



# SUPERIOR UNIVERSITY

**Name :** MUHAMMAD AHMAD

**Roll No :** SU92-BSAIM-F23-135

**Section :** 4-C

**Project :** Organ Donation System

**Date :** 11th May, 2025

**Subject :** Database Lab

**Submitted To :** Prof M. Muneeb Saleem

**: Members :**

<b>Name</b>	<b>Roll No</b>
Hafiz Haider Ali	SU92-BSAIM-F23-130
Alishba Haroon	SU92-BSAIM-F23-116
Muhammad Ahmad	SU92-BSAIM-F23-135

# Organ Donation Network Project and Database Documentation

Version 1.0

May 11, 2025

# Contents

1	Project Overview	3
2	System Architecture	3
2.1	Technologies Used	3
2.2	Application Structure	3
2.3	Database Schema	3
2.3.1	patient	3
2.3.2	organ	4
2.3.3	organisation	4
2.3.4	donor	4
2.3.5	doctor	5
2.3.6	attended_by	5
2.3.7	donated	6
2.3.8	patient_contact	6
2.3.9	donor_contact	6
2.3.10	doctor_contact	7
2.3.11	organisation_contact	7
2.3.12	locations	7
3	Functionalities	7
3.1	Organization Management	7
3.2	Patient Management	8
3.3	Donor Management	8
3.4	Doctor Management	8
3.5	Donor Matching and Transplantation	8
3.6	Utility Functions	8
4	Key Routes and Endpoints	9
5	Initial Data	9
5.1	Organisation.....	10
5.2	Organ.....	10
5.3	Doctor.....	10
5.4	Donor.....	10
5.5	Donated.....	10
5.6	Patient.....	10
5.7	Attended_by.....	10
5.8	Contact Tables.....	10
5.9	Locations.....	10
5.10	Sample Query.....	11
6	Security Considerations	11
7	Setup and Installation	11
7.1	Prerequisites.....	11
7.2	Installation Steps.....	12

8	Usage Instructions	12
9	Limitations	13
10	Future Improvements	13
11	Conclusion	13

# 1 Project Overview

The Organ Donation Network is a web-based application developed using Flask and MySQL to manage organ donation and transplantation processes. The system facilitates interactions among organizations, patients, donors, and doctors, streamlining operations such as donor registration, patient management, organ allocation, and transplant tracking. The database, `organ_donation_network_db`, supports these functionalities by storing and managing relevant data.

## 2 System Architecture

### 2.1 Technologies Used

- Backend: Flask (Python web framework)
- Database: MySQL (Relational database for storing data)
- Frontend: HTML templates with Jinja2 for dynamic rendering
- Other Libraries:
  - Flask-MySQLdb: For MySQL database connectivity
  - datetime: For handling date operations

### 2.2 Application Structure

The Flask application is organized into routes handling different functionalities, including user authentication, data management, and donor-patient matching. Global variables like `organisation_id` and `org_login` track session state, though this is not ideal for production.

### 2.3 Database Schema

The database (`organ_donation_network_db`) consists of 12 tables, each serving a specific purpose. Below is a detailed description of each table, including columns, constraints, and relationships.

#### 2.3.1 patient

Stores patient information.

- Columns:
  - `patient_id`: Integer, Primary Key, Auto-increment
  - `patient_name`: Varchar(20)
  - `pass`: Varchar(20), Not Null
  - `date_of_birth`: Date, Not Null
  - `insurance_no`: Varchar(20)
  - `house_no`: Varchar(20)

- street\_no: Varchar(20)
- city: Varchar(20), Not Null
- state: Varchar(20), Not Null
- organisation\_id: Integer, Not Null

- Constraints:

- Primary Key: patient\_id

### 2.3.2 organ

Stores organ types available for donation.

- Columns:

- organ\_id: Integer, Primary Key, Auto-increment
  - organ\_name: Varchar(20), Not Null

- Constraints:

- Primary Key: organ\_id

### 2.3.3 organisation

Stores organization (e.g., hospital) details.

- Columns:

- organisation\_id: Integer, Primary Key, Auto-increment
  - pass: Varchar(20), Not Null
  - organisation\_name: Varchar(20)
  - head\_name: Varchar(20)
  - office\_no: Varchar(20)
  - street\_no: Varchar(20)
  - city: Varchar(20)
  - state: Varchar(20)

- Constraints:

- Primary Key: organisation\_id

### 2.3.4 donor

Stores donor information.

- Columns:

- donor\_id: Integer, Primary Key, Auto-increment
  - donor\_name: Varchar(20)

- date\_of\_birth: Date, Not Null
- house\_no: Varchar(20)
- street\_no: Varchar(20)
- city: Varchar(20), Not Null
- state: Varchar(20), Not Null
- organisation\_id: Integer

- Constraints:

- Primary Key: donor\_id
- Foreign Key: organisation\_id references organisation(organisation\_id)

### 2.3.5 doctor

Stores doctor information.

- Columns:

- doctor\_id: Integer, Primary Key, Auto-increment
- doctor\_name: Varchar(20)
- date\_of\_birth: Date
- date\_of\_joining: Date
- highest\_degree: Varchar(20)
- organ\_id: Integer
- organisation\_id: Integer
- pass: Varchar(20)

- Constraints:

- Primary Key: doctor\_id
- Foreign Key: organ\_id references organ(organ\_id)
- Foreign Key: organisation\_id references organisation(organisation\_id)

### 2.3.6 attended\_by

Links patients to doctors with demand dates.

- Columns:

- patient\_id: Integer
- doctor\_id: Integer
- date\_of\_demand: Date, Not Null

- Constraints:

- Primary Key: (patient\_id, doctor\_id)
- Foreign Key: patient\_id references patient(patient\_id)
- Foreign Key: doctor\_id references doctor(doctor\_id)

### 2.3.7 donated

Tracks organ donations and transplantations.

- Columns:
  - donor\_id: Integer
  - organ\_id: Integer
  - date\_of\_donation: Date, Not Null
  - date\_of\_expiry: Date
  - transplantation\_date: Date
  - patient\_id: Integer
- Constraints:
  - Primary Key: (donor\_id, organ\_id)
  - Foreign Key: donor\_id references donor(donor\_id)
  - Foreign Key: organ\_id references organ(organ\_id)
  - Check: date\_of\_expiry >= date\_of\_donation
  - Check: date\_of\_expiry >= transplantation\_date
  - Check: transplantation\_date >= date\_of\_donation

### 2.3.8 patient\_contact

Stores patient contact numbers.

- Columns:
  - patient\_id: Integer
  - contact\_number: Varchar(20)
- Constraints:
  - Primary Key: (patient\_id, contact\_number)

### 2.3.9 donor\_contact

Stores donor contact numbers.

- Columns:
  - donor\_id: Integer
  - contact\_number: Varchar(20)
- Constraints:
  - Primary Key: (donor\_id, contact\_number)



### 2.3.10 doctor\_contact

Stores doctor contact numbers.

- Columns:
  - doctor\_id: Integer
  - contact\_number: Varchar(20)
- Constraints:
  - Primary Key: (doctor\_id, contact\_number)
  - Foreign Key: doctor\_id references doctor(doctor\_id)

### 2.3.11 organisation\_contact

Stores organization contact numbers.

- Columns:
  - organisation\_id: Integer
  - contact\_number: Varchar(20)
- Constraints:
  - Primary Key: (organisation\_id, contact\_number)
  - Foreign Key: organisation\_id references organisation(organisation\_id)

### 2.3.12 locations

Stores city and state data for address selection.

- Columns:
  - city: Varchar(20)
  - state: Varchar(20)
- Constraints:
  - Primary Key: (city, state)

## 3 Functionalities

### 3.1 Organization Management

- Login: Organizations log in using their ID and password (/login/org).
- Registration: New organizations can register with details like name, head, address, and contacts (/add\_new\_organisation).
- Dashboard: Displays organization stats (donors, doctors, contact info) (/login/org).
- Transplant Review: View completed transplants (/review\_transplants).

### 3.2 Patient Management

- Login: Patients log in using their ID, password, and organization ID (/login/patient).
- Registration: Add new patients with details like name, DOB, insurance, and contacts (/add\_new\_patient).
- Search: Search patients by ID, name, city, state, or all (/search\_patient/<type>).
- View Details: Display patient details, assigned doctors, and transplant history (/patient\_details/<patient\_id>).
- Delete: Remove patients from the system (/delete\_patient/<patient\_id>).

### 3.3 Donor Management

- Registration: Add new donors with details like name, DOB, address, and donated organs (/add\_new\_donor).
- Search: Search donors by ID, name, city, state, or all (/search\_donor/<type>).
- View Details: Display donor details and donated organs (/donor\_details/<donor\_id>).
- Add Organ: Add new organs donated by existing donors (/add\_new\_organ/<donor\_id>).

### 3.4 Doctor Management

- Login: Doctors log in using their ID and password (/doctor\_login).
- Registration: Add new doctors with details like name, DOB, degree, and organ specialization (/add\_doctor).
- View Patients: Doctors can view their assigned patients (/doctor\_login).
- View Doctors: Organizations can view all doctors (/view\_doctor).

### 3.5 Donor Matching and Transplantation

- Search Donors: Find compatible donors by age, city, state, or without constraints (/search\_for\_donor/<patient\_id>).
- Approve Donation: Assign a donated organ to a patient and record the transplantation date (/approve\_donation/<patient\_id>).

### 3.6 Utility Functions

- load\_loc: Loads city and state data for address selection.
- str\_to\_date: Converts string dates to MySQL-compatible format.

## 4 Key Routes and Endpoints

Route	Method	Description
/	GET	Home page displaying organizations and locations.
/login/<user>	GET	Handles organization or patient login.
/add_new_organisation	POST	Registers a new organization.
/review_transplants	GET	Displays transplant history for an organization.
/view_patient	POST	Displays patient search interface.
/search_patient/<type>	GET	Searches patients by specified criteria.
/add_new_patient/<si>	POST	Registers a new patient.
/patient_details/<patie>	GET	Shows patient details and transplant history.
/add_new_doctor/<pa>	POST	Assigns a doctor to a patient.
/delete_patient/<patie>	GET, POST	Deletes a patient record.
/search_for_donor/<pa>	GET, POST	Searches for compatible donors.
/approve_donation/<p>	GET, POST	Approves an organ donation for a patient.
/view_donor	GET, POST	Displays donor search interface.
/add_new_donor	POST	Registers a new donor.
/search_donor/<type>	GET	Searches donors by specified criteria.
/donor_details/<donor>	POST	Shows donor details and donated organs.
/add_new_organ/<don>	POST	Adds a new organ for a donor.
/doctor_login	GET, POST	Handles doctor login and patient list.
/view_doctor	GET, POST	Displays all doctors in an organization.
/add_doctor	GET, POST	Registers a new doctor.

## 5 Initial Data

The database is populated with sample data to facilitate testing and development.

## 5.1 Organisation

- 2 organizations (e.g., AIIMS in New Delhi and Chandigarh).
- Each has a password, head name, and address details.

## 5.2 Organ

- 12 organ types (e.g., Heart, Kidney, Lung, Liver, Cornea).

## 5.3 Doctor

- 7 doctors, each associated with an organ and organization.
- Includes name, DOB, degree, and joining date.

## 5.4 Donor

- 8 donors with name, DOB, address, and organization ID.
- Locations span multiple Indian states (e.g., Uttar Pradesh, Maharashtra).

## 5.5 Donated

- 21 organ donations, linking donors to organs.
- Donation and expiry dates set to 2012-08-31 and 2025-10-31, respectively.

## 5.6 Patient

- 5 patients with name, DOB, insurance, address, and organization ID.
- All have a default password.

## 5.7 Attended\_by

- 13 records linking patients to doctors with demand dates (2010 to 2020).

## 5.8 Contact Tables

- Each entity (patient, donor, doctor, organization) has 2 contact numbers (e.g., +011-123124, +011-123125).

## 5.9 Locations

- 20 Indian cities and states (e.g., Mumbai-Maharashtra, New Delhi-Delhi).

## 5.10 Sample Query

Retrieve donors for organisation\_id = 1:

SELECT \* FROM donor WHERE organisation\_id = 1; Output:

donor	idonor_name	date_of_bir	house	nsotreet_no	city	state	organisat
1	Thanos	2000-04-13	123	12321	Lucknow	Uttar Pradesh	1
2	Ashish Gupta	2000-05-13	123	12321	Mumbai	Maharashtra	1
3	Jai Shankar	2000-06-13	123	12321	New Delhi	Delhi	1
4	Ambar Das	2000-08-13	123	12321	Kolkata	West Bengal	1
5	Ambuja Cement	1989-04-13	123	12321	Bhopal	Madhya Pradesh	1
6	Bangur Cement	1989-05-13	123	12321	Shimla	Himachal Pradesh	1
7	MDH Masale	1989-07-13	123	12321	Jaipur	Rajasthan	1
8	Ankit Oraon	1989-04-13	123	12321	Ranchi	Jharkhand	1

on\_id

## 6 Security Considerations

- Password Storage: Passwords are stored in plain text (e.g., "password"), a major security risk. Use hashing (e.g., bcrypt) in production.
- SQL Injection: The application uses parameterized queries, reducing SQL injection risks, but input validation needs strengthening.
- Session Management: Global variables (organisation\_id, org\_login) are used for state management, which is insecure. Use Flask sessions instead.
- Input Validation: Form inputs lack robust validation, risking errors or vulnerabilities.
- Foreign Keys: Enforced to maintain referential integrity, but cascading deletes are not defined.

## 7 Setup and Installation

### 7.1 Prerequisites

- Python 3.x

- Required Python packages: flask, flask-mysqldb

## 7.2 Installation Steps

### 1. Clone the Repository:

```
git clone <repository_url> cd  
organ_donation_network
```

### 2. Install Dependencies:

```
pip install flask flask-mysqldb
```

### 3. Set up MySQL:

- Create the database:

```
CREATE DATABASE organ_donation_network_db;  
USE organ_donation_network_db;
```

- Copy the SQL script into a .sql file (e.g., setup.sql) and run:

```
mysql -u <username> -p organ_donation_network_db < setup.sql
```

- Update the MySQL password in the Flask app:

```
app.config['MYSQL_HOST'] = 'localhost'  
app.config['MYSQL_USER'] = 'root'  
app.config['MYSQL_PASSWORD'] = '<your_password>'  
app.config['MYSQL_DB'] = 'organ_donation_network_db'  
app.config['MYSQL_PORT'] = 3306
```

### 4. Run the Application:

```
python app.py
```

The app runs on <http://localhost:5000> in debug mode.

## 8 Usage Instructions

1. Access the Home Page: Open <http://localhost:5000> to view organizations and locations.

2. Login:

- Organizations: Use /login/org with ID and password.
- Patients: Use /login/patient with ID, password, and organization ID.



- Doctors: Use /doctor\_login with ID and password.

### 3. Manage Records:

- Register new organizations, patients, donors, or doctors via respective forms.
- Search and view records using provided interfaces.
- Approve transplants by matching donors to patients.

### 4. Review Transplants: Organizations can view transplant history via /review\_transplants.

## 9 Limitations

- Scalability: Global variables and lack of session management may cause issues in multi-user scenarios.
- Error Handling: Limited error handling for database failures or invalid inputs.
- Frontend: Basic HTML templates lack modern UI/UX features.
- Security: Plain-text passwords and lack of authentication mechanisms are major concerns.
- Sample Data: Uses repetitive contact numbers and simplistic data, not suitable for production.
- Database Indexing: No indexing beyond primary keys, which may impact performance with large datasets.

## 10 Future Improvements

- Implement password hashing and secure session management.
- Add input validation and sanitization for all form inputs.
- Enhance the frontend with a modern framework (e.g., React).
- Introduce role-based access control for different user types.
- Add logging and monitoring for system activities.
- Optimize database queries and add indexes for frequently queried fields.
- Implement cascading deletes for dependent records.
- Use realistic and diverse sample data for testing.

## 11 Conclusion

The Organ Donation Network provides a functional platform for managing organ donation processes, supported by a robust MySQL database. While it meets basic requirements, addressing security, scalability, and usability issues will enhance its reliability and user experience, making it suitable for production use.