

ORGANIZER

ORGAN DONATION SYSTEM

A MINI PROJECT REPORT

Submitted by

ASHOK RAJARAM S (950020104008)

RAMAJEYAM R (950020104033)

VENKATESAN M(950020104045)

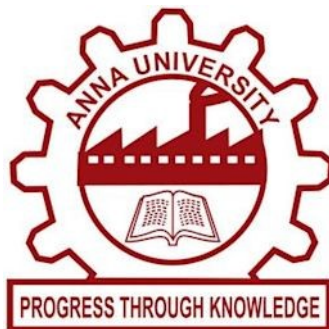
ABDUL FARIS A 9950020104301)

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING



ANNA UNIVERSITY REGIONAL CAMPUS – TIRUNELVELI

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2023

BONAFIDE CERTIFICATE

Certified that this mini project report titled “**ORGAN DONATION SYSTEM**” is the bonafide work of **ASHOK RAJARAM S (950020104008)**, **RAMAJEYAM R (950020104033)**, **VENKATESAN M (950020104045)**, **ABDUL FARIS A (950020104301)** who carried out the project work under my supervision.

Signature of HOD

Dr.K.SARAVANAN

Assistant Professor,
Dept. of Computer Science
and Engineering,
Anna University Regional Campus,
Tirunelveli - 627007

Signature of supervisor

Dr.J.JESU VEDHA NAYAH

Assistant Professor,
Dept. of Computer Science
and Engineering,
Anna University Regional Campus,
Tirunelveli - 627007

Submitted for the CS8611 Viva–Voce examination held on / /2023

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We wish to express our deep sense of gratitude to **Dr.N.SHENBAGA VINAYAGA MOORTHY**, Professor and Dean, Anna University, Regional Campus, Tirunelveli, for his support and encouragement throughout this project work.

We express our sincere thanks to our Head of the Department **Dr.K.SARAVANAN**, Assistant Professor, Department of Computer Science and Engineering, for his motivation, inspiration and encouragement to undertake this work.

We owe our special thanks and gratitude to our internal guide **Dr.J.JESU VEDA NAYAH**, Assistant Professor, Department of Computer Science and Engineering, for her suggestions and supports during the course of our project.

We express our sincere thanks to our coordinator **DR.S.SABENA**, Assistant Professor, Department of Computer and Engineering, for giving us such attention and time.

We are also indebted to all the teaching and non-teaching staff members of our college for helping us directly or indirectly by all means throughout the course of our study and project work.

ASHOK RAJARAM S

RAMAJEYAM R

VENKATESAN M

ABDUL FARIS A

ABSTRACT

This web application acts as an essential role in organ allocation for the patients who are in need of organ for organ transplantation. This helps the recipient to get the required organs from the donors based on the recipients blood group and required organ. This application provides an easy and fast way to search for organs.

This web application's front end is built using HTML, CSS, Bootstrap and Javascript. A person who is willing to donate their organ need to register with the Donor Registration Form which is available in the application. Similarly, a recipient in need of an organ should register with the Recipient Registration form. When a recipient account is created, they can get brief information about the donors sorted based on blood group and required organ match. When a donor is chosen, the donor gets a request notification. .

The objective of this web application is to create awareness about organ donation and to design a framework to maintain necessary information of the Recipients, Donors and report details of the Organ Transplantation. This application will have a login page where the recipient can login and view a sorted list of donors and choose one based on their preference.

The main aim of this web application is to reduce the time to a greater extent to avoid spending time searching for the right donor and the availability of the organ required.

TABLE OF CONTENT

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iv
	LIST OF FIGURES	viii
	LIST OF TABLES	x
	LIST OF ABBREVIATIONS	xi
1	INTRODUCTION	1
	1.1 Objective	2
	1.2 Problem Definition	3
	1.3 Literature Survey	3
2	SYSTEM DESCRIPTION	6
	2.1 SYSTEM REQUIREMENTS	6
	2.1.1 Hardware Requirements	6
	2.1.2 Software Requirements	6
	2.1.3 HTML	7
	2.1.4 CSS	7
	2.1.5 Bootstrap	8
	2.1.6 Django	8
	2.1.7 Javascript	9
	2.1.8 SQLite	9
	2.2 SYSTEM ANALYSIS	10

	2.2.1 Existing System	10
	2.2.2 Proposed Work	12
3	SYSTEM DESIGN	13
	3.1 Architecture Diagram	13
	3.2 Data Flow Diagram	14
	3.2.1 DFD Level 0	14
	3.2.2 DFD Level 1	14
	3.3.3 DFD Level 2	15
	3.3 Usecase Diagram	16
	3.4 Class Diagram	17
	3.5 Entity Relationship Diagram	18
4	SYSTEM DEVELOPMENT	19
	4.1 MODULES	19
	4.1.1 User Module	19
	4.1.2 Donor Filtration	19
	4.1.3 Request and Notification	20
5	SYSTEM IMPLEMENTATION AND TESTING	21
	5.1 IMPLEMENTATION	21
	5.1.1 Donor Module	21
	5.1.2 Recipient Module	23
	5.1.3 Filtration Module	25

5.1.4 Request Module	27
5.1.5 Notification Module	30
5.2 TESTING	31
5.2.1 User Module	32
5.2.2 Donor Filtration	33
5.2.3 Request and Notification	33
CONCLUSIONS	35
APPENDIX	36
REFERENCES	55

LIST OF FIGURES

FIG NO.	NAME	PAGE NO.
3.1	ARCHITECTURE DIAGRAM	13
3.2	DFD LEVEL 0	14
3.3	DFD LEVEL 1	15
3.4	DFD LEVEL 2	15
3.5	USECASE DIAGRAM	16
3.6	CLASS DIAGRAM	17
3.7	ER DIAGRAM	18
5.1	HOME PAGE	21
5.2	DONOR REGISTRATION PAGE	22
5.3	LOGIN PAGE	22
5.4	DONOR PROFILE	23
5.5	RECIPIENT REGISTRATION PAGE	23
5.6	LOGIN ERROR PAGE	24
5.7	RECIPIENT PROFILE	24
5.8	DONOR FILTRATION	25
5.9	FILTERED RESULT	25
5.10	ALL DONOR LIST	26
5.11	DONOR SELECTION	26
5.12	REQUEST	27

5.13	DONOR REQUEST VIEW	27
5.14	PENDING REQUESTS	28
5.15	ACCEPTED REQUESTS	28
5.16	CONTACT SHARING	29
5.17	ENDING REQUEST DETAILS	29
5.18	DONOR REGISTER NOTIFICATION	30
5.19	DONOR REQUEST NOTIFICATION	30
5.20	RECIPIENT CONFIRMATION NOTIFICATION	31

LIST OF TABLES

TABLE NO.	NAME	PAGE NO.
5.1	USER MODULE	32
5.2	DONOR FILTRATION	33
5.3	REQUEST AND NOTIFICATION	33

LIST OF ABBREVIATIONS

DFD	DATA FLOW DIAGRAM
ER	ENTITY RELATIONSHIP
HTML	HYPER TEXT MARK-UP LANGUAGE
CSS	CASCADING STYLE SHEETS
SQL	STRUCTURED QUERY LANGUAGE
GUI	GRAPHICAL USER INTERFACE
CSRF	CROSS-SITE REQUEST FORGERY
ORM	OBJECT RELATIONAL MAPPING
MVT	MODEL-VIEW-TEMPLATE

CHAPTER 1

INTRODUCTION

Organ donation is the process when a person allows an organ of their own to be removed and transplanted to another person, legally, either by consent while the donor is alive. Donation may be for research or, more commonly, healthy transplantable organs and tissues may be donated to be transplanted into another person.

Common transplantations include corneas, kidneys, heart, liver, pancreas, intestines, lungs, bones, bone marrow and skin are Some organs and tissues can be donated by living donors. Organ donors are usually dead at the time of donation, but may be living. For living donors, organ donation typically involves extensive testing before the donation, including psychological evaluation to determine whether the would-be donor understands and consents to the donation. On the day of the donation, the donor and the recipient arrive at the hospital, just like they would for any other major surgery.

- Organ donation is a noble act of transplanting healthy organs from a donor to a patient receiver.
- Human body organs and tissues that function properly are collected and transplanted into the patient's bodies to save their lives.
- Every day, there are approximately 107,000 people on the waiting list nationally for an organ.
- Organ transplantation is the only option to save lives of patients affected by terminal organ failures and improve their quality of life.

1.1 OBJECTIVE

India is struggling with acute shortage of organs for transplantation and there is dire need to increase awareness on organ donation. According to a report, more than 5,00,000 people are waiting for organ transplant in the country and only a handful of 3,500 transplants are performed annually.

The shortage of organs for transplant is a universal issue; However the situation is grave in India. With less than one person per million people, the organ donation ratio in India is estimated to be one of the lowest in the world as per WHO report. Due to prevalence of myths and superstitions, many do not think about donating their organs even after death.

Awareness on organ donation is therefore a priority. By creating Awareness among people, many voluntary donors can be found to donate their organs.

- The primary objective is to promote awareness of life- saving solid organ transplants.
- The secondary objective is to promote awareness of tissue and life-enhancing transplants.
- Engender a greater willingness among the public to donate their organs and tissue.
- The main aim of this web application is to reduce the time to a greater extent to avoid spending time searching for the right donor and the availability of the organ required.
- To develop a responsive website to manage the organ donation system.

1.2 PROBLEM DEFINITION

- India is struggling with acute shortage of organs for transplantation and there is dire need to increase awareness on organ donation.
- In India every year nearly 5,00,000 people die because of the non-availability of organs and this number is expected to grow due to scarcity of organ donor.
- At present most organs for transplants come from living donors, whereby these donors may legally only be the immediate family.
- Platform to identify donor who are willing to donate their organ like blood donation.
- A vast amount of time is wasted in order to search for the right donor.
- So this website has it's main focus to create good communication between donor and recipient, so that they transplant the organ in the location that they both prefer.

1.3 LITERATURE REVIEW

WEBSITE NAME: NOTTO [1]

National Organ and Tissue Transplant Organization (NOTTO) is a National level organization set up under Directorate General of Health Services, Ministry of Health and Family Welfare, Government of India located at 4th and 5th Floor of Institute of Pathology (ICMR) Building in Safdarjung Hospital New Delhi.

Advantages:

- Both donor and hospital registration is available.
- More information about Organ Transplantation.

Disadvantages:

- User Interface is not much appealing.
- Complex Registration process for donors.

WEBSITE NAME: ORGANINDIA[2]

The Organ Receiving and Giving Awareness Network(ORGAN) India, is an initiative that was launched in March 2013 by The Parashar Foundation,(a Delhi-based NGO) to address the dismal state of deceased organ donation in India. They seek to remedy this shortage of organ donors, and help create an ecosystem to facilitate organ donation in India.

Advantages:

- Creates awareness using social media platforms to increase donor pledges.
- Information packed.
- Donors are given a donor card.

Disadvantages:

- Donor registration process is slow and takes time.
- Focus only on awareness and not in the process.
- There is no recipient registration or Hospital Registration.

WEBSITE NAME: ORGANIZE[3]

The organ donation system is grossly inefficient, failing to recover and transplant as many as 28,000 lifesaving organs every year. The problem is a system of unaccountable, government monopoly contractors with a history of failure to recover lifesaving organs, fatal lapses in patient safety, obstructionism, and fraud, waste and abuse. These include the national contractor that runs the organ procurement transplantation network (UNOS) and many local contractors called organ procurement organizations (OPOs).

Advantages:

- Creates awareness using social media platforms to increase donor pledges.
- More information about Organ Transplantation

Disadvantages:

- Donor registration process is slow and takes time.
- Focus only on awareness and not in the process.

CHAPTER 2

SYSTEM DESCRIPTION

2.1 SYSTEM REQUIREMENT

System requirements are the configuration that a system must have in order for a hardware or software application to run smoothly and efficiently. Failure to meet these requirements can result in installation problems or performance problems. The former may prevent a device or application from getting installed, whereas the latter may cause a product to malfunction or perform below expectation or even to hang or crash.

2.1.1 HARDWARE REQUIREMENTS

- Processor : Intel core i3.
- Memory : 8GB RAM.
- Hard Disk : up to 200MB of available space may be required.

2.1.2 SOFTWARE REQUIREMENTS

- Operating System : Windows
- Architecture : Client Server
- Front-end : HTML, CSS, Javascript
- Back-end : Python(Django)
- Database : SQLite

2.1.3 HTML

HTML stands for Hyper Text Markup Language. It is the standard markup language for creating Web pages. It describes the structure of a Web page. It consists of a series of elements tell the browser how to display the content. HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc. Each tag is used for different purpose. We use HTML to put our data on webpage. It is used to put any type of text on webpage

2.1.4 CSS

CSS stands for Cascading Style Sheet. It is used to change the appearance of the content of the webpage. CSS is a style sheet language used for describing the presentation of a document written in a mark-up language like HTML.

There are three types of CSS:

Inline is used by style attribute within the HTML tag. We can use this type of CSS on any HTML tag just by using the style attribute. If we want to apply the CSS on a smart part or on a specific tag then we prefer this type of CSS.

Internal is used by using by typing the CSS code inside the head part of the HTML tag within the style tag. To access any HTML tag in internal CSS we can use some selectors like id, class.

External is used when we want to type the code in separate file to reduce the complexity of the code. We can easily link that external file by giving the reference or address of the file in the head part using style tag.

2.1.5 BOOTSTRAP

Bootstrap is the most popular HTML, CSS and JavaScript framework for developing a responsive and mobile friendly website. It is absolutely free to download and use. It is a front-end framework used for easier and faster web development. It includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many others. It can also use JavaScript plug-ins. It facilitates you to create responsive designs

2.1.6 DJANGO

Django is an open-source web framework written in Python that provides a solid foundation and a set of tools for building web applications efficiently. It follows the model-view-template (MVT) architectural pattern, which is similar to the model-view-controller (MVC) pattern. Django simplifies the process of web development by offering a high level of abstraction and automating many common tasks. It includes a robust ORM for database interaction, allowing developers to work with the database using Python code instead of raw SQL queries. Django's URL routing system provides a flexible way to map URLs to corresponding views, and its template engine enables the creation of dynamic HTML pages. The framework also includes built-in features for user authentication, session management, form handling, and an administrative interface. Django's scalability, security features, and vibrant community make it a popular choice for building a wide range of web applications, from simple websites to complex, high-traffic platforms.

2.1.7 JAVASCRIPT

It is client side scripting language. When we want to run any script on the browser then we use JavaScript as a medium. It is also used for applying the validations over the webpage like checking that a field is blank or not. It is also used for interacting with the user like inputting a value from the user. It is also used for accessing the properties of various elements of the webpage as well as the browser. To use JavaScript we have to write the code in head part of the HTML section inside the script tag. We can access the properties of various elements of webpage by their name or by their id's.

2.1.8 SQLITE

SQLite is an embedded, server-less relational database management system. It is an in-memory open-source library with zero configuration and does not require any installation. Also, it is very convenient as it's less than 500kb in size, which is significantly lesser than other database management systems. SQLite is one of the most popular and easy-to-use relational database systems. It possesses many features over other relational databases. SQLite is an open-source software. The software does not require any license after installation.

SQLite is serverless as it doesn't need a different server process or system to operate. It facilitates you to work on multiple databases on the same session simultaneously, thus making it flexible. It is a cross-platform DBMS that can run on all platforms, including macOS, Windows, etc. SQLite doesn't require any configuration. It needs no setup or administration.

2.2 SYSTEM ANALYSIS

System analysis is a review of a technological system, like a software package, for troubleshooting, development or improvement purposes. Through in-depth analysis, analysts can uncover errors in code, accessibility issues for end-users or design incompatibilities.

Effective review of systems and processes can help organizations provide quality and functional offerings that meet user needs. A systems analysis is one method technology professionals can use to identify concerns and suggest improvements to the systems businesses and organizations use in their operations. Learning what a systems analysis is and how it works can help you better understand its importance and necessity. In this project, we define a systems analysis, list some tools analysts can use to perform their analysis, explain why this type of analysis is important and offer tips for conducting effective reviews.

2.2.1 EXISTING SYSTEM

Manual Organ Donation System

Organ donation and transplantation services portray a sub-system of modern healthcare organizations, which is a complex adaptive system. Across the world, there is a marked gap between the numbers of available organs for transplant and potential recipients in the waiting list. To date, researchers have struggled to comprehend the reasons for these disparities, yet the subject received relatively little attention from the system's perspective.

The outcome of organ transplantation requires the coordinated effort of many actors like donors, recipients, families, physicians, transplant coordinators and other hospital staff. They may belong to cross-functional departments and institutions, often operating with different motivations and objectives. In this study, we use Situation-Actor-Process (SAP) and Learning-Action-Performance (LAP) inquiry model, to systematically inquire the organ donation in India. With SAP-LAP, we attempt to understand the complexities and interactions among the current situation, involved actors and processes affecting organ donation at the macro- and micro-level of policy formation. The SAP part brings an insight into the present condition of the organ donation system in India. A model representing flexible interaction among situation, actors and processes is developed for a better systemic understanding. Then, LAP fetches the learning followed by the suggested actions that need to be taken for improving the performance of the organ donation system.

Through a case study conducted in an Indian hospital, we explore the reasons for operational inefficiencies at the micro-level. The study identified non-value-adding activities like waiting, excess motion, inappropriate processing and defects; and value-adding activities like training intensivist, training transplant coordinator, raising awareness among family members, excess inventory and use of information and communication technology.

The research brings up multiple, self-adjusting, unpredictable and interacting pathways that lead from a potential organ donor to an organ receiver, thus targeting process improvement through a holistic approach.

2.2.2 PROPOSED WORK

The Online Organ Donation Management System (OODMS) is developed mainly for general hospitals (GH), clinics and other health centres to manage the donor registration and user maintenance. The public can retrieve information about organ donation in this web site. People who interested can register themselves through this system. The application will be processed by the administrator and each donor will receive feedback about their application status. Furthermore, the authorized user's account will be maintained by the administrator.

The donor record will be managed by four main users such as administrator, doctor, medical assistant and management staff. Only administrator has the authority and privileges to print organ list report and total donation report according to district from this system. The methodology of this system is Structured System Analysis and Design (SSADM). An analysis study has been done based on the current manual system and all the problems statements and requirements have been identified. Moreover, OODMS is three tier architecture system which involves client tier, business tier and database management tier.

The interfaces for OODMS have been designed according to the requirement and needs of the current market. Rather than that, this system also has been tested and evaluated in real life. This Online Organ Donation Management System will help to improve the performance of current situation and overcome the problems that arise nowadays. In the healthcare context, organ transplantation has raised to great importance in the last years. Improvements in medical techniques and pharmacological anti-reject therapies have made transplantation a powerful and valid way to treat diseases.

CHAPTER 3

SYSTEM DESIGN

System Design is the core concept behind the design of any distributed systems. System Design is defined as a process of creating an architecture for different components, interface and module of the system and providing corresponding data helpful in implementation of elements.

3.1 Architecture Diagram

Architecture diagram is the process of creating visual representations of software system components. In a software system, the term architecture refers to various functions, their implementations, and their interactions with each other.

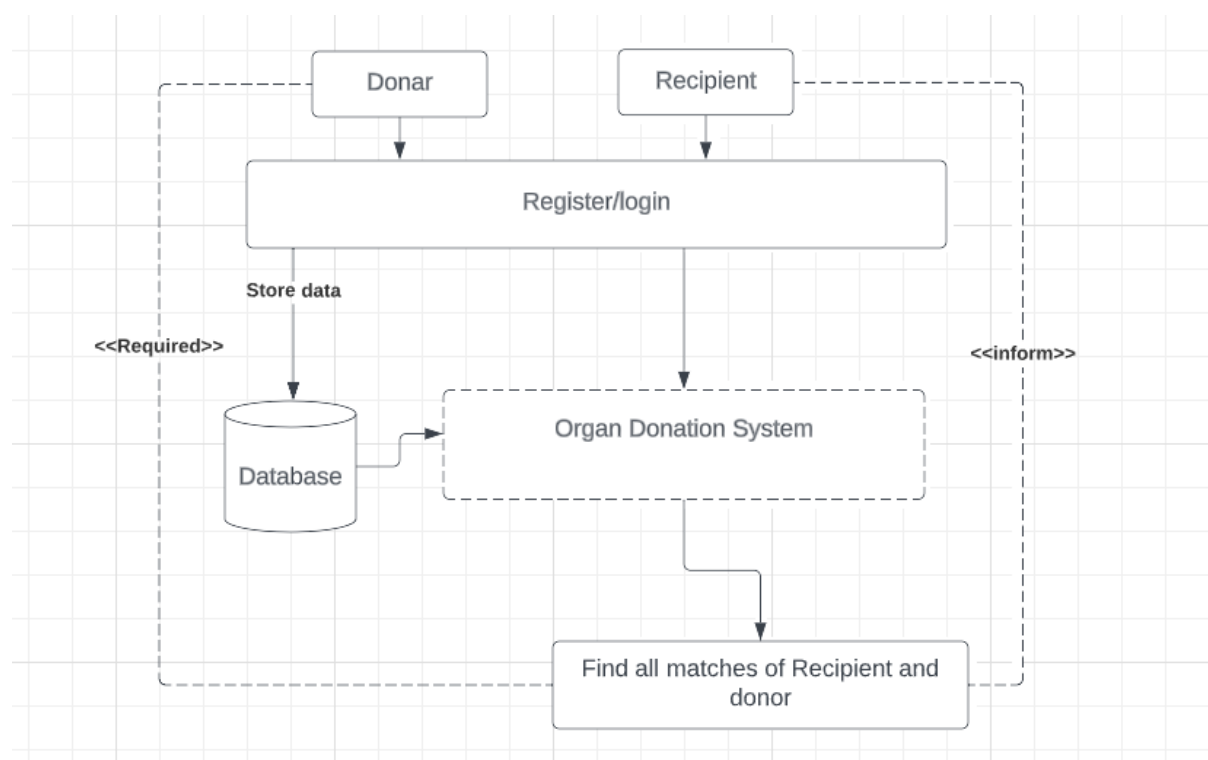


Fig 3.1 Architecture Diagram

3.2 Data Flow Diagram

A DFD diagram is a way of representing a flow of data through a process or a system. The DFD also provides information about the outputs and inputs of each entity and process itself. A data flow diagram has no control flow there are no decision rules and no loops.

3.2.1 DFD Level 0

DFD Level 0 is also called a Context Diagram. It's a basic overview of the whole system or process being analysed or modelled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities.

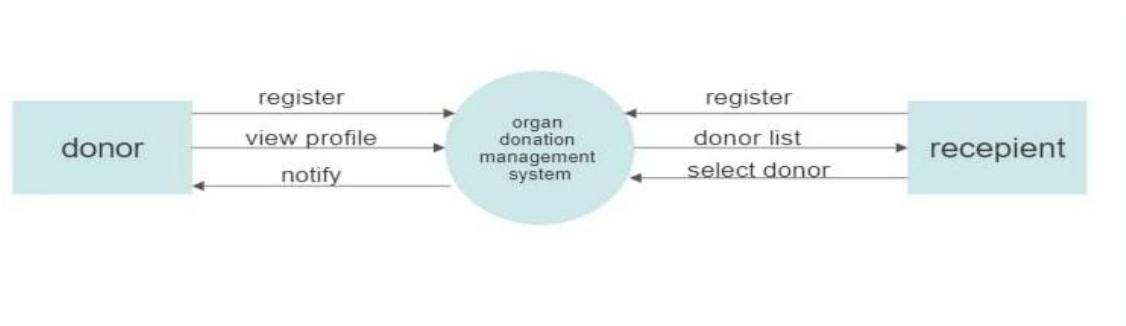


Fig 3.2 DFD level 0

Fig 3.2 shows the basic data flow between the system, donor and recipient.

3.2.2 DFD Level 1

DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its subprocesses.

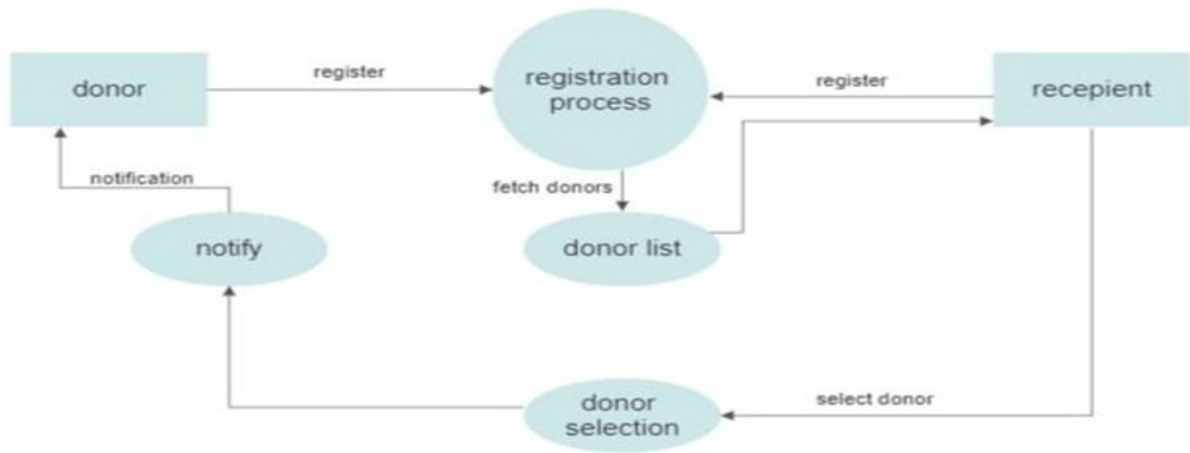


Fig 3.3 DFD level 1

Fig 3.3 gives the data that flow from the system to both donor and recipient to link them to the same connection through the system database.

3.2.3 DFD Level 2

DFD Level 2 then goes one step deeper into parts of Level 1. It may require more text to reach the necessary level of detail about the system's functioning.

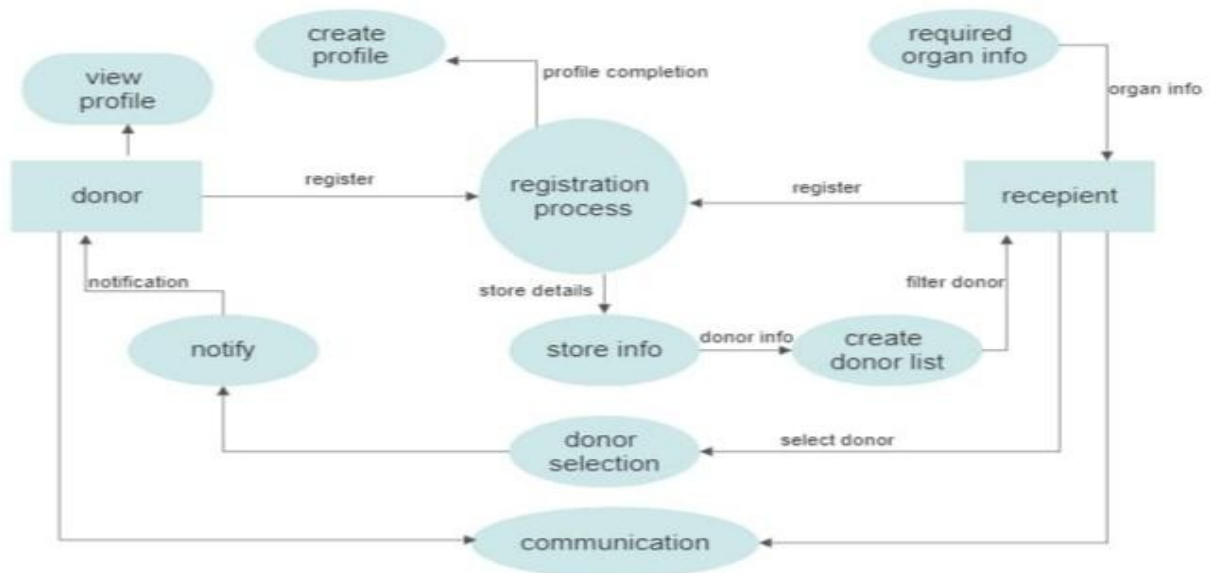


fig 3.4 DFD LEVEL 2

fig 3.4 the complete data flow of the donor donation system with the users.

3.3 Usecase Diagram

A use case diagram is a graphic depiction of the interactions among the elements of the system. A use case is a methodology used in system analysis to identify, clarify and organize system requirements. Use case diagrams are employed in UML, a standard notation for the modelling of real-world objects and systems.

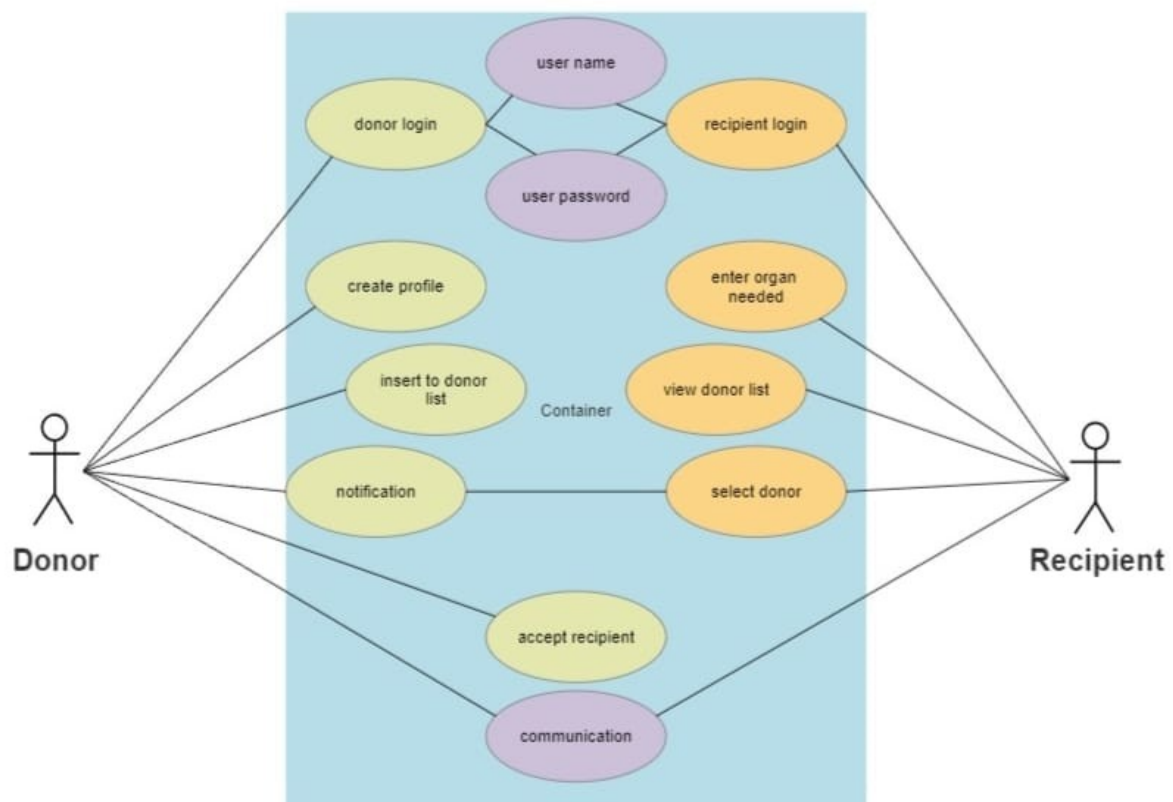


Fig 3.5 Usecase diagram for organ donation system

fig 3.5 donor and recipient are the actors who has the set of steps to perform in the organ donation system. so there details can be stored.

3.4 Class Diagram

The main purpose of class diagrams is to build a static view of an application. It is the only diagram that is widely used for construction, and it can be mapped with object-oriented languages

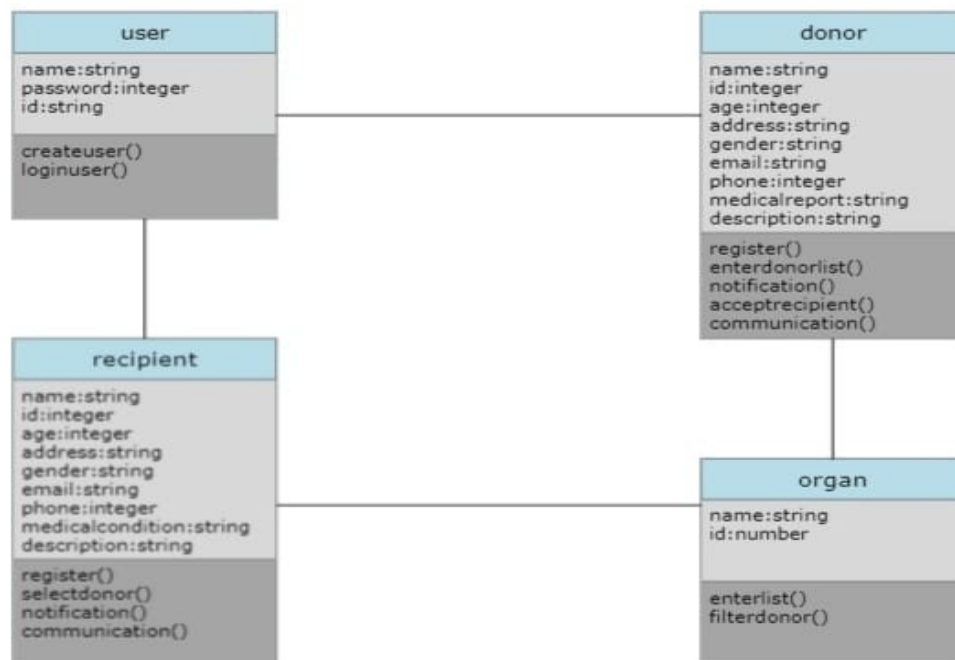


Fig 3.6 class diagram for organ donation system

Fig 3.6 defines the process of each entity and attributes to invoke them in the system and also to make schemas in database to store the required data collected from client.

3.5 Entity Relationship Diagram

An Entity Relationship Diagram (ERD) shows the relationships of entity sets stored in a database. In other words, ER diagrams illustrate the logical structure of databases. It looks like a flow chart. It has the specialized symbols and the meaning of these symbols, that make it unique.

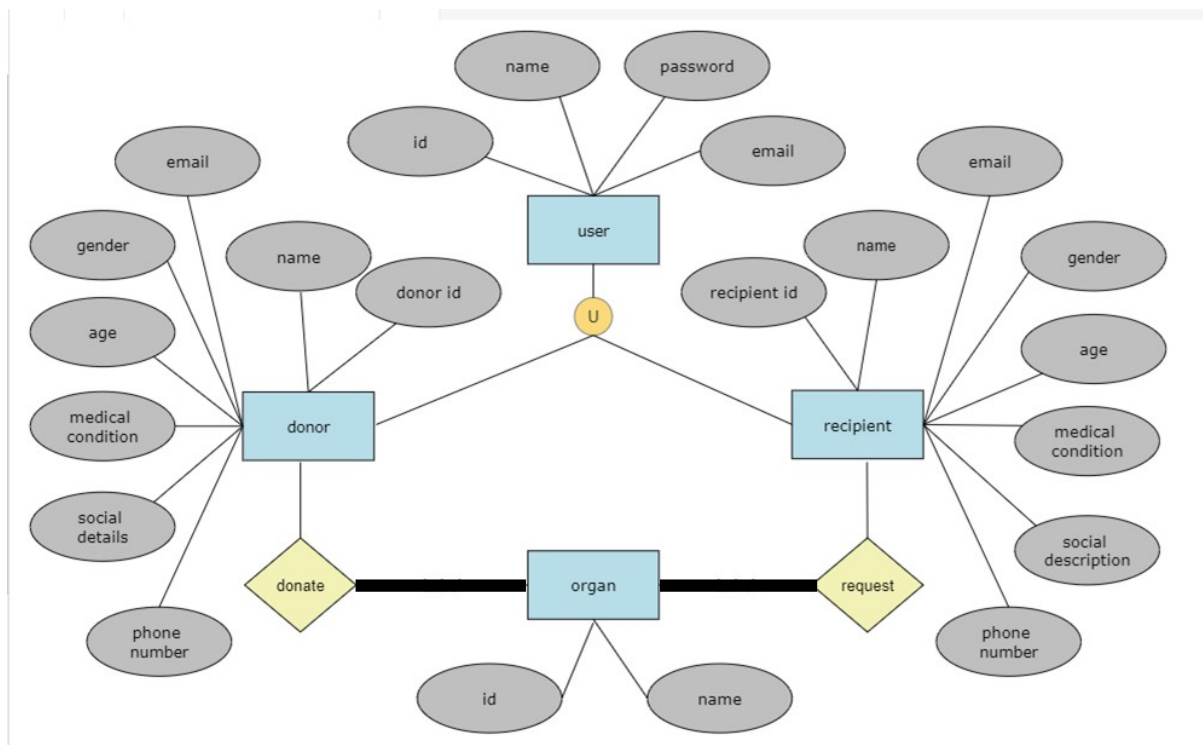


Fig 3.7 Entity relationship diagram

Fig 3.7 the detail of both recipient and donor are the attributes are there own. The donor linked to the system by donating the organ and recipient by requesting the organ.

CHAPTER 4

SYSTEM DEVELOPMENT

4.1 MODULES

One or more independently developed modules make up a program. An enterprise-level software application may contain several different modules, and each module serves unique and separate business operations. Modules make a programmer's job easy by allowing the programmer to focus on only one area of the functionality of the software application.

4.1.1 User Module

- This module takes care of the registration, login and logout of both the donors and recipients. It captures information from donor and recipient from the registration form and stores it in the database.
- Donor can view their profile when logged in.
- Recipients can view donors when they are logged into the application.
- This module accounts for 40% of the project.

4.1.2 Donor Filtration

- This module takes care of the filtering of donor information according to the blood group and organ required by the recipient when the recipient logs in. It displays the filtered donors list in the home page of the recipient.
- This module accounts for 30% of the project.

4.1.3 Request and Notification

- This module includes tools to send request to the donor chosen by the recipient. The request can be viewed by the donor and the donor can either accept or reject the request. After the approval of the request, the contact details of the donor and the recipient are shared with each other for further communication.
- This module accounts for the remaining 30% of the project.

CHAPTER 5

SYSTEM IMPLEMENTATION AND TESTING

5.1 IMPLEMENTATION

Implementation planning is a process in project management that entails creating step-by-step instructions for completing projects. The purpose of this process is to inform members of a project team of the concrete actions and individual tasks required to achieve the team's strategic goals.

5.1.1 Donor Module

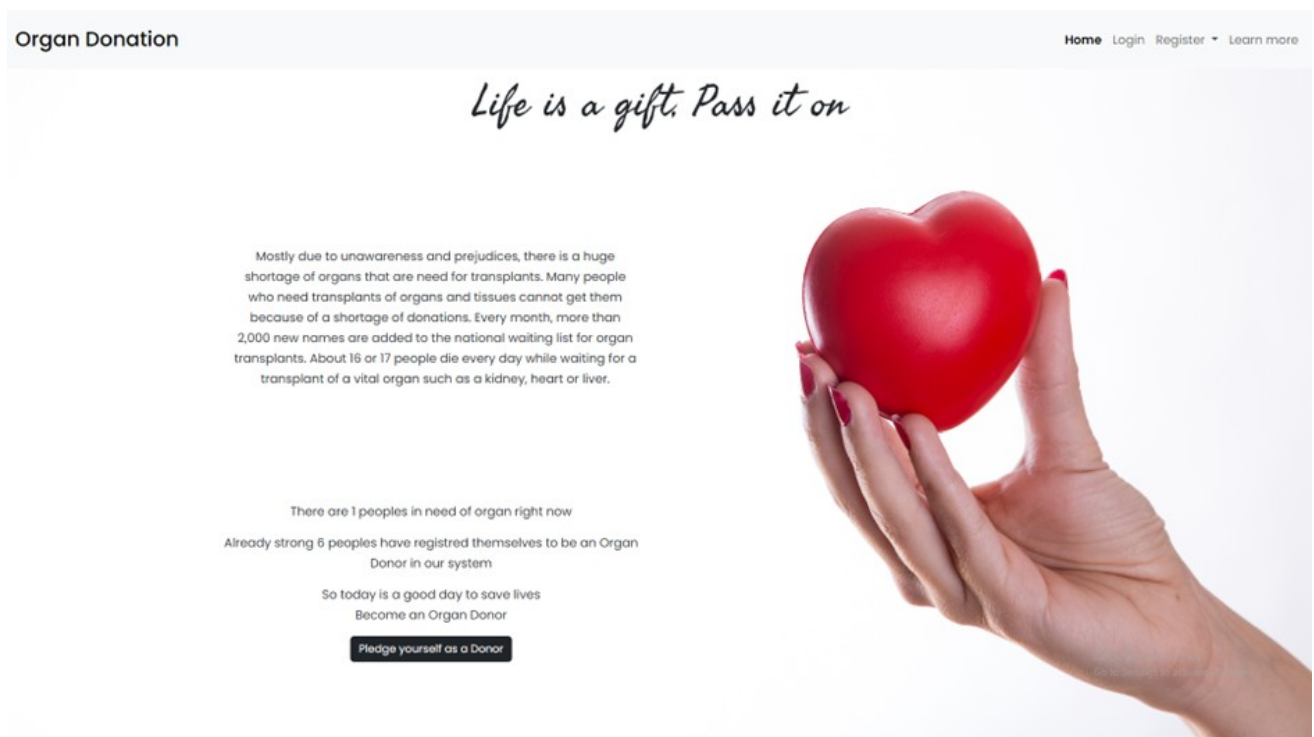
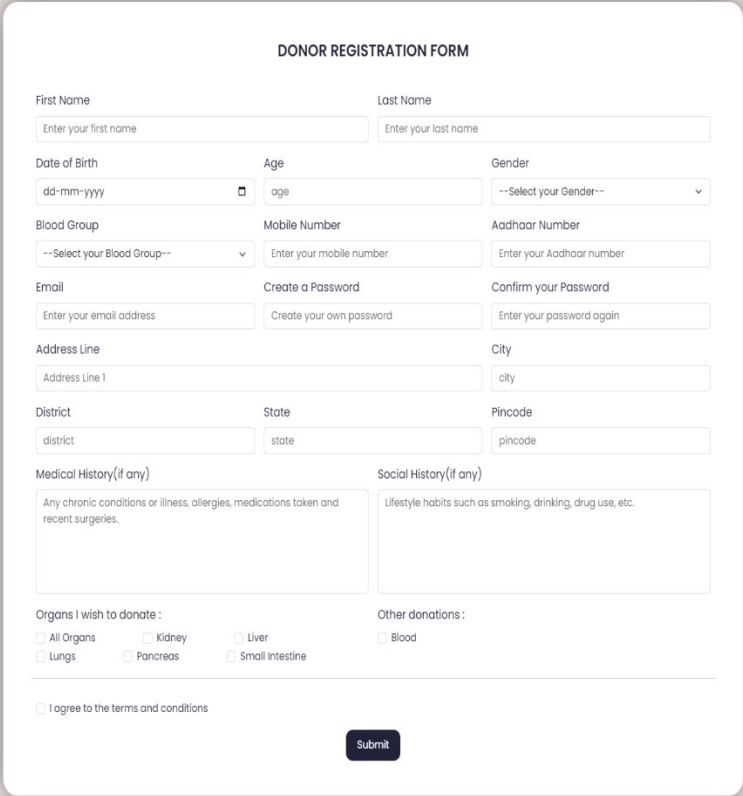


Fig 5.1 Home page of the organ Donation system

Fig 5.1 is the home page common for both donor and recipient.



DONOR REGISTRATION FORM

First Name Last Name

Date of Birth Age Gender

Blood Group Mobile Number Aadhaar Number

Email Create a Password Confirm your Password

Address Line City

District State Pincode

Medical History(if any)

Social History(if any)

Organs I wish to donate : ☐ All Organs ☐ Kidney ☐ Liver ☐ Lungs ☐ Pancreas ☐ Small Intestine

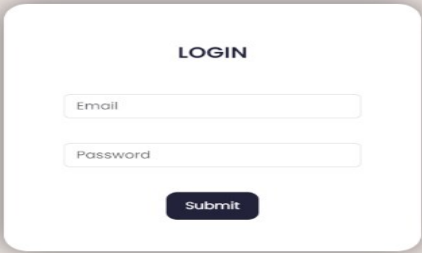
Other donations : ☐ Blood

☐ I agree to the terms and conditions

Activate Windows
Go to Settings to activate Windows.

Fig 5.2 Registration page for donor

Fig 5.2 is the donor registration page where the necessary details of the donor are collected and stored in the database of the system.



LOGIN

Email

Password

Fig 5.3 Login page

Fig 5.3 is the login page for both donor and recipient.

Email : aakashjune2005@gmail.com

Date of Birth : June 15, 2005

Age : 17

Gender : male

Blood Group : O+

Mobile No : 9344175382

Aadhaar No : 987698769876

Address Line : 87E South Ponnagaram

City : Rajapalayam

District : Virudhunagar

State : Tamil Nadu

Pincode : 626113

Organs willing to donate :

Liver Pancreas

Medical History : None

Social History : None

Fig 5.4 Donor profile page

5.1.2 Recipient Module

RECIPIENT REGISTRATION FORM

First Name

Enter your first name

Last Name

Enter your last name

Date of Birth

dd-mm-yyyy

Age

age

Gender

--Gender--

Blood Group

--Blood Group--

Mobile Number

mobile number

Aadhaar Number

aadhaar number

Email

email address

Create a Password

Create your own password

Confirm your Password

Enter your password again

Address Line

Address Line 1

City

city

District

Ariyalur

district

State

state

Pincode

pincode

Organ for transplantation

--Organ--

Current Medical Condition

Recipient's current medical condition and the reason for the transplantation.

Medical History(if any)

Any pre-existing conditions, previous surgeries and ongoing medical treatments.

Submit

Activate Windows
Go to Settings to activate Windows.

Fig 5.5 Recipient registration page

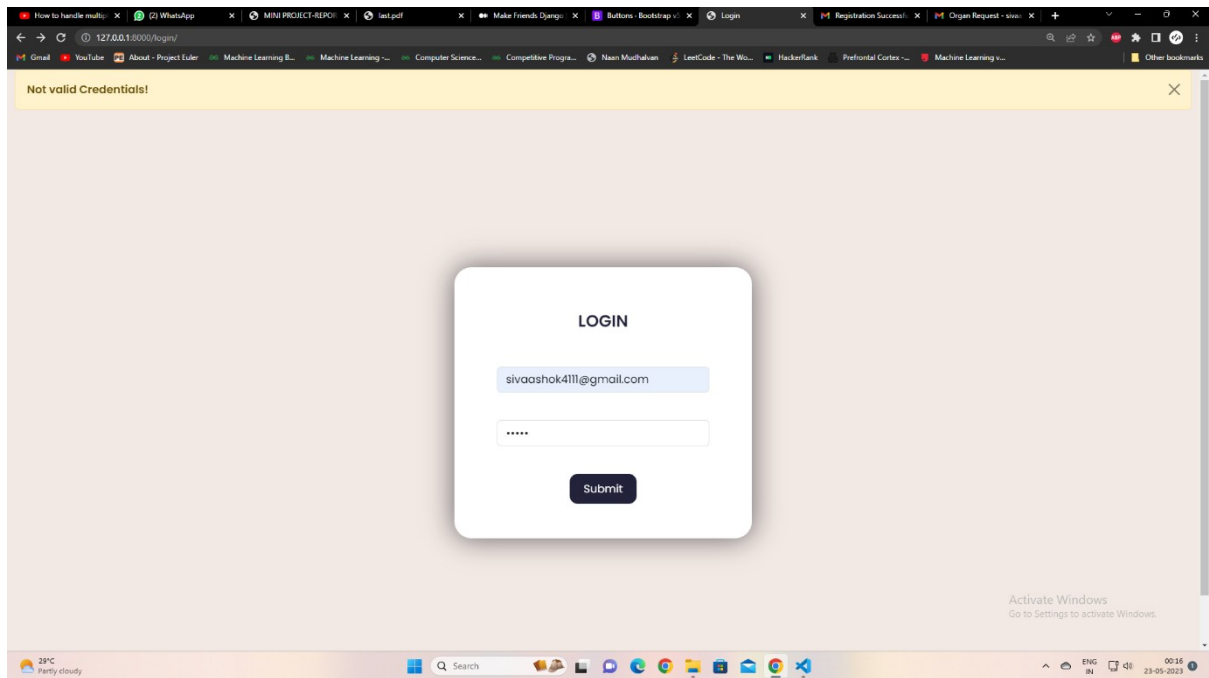


Fig 5.6 Login error page

Fig 5.6 is the error page when there is a mistake in the username or password.

Organ Donation		Profile Donors Requests Logout Learn more
Name : Test T		
Email :	test@gmail.com	
Date of Birth :	Jan. 1, 2001	
Age :	22	
Gender :	male	
Blood Group :	B+	
Mobile No :	9876543210	
Aadhaar No :	3698521473698521	
Address Line :	test address	
City :	test city	
District :	test district	
State :	test state	
Pincode :	123456	
Organ required for transplantation :	Kidney	
Medical Condition :	None	
Medical History :	None	

Fig 5.7 Recipient profile page

Fig 5.7 shows the profile page of the Recipient who has been registered into the system.

5.1.3 Filtration Module

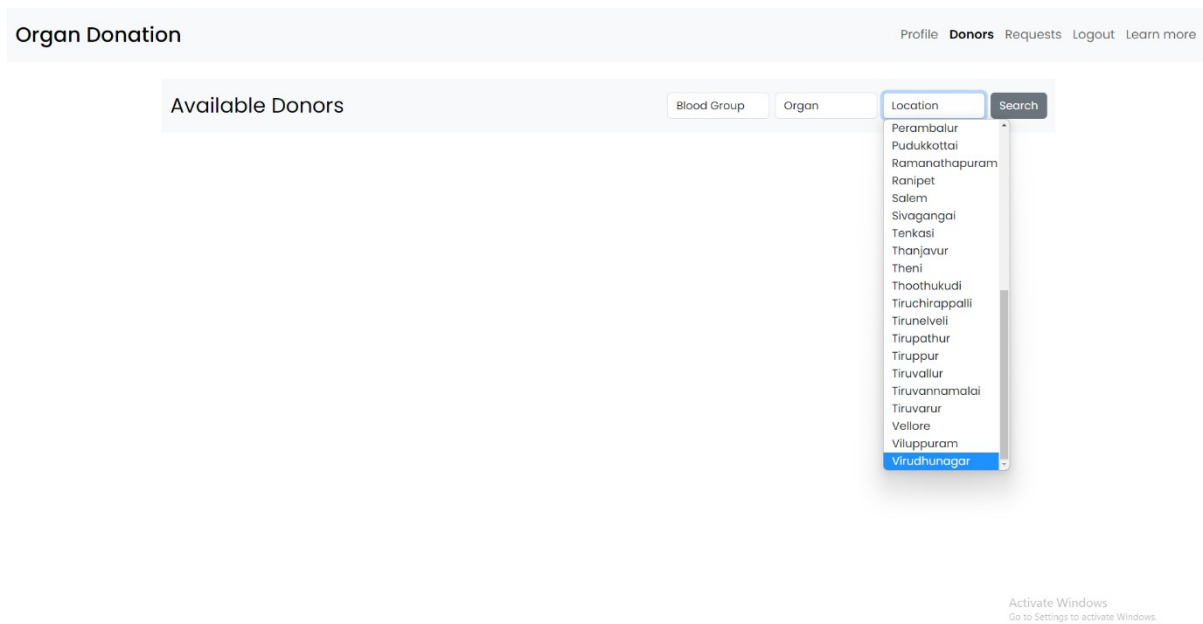


Fig 5.8 Donor Filtration page

Fig 5.8 shows that the recipient can filter donor by organ, blood group and location.

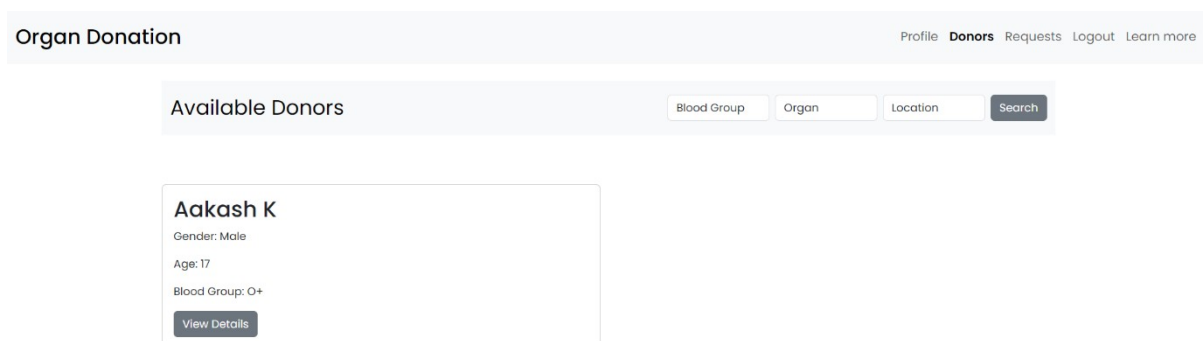


Fig 5.9 Filtered Result

Fig 5.9 shows the result of donor based on the filtered result

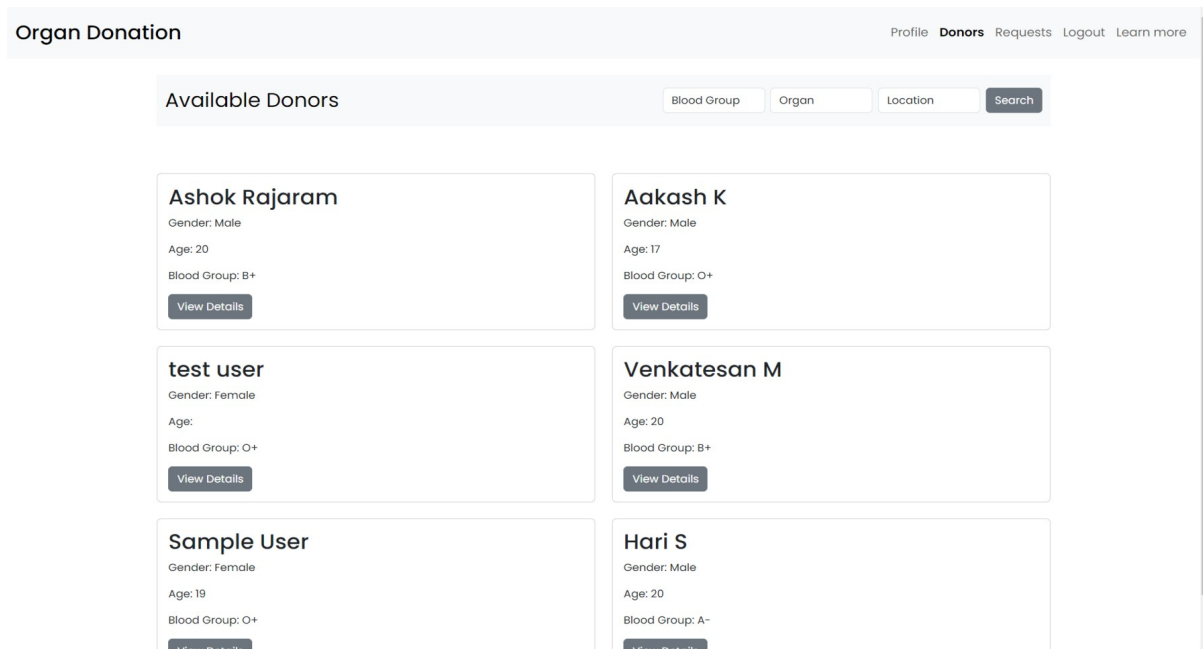


Fig 5.10 All donor list

Fig 5.10 shows the general donor list to the recipient.

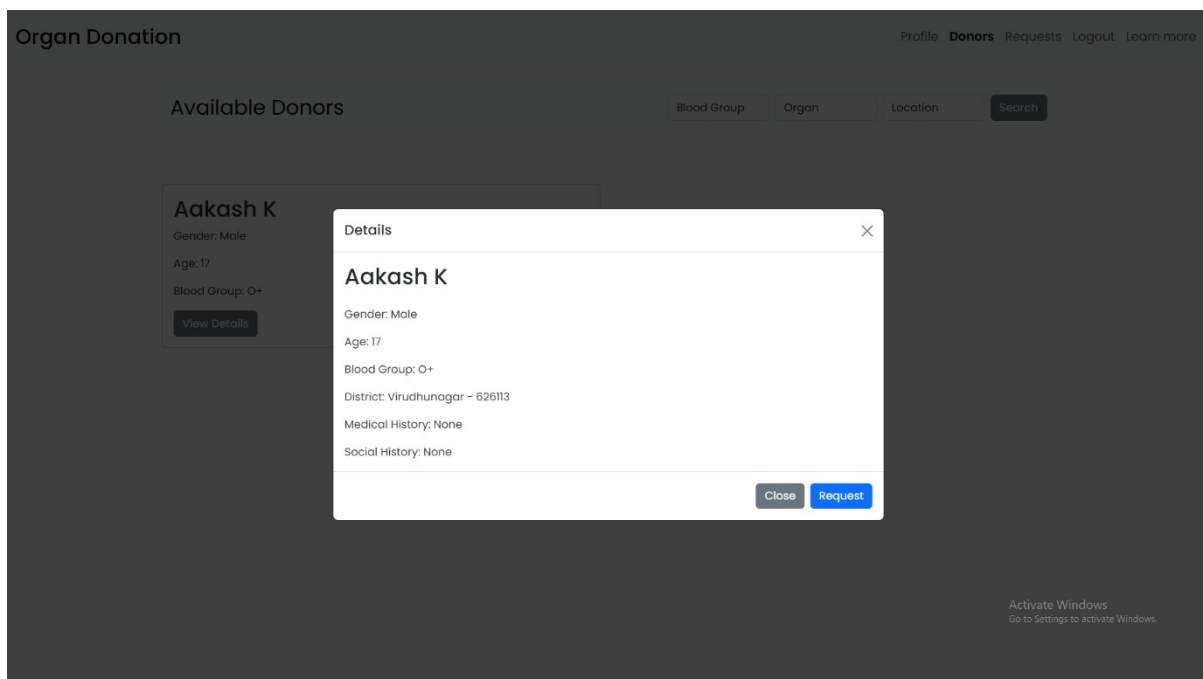


Fig 5.11 selection of donor

Fig 5.11 shows how the details of the donor will be visible to the recipient to send requests to the donor.

5.1.4 Request Module

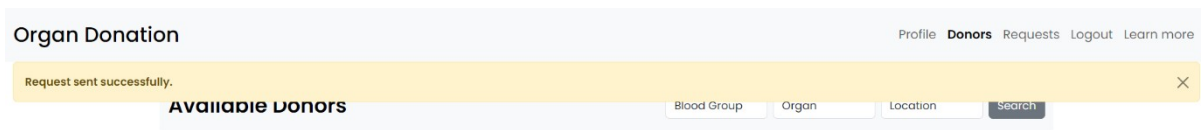


Fig 5.12 Request sent to the donor

Fig 5.12 displays when the request has successfully sent the donor that the recipient have selected.

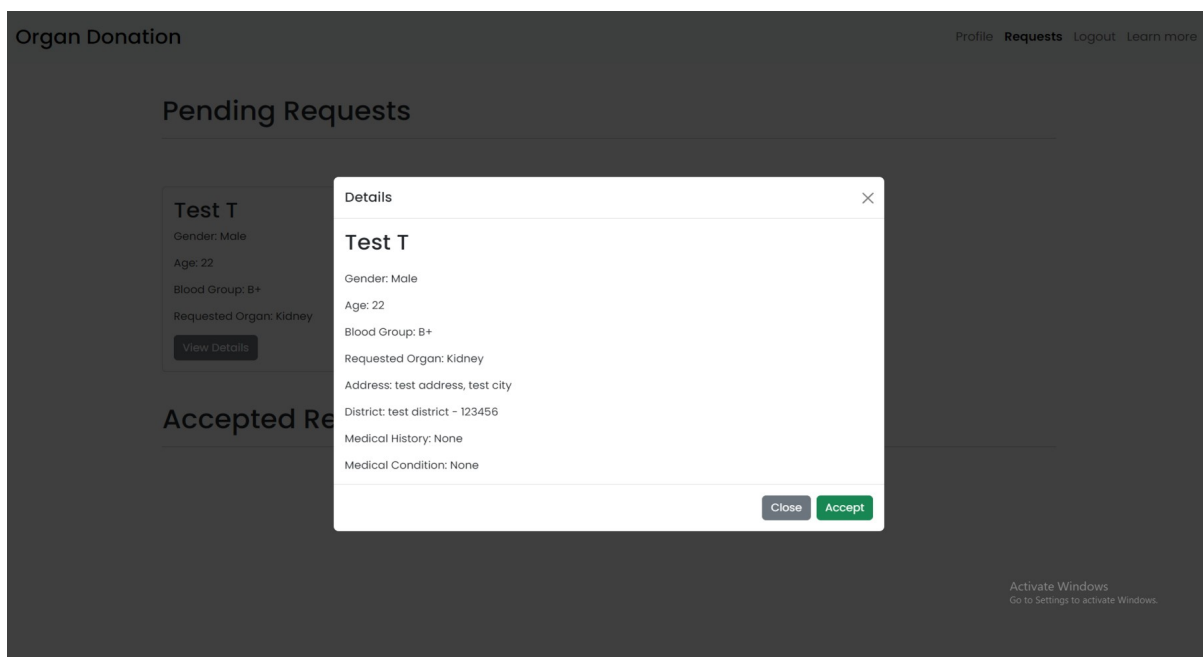


Fig 5.13 donor request view

Fig 5.13 shows how the donor see the request if a recipient has requested him for the organ he is willing to donate.

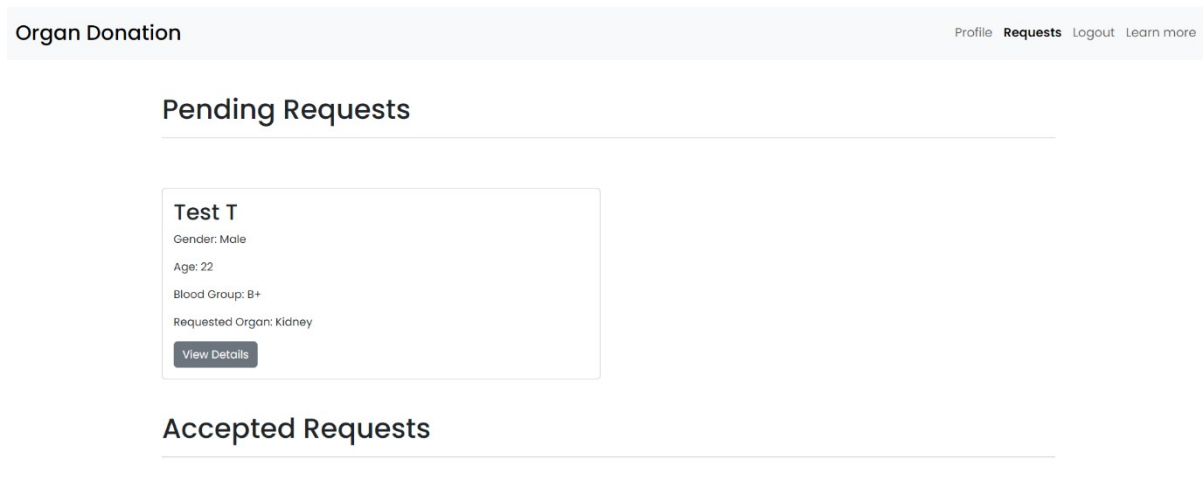


Fig 5.14 Pending requests

Fig 5.14 displays the pending requests that comes to donor.

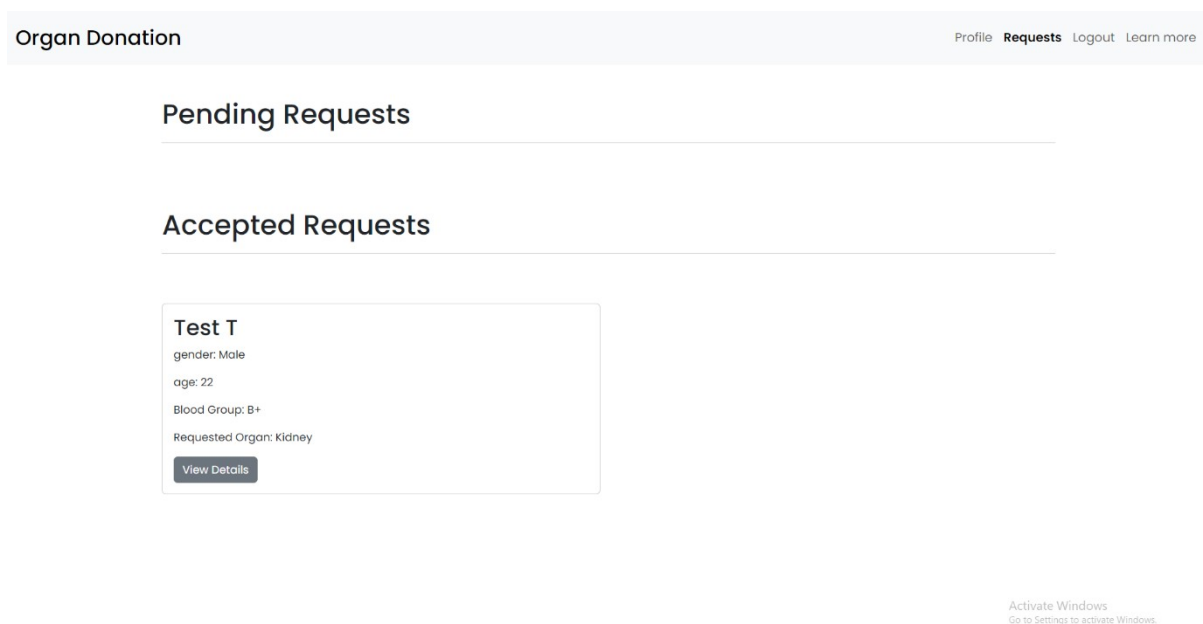


Fig 5.15 Accepted requests

Fig 5.15 displays the accepted requests that donor has accepted the organ to donate.

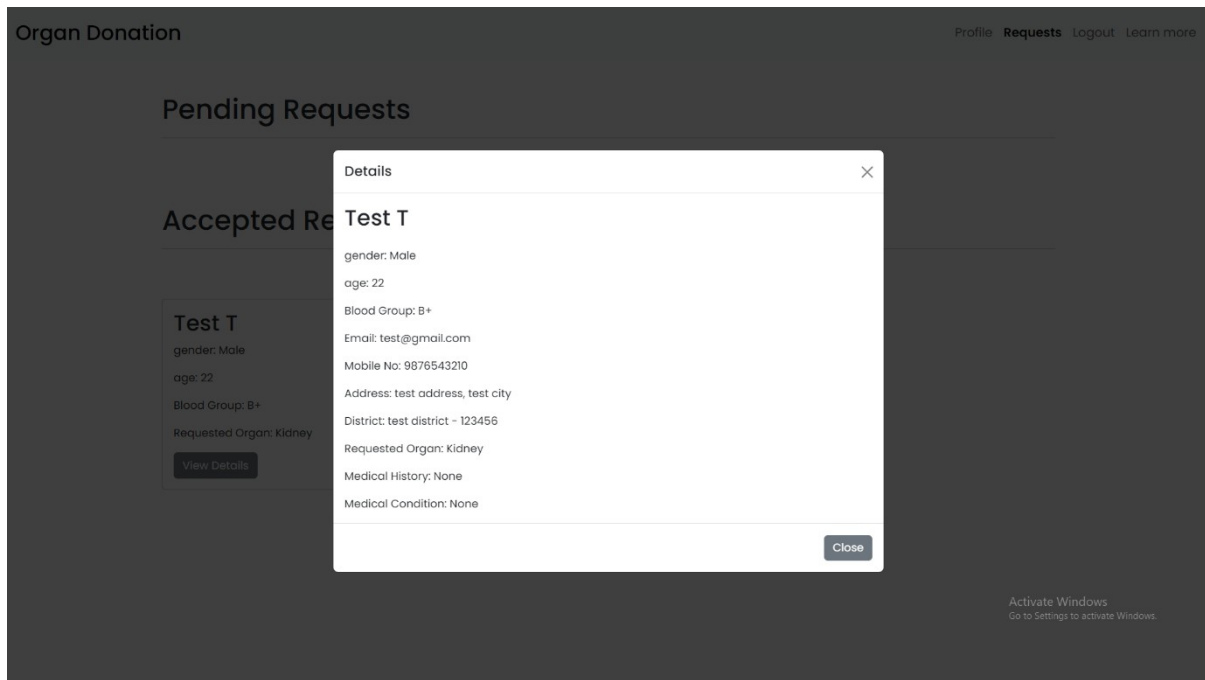


Fig 5.16 Contact Sharing

Fig 5.16 shows the shared contact information once the request is accepted by the donor.

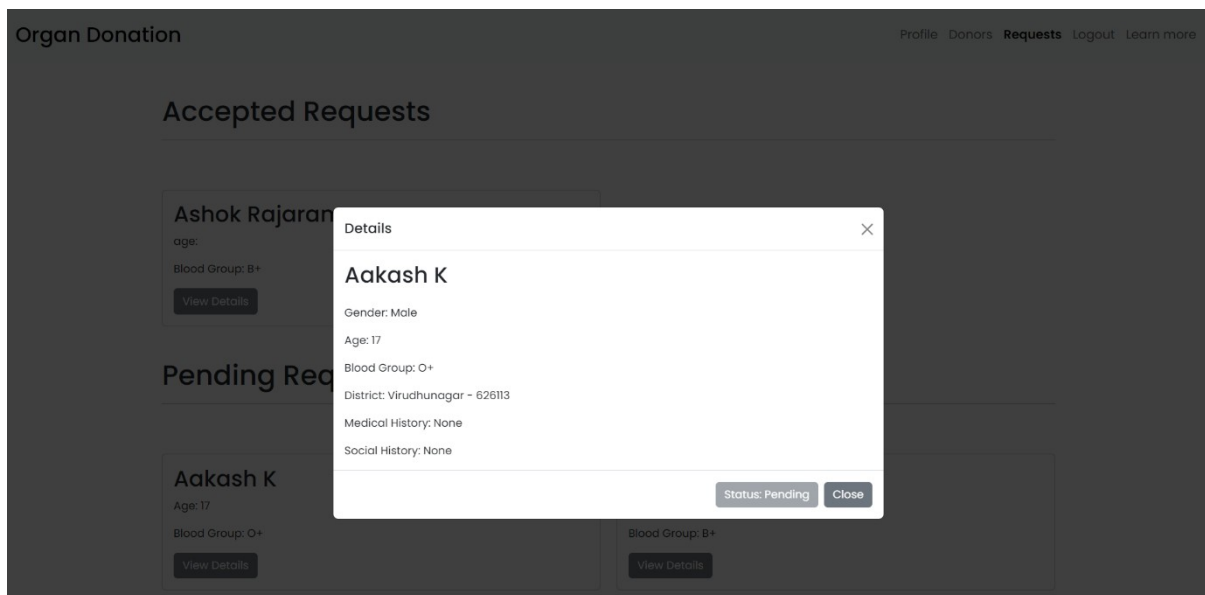


Fig 5.17 Pending request details

Fig 5.17 displays the pending request details that the recipient has sent to the donor.

5.1.5 Notification Module

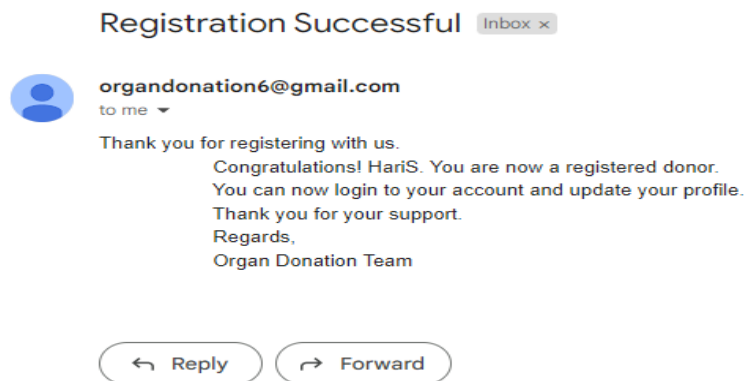


Fig 5.18 Donor registration notification

Fig 5.18 is the mail sent to the donor after registration.

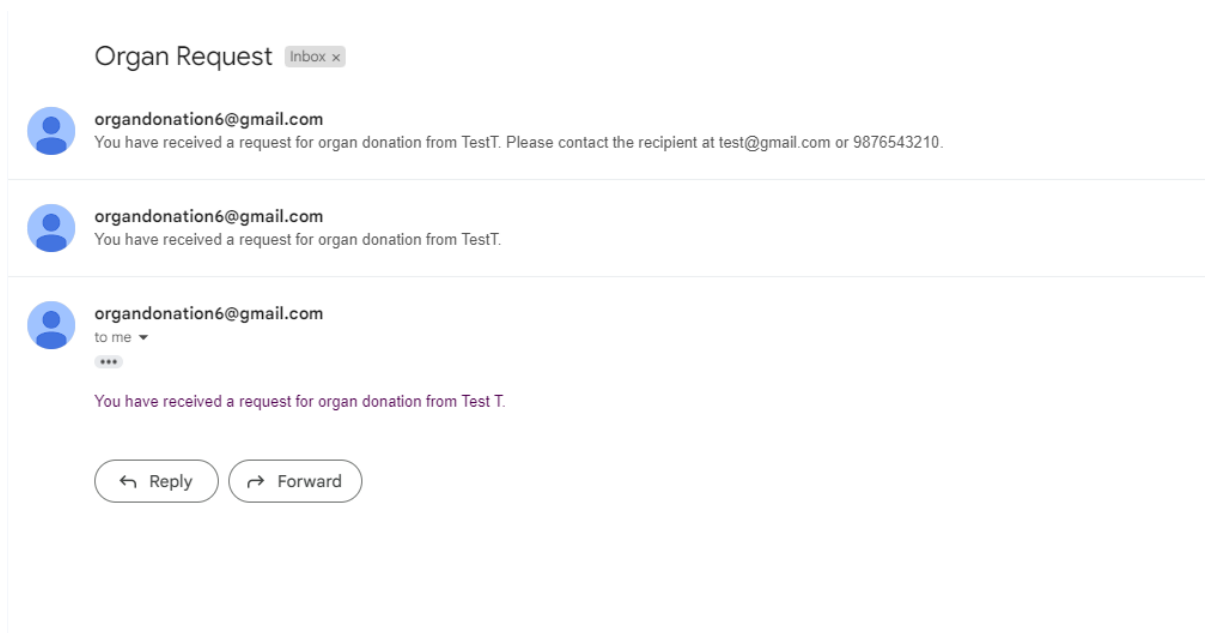


Fig 5.19 Donor request notification

Fig 5.19 is the mail sent to the donor when a recipient sends a request.

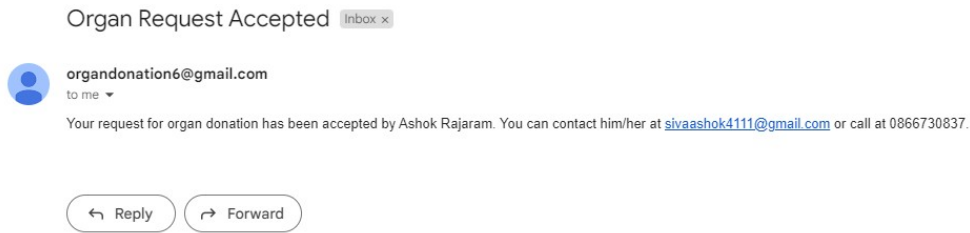


Fig 5.20 Recipient conformation notification

Fig 5.20 is the mail sent to the recipient after donor accepts the request.

5.2 TESTING

Testing is the important aspect of the Software Development Life Cycle. This is the stage where we confirm whether the project has archived the ultimate goal or not testing encompasses a set of activities that were conducted systematically. This begins at the module level and works towards the integration of the entire system. The purpose of testing the system is to identify various bugs and to fix them. It provides a way to check the functionality. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner.

TEST CASE

A test case is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

5.2.1 USER MODULE

S. No	Input Details	Expected Output	Obtained Output	Test case result
1	Donor Registration	Collect Donor information and store it in database	The data is collected and stored in the database	pass
2	Recipient Registration	Collect Recipient information and store it in database	The data is collected and stored in the database	pass
3	Donor login	Login with the registered credentials and display the profile page	Logged in and the profile page is displayed	pass
4	Donor request page	Requests from the recipients has to be displayed here	The requests from recipients are displayed here	pass
5	Recipient login	Login with the registered credentials and display the profile page	Logged in and the profile page is displayed	pass
6	Recipient request page	The requests sent by the recipient has to be shown here	The requests sent by the recipient are shown	pass

Table 5.1

5.2.2 DONOR FILTRATION

S. No	Input Details	Expected Output	Obtained Output	Test case result
1	Search for donors	Display all available donors	All donors are displayed	pass
2	Filter donors using location, blood group and organ	Display the filtered donor list	The Donor list is filtered and displayed	pass
3	View details of donors and send request	The details of donors has to be viewed and request can be sent	Donor details can be viewed and request can also be sent	pass

Table 5.2

5.2.3 REQUEST AND NOTIFICATION

S. No	Input Details	Expected Output	Obtained Output	Test case result
1	Registration successful notification email for both donor and recipient	A registration successful notification email is sent after registration	A registration successful notification email is received after registration	pass
2	Request sent to donor after sending request	A request is received in the donor requests page	A request is received in the donor requests page	pass
3	Notification email sent to donor when request is received	When a recipient send request to donor, the donor has to be notified by a notification email	When a recipient send request to donor, the donor is notified by a notification email	pass
4	Request accepted by the donor	Donor has to be able to accept the request	Donor can accept the request	pass
5	Recipient receive	After donor	After donor	pass

	notification email for donor approving his request	accepts the request, the recipient has to be notified by a notification email	accepts the request, the recipient is notified by a notification email	
6	Recipient can view accepted requests along with contact information	After a request is accepted by the donor, the recipient has to be able to view donor contact information	After a request is accepted by the donor, the recipient can view donor contact information	pass
7	Donor can view the contact information of recipient after approval of request	Donor has to be able to view contact information of recipient after the request is accepted	Donor can view contact information of recipient after the request is accepted	pass

Table 5.3

CHAPTER 6

CONCLUSION

The computerization of "ORGAN DONATION MANAGEMENT SYSTEM" has been proposed to satisfy the needs of the user. The entire system is menu driven and highly interactive. The system can process any volume of data and it is developed in such a way that if any modifications and enhancements are needed in future it can be done at easy without disturbing the proper working of the system.

The system also produces the maximum flexibility to the user. All the validity is taken care of by the system during data entry. The user of the software is extremely satisfied with the system.

Our system can be future improved by including concept of e-mail, where in the hospitals, donors and recipients can be acknowledged with e-mail regarding the various changes, updates and requirements. The usage of our system requires basic computer skills. Organ scarcity reveals mainly due to the lack of proper knowledge about organ donation; not many illustrate come forward to donate organ. In the future, option can be provided such that illustrate people can use it.

APPENDIX

Sample code

models.py

```
from django.contrib.auth.models import AbstractUser

from django.db import models


class User(AbstractUser):

    email = models.EmailField(unique=True)

    USERNAME_FIELD = 'email'

    REQUIRED_FIELDS = []

    is_donor = models.BooleanField("Is Donor", default=False)

    is_recipient = models.BooleanField("Is Recipient", default=False)


class Organ(models.Model):

    name = models.CharField(max_length=20)

    def __str__(self):

        return self.name


class Donor(models.Model):

    user = models.OneToOneField(User, on_delete=models.CASCADE,
primary_key=True)

    email = models.EmailField(max_length=150, null=True)
```

```

first_name = models.CharField(max_length=60)

last_name = models.CharField(max_length=60)

dob = models.DateField(null=True)

age = models.CharField(max_length=2 ,null=True)

gender = models.CharField(max_length=10)

mobile = models.CharField(max_length=10)

blood_group = models.CharField(max_length=3)

aadhaar = models.CharField(max_length=16)

addr_line = models.CharField(max_length=100)

city = models.CharField(max_length=60)

district = models.CharField(null=True, max_length=60)

state = models.CharField(max_length=60)

pincode = models.CharField(max_length=6)

organs = models.ManyToManyField(Organ)

medical_history = models.TextField(max_length=200)

social_history = models.TextField(max_length=200)

def __str__(self):

    return (f'{self.first_name} {self.last_name}')

class Recipient(models.Model):

    email = models.EmailField(max_length=150, null=True)

```



```
user = models.OneToOneField(User, on_delete=models.CASCADE,
primary_key=True)

first_name = models.CharField(max_length=60)

last_name = models.CharField(max_length=60)

dob = models.DateField(null=True)

age = models.IntegerField(null=True)

gender = models.CharField(max_length=10)

mobile = models.CharField(max_length=10)

blood_group = models.CharField(max_length=3)

aadhaar = models.CharField(max_length=16)

addr_line = models.CharField(max_length=100)

city = models.CharField(max_length=60)

district = models.CharField(null=True ,max_length=60)

state = models.CharField(max_length=60)

pincode = models.CharField(max_length=6)

organ_needed = models.CharField(max_length=20)

medical_history = models.TextField(max_length=200)

medical_condition = models.TextField(max_length=200)

def __str__(self):

    return (f'{self.first_name} {self.last_name}')
```

```

class Request(models.Model):

    recipient = models.ForeignKey(Recipient, on_delete=models.CASCADE,
related_name='recipient')

    donor = models.ForeignKey(Donor, on_delete=models.CASCADE,
related_name='donor')

    is_accepted = models.BooleanField(default=False)

    is_rejected = models.BooleanField(default=False)

    def __str__(self):

        return (f'{self.recipient.first_name} {self.recipient.last_name} requested
{self.donor.first_name} {self.donor.last_name}')

```

views.py

```

from django.shortcuts import render, redirect

from django.contrib.auth import authenticate, login, logout

from django.contrib import messages

from django.contrib.auth.models import User

from django.core.mail import send_mail

from .models import User, Recipient, Donor, Organ, Request

from django.contrib.auth.decorators import login_required


# Home page view

def index(request):

    user = request.user

```

```

if user.is_authenticated and user.is_donor:

    return redirect('donorhome')

elif user.is_authenticated and user.is_recipient:

    return redirect('recipientprofile')

donor_count = Donor.objects.all().count()

recipient_count = Recipient.objects.all().count()

context = {

    'donor_count': donor_count,

    'recipient_count': recipient_count,

}

return render(request, 'organdonation/index.html', context)

```

#Donor Registration View

```

def dregister(request):

    if request.method == 'POST':

        first_name = request.POST['first_name']

        last_name = request.POST['last_name']

        dob = request.POST['dob']

        age = request.POST['age']

        gender = request.POST['gender']

        blood_group = request.POST['blood_group']

        mobile_num = request.POST['mobile_num']

```

```

aadhaar = request.POST['aadhaar']

email = request.POST['email']

password1 = request.POST['password1']
password2 = request.POST['password2']

addr_line = request.POST['addr_line']

city = request.POST['city']

district = request.POST['district']

state = request.POST['state']

pincode = request.POST['pincode']

social_history = request.POST['social_history']

medical_history = request.POST['medical_history']

form_organ = request.POST.getlist('organ')

def_organ = ['kidney', 'liver', 'lungs', 'pancreas', 'small_intestine', 'blood']

if password1 == password2:

    if User.objects.filter(email=email).exists():

        messages.success(request, "This email is already registered with an
account")

        return redirect('dregister')

    else:

        is_donor = True

```

```
user = User.objects.create(username = first_name.lower()  
[0:4]+aadhaar[0:4],email = email, first_name=first_name,  
last_name=last_name, is_donor=is_donor)  
  
user.set_password(password1)  
  
user.save()  
  
donor = Donor.objects.create(user=user)  
  
donor.first_name = first_name  
  
donor.last_name = last_name  
  
donor.email = email  
  
donor.dob = dob  
  
donor.age = age  
  
donor.gender = gender  
  
donor.blood_group = blood_group  
  
donor.mobile = mobile_num  
  
donor.aadhaar = aadhaar  
  
donor.addr_line = addr_line  
  
donor.city = city  
  
donor.district = district  
  
donor.state = state  
  
donor.pincode = pincode  
  
donor.social_history = social_history  
  
donor.medical_history = medical_history
```

```

donor.save()

for organ in form_organ:

    if organ in def_organ:

        organ = Organ.objects.get(name = organ)

        donor.organ.add(organ)

send_mail(

    'Registration Successful',

    """Thank you for registering with us.

    Congratulations! """ + donor.first_name + " " + donor.last_name + ".

    You are now a registered donor.

    You can now login to your account and update your profile.

    Thank you for your support.

    Regards,

    Organ Donation Team""",

    'organdonation6@gmail.com',

    [donor.email],

    fail_silently=False,

)

messages.success(request, "Your account has been created
successfully.")

return redirect('login')

```

```
    else:

        messages.success(request, "Password and Confirm Password doesn't
match.")

        return redirect('dregister')

    else:

        return render(request, 'organdonation/dregister.html')
```

#Recipient Registration View

```
def rregister(request):

    if request.method == 'POST':

        first_name = request.POST['first_name']

        last_name = request.POST['last_name']

        dob = request.POST['dob']

        age = request.POST['age']

        gender = request.POST['gender']

        blood_group = request.POST['blood_group']

        mobile_num = request.POST['mobile_num']

        aadhaar = request.POST['aadhaar']

        email = request.POST['email']

        password1 = request.POST['password1']

        password2 = request.POST['password2']

        addr_line = request.POST['addr_line']
```

```

city = request.POST['city']

district = request.POST['district']

state = request.POST['state']

pincode = request.POST['pincode']

organ_needed = request.POST['organ']

medical_condition = request.POST['medical_condition']

medical_history = request.POST['medical_history']

if password1 == password2:

    if User.objects.filter(email=email).exists():

        messages.success(request, "This email is already registered with an
account")

        return redirect('rregister')

    else:

        is_recipient = True

        user = User.objects.create(username = first_name.lower()
[0:4]+aadhaar[0:4],email = email, first_name=first_name,
last_name=last_name, is_recipient=is_recipient)

        user.set_password(password1)

        user.save()

        recipient = Recipient.objects.create(user=user)

        recipient.first_name = first_name

        recipient.last_name = last_name

```



```
recipient.email = email

recipient.dob = dob

recipient.age = age

recipient.gender = gender

recipient.blood_group = blood_group

recipient.mobile = mobile_num

recipient.aadhaar = aadhaar

recipient.addr_line = addr_line

recipient.city = city

recipient.district = district

recipient.state = state

recipient.pincode = pincode

recipient.organ_needed = organ_needed

recipient.medical_condition = medical_condition

recipient.medical_history = medical_history

recipient.save()

send_mail(

    'Registration Successful',

    '"Thank you for registering with us, "' + recipient.first_name +
recipient.last_name + '". You are now a registered recipient.

    You can now login to your account and search for the available
donors.
```

Thank you for your support.

Regards,

Organ Donation Team",

'organdonation6@gmail.com',

[recipient.email],

fail_silently=False,)

messages.success(request, "Your account has been created
successfully.")

return redirect('login')

pass

else:

messages.success(request, "Password and Confirm Password doesn't
match.")

return redirect('rregister')

else:

return render(request, 'organdonation/rregister.html')

#User Login View

def userlogin(request):

if request.method == 'POST':

email = request.POST['email']

password = request.POST['password']

```
user = authenticate(request, email=email, password = password)
```

```
if user is not None:
```

```
    login(request, user)
```

```
    if user.is_donor:
```

```
        return redirect('donorhome')
```

```
    elif user.is_recipient:
```

```
        return redirect('recipientprofile')
```

```
    else:
```

```
        messages.success(request, 'Error logging in.')
```

```
        logout(request)
```

```
        return redirect('login')
```

```
else:
```

```
    messages.success(request, 'Not valid Credentials!')
```

```
    return redirect('login')
```

```
else:
```

```
    return render(request, 'organdonation/login.html', {})
```

```
#User Logout View
```

```
def userlogout(request):
```

```
    logout(request)
```

```
    messages.success(request, 'You have been logged out Successfully')
```

```
    return redirect('index')
```

#Donor Profile View

@login_required(login_url='login')

def donorhome(request):

 current_user = request.user

 if current_user.is_authenticated and current_user.is_donor:

 organlist = (current_user.donor.organs.all())

 return render(request, 'organdonation/donorhome.html', {'user':
current_user, 'organlist': organlist})

#Recipient Profile View

@login_required(login_url='login')

def recipientprofileview(request):

 current_user = request.user

 if current_user.is_authenticated and current_user.is_recipient:

 return render(request, 'organdonation/recipientprofile.html', {'user':
current_user})

#Recipient Donor Filtraion View

@login_required(login_url='login')

def recipientdonorsearch(request):

 current_user = request.user

 donorlist = (Donor.objects.all())

```

if current_user.is_authenticated and current_user.is_recipient:

    if request.method == 'GET':

        bg = request.GET.get('blood_group')

        organ = request.GET.get('organ')

        location = request.GET.get('location')

        if bg != "" and organ != "" and location != "":

            donorlist = donorlist.filter(blood_group = bg, organs__name = organ,
district = location)

        elif bg != "" and organ != "":

            donorlist = donorlist.filter(blood_group = bg, organs__name = organ)

        elif bg != "" and location != "":

            donorlist = donorlist.filter(blood_group = bg, district = location)

        elif organ != "" and location != "":

            donorlist = donorlist.filter(organs__name = organ, district = location)

        elif bg != "":

            donorlist = donorlist.filter(blood_group = bg)

        elif organ != "":

            donorlist = donorlist.filter(organs__name = organ)

        elif location != "":

            donorlist = donorlist.filter(district = location)

    return render(request, 'organdonation/recipientdonorsearch.html', {'user':
current_user, 'donorslist': donorlist})

```

```

#Recipient send request to donor

@login_required(login_url='login')

def recipientsendrequest(request, donor_id):

    current_user = request.user

    if current_user.is_authenticated and current_user.is_recipient:

        donor = Donor.objects.get(user_id = donor_id)

        recipient = Recipient.objects.get(user_id = current_user.id)

        recipient_request, created = Request.objects.get_or_create(donor = donor,
recipient = recipient)

        if created:

            messages.success(request,'Request sent successfully.')

            send_mail( 'Organ Request',

                ""You have received a request for organ donation from "" +
current_user.recipient.first_name + ' ' + current_user.recipient.last_name + "".""",

                'organdonation6@gmail.com',

                [donor.email],

                fail_silently=False,)

            return redirect('recipientdonorsearch')

        else:

            messages.success(request,'Request already sent.')

            return redirect('recipientdonorsearch')

    return render(request, 'organdonation/recipientdonorsearch.html')

```

```

#Show requests to donor

@login_required(login_url='login')

def donorrequest(request):

    current_user = request.user

    if current_user.is_authenticated and current_user.is_donor:

        recipient_request = Request.objects.filter(donor_id =
current_user.donor.user_id, is_accepted = False)

        accepted_request = Request.objects.filter(donor_id =
current_user.donor.user_id, is_accepted = True)

        return render(request, 'organdonation/donorrequest.html', {'requested':
recipient_request, 'accepted' : accepted_request})


#Show requests to recipient

@login_required(login_url='login')

def recipientrequest(request):

    current_user = request.user

    if current_user.is_authenticated and current_user.is_recipient:

        requested_donors = Request.objects.filter(recipient_id =
current_user.recipient.user_id, is_accepted = False)

        accepted_donors = Request.objects.filter(recipient_id =
current_user.recipient.user_id, is_accepted = True)

        return render(request, 'organdonation/recipientrequest.html', {'requested':
requested_donors, 'accepted' : accepted_donors})

```

```

#Donor accept request

@login_required(login_url='login')

def donoracceptrequest(request, recipient_id):

    current_user = request.user

    if current_user.is_authenticated and current_user.is_donor:

        recipient_request = Request.objects.get(donor_id =
current_user.donor.user_id, recipient_id = recipient_id)

        print(recipient_request)

        recipient_request.is_accepted = True

        recipient_request.save()

        send_mail(

            'Organ Request Accepted',

            ""Your request for organ donation has been accepted by "" +
current_user.donor.first_name + ' ' + current_user.donor.last_name + "". You can
contact him/her at "" + current_user.donor.email + "" or call at "" +
current_user.donor.mobile + "". ",

            'organdonation6@gmail.com',

            [recipient_request.recipient.email],

            fail_silently=False,

        )

        return redirect('donorrequest')

    return render(request, 'organdonation/donorrequest.html')

```


urls.py

```
from django.urls import path
```

```
from . import views
```

```
urlpatterns = [
```

```
    path("", views.index, name='index'),
```

```
    path('dregister/', views.dregister, name='dregister'),
```

```
    path('rregister/', views.rregister, name='rregister'),
```

```
    path('login/', views.userlogin, name='login'),
```

```
    path('userlogout/', views.userlogout, name='userlogout'),
```

```
    path('donor/home/', views.donorhome, name='donorhome'),
```

```
    path('donor/requests/', views.donorrequest, name='donorrequest'),
```

```
    path('donor/request/<int:recipient_id>/', views.donoracceptrequest,  
name='donoracceptrequest'),
```

```
    path('recipient/profile/', views.recipientprofileview, name='recipientprofile'),
```

```
    path('recipient/donors/', views.recipientdonorsearch,  
name='recipientdonorsearch'),
```

```
    path('recipient/send_request<int:donor_id>/', views.recipientsendrequest,  
name='recipientsendrequest'),
```

```
    path('recipient/requests/', views.recipientrequest, name='recipientrequest'),
```

```
]
```

REFERENCES

- **HTML** : <https://www.w3schools.com/html/>
- **CSS** : <https://www.w3schools.com/css/>
- **Bootstrap** : <https://getbootstrap.com/docs/5.3/>
- **Javascript** : <https://www.w3schools.com/js/>
- **Django** : <https://www.djangoproject.com/start/>
- **SQLite** : <https://www.sqlite.org/docs.html>