

# **“BLOOD BANK MANAGEMENT SYSTEM”**

**A Project Report Submitted in Partial Fulfilment of the  
Requirements For the Degree of**

## **BACHELOR OF TECHNOLOGY**

**in**

**Computer Science And Engineering**

**by**

**Deepankar Singh (180010770082)**

**Anurag Dubey (180010770052)**

**Amit Sahu (180010770035)**

**Under the Guidance of  
Er.Priyanaka Tripathi**



**FACULTY OF ENGINEERING AND TECHNOLOGY  
UNIVERSITY OF LUCKNOW, LUCKNOW**

**2021**

## **DECLARATION**

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person or material which to a substantial extent has been accepted for the award of any other degree or diploma of the University or other institute of higher education, except where due acknowledgement has been made in the text.

**Student Name 1: Deepankar Singh**

**Roll No.: 180010770082**

**Date: 05/06/2021**

**Student Name 2: Anurag Dubey**

**Roll No.: 180010770052**

**Date: 05/06/2021**

**Student Name 3: Amit Sahu**

**Roll No.: 180010770035**

**Date: 05/06/2021**

## CERTIFICATE

Certified that **Deepankar Singh** (180010770082) **Anurag Dubey** (180010770052)

**Amit Sahu** (180010770035) have carried out the project work presented in this project report entitled “**Blood Bank Management System**” for the award of **Bachelor of Technology** (Computer Science and Engineering) from **Faculty of Engineering and Technology, University of Lucknow, Lucknow** under my guidance. The project report embodies results of original work, and studies are carried out by the student themselves and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**(Er.Priyanka Tripathi)**  
**(Assistant Professor)**

**(Er.Chandrabhan Singh)**  
**(In-Charge,Deptt. of CSE)**

## **ABSTRACT**

The purpose of Blood Bank Management System is to automate the existing manual system by the help of computerized equipments and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.

Blood Bank Management System, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, while being able to reach the information.

The aim is to automate its existing manual system by the help of computerized equipments and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. Basically the project describes how to manage for good performance and better services for the clients.

## ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our Project Mentor “**Er.Priyanka Tripathi**” for their able guidance and support in partially completing my project.

We would also like to extend our gratitude to project coordinator of our team batch “**Er.Zeeshan Ali Siddiqui** ” for providing us with all facility that was required.

*(Signature)*  
**Deepankar Singh**

*(Signature)*  
**Anurag Dubey**

*(Signature)*  
**Amit Sahu**

# TABLE OF CONTENT

DECLARATION.....	ii
CERTIFICATE .....	iii
ABSTRACT .....	iv
ACKNOWLEDGEMENT.....	v
TABLE OF CONTENT .....	vi
<b>LIST OF TABLES.....</b>	<b>viii</b>
<b>LIST OF FIGURES.....</b>	<b>viii</b>
<b>LIST OF ABBREVIATIONS .....</b>	<b>ix</b>
<b>CHAPTER 1 INTRODUCTION .....</b>	<b>1</b>
PROBLEM INTRODUCTION .....	1
MOTIVATION .....	2
PROJECT OBJECTIVES .....	2
SCOPE OF THE PROJECT.....	3
<b>CHAPTER 2 SOFTWARE AND HARDWARE</b>	
<b>REQUIREMENTS .....</b>	<b>4</b>
INITIATION .....	4
SPECIFICATIONS .....	4
Software Requirements .....	4
Hardware Requirements .....	5
FUNCTIONAL REQUIREMENTS.....	5
Application Functionality .....	5
Security Requirements.....	6
<b>CHAPTER 3 SYSTEM DESIGN .....</b>	<b>10</b>
ARCHITECTURE DIAGRAM.....	10
USECASE DIAGRAM .....	11
Purpose of Use Case Diagrams .....	12
Actors .....	12
Use Cases .....	13
SYSTEM CHART.....	14
ER DIAGRAM.....	16
What is ER Model? .....	16
Why use ER Diagrams?.....	16
ER Diagrams Symbols & Notations.....	17
Components of the ER Diagram.....	18
WHAT IS ENTITY? .....	18

Entity set:.....	18
Relationship.....	19
Weak Entities .....	20
Attributes .....	21
DFDs .....	23
DATABASE SCHEMA .....	26
<b>CHAPTER 4 IMPLEMENTATION AND RESULTS</b>	<b>29</b>
IMPLEMENTATION DETAILS.....	29
FRONT END.....	29
HTML.....	29
CSS.....	30
JAVASCRIPT .....	31
BOOTSTRAP.....	32
JQUERY.....	33
AJAX.....	34
BACKEND.....	35
JAVA.....	35
JSP(Java Server Pages).....	36
MYSQL .....	38
TESTING/RESULT AND ANALYSIS .....	39
Unit Testing.....	40
Introduction .....	40
Benefits.....	40
Integration Testing.....	42
Purpose .....	42
Software Verification And Validation.....	44
Introduction .....	44
Classification of Methods.....	46
Test Cases.....	47
Black-Box Testing.....	47
Test Procedures .....	47
Test Cases.....	48
White-Box Testing .....	48
Levels .....	49
Procedures .....	49
Advantages .....	50
Disadvantages.....	51
System Testing .....	51
DESIGN AND PLANNING .....	52
Software Development Cycle.....	52
Waterfall Model.....	52

COMPLETE SNAPSHOTS .....	54
FUTURE SCOPE .....	66
<b>References .....</b>	<b>67</b>

## LIST OF TABLES

Table 1 Hardware Requirement.....	5
Table 2 Software Requirement .....	6
Table 3 Difference Between Strong and weak entity set .....	20

## LIST OF FIGURES

Figure 1 HTTP Authentication .....	8
Figure 2 Form Base Authentication.....	9
Figure 3 Architecture Diagram .....	11
Figure 4 Actor .....	13
Figure 5 Perform Action .....	13
Figure 6 Use Case Diagram .....	14
Figure 7 System Chart .....	15
Figure 8 ER Diagram Info .....	17
Figure 9 Entity Set .....	19
Figure 10 Relationship .....	19
Figure 11 Weak Entity .....	20
Figure 12 Attribute.....	22
Figure 13 ER Diagram .....	23
Figure 14 Basic Structure of DFD .....	25
Figure 15 Data Flow Diagram .....	26
Figure 16 Database Structure.....	27
Figure 17 Database Schema.....	28
Figure 18 Waterfall Model .....	53
Figure 19 Homepage_1 .....	54
Figure 20 Homepage_2.....	54
Figure 21 Homepage_3.....	55
Figure 22 Register as a Doner .....	55
Figure 23 Register as an Organization/Hospital.....	56
Figure 24 Login Panel.....	56
Figure 25 Doner_Panel .....	57
Figure 26 Update_Password_Doner .....	57
Figure 27 Doner_Profile .....	58
Figure 28 Update_Doner_Profile .....	58



Figure 29 Donation_Report .....	59
Figure 30 Donate_Us .....	59
Figure 31 Find_Blood .....	60
Figure 32 Organization/Hospital_Panel .....	60
Figure 33 Organization/Hospital_Details .....	61
Figure 34 Add_Temp_Doner.....	61
Figure 35 Register_Camp/Counter .....	62
Figure 36 Update_Org/Hosp_Details .....	62
Figure 37 Org/Hosp_Dashboard.....	63
Figure 38 Admin_Panel .....	63
Figure 39 Org_List.....	64
Figure 40 Permanent_Doner_List.....	64
Figure 41 Temp_Doner_List .....	65
Figure 42 Stock_Details.....	65

## LIST OF ABBREVIATIONS

- Org.....Organization
- ER Diagram.....Entity Relationship Diagram
- Temp.....Temporary
- Hosp..... Hospital
- AJAX.....Asynchronous JavaScript and XML
- XML.....Extensible Markup Language

# **CHAPTER 1 INTRODUCTION**

## **PROBLEM INTRODUCTION**

At present, the public can only know about the blood donation events through conventional media means such as radio, news paper or television advertisements.

There is no information regarding the blood donation programs available on any of the portal. The current system that is using by the blood bank is manual system.

With the manual system, there are problems in managing the donors' records. The records of the donor might not be kept safely and there might be missing of donor's records due to human error or disasters.

Besides that, errors might occur when the staff keeps more than one record for the same donor. There is no centralized database of volunteer donors.

So, it becomes really tedious for a person to search blood in case of emergency. The only option is to manually search and match donors and then make phone calls to every donor.

Without an automated management system, there are also problems in keeping track of the actual amount of each and every blood type in the blood bank.

In addition, there is also no alert available when the blood quantity is below its per level or when the blood in the bank has expired.

## **MOTIVATION**

- Scarcity of rare blood group.
- Unavailability of blood during emergency.
- Less awareness among people about blood donation and blood transfusion.
- Deaths due to lack of blood during operations.
- Above points is sufficient to motivate to built a blood management system project to make all procedures automated and to manage all these cumbersome jobs.

## **PROJECT OBJECTIVIES**

- To provide a means for the blood bank to publicize and advertise blood donation programs.
- To provide an efficient donor and blood stock management functions to the blood bank by recording the donor and blood details.
- To provide synchronized and centralized donor and blood stock database.
- To provide immediate storage and retrieval of data and information.
- To improve the efficiency of blood stock management by alerting the blood bank staffs when the blood quantity is below it par level or when the blood stock has expired.

## **SCOPE OF THE PROJECT**

- The system is used for maintaining all the process and activities of blood bank management system.
- The system can be extended to be used for maintaining records of hospital, organ donation and other similar sectors.
- While developing the system, there shall be space for further modification.
- As a whole the system is focused to work with blood bank management system and on additional modification it can be also used as management systems of similar organizations.
- There shall be a proper documentation so that further enhancement becomes easy.

# CHAPTER 2 SOFTWARE AND HARDWARE REQUIREMENTS

## INITIATION

## SPECIFICATIONS

### Software Requirements

Software requirement specification is a kind of document which is created by a software analyst after the requirements collected from the various sources - the requirement received by the customer written in ordinary language. It is the job of the analyst to write the requirement in technical language so that they can be understood and beneficial by the development team.

Number	Description	Type
1	Operating System	Windows
2	Database GUI	MySQL Workbench 6.3
3	Database	MySQL
4	IDE	NetBeans 8.2
5	Browser	Google Chrome/MozillaFirefox

## Hardware Requirements

Hardware	Minimum requirements
Computer	4 GHz minimum, multi-core processor
Memory (RAM)	At least 4GB, preferably higher, and comensurate with concurrent usage
Hard disk space	At least 10 GB

*Table 1 Hardware Requirement*

## FUNCTIONAL REQUIREMENTS

A **Functional Requirement** (FR) is a description of the service that the software must offer. It describes a software system or its component. A function is nothing but inputs to the software system, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. Functional Requirements in Software Engineering are also called **Functional Specification**.

### Application Functionality

In software engineering and systems engineering, a Functional Requirement can range from the high-level abstract statement of the sender's necessity to detailed mathematical functional requirement specifications. Functional software requirements help you to capture the intended behaviour of the system.

Functional requirement Number	Function Requirement Description
FR1	Doner should be able to be login and register.
FR2	Organization/Hospital register,login and organize Camp/Counter.
FR3	Anyone request for blood.
FR4	Blood cell manages by admin.

*Table 2 Software Requirement*

## Security Requirements

Web applications are created by application developers who give, sell, or otherwise transfer the application to an application deployer for installation into a runtime environment. Application developers communicate how the security is to be set up for the deployed application **declaratively** by use of the **deployment descriptor** mechanism. A deployment descriptor enables an application's security structure, including roles, access control, and authentication requirements, to be expressed in a form external to the application.

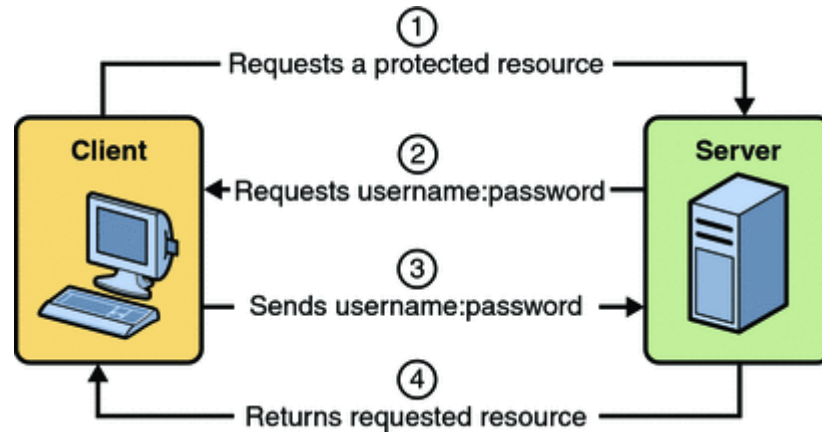
A web application is defined using a standard Java EE web.xml deployment descriptor. A deployment descriptor is an XML schema document that conveys elements and configuration information for web applications. The deployment descriptor must indicate which version of the web application schema (2.4 or 2.5) it is using, and the elements specified within the deployment descriptor must comply with the rules for processing that version of the deployment descriptor

**HTTP Basic Authentication** requires that the server request a user name and password from the web client and verify that the user name and password are valid by comparing them against a database of authorized users. When basic authentication is declared, the following actions occur:

1. A client requests access to a protected resource.
2. The web server returns a dialog box that requests the user name and password.
3. The client submits the user name and password to the server.
4. The server authenticates the user in the specified realm and, if successful, returns the requested resource.

HTTP basic authentication is not a secure authentication mechanism. Basic authentication sends user names and passwords over the Internet as text that is Base64 encoded, and the target server is not authenticated. This form of authentication can expose user names and passwords. If someone can intercept the transmission, the user name and password information can easily be decoded. However, when a secure transport mechanism, such as SSL, or security at the network level, such as the IPSEC protocol or VPN strategies, is used in conjunction with basic authentication, some of these concerns can be alleviated.





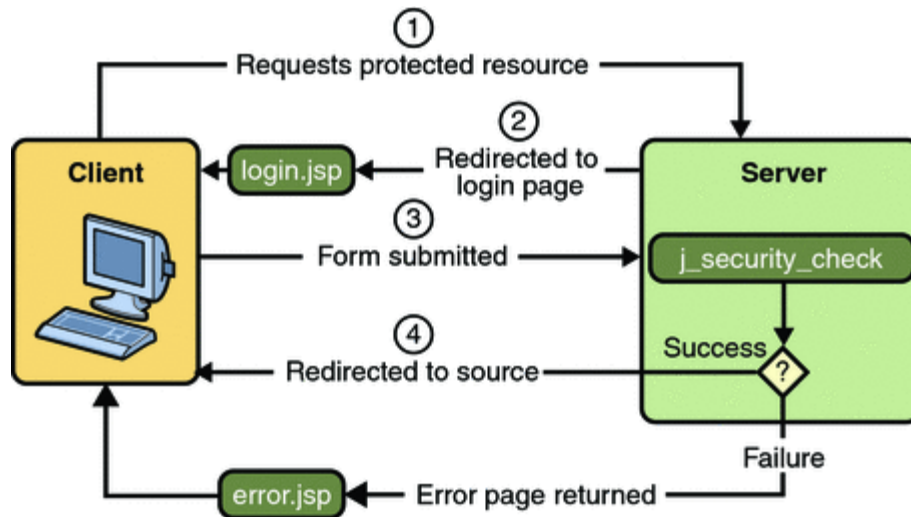
*Figure 1 HTTP Authentication*

**Form-based authentication** allows the developer to control the look and feel of the login authentication screens by customizing the login screen and error pages that an HTTP browser presents to the end user. When form-based authentication is declared, the following actions occur:

1. A client requests access to a protected resource.
2. If the client is unauthenticated, the server redirects the client to a login page.
3. The client submits the login form to the server.
4. The server attempts to authenticate the user.
  - A. If authentication succeeds, the authenticated user's principal is checked to ensure it is in a role that is authorized to access the resource. If the user is authorized, the server redirects the client to the resource using the stored URL path.
  - B. If authentication fails, the client is forwarded or redirected to an error page.

Form-based authentication is not particularly secure. In form-based authentication, the content of the user dialog box is sent as plain text, and the target server is not authenticated. This form of authentication

can expose your user names and passwords unless all connections are over SSL. If someone can intercept the transmission, the user name and password information can easily be decoded. However, when a secure transport mechanism, such as SSL, or security at the network level, such as the IPSEC protocol or VPN strategies, is used in conjunction with form-based authentication, some of these concerns can be alleviated.



*Figure 2 Form Base Authentication*

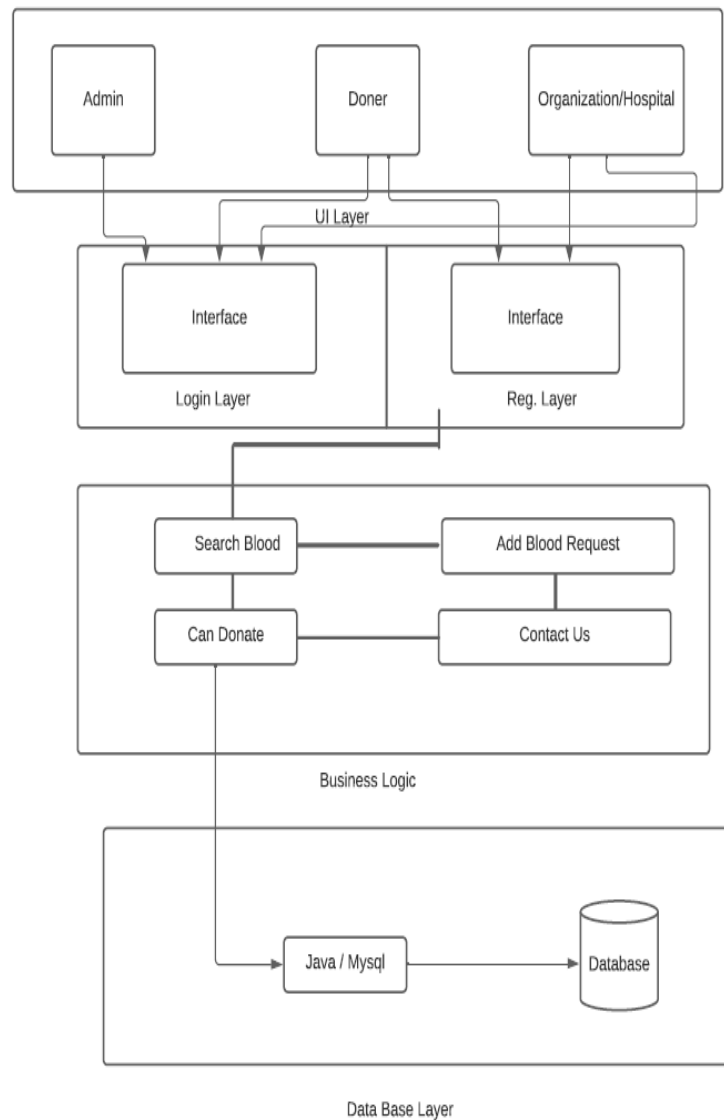
## **CHAPTER 3 SYSTEM DESIGN**

### **ARCHITECTURE DIAGRAM**

An architectural diagram is a diagram of a system that is used to abstract the overall outline of the software system and the relationships, constraints, and boundaries between components. It is an important tool as it provides an overall view of the physical deployment of the software system and its evolution roadmap.

A diagram much like a picture is worth a thousand words. In other words, an architectural diagram must serve several different functions. To allow relevant users to understand a system architecture and follow it in their decision-making, we need to communicate information about the architecture. Architectural diagrams provide a great way to do this. To put down some major functions, an architectural diagram needs to:

- Break down communication barriers
- Reach a consensus
- Decrease ambiguity



*Figure 3 Architecture Diagram*

## USECASE DIAGRAM

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

## Purpose of Use Case Diagrams

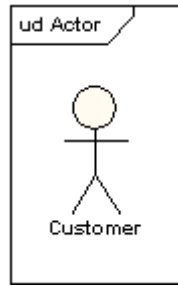
The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

Following are the purposes of a use case diagram given below:

1. It gathers the system's needs.
2. It depicts the external view of the system.
3. It recognizes the internal as well as external factors that influence the system.
4. It represents the interaction between the actors.

### **Actors**

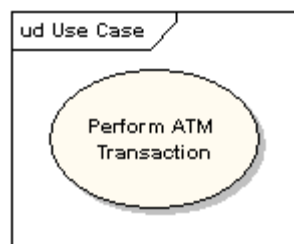
A use case diagram shows the interaction between the system and entities external to the system. These external entities are referred to as actors. Actors represent roles which may include human users, external hardware or other systems. An actor is usually drawn as a named stick figure, or alternatively as a class rectangle with the «actor» keyword.



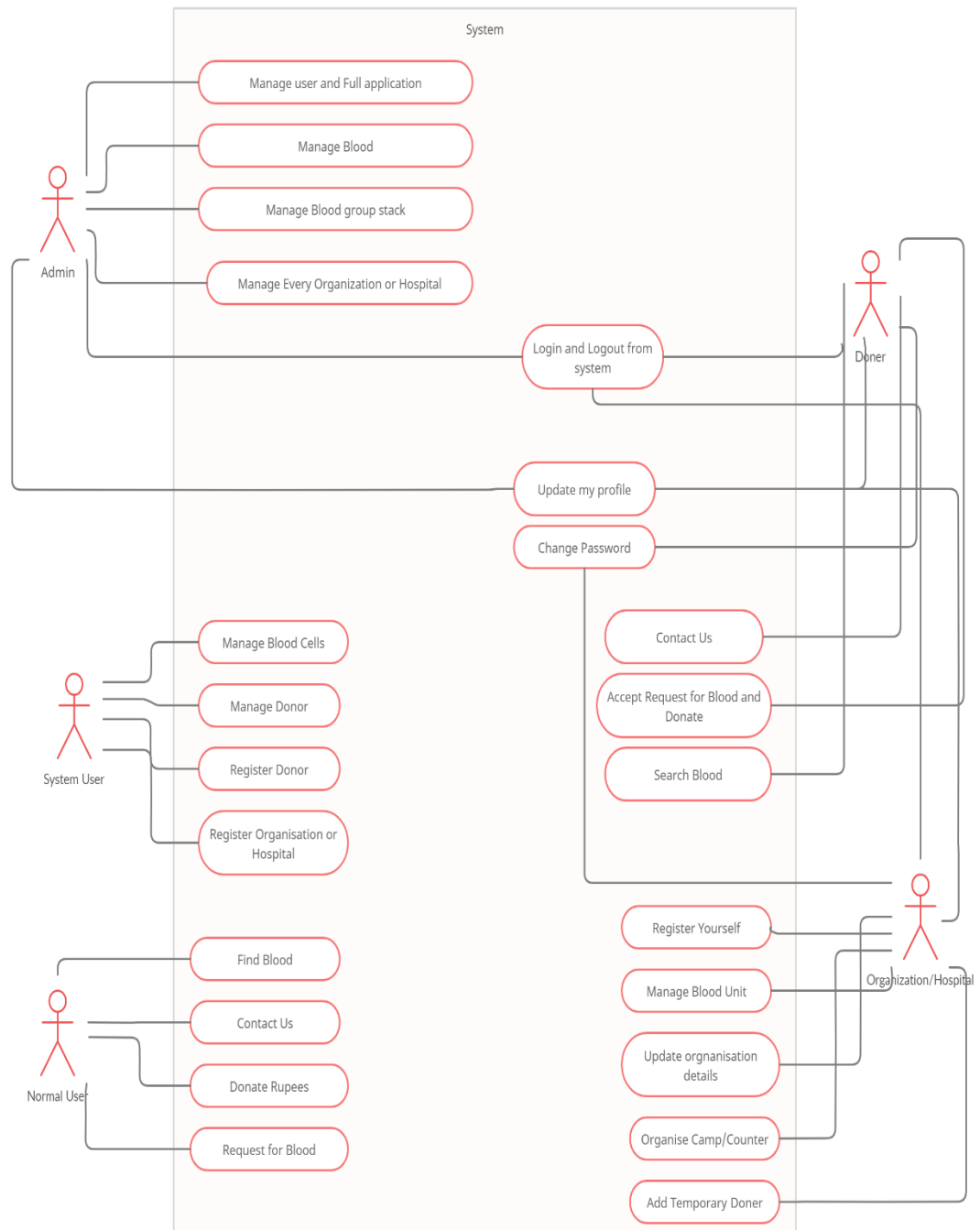
*Figure 4 Actor*

## Use Cases

A use case is a single unit of meaningful work. It provides a high-level view of behavior observable to someone or something outside the system. The notation for a use case is an ellipse.



*Figure 5 Perform Action*



*Figure 6 Use Case Diagram*

## SYSTEM CHART

A **flowchart** is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields

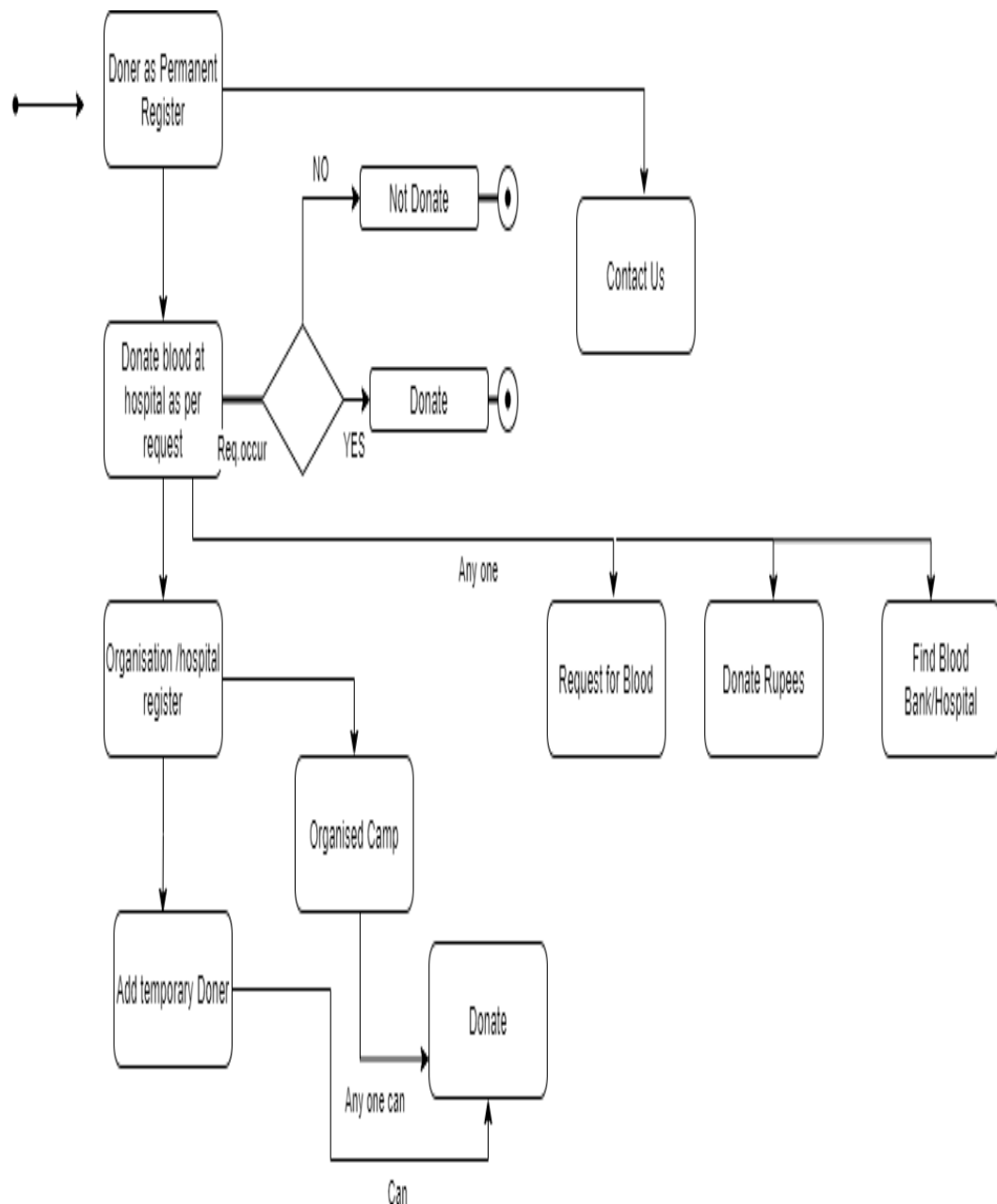


Figure 7 System Chart



## **ER DIAGRAM**

### **What is ER Model?**

**ER Model** stands for Entity Relationship Model is a high-level conceptual data model diagram. ER model helps to systematically analyze data requirements to produce a well-designed database. The ER Model represents real-world entities and the relationships between them. Creating an ER Model in DBMS is considered as a best practice before implementing your database.

ER Modeling helps you to analyze data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modeling before implementing your database.

### **Why use ER Diagrams?**

Here, are prime reasons for using the ER Diagram

- Helps you to define terms related to entity relationship modeling
- Provide a preview of how all your tables should connect, what fields are going to be on each table
- Helps to describe entities, attributes, relationships
- ER diagrams are translatable into relational tables which allows you to build databases quickly
- ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications
- The database designer gains a better understanding of the information to be contained in the database with the help of ERP diagram

- ERD Diagram allows you to communicate with the logical structure of the database to users

## ER Diagrams Symbols & Notations

**Entity Relationship Diagram Symbols & Notations** mainly contains three basic symbols which are rectangle, oval and diamond to represent relationships between elements, entities and attributes. There are some sub-elements which are based on main elements in ERD Diagram. ER Diagram is a visual representation of data that describes how data is related to each other using different ERD Symbols and Notations.

**Following are the main components and its symbols in ER Diagrams:**

- **Rectangles:** This Entity Relationship Diagram symbol represents entity types
- **Ellipses :** Symbol represent attributes
- **Diamonds:** This symbol represents relationship types
- **Lines:** It links attributes to entity types and entity types with other relationship types
- **Primary key:** attributes are underlined
- **Double Ellipses:** Represent multi-valued attributes



*Figure 8 ER Diagram Info*

## Components of the ER Diagram

This model is based on three basic concepts:

- Entities
- Attributes
- Relationships

### WHAT IS ENTITY?

A real-world thing either living or non-living that is easily recognizable and nonrecognizable. It is anything in the enterprise that is to be represented in our database. It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.

An entity can be place, person, object, event or a concept, which stores data in the database. The characteristics of entities are must have an attribute, and a unique key. Every entity is made up of some 'attributes' which represent that entity.

### Examples of entities:

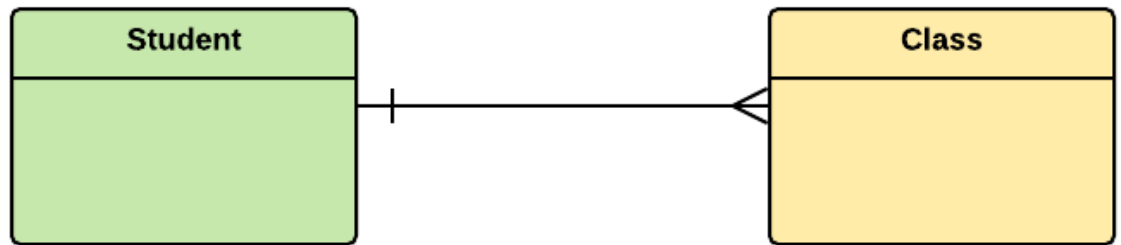
- **Person:** Employee, Student, Patient
- **Place:** Store, Building
- **Object:** Machine, product, and Car
- **Event:** Sale, Registration, Renewal
- **Concept:** Account, Course

### Entity set:

Student

An entity set is a group of similar kind of entities. It may contain entities with attribute sharing similar values. Entities are represented by their

properties, which also called attributes. All attributes have their separate values. For example, a student entity may have a name, age, class, as attributes.



*Figure 9 Entity Set*

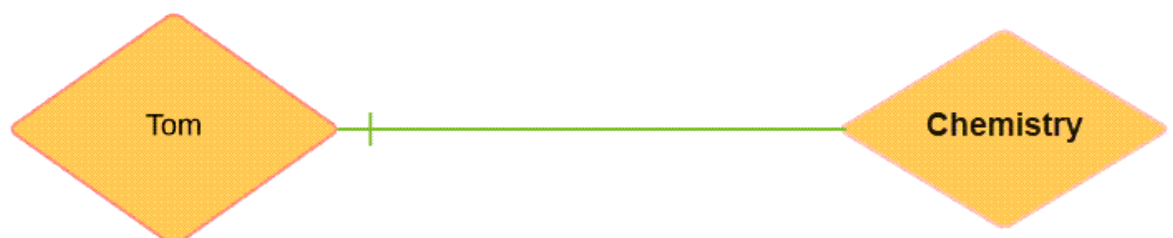
### **Example of Entities:**

A university may have some departments. All these departments employ various lecturers and offer several programs.

Some courses make up each program. Students register in a particular program and enroll in various courses. A lecturer from the specific department takes each course, and each lecturer teaches a various group of students.

### **Relationship**

Relationship is nothing but an association among two or more entities. E.g., Tom works in the Chemistry department.



*Figure 10 Relationship*

Fig

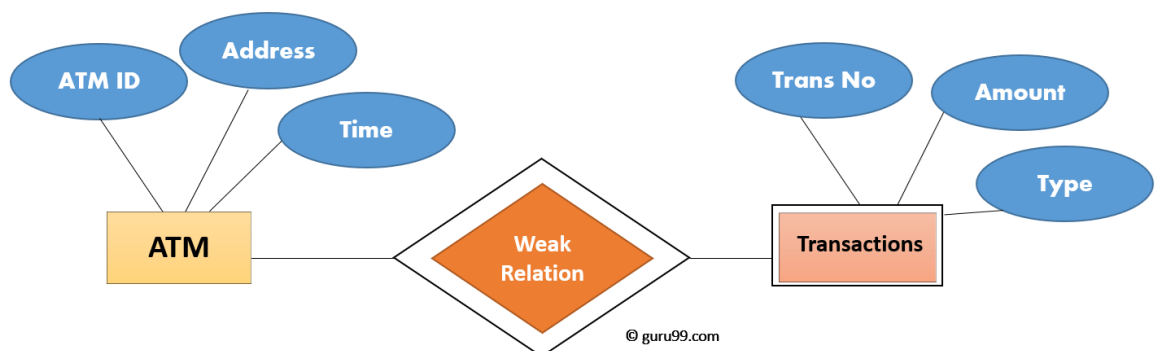
Entities take part in relationships. We can often identify relationships with verbs or verb phrases.

**For example:**

- You are attending this lecture
- I am giving the lecture
- Just like entities, we can classify relationships according to relationship-types:
- A student attends a lecture
- A lecturer is giving a lecture.

**Weak Entities**

A weak entity is a type of entity which doesn't have its key attribute. It can be identified uniquely by considering the primary key of another entity. For that, weak entity sets need to have participation.



*Figure 11 Weak Entity*

In above ER Diagram examples, "Trans No" is a discriminator within a group of transactions in an ATM.

*Table 3 Difference Between Strong and weak entity set*

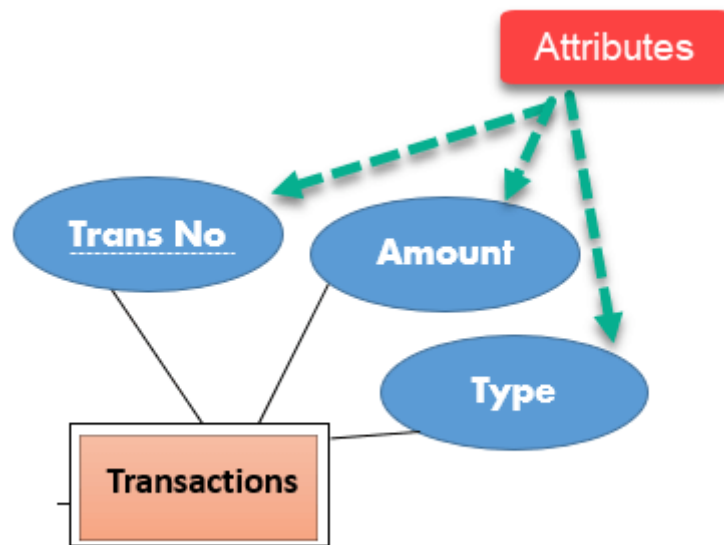
<b>Strong Entity Set</b>	<b>Weak Entity Set</b>
Strong entity set always has a primary key.	It does not have enough attributes to build a primary key.
It is represented by a rectangle symbol.	It is represented by a double rectangle symbol.
It contains a Primary key represented by the underline symbol.	It contains a Partial Key which is represented by a dashed underline symbol.
The member of a strong entity set is called as dominant entity set.	The member of a weak entity set called as a subordinate entity set.
Primary Key is one of its attributes which helps to identify its member.	In a weak entity set, it is a combination of primary key and partial key of the strong entity set.
In the ER diagram the relationship between two strong entity set shown by using a diamond symbol.	The relationship between one strong and a weak entity set shown by using the double diamond symbol.
The connecting line of the strong entity set with the relationship is single.	The line connecting the weak entity set for identifying relationship is double.

### **Attributes**

It is a single-valued property of either an entity-type or a relationship-type.

For example, a lecture might have attributes: time, date, duration, place, etc.

An attribute in ER Diagram examples, is represented by an Ellipse



*Figure 12 Attribute*

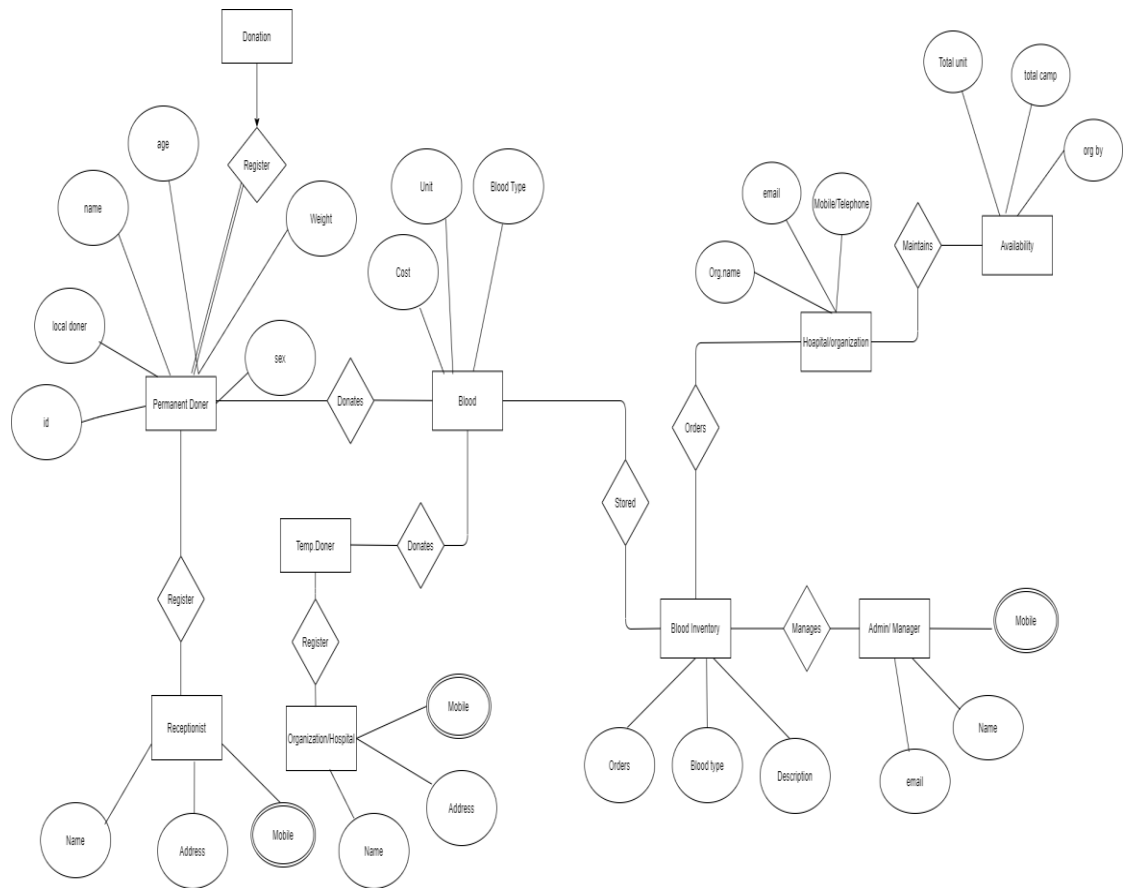


Figure 13 ER Diagram

## DFDs

**DFD** is the abbreviation for **Data Flow Diagram**. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modeling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.



## Components of DFD

The Data Flow Diagram has 4 components:

- **Process**

Input to output transformation in a system takes place because of process function. The symbols of a process are rectangular with rounded corners, oval, rectangle or a circle. The process is named a short sentence, in one word or a phrase to express its essence

- **DataFlow**

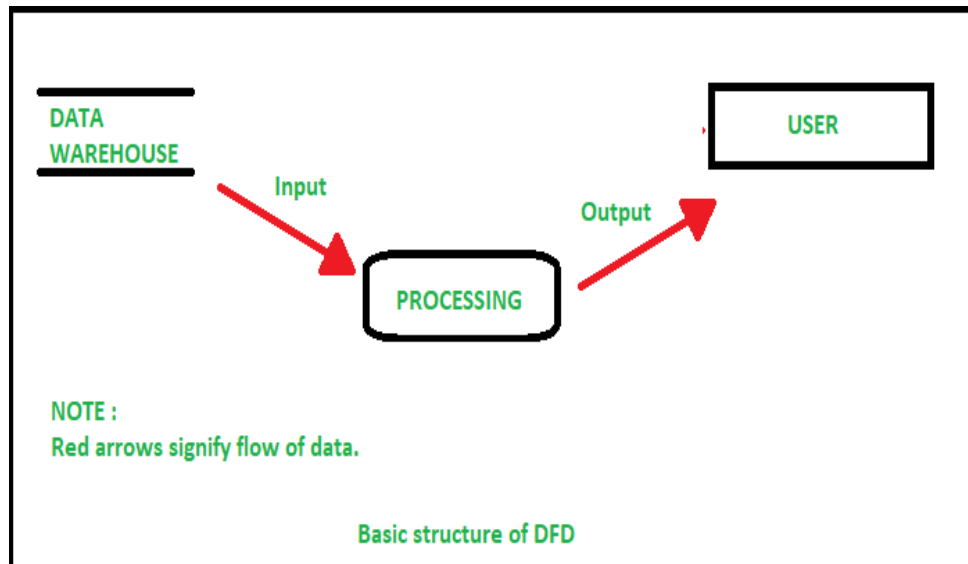
Data flow describes the information transferring between different parts of the systems. The arrow symbol is the symbol of data flow. A relatable name should be given to the flow to determine the information which is being moved. Data flow also represents material along with information that is being moved. Material shifts are modeled in systems that are not merely informative. A given flow should only transfer a single type of information. The direction of flow is represented by the arrow which can also be bi-directional.

- **Warehouse**

The data is stored in the warehouse for later use. Two horizontal lines represent the symbol of the store. The warehouse is simply not restricted to being a data file rather it can be anything like a folder with documents, an optical disc, a filing cabinet. The data warehouse can be viewed independent of its implementation. When the data flow from the warehouse it is considered as data reading and when data flows to the warehouse it is called data entry or data updation.

- **Terminator**

The Terminator is an external entity that stands outside of the system and communicates with the system. It can be, for example, organizations like banks, groups of people like customers or different departments of the same organization, which is not a part of the model system and is an external entity. Modeled systems also communicate with terminator.

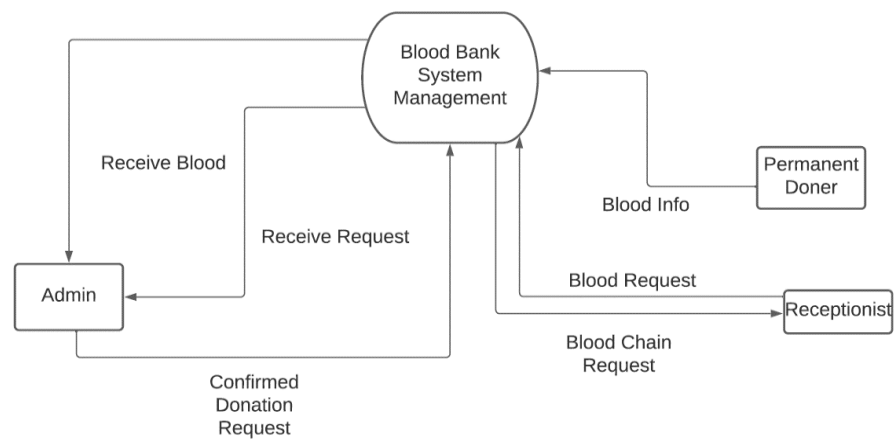


*Figure 14 Basic Structure of DFD*

### Levels of DFD

DFD uses hierarchy to maintain transparency thus multilevel DFD's can be created. Levels of DFD are as follows:

- 0-level DFD
- 1-level DFD:
- 2-level DFD:



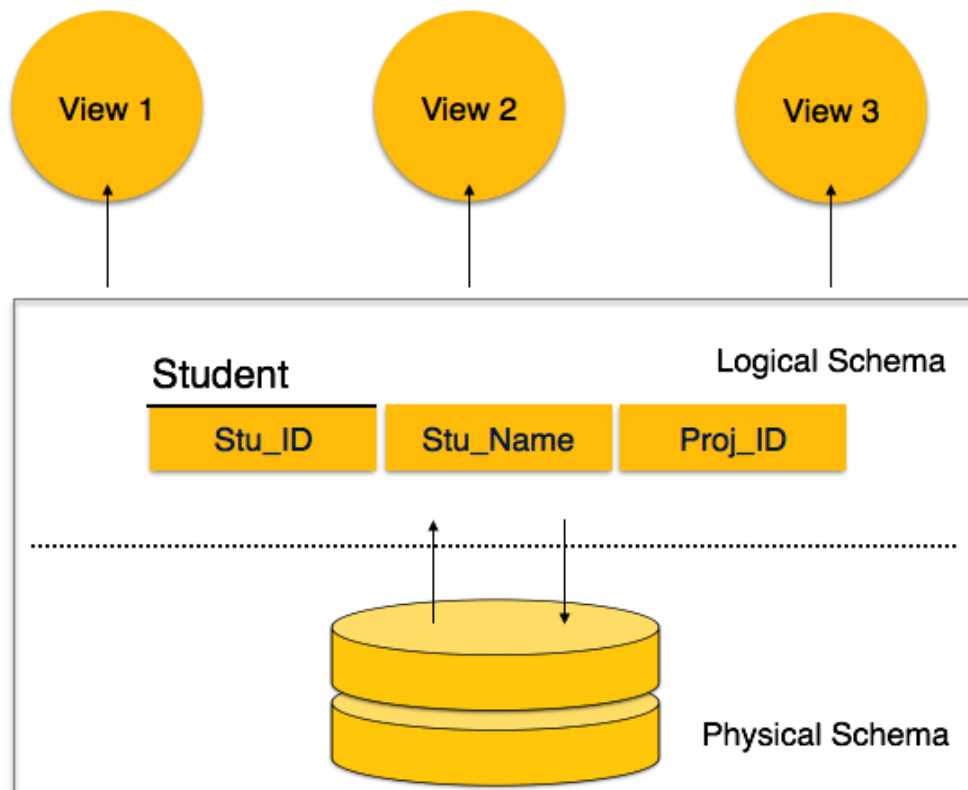
*Figure 15 Data Flow Diagram*

## DATABASE SCHEMA

### Database Schema

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.



*Figure 16 Database Structure*

A database schema can be divided broadly into two categories –

- **Physical Database Schema** – This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
- **Logical Database Schema** – This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

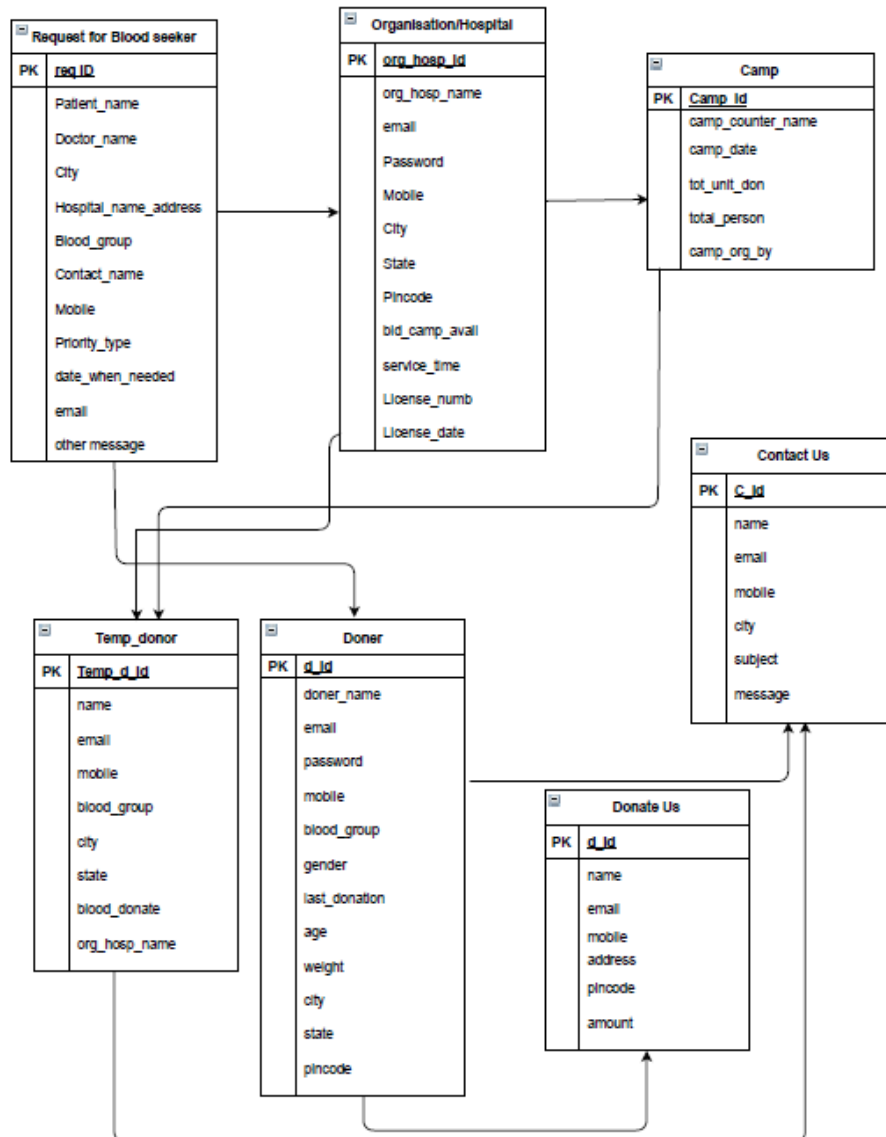


Figure 17 Database Schema

# **CHAPTER 4 IMPLEMENTATION AND RESULTS**

## **IMPLEMENTATION DETAILS**

### **FRONT END**

#### **HTML**

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as `<img />` and `<input />` directly introduce content into the page. Other tags such as `<p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

## **CSS**

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

CSS information can be provided from various sources. These sources can be the web browser, the user and the author. The information from the author can be further classified into inline, media type, importance, selector specificity, rule order, inheritance and property definition. CSS style information can be in a separate document or it can be embedded into an HTML document. Multiple style sheets can be imported. Different styles can be applied depending on the output device being used; for example, the screen version can be quite different from the printed version, so that authors can tailor the presentation appropriately for each medium. The style sheet with the highest priority controls the content display. Declarations not set in

the highest priority source are passed on to a source of lower priority, such as the user agent style. The process is called cascading.

One of the goals of CSS is to allow users greater control over presentation. Someone who finds red italic headings difficult to read may apply a different style sheet. Depending on the browser and the web site, a user may choose from various style sheets provided by the designers, or may remove all added styles and view the site using the browser's default styling, or may override just the red italic heading style without altering other attributes.

## **JAVASCRIPT**

JavaScript is a high-level, interpreted scripting language that conforms to the ECMAScript specification. JavaScript has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions. Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it, and major web browsers have a dedicated JavaScript engine to execute it. As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has APIs for working with text, arrays, dates, regular expressions, and the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities. It relies upon the host environment in which it is embedded to provide these features.

Initially only implemented client-side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases, and in non-web



programs such as word processors and PDF software, and in runtime environments that make JavaScript available for writing mobile and desktop applications, including desktop widgets.

The terms Vanilla JavaScript and Vanilla JS refer to JavaScript not extended by any frameworks or additional libraries. Scripts written in Vanilla JS are plain JavaScript code. Google's Chrome extensions, Opera's extensions, Apple's Safari 5 extensions, Apple's Dashboard Widgets, Microsoft's Gadgets, Yahoo! Widgets, Google Desktop Gadgets, and Serence Klipfolio are implemented using JavaScript.

## **BOOTSTRAP**

Bootstrap is a free and open-source tool collection for creating responsive websites and web applications. It is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites. It solves many problems which we had once, one of which is the cross-browser compatibility issue. Nowadays, the websites are perfect for all the browsers (IE, Firefox, and Chrome) and for all sizes of screens (Desktop, Tablets, Phablets, and Phones). All thanks to Bootstrap developers -Mark Otto and Jacob Thornton of Twitter, though it was later declared to be an open-source project.

### **Why Bootstrap?**

- Faster and Easier Web Development.
- It creates Platform-independent web pages.
- It creates Responsive Web-pages.
- It designed to be responsive to mobile devices too.
- It is Free! Available on [www.getbootstrap.com](http://www.getbootstrap.com)

## JQUERY

**jQuery** is an open source JavaScript library that simplifies the interactions between an HTML/CSS document, or more precisely the Document Object Model (DOM), and JavaScript.

Elaborating the terms, jQuery simplifies HTML document traversing and manipulation, browser event handling, DOM animations, Ajax interactions, and cross-browser JavaScript development.

**Note:** The only library available today that meets the needs of both designer types and programmer types is jQuery.

jQuery is widely famous with its philosophy of “**Write less, do more.**” This philosophy can be further elaborated as three concepts:

- Finding some elements (via CSS selectors) and doing something with them (via jQuery methods) i.e. locate a set of elements in the DOM, and then do something with that set of elements.
- Chaining multiple jQuery methods on a set of elements
- Using the jQuery wrapper and implicit iteration

### **Using jQuery (JS) library on HTML page**

There are several ways to start using jQuery on your web site.

1. Use the Google-hosted/ Microsoft-hosted content delivery network (CDN) to include a version of jQuery.
2. Download own version of jQuery from [jquery.com](http://jquery.com) and host it on own server or local filesystem.

**Note:** All jQuery methods are inside a document-ready event to prevent any jQuery code from running before the document is finished loading (is ready).

## AJAX

Ajax is an acronym for Asynchronous Javascript and XML. It is used to communicate with the server without refreshing the web page and thus increasing the user experience and better performance.

There are no such pre-requisites required to understand the latter portion of the article. Only the basic knowledge of HTML, CSS, and Javascript are good to go.

### How does it work?

First, let us understand what does asynchronous actually mean. There are two types of requests synchronous as well as asynchronous. Synchronous requests are the one which follows sequentially i.e if one process is going on and in the same time another process wants to be executed, it will not be allowed that means the only one process at a time will be executed. This is not good because in this type most of the time CPU remains idle such as during I/O operation in the process which are the order of magnitude slower than the CPU processing the instructions. Thus to make the full utilization of the CPU and other resources use asynchronous calls. For more information visit this [link](#). Why the word javascript is present here.

Actually, the requests are made through the use of javascript functions. Now the term XML which is used to create **XMLHttpRequest object**.

Thus the summary of the above explanation is that Ajax allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. Now discuss the important part and its implementation. For implementing Ajax, only be aware of XMLHttpRequest object. Now, what actually it is. It is an object used to exchange data with the server behind the scenes. Try to remember the paradigm of OOP which says that object communicates through calling methods (or in general sense message passing). The same case applied here as well. Usually, create this object and use it to call the methods which result in effective communication. All modern browsers support the

XMLHttpRequest object.

## **BACKEND**

### **JAVA**

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]). The latest release of the Java Standard Edition is Java SE 8. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. For example: J2EE for Enterprise Applications, J2ME for Mobile Applications. The new J2 versions were renamed as Java SE, Java EE, and Java ME respectively. Java is guaranteed to be Write Once, Run Anywhere. Java is:

- **Object Oriented:** In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- **Platform Independent:** Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
- **Simple:** Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
- **Secure:** With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architecture-neutral:** Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.

- **Portable:** Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
- **Robust:** Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
- **Multithreaded:** With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
- **Interpreted:** Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
- **High Performance:** With the use of Just-In-Time compilers, Java enables high performance.
- **Distributed:** Java is designed for the distributed environment of the internet.
- **Dynamic:** Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

## **JSP(Java Server Pages)**

**JSP** technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL, etc.

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

- It stands for **Java Server Pages**.
- It is a server side technology.
- It is used for creating web application.
- It is used to create dynamic web content.
- In this JSP tags are used to insert JAVA code into HTML pages.
- It is an advanced version of Servlet Technology.
- It is a Web based technology helps us to create dynamic and platform independent web pages.
- In this, Java code can be inserted in HTML/ XML pages or both.
- JSP is first converted into servlet by JSP container before processing the client's request.

### **JSP pages are more advantageous than Servlet:**

- They are easy to maintain.
- No recompilation or redeployment is required.
- JSP has access to entire API of JAVA .
- JSP are extended version of Servlet.
- 

### **Features of JSP**

- **Coding in JSP is easy :-** As it is just adding JAVA code to HTML/XML.
- **Reduction in the length of Code :-** In JSP we use action tags, custom tags etc.

- **Connection to Database is easier** :-It is easier to connect website to database and allows to read or write data easily to the database.
- **Make Interactive websites** :- In this we can create dynamic web pages which helps user to interact in real time environment.
- **Portable, Powerful, flexible and easy to maintain** :- as these are browser and server independent.
- **No Redeployment and No Re-Compilation** :- It is dynamic, secure and platform independent so no need to re-compilation.
- **Extension to Servlet** :- as it has all features of servlets, implicit objects and custom tags

## MYSQL

MySQL is an open source relational database management system (RDBMS) based on Structured Query Language (SQL). It is one part of the very popular LAMP platform consisting of Linux, Apache, My SQL, and PHP. Currently My SQL is owned by Oracle. My SQL database is available on most important OS platforms. It runs on BSD Unix, Linux, Windows, or Mac OS. Wikipedia and YouTube use My SQL. These sites manage millions of queries each day. My SQL comes in two versions: My SQL server system and My SQL embedded system.

## RDBMS TERMINOLOGY

Before we proceed to explain MySQL database system, let's revise few definitions related to database.

- **Database:** A database is a collection of tables, with related data.
- **Table:** A table is a matrix with data. A table in a database looks like a simple spreadsheet.

- **Column:** One column (data element) contains data of one and the same kind, for example the column postcode.
- **Row:** A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.
- **Redundancy:** Storing data twice, redundantly to make the system faster.
- **Primary Key:** A primary key is unique. A key value cannot occur twice in one table. With a key, you can find at most one row.
- **Foreign Key:** A foreign key is the linking pin between two tables.
- **Compound Key:** A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.
- **Index:** An index in a database resembles an index at the back of a book.
- **Referential Integrity:** Referential Integrity makes sure that a foreign key value always points to an existing row.

## TESTING/RESULT AND ANALYSIS

The term implementation has different meanings ranging from the conversion of a basic application to a complete replacement of a computer system. The procedures however, are virtually the same. Implementation includes all those activities that take place to convert from old system to new. The new system may be totally new replacing an existing manual or automated system or it may be major modification to an existing system. The method of implementation and time scale to be adopted is found out initially. Proper implementation is essential to provide a reliable system to meet organization requirement.



## **Unit Testing**

### **Introduction**

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

### **Benefits**

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits.

**1) Find problems early :** Unit testing finds problems early in the development cycle. In test-driven development (TDD), which is frequently used in both extreme programming and scrum, unit tests are

created before the code itself is written. When the tests pass, that code is considered complete. The same unit tests are run against that function frequently as the larger code base is developed either as the code is changed or via an automated process with the build. If the unit tests fail, it is considered to be a bug either in the changed code or the tests themselves. The unit tests then allow the location of the fault or failure to be easily traced. Since the unit tests alert the development team of the problem before handing the code off to testers or clients, it is still early in the development process.

**2 ) Facilitates Change :** Unit testing allows the programmer to refactor code or upgrade system libraries at a later date, and make sure the module still works correctly (e.g., in regression testing). The procedure is to write test cases for all functions and methods so that whenever a change causes a fault, it can be quickly identified. Unit tests detect changes which may break a design contract.

**3 ) Simplifies Integration :** Unit testing may reduce uncertainty in the units themselves and can be used in a bottom-up testing style approach. By testing the parts of a program first and then testing the sum of its parts, integration testing becomes much easier.

**4 ) Documentation :** Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit, and how to use it, can look at the unit tests to gain a basic understanding of the unit's interface (API). Unit test cases embody characteristics that are critical to the success of the unit. These characteristics can indicate appropriate/inappropriate use of a unit as well as negative behaviors that are to be trapped by the unit. A unit test case, in and of itself, documents these critical characteristics, although many

software development environments do not rely solely upon code to document the product in development.

## **Integration Testing**

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

## **Purpose**

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e., assemblages (or groups of units), are exercised through their interfaces using black-box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e., unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages. Software integration testing is performed according to the software development

life cycle (SDLC) after module and functional tests. The cross-dependencies for software integration testing are: schedule for integration testing, strategy and selection of the tools used for integration, define the cyclomathical complexity of the software and software architecture, reusability of modules and life-cycle and versioning management. Some different types of integration testing are big-bang, top-down, and bottom-up, mixed (sandwich) and risky-hardest. Other Integration Patterns[2] are: collaboration integration, backbone integration, layer integration, client-server integration, distributed services integration and high-frequency integration.

### **Big Bang**

In the big-bang approach, most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. This method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing. A type of big-bang integration testing is called "usage model testing" which can be used in both software and hardware integration testing. The basis behind this type of integration testing is to run user-like workloads in integrated user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use. Usage Model testing takes an optimistic approach to testing, because it expects to have few problems with the individual components. The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the

interaction of the components in the environment. For integration testing, Usage Model testing can be more efficient and provides better test coverage than traditional focused functional integration testing. To be more efficient and accurate, care must be used in defining the user-like workloads for creating realistic scenarios in exercising the environment. This gives confidence that the integrated environment will work as expected for the target customers.

## **Top Down And Bottom Up**

Bottom-up testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage. Top-down testing is an approach to integrated testing where the top integrated modules are tested and the branch of the module is tested step by step until the end of the related module. Sandwich testing is an approach to combine top down testing with bottom up testing.

## **Software Verification And Validation**

### **Introduction**

In software project management, software testing, and software engineering, verification and validation (V&V) is the process of checking that a software system meets specifications and that it fulfills its intended purpose. It may also be referred to as software quality control. It is normally the responsibility of software testers as part of the software development lifecycle. Validation checks that the product design satisfies or fits the intended use (high-level checking), i.e., the software meets the user requirements. This is done through dynamic testing and other forms of review. Verification and validation are not the same thing, although they are often confused. Boehm succinctly expressed the difference between

- Validation : Are we building the right product?
- Verification : Are we building the product right?

According to the Capability Maturity Model (CMMI-SW v1.1)

**Software Verification:** The process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

**Software Validation:** The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.

In other words, software verification is ensuring that the product has been built according to the requirements and design specifications, while software validation ensures that the product meets the user's needs, and that the specifications were correct in the first place. Software verification ensures that "you built it right". Software validation ensures

that "you built the right thing". Software validation confirms that the product, as provided, will fulfill its intended use.

### From Testing Perspective

- Fault – wrong or missing function in the code.
- Failure – the manifestation of a fault during execution.
- Malfunction – according to its specification the system does not meet its specified functionality

Both verification and validation are related to the concepts of quality and of software quality assurance. By themselves, verification and validation do not guarantee software quality; planning, traceability, configuration management and other aspects of software engineering are required. Within the modeling and simulation (M&S) community, the definitions of verification, validation and accreditation are similar:

- M&S Verification is the process of determining that a • computer model, simulation, or federation of models and simulations implementations and their associated data accurately represent the developer's conceptual description and specifications.
- M&S Validation is the process of determining the degree to which a model, simulation, or federation of models and simulations, and their associated data are accurate representations of the real world from the perspective of the intended use(s).

### **Classification of Methods**

In mission-critical software systems, where flawless performance is absolutely necessary, formal methods may be used to ensure the correct

operation of a system. However, often for non-mission-critical software systems, formal methods prove to be very costly and an alternative method of software V&V must be sought out. In such cases, syntactic methods are often used.

## **Test Cases**

A test case is a tool used in the process. Test cases may be prepared for software verification and software validation to determine if the product was built according to the requirements of the user. Other methods, such as reviews, may be used early in the life cycle to provide for software validation.

## **Black-Box Testing**

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing: unit, integration, system and acceptance. It typically comprises most if not all higher level testing, but can also dominate unit testing as well.

## **Test Procedures**

Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is aware of what the software is supposed to do but is not aware of how it does it. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of how the software produces the output in the first place.



## **Test Cases**

Test cases are built around specifications and requirements, i.e., what the application is supposed to do. Test cases are generally derived from external descriptions of the software, including specifications, requirements and design parameters. Although the tests used are primarily functional in nature, non-functional tests may also be used. The test designer selects both valid and invalid inputs and determines the correct output, often with the help of an oracle or a previous result that is known to be good, without any knowledge of the test object's internal structure.

## **White-Box Testing**

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings 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 used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT). White-box testing can be applied at the unit, integration and system levels of the software testing process. Although traditional testers tended to think of white-box testing as being done at the unit level, it is used for integration and system testing more frequently today. It can test paths within a unit, paths between units during integration, and between subsystems during a system-level test. Though this method of test design can uncover many errors or problems,

it has the potential to miss unimplemented parts of the specification or missing requirements.

## **Levels**

**1 ) Unit testing :** White-box testing is done during unit testing to ensure that the code is working as intended, before any integration happens with previously tested code. White-box testing during unit testing catches any defects early on and aids in any defects that happen later on after the code is integrated with the rest of the application and therefore prevents any type of errors later on.

**2 ) Integration testing :** White-box testing at this level are written to test the interactions of each interface with each other. The Unit level testing made sure that each code was tested and working accordingly in an isolated environment and integration examines the correctness of the behaviour in an open environment through the use of white-box testing for any interactions of interfaces that are known to the programmer.

**3 ) Regression testing :** White-box testing during regression testing is the use of recycled white-box test cases at the unit and integration testing levels.

## **Procedures**

White-box testing's basic procedures involves the tester having a deep level of understanding of the source code being tested. The programmer must have a deep understanding of the application to know what kinds of test cases to create so that every visible path is exercised for testing. Once the source code is understood then the source code can be analyzed for

test cases to be created. These are the three basic steps that white-box testing takes in order to create test cases:

- Input involves different types of requirements, functional specifications, detailed designing of documents, proper source code, security specifications. This is the preparation stage of white-box testing to layout all of the basic information.
- Processing involves performing risk analysis to guide whole testing process, proper test plan, execute test cases and communicate results. This is the phase of building test cases to make sure they thoroughly test the application the given results are recorded accordingly.
- Output involves preparing final report that encompasses all of the above preparations and results.

### **Advantages**

White-box testing is one of the two biggest testing methodologies used today. It has several major advantages:

- Side effects of having the knowledge of the source code is beneficial to thorough testing.
- Optimization of code by revealing hidden errors and being able to remove these possible defects.
- Gives the programmer introspection because developers carefully describe any new implementation.
- Provides traceability of tests from the source, allowing future changes to the software to be easily captured in changes to the tests.

- White box testing give clear, engineering-based, rules for when to stop testing.

## **Disadvantages**

Although white-box testing has great advantages, it is not perfect and contains some disadvantages:

- White-box testing brings complexity to testing because the tester must have knowledge of the program, including being a programmer. White-box testing requires a programmer with a high level of knowledge due to the complexity of the level of testing that needs to be done.
- On some occasions, it is not realistic to be able to test every single existing condition of the application and some conditions will be untested.
- The tests focus on the software as it exists, and missing functionality may not be discovered.

## **System Testing**

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together

(called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s).

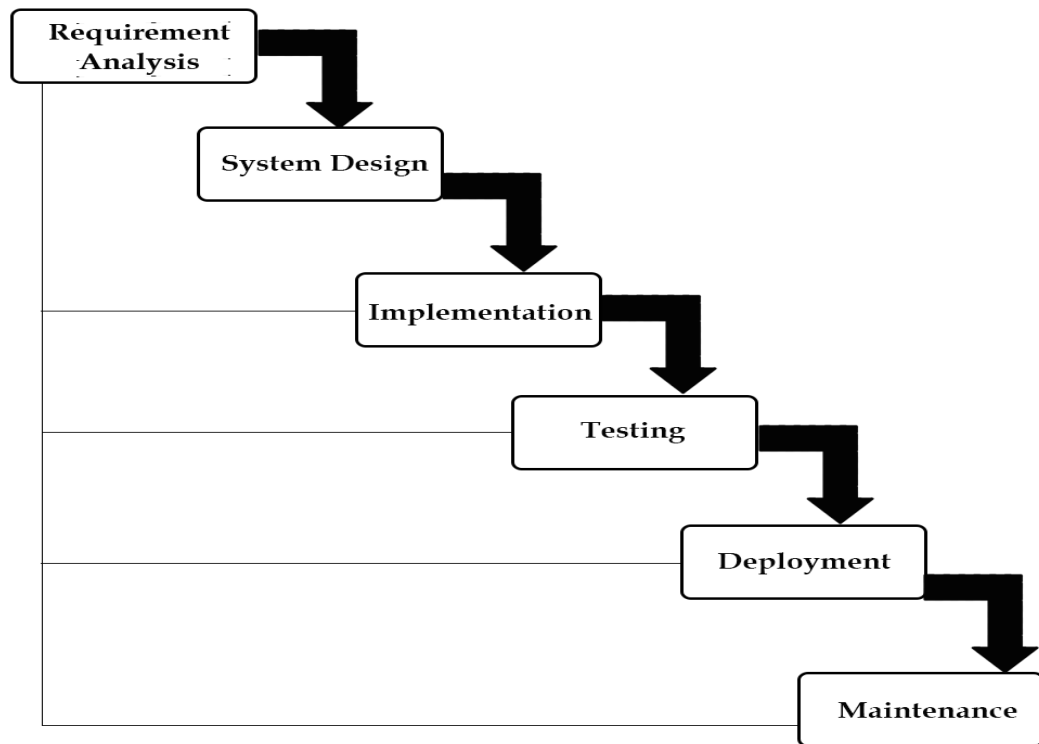
## **DESIGN AND PLANNING**

### **Software Development Cycle**

#### **Waterfall Model**

The waterfall model was selected as the SDLC model due to the following reasons:

- Requirements were very well documented, clear and fixed.
- Technology was adequately understood.
- Simple and easy to understand and use.
- There were no ambiguous requirements.
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Clearly defined stages.
- Well understood milestones. Easy to arrange tasks.



*Figure 18 Waterfall Model*

## COMPLETE SNAPSHOTS

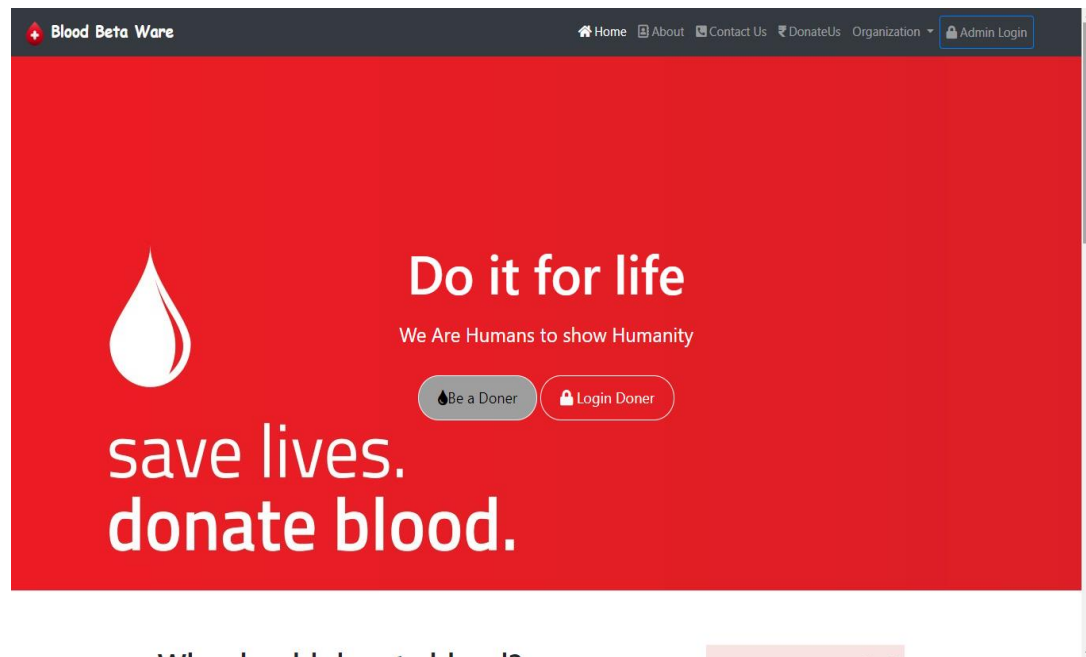


Figure 19 Homepage\_1

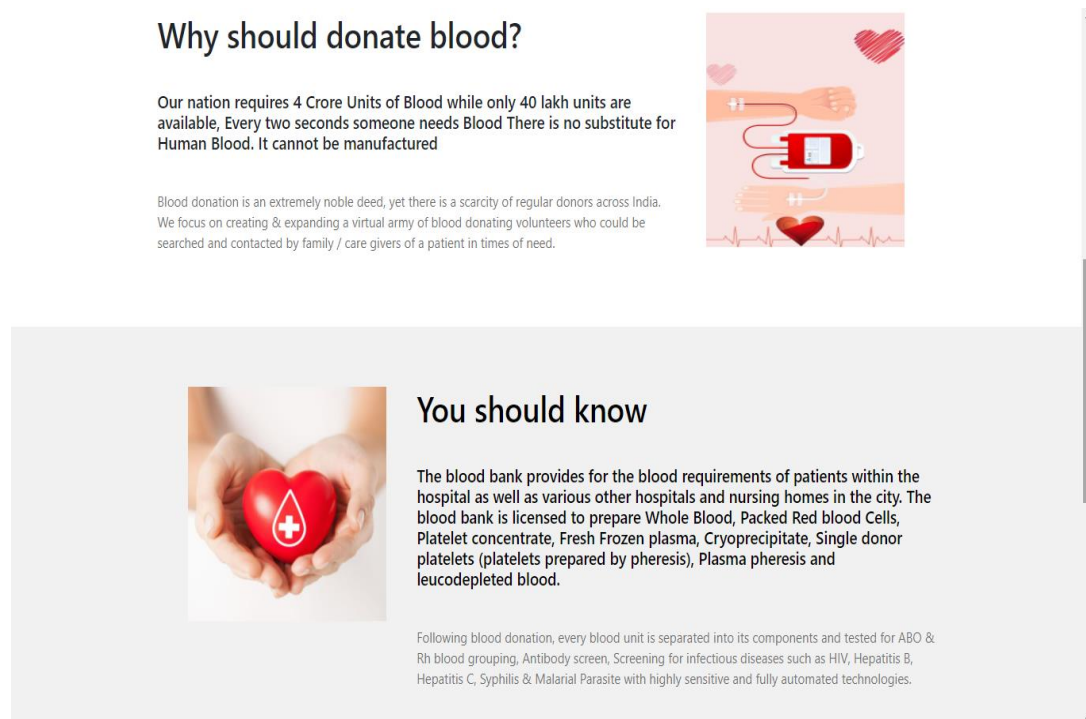


Figure 20 Homepage\_2

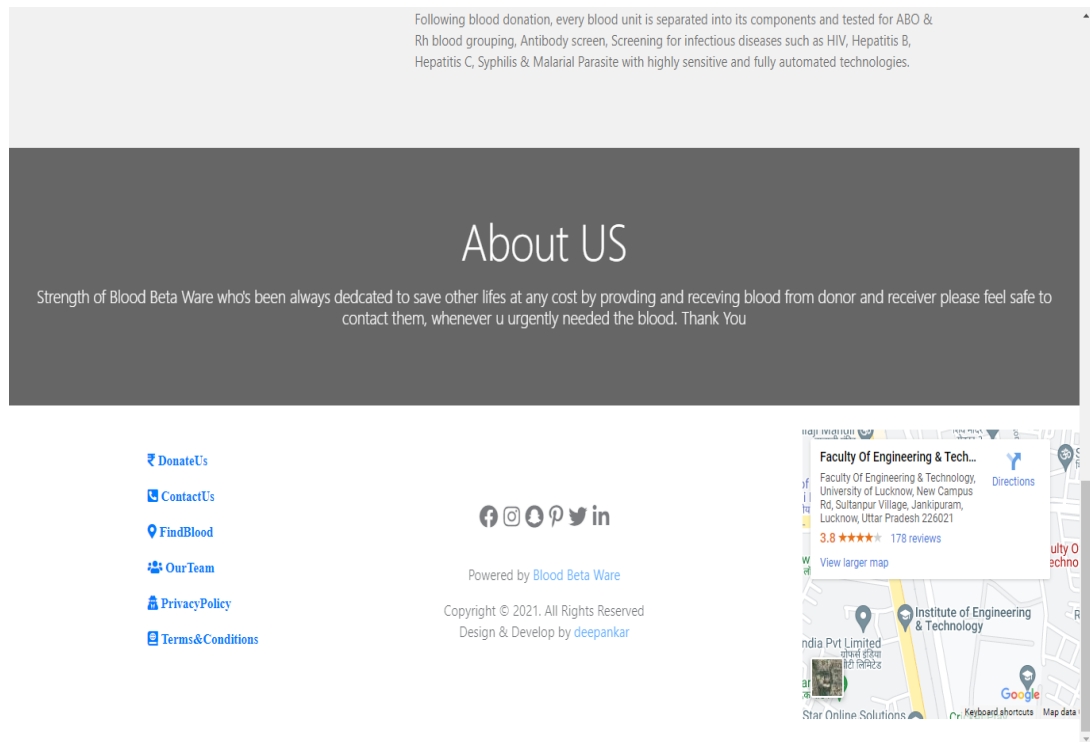


Figure 21 Homepage\_3

**Blood Beta Ware** Home About Contact Us Admin Login

**Welcome**

Your's small help give other's life

[Org Login](#)

**Register as a Doner**

[Doner](#) [Organization](#)

Your Name \*

Your Email \*

Password \*

Your Phone \*

Your Age \*

Blood Group

Your Weight\*

State

City \*

Pincode \*

Last Donation


dd-----yyyy

☒ Male ☐ Female


[Register](#)

Figure 22 Register as a Doner




**Blood Beta Ware**

[Home](#)
[About](#)
[Contact Us](#)
[Admin Login](#)



**Welcome**

Your's small help give other's life

[Org Login](#)

Doner

Organization

### Register as an Organization

Org/Hospital Name \*

Your Email \*

Password \*

Your Phone \*

Blood Component Available

Service Time

License Number \*

State

City \*


Pincode \*

License Date


dd----yyyy

[Register](#)

*Figure 23 Register as an Organization/Hospital*


**Blood Beta Ware**

[Home](#)
[About](#)
[Contact Us](#)
[Admin Login](#)




### Blood Beta Ware

Email address

Email address

Password





Password

☐ I'm not a robot
 

☐ Remember password

[Forgot password?](#)

[SIGN IN](#)

Powered by [Blood Beta Ware](#)

*Figure 24 Login Panel*

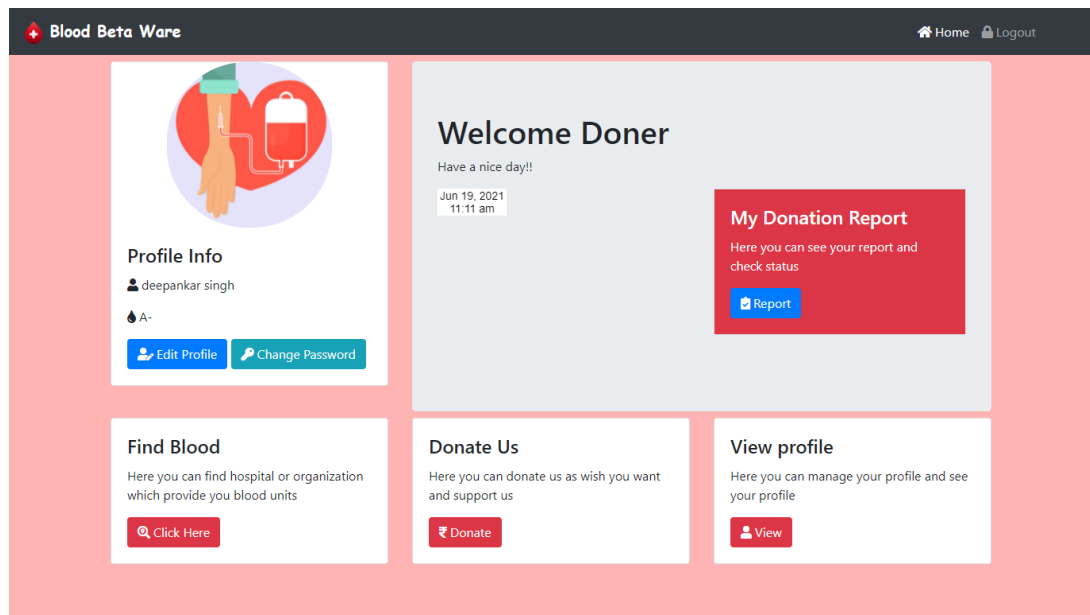


Figure 25 Doner\_Panel

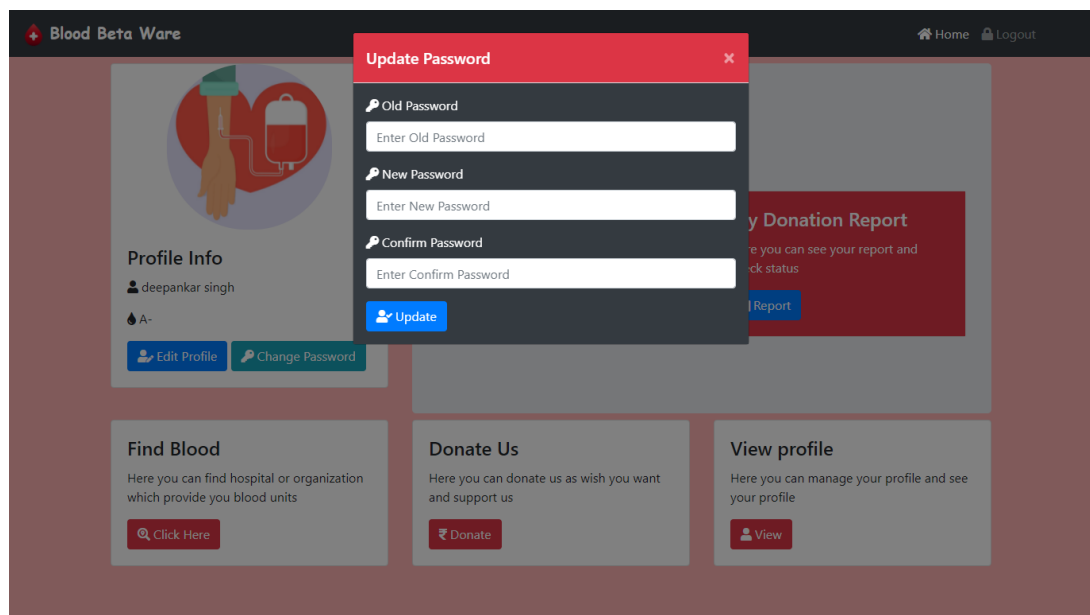



Figure 26 Update\_Password\_Doner

**Blood Beta Ware** Home Logout

 **deepankar singh** [Blood Doner](#) [Edit Profile](#)

Blood Group : A-

About

Doner Id	9
Name	deepankar singh
Email	abc@gmail.com
Phone	7894561230
Gender	Male
Last Donation	2021-06-18
Age	18
Weight	40
City	Lucknow
State	Uttar Pradesh
Pincode	226021

Figure 27 Doner\_Profile

**Update Details**

Name

Mobile

Age

Weight

City

State

Pincode

[Update](#)

Figure 28 Update\_Doner\_Profile

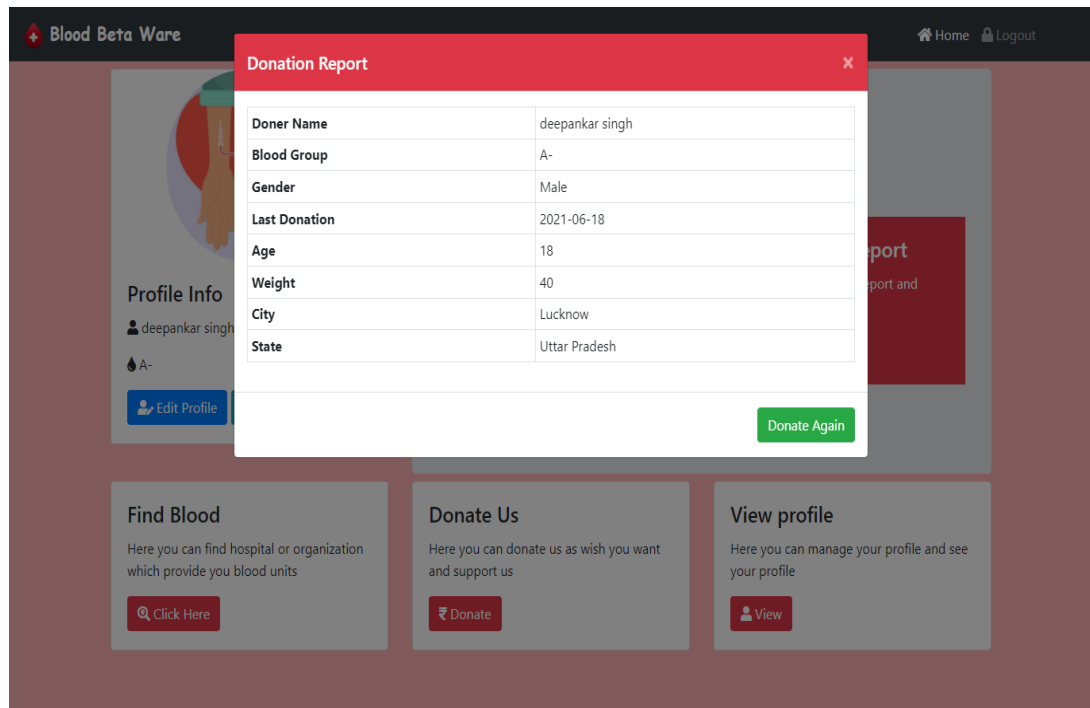


Figure 29 Donation\_Report

**Donate Us with paytm**  
UPI Mobile Number:  
9525874632

Name:

Email:

Address:

Amount:

Figure 30 Donate\_Us

[Home](#)
[About](#)
[Contact Us](#)
[DonateUs](#)
[Organization](#)
[Admin Login](#)



Powered by [Blood Beta Ware](#)  
 Copyright © 2021. All Rights Reserved  
 Design & Develop by [deepankar](#)

Figure 31 Find\_Blood

## Welcome ABC Hospital

[My Profile](#)
[Doner Registration](#)
[See Details](#)
[Others](#)
[Camp/Counter Registration](#)
[Logout](#)

Name	ABC Hospital
Email	abc@gmail.com
Mobile	1234567890

[Update Profile Details](#)

### Update Profile

Name

Email address

Mobile Number

[Change Password](#)

[Update Organization Details](#)

Figure 32 Organization/Hospital\_Panel

## Welcome ABC Hospital

[My Profile](#) [Doner Registration](#) [See Details](#) [Others](#) [Camp/Counter Registration](#) [Logout](#)

### Organization/Hospital Details

Name	ABC Hospital
Email	abc@gmail.com
Mobile	1234567890
City	Noida
State	Uttar Pradesh
Pincode	226000
Blood Component Available	Yes
License Number	2021/25/ABC
License Date	2021-03-23



Figure 33 Organization/Hospital\_Details

Welcome ABC Hospital

[My Profile](#) [Doner Registration](#) [See Details](#) [Others](#) [Camp/Counter Registration](#) [Logout](#)

Name	ABC Hospital
Email	abc@gmail.com
Mobile	1234567890

Registration

Doner Detail's

Name of Doner

Name \*

Email

Email \*

Mobile

Mobile \*

Blood Group

Blood Group

City

Enter City

State

State

Blood Donate

Blood Donate

Name of Org/Hospital

ABC Hospital

+ Register

Update Profile

Update

Figure 34 Add\_Temp\_Doner

Welcome ABC Hospital

My Profile Doner Registration See C

Name	ABC Hospital
Email	abc@gmail.com
Mobile	1234567890

Illustration of a heart with a cross and people donating blood.

**Registration**

**Camp/Counter Registration**

Camp/Counter name

Name \*

Camp/Counter Date

dd - ---- - yyyy

Total Unit Donation

Total Unit \*

Total Person

Total Person \*

Name of Org/Hospital

ABC Hospital

**+ Register**

**Update**

[Change Password](#)

[Update Organization Details](#)

Figure 35 Register\_Camp/Counter

Mobile 1234567890

Illustration of a heart with a cross and people donating blood.

[Update Organization Details](#)

**Update Organization Or Hospital**

Name

ABC Hospital

City

Noida

State

State

Pincode

226000

Blood Component Available

Select

Service Time

Select

Licence Number

2021/25/ABC

Licence Date

23-Mar-2021

**Update**

Figure 36 Update\_Org/Hosp\_Details

## Welcome ABC Hospital

[My Profile](#) [Doner Registration](#) [See Details](#) [Others](#) [Camp/Counter Registration](#) [Logout](#)



Total Doner

8



Total Camp/Counter

3



Total Unit Available

9 Unit

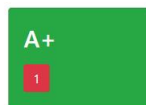


Figure 37 Org/Hosp\_Dashboard

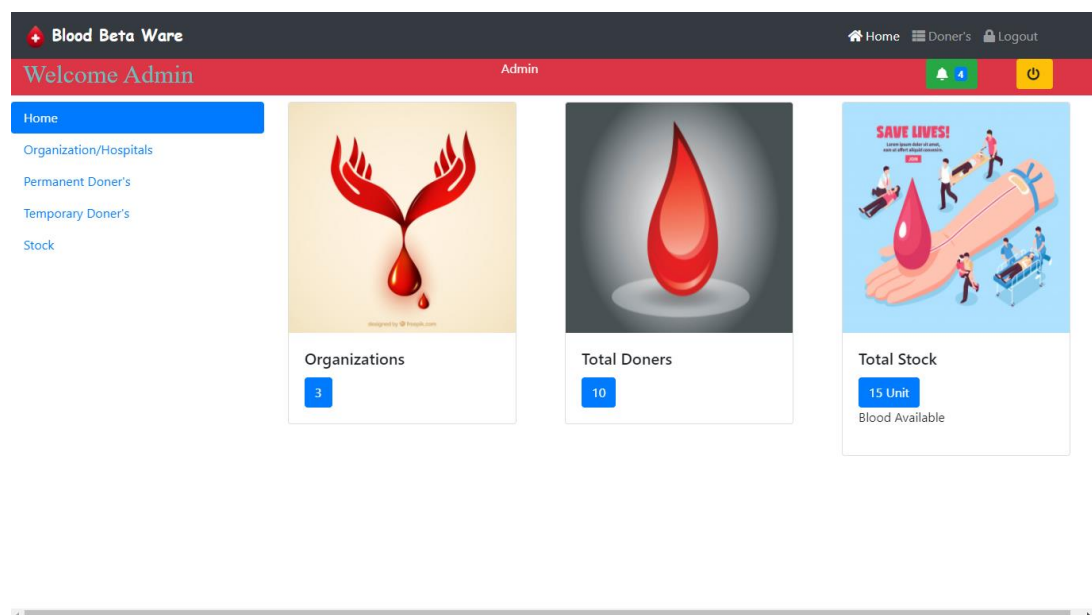


Figure 38 Admin\_Panel



<div> <div>Blood Beta Ware</div> <div> <a href="#">Home</a> <a href="#">Doner's</a> <a href="#">Logout</a> </div> </div> <div> <div>Welcome Admin</div> <div>Admin</div> <div> </div> </div>																																																						
<div> <a href="#">Home</a> <a href="#">Organization/Hospitals</a> <a href="#">Permanent Doner's</a> <a href="#">Temporary Doner's</a> <a href="#">Stock</a> </div> <div> <div>Organization's List</div> <table> <tr> <th>#</th><th>Name</th><th>Mobile</th><th>City</th><th>State</th><th>Available</th><th>ServiceTime</th><th>Lic-Num</th><th>Lic_date</th><th>Pincode</th><th>Action</th></tr> <tr> <td>1</td><td>ABC Hospital</td><td>1234567890</td><td>Noida</td><td>Uttar Pradesh</td><td>Yes</td><td>24Hr</td><td>2021/25/ABC</td><td>2021-03-23</td><td>226000</td><td> </td></tr> <tr> <td>2</td><td>XYZ</td><td>7894561230</td><td>Lucknow</td><td>Uttar Pradesh</td><td>Yes</td><td>24Hr</td><td>XYZ/2021/01</td><td>2021-06-06</td><td>142536</td><td> </td></tr> <tr> <td>4</td><td>Apollo Hospital</td><td>1472583695</td><td>Delhi</td><td>Delhi</td><td>Yes</td><td>24Hr</td><td>apollo/2021/01</td><td>2021-06-06</td><td>452563</td><td> </td></tr> </table> </div>											#	Name	Mobile	City	State	Available	ServiceTime	Lic-Num	Lic_date	Pincode	Action	1	ABC Hospital	1234567890	Noida	Uttar Pradesh	Yes	24Hr	2021/25/ABC	2021-03-23	226000		2	XYZ	7894561230	Lucknow	Uttar Pradesh	Yes	24Hr	XYZ/2021/01	2021-06-06	142536		4	Apollo Hospital	1472583695	Delhi	Delhi	Yes	24Hr	apollo/2021/01	2021-06-06	452563	
#	Name	Mobile	City	State	Available	ServiceTime	Lic-Num	Lic_date	Pincode	Action																																												
1	ABC Hospital	1234567890	Noida	Uttar Pradesh	Yes	24Hr	2021/25/ABC	2021-03-23	226000																																													
2	XYZ	7894561230	Lucknow	Uttar Pradesh	Yes	24Hr	XYZ/2021/01	2021-06-06	142536																																													
4	Apollo Hospital	1472583695	Delhi	Delhi	Yes	24Hr	apollo/2021/01	2021-06-06	452563																																													

Figure 39 Org\_List

Blood Beta Ware

Home

Doner's

Logout

Welcome Admin

Admin

4

Home

Organization/Hospitals

Permanent Doner's

Temporary Doner's

Stock

Doner's List

#	Name	Email	BloodGroup	Gender	Age	Weight	City	State	Action
9	deepankar singh	abc@gmail.com	A-	Male	18	40	Lucknow	Uttar Pradesh	<div><div></div><div></div></div>
10	anurag	a@gmail.com	O-	Male	20	54	Varanasi	Uttar Pradesh	<div><div></div><div></div></div>
12	Arun	arun@gmail.com	A-	Male	21	51	Kanpur	Uttar Pradesh	<div><div></div><div></div></div>
13	Ali	ali@gmail.com	A+	Male	25	35	Kanpur	Uttar Pradesh	<div><div></div><div></div></div>
14	simran	sim@gmail.com	AB+	Female	25	26	Bareilly	Uttar Pradesh	<div><div></div><div></div></div>
15	sofia	sof@gmail.com	AB-	Female	20	50	Azamgarh	Uttar Pradesh	<div><div></div><div></div></div>
16	karan	karan@gmail.com	O+	Male	30	55	Balia	Uttar Pradesh	<div><div></div><div></div></div>

Figure 40 Permanent\_Doner\_List

Blood Beta Ware

Home

Doner's

Logout

Welcome Admin

Admin

1

2

Home

Organization/Hospitals

Permanent Doner's

Temporary Doner's

Stock

Temporary Doner's List

#	Name	Email	Mobile	Blood Group	City	State	Belongs To	Action
11	deepankar	dp@gmail.com	1234567890	B+	Lucknow	Uttar Pradesh	ABC Hospital	<div></div>
12	amit	amit@gmail.com	1472583692	A+	Kanpur	Uttar Pradesh	ABC Hospital	<div></div>
13	karan	k@gmail.com	1473692581	A-	Varanasi	Uttar Pradesh	ABC Hospital	<div></div>
14	rani	r@gmail.com	4561237892	B-	Rampur	Uttar Pradesh	ABC Hospital	<div></div>
15	Fatima	f@gmail.com	7894561232	O+	Noida	Uttar Pradesh	ABC Hospital	<div></div>
16	ram	ram@gmail.com	1234567895	O-	Balia	Uttar Pradesh	ABC Hospital	<div></div>
17	kamal	k@gmail.com	1452365287	AB+	Hardoi	Uttar Pradesh	ABC Hospital	<div></div>
18	payal	p@gmail.com	7854965825	AB-	Sitapur	Uttar Pradesh	ABC Hospital	<div></div>
19	raman	r@gmail.com	1234567896	A-	Lucknow	Uttar Pradesh	Amala Hospital	<div></div>

Figure 41 Temp\_Doner\_List

Blood Beta Ware

Home

Doner's

Logout

Welcome Admin

Admin

4

Home

Organization/Hospitals

Permanent Doner's

Temporary Doner's

Stock

Doner Details from different Blood Groups

A-2

A+1

AB-1

AB+1

B-1

B+1

O-1

O+1

Total Permanent Doner10

Total Temporary Doner15

Total Available Blood Unit17 Unit

Total Camp/Couter Organized4

Figure 42 Stock\_Details

## **FUTURE SCOPE**

- System can be expanded with availability over worldwide.
- Reaching as close as possible of the donor from emergency zone.
- A smart phone application of the system can be made.
- Providing Doners an option of change his/her availability.

## References

- <https://www.tutorialspoint.com/index.htm>
- <https://www.javatpoint.com>
- <https://www.w3schools.com>
- <https://html.com>
- <https://www.geeksforgeeks.org/>