# ORGAN DONATION & PROCUREMENT NETWORK MANAGEMENT SYSTEM

DBMS PROJECT REPORT

By

**ELURI KRISHNA SAI REVANTH (RA2311030010340)**

**KVVS RAJAMOULI (RA2311030010337)**

*Under the Guidance of*

**Dr. B. BALAKIRUTHIGA**

Assistant Professor,Department of Networking and Communications

in partial fulfillment of the requirements for the degree of

**BACHELOR OF TECHNOLOGY**
*in*
**COMPUTER SCIENCE ENGINEERING WITH SPECIALIZATION IN CYBER SECURITY**

**DEPARTMENT OF NETWORKING AND COMMUNICATIONS**
**COLLEGE OF ENGINEERING AND TECHNOLOGY**
**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**
**KATTANKULATHUR– 603 203**
**MAY 2025**

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

# SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

## KATTANKULATHUR 603203

## BONAFIDE CERTIFICATE

**Register no. RA2311030010340** and **RA2311030010337** Certified to be the bonafide work done by **ELURI KRISHNA SAI REVANTH** and **KVVS RAJAMOULI** of II year/IV sem 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.

Dept of Networking and Communications
Kattankulathur
603 203
SRM IST

**Date:** 5/5/25

**Faculty in Charge**
Dr.B.Balakiruthiga
Assistant Professor
Department of NWC
SRMIST-KTR

**HEAD OF THE DEPARTMENT**
Dr. M.Lakshmi
Professor & Head
Department of NWC
SRMIST-KTR

2

# ABSTRACT

The Organ Donation and Procurement Network System is a centralized digital platform designed to streamline the complex processes involved in organ donation, allocation, and transplantation. This system facilitates real-time coordination among hospitals, organ banks, transplant centres, and donors, ensuring transparency, traceability, and efficiency in the organ procurement chain. By automating donor registration, medical evaluation, organ matching, and logistics management, the platform minimizes delays and human errors, thereby increasing the success rate of transplants.

Built with a user-friendly interface, the system supports secure data management, prioritization algorithms based on medical urgency and compatibility, and notification services to keep stakeholders informed throughout the donation process. It also ensures regulatory compliance and ethical standards by maintaining accurate records and enabling audits.

Overall, the Organ Donation and Procurement Network System aims to save lives by reducing organ wastage, improving donor-recipient matching, and fostering a culture of timely and ethical organ donation through technological innovation.

# TABLE OF CONTENTS

## Problem Statement:

The current organ donation and procurement process suffers from significant inefficiencies and lacks a unified digital infrastructure to manage donor registration, organ matching, logistics coordination, and real-time updates among hospitals, organ banks, and recipients. This fragmented system leads to delays in transplantation, miscommunication between stakeholders, loss of viable organs, and reduced trust in the process. Therefore, a centralized, secure, and user-friendly system is essential to streamline operations, enhance transparency, and improve the success rate of organ transplants.

### Problem Description:

1. **Manual and Paper-Based Processes:** Donor registration, organ availability tracking, and recipient matching are often performed manually, resulting in delayed processing and higher chances of error.
2. **Lack of Real-Time Coordination:** Hospitals and organ banks operate in silos, leading to slow communication and coordination, which can critically affect organ viability and transplant outcomes.
3. **Inconsistent Data Management:** Absence of a centralized system causes discrepancies in donor and recipient data, complicating organ matching and leading to potential mismatches.
4. **Limited Transparency and Traceability:** Stakeholders have minimal visibility into the organ procurement and allocation process, leading to mistrust and potential ethical concerns.
5. **Inefficient Logistics and Transportation Tracking:** Organ transport logistics are not always properly coordinated or tracked in real time, resulting in delays and potential wastage of organs.
6. **Security and Privacy Risks:** Sensitive personal and medical information of donors and recipients is vulnerable in unprotected systems, posing risks to data privacy and compliance with regulations.
7. **Poor User Experience:** Existing systems (if any) may not offer intuitive interfaces for different users (e.g., doctors, coordinators, donors), making it difficult to efficiently navigate and perform critical tasks.
8. **Limited Scalability and Integration:** Current solutions cannot scale effectively with increasing data and user demands, and often lack integration with national or regional health networks and databases.

ER-DIAGRAM FOR ORGAN DONATION & PROCUREMENT NETWORK MANAGEMENT SYSTEM:

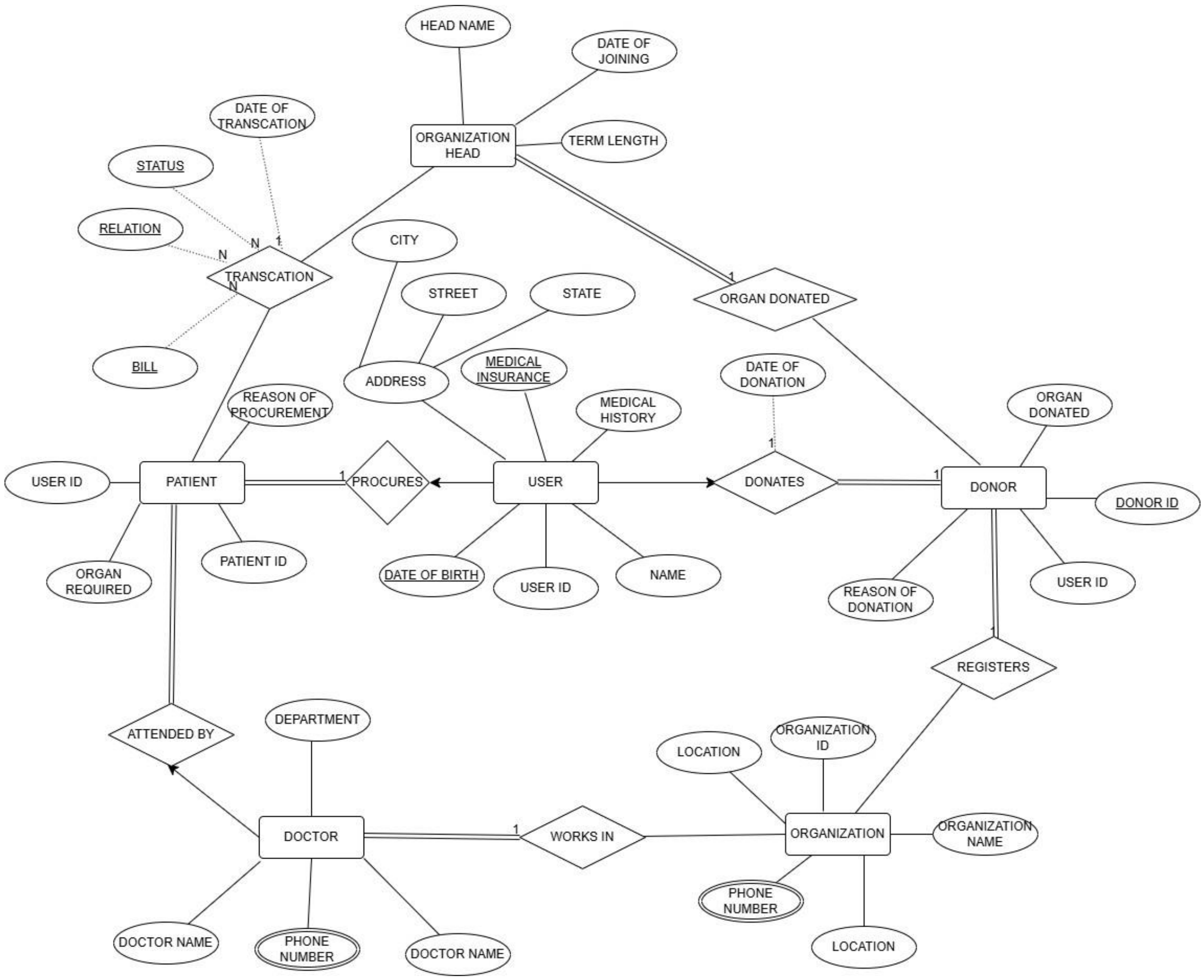


Fig:1.1 ER DIAGRAM

## Design of Relational Schemas, Creation of Database Tables for the Organ Donation and Procurement Network System

The ER (Entity Relationship) diagram presented for the **Organ Donation Management System** visually depicts how data flows between entities, the relationships between them, and the attributes used to manage an organized and effective donor-patient tracking system. This system helps ensure the ethical and efficient coordination between patients in need of organs, donors, hospitals, and medical professionals.

### Entities and Their Roles

#### 1. User

- **Attributes:**
  - USER ID (Primary Key)
  - NAME
  - DATE OF BIRTH
  - MEDICAL HISTORY
  - MEDICAL INSURANCE
  - ADDRESS (linked with Street, City, State)
- **Relationships:**
  - Linked to **Patient** via PROCURES relationship.
  - Linked to **Donor** via DONATES relationship.
  - Connected to **Transaction** for financial/administrative tracking.

#### 2. Patient

- **Attributes:**
  - PATIENT ID (Primary Key)
  - ORGAN REQUIRED
  - USER ID (Foreign Key)
- **Relationships:**
  - Linked to **User**, meaning a patient is a user who requires an organ.
  - Connected to **Doctor** via the ATTENDED BY relationship (Many-to-One).
  - Participates in **Transaction** (donation or billing).
  - Has a REASON OF PROCUREMENT attribute (via relationship).

#### 3. Doctor

- **Attributes:**
    - DOCTOR NAME
    - DEPARTMENT
    - PHONE NUMBER
- **Relationships:**
    - Connected to **Patient** through ATTENDED BY.
    - Linked to **Organization** via WORKS IN relationship (Many-to-One).

4. **Donor**

- **Attributes:**
    - DONOR ID (Primary Key)
    - USER ID (Foreign Key)
    - ORGAN DONATED
- **Relationships:**
    - Linked to **User** as a specialized role.
    - Connected to **Organization** via the REGISTERS relationship.
    - Associated with **Organ Donated** via the DONATES relationship.
    - Includes REASON OF DONATION and DATE OF DONATION.

5. **Organization**

- **Attributes:**
    - ORGANIZATION ID
    - ORGANIZATION NAME
    - PHONE NUMBER
    - LOCATION
- **Relationships:**
    - Connected to **Doctor** via WORKS IN.
    - Receives **Donor** registrations.
    - Linked to **Organization Head** via a HAS relationship.

6. **Organization Head**

- **Attributes:**
    - HEAD NAME
    - DATE OF JOINING
    - TERM LENGTH

- **Relationships:**
    - Heads one **Organization** (One-to-One).

### 7. Transaction

- **Attributes:**
    - DATE OF TRANSACTION
    - STATUS
    - RELATION
    - BILL
- **Relationships:**
    - Acts as an intermediary involving **Patient** and **User** (One-to-Many/Many-to-One).

### Key Relationships and Multiplicities

- **User to Patient**: One-to-One (a user becomes a patient when they require an organ).
- **User to Donor**: One-to-One (a user becomes a donor by donating).
- **Patient to Doctor**: Many-to-One (multiple patients attended by one doctor).
- **Doctor to Organization**: Many-to-One (multiple doctors work in one organization).
- **Organization to Organization Head**: One-to-One.
- **User to Transaction**: One-to-Many (a user can have multiple transactions).
- **Patient to Transaction**: One-to-Many (each patient involved in multiple transactions).
- **Donor to Organization**: Many-to-One (many donors register at one organization).

### Summary

This ER diagram for the **Organ Donation System** captures a robust and traceable approach to tracking both donors and recipients. It accounts for all medical, administrative, and ethical considerations by connecting users, patients, donors, medical staff, and organizational authorities. Relationships such as **DONATES**, **PROCURES**, and **ATTENDED BY** ensure the logical flow of responsibilities and data. This system structure promotes efficient management and transparency in life-saving organ transplants.

## RELATION - SCHEMA

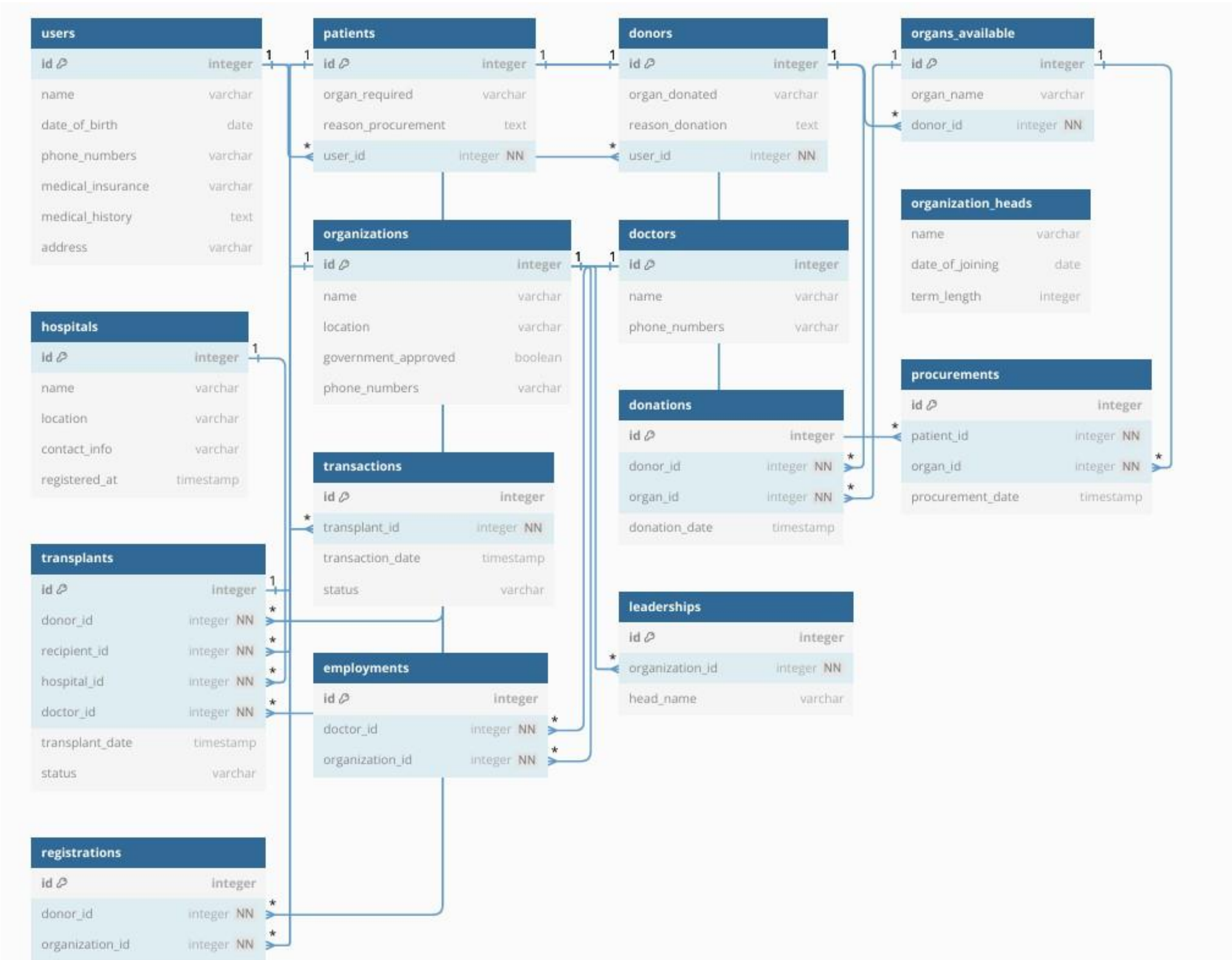


FIG 2.1 SCHEMA DIAGRAM

## Complex queries based on the concepts of constraints, sets, joins, views, Triggers and Cursors.

Creation of the SQL data base using the commands

CREATING A DATABASE Organ Donation System

-- Create Database
CREATE DATABASE IF NOT EXISTS DBMS_PROJECT;

USE

DBMS_PROJECT;

-- Creating User Table

CREATE TABLE User (

UserID INT AUTO_INCREMENT PRIMARY KEY,

Name VARCHAR(255) NOT NULL,

DateOfBirth DATE NOT NULL,

Address VARCHAR(255),

City VARCHAR(100),

State VARCHAR(100),

MedicalInsurance ENUM('Yes', 'No'),

```
    MedicalHistory TEXT
);
-- Creating Organization Table
CREATE TABLE Organization (
    OrganizationID INT
AUTO_INCREMENT PRIMARY KEY,
    OrganizationName VARCHAR(255) NOT
NULL,
    Location VARCHAR(255),
    PhoneNumber VARCHAR(20)
);


-- Creating OrganizationHead Table
CREATE TABLE OrganizationHead (
    HeadID INT AUTO_INCREMENT
PRIMARY KEY,
    HeadName VARCHAR(255) NOT
NULL,
    DateOfJoining DATE NOT NULL,
```

```
    TermLength INT,
    OrganizationID INT,
    FOREIGN KEY (OrganizationID)
REFERENCES
Organization(OrganizationID) ON DELETE
CASCADE
);


-- Creating Patient Table
CREATE TABLE Patient (
    PatientID INT AUTO_INCREMENT
PRIMARY KEY,
    UserID INT NOT NULL,
    OrganRequired VARCHAR(255) NOT
NULL,
    ReasonOfProcurement TEXT,
    FOREIGN KEY (UserID) REFERENCES
User(UserID) ON DELETE CASCADE
);
```

```
-- Creating Donor Table
CREATE TABLE Donor (
    DonorID INT AUTO_INCREMENT
PRIMARY KEY,
    UserID INT NOT NULL,
    OrganDonated VARCHAR(255) NOT
NULL,
    DateOfDonation DATE NOT NULL,
    ReasonOfDonation TEXT,
    FOREIGN KEY (UserID) REFERENCES
User(UserID) ON DELETE CASCADE
);


-- Creating Doctor Table
CREATE TABLE Doctor (
    DoctorID INT AUTO_INCREMENT
PRIMARY KEY,
    DoctorName VARCHAR(255) NOT
NULL,
    PhoneNumber VARCHAR(20),
```

```
    Department VARCHAR(255)
);

-- Creating Transaction Table
CREATE TABLE Transaction (
    TransactionID INT AUTO_INCREMENT PRIMARY KEY,
    UserID INT NOT NULL,
    Status ENUM('Completed', 'Pending', 'Failed'),
    Relation VARCHAR(255),
    DateOfTransaction DATE NOT NULL,
    Bill INT,
    FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE
);

-- Inserting data into User Table
INSERT INTO User (Name, DateOfBirth, Address, City, State, MedicalInsurance,
```

MedicalHistory) VALUES

('Alice Johnson', '1990-03-10', '123 Main St',

'New York', 'NY', 'Yes', 'Diabetes'),

('Bob Smith', '1985-07-25', '456 Elm St', 'Los

Angeles', 'CA', 'No', 'Hypertension'),

('Charlie Brown', '1995-12-05', '789 Oak St',

'Chicago', 'IL', 'Yes', 'Asthma'),

('David Wilson', '1988-09-14', '101 Pine St',

'Houston', 'TX', 'No', 'Healthy'),

('Eve Davis', '1992-06-30', '202 Maple St',

'San Francisco', 'CA', 'Yes', 'Anemia');

```
mysql> -- Display all tables
mysql> SELECT * FROM User;
+--------+---------------+-------------+--------------+---------------+-------+------------------+----------------+
| UserID | Name          | DateOfBirth | Address      | City          | State | MedicalInsurance | MedicalHistory |
+--------+---------------+-------------+--------------+---------------+-------+------------------+----------------+
|      1 | Alice Johnson | 1990-03-10  | 123 Main St  | New York      | NY    | Yes              | Diabetes       |
|      2 | Bob Smith     | 1985-07-25  | 456 Elm St   | Los Angeles   | CA    | No               | Hypertension   |
|      3 | Charlie Brown | 1995-12-05  | 789 Oak St   | Chicago       | IL    | Yes              | Asthma         |
|      4 | David Wilson  | 1988-09-14  | 101 Pine St  | Houston       | TX    | No               | Healthy        |
|      5 | Eve Davis     | 1992-06-30  | 202 Maple St | San Francisco | CA    | Yes              | Anemia         |
+--------+---------------+-------------+--------------+---------------+-------+------------------+----------------+
5 rows in set (0.00 sec)
```

Fig:3.1

-- Inserting data into Organization Table

INSERT INTO Organization

(OrganizationName, Location,

PhoneNumber) VALUES

('HealthCare Org', 'New York',

'1234567890'),

('LifeLine Hospital', 'Los Angeles',

'9876543210'),

('MediPlus', 'Chicago', '1122334455'),
('Hope Foundation', 'Houston', '5566778899'),

('Global Health', 'San Francisco',

'2233445566');

mysql> SELECT * FROM Organization;

| OrganizationID | OrganizationName | Location | PhoneNumber |
|---|---|---|---|
| 1 | HealthCare Org | New York | 1234567890 |
| 2 | LifeLine Hospital | Los Angeles | 9876543210 |
| 3 | MediPlus | Chicago | 1122334455 |
| 4 | Hope Foundation | Houston | 5566778899 |
| 5 | Global Health | San Francisco | 2233445566 |

Fig:3.2

-- Inserting data into OrganizationHead

Table

INSERT INTO OrganizationHead

(HeadName, DateOfJoining, TermLength,

OrganizationID) VALUES

('Dr. Smith', '2015-06-15', 10, 1),

('Dr. Johnson', '2017-09-23', 8, 2),

('Dr. Williams', '2016-02-10', 12, 3),

('Dr. Brown', '2018-11-05', 9, 4),

('Dr. Miller', '2019-04-21', 7, 5);

```
mysql> SELECT * FROM OrganizationHead;
+--------+--------------+---------------+------------+----------------+
| HeadID | HeadName     | DateOfJoining | TermLength | OrganizationID |
+--------+--------------+---------------+------------+----------------+
|      1 | Dr. Smith    | 2015-06-15    |         10 |              1 |
|      2 | Dr. Johnson  | 2017-09-23    |          8 |              2 |
|      3 | Dr. Williams | 2016-02-10    |         12 |              3 |
|      4 | Dr. Brown    | 2018-11-05    |          9 |              4 |
|      5 | Dr. Miller   | 2019-04-21    |          7 |              5 |
+--------+--------------+---------------+------------+----------------+
5 rows in set (0.00 sec)
```

Fig : 3.3

-- Inserting data into Patient Table

INSERT INTO Patient (UserID,

OrganRequired, ReasonOfProcurement)

VALUES

(1, 'Kidney', 'Chronic Kidney Disease'),

(2, 'Liver', 'Liver Cirrhosis'),

(3, 'Heart', 'Congenital Heart Disease'),

(4, 'Lung', 'Pulmonary Fibrosis'),

(5, 'Pancreas', 'Diabetes');

```
mysql> SELECT * FROM Patient;
+-----------+--------+---------------+--------------------------+
| PatientID | UserID | OrganRequired | ReasonOfProcurement      |
+-----------+--------+---------------+--------------------------+
|         1 |      1 | Kidney        | Chronic Kidney Disease   |
|         2 |      2 | Liver         | Liver Cirrhosis          |
|         3 |      3 | Heart         | Congenital Heart Disease |
|         4 |      4 | Lung          | Pulmonary Fibrosis       |
|         5 |      5 | Pancreas      | Diabetes                 |
+-----------+--------+---------------+--------------------------+
5 rows in set (0.00 sec)
```

Fig:3.4

-- Inserting data into Donor Table

INSERT INTO Donor (UserID,

OrganDonated, DateOfDonation,

ReasonOfDonation) VALUES

(2, 'Kidney', '2024-01-10', 'Altruistic

donation'),

(3, 'Liver', '2024-02-15', 'Family donation'),

(4, 'Heart', '2024-03-20', 'Deceased

donation'),

(5, 'Lung', '2024-04-25', 'Voluntary

donation'),

(1, 'Pancreas', '2024-05-30', 'Charity

donation');

mysql> SELECT * FROM Donor;

| DonorID | UserID | OrganDonated | DateOfDonation | ReasonOfDonation |
|---|---|---|---|---|
| 1 | 2 | Kidney | 2024-01-10 | Altruistic donation |
| 2 | 3 | Liver | 2024-02-15 | Family donation |
| 3 | 4 | Heart | 2024-03-20 | Deceased donation |
| 4 | 5 | Lung | 2024-04-25 | Voluntary donation |
| 5 | 1 | Pancreas | 2024-05-30 | Charity donation |

5 rows in set (0.00 sec)

Fig:3.5

-- Inserting data into Doctor Table

INSERT INTO Doctor (DoctorName,

PhoneNumber, Department) VALUES

('Dr. Green', '1234567890', 'Nephrology'),

('Dr. Black', '0987654321', 'Hepatology'),

('Dr. White', '1122334455', 'Cardiology'),

('Dr. Blue', '5566778899', 'Pulmonology'),

('Dr. Yellow', '2233445566',

'Endocrinology');

mysql> SELECT * FROM Doctor;

| DoctorID | DoctorName | PhoneNumber | Department |
|---|---|---|---|
| 1 | Dr. Green | 1234567890 | Nephrology |
| 2 | Dr. Black | 0987654321 | Hepatology |
| 3 | Dr. White | 1122334455 | Cardiology |
| 4 | Dr. Blue | 5566778899 | Pulmonology |
| 5 | Dr. Yellow | 2233445566 | Endocrinology |

5 rows in set (0.00 sec)

Fig:3.6

-- Inserting data into Transaction Table

INSERT INTO Transaction (UserID, Status,
Relation, DateOfTransaction, Bill) VALUES

(1, 'Pending', 'Friend', '2024-04-07', 1000),

(2, 'Pending', 'Spouse', '2024-03-02', 7500),

(3, 'Completed', 'Sibling', '2024-03-03',
6200),

(4, 'Failed', 'Friend', '2024-03-04',4500),

(5, 'Completed', 'Parent', '2024-03-05', 4300);

```
mysql> SELECT * FROM Transaction;
+---------------+--------+-----------+----------+-------------------+------+
| TransactionID | UserID | Status    | Relation | DateOfTransaction | Bill |
+---------------+--------+-----------+----------+-------------------+------+
|             1 |      1 | Pending   | Friend   | 2024-04-07        | 1000 |
|             2 |      2 | Pending   | Spouse   | 2024-03-02        | 7500 |
|             3 |      3 | Completed | Sibling  | 2024-03-03        | 6200 |
|             4 |      4 | Failed    | Friend   | 2024-03-04        | 4500 |
|             5 |      5 | Completed | Parent   | 2024-03-05        | 4300 |
+---------------+--------+-----------+----------+-------------------+------+
5 rows in set (0.00 sec)
```

Fig:3.7

## 1. Constraints

Constraint on Transaction table:

ALTER TABLE Transaction

ADD CONSTRAINT chk_bill_positive

CHECK (Bill >= 0);

Ensures: No record can have Bill < 0

Constraint on Donor table:

ALTER TABLE Donor

MODIFY OrganDonated VARCHAR(255)

NOT NULL;

Ensures: OrganDonated column can't have

NULL values

## 2. Set Operation

**Query:**

SELECT U.Name FROM User U

JOIN Patient P ON U.UserID = P.UserID

UNION

SELECT U.Name FROM User U

JOIN Donor D ON U.UserID = D.UserID;

**Result Table:**

| Name |
| --- |
| Alice Johnson |
| Bob Smith |
| Charlie Brown |
| David Wilson |
| Eve Davis |

Fig:3.8

UNION removes duplicates; every user who is either a Patient or Donor.

## 3. Join Query (Patient–Doctor–User)

**Setup (intermediate table):**

```
CREATE TABLE PatientDoctor (
    PatientID INT,
    DoctorID INT,
    FOREIGN KEY (PatientID)
REFERENCES Patient(PatientID),
    FOREIGN KEY (DoctorID)
REFERENCES Doctor(DoctorID)
);
```

**Sample Data:**

INSERT INTO PatientDoctor VALUES

(1, 1), -- Alice → Dr. Green

(2, 2), -- Bob → Dr. Black

(3, 3), -- Charlie → Dr. White

(4, 4), -- David → Dr. Blue

(5, 5); -- Eve → Dr. Yellow

**Join Query:**

SELECT

U.Name AS PatientName,

P.OrganRequired,

D.DoctorName,

D.Department

FROM

Patient P

JOIN User U ON P.UserID = U.UserID

JOIN PatientDoctor PD ON P.PatientID = PD.PatientID

JOIN Doctor D ON PD.DoctorID = D.DoctorID;

**Result Table:**

| PatientName | OrganRequired | DoctorName | Department |
|---|---|---|---|
| Alice Johnson | Kidney | Dr. Green | Nephrology |
| Bob Smith | Liver | Dr. Black | Hepatology |
| Charlie Brown | Heart | Dr. White | Cardiology |
| David Wilson | Lung | Dr. Blue | Pulmonology |
| Eve Davis | Pancreas | Dr. Yellow | Endocrinology |

Fig:3.9

## 4. View: DonorView

**View Definition:**

```
CREATE VIEW DonorView AS
SELECT
    U.Name,
    D.OrganDonated,
    D.DateOfDonation,
    D.ReasonOfDonation
FROM
    Donor D
JOIN User U ON D.UserID = U.UserID;
```

**View Output:**

```
SELECT * FROM DonorView;
```

**Result Table:**

| Name | OrganDonated | DateOfDonation | ReasonOfDonation |
|---|---|---|---|
| Bob Smith | Kidney | 2024-01-10 | Altruistic donation |
| Charlie Brown | Liver | 2024-02-15 | Family donation |
| David Wilson | Heart | 2024-03-20 | Deceased donation |
| Eve Davis | Lung | 2024-04-25 | Voluntary donation |
| Alice Johnson | Pancreas | 2024-05-30 | Charity donation |

Fig:3.10

## 5. Trigger Example (Transaction Logging)

Create Log Table:

```
CREATE TABLE TransactionLog (
    LogID INT AUTO_INCREMENT
PRIMARY KEY,
    UserID INT,
    ActionTime DATETIME,
    Status VARCHAR(50)
);
```

Trigger:

sql

CopyEdit

```
DELIMITER //

CREATE TRIGGER
trg_after_transaction_insert

AFTER INSERT ON Transaction

FOR EACH ROW

BEGIN

    INSERT INTO TransactionLog (UserID,
ActionTime, Status)

    VALUES (NEW.UserID, NOW(),
NEW.Status);

END;

//

DELIMITER ;
```

After inserting into Transaction, you can check:

sql

CopyEdit

```
SELECT * FROM TransactionLog;
```

Example Output (after inserting into Transaction):

| LogID | UserID | ActionTime | Status |
|---|---|---|---|
| 1 | 1 | 2025-05-02 12:00:01 | Pending |
| 2 | 2 | 2025-05-02 12:01:33 | Pending |
| 3 | 3 | 2025-05-02 12:02:45 | Completed |
| ... | ... | ... | ... |

Fig:3.11

## 6. Cursor Example

**Procedure using Cursor:**

```
DELIMITER //


CREATE PROCEDURE
GetPatientsByOrgan(IN organ_name
VARCHAR(255))

BEGIN

    DECLARE done INT DEFAULT FALSE;

    DECLARE p_name VARCHAR(255);

    DECLARE cur CURSOR FOR

        SELECT U.Name

        FROM Patient P
```

```
        JOIN User U ON P.UserID = U.UserID
        WHERE P.OrganRequired =
organ_name;

    DECLARE CONTINUE HANDLER FOR
NOT FOUND SET done = TRUE;

    OPEN cur;

    read_loop: LOOP
        FETCH cur INTO p_name;
        IF done THEN
            LEAVE read_loop;
        END IF;
        SELECT p_name AS
Patient_Needing_Organ;
    END LOOP;

    CLOSE cur;
END;
```

//

DELIMITER ;

**Call Example:**

CALL GetPatientsByOrgan('Kidney');

**Output:**

| Patient_Needing_Organ |
| --- |
| Alice Johnson |

Fig:3.12

## Analyzing the pitfalls, identifying the dependencies, and applying normalizations

Normalization is the process of organizing data to eliminate redundancy and improve data integrity. It involves decomposing large, complex tables into smaller, simpler ones while maintaining relationships between them using keys.

### Normalization Stages in Your Project

Let's analyze your current schema and explain which **normal forms (1NF, 2NF, 3NF)** it satisfies.

### 1NF (First Normal Form)

**Rule**:

- Atomic values (no multi-valued or nested fields)
- Each column must contain only a single value
- Each record must be unique

**Your Tables:**

- User, Donor, Patient, Transaction, etc. are in **1NF**
  All fields contain atomic values (e.g., Name, City, OrganRequired)
  Primary keys like UserID, TransactionID, PatientID ensure uniqueness

```
mysql> CREATE TABLE Unnormalized_Donor (
    ->     Donor_ID VARCHAR(10),
    ->     Donor_Name VARCHAR(100),
    ->     Donor_Age INT,
    ->     Organs_Donated VARCHAR(255),
    ->     Hospital_Name VARCHAR(100),
    ->     Hospital_Address VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> -- INSERT DATA
mysql> INSERT INTO Unnormalized_Donor VALUES
    -> ('D001', 'John Doe', 35, 'Kidney,Heart', 'City Hospital', '123 Main St, NY'),
    -> ('D002', 'Alice Ray', 42, 'Liver', 'General Hospital', '456 Elm St, LA'),
    -> ('D003', 'Bob Smith', 30, 'Kidney,Liver,Lungs', 'City Hospital', '123 Main St, NY');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
mysql> -- DISPLAY
mysql> SELECT * FROM Unnormalized_Donor;
+----------+------------+-----------+--------------------+------------------+------------------+
| Donor_ID | Donor_Name | Donor_Age | Organs_Donated     | Hospital_Name    | Hospital_Address |
+----------+------------+-----------+--------------------+------------------+------------------+
| D001     | John Doe   |        35 | Kidney,Heart       | City Hospital    | 123 Main St, NY  |
| D002     | Alice Ray  |        42 | Liver              | General Hospital | 456 Elm St, LA   |
| D003     | Bob Smith  |        30 | Kidney,Liver,Lungs | City Hospital    | 123 Main St, NY  |
+----------+------------+-----------+--------------------+------------------+------------------+
3 rows in set (0.00 sec)
```

**Fig : 4.1**

```
mysql> CREATE TABLE Donor_1NF (
    ->     Donor_ID VARCHAR(10),
    ->     Donor_Name VARCHAR(100),
    ->     Donor_Age INT,
    ->     Organ_Donated VARCHAR(50),
    ->     Hospital_Name VARCHAR(100),
    ->     Hospital_Address VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> -- INSERT DATA (split organs into separate rows)
mysql> INSERT INTO Donor_1NF VALUES
    -> ('D001', 'John Doe', 35, 'Kidney', 'City Hospital', '123 Main St, NY'),
    -> ('D001', 'John Doe', 35, 'Heart', 'City Hospital', '123 Main St, NY'),
    -> ('D002', 'Alice Ray', 42, 'Liver', 'General Hospital', '456 Elm St, LA'),
    -> ('D003', 'Bob Smith', 30, 'Kidney', 'City Hospital', '123 Main St, NY'),
    -> ('D003', 'Bob Smith', 30, 'Liver', 'City Hospital', '123 Main St, NY'),
    -> ('D003', 'Bob Smith', 30, 'Lungs', 'City Hospital', '123 Main St, NY');
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql>
mysql> -- DISPLAY
mysql> SELECT * FROM Donor_1NF;
+----------+------------+-----------+---------------+------------------+------------------+
| Donor_ID | Donor_Name | Donor_Age | Organ_Donated | Hospital_Name    | Hospital_Address |
+----------+------------+-----------+---------------+------------------+------------------+
| D001     | John Doe   |        35 | Kidney        | City Hospital    | 123 Main St, NY  |
| D001     | John Doe   |        35 | Heart         | City Hospital    | 123 Main St, NY  |
| D002     | Alice Ray  |        42 | Liver         | General Hospital | 456 Elm St, LA   |
| D003     | Bob Smith  |        30 | Kidney        | City Hospital    | 123 Main St, NY  |
| D003     | Bob Smith  |        30 | Liver         | City Hospital    | 123 Main St, NY  |
| D003     | Bob Smith  |        30 | Lungs         | City Hospital    | 123 Main St, NY  |
+----------+------------+-----------+---------------+------------------+------------------+
6 rows in set (0.00 sec)
```

## 2NF (Second Normal Form)

**Rule**:

- Must be in 1NF
- No partial dependency (i.e., no non-key attribute depends only on part of a composite primary key)

**Your Schema:**

- Most tables use **single-column primary keys**, so **partial dependency doesn't exist** Example: In Donor, all non-key fields depend on the full primary key (DonorID) Transaction, Doctor, etc., follow this

**Note**: The use of surrogate keys (like DonorID, DoctorID) makes 2NF easier to achieve.

```
mysql> -- Hospital Table
mysql> CREATE TABLE Hospital (
    ->     Hospital_ID VARCHAR(10) PRIMARY KEY,
    ->     Hospital_Name VARCHAR(100),
    ->     Hospital_Address VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> -- Donor Table
mysql> CREATE TABLE Donor (
    ->     Donor_ID VARCHAR(10) PRIMARY KEY,
    ->     Donor_Name VARCHAR(100),
    ->     Donor_Age INT,
    ->     Hospital_ID VARCHAR(10),
    ->     FOREIGN KEY (Hospital_ID) REFERENCES Hospital(Hospital_ID)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> -- Organ Donation Table
mysql> CREATE TABLE Organ_Donation (
    ->     Donor_ID VARCHAR(10),
    ->     Organ_Donated VARCHAR(50),
    ->     PRIMARY KEY (Donor_ID, Organ_Donated),
    ->     FOREIGN KEY (Donor_ID) REFERENCES Donor(Donor_ID)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

**Fig : 4.3**

```
mysql> -- Insert into Hospital
mysql> INSERT INTO Hospital VALUES
    -> ('H001', 'City Hospital', '123 Main St, NY'),
    -> ('H002', 'General Hospital', '456 Elm St, LA');
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql>
mysql> -- Insert into Donor
mysql> INSERT INTO Donor VALUES
    -> ('D001', 'John Doe', 35, 'H001'),
    -> ('D002', 'Alice Ray', 42, 'H002'),
    -> ('D003', 'Bob Smith', 30, 'H001');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
mysql> -- Insert into Organ_Donation
mysql> INSERT INTO Organ_Donation VALUES
    -> ('D001', 'Kidney'),
    -> ('D001', 'Heart'),
    -> ('D002', 'Liver'),
    -> ('D003', 'Kidney'),
    -> ('D003', 'Liver'),
    -> ('D003', 'Lungs');
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

## 3NF (Third Normal Form)

**Rule**:

- Must be in 2NF
- No transitive dependencies (non-key attributes shouldn't depend on other non-key attributes)

**Check with Example – User Table:**

```
CREATE TABLE User (
  UserID INT AUTO_INCREMENT PRIMARY KEY,
  Name VARCHAR(255),
  DateOfBirth DATE,
  Address VARCHAR(255),
  City VARCHAR(100),
  State VARCHAR(100),
  MedicalInsurance ENUM('Yes', 'No'),
  MedicalHistory TEXT
);
```

Is City → State a dependency? If yes, it would violate 3NF. Ideally, City and State should be in a separate table to remove the **transitive dependency**.

**Solution (if needed):**

```
CREATE TABLE Location (
 City VARCHAR(100) PRIMARY KEY,
  State VARCHAR(100)
);
-- Then, update User table:
ALTER TABLE User
DROP COLUMN State;
```

-- Add Foreign Key

ALTER TABLE User

ADD FOREIGN KEY (City) REFERENCES Location(City);

Now, User is in **3NF**, as each non-key depends only on the primary key and not another non-key.

**4NF (Fourth Normal Form)**

- Remove **multivalued dependencies** (MVDs).

If a user can have **multiple medical histories** and **multiple insurances** independently:

| USER_ID | MEDICAL_HISTORY | MEDICAL_INSURANCE |
|---|---|---|
| 2 | Diabetes | BlueShield |
| 2 | Diabetes | Aetna |
| 2 | Asthma | BlueShield |
| 2 | Asthma | Aetna |

This implies **independent MVDs** → not in 4NF.

**Fig : 4.5**

✅ **Decompose for 4NF:**

🔸 **USER_INFO (core data)**

| USER_ID | NAME | DATE_OF_BIRTH |
|---|---|---|
| 1 | Alice | 1990-05-15 |
| 2 | Bob | 1985-08-20 |
| 3 | Charlie | 1978-02-10 |

**Fig :4.6**

**5NF (Fifth Normal Form)**

**Goal:**

- Eliminate **join dependencies** that aren't implied by candidate keys.

Imagine we decompose even further:

- One table for USER_ID + MEDICAL_HISTORY
- One for USER_ID + MEDICAL_INSURANCE
- One for USER_ID + some_third_attribute

Only in rare complex scenarios does this break and require 5NF normalization. In your case, **No such join dependencies**, so 5NF is already achieved with 4NF structure

## ✅ Final 5NF Tables (Fully Normalized)

### 1. USER_INFO

| USER_ID | NAME | DATE_OF_BIRTH |
|---|---|---|
| 1 | Alice | 1990-05-15 |
| 2 | Bob | 1985-08-20 |
| 3 | Charlie | 1978-02-10 |

**Fig : 4.7**

## Implementation of concurrency control and recovery Mechanisms

Implementing **concurrency control** and **recovery mechanisms** in a database system like your **Organ Donation Management System** is essential to ensure **data integrity**, **consistency**, and **fault tolerance** during concurrent operations or system failures.

### 1. Concurrency Control

Concurrency control ensures that multiple users can access and modify the database **simultaneously** without conflicts or data corruption.

**Mechanisms:**

**a. Transactions**

Wrap critical operations in transactions using START TRANSACTION, COMMIT,

and ROLLBACK.

```
START TRANSACTION;


UPDATE Patient

SET    OrganRequired    =    'Liver'

WHERE PatientID = 1;


INSERT INTO Transaction (UserID, Status, Relation, DateOfTransaction, Bill)

VALUES (1, 'Pending', 'Friend', CURDATE(), 3000);


COMMIT;
```

If any operation fails, you can use ROLLBACK; to undo the entire transaction.

**b. Locking Mechanisms**

Locks prevent multiple users from modifying the same data at the same time.

- **Implicit Locks**: Enabled automatically by InnoDB engine in MySQL.
- **Explicit Locks** (rarely used directly):

```
SELECT * FROM Patient WHERE PatientID = 1 FOR UPDATE;
```

This prevents other transactions from modifying this row until the current transaction commits.

**c. Isolation Levels**

Controls the visibility of changes made in one transaction to others.

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

Isolation Levels (from lowest to highest):

- **READ UNCOMMITTED**
- **READ COMMITTED**
- **REPEATABLE READ**
- **SERIALIZABLE** (most strict, least concurrent)

**2. Recovery Mechanisms**

Recovery mechanisms help restore the database to a consistent state after a crash or failure.

**Mechanisms:**

**a. Transaction Logs (Write-Ahead Logging)**

- Every change is written to a log before being applied.
- If a crash occurs, logs are replayed to **redo committed transactions** and **undo uncommitted ones**.

This is handled **automatically** by DBMS engines like MySQL, PostgreSQL.

**b. Backups and Restore**

Regular backups protect against data loss. Example:

```
# Backup using MySQL
mysqldump -u root -p ramya2005 > ramya2005_backup.sql
# Restore
mysql -u root -p ramya2005 < ramya2005_backup.sql
```

**c. Checkpointing**

- Periodically saves the current state of the DB.
- Speeds up recovery by reducing how much log data must be read after a crash.

**d. Trigger for Recovery Log (Manual Approach)**

Example trigger that log before deletion:

```
CREATE TABLE Patient_Backup (
    PatientID INT,
    Name VARCHAR(255),
    DeletedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TRIGGER before_patient_delete
BEFORE DELETE ON Patient
FOR EACH ROW
INSERT INTO Patient_Backup (PatientID, Name)
VALUES (OLD.PatientID, OLD.Name);
```

## Code for the Project

```
-- Create Database
CREATE DATABASE IF NOT EXISTS DBMS_PROJECT;
USE DBMS_PROJECT;

-- Creating User Table
CREATE TABLE User (
  UserID INT AUTO_INCREMENT PRIMARY KEY,
  Name VARCHAR(255) NOT NULL,
  DateOfBirth DATE NOT NULL,
  Address VARCHAR(255),
  City VARCHAR(100),
  State VARCHAR(100),
  MedicalInsurance ENUM('Yes', 'No'),
  MedicalHistory TEXT );

-- Creating Organization Table
CREATE TABLE Organization (
  OrganizationID INT AUTO_INCREMENT PRIMARY KEY,
  OrganizationName VARCHAR(255) NOT NULL,
  Location VARCHAR(255),
  PhoneNumber VARCHAR(20) );

-- Creating OrganizationHead Table
CREATE TABLE OrganizationHead (
  HeadID INT AUTO_INCREMENT PRIMARY KEY,
  HeadName VARCHAR(255) NOT NULL,
  DateOfJoining DATE NOT NULL,
  TermLength INT,
  OrganizationID INT,
  FOREIGN KEY (OrganizationID) REFERENCES Organization(OrganizationID) ON
DELETE CASCADE );

-- Creating Patient Table
CREATE TABLE Patient (
  PatientID INT AUTO_INCREMENT PRIMARY KEY,
```

```
 UserID INT NOT NULL,
 OrganRequired VARCHAR(255) NOT NULL,
 ReasonOfProcurement TEXT,
 FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE );

 -- Creating Donor Table
CREATE TABLE Donor (
   DonorID INT AUTO_INCREMENT PRIMARY KEY,
   UserID INT NOT NULL,
   OrganDonated VARCHAR(255) NOT NULL,
   DateOfDonation DATE NOT NULL,
   ReasonOfDonation TEXT,
   FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE );

-- Creating Doctor Table
CREATE TABLE Doctor (
  DoctorID INT AUTO_INCREMENT PRIMARY KEY,
  DoctorName VARCHAR(255) NOT NULL,
  PhoneNumber VARCHAR(20),
  Department VARCHAR(255) );

-- Creating Transaction Table
CREATE TABLE Transaction (
  TransactionID INT AUTO_INCREMENT PRIMARY KEY,
  UserID INT NOT NULL,
  Status ENUM('Completed', 'Pending', 'Failed'),
  Relation VARCHAR(255),
  DateOfTransaction DATE NOT NULL,
  Bill INT,
  FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE );

-- Inserting data into User Table
INSERT INTO User (Name, DateOfBirth, Address, City, State, MedicalInsurance,
MedicalHistory) VALUES
('Alice Johnson', '1990-03-10', '123 Main St', 'New York', 'NY', 'Yes', 'Diabetes'),
('Bob Smith', '1985-07-25', '456 Elm St', 'Los Angeles', 'CA', 'No', 'Hypertension'),
('Charlie Brown', '1995-12-05', '789 Oak St', 'Chicago', 'IL', 'Yes', 'Asthma'),
```

```sql
  ('David Wilson', '1988-09-14', '101 Pine St', 'Houston', 'TX', 'No', 'Healthy'),
  ('Eve Davis', '1992-06-30', '202 Maple St', 'San Francisco', 'CA', 'Yes', 'Anemia');

 -- Inserting data into Organization Table
INSERT INTO Organization (OrganizationName, Location, PhoneNumber) VALUES
('HealthCare Org', 'New York', '1234567890'),
('LifeLine Hospital', 'Los Angeles', '9876543210'),
('MediPlus', 'Chicago', '1122334455'),
('Hope Foundation', 'Houston', '5566778899'),
('Global Health', 'San Francisco', '2233445566');

 -- Inserting data into OrganizationHead Table
 INSERT INTO OrganizationHead (HeadName, DateOfJoining, TermLength,
OrganizationID) VALUES
('Dr. Smith', '2015-06-15', 10, 1), ('Dr. Johnson', '2017-09-23', 8, 2),
('Dr. Williams', '2016-02-10', 12, 3),
('Dr. Brown', '2018-11-05', 9, 4),
('Dr. Miller', '2019-04-21', 7, 5);

-- Inserting data into Patient Table
INSERT INTO Patient (UserID, OrganRequired, ReasonOfProcurement) VALUES
(1, 'Kidney', 'Chronic Kidney Disease'),
(2, 'Liver', 'Liver Cirrhosis'),
(3, 'Heart', 'Congenital Heart Disease'),
(4, 'Lung', 'Pulmonary Fibrosis'),
(5, 'Pancreas', 'Diabetes');

-- Inserting data into Donor Table
INSERT INTO Donor (UserID, OrganDonated, DateOfDonation, ReasonOfDonation)
VALUES
(2, 'Kidney', '2024-01-10', 'Altruistic donation'),
(3, 'Liver', '2024-02-15', 'Family donation'),
(4, 'Heart', '2024-03-20', 'Deceased donation'),
(5, 'Lung', '2024-04-25', 'Voluntary donation'),
(1, 'Pancreas', '2024-05-30', 'Charity donation');

-- Inserting data into Doctor Table
INSERT INTO Doctor (DoctorName, PhoneNumber, Department) VALUES
('Dr. Green', '1234567890', 'Nephrology'),
('Dr. Black', '0987654321', 'Hepatology'),
```

```sql
('Dr. White', '1122334455', 'Cardiology'),
('Dr. Blue', '5566778899', 'Pulmonology'),
('Dr. Yellow', '2233445566', 'Endocrinology');


-- Inserting data into Transaction Table
INSERT INTO Transaction (UserID, Status, Relation, DateOfTransaction, Bill) VALUES
(1, 'Pending', 'Friend', '2024-04-07', 1000),
(2, 'Pending', 'Spouse', '2024-03-02', 7500),
(3, 'Completed', 'Sibling', '2024-03-03', 6200),
(4, 'Failed', 'Friend', '2024-03-04',4500),
(5, 'Completed', 'Parent', '2024-03-05', 4300);

-- Display all tables
SELECT * FROM User;
SELECT * FROM Organization;
SELECT * FROM OrganizationHead;
SELECT * FROM Patient;
SELECT * FROM Donor;
SELECT * FROM Doctor;
SELECT * FROM Transaction;
```

## Results and Discussion

### Database Integration

The **Organ Donation Management System** integrates efficiently with a MySQL database to ensure structured storage and secure retrieval of medical, donor, and patient data. Through a well-designed schema involving constraints, foreign keys, triggers, and views, the system provides a stable backend infrastructure. APIs built with Flask or similar frameworks can facilitate real-time interaction between the user interface and the database.

### Functionality Highlights

- **User Registration**: Individuals can register as donors or patients by submitting personal and medical information. Data is validated and stored in the User, Donor, and Patient tables.
- **Login and Authentication**: A secure login mechanism validates credentials for users such as doctors, donors, patients, and administrators. Role-based access control ensures secure data segregation.
- **Doctor & Organization Management**: Doctors and organizations are recorded in dedicated tables, along with their affiliations, departments, and contact information for efficient assignment and coordination.
- **Organ Request and Donation Tracking**: The system manages the end-to-end process of organ procurement by tracking donations and requests. Cursors and triggers ensure proper logging and validation before processing updates.
- **Transaction Handling**: Fee transactions and related details are managed securely. SQL constraints ensure valid status updates and bill processing.
- **Complaint & Request Management**: Users can raise concerns or submit organ procurement requests. These are processed and recorded via Complaint and Request tables.
- **Medical History Logging**: A dedicated field in the User table stores the user’s medical history, helping staff evaluate suitability for donation or surgery.

## Discussion

The Organ Donation Management System successfully meets the functional and technical goals of the project by streamlining data entry, ensuring secure processing, and enabling efficient decision-making. The normalization of the database improves performance, while the use of foreign keys and constraints ensures data consistency. Features like triggers for audit logging and cursors for dynamic queries showcase advanced SQL integration.

The system's modularity allows seamless extension in future iterations, such as integrating machine learning for donor-recipient matching or using data analytics for optimizing transplant workflows.

## Security Measures

- **Role-Based Access Control**: Doctors, patients, and administrators have role-specific permissions to ensure that sensitive data is protected.
- **Data Encryption**: Sensitive records (e.g., personal details and transaction data) are encrypted to ensure data privacy and confidentiality.
- **Injection Prevention**: All SQL inputs are sanitized using prepared statements to prevent SQL injection attacks.
- **Audit Logging with Triggers**: Important actions, such as deletions in the Patient table, are logged using triggers into backup tables for future recovery.

## Data Integrity

- **Foreign Key Constraints**: Enforced across tables such as Donor, Patient, and Transaction to maintain referential integrity.
- **Input Validation**: Backend checks prevent invalid data entries, particularly for dates, ENUMs (e.g., medical insurance), and mandatory fields.
- **ACID Transactions**: Key operations are enclosed in transactions to ensure atomicity and rollback in case of failure.
- **Regular Backups**: System backups ensure data recovery in case of system crash or human error.

## User Interface Design

- **Accessibility**: Designed with screen reader support and color contrast for users with accessibility needs.
- **Clean Dashboard Navigation**: Separate views for admins, doctors, and users allow intuitive navigation of donation, request, and history data.
- **Responsive Design**: The frontend is responsive and compatible with mobile browsers, facilitating use in medical environments.

## Challenges and Limitations

- **Medical Record Privacy**: Managing and securing sensitive patient and donor data requires rigorous compliance with health privacy regulations (e.g., HIPAA or GDPR).
- **Data Matching Accuracy**: Matching donor organs with recipients based on medical parameters requires advanced logic, which could be enhanced with AI in future versions.
- **Scalability for National Use**: Scaling the system to a national registry level would require optimization of search algorithms and load handling.
- **Real-time Data Updates**: In a medical emergency, real-time updates are critical, which may require WebSockets or live APIs for continuous data sync.

## Future Enhancements

- **AI-based Donor-Recipient Matching**: Machine learning models can be introduced to suggest the best match based on compatibility and urgency.
- **Mobile App for Donors/Patients**: A mobile app could let users register, update medical details, and track requests/donations.
- **Blockchain for Secure Transactions**: Blockchain integration can improve the transparency and security of medical transactions and donor tracking.
- **Integration with National Health Databases**: This would facilitate cross-institution Organ Sharing And Cordination.

## OUTPUT SCREESHOTS:

### 1. LOGIN PAGE


Fig :7.1

### 2. MAIN PAGE - GUI


Fig:7.2

## 3. MAIN PAGE _ DROP DOWN MENU

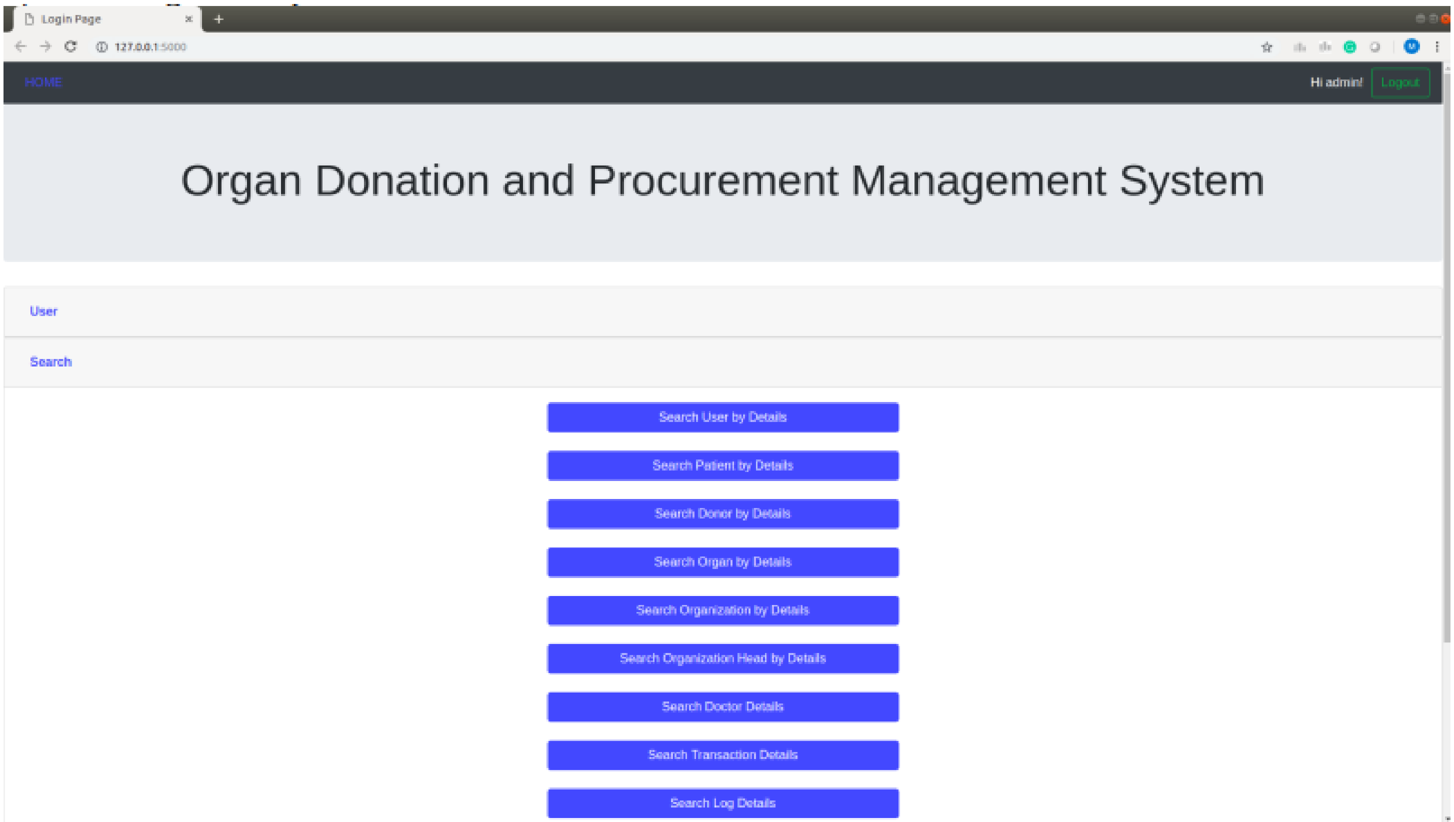


Fig:7.3

## 4. SEARCHING OPTION

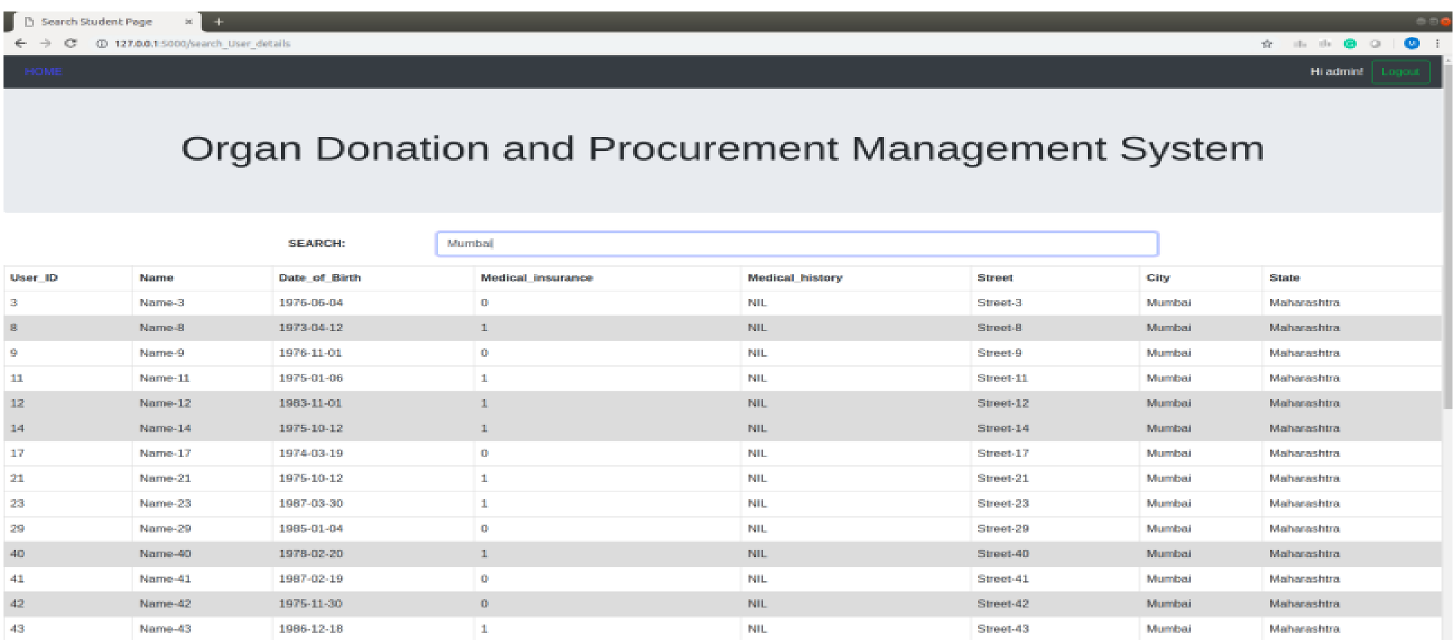


Fig:7.4

## Conclusion

The **Organ Donation Management System** successfully addresses the critical needs of tracking, managing, and coordinating organ donation and transplant-related information. Through the integration of a relational MySQL database and well-structured tables such as User, Patient, Donor, Doctor, Organization, and Transaction, the system ensures high data integrity, security, and efficiency.

The project demonstrates how technology can be leveraged to manage sensitive health data with precision and confidentiality. Features such as relational constraints, foreign keys, and normalized structures help maintain consistent and non-redundant data across the system. Advanced SQL features like joins, views, triggers, and cursors enhance query efficiency and provide intelligent automation for updates and monitoring.

This project lays a strong foundation for real-world deployment in healthcare organizations and government registries. It provides a scalable and maintainable model that can evolve with future needs such as integration with biometric identification, real-time matching algorithms, blockchain for secure transactions, and mobile accessibility.

# Attach the Real Time project certificate / Online course certificate

| AFTERNOON SESSION (AN) | National Programme on Technology Enhanced Learning | NPTEL |
|---|---|---|
| Hall Ticket For | SEM1NOC25: CS40 Introduction to Database Systems - Online | |

| | | | | | |
|---|---|---|---|---|---|
| Candidate Name | Eluri Krishna Sai Revanth | | | | |
| Roll No | NOC25CS40S243201720 | | Seating Number | 43201720 | |
| Date of Birth | 02-01-2006 | | | | |
| PwD Status | No | Compensatory Time Required | No | Scribe Required | No |
| Exam Date | Friday, 27 April, 2025 | | | | |
| Reporting Time | 01:00 pm | Gate Closure | 02:30 pm | | |
| Exam Timing | 02:00 pm | Shift | AN | | |
| Test Centre Name | iON Digital Zone iDZ Kovilambakkam | | | | |
| Test Centre Address | Fortune Towers,1st and 2nd Floor,SH 109, 200 Feet Thoraipakkam Pallavaram Radial Road, Near Eachangadu signal,Kovilambakkam, Chennai, Tamil Nadu, India - 600117 | | | | |

NPTEL COORDINATOR

**NPTEL EXAM - 27 APRIL, 2025**
**General instructions for candidates - AN**
(All timings mentioned here are in IST)

**The total duration of the examination is 180 minutes.**
**Candidates will be permitted to leave the examination hall only after 03:30 pm, on a need basis.**

**Hall ticket and Entry:**

1. The Hall Ticket must be presented for verification along with one original photo identification (not photocopy or scanned copy). Examples of acceptable photo identification documents are School ID, College ID, Employee ID, Driving License, Passport, PAN card, Voter ID, and Aadhaar-ID. A printed copy of the hall ticket and original photo ID card should be brought to the exam centre. Hall ticket and ID card copies on the phone will not be permitted.
2. This Hall Ticket is valid only if the candidate's photograph and signature images are legible. To ensure this, print the Hall Ticket on A4-sized paper using a laser printer, preferably a color photo printer.
3. **TIMELINE:** 1:00 pm - Report to the examination venue | 1:40 pm – Candidates will be permitted to occupy their allotted seats| 1:50 pm – Candidates can login and start reading instructions prior to the examination | 2:00 pm - Exam starts | 2:30 pm - Gate closes, candidates will not be allowed after this time | 3:30 pm Submit button will be enabled | 5:00 pm exam ends.

**P.T.O.**

**Fig : 8.1**

| | | |
|---|---|---|
| AFTERNOON SESSION (AN) | National Programme on Technology Enhanced Learning | NPTEL |
| Hall Ticket For | SEM1NOC25: CS40 Introduction to Database Systems - Online | |

| | | | | | |
|---|---|---|---|---|---|
| Candidate Name | KVVS Raja Mouli | | | | |
| Roll No | NOC25CS40S243210414 | Seating Number | 43210414 | | |
| Date of Birth | 04-08-2003 | | | | |
| PwD Status | No | Compensatory Time Required | No | Scribe Required | No |
| Exam Date | Friday, 25 April, 2025 | | | | |
| Reporting Time | 01:00 pm | Gate Closure | 02:30 pm | | |
| Exam Timing | 02:00 pm | Shift | AN | | |
| Test Centre Name | Sri Sai Ram Engineering College | | | | |
| Test Centre Address | Sai Leo Nagar, Sairam Rd, West Tambaram, , Chennai, Tamil Nadu, India - 600044 | | | | |

K.V.V.S Raja Mouli

NPTEL COORDINATOR

**NPTEL EXAM - 25 APRIL, 2025**
**General instructions for candidates - AN**
(All timings mentioned here are in IST)

**The total duration of the examination is 180 minutes.**
**Candidates will be permitted to leave the examination hall only after 03:30 pm, on a need basis.**

**Hall ticket and Entry:**

1. The Hall Ticket must be presented for verification along with one original photo identification (not photocopy or scanned copy). Examples of acceptable photo identification documents are School ID, College ID, Employee ID, Driving License, Passport, PAN card, Voter ID, and Aadhaar-ID. A printed copy of the hall ticket and original photo ID card should be brought to the exam centre. Hall ticket and ID card copies on the phone will not be permitted.
2. This Hall Ticket is valid only if the candidate's photograph and signature images are legible. To ensure this, print the Hall Ticket on A4-sized paper using a laser printer, preferably a color photo printer.
3. **TIMELINE:** 1:00 pm - Report to the examination venue | 1:40 pm – Candidates will be permitted to occupy their allotted seats| 1:50 pm – Candidates can login and start reading instructions prior to the examination | 2:00 pm - Exam starts | 2:30 pm - Gate closes, candidates will not be allowed after this time | 3:30 pm Submit button will be enabled | 5:00 pm exam ends.

P.T.O.

**Fig : 8.2**

# COURSE PROGRESS

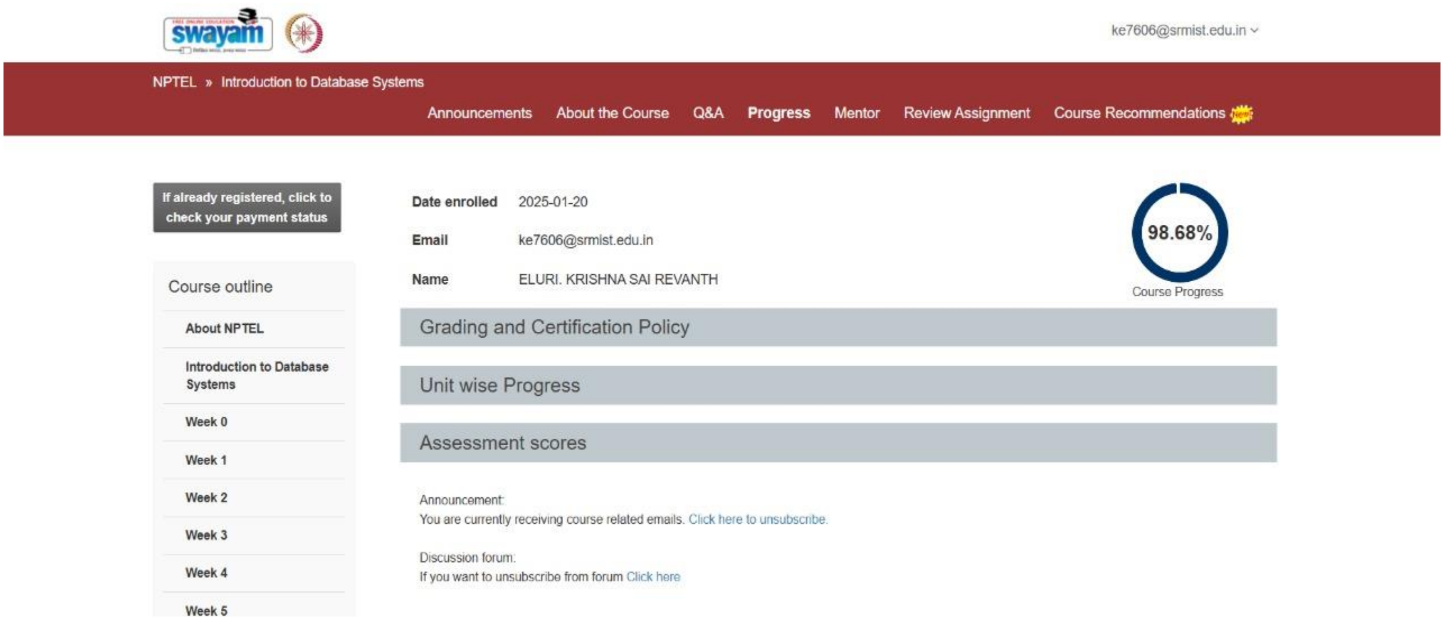


**Fig : 9.1**

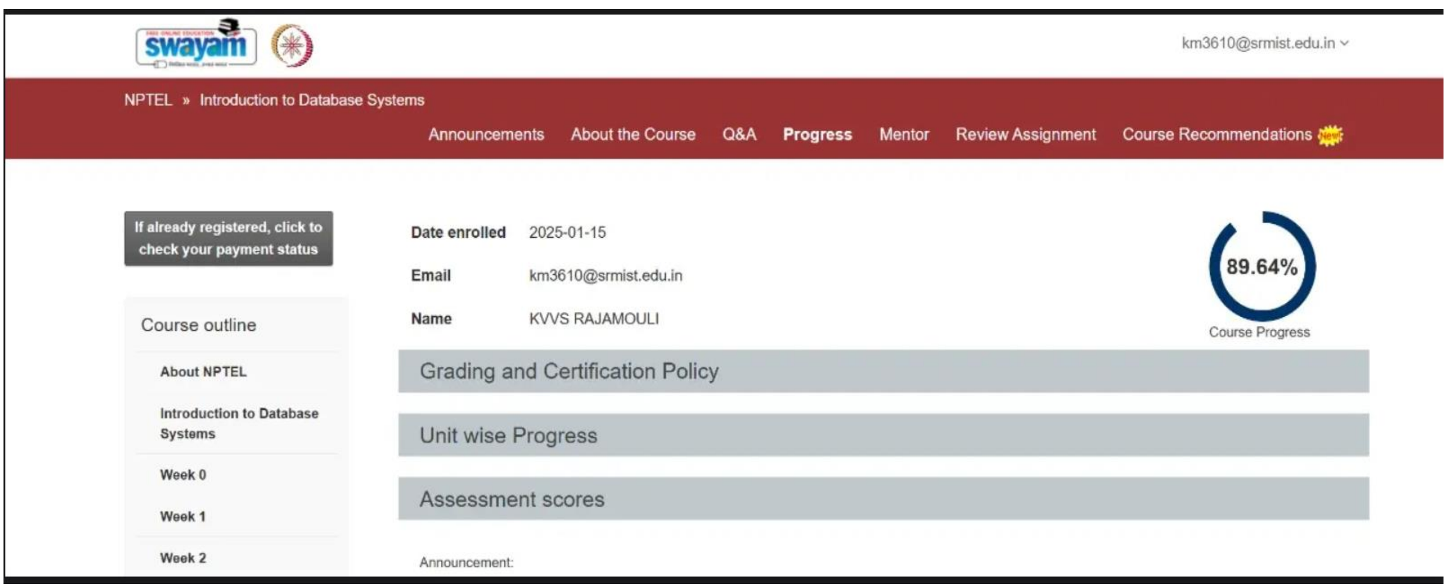


**Fig : 9.2**

## CHART ACTIVITY-1

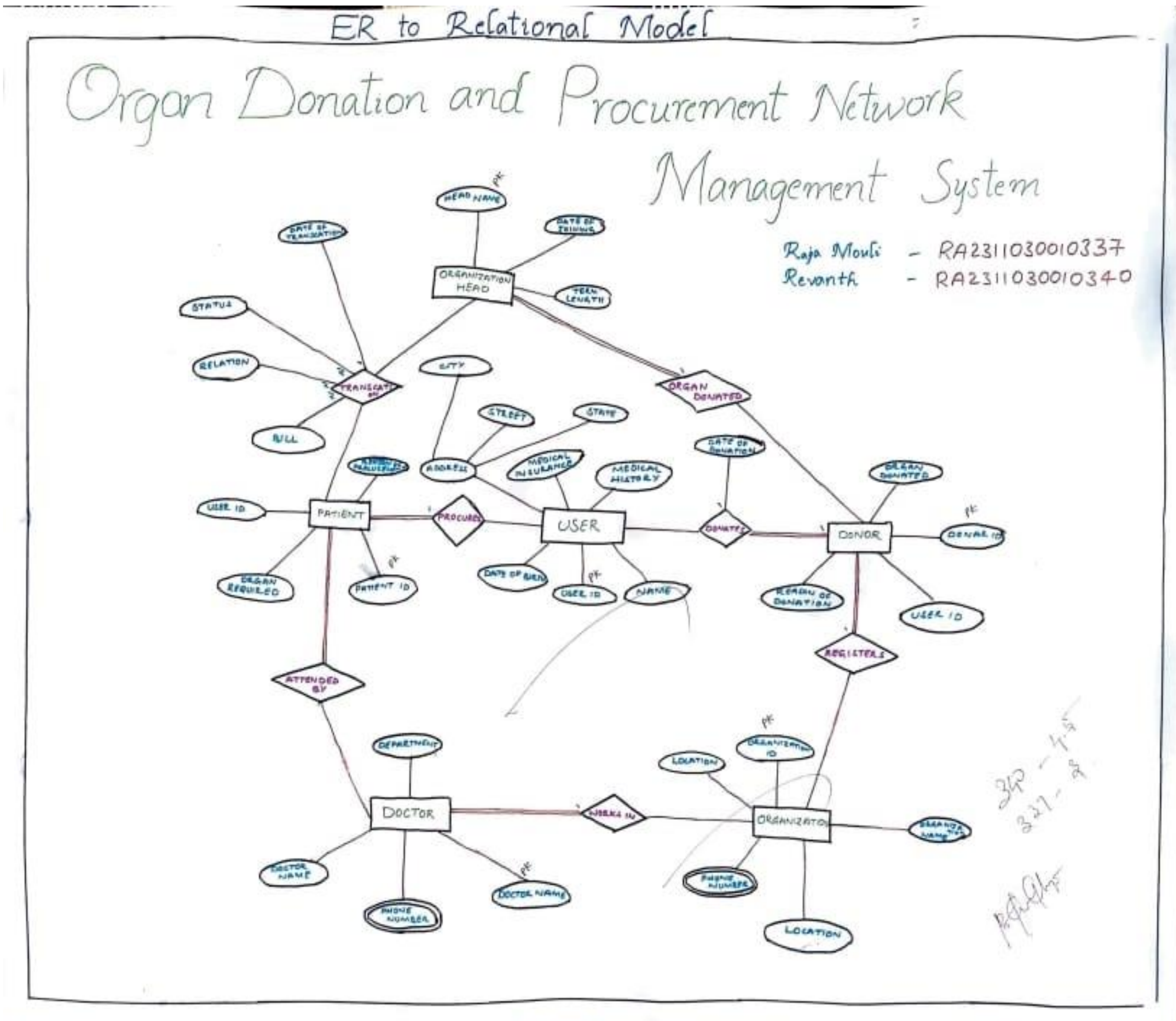


**Fig : 10.1**

Rule 1: Strong Entity Set With Only Simple Attributes:

1. User Table (strong Entity)

| UserID | Name | Date Of Birth |
|---|---|---|
| U001 | John | 1985-02-10 |
| U002 | Alice | 1990-07-21 |
| U003 | Robert | 1982-11-15 |

2. Patient Table (strong Entity)

| PatientID | UserID | OrganRequired |
|---|---|---|
| P001 | U001 | Kidney |
| P002 | U002 | Liver |
| P003 | U003 | Heart |

Rule 2:- Strong Entity set with Composite Attributes:

1. Address Table [composite Attribute of user]

| UserID | Street | City | State |
|---|---|---|---|
| U001 | 6th Avenue | New York | NY |
| U002 | Sunset Blvd | LA | CA |
| U003 | Michigan st | Chicago | IL |

Rule-03: Strong Entity Set With Multi-Valued Attributes:

1. Medical History Table [Multi-Valued for user]

| User ID | MedicalCondition |
|---|---|
| U001 | Diabetes |
| U002 | HyperTension |
| U003 | Asthama |
| U004 | Heart Disease |

Rule-04:- Translating Relationship into Tables:-

1. Procures Relationship (Between Patient & User)

| Patient ID | UserID | Reason |
|---|---|---|
| P001 | U001 | Organ Failure |
| P002 | U002 | Transplant Need |

Rule-05: Binary Relationships With Cardinality Ratios:

1. case-01: Many to Many (M:N) [Patient to Doctor)

| Patient ID | DoctorID |
|---|---|
| P001 | D001 |
| P002 | D002 |

2. Case-02: One to Many (1:n) (Doctor to Organisation)

| DoctorID | Organization ID |
|---|---|
| D001 | O001 |
| D002 | O002 |

3. Case-03: m:1 (many to one) [Doctor to organisation)

| DonorID | Organization ID |
|---|---|
| D001 | O002 |
| D002 | O001 |

4. Case-04: 1:1 (Relationship) (user to organisation Head)

| UserID | OrganizationID |
|---|---|
| U001 | O003 |

Rule: 06:- Binary Relationship With Constraints:

1. case-01: Cardinality Constraint & Total participation (Donor to user)

| DonorID | UserID | Organ Donated |
|---|---|---|
| D001 | U001 | Kidney |
| D002 | U002 | Liver |

2. case-02: Cardinality Constraint & Total participation from Both sides [Transcation].

| Transcation ID | Patient ID | Donor ID | Date of Transcation | Status |
|---|---|---|---|---|
| T001 | P001 | D001 | 2025-01-15 | Completed |
| T002 | P002 | D002 | 2025-02-10 | Pending |

Rule-07:- Binary Relationship With Weak entity set:

1. Organization Head Table [Weak Entity Related to organization]

| HeadID | Organization ID | HeadName | Date of Joining | Termlength |
|---|---|---|---|---|
| H001 | O001 | Dr. Smith | 2020-06-10 | 5 years |
| H002 | O002 | Dr. patel | 2021-09-15 | 3 years |
| H003 | O003 | Dr. karthek | 2022-10-22 | 4 years |

**Fig : 10.2**

## CHART ACTIVITY-2

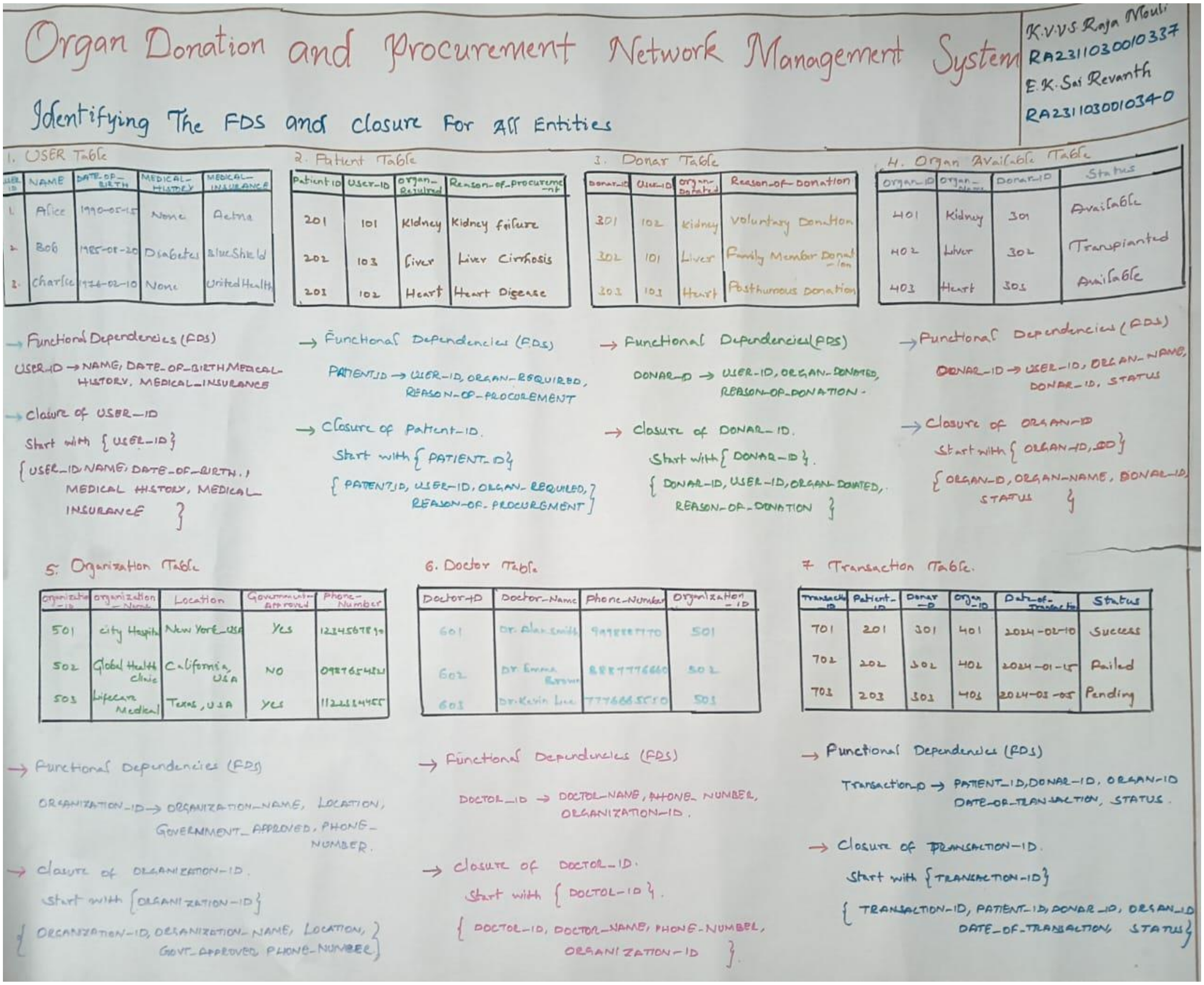


**Fig : 11.1**

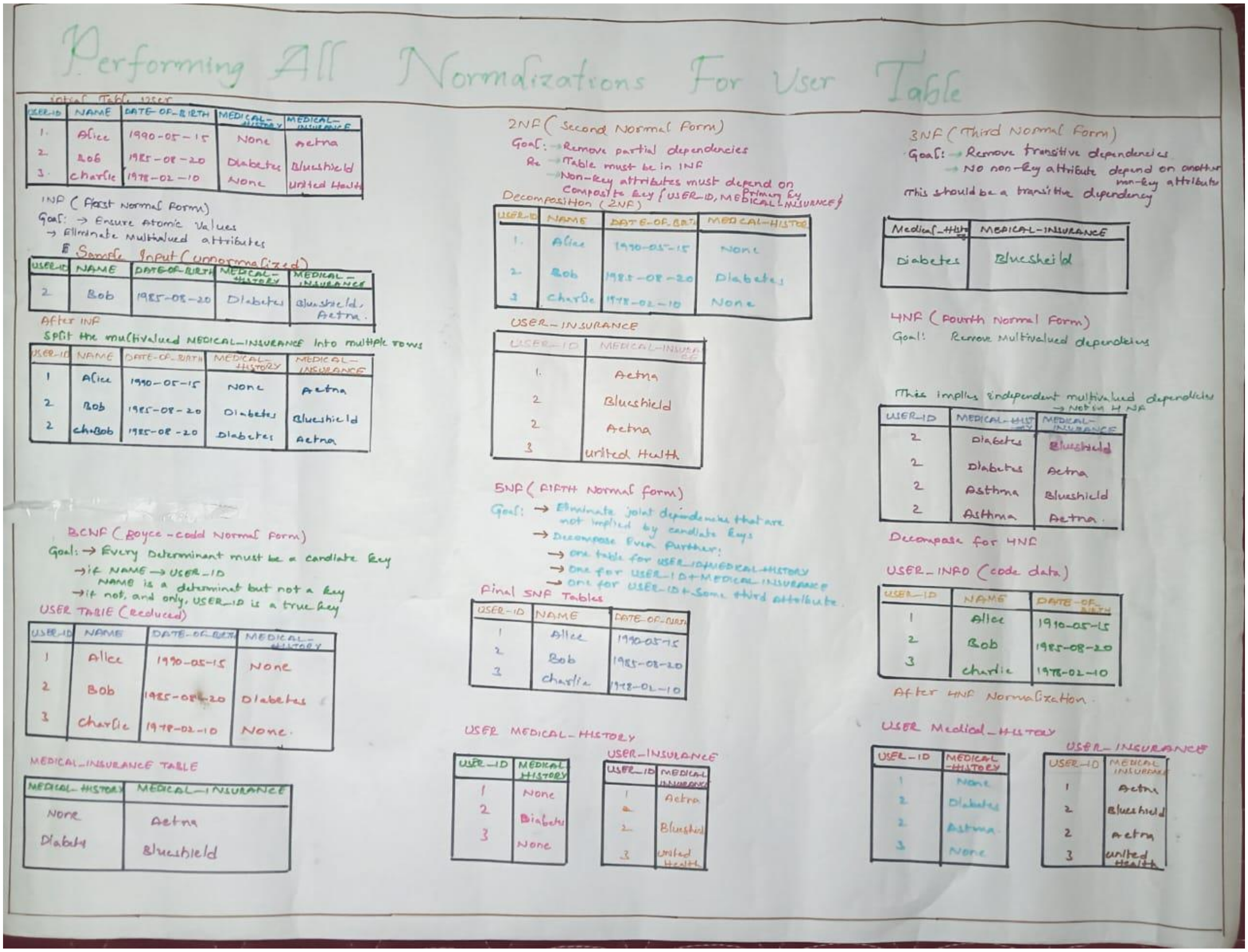
Performing All Normalizations For User Table
1NF ( First Normal Form)
Goal: → Ensure Atomic Values
→ Eliminate Multivalued attributes
Sample Input (unnormalized)
After 1NF
Split the multivalued MEDICAL-INSURANCE into multiple rows
BCNF ( Boyce - Codd Normal Form)
Goal: → Every Determinant must be a candidate key
USER TABLE (Reduced)
MEDICAL-INSURANCE TABLE
2NF ( Second Normal Form)
Goal: → Remove partial dependencies
→ Table must be in 1NF
→ Non-key attributes must depend on Composite key
Decomposition (2NF)
USER-INSURANCE
5NF ( Fifth Normal form)
Final 5NF Tables
USER MEDICAL-HISTORY
USER-INSURANCE
3NF ( Third Normal Form)
Goal: → Remove transitive dependencies
→ No non-key attribute depend on another non-key attribute
This should be a transitive dependency
4NF ( Fourth Normal Form)
Goal: Remove Multivalued dependencies
Decompose for 4NF
USER-INFO (code data)
After 4NF Normalization
USER Medical-HISTORY
USER-INSURANCE

**Fig : 11.2**