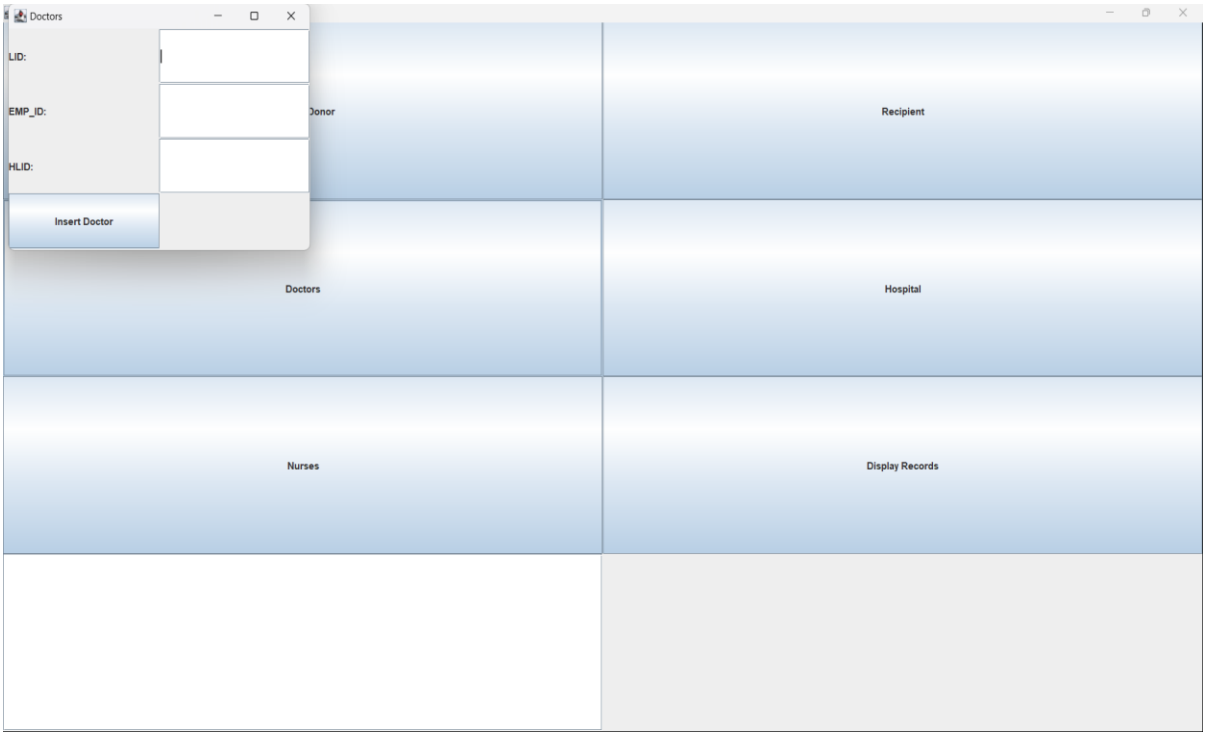

Doctors
LID:
EMP_ID:
HLID:
Insert Doctor
Donor
Recipient
Doctors
Hospital
Nurses
Display Records

Fig 7.5

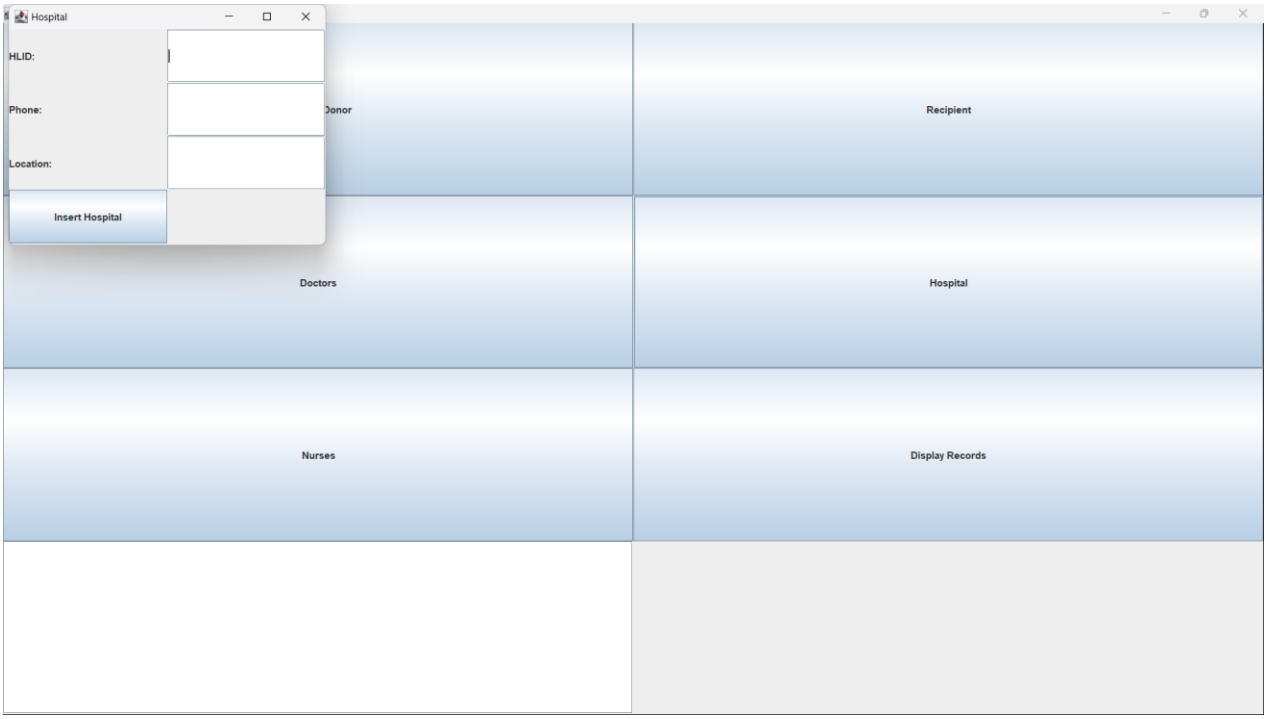

Hospital
HLID:
Phone:
Location:
Insert Hospital
Donor
Recipient
Doctors
Hospital
Nurses
Display Records

Fig 7.6

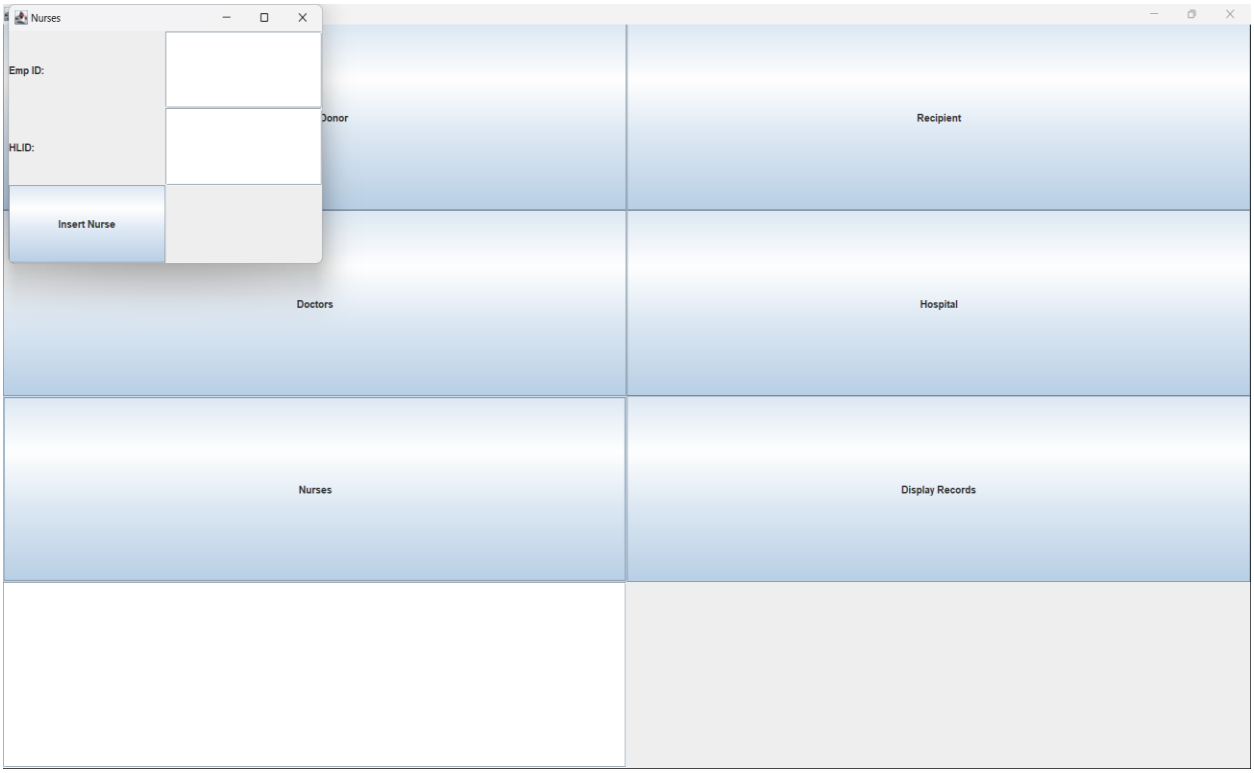

Nurses
Emp ID:
HLID:
Insert Nurse
Donor
Recipient
Doctors
Hospital
Nurses
Display Records

Fig 7.7

Figure 7.1 shows the User Interface of the Organ Transplantation Database System application. The opening window has 5 buttons to select various tables, one button to display data in database and one display section to see the output of the desired input. Each button opens a new window which allows the user to input data into the database.

Figures 7.2 to 7.6 shows the various windows the buttons open. These windows have text input boxes where the user can input database values and a button to insert the input data into the database

Figure 7.7 finally shows the data that has been inserted into the database. The “Display Records” button will extract data from the database and insert it into the display section.

CONCLUSION

The development of an Organ Transplantation Database System represents a critical step forward in optimizing the complex and multifaceted processes involved in organ transplantation. Through a meticulous blend of traditional and modern design methodologies, this project endeavours to create a robust, user-centric platform that streamlines every aspect of the transplantation journey, from donor registration to transplantation. The adoption of modern design principles emphasizes the importance of usability, intuitiveness, and adaptability in the database system. By engaging directly with stakeholders and incorporating their feedback throughout the design and development process, the system is tailored to meet the diverse needs and preferences of healthcare professionals, patients, and organ procurement organizations.

In conclusion, the Organ Transplantation Database System represents a significant advancement in the field of organ transplantation, promising to improve patient outcomes, enhance healthcare efficiency, and ultimately save lives.

CHAPTER 8

REAL TIME PROJECT CERTIFICATES

1. SHREEMAN M (RA2211028010166)

CERTIFICATE
OF EXCELLENCE

THIS CERTIFICATE IS AWARDED TO

SCALER Topics

SHREEMAN MURUGANANDAM

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

74 Video Tutorials 16 Modules 16 Challenges 16 March 2024

Anshuman Singh

Co-founder SCALER

CERTIFICATE OF EXCELLENCE BY SCALER

2. M KRISHNA CHAITANYA (RA2211028010165)

CERTIFICATE
OF EXCELLENCE

THIS CERTIFICATE IS AWARDED TO

SCALER Topics

KRISHNA CHAITANAYA MUTTEVI

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

74 Video Tutorials 16 Modules 16 Challenges

12 March 2024

Anshuman Singh
Co-founder SCALER

CERTIFICATE OF EXCELLENCE BY SCALER

3. S CHARITH (RA22110280197)

CERTIFICATE
OF EXCELLENCE

THIS CERTIFICATE IS AWARDED TO

SCALER Topics

chairth .S

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

74 Video Tutorials 16 Modules 16 Challenges 13 March 2024

Anshuman Singh

Co-founder SCALER

CERTIFICATE OF EXCELLENCE BY SCALER