1.INTRODUCTION

ABOUT THE PROJECT

Project entitled "**U-Donate**" is to automate the complete operations of the blood donation Centre. The need for organs for donation is far greater than blood availability. In the last decade this has led to restructuring and investment in the blood donation programmer with political and public support. This software helps to register all the donors and receiver details, reports etc. The "blood donation Management System" helps to ensure the day to day transactions smooth & safe. Our project is developed using PYTHON with MySQL.

2.SYSTEM STUDY AND ANALYSIS

2.1 EXISTING SYSTEM

The existing system of handling the day to day transactions of a blooddonation Centre is a tedious task and complex. The donor as well as the receiver communicates with the Centre with the help of application forms. They need to maintain hundreds of thousands of records. This makes the existing system very difficult.

2.2.LIMITATION OF EXISTING SYSTEM

The donor and receiver have to submit their applications directly or through somebody else. In some urgent situations this is not possible like emergency cases. Here comes the need for the computerized system for fast safe and low risk.

In other way the organization has to check the different categories of application one by one. This requires more time and human effort. In order to tackle all these situations the computerization of the existing system is necessary.

2.3.PROPOSED SYSTEM

The proposed computerization is developed using MYSQL Server as back end and the most powerful PYTHON as front end. The main operations software are Approve donor, View donation details, View and publish reports, Add staff, Donor registration, Add donor health information, Add receiver health information, find matching donor ,Receiver registration, Donation details

Advantages of proposed system

In brief the proposed system will helps to mange to keep the all the details in a neat format and also searching should be very faster so they can find required details instantly.

FEASIBLITY STUDY

Feasibility study is done in my software development as a part of preliminary investigation. Specific method used by the analysis for collecting data about requirements are fact finding techniques. These include record review, observations, interview and questionnaires. When the request is made, the first system activity, the preliminary investigation begins. This activity has 3 steps among them feasibility

study is important. Feasibility study is an important outcome of the preliminary investigation and is the determination that the system request is feasible. And my software **Blood Donation** satisfies the different types of the feasibility studies. They are listed below:

The study done in three phases

- Operational feasibility
- Technical feasibility
- Economical feasibility

Operational Feasibility

Proposed systems are beneficial only if they can be turned into information systems. That is it will meet the organizations operating requirements and also checks that whether the system will work when it is developed and installed. The software **Blood Donation** software supports the operational feasibility to a great extends. The performance of this software is more accurate, more user friendly, effective, error free.

A feasibility study is not warranted systems, in which economic justification is obvious, technical risk is low, few legal problems are expected and no reasonable alternative exists. However, if any of the preceding conditions fail, a study of that area should be conducted. Economic justification include a broad range of concerns that include cost-benefit analysis, long-term corporate income strategies, impact on other profit concerns or products, cost of resources needed for development and potential market growth.

Technical Feasibility

This is related to the technicality of the project. This evaluation determines whether the technology needed for the proposed system is available or not. It deals with hardware as well as software requirements. That is, type of hardware, software and the methods required for running the systems are analyzed. This involves financial consideration to accommodate technical enhancement. If the budget is a serious constraint, then the project is judged not feasible.

The software **Blood Donation** software supports the technical feasibility to a great extends. That is, this software can be operated with the minimum technical support. It uses PYTHON as front end, MYSQL as database at windows platform and Mozilla Firefox and Google Chrome as browser. And also it provides accuracy, reliability, ease of access and data security.

Economical Feasibility

Economical analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. More commonly known as cost/benefit analysis: the procedure is to determine the benefits and saving that are expected from a proposed system and compare them with cost. If benefits outweigh cost, a decision is taken to design and implement the system. Otherwise, further justification or

alternative in the proposed system will have to be made if it is to have a chance of being approved. This is an ongoing effort that improves in accuracy at each phase of the system life cycle.

Hence the engineer will not find any difficulty at the installation time and after installation user also newer find difficulty i.e. hang, slow speed or slow response time. One project is compulsory for each student this project is either dummy or lives. If I am developing a live project then it gives a lot of confidence. It is better for me and for company because, I am developing a system without any money. So everything is in favour now, I can say the cost of this software is I think negligible. Hence the economical feasibility is very good

3.REQUIREMENT ANALYSIS AND SPECIFICATION

HARDWARE REQUIREMENTS

Hardware Specification

The selection of hardware configuration is a very important task related to the software development. Random access memory may affect adversely on the speed and correspondingly on the efficiency of the entire system. The processor should be powerful to handle all the operations. The hard disk should have sufficient capacity to store the database and the application. The network should be efficient to handle the communication fast.

Processor : Intel i3

RAM : 4 GB

Hard Disk : 1 TB

Keyboard : Standard 101/102 key

Mouse : Optical mouse

Monitor : LED monitor

Printer : Laser

SOFTWARE SPECIFICATION

Operating System : Windows 64 Bit

Language : Python 3.7.8 or Above

Front End : HTML

Back End : MY SQL

IDE : Pycharm

OPERATING SYSTEM- WINDOWS 7

Windows 7 is the Microsoft Windows operating system released commercially in October 2009. In development, Windows 7 was known by the code names "Blackcomb" and "Vienna. Windows 7 is built on the Vista kernel. Too many end users the biggest changes between Vista and Windows 7 are faster boot times, new user interfaces and the addition of Internet Explorer 8. The OS is widely available in three

retail editions: Windows 7 Home Premium, Professional and Ultimate. Starter, OEM and Enterprise editions are available in some markets.

The WFU standard operating system for the ThinkPad is the Microsoft 32 bit Windows 7 Enterprise edition. The primary features of Windows 7 are:

Starts Menu – The Start Menu provides the primary access point for programs and applications on your ThinkPad.

Taskbar and Notification Area – The Taskbar contains 3 main components, the Start button, the Task/Quick launch bar and the System Notification Area.

Windows Snipping Tool– Windows 7 includes an application to capture, save and share "snipped" images from your desktop.

Displaying to a External Monitor or Projector – The ThinkPad is equipped with one or more external display connectors so that you may connect to an external monitor or the multimedia projector.

Using Local Area Networks (LAN/Wi-Fi) – The ThinkPad is configured to access both wired and wireless network resources.

Windows Explorer Favourites – File system favourites are quick links to specific directories (folders) located on your hard drive.

Windows Explorer Libraries – Libraries are an easy way to collect and track documents on your computer that are related to one another but are not necessarily located in the same directories (folders).

FRONT END -PYTHON

PYTHON is a programming language devised by Rasmus Lerdorf in 1994 for building dynamic, interactive web sites. It is maturing as a technology is evidenced by its totally revamped and upgraded support for object-oriented programming (OOP) principles and improved support for XML.PYTHON's main use is a cross platform, HTML-embedded, server-side web scripting language.

- **Cross-Platform** : Most PYTHON code can be processed without alteration on computers running many different operating system.
- **HTML-embedded** : PYTHON code can be written in files containing a mixture of PYTHON instructions and HTML code.
- **Server-Side** : The PYTHON programs are run on server specifically a web server.
- **Web-scripting language** : PYTHON programs run via web browser.

This means you'll write programs that mix PYTHON code and HTML, run then on web server, and access them from a web browser that displays the result of your PYTHON processing by showing you the HTML returned by the web server. In other words, you can make your programs available for other people to access across the web, simply by placing them on a public web server.

HTML is the main language used to create web pages, combining

plain text with special tags that tell browsers how to treat text.HTML is used to describe how different elements in a web page should be displayed, how pages should be linked, where to put images, and so on.

We can't create a web site like that using raw HTML, and that's where PYTHON comes in what sort of things can be with it? Well you can program sites that

- Present data from a wide variety of source, such as database, files or even other web pages.
- Incorporate interactive elements, such as search facilities, message boards, and straw polls.
- Enable the user to perform actions, such as sending e-mail or buying something.

We'll use PYTHON to build

- A simple, online text editor.
- A web bases e-mail application.
- Object-oriented contact manager applications.
- An object-oriented logging agent.

One of the best things about PYTHON is the large number of internet service providers (ISP) and web hosting companies that support it.

PYTHON is a cross platform technology. It's easy to get it up and running over web server. Comparing PYTHON with ASP is a more balanced comparison ASP does not well on a variety of platforms.

DATABASE- MY SQL

MySQL is the world most popular open source database. Open source means that the source code, the programming code that makes up MySQL, is freely available to anyone. MySQL is a relational database management system (RDBMS). It is a program capable of storing an enormous amount and a wide variety of data and serving it up to meet the needs of any type of organization, from mum and pop retail establishments to the largest commercial enterprises and governmental bodies. MySQL competes with well-known proprietary RDBMS,such as Oracle,SQL,and DB2.

MySQL comes with everything needed to install the program, set up differing levels of user access, administer the system, and secure and back up the data. We can develop database applications in most programming languages used today and run them on most operating systems, including some of which you have probably never heard. MySQL utilizes structured query language(SQL),the language used by all relational databases .Sql enables us to create databases as well as add, manipulate, and retrieve data according to specific criteria.

The easiest way to understand a database is as collection of related files. A database and the software that controls the database, called a database management system (DBMS). The most databases today are relational databse,named such because they deal with tables of data related by a common field.

Database Terminology

The database contains tables, each table contains many rows and columns. Each row contains data about one single entity. This is called a record. Each column contains one piece of data that relates to the record, called an attribute. An attribute, when referring to a database, is called field.

The kind of structure, the database gives as a way to manipulate this data. SQL is a powerful way to search for records or make changes. Almost all DBMS use SQL.

4.SYSTEM DESIGN

INTRODUCTION

System design involves translating information requirements and conceptual design into technical specification and general flow of processing.

The system's design converts the theoretical solution introduced by the feasibility study into a logical reality.

Design is the second phase in the system development life cycle. Software design is the first of the three technical activities in the software development process such as design, coding and testing.

During this phase, the analyst schedules design activities, works with the user to determine the various data inputs to the system, plans how data will flow through the system, designs required outputs and writes program specifications. Again the analyst's activities focus on solving a user's problem in logical terms.

During this second step, analysts employ a variety of tools such as data flow diagrams, entity-relationship diagrams, data dictionaries and Gantt chart.

- Draws a model of the new system, using data flow and entity-relationship diagrams.
- Develop methods for collecting and inputting data.
- Defines the detailed data requirements with a data dictionary.
- Writes program specifications.
- Specifies control techniques for the system's outputs, databases and inputs.
- Identifies and orders any hardware or software that the system will need.

In the physical design phase, necessary software is developed to accept input from the user, to perform necessary calculations through the manipulation of data stored in the databases to produce the appropriate result.

1. INPUT DESIGN

Input design is the process of converting user oriented input into a computer based format. The data flow diagram indicates logical data flows, data stores source and destinations. Input data are collected and organized into a group of numbered data. Once identified appropriate input data media are selected for processing.

The goal of input design is to make data entry as easy and free from errors as much as possible. Proper data validation checking is exercised to correct the mistakes made during data entry.

The options for input data entry are as follows.

SUBMIT

The submit option in each of the forms helps inputing data to the system. Web based features like emailing of the inputing data to the user etc. are included in this. For commercial and security reasons some of the adding options are restricted to the administrator of the site.

According to our software we can submit the student data or the candidate data and always submit the company details.Through the successfull submission of data the candidate and the company can register in the field and through this they can successfully do their work also.

EDIT

The editing option helps in updating the user data. For commercial and security reasons some of the editing options are restricted to the administrators of the site.

Based on our software we can efficiently edit our details. In the case of a candidate they can edit their profile after submission. They can change their personal, educational as well as experience details .According to a company they can also change their details by the edit button. They can add more details and can edit their profile by this button.

DELETE

The delete option helps in removal of data from the system. For commercial and security reasons some of the deletion options are restricted to the administrators of the site.

The delete option helps to delete unwanted details about a candidate or a Company. In the case of a company we can reject the company by this delete option.

B. OUTPUT DESIGN

Output design is an ongoing activity almost from the beginning of the project. Computer output is the most important and direct source of information to the user. Output design is a process that involves designing necessary outputs that have to be given to various users according to their requirements. Efficient output design should improve the system's relationship with the user and help in decision making .The system has got the capability to display standard screen layouts. These layouts should be designed around the output requirements and they must be designed with utmost care and details in the screen layouts must be simple, descriptive and clear to the user.

While designing a system's output, the analyst must make several independent decisions. Every system produces some kind of reports. No matter what is the content of the report. The following guidelines can be applied to any report.

- The information should be clear, accurate, concise and restricted to relevant data.
- Reports should have titles, date and descriptive headings for columns of data, numbered pages and so on. If printed, they should also appear on standard size of paper.

- The report's contents should be in a logical arrangement so that user can easily locate what they need.
- The report should come on an output medium that best suits the user's needs.

C. DATABASE DESIGN

Database is a collection of related tables. We use this to produce information to the committees. So database design should be done in a way we can store all the needed information correctly and clearly. Redundancy and loss of flexibility must be avoided. In the computerized inventory control system we design a database, which successfully avoid these problems.

The most important aspect of building software systems is database design. The highest level in the hierarchy is the database. It is a set of inter-related files for real time processing. It contains the necessary data for problem solving and can be used by several users accessing data concurrently. The general objective of database design is to make the data access easy, inexpensive and flexible to the user. Database design is used to define and then specify the structure of business used in the client/server system. A business object is nothing but information that is visible to the user of the system. The database must be a normalized one.

Database Management system (DBMS) allows the data to be protected and organized separately from other resources like hardware, software and programs. DBMS is a software package, which contains components that are not found in other data management packages. The significance of DBMS is the separation of data as seen by the program and data as stored on the direct access storage devices, i.e. the difference between logical and physical data.

In our project, we have used MySQL as the database to implement the data stored part. The most important part in the database design is the identification of table to be used. The database design in our project is named "DON" which is used to store data into the corresponding tables.

TABLE

Table Name-doreg

Primary key-DonorCode

Field	Type	Constraints	Description
DonorCode	varchar(20)	Primary key	Donor code
Nationality	varchar(20)	Not null	Nationality
First name	varchar(20)	Not null	First name
Last name	varchar(20)	Not null	Last name
Age	varchar(20)	Not null	Age
Sex	varchar(20)	Not null	Sex
Blood group	varchar(20)	Not null	Blood group
Organs	varchar(20)	Not null	Organs details
I.D card No	varchar(20)	Not null	Id card number
Mobile	int(20)	Not null	Mobile number
Person name	varchar(20)	Not null	Person name
Address	varchar(20)	Not null	Address
status	int(11)	Not null	status

Table Name-usercreate

Primary key-Username

Field	Type	Constraints	Description
Username	varchar(30)	Primary key	username
First name	varchar(30)	Not null	First name
Last name	varchar(30)	Not null	Last name
Designation	varchar(30)	Not null	Designation
Contactno	int(30)	Not null	Contact number
Password	varchar(30)	Not null	password

Table Name-login

Primary key-Username

Field	Type	Constraints	Description
username	varchar(20)	Primary key	Username
password	varchar(20)	Not null	password

Table Name-accounthead

Primary key-hcode

Field	Type	Constraints	Description
hcode	int(11)	Primary key	Head code
hname	varchar(50)	Not null	Head name

Table Name-bill

Primary key-billno

Field	Type	Constraints	Description
billno	int(11)	Primary key	Bill number
billdate	varchar(10)	Not null	Bill date
hcode	int(11)	Foreign key	Head code
amount	decimal(20,0)	Not null	Bill amount
toperson	varchar(50)	Not null	Person name

Table Name-dodetails

Primary key-Donationid

Field	Type	Constraints	Description
Donationid	int(11)	Primary key	Donation id
Donor code	varchar(20)	Not null	Donor code
Patient code	varchar(20)	Not null	Patient code
Donatiion date	varchar(20)	Not null	Donation date
Admitted Hospital	varchar(20)	Not null	Admitted hospital name
Name of doctor	varchar(20)	Not null	Name of doctor
Date of admission	varchar(20)	Not null	Date of admission
Other Details	varchar(20)	Not null	Other details

Table Name-healthrecord

Primary key-helthid

Field	Type	Constraints	Description
helthid	int(11)	Primary key	Health id
Family physician	varchar(40)	Not null	Family physician
Address	varchar(40)	Not null	Address
Medical Insurance Company	varchar(40)	Not null	Medical insurance company detail
Policy no	int(40)	Not null	Policy number
Date of Entrance to University	varchar(40)	Not null	Date of Entrance to University
Name any chronic illness	varchar(40)	Not null	Name any chronic illness
List medications you are currently taking	varchar(40)	Not null	List medications you are currently taking
List any medicine,food etc	varchar(40)	Not null	List any medicine,food etc
Patientcode	varchar(50)	Not null	Patientcode

Table Name-payment

Primary key-pid

Field	Type	Constraints	Description
pid	int(11)	Primary key	Payment id
pdate	varchar(50)	Not null	Payment date
donationid	int(11)	Foreign key	Donation id
paidamount	bigint(20)	Not null	Paid amount
paidpersondetails	varchar(255)	Not null	Paid person details
contactno	varchar(20)	Not null	Contact number
paymentmode	varchar(20)	Not null	Payment mode
otherdetails	varchar(255)	Not null	Other details

Table Name-receiver

Primary key-DRcode

Field	Type	Constraints	Description
Rcode	varchar(50)	Primary key	Receiver code
fname	varchar(50)	Not null	First name
lname	varchar(50)	Not null	Last name
age	int(11)	Not null	Age
gender	varchar(15)	Not null	Gender
bloodgroup	varchar(10)	Not null	Blood group
organs	varchar(100)	Not null	Organs
mobileno	varchar(20)	Not null	Mobile number
emergancyname	varchar(50)	Not null	Emergency name
emmobno	varchar(50)	Not null	Emergency mobile number
regdate	varchar(50)	Not null	Registration date

D. DATA NORMALIZATION

The entities along with their attributes can be stored in many different ways into a set of tables. The methods of arranging these attributes are called normal form. The theory behind the arrangement of attributes into table is known as normalization theory. Normalization is a series of test which we use against the data to eliminate redundancy and make sure that the data is associated with the correct table or relationship.

It helps in,

- Minimization of duplicate data.
- Providing flexibility to support different functional requirements
- Enabling the model to be translated to database design.

All relations in a relational database are required to satisfy the following conditions.

Data in First Normal Form:

- Remove repeating data from table
- From the removed data, created one or more tables and relationships

Data in Second Normal Form:

- Identify tables and relationships with more than one key.
- Remove data that depends on only one part of the key.
- From the removed data, created one or more tables and relationships.

Data in Third Normal Form:

- Remove that depends on other hand in the table or relationship and not on the key.
- From the removed data, created one or more tables and relationship.

Advantages of normalization are:

- Helps in reduction in the complexity of maintaining data integrity by removing the redundant data.
- It reduces inconsistency of data.
- Eliminates the repeating fields.

The second normal form has the characteristics of the first normal form and all the attributes must fully be dependent on the primary key. The proposed system is using second normal form as it is found most suitable.

5.Diagram

A. DATA FLOW DIAGRAM

Data Flow Diagram (DFD) is an important tool used by system analyst. DFD provide an overview of what data a system would process, what transformation of data are done, what files are used and where the results flow. The graphical representation of the system makes it a good communication tool between the user and the analyst.

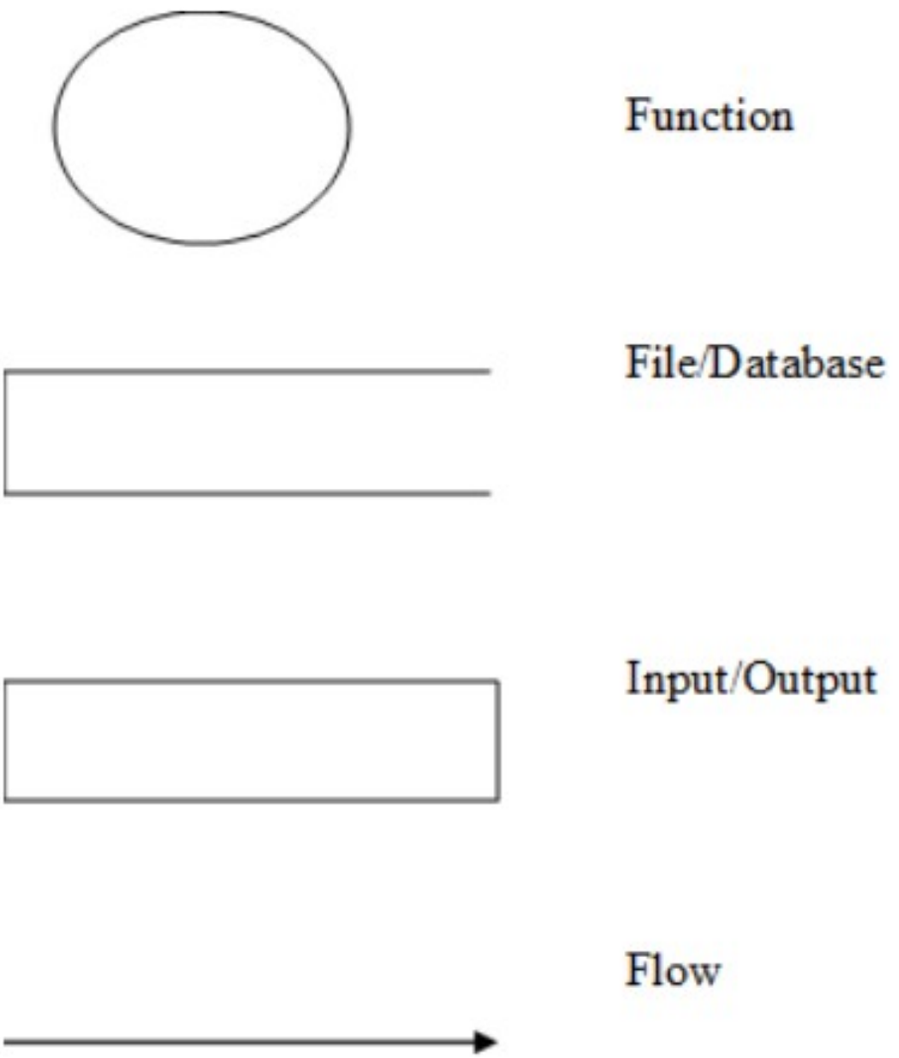
Analysis model help us to understand the relationship between different components in the design. Analysis model shows the user clearly how a system will function. This is the first technical representation of the system.

The analysis modeling must achieve three primary objectives

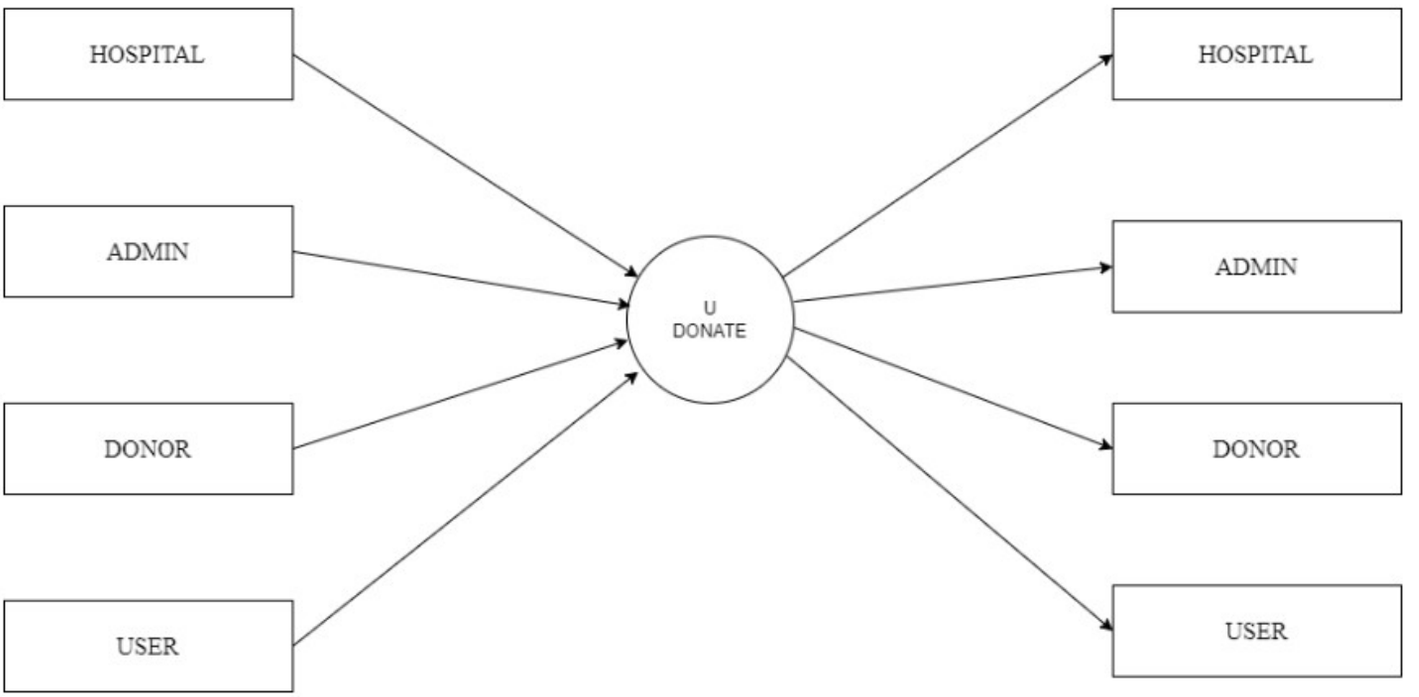
- To establish a basis for creation of software design
- To describe what the user requires
- To define set of requirements that can be validated once the software us build.

A data flow diagram is a graphical technique tat depicts information flow and transforms that are applied as data move from input to output. The DFD is used to represent increasing information flow and functional details. A level 0 DFD also called fundamental system model represents the entire software elements as single bubble with input and output indicated by incoming and outgoing arrow respectively.

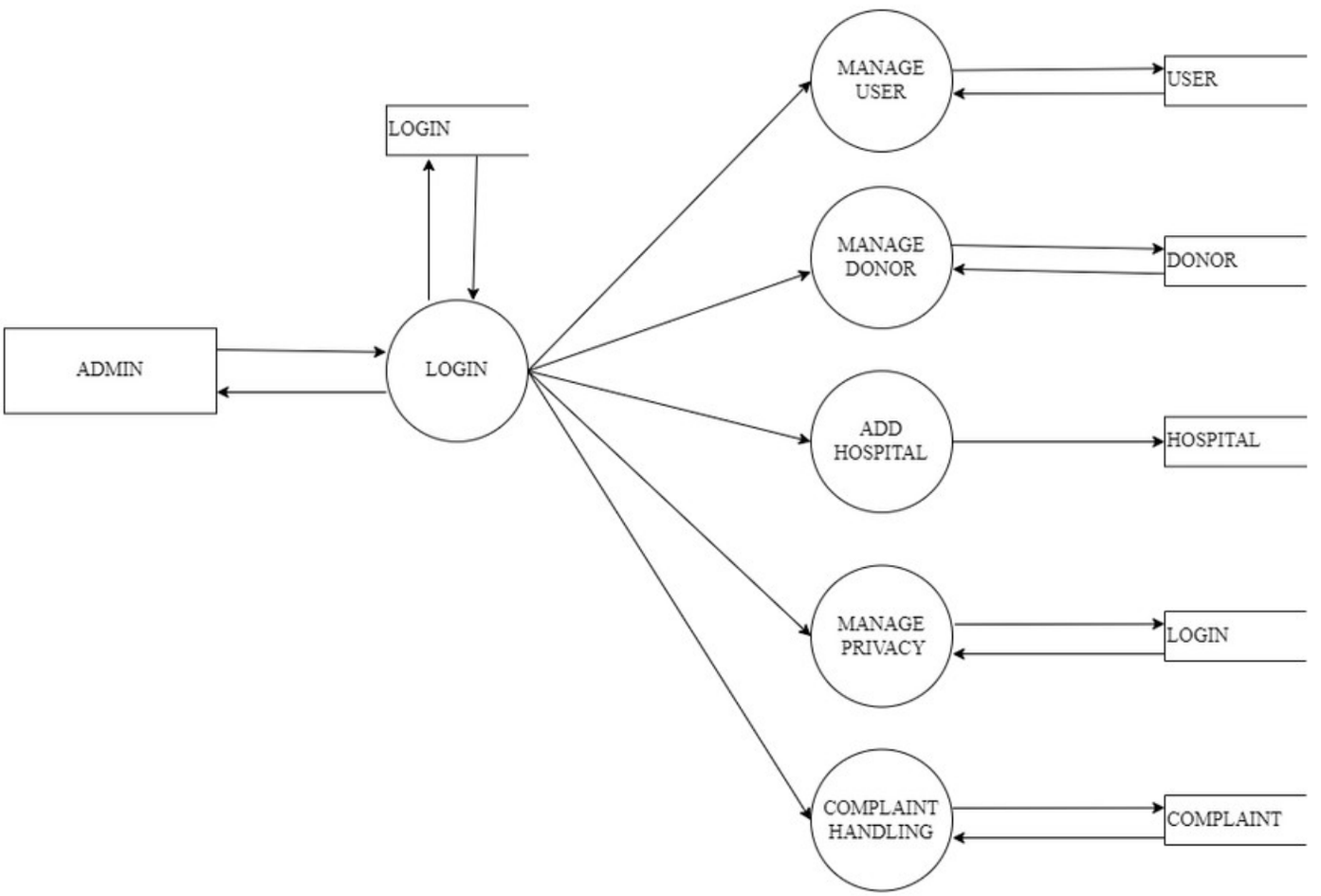
Components of Data Flow Diagram



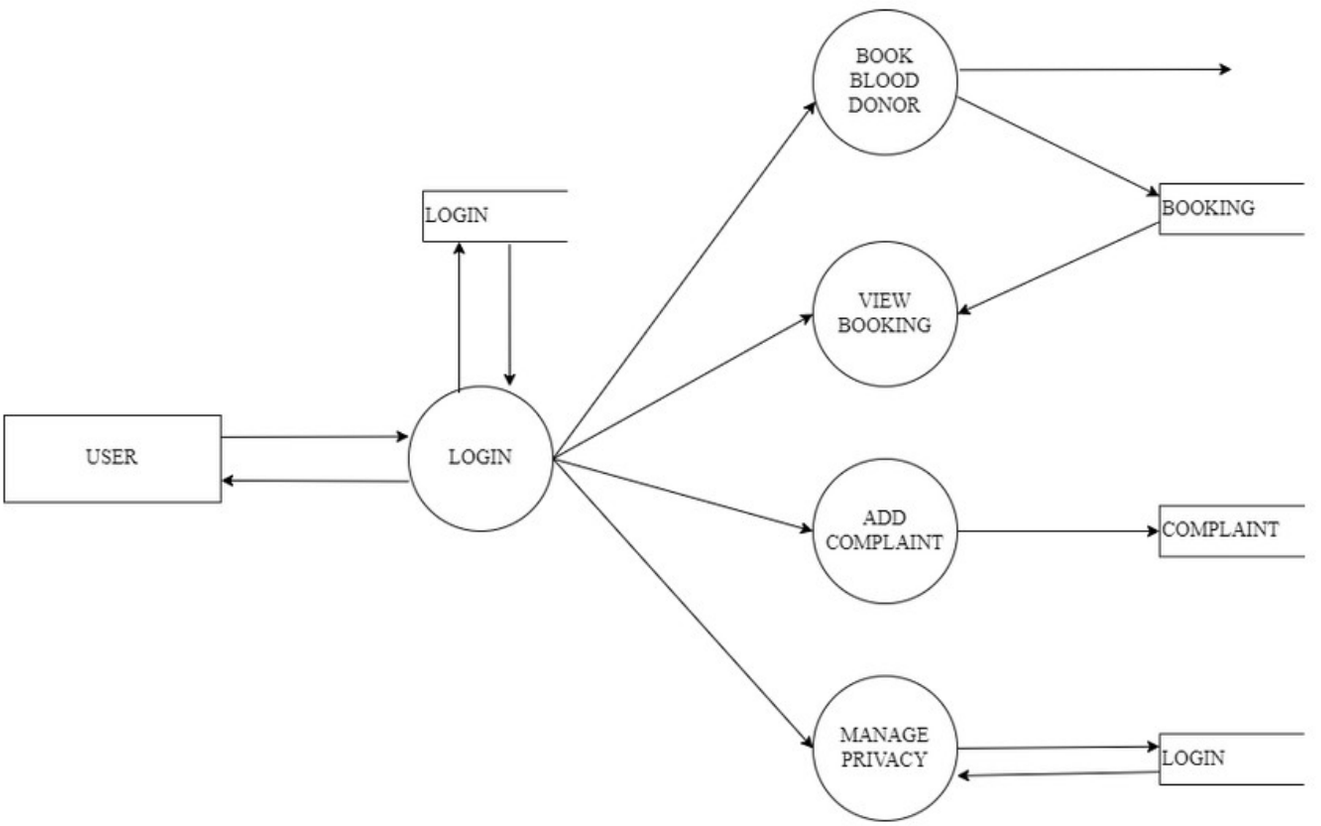
CONTEXT LEVEL DFD



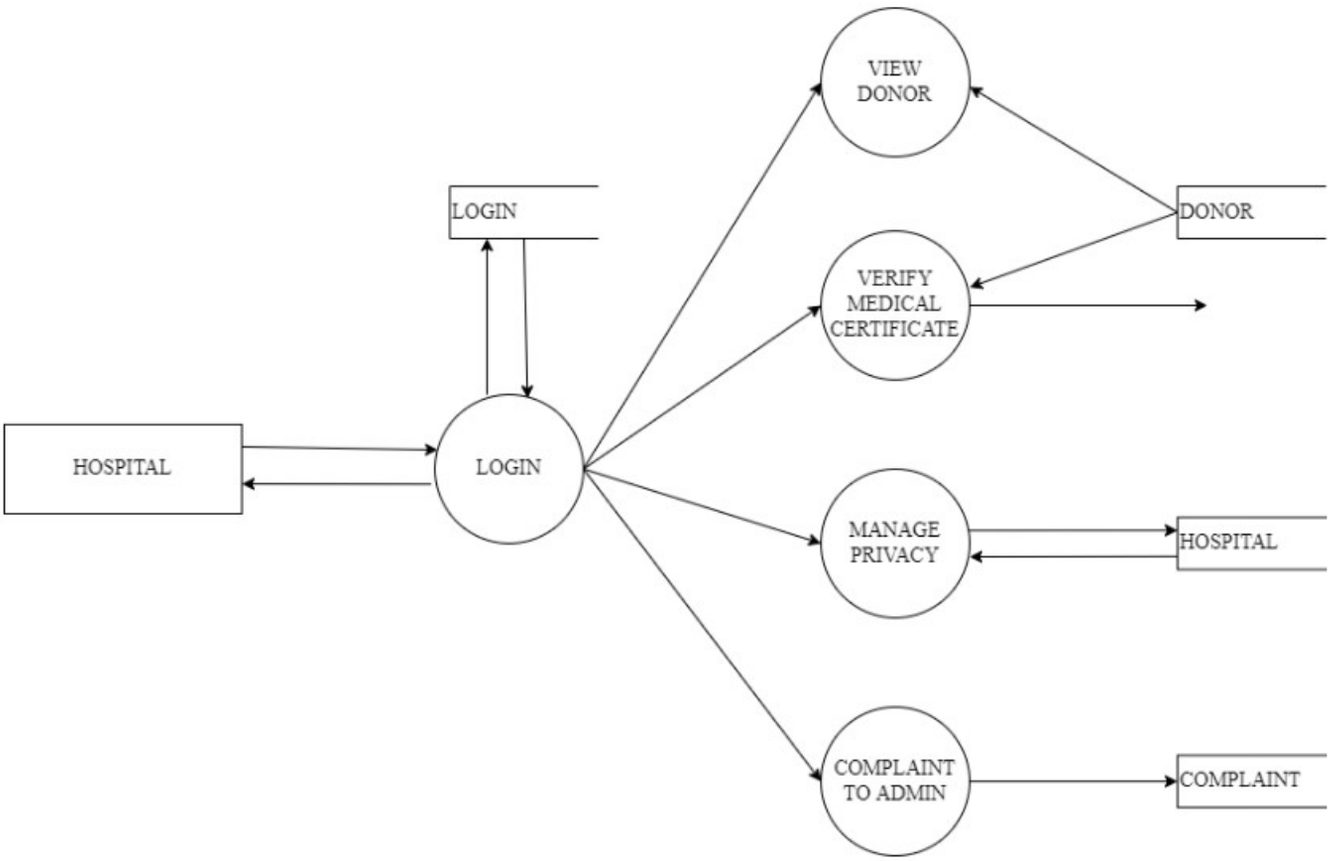
LEVEL 1[ADMIN]



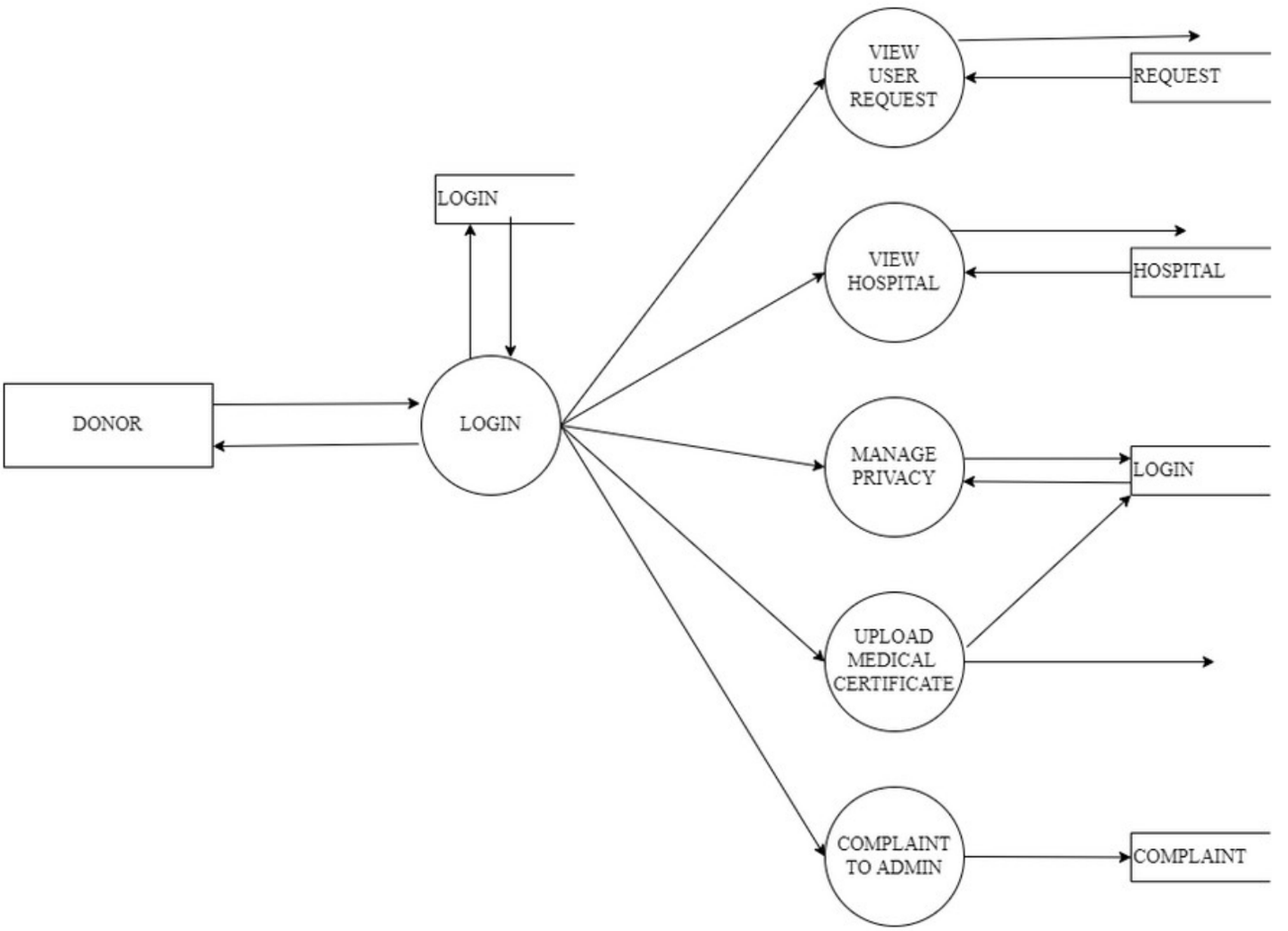
LEVEL 2:[USER]


BOOK
BLOOD
DONOR
LOGIN
BOOKING
VIEW
BOOKING
USER
LOGIN
ADD
COMPLAINT
COMPLAINT
MANAGE
PRIVACY
LOGIN

LEVEL 3:[HOSPITAL]



LEVEL 4:[DONOR LOGIN]



FLOW CHART

Flow chart that can be used for representing the flow of data in the system Flow chart that can be created before the coding is started. Because flow chart is very helpful to developer to identify the main functions coming in the system, and flow of data in the system. By analyzing the flow chart of a system we can develop that system very easily. Flow charts are used n designing and documenting complex processes. Like other types of diagram, the help to visualize what is going on an thereby help the viewer to understand a process , and perhaps also find flaws, bottlenecks, and other less- obvious features within it. There are many different types of flowcharts, and each type has its own repertoire of boxes and notational conventions. The two most common types of boxes in a flowchart are:

- A processing step, usually called activity, and denoted as rectangular box
- A decision usually denoted as a diamond

A flowchart is described as “cross-functional” when the page is divided into different swim lanes describing the control of different organizational units. A symbol appearing in a particular “ lane” is within the control of that organizational unit. This technique allows the author to locate the responsibility for performing an action or making a decision correctly, showing the responsibility of each organizational unit for different parts of a single process.

Symbols used in the Flowchart

: Start or Stop

Rounded Rectangle

: Processing

Rectangle

: Input or Output

Parallelogram

: Condition

Diamond

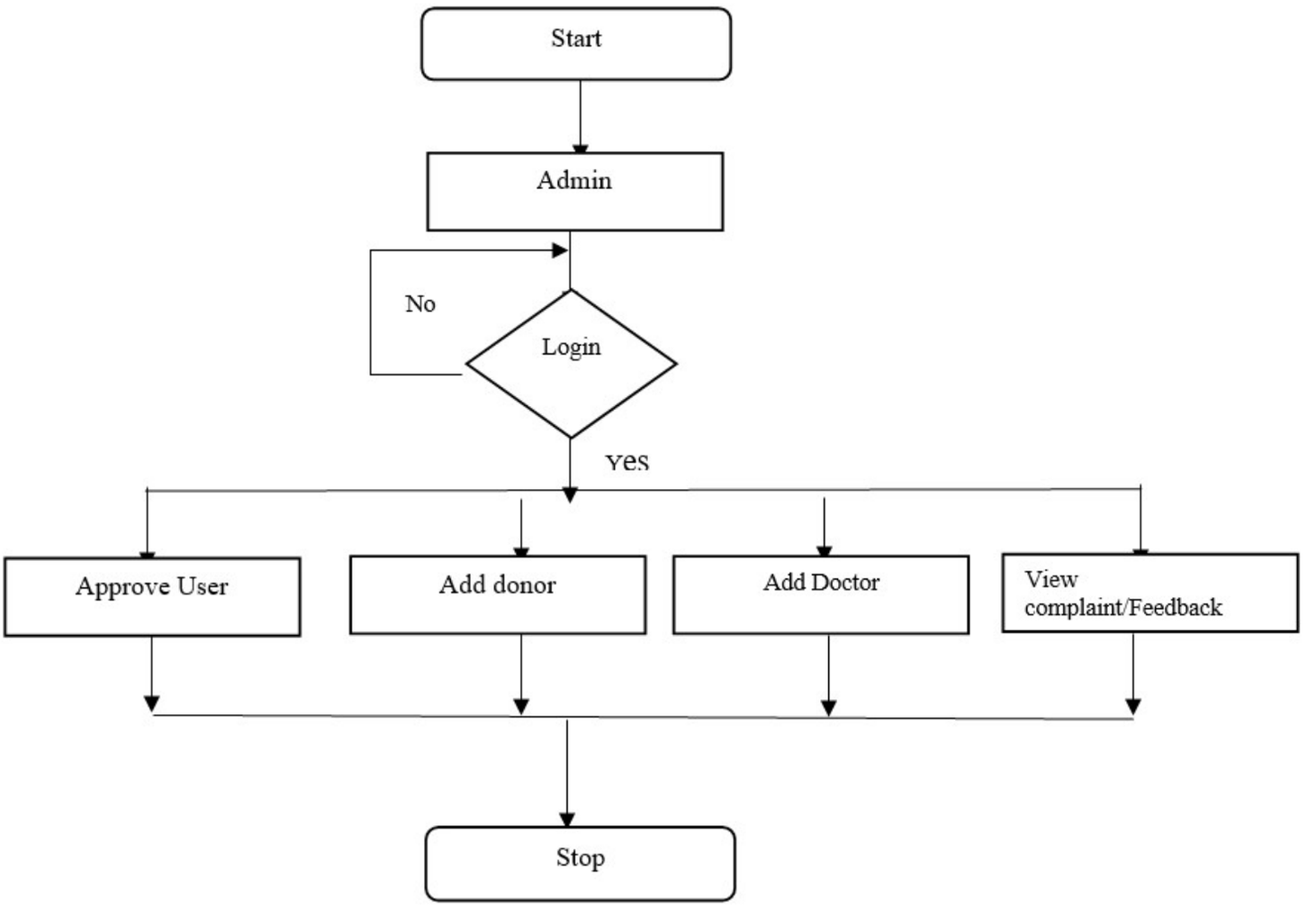
: Connector

Circle

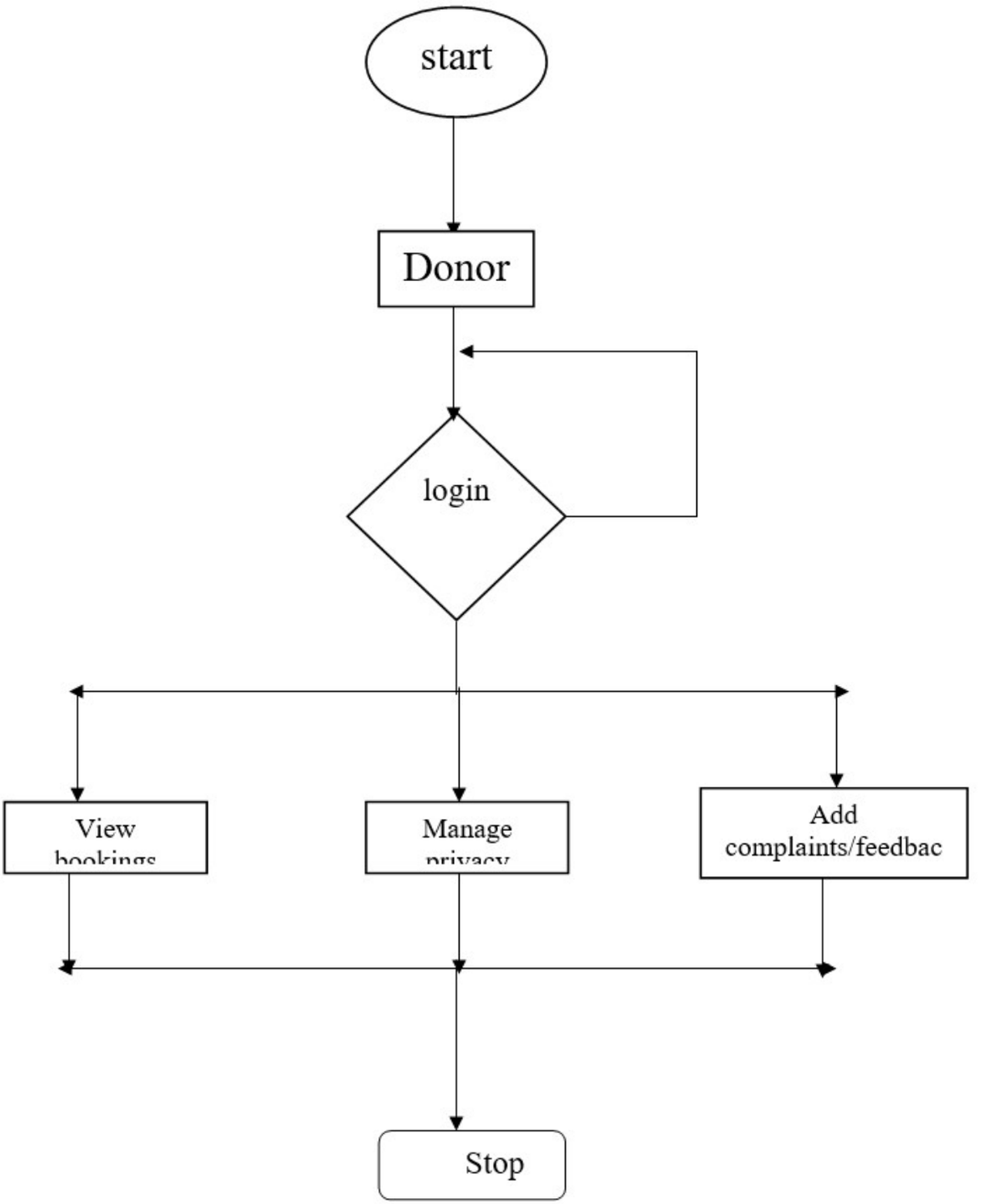
: Direction (Logi Flow)

Flow Line

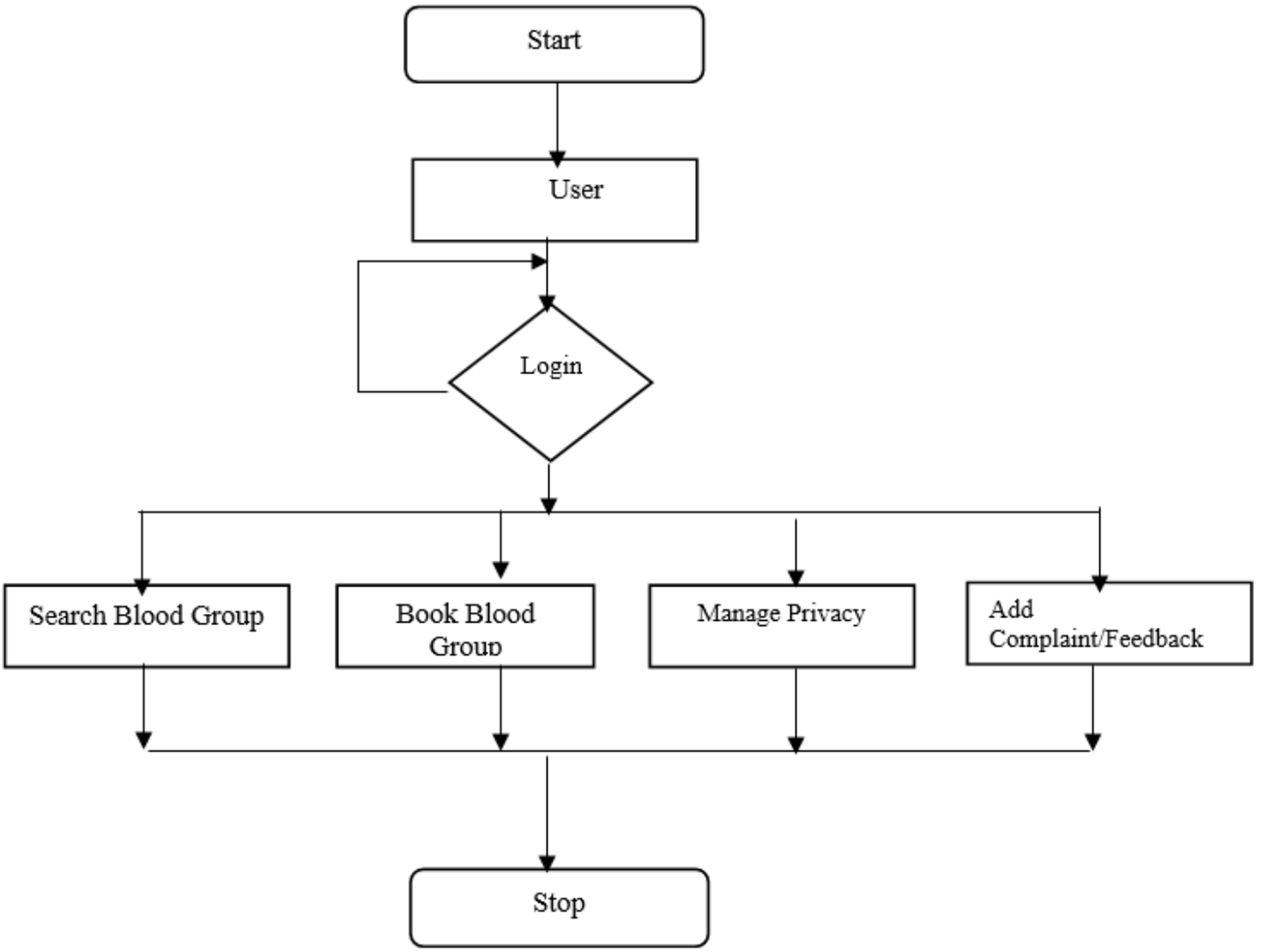
Admin


Start
Admin
No
Login
YES
Approve User
Add donor
Add Doctor
View complaint/Feedback
Stop

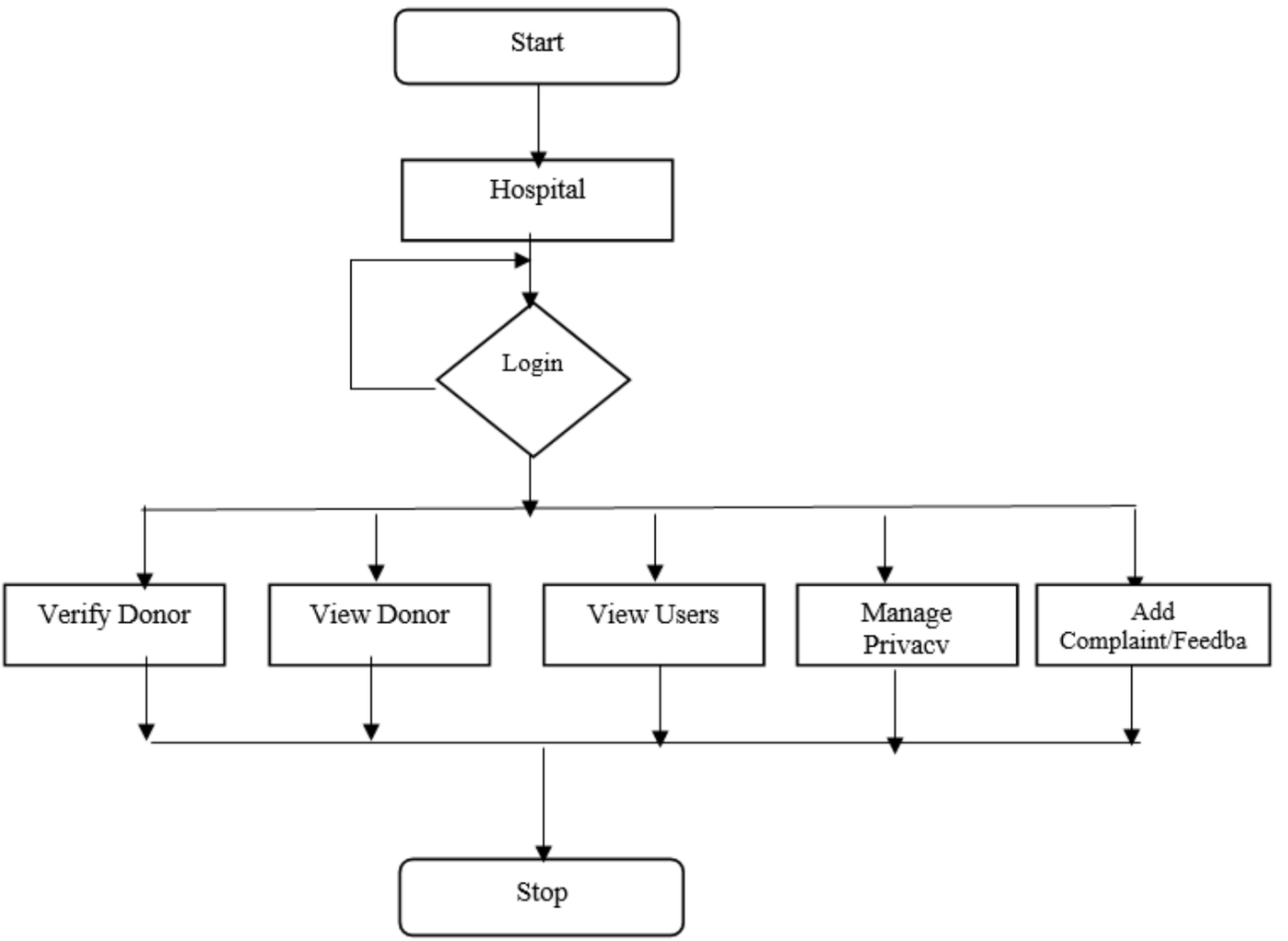
Donor



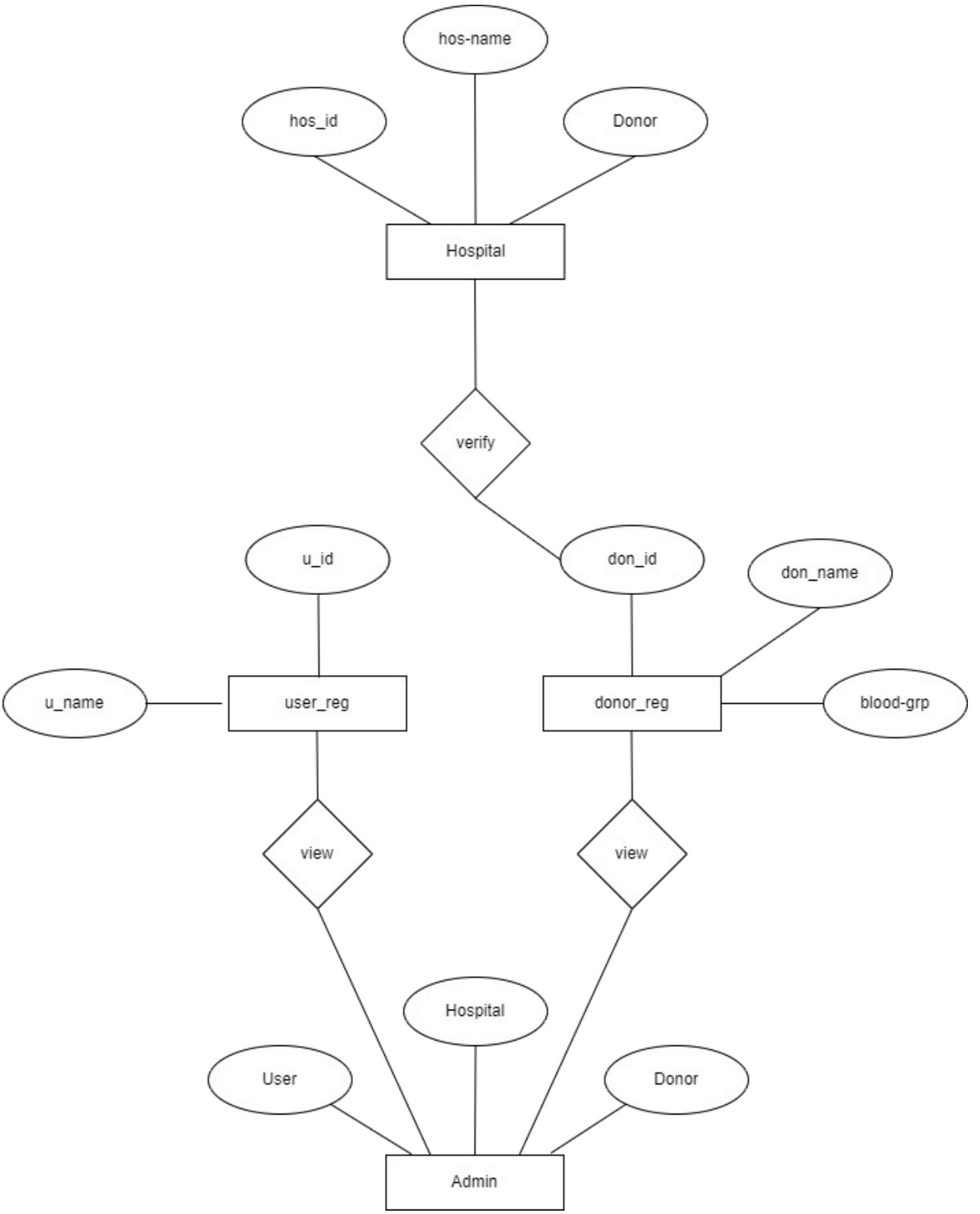
User


Start
User
Login
Search Blood Group
Book Blood Group
Manage Privacy
Add Complaint/Feedback
Stop

Hospital


Start
Hospital
Login
Verify Donor
View Donor
View Users
Manage Privacy
Add Complaint/Feedba
Stop

E R DIAGRAM


hos-name
hos_id
Donor
Hospital
verify
u_id
don_id
don_name
u_name
user_reg
donor_reg
blood-grp
view
view
Hospital
User
Donor
Admin

STRUCTURE CHART

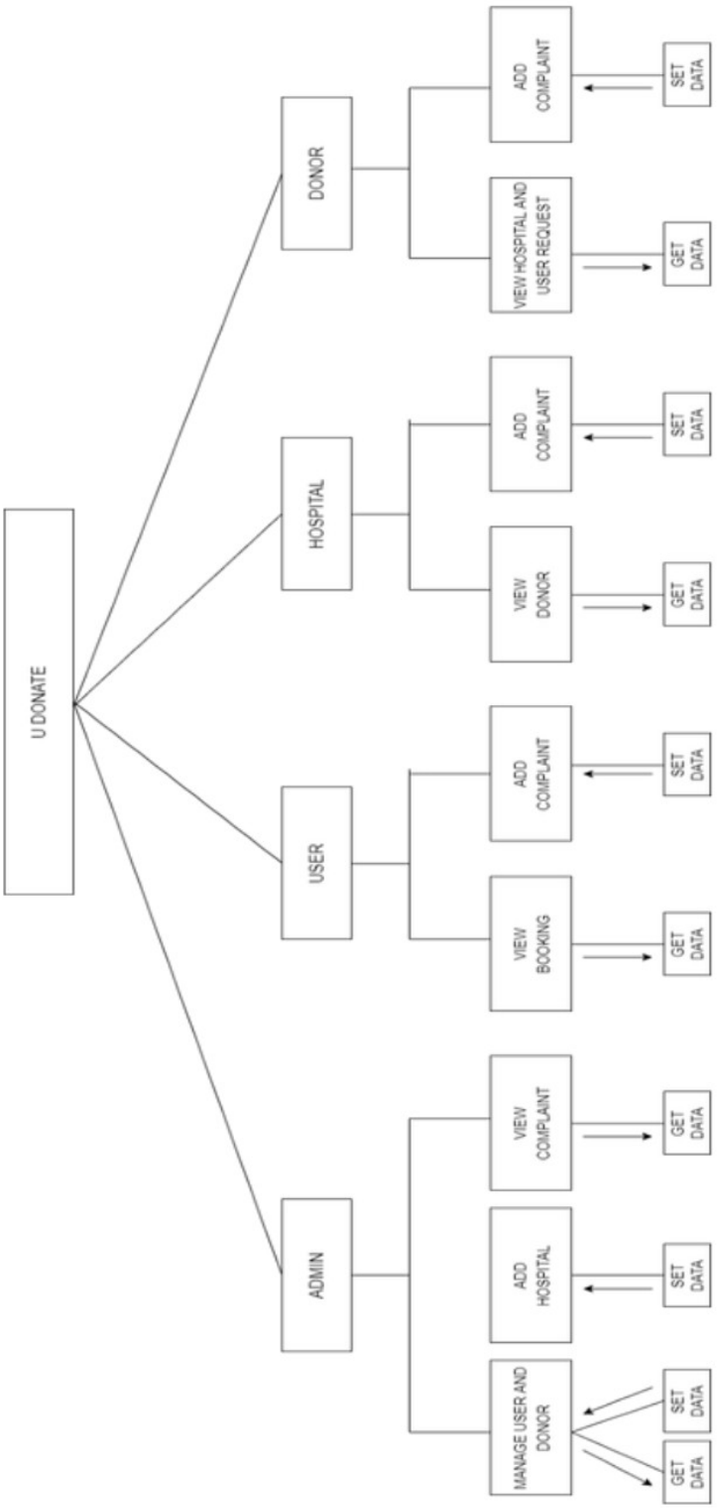
Structure chart is a design tool that pictorially shows the relation between processing module in computer software. Describes the hierarchy of components and the data are transmitted between them. Includes analysis of input-to-output transformations and analysis of transaction. Structure charts show the relation of processing modules in computer software. Structure charts are developed prior to the writing of program code. They identify the data passes existing between individual modules that interact with one another.

The basic building blocks using the structure are:

- A rectangle box represents a module (level). It is notated with the same name of the module.
- An arrow connecting two modules implies that during program execution, control is passed from one module to another in the direction of the connecting arrow.
- Data flow arrows are small arrows appearing along the side module invocation arrows. It is associated with the corresponding data same.

A structure chart (SC) in software engineering and organizational theory is a chart, which shows the breakdown of the configuration system to the lowest manageable levels. This chart is used in structured programming to arrange the program modules in a tree structure. Each module is represented by a box, which contains the modules name. The tree structure visualizes the relationships between the modules. A structure chart is a top down modular design tool constructed of squares representing the different modules in the system, and the lines that connect them. The lines represents the connection and or ownership between activities and sub activities as they are used in organization chart.

STRUCTURE CHART



7.CODING

```
 def index(request):
    msg=request.GET.get("msg",")
    if(request.session.get('role', ' ')=="admin"):
        response = redirect('/admin')
        return response
    elif(request.session.get('role', ' ')=="provider"):
        response = redirect('/staff')
        return response
    elif(request.session.get('role', ' ')=="user"):
        response = redirect('/user')
        return response
    else:
        return render(request,"index.html",{"msg":msg})
def admin(request):
    if(request.session.get('role', ' ')=="admin"):
        return render(request,"adminhome.html")
    else:
        response = redirect('/index'+"?msg=session expired login again")
        return response
def health(request):
    if(request.session.get('role', ' ')=="health"):
        msg="Registers successfuly"
        return render(request,"userhome.html")
    else:
        response = redirect('/index'+"?msg=session expired login again")
        return response
def staff(request):
    if(request.session.get('role', ' ')=="hospital"):
        return render(request,"staffhome.html")
```

```
    else:
        response = redirect('/index'+"?msg=session expired login again")
        return response
def User_Reg(request):
    msg=request.GET.get("msg",")
    return render(request,"UserReg.html",{"msg":msg})

def about(request):
    msg=request.GET.get("msg",")
    return render(request,"about.html",{"msg":msg})

def contact(request):
    msg=request.GET.get("msg",")
    return render(request,"contact.html",{"msg":msg})

def nurse(request):
    if(request.session.get('role', ' ')=="Nurse"):
        return render(request,"nursehome.html")
    else:
        response = redirect('/index'+"?msg=session expired login again")
        return response

def register_user(request):
    msg=""
    if request.POST:
        t1 = request.POST["t1"]
        t2 = request.POST["t2"]
        t3 = request.POST["t3"]
        t4 = request.POST["t4"]
        t7 = request.POST["t5"]
        t8 = request.POST["t6"]
        log.objects.create(username=t7, password=t8, role="user")
```

```
        data=log.objects.last()
     usr.objects.create(User_name=t1,User_email=t2,User_phone=t3,User_address=t4,
User_status="approved",Log_id=data)
        msg="Registers successfuly"
        return render(request,"index.html",{"msg":msg})
    else:
        msg = ""
    data1=usr.objects.all()
    return render(request,"UserReg.html",{"msg":msg,"data":data1})


def addhospital(request):
    msg=""
    data1=cate.objects.all()
    return render(request,"Staff_Reg.html",{"msg":msg,"data":data1})
def register_provider(request):
    msg=""
    if request.POST:
        t1 = request.POST["t1"]
        t2 = request.POST["t2"]
        t3 = request.POST["t3"]
        t4 = request.POST["t4"]
        t5 = request.POST["t5"]
        td = date.today()
        log.objects.create(username=t4, password=t5, role="hospital")
        data=log.objects.last()
        stf.objects.create(hname=t1,address=t2,phone=t3,Staff_logid=data)
        msg="Registers successfuly"
        data1=stf.objects.all()
        response = redirect('/admin')
        return response
    else:
def index(request):
```

```
    msg=request.GET.get("msg",")
    if(request.session.get('role', ' ')=="admin"):
        response = redirect('/admin')
        return response
    elif(request.session.get('role', ' ')=="provider"):
        response = redirect('/staff')
        return response
    elif(request.session.get('role', ' ')=="user"):
        response = redirect('/user')
        return response
    else:
        return render(request,"index.html",{"msg":msg})
def admin(request):
    if(request.session.get('role', ' ')=="admin"):
        return render(request,"adminhome.html")
    else:
        response = redirect('/index'+"?msg=session expired login again")
        return response
def health(request):
    if(request.session.get('role', ' ')=="health"):
        msg="Registers successfuly"
        return render(request,"userhome.html")
    else:
        response = redirect('/index'+"?msg=session expired login again")
        return response
def staff(request):
    if(request.session.get('role', ' ')=="hospital"):
        return render(request,"staffhome.html")
    else:


 response = redirect('/index'+"?msg=session expired login again")
        return response
```

```
def User_Reg(request):
    msg=request.GET.get("msg","")
    return render(request,"UserReg.html",{"msg":msg})

def about(request):
    msg=request.GET.get("msg","")
    return render(request,"about.html",{"msg":msg})

def contact(request):
    msg=request.GET.get("msg","")
    return render(request,"contact.html",{"msg":msg})

def nurse(request):
    if(request.session.get('role', ' ')=="Nurse"):
        return render(request,"nursehome.html")
    else:
        response = redirect('/index'+"?msg=session expired login again")
        return response

def register_user(request):
    msg=""
    if request.POST:
        t1 = request.POST["t1"]
        t2 = request.POST["t2"]
        t3 = request.POST["t3"]
        t4 = request.POST["t4"]
        t7 = request.POST["t5"]
        t8 = request.POST["t6"]
        log.objects.create(username=t7, password=t8, role="user")
        data=log.objects.last()
   usr.objects.create(User_name=t1,User_email=t2,User_phone=t3,User_address=t4,
User_status="approved",Log_id=data)
```

```
        msg="Registers successfuly"
        return render(request,"index.html",{"msg":msg})
    else:
        msg = ""
    data1=usr.objects.all()
    return render(request,"UserReg.html",{"msg":msg,"data":data1})


def addhospital(request):
    msg=""
    data1=cate.objects.all()
    return render(request,"Staff_Reg.html",{"msg":msg,"data":data1})
def register_provider(request):
    msg=""
    if request.POST:
        t1 = request.POST["t1"]
        t2 = request.POST["t2"]
        t3 = request.POST["t3"]
        t4 = request.POST["t4"]
        t5 = request.POST["t5"]
        td = date.today()
        log.objects.create(username=t4, password=t5, role="hospital")
        data=log.objects.last()
        stf.objects.create(hname=t1,address=t2,phone=t3,Staff_logid=data)
        msg="Registers successfuly"
        data1=stf.objects.all()
        response = redirect('/admin')
        return response
    else:
        msg = ""
    data1=stf.objects.all()
    return render(request,"List_staff.html",{"msg":msg,"data":data1})
```

```
def Login(request):
    msg=""
    data1=cate.objects.all()
    return render(request,"Login.html",{"msg":msg,"data":data1})
def Login_now(request):
    if request.POST:
        user = request.POST["username"]
        password = request.POST["password"]
        try:
            data = log.objects.get(username=user, password=password)

            if (data.role == "admin"):
                request.session['username'] = data.username
                request.session['role'] = data.role
                request.session['id'] = data.log_id
                response = redirect('/admin')
                return response
            elif (data.role == "hospital"):
                request.session['username'] = data.username
                request.session['role'] = data.role
                request.session['id'] = data.log_id
                response = redirect('/staff')
                return response
            elif(data.role =="health"):
                request.session['username'] = data.username
                request.session['role'] = data.role
                request.session['id'] = data.log_id
                response = redirect('/health')
                return response

            else:
                return render(request, "index.html", {"msg": "invalid account Details"})
```

```
        except:
            return render(request, "index.html", {"msg": "invalid user name or password "})
    else:
        response = redirect('/index')
        return response


def Profile(request):
    msg=""
    ids=request.session["id"]
    if request.POST:
        if request.session["role"] =="hospital":
            t2 = request.POST["t2"]
            t3 = request.POST["t3"]
            t4 = request.POST["t4"]
         stf.objects.filter(Staff_logid=ids).update(hname=t2,address=t3,phone=t4)
        elif request.session["role"] =="user":
            t1 = request.POST["t1"]
            t2 = request.POST["t2"]
            t3 = request.POST["t3"]
            t4 = request.POST["t4"]

usr.objects.filter(Log_id=ids).update(User_name=t1,User_address=t2,User_email=t4,User_phone=t3)



    if request.session["role"] =="hospital":
        data1=stf.objects.get(Staff_logid=ids)
        returnpage="StaffProfile.html"
    elif request.session["role"] =="user":
        data1=usr.objects.get(Log_id=ids)
        returnpage="Userprofile.html"
    else:
        response = redirect('/index'+"?msg=session expired login again")
```

```
        return response
    return render(request,returnpage,{"msg":msg,"datas":data1})


def viewhospital(request):
    msg=""
    data1=stf.objects.all()
    return render(request,"List_Staff.html",{"msg":msg,"data":data1})

def complaints(request):
        msg=""
        #datax=usr.objects.get(Log_id=request.session["id"])
        if request.POST:
                t1 = request.POST["t1"]
                t2 = request.POST["t2"]


                msg="updated sucessfully"
                comp.objects.filter(Complaint_id=t1).update(Complaint_reply=t2)
        data1=comp.objects.all()
        return render(request, "Answer_Queries.html",{"msg":msg,"data":data1})


def application(request):
        msg=""
        d=request.session['id']
        datal=log.objects.get(log_id=d)
        datau=stf.objects.get(Staff_logid=datal)
        #datax=usr.objects.get(Log_id=request.session["id"])
        if request.POST:
                ids=request.session["id"]
                t1 = request.POST["t1"]
                t2 = request.POST["t2"]


                msg="updated sucessfully"
```

```
cate.objects.create(r_date=t1,request=t2,adm_status="waiting",health_status="waiting",hid=datau)
        data1=cate.objects.filter(hid=datau).all()
        return render(request, "OrganApplication.html",{"msg":msg,"data":data1})


def healthApprove(request):
        msg=""
        d=request.session['id']
        datal=log.objects.get(log_id=d)
        #datax=usr.objects.get(Log_id=request.session["id"])
        if request.POST:
                ids=request.session["id"]
                t1 = request.POST["t1"]
                t2 = request.POST["t2"]


                msg="updated sucessfully"

cate.objects.create(r_date=t1,request=t2,adm_status="waiting",health_status="waiting",hid=datau)
        data1=cate.objects.all()
        return render(request, "healthApprove.html",{"msg":msg,"data":data1})


def view_medicare(request):
    msg = ""
    d=request.session['id']
    datal=log.objects.get(log_id=d)
    datau=stf.objects.get(Staff_logid=datal)
    data1=cate.objects.filter(hid=datau).all()
    return render(request,"view_medicare.html",{"msg":msg,"data":data1})


def adddoctor(request):
        msg=""
        d=request.session['id']
```

```
    datal=log.objects.get(log_id=d)
    datau=stf.objects.get(Staff_logid=datal)
    #datax=usr.objects.get(Log_id=request.session["id"])
    if request.POST:
        ids=request.session["id"]
        t1 = request.POST["t1"]
        t2 = request.POST["t2"]
        t3 = request.POST["t3"]
        t4 = request.POST["t4"]
        t5 = request.POST["t5"]
        msg="updated sucessfully"

comp.objects.create(dname=t1,doc_email=t2,doc_no=t3,doc_details=t4,doc_blood=t5,h_id=datau)
    data1=comp.objects.filter(h_id=datau).all()
    return render(request, "adddoctor.html",{"msg":msg,"data":data1})

def RegisteredUsers(request):
  msg = ""
  d=request.session['id']
  datal=log.objects.get(log_id=d)
  datau=stf.objects.get(Staff_logid=datal)
  data1=usr.objects.filter(hos_id=datau,User_role="donor").all()
  return render(request,"RegisteredUsers.html",{"msg":msg,"data":data1})
def view_receiver(request):
  msg=""
  d=request.session['id']
  datal=log.objects.get(log_id=d)
  datau=stf.objects.get(Staff_logid=datal)
  data1=usr.objects.filter(hos_id=datau,User_role="receiver").all()
  return render(request,"view_receiver.html",{"msg":msg,"data":data1})

def view_doner(request):
```

```
    msg=""
    d=request.session['id']
    datal=log.objects.get(log_id=d)
    datau=stf.objects.get(Staff_logid=datal)
    data1=pay.objects.filter(hsp_id=datau).all()
    return render(request,"view_response.html",{"msg":msg,"data":data1})

def view_doners(request):
    msg = ""
    data1=usr.objects.filter(User_role="donor").all()
    return render(request,"view_doner.html",{"msg":msg,"data":data1})

def view_receivers(request):
    msg=""

    data1=usr.objects.filter(User_role="receiver").all()
    return render(request,"view_receivers.html",{"msg":msg,"data":data1})


def viewdoctor(request):
    msg = ""
    d=request.session['id']
    datal=log.objects.get(log_id=d)
    datau=stf.objects.get(Staff_logid=datal)
    data1=comp.objects.filter(h_id=datau).all()
    return render(request,"viewdoctor.html",{"msg":msg,"data":data1})

def viewblood(request):
    msg = ""
    d=request.session['id']
    datal=log.objects.get(log_id=d)
    datau=stf.objects.get(Staff_logid=datal)
```

```
    data1=fed.objects.filter(hospi_id=datau).all()
    return render(request,"viewblood.html",{"msg":msg,"data":data1})

def viewnearhospital(request):
    msg = ""
    d=request.session['id']
    datal=log.objects.get(log_id=d)
    datau=stf.objects.get(Staff_logid=datal)
    data1=ser.objects.filter(hid=datau).all()
    return render(request,"viewnearhospital.html",{"msg":msg,"data":data1})


def view_notification(request):
    msg = ""
    d=request.session['id']
    datal=log.objects.get(log_id=d)
    datau=stf.objects.get(Staff_logid=datal)
    data1=msgg.objects.filter(hsp_id=datau).all()
    return render(request,"view_notification.html",{"msg":msg,"data":data1})

def Logout(request):
    try:
        del request.session['id']
        del request.session['role']
        del request.session['username']
        response = redirect("/index?id=logout")
        return response
    except:
        response = redirect("/index?id=logout")
        return response
```

8. SYSTEM TESTING

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer based system.it begins where integration testing is completed.

System Testing involves testing the software code for following

- Testing the fully integrated applications including external peripherals in order to check how components interact with one another and with the system as a whole. This is also called End to End testing scenario.
- Verify thorough testing of every input in the application to check for desired outputs.
- Testing of the user's experience with the application. .

Testing is vital to success of the system. System testing makes logical assumption that if all parts of the system are correct, the goal will be successfully achieved. Another reason for system testing is its utility as a user- oriented vehicle before implementation.

TEST PROCEDURE

For any software that is newly developed, first and foremost preference is given to the testing of the system. It is developer‘s last chance to detect and correct the errors. That may occur possibly in the software. The programmers will generate a set of test data, which will give the maximum possibility of finding all most all types of errors that can occur in the system.

A. UNIT TESTING

The primary goal of unit testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and determine whether it behaves exactly as you expect. Each unit is tested separately before integrating them into modules to test the interfaces between the modules. Unit testing has proven its value in that a large percentage of defects are identified during its use.

The most common approach to unit testing requires drivers and stubs to be written. The driver simulates a calling unit and the stub simulates a called unit. The investment of developer time in this activity sometimes results in demoting unit testing to a lower level of priority and that is almost always a mistake. Even though the drivers and stubs cost time and money, unit testing provides some undeniable advantages. It allows for automation of the testing process, reduces difficulties of discovering errors contained in more complex pieces of the application, and test coverage is often enhanced because attention is given to each unit.

B. BLACK BOX TESTING

In black box testing a test cases are designed from an examination of the input or output values only and no knowledge of design or code is required. The following are the two main Approaches to designing Black box test cases.

3° Equivalence class partitioning. '3' Boundary value analysis.

C.WHITE BOX TESTING

This method of software testing tests the internal structures or working of an application, as opposed to its functionality (i.e. black-box testing). In white box testing an internal perspective of the system, as well as programming skills, are required and used to design test case. In this we choose inputs to exercise paths through the code and determine the appropriate outputs.

White box testing is a testing case design method that uses control structure of the procedural design to derive the test cases. The entire independent path in a module is exercised at least once. All the logical decisions are exercised at least once. Executing all the 1oops at boundaries and within their operational bounds exercise internal structure to ensure their validity.

D.VALIDATION TESTING

Data validation is the process of testing the accuracy of data. A set of rule we can apply to a control to specify the type and range of data that can enter. It can be used to display error alert when users enter incorrect values in to a form.Now performing validation testing in system Centralized Social Welfare by undergoing validation for each tools and the validation succeeded when the software function in a manner that can be reasonably accepted, by the user.

E. ALPHA AND BETA TESTING

After all the development tests are carried out, it is necessary to test the system in user's environment. It would help the developer to know how the customer will really use the system. The developer does alpha testing in collaboration with the end user. The developer guides the end user all the way while observing reactions and feedback. Beta testing, on the other hand, releases the product to the end user. The user records all the problems (real and imaginary) that are encountered during beta testing and reports these to the developer at regular intervals. As a result of this the software developer makes modifications and then prepares for the release of the software product to entire customer base. The customer conducts the alpha testing at the developer's site and the beta testing is conducted at one or more customer's site by the end user of the software.

F. OUTPUT TESTING

After performing the validation testing, the next step is output testing of the proposed system since no system could be useful if it doesn't produce the required data in the specific format. The output displayed or generated by the system under consideration is tested by, asking the user about

the format displayed. The output format on the screen is found to be correct as the format was designed in the system according to the user needs. Hence the output testing doesn't result in any correction of the system.

G. ACCEPTANCE TESTING

This the final stage in the testing process before the system is accepted for operational use. Any requirement problem or requirement definition problem relived from acceptance testing are considered and made error free. User acceptance of the system is the key factor for the success of the system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing and making change wherever required. This is done with regard to Output screen design, input screen design, menu driven system.

H. NEED FOR TESTING

Testing is essential as:

- Existence of program defects of inadequacies is inferred.
- Verifies whether the software behaves as intended by its designer.
- Checks conformance with requirement specification/user needs.
- Assesses the operational reliability7 of the system.
- Tests the performance of the system.
- Reflect the frequency of actual user inputs.
- Finds the fault, which caused the output anomaly.
- Detects flaws and deficiencies in the requirements. Checks whether the software is operationally useful.
- Exercises the program using data like the real data processed by the program.

9. IMPLEMENTATION

A. INTRODUCTION TO IMPLEMENTATION

Implementation is the process of converting a new or revised system design into an operation. It is the key stage in achieving a successful new system because, usually it reveals a lot of up heal the use department. It must therefore be carefully planned and controlled. Apart from planning the two major tasks of preparing for implementation are education and training of users and testing of the system. Education of users should really take place much earlier in the project i.e. when they are involved in the investigation and design work .Training has to be given to the staff regarding the new system. Once staff has been trained, the system can be tested .Implementation is the stage of the project where the theoretical design is turned into working system or it is the key stage in achieving a successful new system, because usually it involves a lot of up heal in the user department. Therefore it must be carefully planned and controlled. It can also be considered to be the most crucial stage in achieving a successful new system and in giving the user confidence that the new system will work and be effective. Implementation is the final and important phase.

It is the phase where theoretical design is turned into working system, which works for the user in the most effective manner. It involves careful planning; investigation of the present system and the constraints involved, user training, system testing and successful running of developed proposed system. The implementation process begins with preparing a plan for the implementation of the system. According to this plan, the activities are to be carried out, discussions made regarding the equipment and resources and the additional equipment has to be acquired to implement the new system. The user tests the developed system and changes are made according to their needs. The testing phase involves the testing of a system using various kinds of data. This method also offers the greatest security since the old system can take over if the errors are found or inability to handle certain type of transactions while using new system.

B. IMPLEMENTATION OF PROPOSED SYSTEM

After having user acceptance for the system developed, the implementation phase begins. Implementation is the stage of project during which theory is turned into practice. During this phase, all the programs of the system are loaded into the user's computers. After loading the system, training of the users starts. Such type of training includes:

- How to execute the package.
- How to enter the data.
- How to process the data (processing details).
- How to take the reports.

The following two strategies are followed for running the system:

- Parallel Run: In such run for a certain defined period, both the systems therefore computerized and manual are executed in parallel. This strategy is helpful because of the following:
- Manual result can be compared with the result of computerized system. For the case of demonstration of the success of this system, it was implemented with successfully running; manual systems are results are verified.
- Failure of a computerized system at an early stage, do not affect the work of the Organization, because the manual system continues to work as it used to do.
- Pilot Run: In this type of run, the new system is installed in parts. Some parts of the new system is installed first and executed successfully for the considerable time period.
- When the results are found satisfactory, only then the other parts are implemented. This strategy builds the confidence and the errors are traced easily.

C. SYSTEM INSTALLISATION PROCEDURE

To install the system, the primary needs is web based environments without which the system will not have proper utilization. To install the system, it is a must to setup a centralized server which

can hold social networking website including the user's information database. The database is accessed through web pages using browser at the client end. In order to have the server setup for the GRUMBLE CELL, the following components are needed at the server end.

- Before installing the software, make sure that the Adobe Dreamweaver has installed in the system.
- Apache HTTP Server 2.2 and PYTHON must be installed.
- MS SQL 2005 Server (with our user information database) or higher version is installed before installation.
- You have to make sure that all the application has to be closed before installation.
- The database can be SQL Server 2005. The client side can have any operating system capable of getting connected to internet.

D. LINUX

Linux is a Unix-like and POSlX -compliant computer operating system assembled under the model of free and open source software development and distribution. The defining component of Linux is the Linux kernel, an operating system kernel first released on 5 October 1991 by Linus Torvalds.

Linux was originally developed as a free operating system for Intel x86-based personal computers. It has since been ported to more computer hardware platforms than any other operating system. It is a leading operating system on servers and other big iron systems such as mainframe computers and supercomputers: as of June 2013, more than 9500 of the world's 500 fastest supercomputers run some variant of Linux, including all the 44 fastest.

The development of Linux is one of the most prominent examples of free and open source software collaboration: the underlying source code may be used, modified, and distributed commercially or non commercially by anyone under licenses such as the GNU General Public License. Typically, Linux is packaged in a format known as a Linux distribution for desktop and server use. Some popular mainstream Linux distributions include Debian , Ubuntu, Linux Mint, Fedora, Arch Linux, and the commercial Red Hat Enterprise Linux and SUSE Linux Enterprise Server. Linux distributions include the Linux kernel, supporting utilities and libraries and usually a large amount of application software to fulfill the distribution's intended use.

A distribution oriented toward desktop use will typically include X11 or Wayland as the windowing system, and an accompanying desktop environment such as GNOME or the KDE Software Compilation. Some such distributions may include a less resource intensive desktop such as LXDE or Xfce, for use on older or less powerful computers. A distribution intended to run as a server may omit all graphical environments from the standard install, and instead include other software to set up and operate a solution stack such as LAMP. Because Linux is freely redistributable, anyone may create a distribution for any intended use .Linux kernel-based operating systems have found wide adoption and a very far-reaching range of use. All the advantages and benefits of free and open-source software apply to the Linux kernel itself and also to most of the rest of the system software.

Linux on the desktop

The common human interface devices (HIDs) available for desktop computers, 1aptops and similar devices determine the design of the (graphical) human-computer interface implemented into software. There are a couple of software packages to chose among, when building an accordingly designed graphical user interface. The generic input driver for the Linux kernel is evdev, but here are several input methods implemented as middleware, i.e. on top and not as part of the Linux kernel.

MAINTENANCE

Maintenance is a characteristic of design and implementation, which is expressed, as a probability that can an item will be retained in or restore to a specific condition within given period of time, when maintenance is performed to accordance with the prescribed procedures and resources.

Maintenance is the enigma of system development. It holds the software industry captive, trying up programming resources. Analyst and programmers spend far more time maintaining program than they do writing them.

Maintenance can classified as corrective adaptive, prefecture. Corrective maintenance means repairing process or performance or modifying the programming to respond to the users additional or changing needs of this type more money as Spend on a prefecture than on corrective and adaptive maintenance.

Technical and management approaches to the maintenance phase can be implemented with the upheaval. However tasks performed during the software engineering process defined maintainability and have an important on the success of maintenance approach.

CONCLUSION

This page is meant for the technology conclusions of the project report. In the former pages, all details about the development of software have been explained.

Blood Donation is developed as web application using **PYTHON** as front end and **MySQL** as back end. Every effort has been made to make the system as user friendly as possible. All the activities provide a feeling like an easy walk over to the user who is interacting with the system. Trial run of the system has been made and gave good results.

The new system has overcome most of the limitations of the existing system and works according to the design specification given. The developed systems dispense the problem and meet the needs of by providing reliable and comprehensive information. All the requirements projected by the user have been met by the system.

All the modules are tested separately and put together to from the main system.

Blood Donation is designed in an attractive fashion to generate user interest. Thus the project was successfully completed in the allotted span of time with satisfactory performance.

The key features of the system are:

- User friendly
- Resource requirement is less
- Ease in handling and implementation
- Adaptable to future enhancements

FURTHER ENHANCEMENT

Enhancement means adding, modifying, or developing the code to support the changes in the specification. It is the process of adding new capabilities such as reports, new interfaces with other systems and new feature such as better screen or report layout

The proposed system is developed and tested with some amount of sample data, which satisfy all the requirements. It has high degree of accuracy, user friendliness efficiency. The system is flexible and changes, if any can be made with out much difficulty. Further improvements and extensions can be made in the system to make overall work easier.

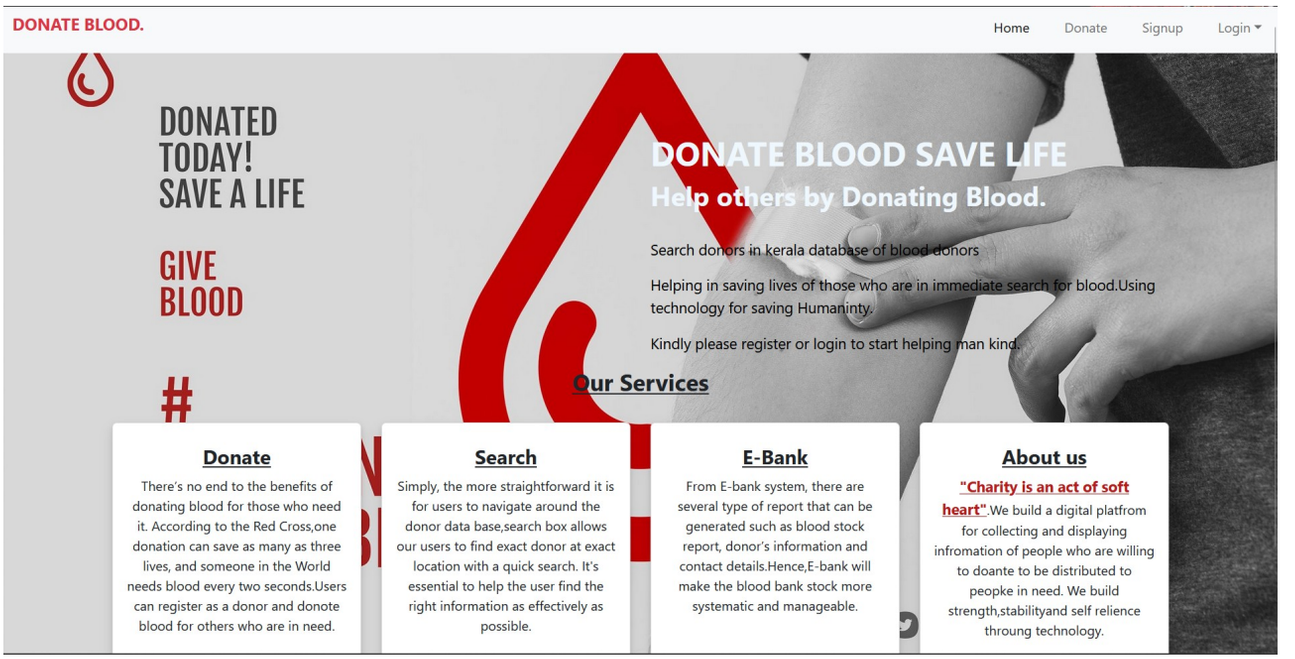
Further modifications and extensions in the system can made to make overall work easier like the library make online. Those options are not in the software can be included to improve the efficiency of the software.

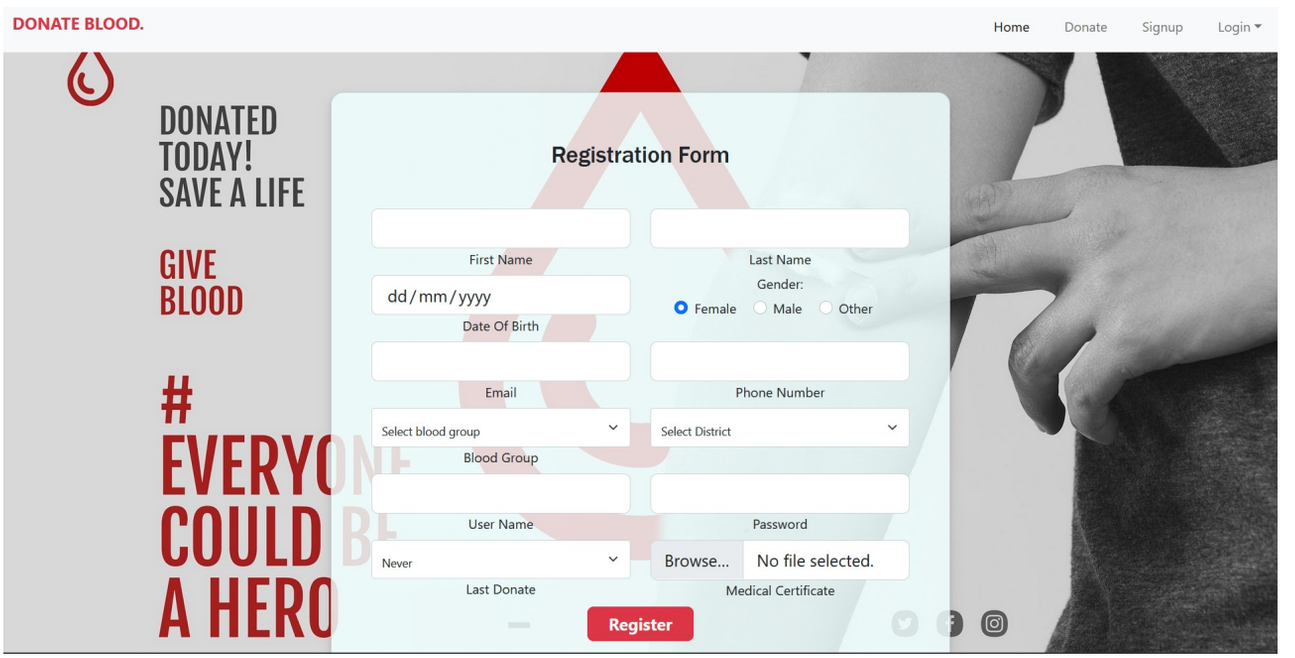
The major enhancements are:

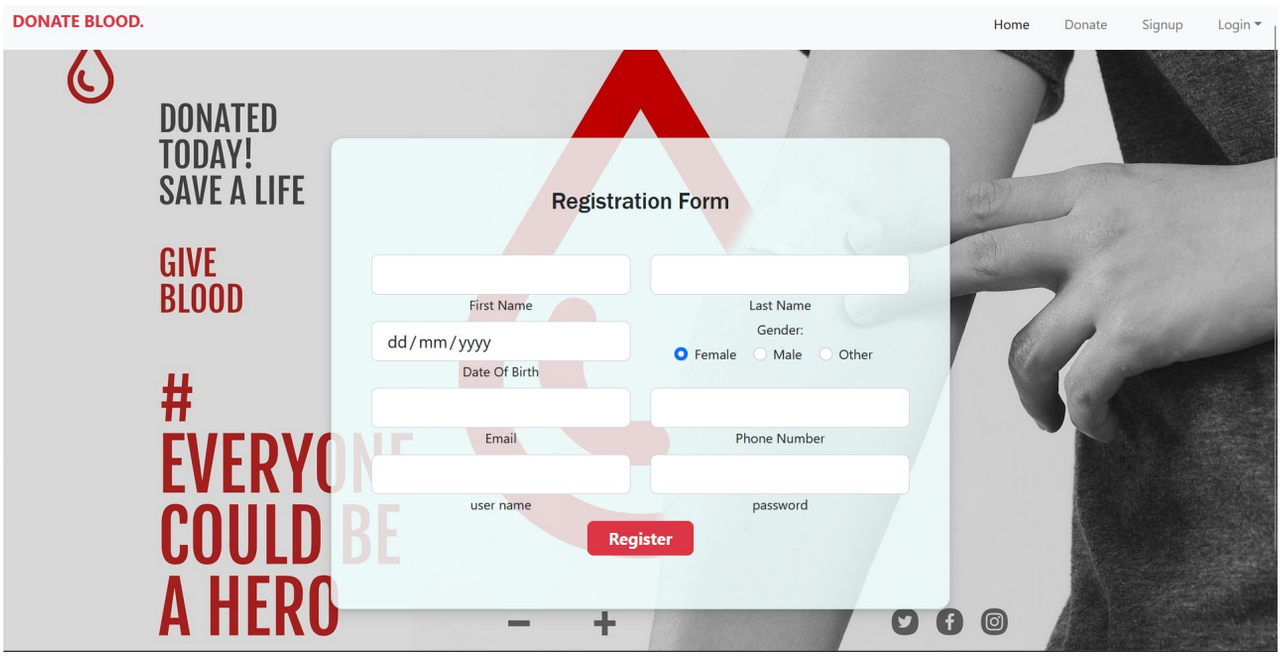
- Online facility
- More report preparation
- Adding more option for searching

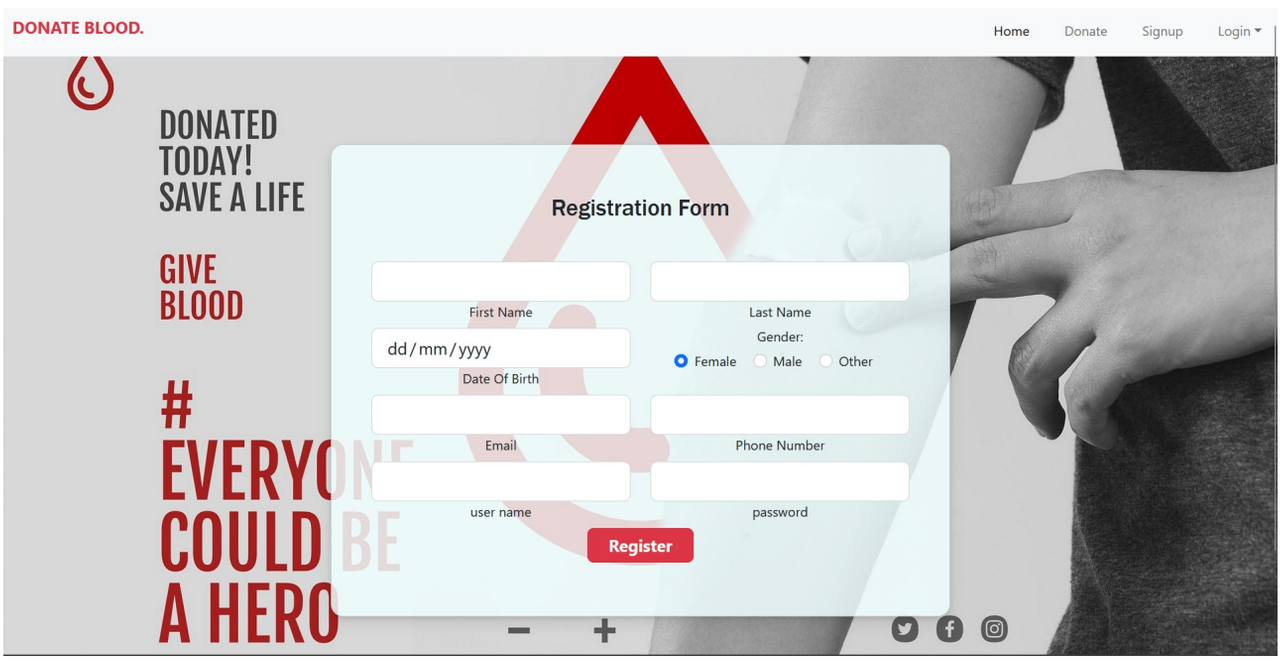
APPENDIX

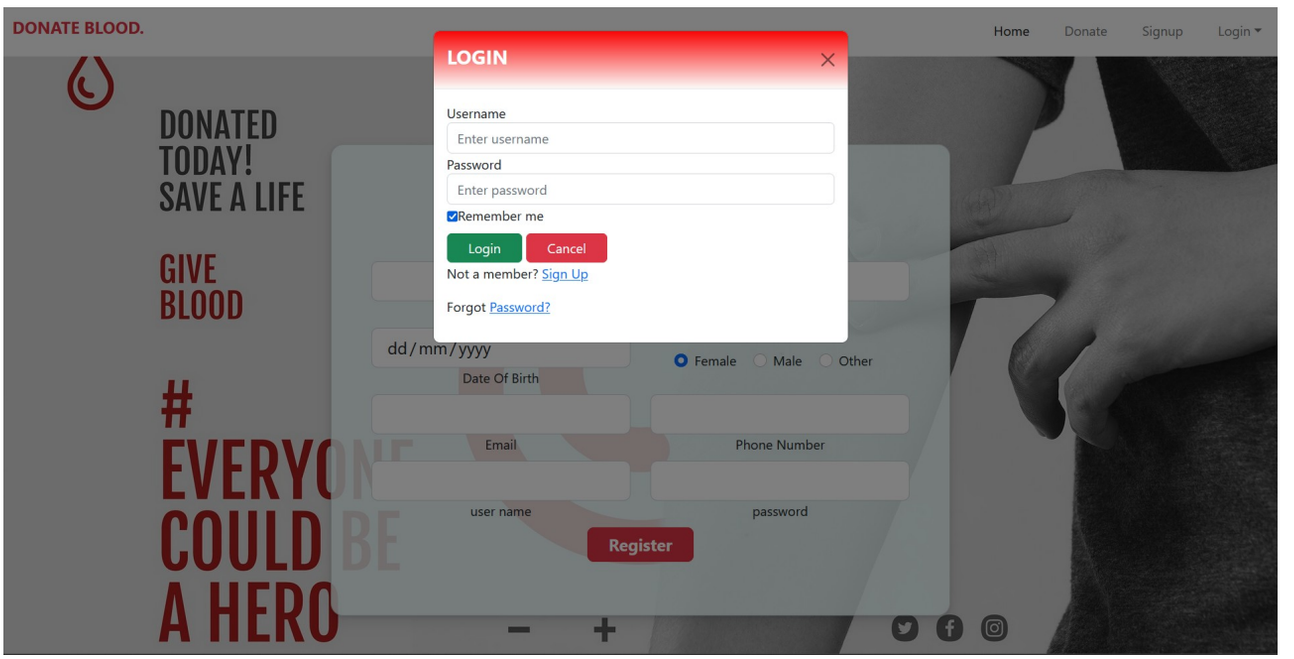
APPENDIX 1

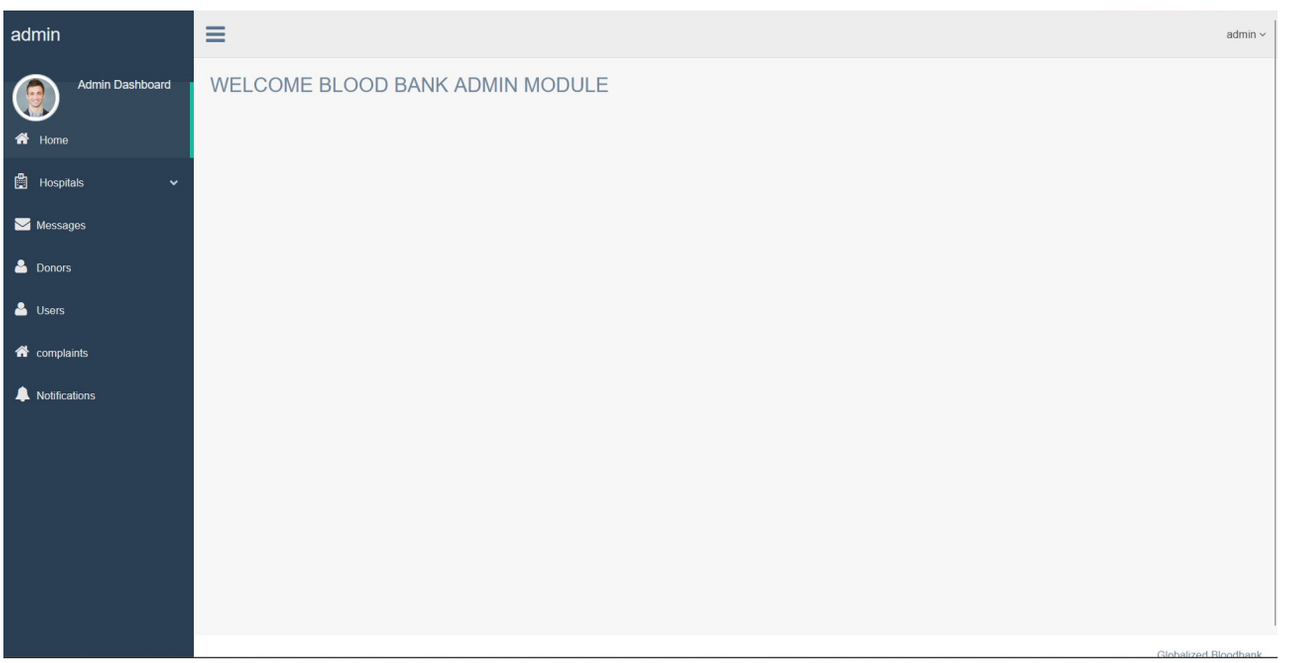


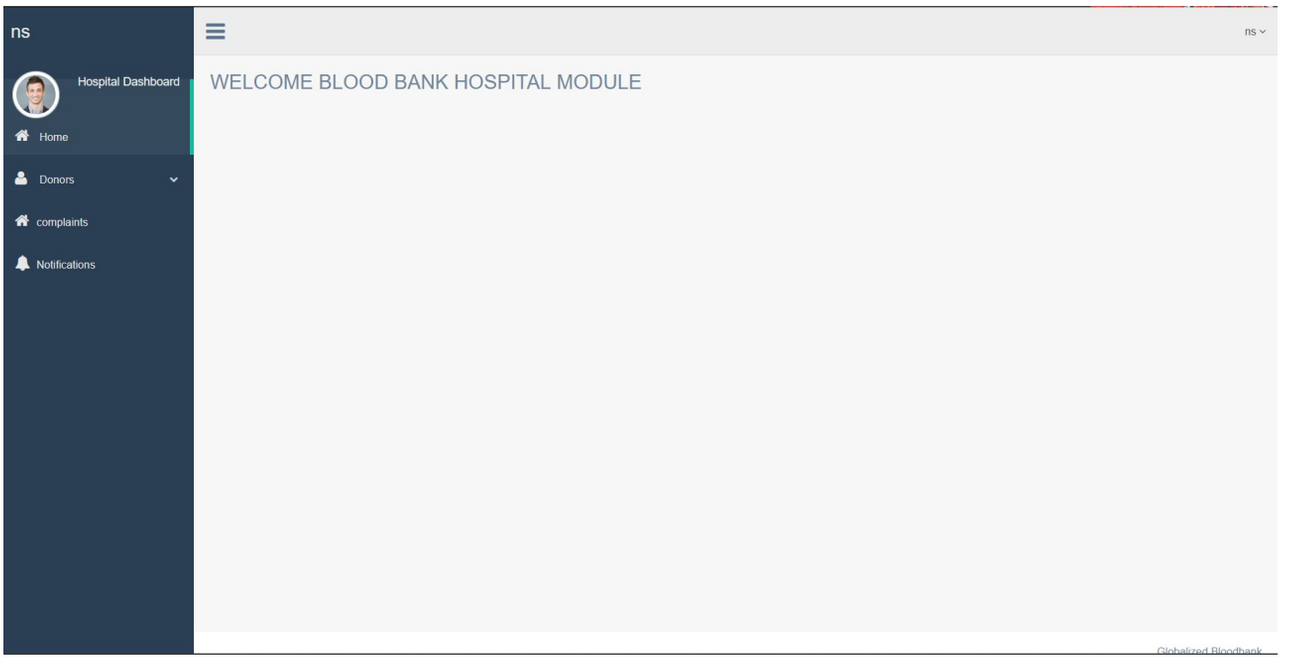



DONATE BLOOD.
Home
Donate
Signup
Login
DONATED TODAY! SAVE A LIFE
GIVE BLOOD
EVERYONE COULD BE A HERO
Registration Form
First Name
Last Name
Gender:
dd/mm/yyyy
Female
Male
Other
Date Of Birth
Email
Phone Number
user name
password
Register

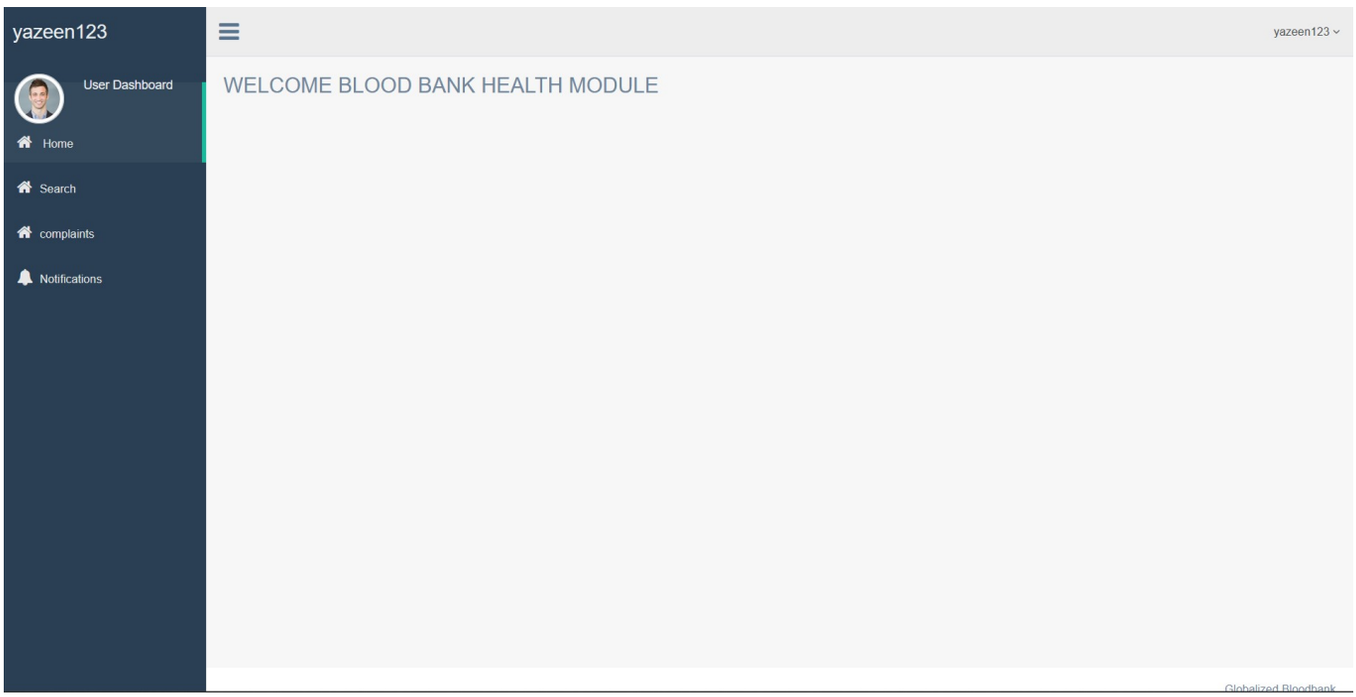

DONATE BLOOD.
Home
Donate
Signup
Login
DONATED TODAY! SAVE A LIFE
GIVE BLOOD
EVERYONE COULD BE A HERO
Registration Form
First Name
Last Name
Gender:
dd/mm/yyyy
Female
Male
Other
Date Of Birth
Email
Phone Number
user name
password
Register


DONATE BLOOD.
Home
Donate
Signup
Login
DONATED TODAY! SAVE A LIFE
GIVE BLOOD
EVERYONE COULD BE A HERO
LOGIN
Username
Enter username
Password
Enter password
Remember me
Login
Cancel
Not a member? Sign Up
Forgot Password?
dd / mm / yyyy
Date Of Birth
Female
Male
Other
Email
Phone Number
user name
password
Register


admin
Admin Dashboard
Home
Hospitals
Messages
Donors
Users
complaints
Notifications
admin
WELCOME BLOOD BANK ADMIN MODULE


ns
Hospital Dashboard
Home
Donors
complaints
Notifications
ns
WELCOME BLOOD BANK HOSPITAL MODULE
Globalized Bloodbank

hari
Donor Dashboard
Home
New Request
Search
complaints
Notifications
hari
WELCOME BLOOD BANK HEALTH MODULE
Globalized Bloodbank


yazeen123
User Dashboard
Home
Search
complaints
Notifications
yazeen123
WELCOME BLOOD BANK HEALTH MODULE
Globalized Bloodbank

APPENDIX 2

MEETING MINUTES

Group Members

MINUTES

Date : 10/06/2022

Time : 10:00 am

Location : KOLLAM

We decided to do a project named “BLOOD DONATION”, and we chosen Python language with HTML as the front end and MYSQL as the back end. System study and analysis are also done in between 10/06/2022 to 16/06/2022.

MINUTES

Date : 17/06/2022

Time : 10:00 am

Location : KOLLAM

From 17/06/2022 to 06/06/2022 we started the design process, forms and data structures.

MINUTES

Date : 07/07/2022

Time : 10:00 am

Location : KOLLAM

The design process is over then we started coding from 07/07/2022 to 27/07/2022. Along with coding we did the documentation also.

MINUTES

Date : 28/07/2022

Time : 10:00 am

Location : KOLLAM

From 28/07/2022 to 20/08/2021 about 90% of coding is completed, we started testing process. It can take a long time to complete the process

MINUTES

Date : 21/082022

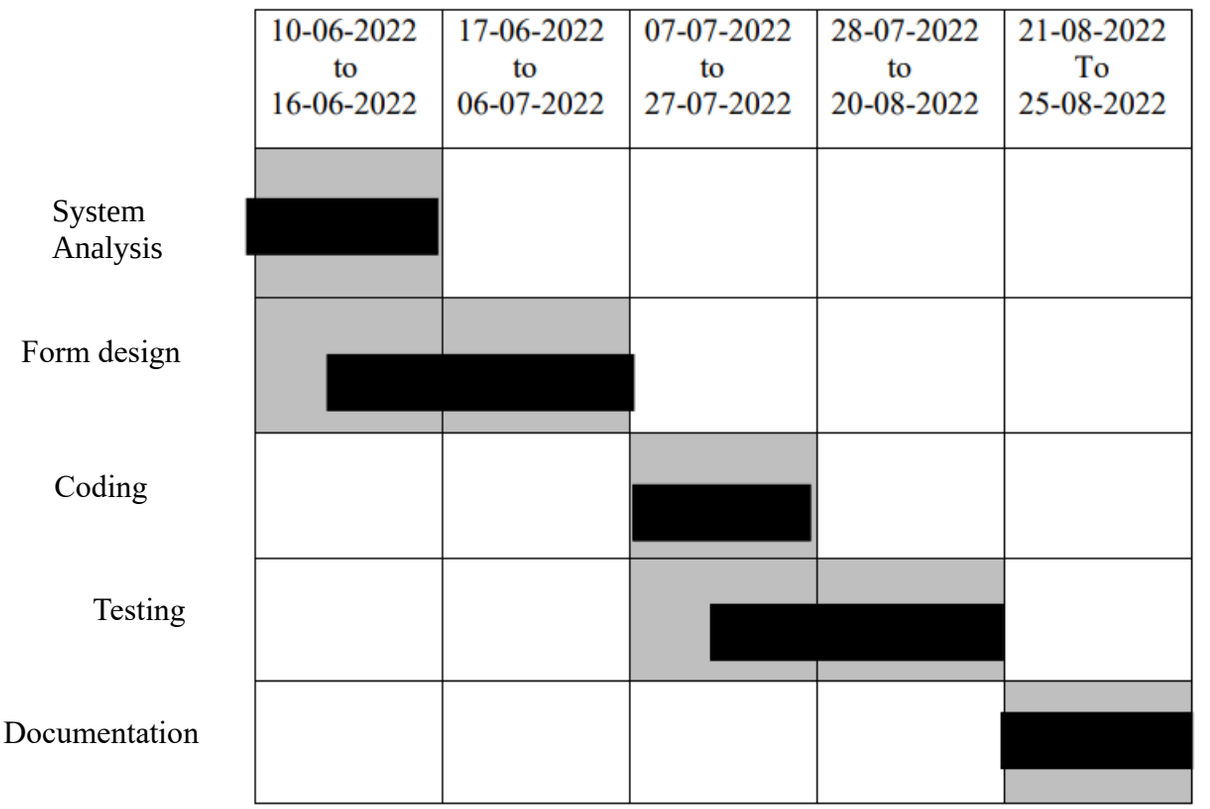
Time : 10:00 am

Location : KOLLAM

We successfully completed our project and the documentation on 25/09/2022.

APPENDIX 3

GANTT CHART



BIBLIOGRAPHY

- Rajib Mall :Fundamentals of software engineering _Prentice, Hall of India private Limited 7 Edition 2000
- Python™: The Complete Reference,. Seventh Edition. Herbert Schildt.
- The Python Handbook,Osborne/McGraw-HillISBN: 0-078-82199-1
- TcX AB, Detron HB & MySQL Finland AB ,MySQL Reference manual,