

Sharda Shaikshanik & Samajik Sanstha's
**Vidhydhan Commerce & Science
College Walwadi, Dhule(200111)**



A PROJECT REPORT ON
“Blood Bank Management System”
Submitted By
GUIDE BY
PROF. MRS. Punam D. Patil
IN PARTIAL FULLFILLMENT OF DEGREE OF
BACHORAL OF COMPUTER APPLICATION (BCA) 21-22



**Kavayitri Bahinabai Chaudhari
North Maharashtra University, Jalgaon**

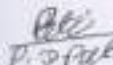


Certificate

This is to certify that **Mis. Priyanka S. Patil** final year student of Bachelor in Bachelor of Computer Applications (BCA) has been successfully complete these project entitled "**Blood Bank Management System**" as the record of the work carried out by his is accepted in partial fulfillment of the requirement for degree of Bachelor in Bachelor of Computer Applications (BCA) in the Bachelor of Computer Applications (BCA) vidyadhan Commerce & Science college Dhule under the guidance of **Prof. Mrs. Punam D.Patil** in the academic year 2021-2022.

Place: **DHULE**

Date **23/5/2022**

Prof. 
(Project Guide)



Head of Department
Computer & Management
Vidyadhan College, Valwadi

Internal Examiner


Principal
Sharda Educ. & Social Sanstha
Vidyadhan Com. & Sci College
Valwadi (Dhule)


External Examiner

INDEX

“Blood Bank Management System”

Index

❖ **Introduction**

❖ **Requirement Analysis**

❖ **Module Description**

❖ **Design**

❖ **Database**

❖ **Testing**

❖ **Input Screen**

❖ **Output Screen**

❖ **Conclusion**

❖ **Bibliography**

INTRODUCTION

Introduction

The BLOOD BANK MANAGEMENT SYSTEM is a great project. This project is designed for the successful completion of a project on a blood bank management system. The basic building aim is to provide blood donation services to the city recently. Blood Bank Management System (BBMS) is a browser-based system that is designed to store, process, retrieve, and analyze information concerned with the administrative and inventory management within a blood bank. This project aims at maintaining all the information pertaining to blood donors, different blood groups available in each blood bank, and help them manage in a better way. The aim is to provide transparency in this field, make the process of

obtaining blood from a blood bank hassle free and corruption free and make the system of blood bank management effective.

The *Blood bank system project report* contain information related to blood like

- Blood type
- Date of Donation of blood
- validity of Blood s
- Available Blood group

Need of Blood Bank Management System



Bank blood donation system in java is planned to collect blood from many donators in short from various sources and distribute that blood to needy people who require blood. To do all this we require high quality software to manage those jobs. The government spending lot of money to develop high quality "Blood Bank management system project". Help Line is an voluntary and non-governmental organization.It maintains Online library of blood donors in India. Sometimes Doctors and Blood bank project have to face the difficulty in finding the blood group Donors at right time. Help Line has attempted to provide the answer by taking upon itself the task of collecting Blood bank project nationwide for the cause and care of people in need.

At any point of time the people who are in need can reach the donors through our search facility. By mobilizing people and organization who desire to make a difference in the lives of people in need. On the basis of humanity, Everyone is welcome to register as a blood donor._Blood Bank Management System (BBMS) is a browser based system that is designed to store, process, retrieve and analyze information concerned with the administrative and inventory management within a blood bank. This project aims at maintaining all the information pertaining to blood donors, different blood groups available in each blood bank and help them manage in a better way.

REQUIREMENT ANALYSIS

Requirement analysis

Hardware Requirement

 Processor	: Intel Core Duo 2.0 GHz or more
 RAM	: 1 GB or More
 Harddisk	: 80GB or more

✚ Monitor	: 15” CRT, or LCD monitor
✚ Keyboard	: Normal or Multimedia
✚ Mouse	: Compatible mouse

Software Requirement

✚ Project Name	Blood Bank Management System
✚ Language Used	PHP5.6, PHP7.x
✚ Database	MySQL 5.x
✚ User Interface Design	HTML, AJAX,JQUERY,JAVASCRIPT
✚ Web Browser	Mozilla, Google Chrome, IE8, OPERA
✚ Software	XAMPP / Wamp / Mamp/ Lamp (anyone)

PHP is now officially known as “**PHP: Hypertext Preprocessor**”. It is a server-side scripting language usually written in an HTML context. Unlike an ordinary HTML page, a PHP script is not sent directly to a client by the server; instead, it is parsed by the PHP binary or module, which is server-side installed. HTML elements in the script are left alone, but PHP code is interpreted and executed. PHP code in a script can query databases, create images, read and write files, talk to remote servers – the possibilities is endless. The output from PHP code is combined with the HTML in the script and the

result sent to the user's web-browser, therefore it can never tell the user whether the web-server uses PHP or not, because the entire browser sees is HTML.

PHP's support for Apache and MySQL further increases its popularity. Apache is now the most-used web-server in the world, and PHP can be compiled as an Apache module. MySQL is a powerful free SQL database, and PHP provides a comprehensive set of functions for working with it. The combination of Apache, MySQL and PHP is all but unbeatable.

That doesn't mean that PHP cannot work in other environments or with other tools. In fact, PHP supports an extensive list of databases and web-servers. While in the mid-1990s it was ok to build sites, even relatively large sites, with hundreds of individual hard-coded HTML pages, today's webmasters are making the most of the power of databases to manage their content more effectively and to personalize their sites according to individual user preferences.

Reasons for using PHP

There are some indisputable great reasons to work with PHP. As an open source product, PHP is well supported by a talented production team and a committed user community. Furthermore, PHP can be run on all the major operating systems with most servers.

Learning PHP is easy

Basic is easy any interpreted language should be easy to learn. Since you are isolated from the system (no pointers to use, no memory to allocate). The other advantage that all modern interpreted languages share is good associative array constructs.

Its Performance

While we can build an application that serves millions of pages a day on a server, when we really look at the performance of the language it sucks. We are still orders of magnitude from real performance. Not only that, but since PHP is designed around a single process model our ability to share data structures or connection pool resources is left to native code libraries.

PHP Syntax

You cannot view the PHP source code by selecting "View source" in the browser – you will only see the output from the PHP file, which is plain HTML. This is because the scripts are executed on the server before the result is sent back to the browser.

Basic PHP Syntax

A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document. On servers with shorthand support enabled you can start a scripting block with **<?** and end with **?>**. However, for maximum compatibility, we recommend that you use the standard form (**<?php**) rather than the shorthand form.

A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code.

HTML

HTML or **Hyper Text Markup Language** is the standard markup language used to create web pages.

HTML was created in 1991 by Tim Berners-Lee at CERN in Switzerland. It was designed to allow scientists to display and share their research.

HTML is written in the form of HTML elements consisting of *tags* enclosed in angle brackets (like **<html>**). HTML tags most commonly come in pairs like **<h1>** and **</h1>**, although some tags represent *empty elements* and so are unpaired, for example ****. The first tag in a pair is the *start tag*, and the second tag is the *end tag* (they are also called *opening tags* and *closing tags*).

The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language rather than a programming language.

HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as Java Script which affect the behavior of HTML web pages.

HTML is descriptive markup language. Library of various markup languages is defined in various browsers.

HTML Images - The Tag and the Src Attribute

In HTML, images are defined with the **** tag.

The **** tag is empty, which means that it contains attributes only, and has no closing tag.

To display an image on a page, you need to use the src attribute. Src stands for "source". The value of the src attribute is the URL of the image you want to display.

Syntax for defining an image:

```
<imgsrc="url" alt="some_text">
```

HTML FORMS

HTML forms are used to pass data to a server.

An HTML form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label elements.

Image tag () :

To add an image to an HTML document, we just need to include an tag with a reference to the desired image. The tag is an empty element i.e. it doesn't require a

closing tag and we can use it to include from small icons to large images.

Syntax: <imgsrc="URL" alt="alternative text">

3 HTML 5

HTML5 will be the new standard for HTML. The previous version of HTML, HTML 4.01, came in 1999. The web has changed a lot since then. HTML5 is still a work in progress.

However, the major browsers support many of the new HTML5 elements and APIs.

HTML5 is cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).

WHATWG was working with web forms and applications, and W3C was working with XHTML 2.0. In 2006, they decided to cooperate and create a new version of HTML.

Some rules for HTML5 were established:

- a) New features should be based on HTML, CSS, DOM, and JavaScript
- b) Reduce the need for external plug-ins (like Flash)
- c) Better error handling
- d) More markup to replace scripting
- e) HTML5 should be device independent

f) The development process should be visible to the public

CSS

CSS tutorial or CSS 3 tutorial provides basic and advanced concepts of CSS technology. Our CSS tutorial is developed for beginners and professionals. The major points of CSS are given below:

CSS stands for Cascading Style Sheet.

CSS is used to design HTML tags.

CSS is a widely used language on the web.

HTML, CSS and JavaScript are used for web designing. It helps the web designers to apply style on HTML tags.

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. While most often used to style web pages and user interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL. CSS is a cornerstone specification of the web and almost all web pages use CSS style sheets to describe their presentation.

CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content (such as by allowing for table less web design).

CSS can also allow the same markup page to be presented in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or screen reader) and on Braille-based, tactile devices. It can also be used to allow the web page to display differently depending on the screen size or device on which it is being viewed. While the author of a document typically links that document to a CSS file, readers can use a different style sheet, perhaps one on their own computer, to override the one the author has specified.

With plain HTML you define the colors and sizes of text and tables throughout your pages. If

JAVASCRIPT

JavaScript (JS) is a dynamic computer programming language. It is most commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. It is also being used in server-side network programming (with Node.js), game development and the creation of desktop and mobile applications.

JavaScript is a prototype-based scripting language with dynamic typing and has first-class functions. Its syntax was influenced by C. JavaScript copies many names and naming conventions from Java, but the two languages are otherwise unrelated and have very different semantics. The key design principles within JavaScript are taken from the Self and Scheme programming languages. It is a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles.

The application of JavaScript in use outside of web pages—for example, in PDF documents, site-specific browsers, and desktop widgets—is also significant. Newer and faster JavaScript VMs and platforms built upon them (notably Node.js) have also increased the popularity of JavaScript for server-side web applications. On the client side, JavaScript was traditionally implemented as an interpreted language but just-in-time compilation is now performed by recent (post-2012) browsers.

JavaScript was formalized in the ECMA Script language standard and is primarily used as part of a web browser (client-side JavaScript). This enables programmatic access to objects within a host environment.

JavaScript is the most popular programming language in the world.

It is the language for HTML, for the Web, for computers, servers, laptops, tablets, smart phones, and more.

You can use JavaScript to:

- a) Change HTML elements
 - Delete HTML elements
 - Create new HTML elements
 - Copy and clone HTML elements

7.6 MySQL's Logical Architecture

The database has become an integral part of almost every human's life. Without it, many things we do would become very tedious, perhaps impossible tasks. Banks, universities, and libraries are three examples of organizations that depend heavily on some sort of database system. On the Internet, search engines, online shopping, and even the website naming convention would be impossible without the use of a database. A database that is implemented and interfaced on a computer is often termed a database server.

The topmost layer contains the services that aren't unique to MySQL. They're services most network-based client/server tools or servers need: connection handling, authentication, security, and so forth

The third layer contains the storage engines. They are responsible for storing and retrieving all data stored "in" MySQL. Like the various filesystems available for GNU/Linux, each storage engine has its own benefits and drawbacks. The server communicates with them through the *storage engine API*. This interface hides differences between storage engines and makes them largely transparent at the query layer. The API contains a couple of dozen low-level functions that perform operations such as "begin a transaction" or "fetch the row that has this primary key." The storage engines don't parse SQL^[4] or communicate with each other; they simply respond to requests from the server.

softwares and tools used:

7.7.1 My Sql:

Introduction:

The database has become an integral part of almost every human's life. Without it, many things we do would become very tedious, perhaps impossible tasks. Banks, universities, and libraries are three examples of organizations that depend heavily on some sort of database system. On the Internet, search engines, online shopping, and even the website naming convention would be impossible without the use of a database. A database that is implemented and interfaced on a computer is often termed a database server.

One of the fastest SQL (Structured Query Language) database servers currently on the market is the MySQL server, developed by T.c.X. DataKonsultAB. MySQL, available for download at www.mysql.com, offers the database programmer with an array of options and capabilities rarely seen in other database servers. MySQL is free of

charge for those wishing to use it for private and commercial use. Those wishing to develop applications specifically using MySQL should consult MySQL's licensing section,

These capabilities range across a number of topics, including the following:

- a) Ability to handle an unlimited number of simultaneous users.
- b) Capacity to handle 50,000,000+ records.
- c) Very fast command execution, perhaps the fastest to be found on the market.
- d) Easy and efficient user privilege system.

However, perhaps the most interesting characteristic of all is the fact that it's free. That's right, T.c.X offers MySQL as a free product to the general public.

Reasons to Use MySQL

a) Scalability and Flexibility

The MySQL database server provides the ultimate in scalability, sporting the capacity to handle deeply embedded applications with a footprint of only 1MB to running massive data warehouses holding terabytes of information. Platform flexibility is a stalwart feature of MySQL with all flavors of Linux, UNIX, and Windows being supported.

b) High Performance

A unique storage-engine architecture allows database professionals to configure the MySQL database server specifically for particular applications, with the end result being amazing performance results.

C) High Availability

Rock-solid reliability and constant availability are hallmarks of MySQL, with customers relying on MySQL to guarantee around-the-clock uptime. MySQL offers a variety of high-availability options from high-speed master/slave replication configurations, to specialized Cluster servers offering instant failover, to third party vendors offering unique high-availability solutions for the MySQL database server.

d) Robust Transactional Support

MySQL offers one of the most powerful transactional database engines on the market. Features include complete ACID (atomic, consistent, isolated, durable) transaction

support, unlimited row-level locking, distributed transaction capability, and multi-version transaction support where readers never block writers and vice-versa.

e) Web and Data Warehouse Strengths

MySQL is the de-facto standard for high-traffic web sites because of its high-performance query engine, tremendously fast data inserts capability, and strong support for specialized web functions like fast full text searches.

f) Strong Data Protection

Because guarding the data assets of corporations is the number one job of database professionals, MySQL offers exceptional security features that ensure absolute data protection. In terms of database authentication, MySQL provides powerful mechanisms for ensuring only authorized users have entry to the database server, with the ability to block users down to the client machine level being possible.

g) Management Ease

MySQL offers exceptional quick-start capability with the average time from software download to installation completion being less than fifteen minutes. This rule holds true whether the platform is Microsoft Windows, Linux, Macintosh, or UNIX.

PHP Main Features of MySQL

- Tested with a broad range of different compilers.
- Works on many different platforms.
- The MySQL Server design is multi-layered with independent modules.
- Fully multi-threaded using kernel threads. It can easily use multiple CPUs if they are available.
- Provides transactional and non-transactional storage engines.
- Uses very fast B-tree disk tables with index compression.
- Relatively easy to add other storage engines. This is useful if you want to provide an SQL interface for an in-house database.
- A very fast thread-based memory allocation system.
- Very fast joins using an optimized one-sweep multi-join.
- In-memory hash tables, which are used as temporary tables.

- SQL functions are implemented using a highly optimized class library and should be as fast as possible. Usually there is no memory allocation at all after query initialization.

Software Development Process:

Life Cycle Used to develop this Project

Life cycle used ---- **SDLC**

Systems Development Life Cycle (SDLC), or *Software Development Life Cycle*, in systems engineering and software engineering relates to the process of developing systems, and the models and methodologies, that people use to develop these systems, generally computer or information systems.

In software engineering this SDLC concept is developed into all kinds of software development methodologies, the framework that is used to structure, plan, and control the process of dev

Overview

Systems Development Life Cycle (SDLC) is any logical process used by a systems analyst to develop an information system, including requirements, validation, training, and user ownership. An SDLC should result in a high quality system that meets or exceeds customer expectations, within time and cost estimates, works effectively and efficiently in the current and planned Information Technology infrastructure, and is cheap to maintain and cost-effective to enhance.

Computer systems have become more complex and usually (especially with the advent of Service-Oriented Architecture) link multiple traditional systems often supplied by different software vendors. To manage this, a number of system development life cycle (SDLC) models have been created: waterfall, fountain, spiral, build and fix, rapid prototyping, incremental, and synchronize and stabilize. Although in the academic sense, SDLC can be used to refer to various models, SDLC is typically used to refer to a waterfall methodology.

In project management a project has both a life cycle and a "systems development life cycle" during which a number of typical activities occur. The project life cycle (PLC) encompasses all the activities of the project, while the systems development life cycle (SDLC) is focused on accomplishing the product requirements.

Systems Development Phases

Systems Development Life Cycle (SDLC) adheres to important phases that are essential for developers, such as planning, analysis, design, and implementation, and are explained in the section below. There are several Systems Development Life Cycle Models in existence. The oldest model, that was originally regarded as "the Systems Development Life Cycle" is the waterfall model: a sequence of stages in which the output of each stage becomes the input for the next. These stages generally follow the same basic steps but many different waterfall methodologies give the steps different names and the number of steps seems to vary between 4 and 7. There is no definitively correct Systems Development Life Cycle model, but the steps can be characterized and divided in several steps.

Feasibility study

A feasibility study specifies whether the proposed software project is practically possible or not. Whenever there arises a need for any software you don't directly jump in developing the particular software. Instead, we must first analyze certain facts to realize whether the software is worthwhile or not and this is called the feasibility study.

The feasibility study determines whether the software project can be achieved technically, organizationally, economically or not, and there are many other parameters to consider. Well, each organization performs the feasibility study in different ways. Some do it elaboratively and systematically, some might do it in an ad hoc fashion and some may not do it all.

Content: Feasibility Study

1. [Importance](#)
2. [Types](#)
3. [Benefits](#)

Importance of Feasibility Study

The feasibility study is of much more importance if done correctly. If the feasibility study is not done correctly it is of no use. Key points to determine the importance of feasibility study are:



- ✓ It prevents you from committing, investing your resources and capital for an impractical project.
- ✓ It might discover new ideas that might completely change the scope of your software.
- ✓ Improves the confidence of the team to concentrate on their project.
- ✓ Validate the need for the software.
- ✓ Estimates the risk involved in developing the software and analyzed if they can be minimized.
- ✓ Types of Feasibility Study
- ✓ There are three components of feasibility study i.e technical, organizational and economical. We have discussed each type in the section below:

The technical feasibility study determines whether the current technology and technical staff available with the organization will accomplish the software's development or not. The key issues that must be addressed during the technical feasibility study are as follow:

1. Does the technical team available has knowledge to analyze, design, code for software? And they must also be able to install, operate and maintain the software.
2. Can the proposed software meet the user's requirement?
3. Technology is getting advanced with each passing day so the developed software product must sustain with the advancing technology in the future.
4. Do we have the approved and accepted tools and equipment's to develop the software?

5. Examine the complexity of the software project. A project with higher complexity must be provided with a high degree of technical skills and experience.

2. Organizational Feasibility

Though the software project is feasible from the technical aspect what if you need to make major organizational changes. It will make things more complex. To check if the software is organizationally or operationally feasible or not we must consider the following key points:

1. Identify whether the organizational staff is ready to accept the new software or not
2. Is the organizational staff technical sound to cope up with new technologies?
3. Does the organization hardware infrastructure capable to support the new software?

Now, if the new software gets the staffs acceptance then it is more like that the software will be successful. If the organization has to make more changes in its policies, structures and standers then the software's failure risk gets increased. Apart from the above considerations, there are some more analyses as discussed below:

- **Competition:** Before we start developing the software we must consider the current trends of the market and the competitive software available in the market. This will help in the development of the software.
- **Uniqueness:** Though there are several competitor software available in the market the newly developed software must be known for its uniqueness. It must offer something that the existing software in the market doesn't offer.
- **Performance:** The performance of the software needs to be traced to whether it is performing financially and technically well corresponding to its requirement.

3. Economical Feasibility

For software to be economically feasible the cost expends to develop the software must exceed its benefits. The cost expends to develop a software includes:

1. An overall analysis of the system and its design.
2. Hardware purchased to develop the software
3. Software required in the development of the software.
4. Training required for the organization personals.
5. Cost invested in Installation of the software.
6. The cost required in operating the software

Now the benefits of the developed software must outweigh the cost required to develop the software. Some benefits can not be measured such as:

- Benefits of reduced labour cost.
- Benefits of faster processing.
- Benefits of improved customer service.
- Benefits of reduced errors.
- Benefits of Feasibility Study

A feasibility study gets you a clear-cut idea of whether the software will be a success even before you invest in the software.

- It increases the focus of the software development team.
- The software developer team may come across new opportunities.
- Confirms the developer team whether to accept the software project or not.
- Helps in evaluating the success rate of the software.

So, this is all about the feasibility study. We have learned about the feasibility study, in brief, we have also discussed its importance. Then we have discussed the types of components of the feasibility study. We have concluded our discussion with its benefits

MODULE DESCRIPTION

Modules of BBMS

- Donor Registration
- Blood Login
- Blood Donor
- Equipments
- Stick
- Blood Recipient
- Blood collection
- Camp
- Stock details
- Blood issued

BLOOD DONATION is a website based on PHP. The purpose of this project was to develop a blood management information system to assist in the management of blood donor records and ease or control the distribution of blood in various part of country basing on the hospitals demand. . This project includes mainly two modules i.e. login and main page.

- ◆ **Login:**
- ◆ **Admin**
- ◆ **User**

Admin: The page require user name and password to start the application. Login is a process by which individual access to a computer system is controlled by identifying and authenticating the user through the cardinalities presented by the user. Admin can add update or delete the user, city, state, camp etc.

DESIGN

Design

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

It shows how data enters and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

The following observations about DFDs are essential:

1. All names should be unique. This makes it easier to refer to elements in the DFD.
2. Remember that DFD is not a flow chart. Arrows in a flow chart that represents the order of events; arrows in DFD represents flowing data. A DFD does not involve any order of events.
3. Suppress logical decisions. If we ever have the urge to draw a diamond-shaped box in a DFD, suppress that urge! A diamond-shaped box is used in flow charts to represent decision points with multiple exists paths of which the only one is taken. This implies an ordering of events, which makes no sense in a DFD.
4. Do not become bogged down with details. Defer error conditions and error handling until the end of the analysis.

Standard symbols for DFDs are derived from the electric circuit diagram analysis and are shown in fig:

Symbol	Name	Functions
	Data flow	Used to Connect Processes to each other, to sources or Sinks; an arrow head indicates direction of data flow.
	Process	Performs some transformation of input data to yield output data.
	Source or Sink (External Entity)	A Source of System inputs or Sink of System outputs.
	Data Store	A repository of data; the arrow heads indicate not inputs and not outputs to store.

Symbols for Data Flow Diagrams

Circle: A circle (bubble) shows a process that transforms data inputs into data outputs.

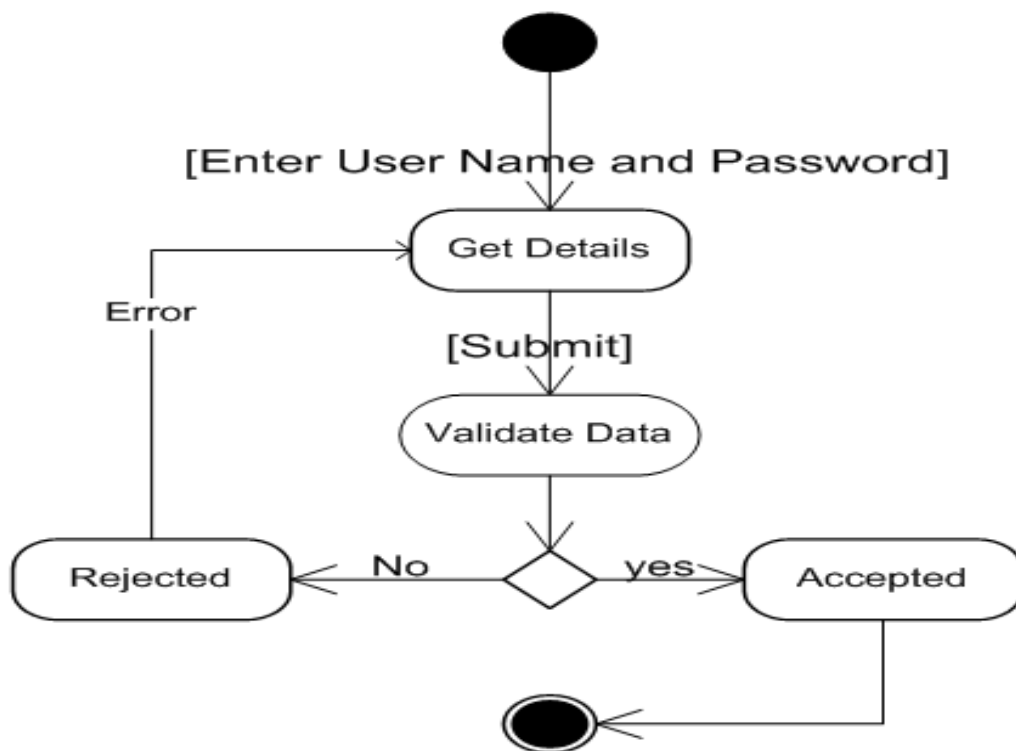
Data Flow: A curved line shows the flow of data into or out of a process or data store.

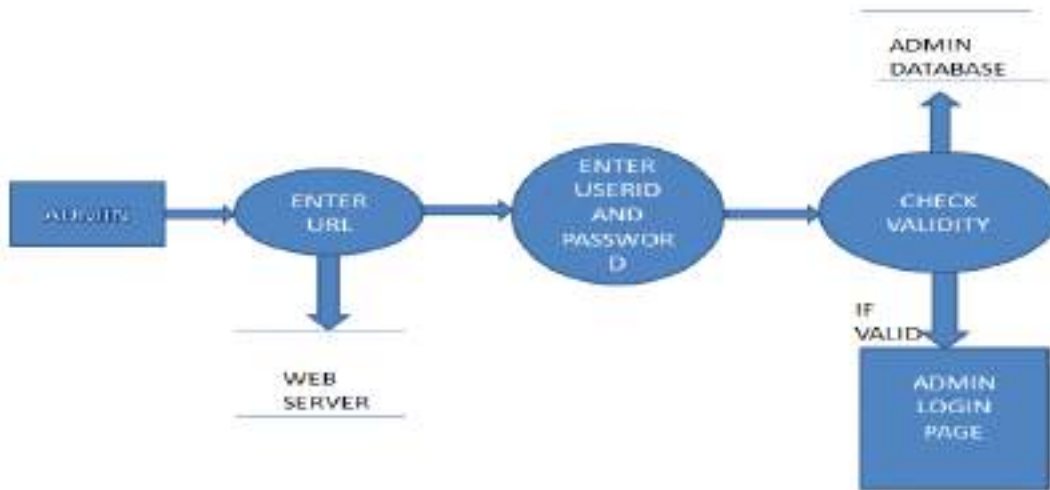
Data Store: A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.

Source or Sink: Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.

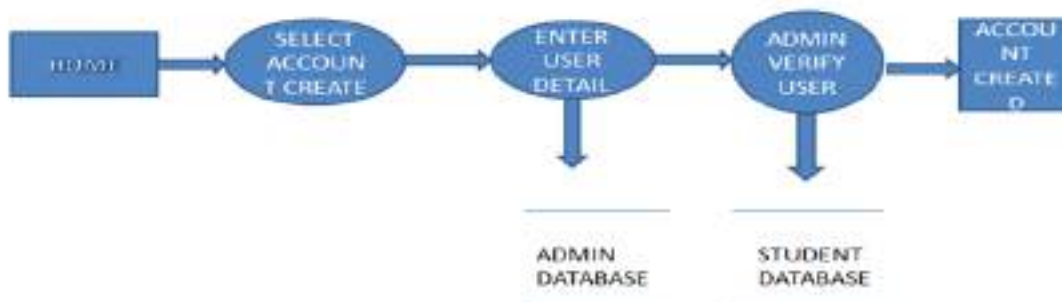
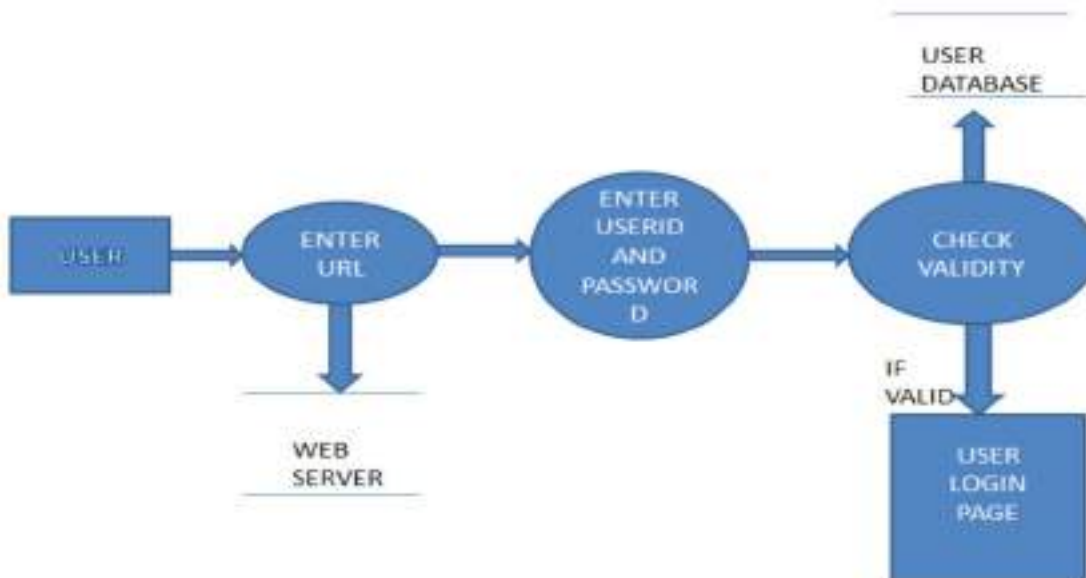
Levels in Data Flow Diagrams (DFD)

The DFD may be used to perform a system or software at any level of abstraction. Infact, DFDs may be partitioned into levels that represent increasing information flow and functional detail. Levels in DFD are numbered 0, 1, 2 or beyond. Here, we will see primarily three levels in the data flow diagram, which are: 0-level DFD, 1-level DFD, and 2-level DFD.



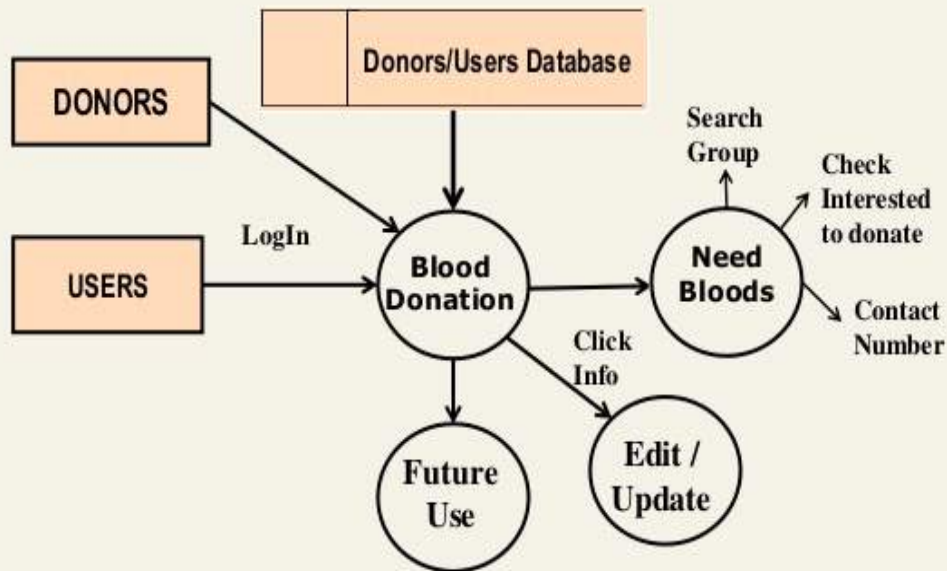


DFD for user login



Data Flow Diagram

Diagram2 [Blood Donation process]



ER-Diagram

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.

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



ER Diagram Symbols

Components of the ER Diagram

This model is based on three basic concepts:

- Entities
- Attributes
- Relationships

ER Diagram Examples

For example, in a University database, we might have entities for Students, Courses, and Lecturers. Students entity can have attributes like Rollno, Name, and DeptID. They might have relationships with Courses and Lecturers.

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:

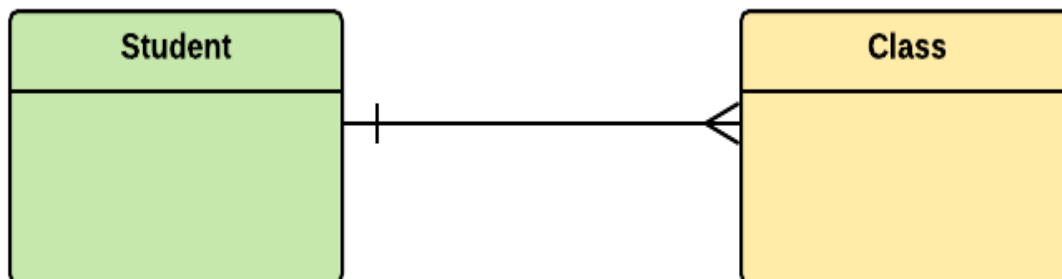
- Person: Employee, Student, Patient
- Place: Store, Building
- Object: Machine, product, and Car
- Event: Sale, Registration, Renewal
- Concept: Account, Course

Notation of an Entity

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.



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

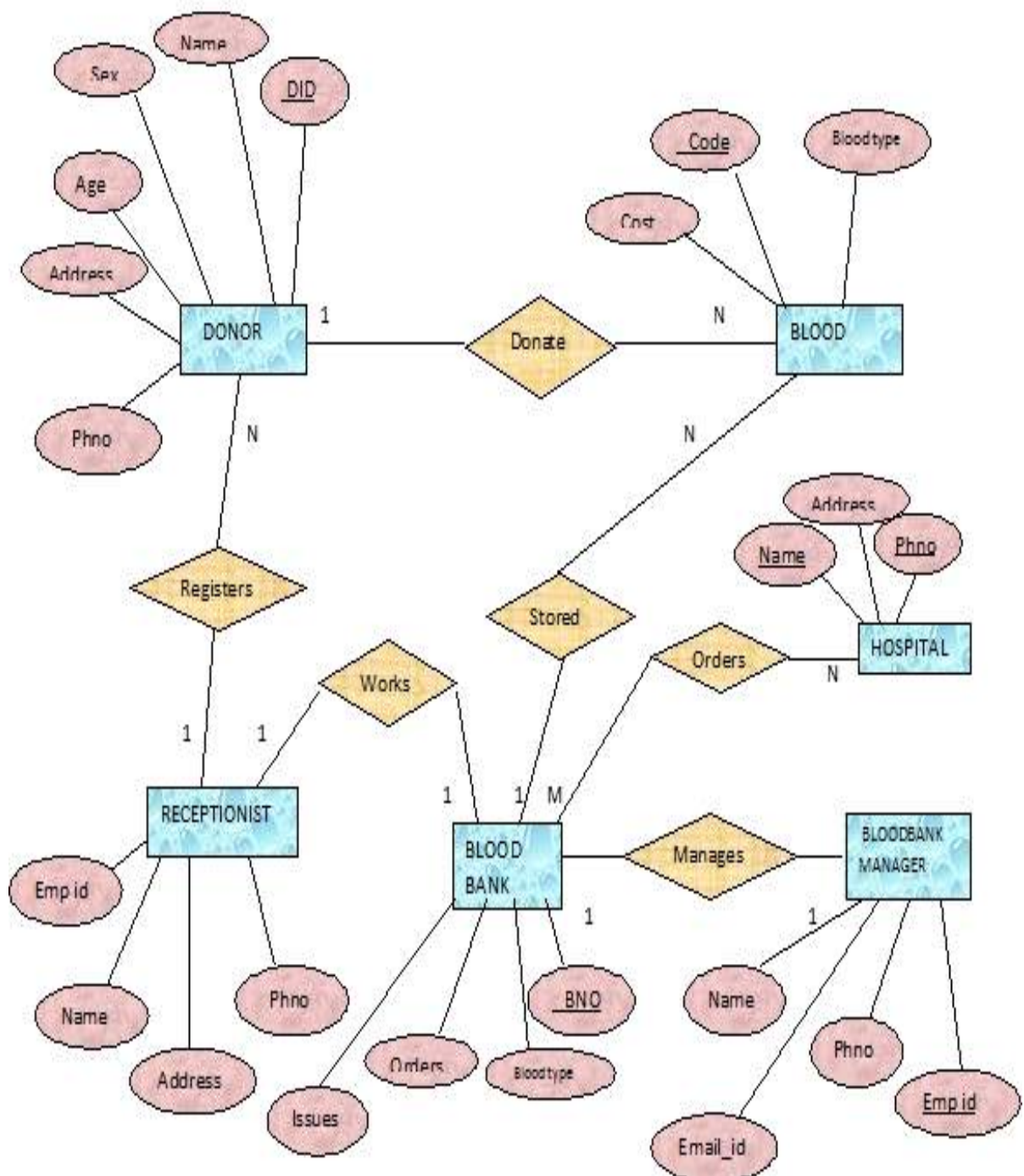
Relationship is nothing but an association among two or more entities. E.g., Tom works in the Chemistry department.

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

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.

THE COMPLETE ER- DIAGRAM



DATABASE

Database

Collected information which is in an organized form for easier access, management, and various updating is known as a database.

Before going into a further discussion of databases, we must have a prior knowledge of exactly what is a DATA? Data can be defined as a collection of facts and records on which we can apply reasoning or can-do discussion or some calculation. The data is always easily available and is in plenty. It can be used for processing some useful information from it. Also, it can be in redundant, can be irrelevant. Data can exist in form of graphics, reports, tables, text, etc. that represents every kind of information, that allows easy retrieval, updating, analysis, and output of data by systematically organized or structured repository of indexed information.

Containers having a huge amount of data are known as databases, for example, a public library stores books. Databases are computer structures that save, organize, protect, and deliver data. Any system that manages databases is called a **database management system**, or DBM. The typical diagram representation for a database is a cylinder. Inside a database, the data is recorded in a table which is a collection of rows, columns, and it is indexed so that to find relevant information becomes an easier task. As new information is added, data gets updated, expanded and deleted. The various processes of databases create and update themselves, querying the data they contain and running applications against it.

There are several different types of database models have been developed so far, for example, *flat*, *hierarchical*, *network* and *relational*. These models describe the operations that can be performed on them as well as the structure of the conforming databases. Normally there is a database schema which describes the exact model, entity types, and relationships among those entities.

Flat Databases have the following characteristics –

- simple
- long and dominant
- useful for very small scale and simple applications.

A **Relational Database** has the following characteristics –

- organizes data such that it appears to the user to be stored in a series of interrelated tables
- used for high-performance applications
- efficient

Types of Databases

Here are some popular types of databases.

Distributed databases:

A distributed database is a type of database that has contributions from the common database and information captured by local computers. In this type of database system, the data is not in one place and is distributed at various organizations.

Relational databases:

This type of database defines database relationships in the form of tables. It is also called Relational DBMS, which is the most popular DBMS type in the market. Database example of the RDBMS system include MySQL, Oracle, and Microsoft SQL Server database.

Object-oriented databases:

This type of computers database supports the storage of all data types. The data is stored in the form of objects. The objects to be held in the database have attributes and methods that define what to do with the data. PostgreSQL is an example of an object-oriented relational DBMS.

Centralized database:

It is a centralized location, and users from different backgrounds can access this data. This type of computers databases store application procedures that help users access the data even from a remote location.

Open-source databases:

This kind of database stored information related to operations. It is mainly used in the field of marketing, employee relations, customer service, of databases.

Cloud databases:

A cloud database is a database which is optimized or built for such a virtualized environment. There are so many advantages of a cloud database, some of which can pay for storage capacity and bandwidth. It also offers scalability on-demand, along with high availability.

Data warehouses:

Data Warehouse is to facilitate a single version of truth for a company for decision making and forecasting. A Data warehouse is an information system that contains

historical and commutative data from single or multiple sources. Data Warehouse concept simplifies the reporting and analysis process of the organization

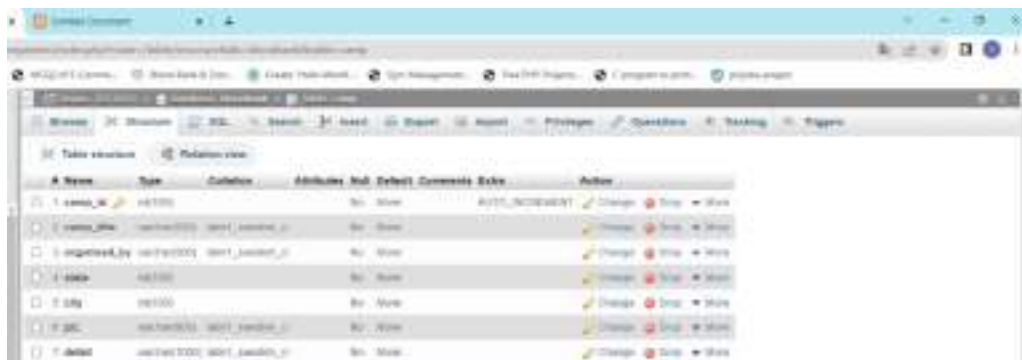
Advertisement



Blood group



Camp



city

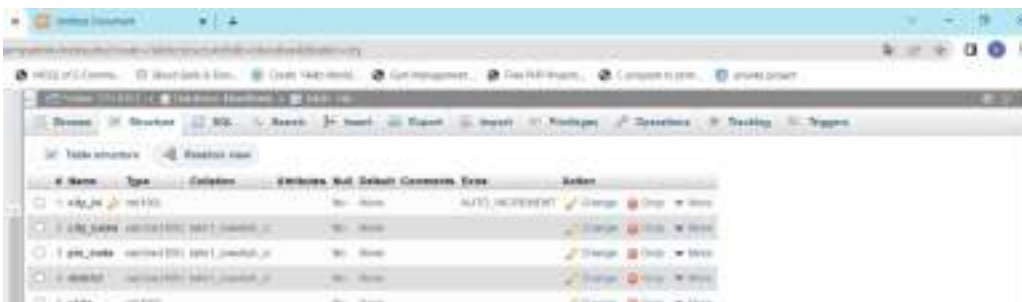
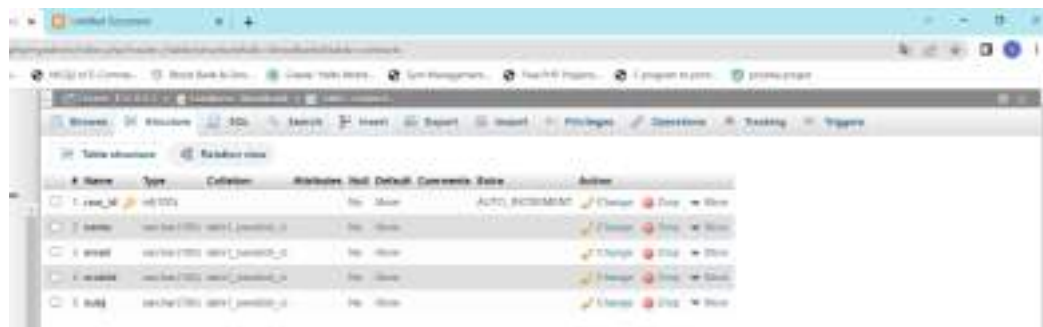


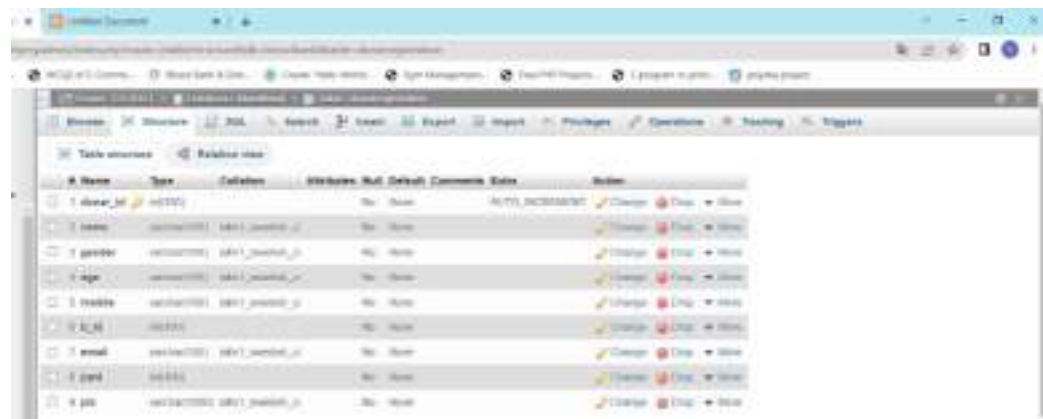
Table contract



The screenshot shows the 'Table contract' view for a table named 'contract'. The table has the following columns:

#	Name	Type	Collation	Nullable	Not Default	Comments	Size	Index
1	contract_id	int(4)		No	None			
2	contract_no	varchar(100)	utf8_unicode_ci	No	None			
3	contract_date	datetime	utf8_unicode_ci	No	None			
4	contract_status	varchar(100)	utf8_unicode_ci	No	None			
5	contract_desc	varchar(100)	utf8_unicode_ci	No	None			

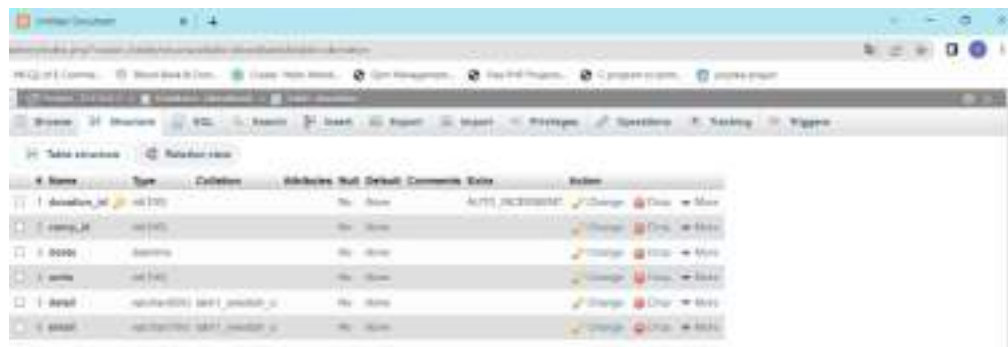
Donor registrations



The screenshot shows the 'Table contract' view for a table named 'donor_registrations'. The table has the following columns:

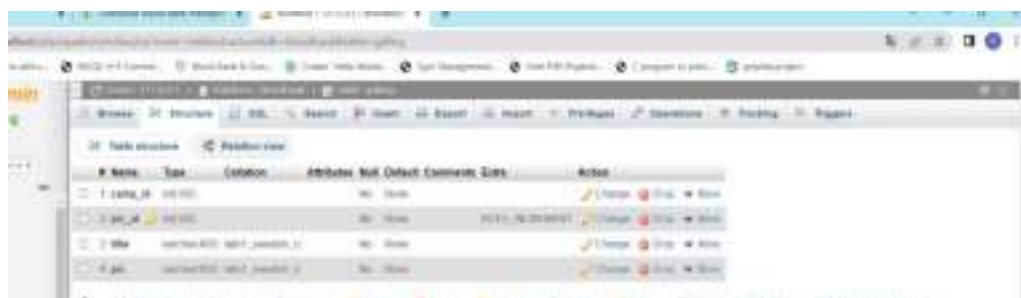
#	Name	Type	Collation	Nullable	Not Default	Comments	Size	Index
1	donor_id	int(4)		No	None			
2	donor_name	varchar(100)	utf8_unicode_ci	No	None			
3	donor_gender	varchar(100)	utf8_unicode_ci	No	None			
4	donor_age	int(4)		No	None			
5	donor_mobile	varchar(100)	utf8_unicode_ci	No	None			
6	donor_email	varchar(100)	utf8_unicode_ci	No	None			
7	donor_password	varchar(100)	utf8_unicode_ci	No	None			
8	donor_created_at	datetime	utf8_unicode_ci	No	None			

Done station



The screenshot shows the 'Table contract' view for a table named 'done_station'. The table has the following columns:

#	Name	Type	Collation	Nullable	Not Default	Comments	Size	Index
1	station_id	int(4)		No	None			
2	station_name	varchar(100)	utf8_unicode_ci	No	None			
3	station_address	varchar(100)	utf8_unicode_ci	No	None			
4	station_phone	varchar(100)	utf8_unicode_ci	No	None			
5	station_email	varchar(100)	utf8_unicode_ci	No	None			



The screenshot shows the 'Table contract' view for a table named 'done_station'. The table has the following columns:

#	Name	Type	Collation	Nullable	Not Default	Comments	Size	Index
1	station_id	int(4)		No	None			
2	station_name	varchar(100)	utf8_unicode_ci	No	None			
3	station_address	varchar(100)	utf8_unicode_ci	No	None			
4	station_phone	varchar(100)	utf8_unicode_ci	No	None			
5	station_email	varchar(100)	utf8_unicode_ci	No	None			

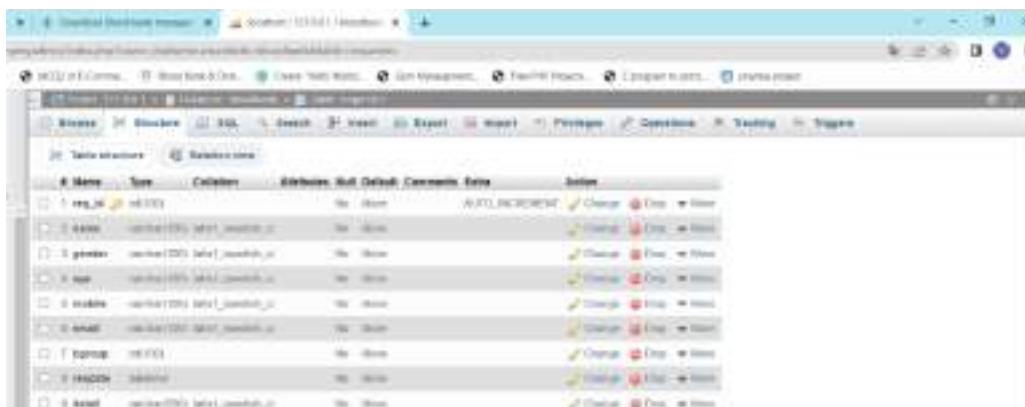
News



The screenshot shows the 'Tables' view of a SQL Server database. The table 'News' is selected, and its structure is displayed in a table format. The columns are: News_ID (int), Title (nvarchar(255)), Author (nvarchar(255)), and Content (text). The table has a primary key on News_ID and a foreign key relationship with the 'Author' column in the 'User' table.

Name	Type	Collation	Allows Null	Default	Comments	Extra	Index
News_ID	int		No	None		AUTO_INCREMENT	Change Drop View
Title	nvarchar(255)	Latin1_General_CI_AS	No	None			Change Drop View
Author	nvarchar(255)	Latin1_General_CI_AS	No	None			Change Drop View

Request



The screenshot shows the 'Tables' view of a SQL Server database. The table 'Request' is selected, and its structure is displayed in a table format. The columns are: Request_ID (int), Title (nvarchar(255)), Author (nvarchar(255)), and Content (text). The table has a primary key on Request_ID and a foreign key relationship with the 'Author' column in the 'User' table.

Name	Type	Collation	Allows Null	Default	Comments	Extra	Index
Request_ID	int		No	None		AUTO_INCREMENT	Change Drop View
Title	nvarchar(255)	Latin1_General_CI_AS	No	None			Change Drop View
Author	nvarchar(255)	Latin1_General_CI_AS	No	None			Change Drop View
Content	text		No	None			Change Drop View

State



The screenshot shows the 'Tables' view of a SQL Server database. The table 'State' is selected, and its structure is displayed in a table format. The columns are: State_ID (int), Title (nvarchar(255)), and Content (text). The table has a primary key on State_ID and a foreign key relationship with the 'Author' column in the 'User' table.

Name	Type	Collation	Allows Null	Default	Comments	Extra	Index
State_ID	int		No	None		AUTO_INCREMENT	Change Drop View
Title	nvarchar(255)	Latin1_General_CI_AS	No	None			Change Drop View
Content	text		No	None			Change Drop View

User



The screenshot shows the 'Tables' view of a SQL Server database. The table 'User' is selected, and its structure is displayed in a table format. The columns are: User_ID (int), Username (nvarchar(255)), and Password (nvarchar(255)). The table has a primary key on User_ID.

Name	Type	Collation	Allows Null	Default	Comments	Extra	Index
User_ID	int		No	None			Change Drop View
Username	nvarchar(255)	Latin1_General_CI_AS	No	None			Change Drop View
Password	nvarchar(255)	Latin1_General_CI_AS	No	None			Change Drop View

TESTING

Testing

Software testing is a procedure of implementing software or the application to identify the defects or bugs. For testing an application or software, we need to follow some principles to make our product defects free, and that also helps the test engineers to test the software with their effort and time. Here, in this section, we are going to learn about the seven essential principles of software testing.

Let us see the seven different testing principles, one by one:

- Testing shows the presence of defects
- Exhaustive Testing is not possible
- Early Testing
- Defect Clustering
- Pesticide Paradox
- Testing is context-dependent
- Absence of errors fallacy

Testing shows the presence of defects

The test engineer will test the application to make sure that the application is bug or defects free. While doing testing, we can only identify that the application or software has any errors. The primary purpose of doing testing is to identify the numbers of unknown bugs with the help of various methods and testing techniques because the entire test should be traceable to the customer requirement, which means that to find any defects that might cause the product failure to meet the client's needs.

Exhaustive Testing is not possible

Sometimes it seems to be very hard to test all the modules and their features with effective and non-effective combinations of the inputs data throughout the actual testing process.

Hence, instead of performing the exhaustive testing as it takes boundless determinations and most of the hard work is unsuccessful. So we can complete this type of variations according to the importance of the modules because the product timelines will not permit us to perform such type of testing scenarios.

Early Testing

Here early testing means that all the testing activities should start in the early stages of

the software development life cycle's **requirement analysis stage** to identify the defects because if we find the bugs at an early stage, it will be fixed in the initial stage itself, which may cost us very less as compared to those which are identified in the future phase of the testing process.

To perform testing, we will require the requirement specification documents; therefore, if the requirements are defined incorrectly, then it can be fixed directly rather than fixing them in another stage, which could be the development phase.

Defect clustering

The defect clustering defined that throughout the testing process, we can detect the numbers of bugs which are correlated to a small number of modules. We have various reasons for this, such as the modules could be complicated; the coding part may be complex, and so on.

These types of software or the application will follow the **Pareto Principle**, which states that we can identify that approx. Eighty percent of the complication is present in 20 percent of the modules. With the help of this, we can find the uncertain modules, but this method has its difficulties if the same tests are performing regularly, hence the same test will not able to identify the new defects.

Pesticide paradox

This principle defined that if we are executing the same set of test cases again and again over a particular time, then these kinds of the test will not be able to find the new bugs in the software or the application. To get over these pesticide paradoxes, it is very significant to review all the test cases frequently. And the new and different tests are necessary to be written for the implementation of multiple parts of the application or the software, which helps us to find more bugs.

Testing is context-dependent

Testing is a context-dependent principle states that we have multiple fields such as e-commerce websites, commercial websites, and so on are available in the market. There is a definite way to test the commercial site as well as the e-commerce websites because every application has its own needs, features, and functionality. To check this type of application, we will take the help of various kinds of testing, different technique, approaches, and multiple methods. Therefore, the testing depends on the context of the application.

Absence of errors fallacy

Once the application is completely tested and there are no bugs identified before the release, so we can say that the application is 99 percent bug-free. But there is the

chance when the application is tested beside the incorrect requirements, identified the flaws, and fixed them on a given period would not help as testing is done on the wrong specification, which does not apply to the client's requirements. The absence of error fallacy means identifying and fixing the bugs would not help if the application is impractical and not able to accomplish the client's requirements and needs.

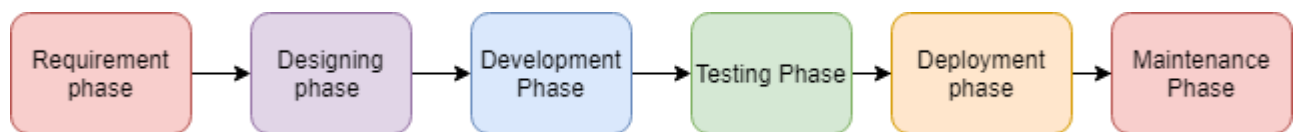
Software Development Life Cycle (SDLC)

SDLC is a process that creates a structure of development of software. There are different phases within SDLC, and each phase has its various activities. It makes the development team able to design, create, and deliver a high-quality product.

SDLC describes various phases of software development and the order of execution of phases. Each phase requires deliverable from the previous phase in a life cycle of software development. Requirements are translated into design, design into development and development into testing; after testing, it is given to the client.

Let's see all the phases in detail:

Different phases of the software development cycle



- Requirement Phase
- Design Phase
- Build /Development Phase
- Testing Phase
- Deployment/ Deliver Phase
- Maintenance

1. Requirement Phase

This is the most crucial phase of the software development life cycle for the developing team as well as for the project manager. During this phase, the client states requirements, specifications, expectations, and any other special requirement related to the product or software. All these are gathered by the business manager or project manager or analyst of the service providing company.

The requirement includes how the product will be used and who will use the product to determine the load of operations. All information gathered from this phase is critical to developing the product as per the customer requirements.

2. Design Phase

The design phase includes a detailed analysis of new software according to the requirement phase. This is the high priority phase in the development life cycle of a system because the logical designing of the system is converted into physical designing. The output of the requirement phase is a collection of things that are required, and the design phase gives the way to accomplish these requirements. The decision of all required essential tools such as programming language like Java, .NET, PHP, a database like Oracle, MySQL, a combination of hardware and software to provide a platform on which software can run without any problem is taken in this phase.

There are several techniques and tools, such as data flow diagrams, flowcharts, decision tables, and decision trees, Data dictionary, and the structured dictionary are used for describing the system design.

3. Build /Development Phase

After the successful completion of the requirement and design phase, the next step is to implement the design into the development of a software system. In this phase, work is divided into small units, and coding starts by the team of developers according to the design discussed in the previous phase and according to the requirements of the client discussed in requirement phase to produce the desired result.

Front-end developers develop easy and attractive GUI and necessary interfaces to interact with back-end operations and back-end developers do back-end coding according to the required operations. All is done according to the procedure and guidelines demonstrated by the project manager. Since this is the coding phase, it takes the longest time and more focused approach for the developer in the software development life cycle.

4. Testing Phase

Testing is the last step of completing a software system. In this phase, after getting the developed GUI and back-end combination, it is tested against the requirements stated in the requirement phase. Testing determines whether the software is actually giving the result as per the requirements addressed in the requirement phase or not. The Development team makes a test plan to start the test. This test plan includes all types of essential testing such as integration testing, unit testing, acceptance testing, and system testing. Non-functional testing is also done in this phase.

If there are any defects in the software or it is not working as per expectations, then the testing team gives information to the development team in detail about the issue. If it is a valid defect or worth to sort out, it will be fixed, and the development team replaces it

with the new one, and it also needs to be verified.

5. Deployment/ Deliver Phase

When software testing is completed with a satisfying result, and there are no remaining issues in the working of the software, it is delivered to the customer for their use. As soon as customers receive the product, they are recommended first to do the beta testing. In beta testing, customer can require any changes which are not present in the software but mentioned in the requirement document or any other GUI changes to make it more user-friendly. Besides this, if any type of defect is encountered while a customer using the software; it will be informed to the development team of that particular software to sort out the problem. If it is a severe issue, then the development team solves it in a short time; otherwise, if it is less severe, then it will wait for the next version. After the solution of all types of bugs and changes, the software finally deployed to the end-user.

6. Maintenance

The maintenance phase is the last and long-lasting phase of SDLC because it is the process which continues until the software's life cycle comes to an end. When a customer starts using software, then actual problems start to occur, and at that time there's a need to solve these problems. This phase also includes making changes in hardware and software to maintain its operational effectiveness like to improve its performance, enhance security features and according to customer's requirements with upcoming time. This process to take care of product time to time is called maintenance.

White Box Testing

The box testing approach of software testing consists of black box testing and white box testing. We are discussing here white box testing which also known as glass box is **testing, structural testing, clear box testing, open box testing and transparent box testing**. It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs. It is based on inner workings of an application and revolves around internal structure testing. In this type of testing programming skills are required to design test cases. The primary goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.

Black box testing

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer.

In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed. The test team reports the result to the development team and then tests the next function. After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.



Generic steps of black box testing

- The black box test is based on the specification of requirements, so it is examined in the beginning.
- In the second step, the tester creates a positive test scenario and an adverse test scenario by selecting valid and invalid input values to check that the software is processing them correctly or incorrectly.
- In the third step, the tester develops various test cases such as decision table, all pairs test, equivalent division, error estimation, cause-effect graph, etc.
- The fourth phase includes the execution of all test cases.
- In the fifth step, the tester compares the expected output against the actual output.
- In the sixth and final step, if there is any flaw in the software, then it is cured and tested again.

INPUT SCREEN

Font page



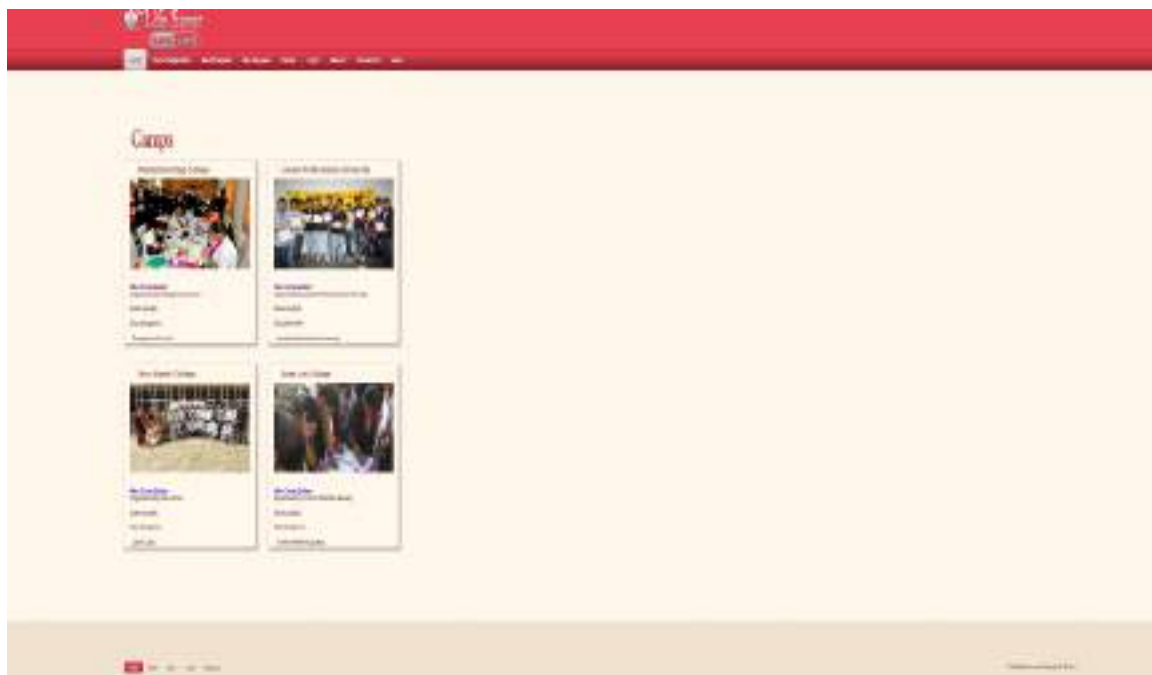
Registration



Request blood

The screenshot shows the 'Life Saver' website header with a red background and the text 'Life Saver GIVE LIFE'. Below the header is a navigation menu with links: Home, Donate Registration, Send Request, View Request, Camps, Login, Search, Contact Us, and About. The main content area features a central white box titled 'Requests For Blood' with a red ribbon icon. The form contains the following fields: Name (text input), Gender (radio buttons for Male and Female), Age (text input), Mobile No. (text input), Select Blood Group (dropdown menu), E-Mail (text input), To Request Date (calendar input), and Detail (text area). A red 'Submit' button is located at the bottom of the form. The footer of the page is light beige and contains a navigation menu with Home, Search, Login, About, and Contact Us, along with the copyright notice '© All Rights Reserved | Design by M. Shetty'.

Camp



Donor Login



Search



Contract



Donor panel



Update Profile

Update Profile



Name:

Gender: Male Female

Age:

Mobile No:

Blood donated

Blood Donated

Select camp:

Date: / /

No. Of Units:

Other Detail:

Request for blood



The screenshot shows the 'Life Saver' website interface. The header includes the logo and navigation links: Home, Blood Registration, Send Request, View Request, Contact, Log In, Search, Contact Us, and About. The main content area is titled 'Requests For Blood' and displays a table with the following data:

Blood Group	Name	Gender	Contact No.	Mobile No.	Email	Till Required Date
7	Arifuddin Bangsi	Male	22	0427420208	isalewan11@gmail.com	19
8	Jayaban	Female	20	0016291264	pinwan69@gmail.com	13
9	Narash	Female	21	0427420201	rona@fwd713@gmail.com	17
10	Taksa	Male	08	168025608	www@yahoo.com	14

The footer contains a 'Home' button, navigation links, and a copyright notice: '© All Rights Reserved'.

Result



The screenshot shows the 'Life Saver' website interface displaying the 'Results' section. The header and navigation menu are identical to the previous screenshot. The main content area is titled 'Results' and displays two donor profiles:

- Donor 1:**
 - Name: Pinanggiel Raka
 - Gender: female
 - Mobile No: 0295750630
 - Email: rka@yaho.com
 - Blood Group: AB+
- Donor 2:**
 - Name: Isdip Danga
 - Gender: male
 - Mobile No: 9070907543
 - Email: hbang@yahoo.com
 - Blood Group: AB+

The footer contains a 'Home' button, navigation links, and a copyright notice: '© All Rights Reserved (Design by Mr. Shweta)'.

CONCLUSION

Conclusion

With the theoretical inclination of our syllabus it becomes very essential to take the utmost advantage of any opportunity of gaining practical experience that comes along. The building blocks of this Major Project "BLOOD BANK Management System" was one of these opportunities. It gave us the requisite practical knowledge to supplement the already taught theoretical concepts thus making us more competent as a computer engineer. The project from a personal point of view also helped us in understanding the following aspects of project development:

- The planning that goes into implementing a project.
- The importance of proper planning and an organized methodology.
- The key element of team spirit and co-ordination in a successful project.

The project also provided us the opportunity of interacting with our teachers and to gain from their best experience

Future Recommendation

BLOOD BANK MANAGEMENT is a software application to built such a way that it should suits for all type of blood banks in **future**.

One important future scope is availability of location based blood bank details and extraction of location based donor's detail, which is very helpful to the acceptant people. All the time the network facilities cannot be use. This time donor request does not reach in proper time, this can be avoid through adding some message sending procedure this will help to find proper blood donor in time. This will provide availability of blood in time.

BIBLIOGRAPHY

Bibliography

PHP Manual www.php.net/

<https://www.google.com>

<http://www.w3schools.com>

<http://www.indianbloodgroup.com>