

1. Introduction (BLOOD & OXYGEN BANK MANAGEMENT SYSTEM):

The **Blood & Oxygen Bank Management System** is great project. This project is designed for successful completion of project on blood & oxygen bank management system. The basic building aim is to provide blood donation service to the city recently. Blood & Oxygen Bank Management System is a window-based system that is designed to store, process, retrieve and analyze information concerned with the administrative and inventory management within a blood & oxygen bank. This project aims at maintaining all the information pertaining to , different blood groups available in each blood & oxygen bank and helps them manage in a better way. Aim is to provide transparency in this field, make the process of obtaining blood & oxygen from a blood & oxygen bank hassle free and corruption free and make the system of blood & oxygen bank management effective.

The supplemental of the oxygen therapy helps the people with COPD, COVID-19, emphysema, sleep apnea and other breathing problems get enough oxygen to function and stay well low blood oxygen levels (hypoxemia) can damage organs and be life-threatening you may need oxygen therapy for life or temporarily. The **Blood & Oxygen Bank System Project** contain modules that's information are as follows:-

- ❖ **Registration**
- ❖ **Login**
- ❖ **MENU BAR**
 - **Donor**
 - **Stock Details**
 - **Blood Donation**
 - **Exit**
- ❖ **DONOR**
 - **Add**
 - **Update**
 - **Donor Report**
 - **Donor Details**
- ❖ **STOCK DETAILS**
 - **Blood**
 - **Oxygen**
- ❖ **BLOOD DONATION**
 - **Blood**
 - **Plasma**
 - **Platelet**
- ❖ **EXIT**

2. Objective:

The main objective of this application is to automate the complete operations of the blood bank. They need maintain hundreds of thousands of records. Also searching should be very faster so they can find required details instantly.

To develop a web-based portal to facilitate the co-ordination between supply and demand of blood. This system makes conveniently available good quality, safe blood and other blood components, which can be provided in a sound, ethical and acceptable manner, consistent with the long-term wellbeing of the community. It actively encourage voluntary blood donation, motivate and maintain a well-indexed record of blood donors and educate the community on the benefits of blood donation. This will also serve as the site for interaction of best practices in reducing unnecessary utilization of blood and help the state work more efficiently towards self-sufficiency in blood.

The system will provide the user the option to look at the details of the existing Donor List, Blood Group and to add a new Donor. It also allows the user to modify the record. The administrator can alter all the system data.

2.0 Need of Blood & Oxygen Bank Management System:

Bank blood & oxygen 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 & Oxygen Bank Management System”. For do all those kinds of need Blood & Oxygen Bank Management System project in java contain modules which are include the detail of following areas:

2.1 Abstract of Blood & Oxygen Bank Management System:

Help Line is a voluntary and non-governmental organization. It maintains library of blood donors in India. Sometimes Doctors and Blood & Oxygen 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 & Oxygen 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 & Oxygen Bank Management System (BOBMS) is a window based system that is designed to store, process, retrieve and analyze information concerned with the administrative and inventory management within a blood & Oxygen 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. 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 & Oxygen bank management effective.

2.2 Benefits:

❖ Our Vision:

In the IT a, there are almost not any fields exist where computers are not used. Tech shot would like to contribute to the total **SATISFACTION** to its esteemed **CUSTOMERS** by providing them with the high quality products.

Tech shot wants to make products highly reliable, affordable & consistent which will serve the customer domain. Tech shot concerned for its customers & serves them in precise time, with right product to fright quality. By enhancing consulting and other potentials we help move customers forward in each & every part of their businesses, from strategic planning to day-to-day operations.

Our Clients benefit from access to information solutions that help them better cope-up their business, cooperate with customers and make financial and operational decisions.

❖ Our Mission:

To endow with strategic and technical expertise to companies wanting to leverage the latest innovations. Our mission is to Define Quality Policy for the IT era, set new span for Services to customers.

3. System Analysis:

❖ 3.1 Preliminary Investigation:

First in the system development process is preliminary Investigation. Preliminary Investigation is conducted in the following phases.

- ❖ Project clarification
- ❖ Feasibility study
- ❖ Project appraisal

Project clarification is the process of selecting a project request for further study. When a system development or modification request is made, the first systems activity, the preliminary investigation, begins the activity has three parts: Request clarification, feasibility study and project appraisal. Many request from employees and users in organization are not clearly stated.

Therefore before any systems investigation can be considered, the project request must be examined to determine precisely what the originator wants. This is called Request clarification. As important outcome of the preliminary investigation is the determination that the system request is feasible.

4. Feasibility study:

The feasibility study is performed to determine whether the proposed system is viable considering the Technical, Operational and Economical factors. After going through feasibility study we can have a clear-cut view of the system's benefits and drawbacks.

- **Technical Feasibility:**

The proposed system is developed using Active Server Page, VB Script and HTML as front-end tool and Oracle 8 as the back end. The proposed system needs a Personal Web Server to serve the requests submitted by the users. The Web browser is used to view the web page that is available within the Windows operating system itself. The proposed system will run under Win9x, NT, and win2000 environment. As Windows is very user friendly and GUI OS it is very easy to use. All the required hardware and software are readily available in the market. Hence the system is technically feasible.

- **Operational Feasibility:**

- The proposed system is operationally feasible because of the following reasons.
- The customer is benefited more as most of his time is saved.
- The customer is serviced at his place of work.
- The cost of the proposed system is almost negligible when compared to the benefits gained.

- **Economic Feasibility:**

As the necessary hardware and software are available in the market at a low cost, the initial investment is the only cost incurred and does not need any further enhancements. Hence it is economically feasible.

5. Front End:

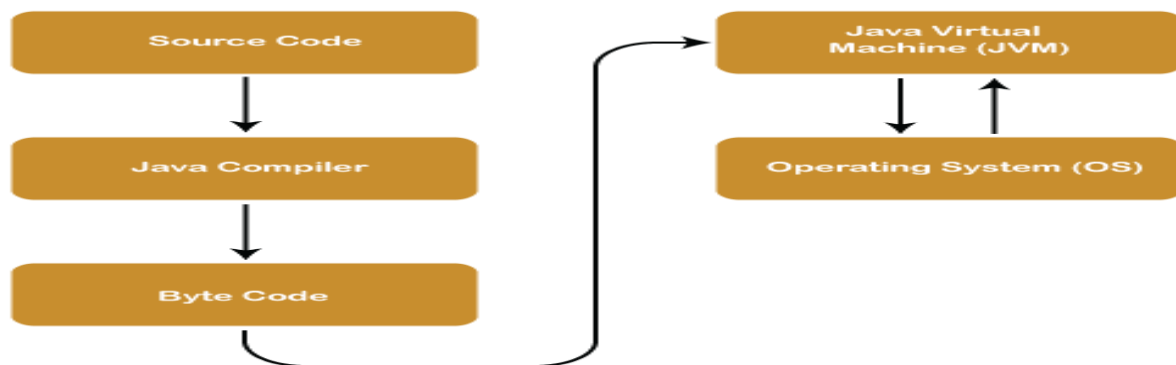
The front end is an interface between the user and the back end. The front and back ends may be distributed amongst one or more systems.

In network computing, front end can refer to any hardware that optimizes or protects network traffic called application front-end hardware because it is placed on the network's outward-facing front end or boundary. Network traffic passes through the front-end hardware before entering the network.

In compilers, the frontend translates a computer programming source code into an intermediate representation, and the backend works with the intermediate representation to produce code in a computer output language. The back end usually optimizes to produce code that runs faster. The front-end/back-end distinction can separate the parser section that deals with source code and the backend that generates code and optimizes.

These days, front-end development refers to the part of the web users interact with. In the past, web development consisted of people who worked with JAVA & JAVA SWING.

Most of everything you see on any window based is a mixture java & java swing which are all controlled by the wamp server & oracle. For example, if you're using Google Chrome or Firefox, the browser is what translates all of the code in a manner for you to see and with which to interact, such as fonts, colors, drop-down menus, sliders, forms, etc. In order for all of this to work, though, there has to be something to support the front-end; this is where the backend comes into play.



5.1 Java:

Java Architecture is a collection of components, i.e., **JVM**, **JRE**, and **JDK**. It integrates the process of interpretation and compilation. It defines all the processes involved in creating a Java program. **Java Architecture** explains each and every step of how a program is compiled and executed. **Java Architecture** can be explained by using the following steps:

- There is a process of compilation and interpretation in Java.
- Java compiler converts the Java code into byte code.
- After that, the JVM converts the byte code into machine code.
- The machine code is then executed by the machine.

The following figure represents the **Java Architecture** in which each step is elaborate graphically. Now let's dive deep to get more knowledge about **Java Architecture**. As we know that the Java architecture is a collection of components, so we will discuss each and every component into detail.

Components of Java Architecture:

The Java architecture includes the three main components:

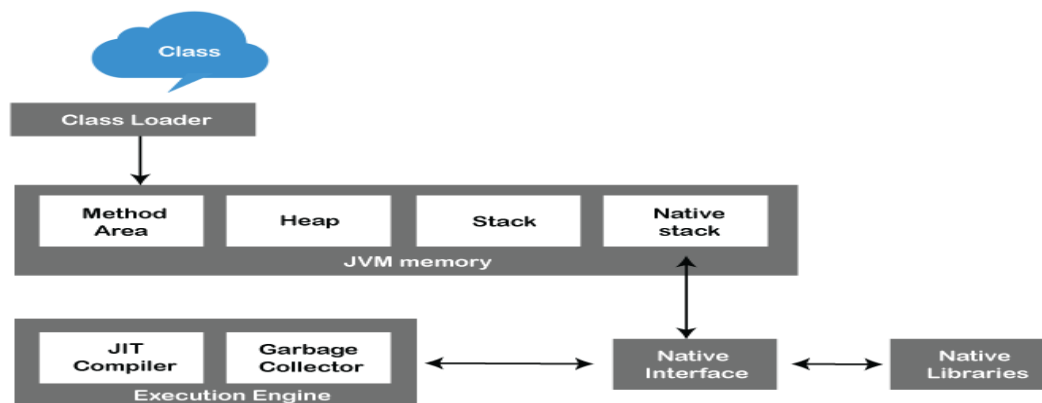
- Java Virtual Machine (JVM)
- Java Runtime Environment (JRE)
- Java Development Kit (JDK)

Java Virtual Machine:

The main feature of Java is **WORA**. WORA stands for **Write Once Run Anywhere**. The feature states that we can write our code once and use it anywhere or on any operating system. Our Java program can run any of the platforms only because of the Java Virtual Machine. It is a Java platform component that gives us an environment to execute java programs. JVM's main task is to convert byte code into machine code.

JVM, first of all, loads the code into memory and verifies it. After that, it executes the code and provides a runtime environment. Java Virtual Machine (JVM) has its own architecture, which is given below:

JVM Architecture JVM is an abstract machine that provides the environment in which Java byte code is executed. The falling figure represents the architecture of the JVM.



Class Loader: Class Loader is a subsystem used to load class files. Class Loader first loads the Java code whenever we run it.

Class Method Area: In the memory, there is an area where the class data is stored during the code's execution. Class method area holds the information of static variables, static methods, static blocks, and instance methods.

Heap: The heap area is a part of the JVM memory and is created when the JVM starts up. Its size cannot be static because it increase or decrease during the application runs.

Stack: It is also referred to as thread stack. It is created for a single execution thread. The thread uses this area to store the elements like the partial result, local variable, data used for calling method and returns etc.

Native Stack: It contains the information of all the native methods used in our application.

Execution Engine: It is the central part of the JVM. Its main task is to execute the byte code and execute the Java classes. The execution engine has three main components used for executing Java classes.

Interpreter: It converts the byte code into native code and executes. It sequentially executes the code. The interpreter interprets continuously and even the same method multiple times. This reduces the performance of the system, and to solve this, the JIT compiler is introduced.

JIT Compiler: JIT compiler is introduced to remove the drawback of the interpreter. It increases the speed of execution and improves performance.

Garbage Collector: The garbage collector is used to manage the memory, and it is a program written in Java. It works in two phases, i.e., **Mark** and **Sweep**. Mark is an area where the garbage collector identifies the used and unused chunks of memory. The Sweep removes the identified object from the **Mark**.

Java Native Interface: Java Native Interface works as a mediator between Java method calls and native libraries.

Java Runtime Environment: It provides an environment in which Java programs are executed. JRE takes our Java code, integrates it with the required libraries, and then starts the JVM to execute it.

Java Development Kit: It is a software development environment used in the development of Java applications and applets. Java Development Kit holds JRE, a compiler, an interpreter or loader, and several development tools in it.

JAVA was developed by James Gosling at **Sun Microsystems Inc.** in the year **1995**, later acquired by Oracle Corporation. It is a simple programming language. Java makes writing, compiling, and debugging programming easy. It helps to create reusable code and modular programs. Java is a class-based, object-oriented programming language and is designed to have as few implementation dependencies as possible. A general-purpose programming language made for developers to write once run anywhere that is compiled Java code can run on all platforms that support Java. Java applications are compiled to byte code that can run on any Java Virtual Machine. The syntax of Java is similar to c/c++.

History:

Java's history is very interesting. It is a programming language created in 1991. James Gosling, Mike Sheridan, and Patrick Naughton, a team of Sun engineers known as the **Green team** initiated the Java language in 1991. **Sun Microsystems** released its first public implementation in 1996 as **Java 1.0**. It provides no-cost run-times on popular platforms. Java 1.0 compiler was re-written in Java by Arthur Van Hoff to strictly comply with its specifications. With the arrival of Java 2, new versions had multiple configurations built for different types of platforms.

In 1997, Sun Microsystems approached the ISO standards body and later formalized Java, but it soon withdrew from the process. At one time, Sun made most of its Java implementations available without charge, despite their proprietary software status. Sun generated revenue from Java through the selling of licenses for specialized products such as the Java Enterprise System.

On November 13, 2006, Sun released much of its Java virtual machine as free, open-source software. On May 8, 2007, Sun finished the process, making all of its JVM's core code available under open-source distribution terms.

5.1.2 Reasons for using JAVA:

Learning JAVA is easy:

Java is easy to learn. Java was designed to be easy to use and is therefore easy to write, compile, debug, and learn than other programming languages. Java is object-oriented. This allows you to create modular programs and reusable code.

It's Performance:

The performance of a Java byte code compiled Java program depends on how optimally its given tasks are managed by the host Java virtual machine (JVM), and how well the JVM exploits the features of the computer hardware and operating system (OS) in doing so.

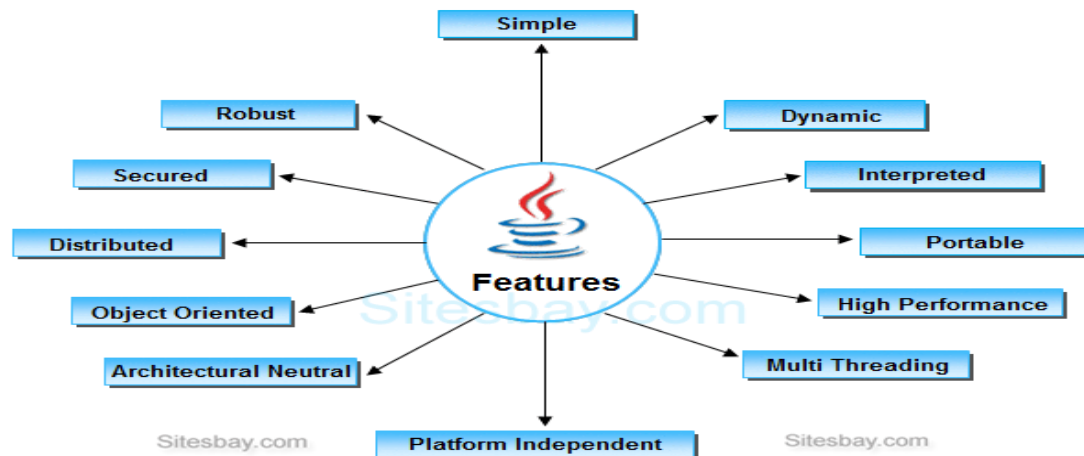
The low cost:

Given a cost matrix `cost [][]` and a position `(m, n)` in `cost[][]`, write a function that returns cost of minimum cost path to reach `(m, n)` from `(0, 0)`. Each cell of the matrix represents a cost to traverse through that cell.

It's Open Source, We can modify it:

The term open source refers to any program whose source code is made available for use or modification as users or other developers see fit. Unlike proprietary software, open source software is computer software that is developed as a public, open collaboration and made freely available to the public.

Features of Java:

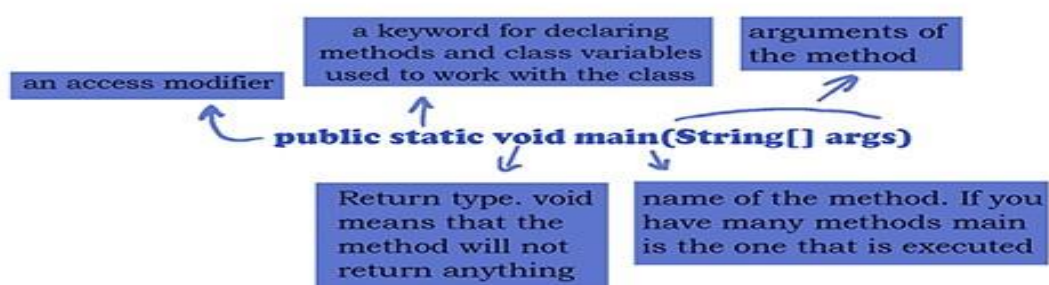


JAVA Syntax:

In Java, there are 51 keywords. Some of the keywords in Java are: abstract, Boolean , byte, char, long, short, int , finally, extends, implements, interface, package, new, while, for, do, break, continue, return, public, private, this, case, final, default, etc.



Basic JAVA Syntax: Java Syntax is a basic of the language, all the main rules, commands, constructions to write programs that the compiler and computer “understands”. Every programming language has its syntax as well as human language.



5.1.4 Java Swing:

Swing in Java is a lightweight GUI toolkit which has a wide variety of widgets for building optimized window based applications. It is a part of the JFC (Java Foundation Classes). It is built on top of the AWT API and entirely written in java. It is platform independent unlike AWT and has lightweight components.

It becomes easier to build applications since we already have GUI components like button, checkbox etc. This is helpful because we do not have to start from the scratch.

Container Class

Any **class** which has other components in it is called as a container class. For building GUI applications at least one container class is necessary.

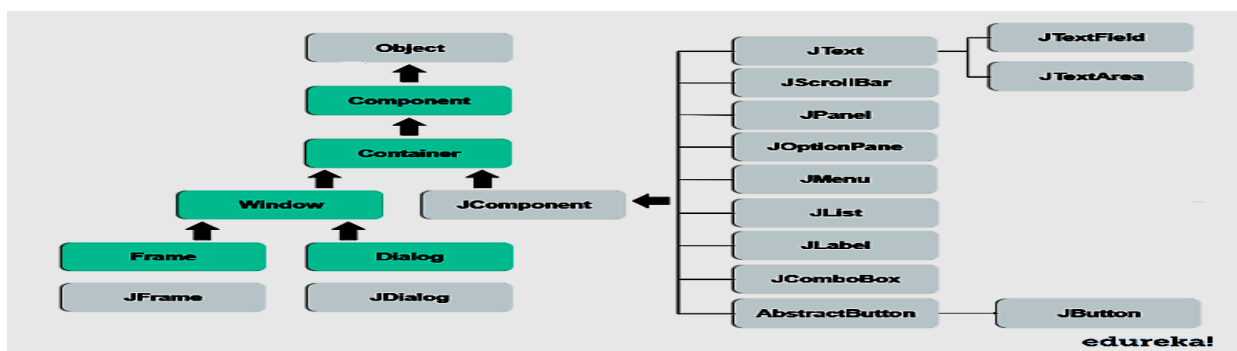
Following are the three types of container classes:

- Panel – It is used to organize components on to a window
- Frame – A fully functioning window with icons and titles
- Dialog – It is like a pop up window but not fully functional like the frame

Difference between AWT and Swing

<u>AWT</u>	<u>SWING</u>
Platform Dependent	Platform Independent
Does not follow MVC	Follows MVC
Lesser Components	More powerful components
Does not support pluggable look and feel	Supports pluggable look and feel
Heavyweight	Lightweight

Java Swing Class Hierarchy:



6. Back End:

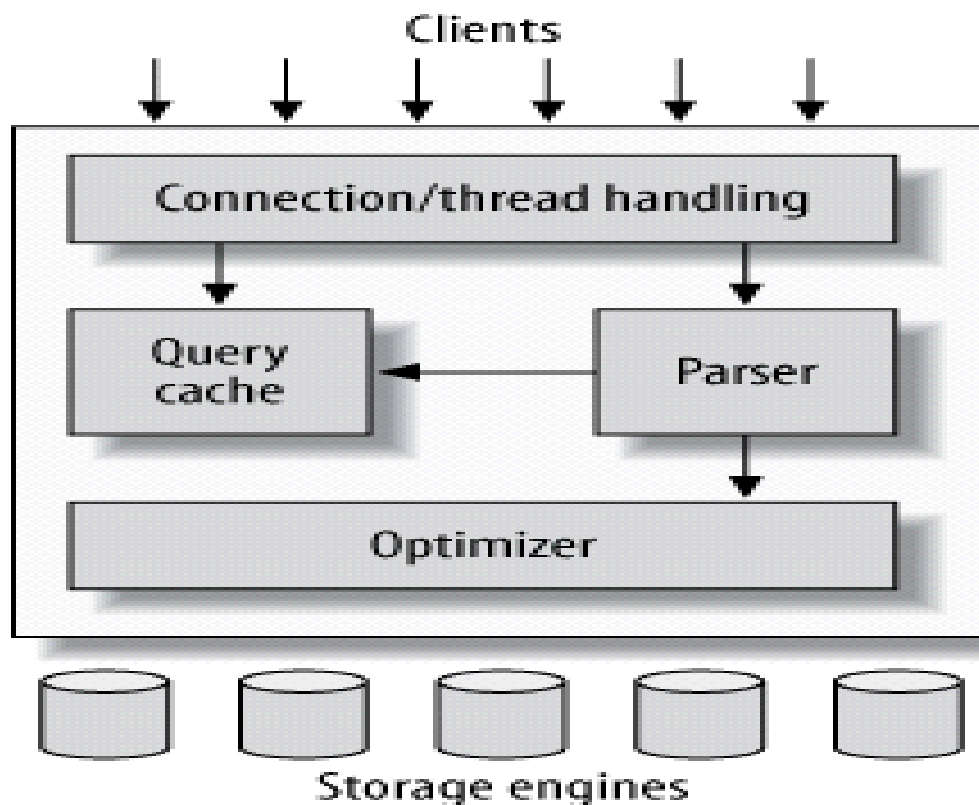
In a previous blog, we talked about how web programmers are concerned with launching websites, updates, and maintenance, among other things. All of that works to support the front-end of the website. The back-end has three parts to it: server, application, and database.

To better explain how all of this works, let's use the example of a customer trying to purchase a plane ticket using a website. Everything that the customer sees on the webpage is the front-end, as we have explained before, but when that customer enters all of his or her information, such as their name, billing address, destination, etc, the web application stores the information in a database that was created previously on the server in which the website is calling for information.

The web application creates, deletes, changes, renames, etc items in the database. For example, when a customer purchases a ticket, that creates an item in the database, but when they have a change in their order or they wish to cancel, the item in the database is changed. In short, when a customer wants to buy a ticket, the backend operation is the web application communicating with the server to make a change in a database stored on said server. Technologies like PHP, Ruby, Python, and others are the ones backend programmers use to make this communication work smoothly, allowing the customer to purchase his or her ticket with ease.

6.1 My SQL's Logical Architecture:

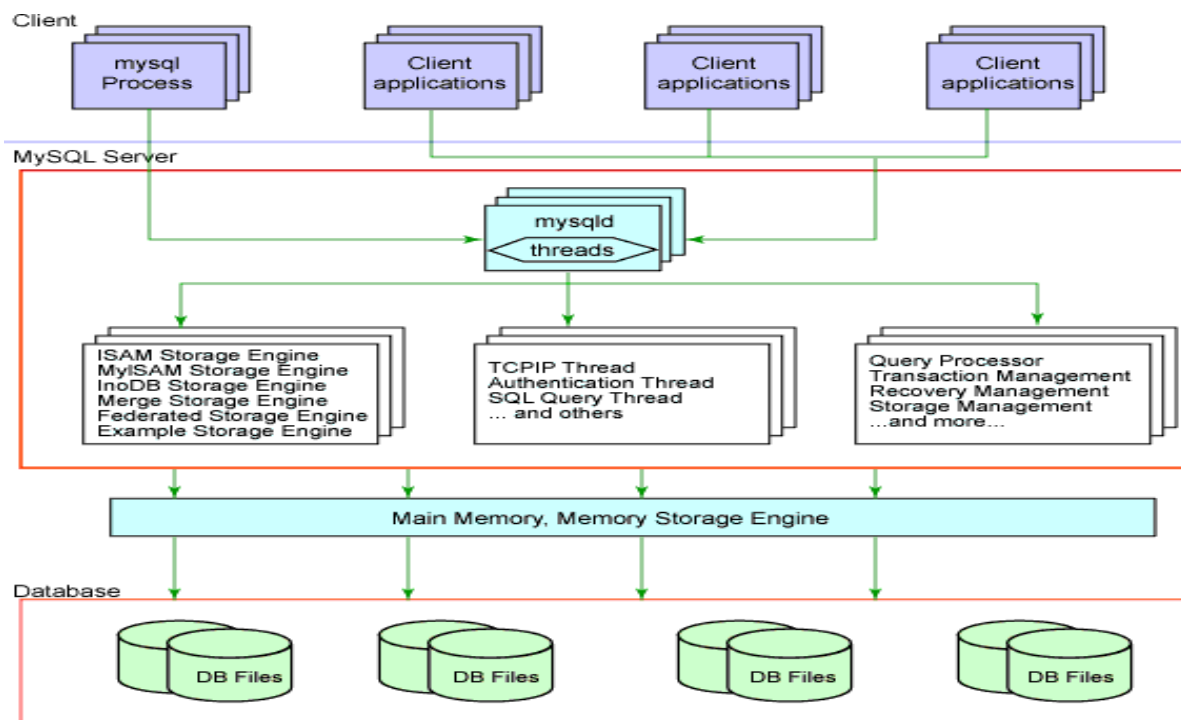
The topmost layer contains the services that aren't unique to My SQL. They're services most network-based client/server tool so servers need: connection handling, authentication, security, and so forth.





The third layer contains the storage engines. They are responsible for storing and retrieving all data stored “in” My SQL. Like the various file systems 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 or communicate with each other; they simply respond to requests from the server.



Software’s and tools used:

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 My SQL server, developed by T.c.X. Data consult AB. My SQL, available for download at

www.mysql.com, offers the database programmer with an array of options and capabilities rarely seen in other database servers. My SQL is free of charge for those wishing to use it for private and commercial use. Those wishing to develop applications specifically using My SQL should consult My SQL's licensing section, as there is charge for licensing the product.

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

- Ability to handle a nun limited number of simultaneous users.
- Capacity to handle 50,000,000+ records.
- Very fast command execution, perhaps the fastest to be found on the market.
- 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 .offers My SQL as a free product to the general public.

6.1.2 Reasons to use MySQL:

Scalability and Flexibility:

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

High Performance

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

High Availability

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

Robust Transactional Support

MySQLoffersoneofthemostpowerfultxnsactionaldatabaseenginesonthemarket.Featuresinclude 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.

We band Data Warehouse Strengths

My SQL 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.

Strong Data Protection

Because guarding the data as sets of corporations is the number one job of database professionals, My SQL offers exceptional security features that ensure absolute data protection. In terms of database authentication, My SQL 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.

Management Ease

My SQL 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.

7. System Requirements:

7.1 Hardware Requirement:

Processor	:	IntelCoreDuo2.0 GHz or more
RAM	:	3GB or More
Hard disk	:	80GB or more
Monitor	:	15” CRT, or LCD monitor
Keyboard	:	Normal or Multimedia
Mouse	:	Compatible mouse

7.2 Software Requirement

Front End	:	NETBEANS 8.2& higher, SUPPORTED JAR FILE, and JDK 19
Back End	:	My SQL Server OR ORACLE.
OS	:	Windows10 with server pack1 and higher version.

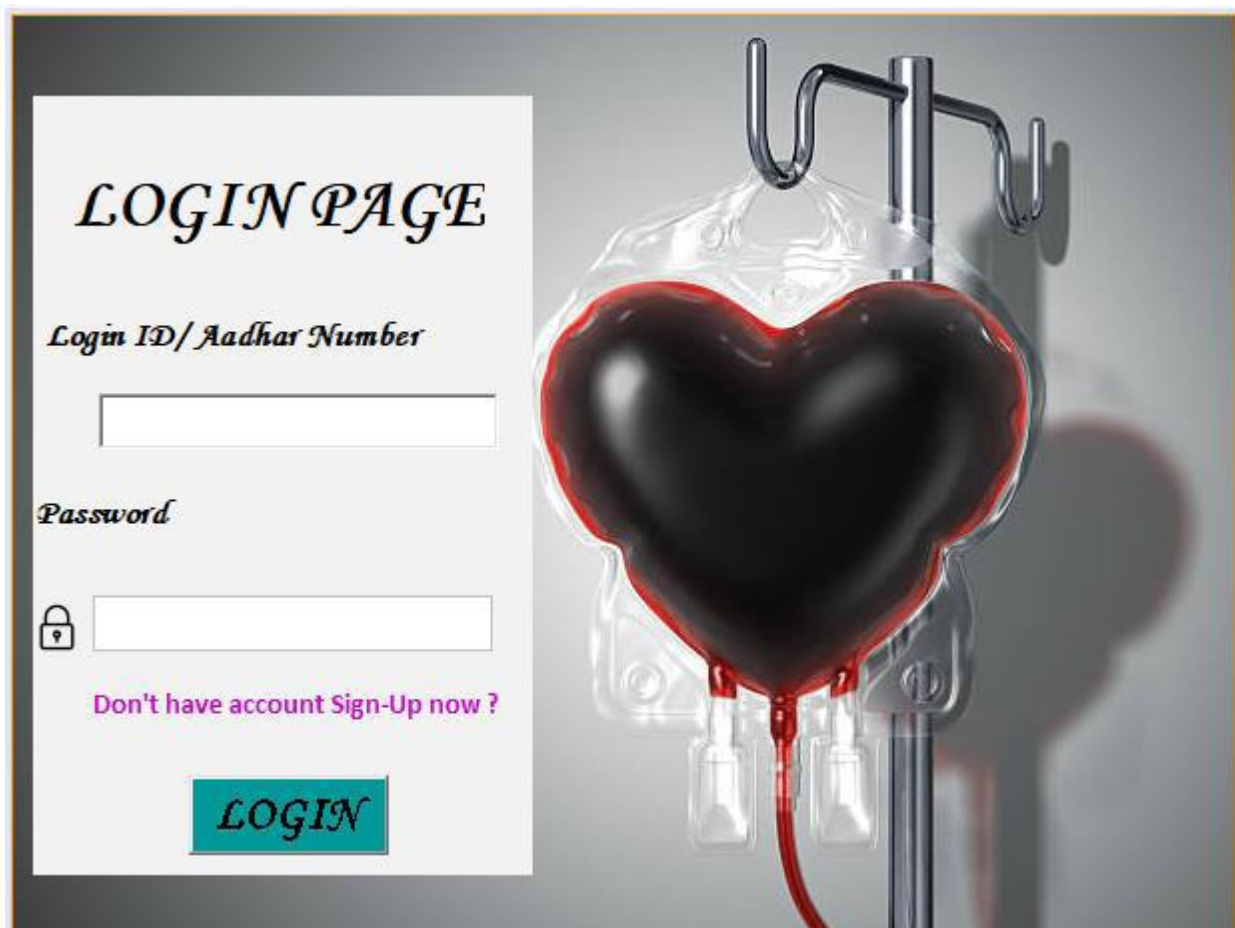
8. Module Description:

BLOOD DONATION is a website based on Java. 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.

8.1.1 Login:

The page requires donor id and password to open the donor panel. 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. Donor can change password, update profile or view donations etc

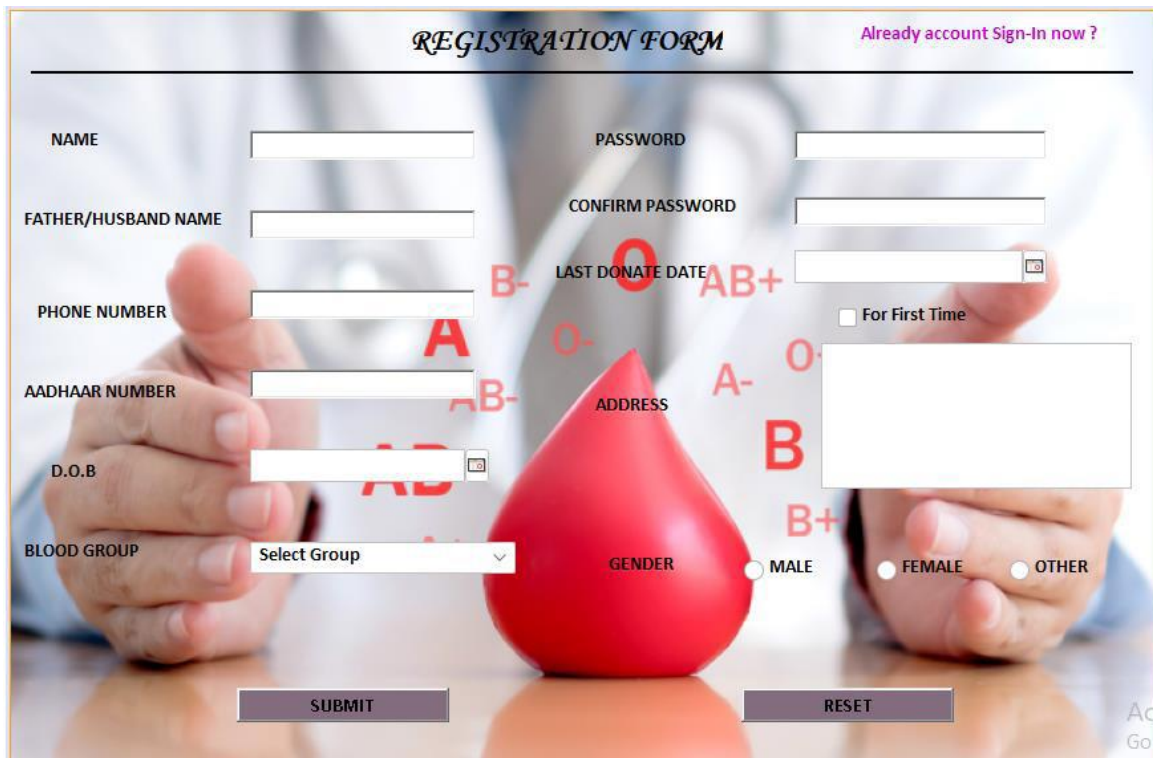
Login Page:



8.1.2 Registration Page:

Registration page includes the information of the donor who want to register. Donor can register the account by clicking on new register. He/she can add the account for the further enquiry of the blood donation.

Registration Page:

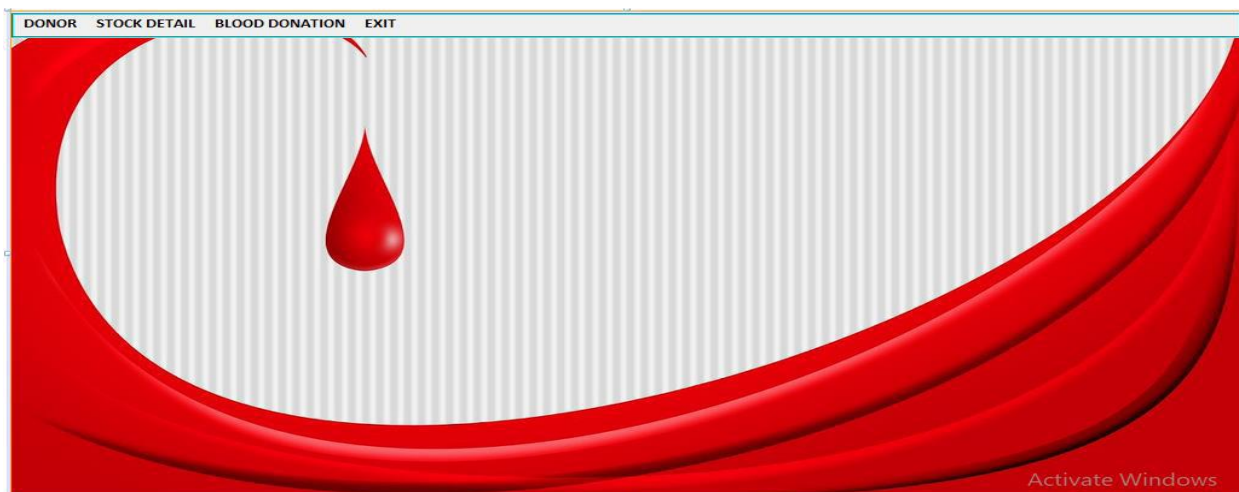


The registration form is titled "REGISTRATION FORM" and includes a link "Already account Sign-In now ?". It features input fields for NAME, FATHER/HUSBAND NAME, PHONE NUMBER, AADHAAR NUMBER, D.O.B, BLOOD GROUP (with a dropdown menu), PASSWORD, CONFIRM PASSWORD, LAST DONATE DATE, and ADDRESS. There are also radio buttons for GENDER (MALE, FEMALE, OTHER) and a checkbox for "For First Time". The form has "SUBMIT" and "RESET" buttons at the bottom. The background shows a person's hands holding a large red blood drop, with blood group labels (A, B, AB, O) visible.

8.1.3 Main Window:

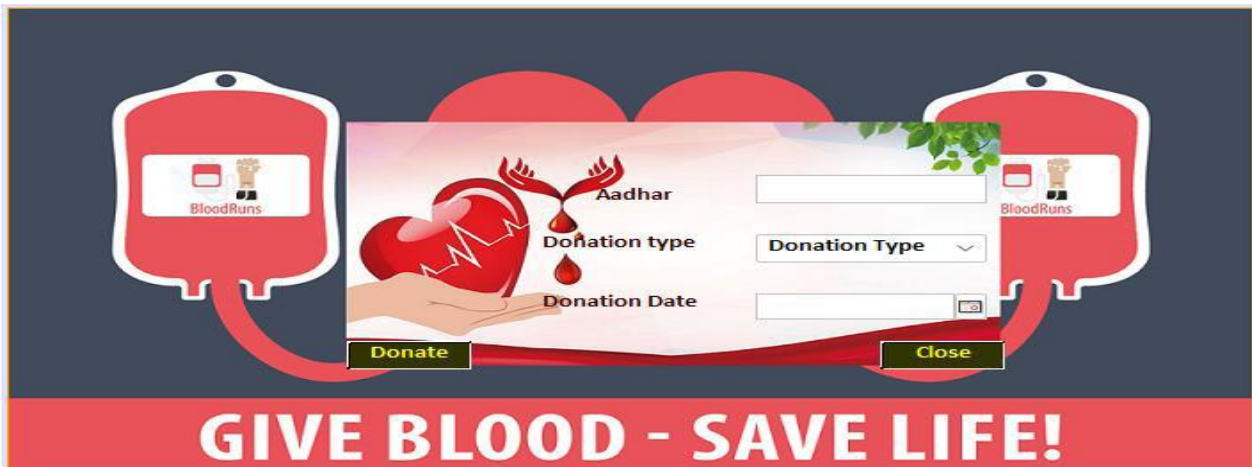
The BLOOD BANK MANAGEMENT SYSTEM is great project. This project is designed for successful completion of project on blood bank management system. The basic building aim is to provide blood donation service to the city recently. Blood and Oxygen Management System (BOMS) is a browser-based system that is designed to store process, retrieve and analyses 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.

Menu Page:



8.1.4 Other Pages:

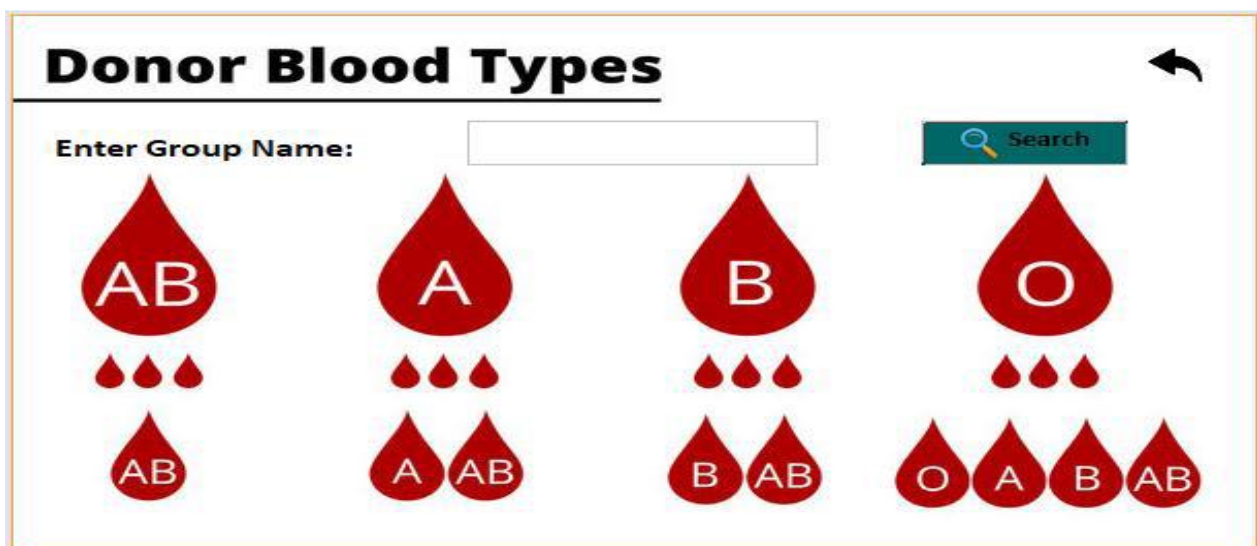
Donation Page:



The Donation Page features a central form for blood donation registration. The form includes fields for 'Aadhar', 'Donation type', and 'Donation Date'. A 'Donate' button is positioned below the 'Donation type' field, and a 'Close' button is below the 'Donation Date' field. The form is flanked by two red blood bags with 'BloodRuns' logos. A large red banner at the bottom reads 'GIVE BLOOD - SAVE LIFE!'.

GIVE BLOOD - SAVE LIFE!

Stock Details:



The Donor Blood Types interface displays a table of blood types and their corresponding stock details. The table has columns for 'Enter Group Name', 'Search', and 'Stock Details'. The stock details are represented by red blood drops.

Enter Group Name:	Search	Stock Details
AB		AB
A		A AB
B		B AB
O		O A B AB


Donor Details:




The Donor Details interface shows a list of donors with their details. The interface includes a 'Back' button, an 'Enter Aadhar' field, and a 'Search' button. The donor details are listed in a table with columns for 'Title 1' through 'Title 11'.


Title 1	Title 2	Title 3	Title 4	Title 5	Title 6	Title 7	Title 8	Title 9	Title 10	Title 11

Donor Report:



Enter Aadhar number

 Search

 Print

PATIENT NAME:

GENDER:

REFFERED BY DOCTOR:

AGE: years

SAMPLE COLLECTION AT:

DATE:

REPORT ON BLOOD SUGAR ESTIMATION

TEST	RESULT	NORMAL VALUES
URINE SUGAR:	<input type="text"/>	0 to 0.8 mmol/L
URINE ACETONE:	<input type="text"/>	1.665 to 2.22 mmol/L
BLOOD GROUP:	<input type="text"/>	
FASTING BLOOD SUGAR:	<input type="text"/>	3.9 and 5.6 mmol/L

Activate Windows

Oxygen Page:


Oxygen Bank

Enter Aadhar:

Enter Date:

Buy

close



Update Page:

UPDATE DONOR

Enter Aadhar Number

NAME

FATHER/HUSBAND NAME

PHONE NUMBER

LAST DONATE DATE

D.O.B

GENDER ☒ MALE ☐ FEMALE ☐ OTHER

BLOOD GROUP

PASSWORD

ADDRESS

Activate Windows
Go to Settings to activate Windows

9. DATABASE MODULE:

localhost ▶ blood_oxygen_bank

Structure

SQL

Search

Query

Export

Import

Operations

Privileges

Table	Action						Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> donation_table							6	InnoDB	latin1_swedish_ci	16.0 KiB	-
<input type="checkbox"/> oxygen							2	InnoDB	latin1_swedish_ci	16.0 KiB	-
<input type="checkbox"/> registration_table							15	InnoDB	latin1_swedish_ci	32.0 KiB	-
3 tables	Sum						23	InnoDB	latin1_swedish_ci	64.0 KiB	0 B

Registration table:

localhost ▶ blood_oxygen_bank ▶ registration_table									
Browse	Structure	SQL	Search	Insert	Export	Import	Operations		
#	Column	Type	Collation	Attributes	Null	Default	Extra	Action	
<input type="checkbox"/> 1	ID	varchar(8)	latin1_swedish_ci		No	None		Change	Drop More ▼
<input type="checkbox"/> 2	NAME	text	latin1_swedish_ci		No	None		Change	Drop More ▼
<input type="checkbox"/> 3	FATHER_HUSBAND_NAME	text	latin1_swedish_ci		No	None		Change	Drop More ▼
<input type="checkbox"/> 4	PHONE_NUMBER	bigint(10)			No	None		Change	Drop More ▼
<input type="checkbox"/> 5	AADHAR_NUMBER	bigint(12)			No	None		Change	Drop More ▼
<input type="checkbox"/> 6	DATE_OF_BIRTH	date			No	None		Change	Drop More ▼
<input type="checkbox"/> 7	PASSWORD	varchar(15)	latin1_swedish_ci		No	None		Change	Drop More ▼
<input type="checkbox"/> 8	LAST_DONATION_DATE	date			No	None		Change	Drop More ▼
<input type="checkbox"/> 9	ADDRESS	varchar(60)	latin1_swedish_ci		No	None		Change	Drop More ▼
<input type="checkbox"/> 10	GENDER	text	latin1_swedish_ci		No	None		Change	Drop More ▼
<input type="checkbox"/> 11	BLOOD_GROUP	varchar(3)	latin1_swedish_ci		No	None		Change	Drop More ▼

Oxygen table:

localhost ▶ blood_oxygen_bank ▶ oxygen								
Browse Structure SQL Search Insert Export Import Operations								
#	Column	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	1 Aadhar_number	bigint(12)			No	None		Change Drop More ▼
<input type="checkbox"/>	2 buy_date	date			No	None		Change Drop More ▼

Donation table:

localhost ▶ blood_oxygen_bank ▶ donation_table								
Browse Structure SQL Search Insert Export Import Operations								
#	Column	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	1 Aadhar_number	bigint(12)			No	None		Change Drop More ▼
<input type="checkbox"/>	2 Donationtype	text	latin1_swedish_ci		No	None		Change Drop More ▼
<input type="checkbox"/>	3 Donation_date	date			No	None		Change Drop More ▼

10. Data Flow Diagram:

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:


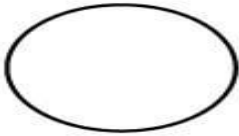
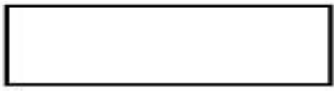

All names should be unique. This makes it easier to refer to elements in the DFD.

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.

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 existing paths of which the only one is taken. This implies an ordering of events, which makes no sense in a DFD.

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	Function
	Data flow	Used to Connect Processes to each other, to sources or Sinks; the arrow head indicates direction of data flow.
	Process	Performs Some transformation of Input data to yield output data.
	Source of Sink (External Entity)	A Source of System inputs or Sink of System outputs.
	Data Store	A repository of data; the arrow heads indicate net inputs and net 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.

10.1 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.

0-level DFD

It is also known as fundamental system model, or context diagram represents the entire software requirement as a single bubble with input and output data denoted by incoming and outgoing arrows. Then the system is decomposed and described as a DFD with multiple bubbles. Parts of the system represented by each of these bubbles are then decomposed and documented as more and more detailed DFDs. This process may be repeated at as many levels as necessary until the program at hand is well understood. It is essential to preserve the number of inputs and outputs between levels, this concept is called leveling by De Macro. Thus, if bubble "A" has two inputs x_1 and x_2 and one output y , then the expanded DFD, that represents "A" should have exactly two external inputs and one external output as shown in fig:

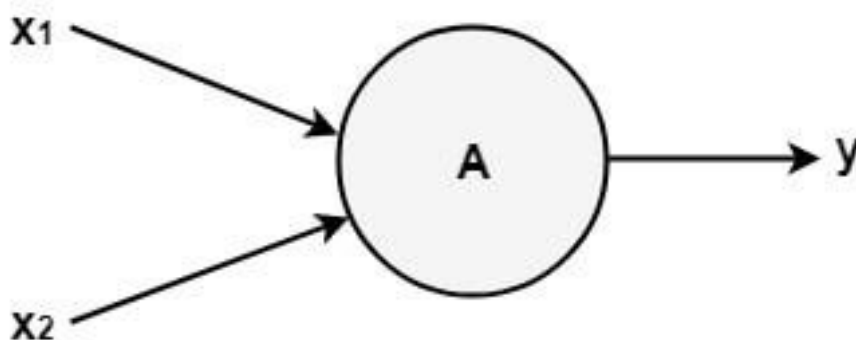
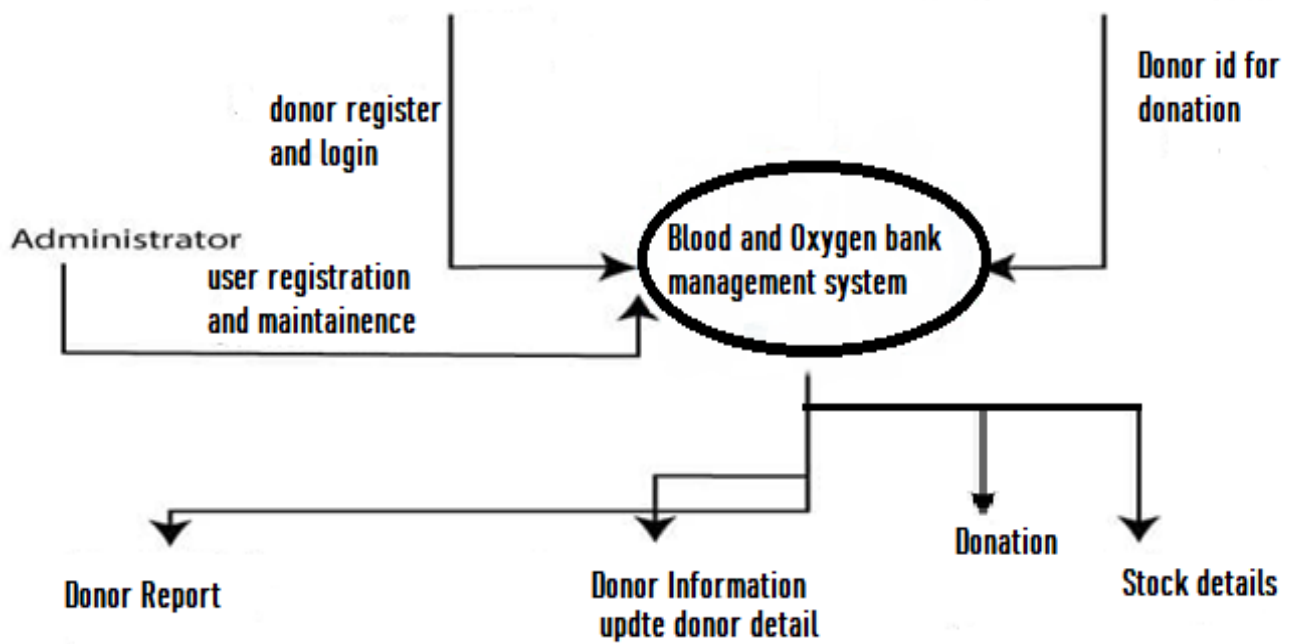


Fig: Level-0 DFD.

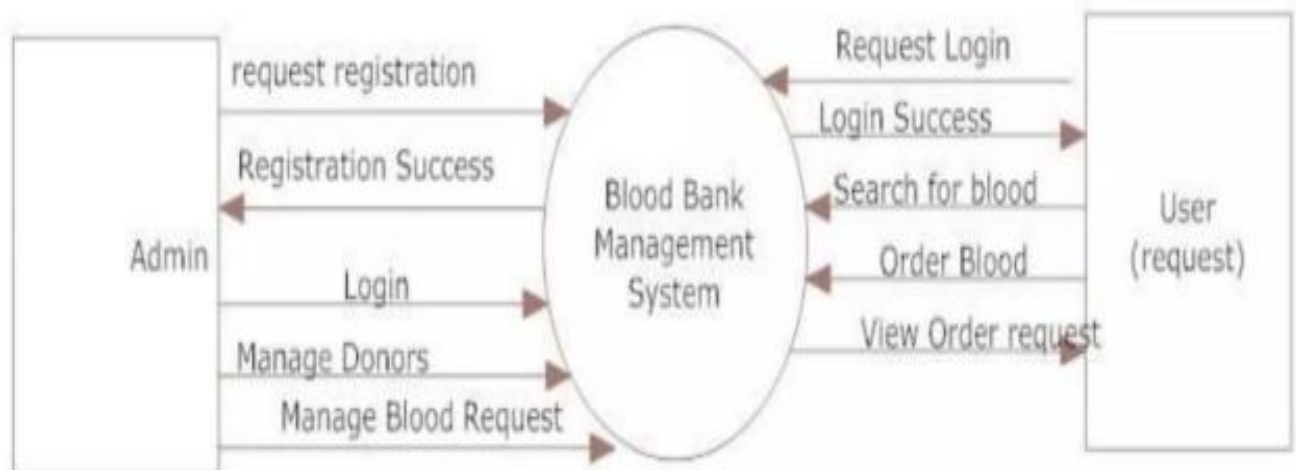
The Level-0 DFD, also called context diagram of the result management system is shown in fig. As the bubbles are decomposed into less and less abstract bubbles, the corresponding data flow may also be needed to be decomposed.



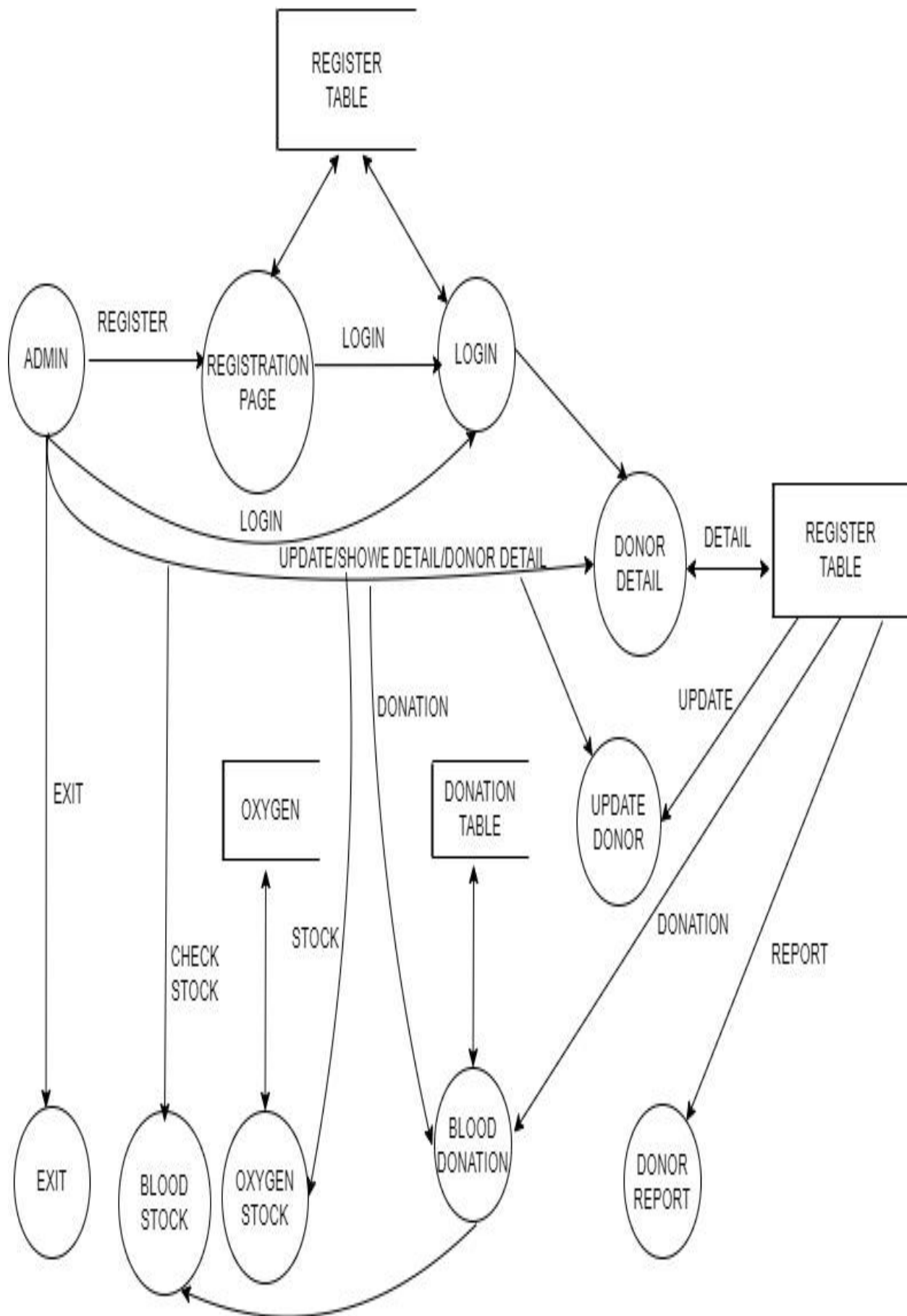
Level 0 DFD of Blood & Oxygen bank management system

1-level DFD

In 1-level DFD, a context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main objectives of the system and breakdown the high-level process of 0-level DFD into sub processes.



Level 1 DFD of Blood & Oxygen bank management system

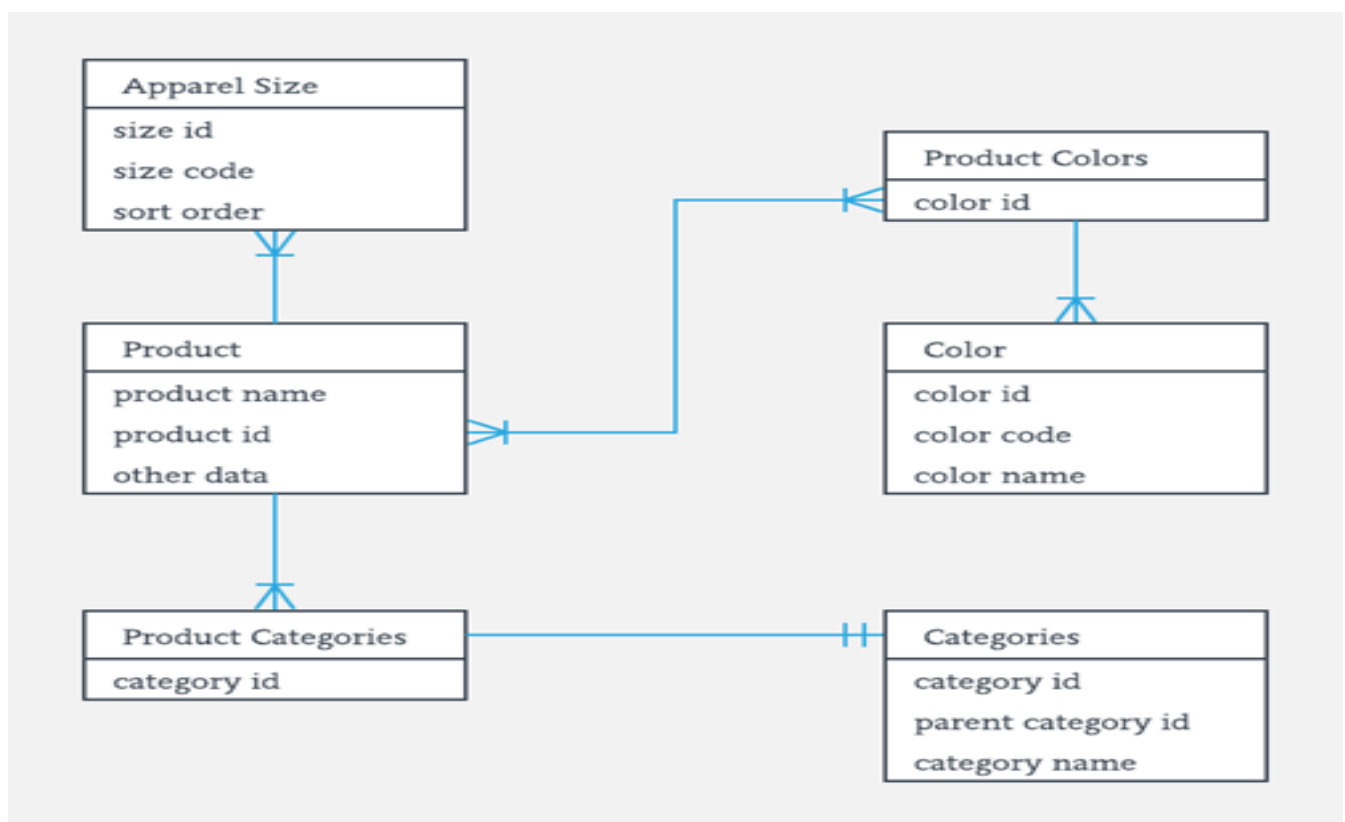


11. ER Diagram:

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 Example

11.1 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.

11.2 History of ER Models:

ER diagrams are visual tools that are helpful to represent the ER model. Peter Chen proposed ER Diagram in 1971 to create a uniform convention that can be used for relational databases and networks. He aimed to use an ER model as a conceptual modeling approach.

11.3 Use of 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

11.4 Facts about ER Diagram Model:

Now in this ERD Diagram Tutorial, let's check out some interesting facts about ER Diagram Model:

- ER model allows you to draw Database Design
- It is an easy to use graphical tool for modeling data
- Widely used in Database Design
- It is a GUI representation of the logical structure of a Database
- It helps you to identify the entities which exist in a system and the relationships between those entities.

11.5 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



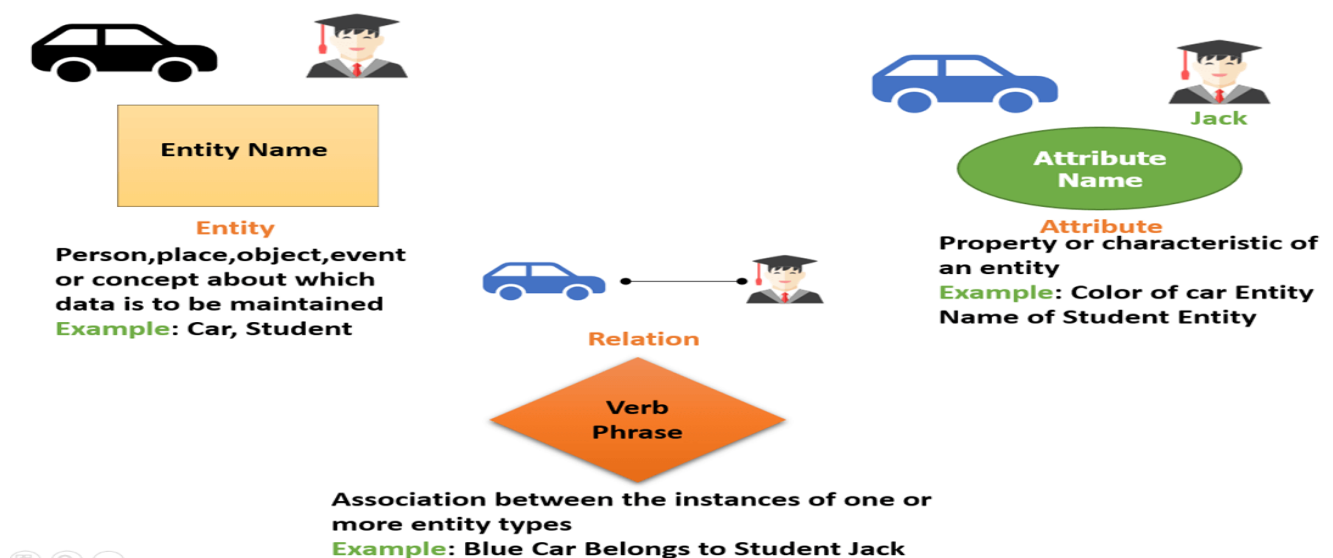
11.6 Components of the ER Diagram:

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.



Components of the ER Diagram

11.7 WHAT IS ENTITY?

A real-world thing either living or non-living that is easily recognizable and non recognizable. 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

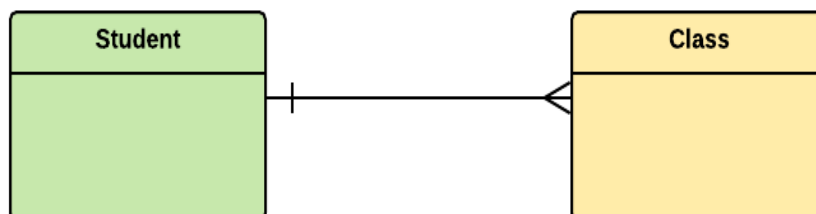
Concept: Account, Course

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.



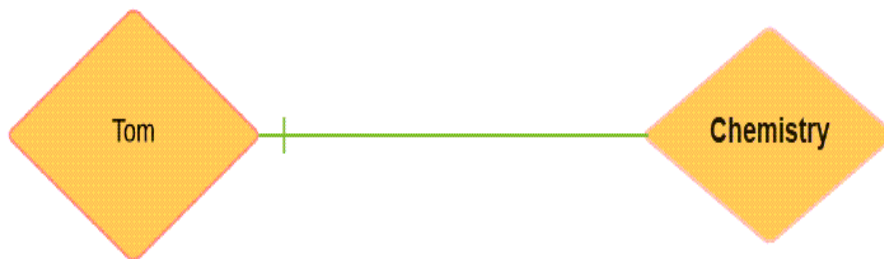
Example of Entities:

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 phrases.



For example:

You are attending this lecture

I am giving the lecture

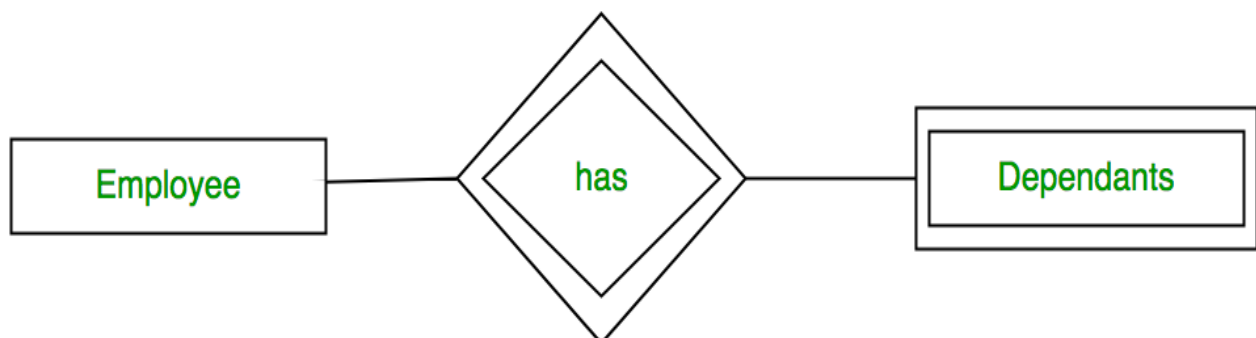
Just like entities, we can classify relationships according to relationship-types:

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.



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

Let's learn more about a weak entity by comparing it with a Strong Entity

Strong Entity Set

Weak Entity Set

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 entities 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.

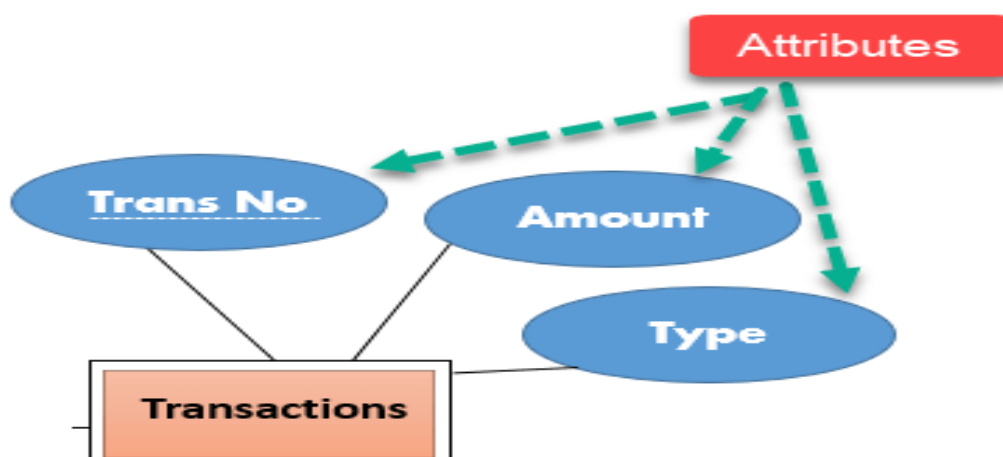
The line connecting the weak entity set for identifying relationship is double.

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



Types of Attributes

Description

Simple attribute

Simple attributes can't be divided any further. For example, a student's contact number. It is also called an atomic value.

Composite attribute

It is possible to break down composite attribute. For example, a student's full name may be further divided into first name, second name, and last name.

Derived attribute

This type of attribute does not include in the physical database. However, their values are derived from other attributes present in the database. For example, age should not be stored directly. Instead, it should be derived from the DOB of that employee.

Multi valued attribute

Multi valued attributes can have more than one values. For example, a student can have more than one mobile number, email address, etc.

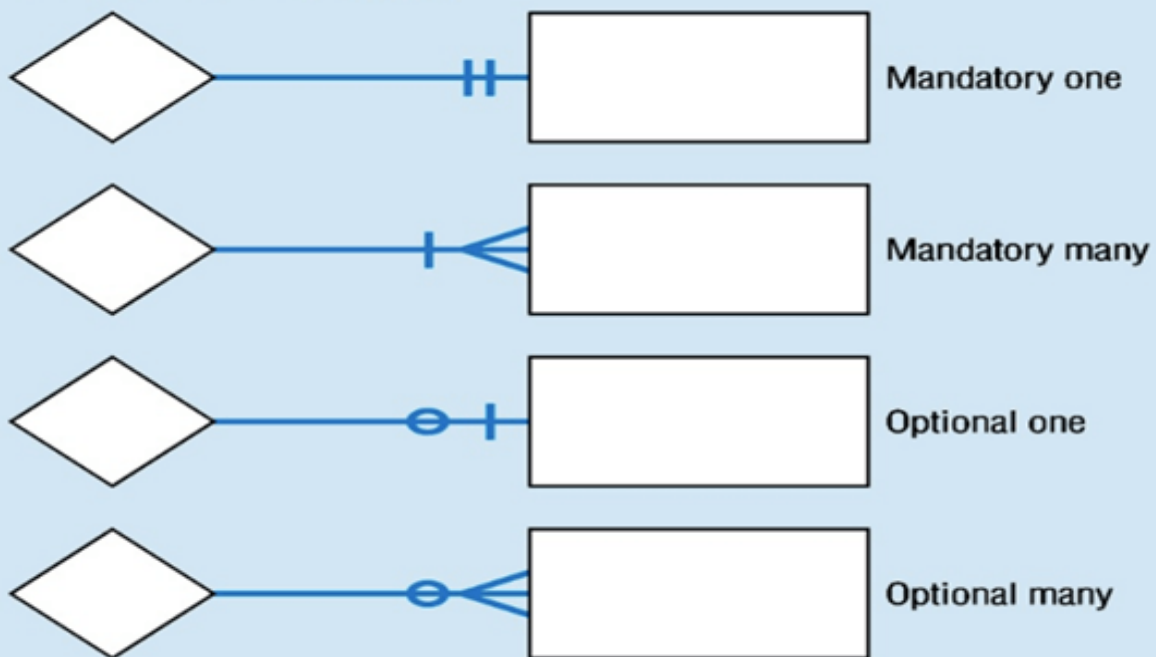
Cardinality

Defines the numerical attributes of the relationship between two entities or entity sets.

Different types of cardinal relationships are:

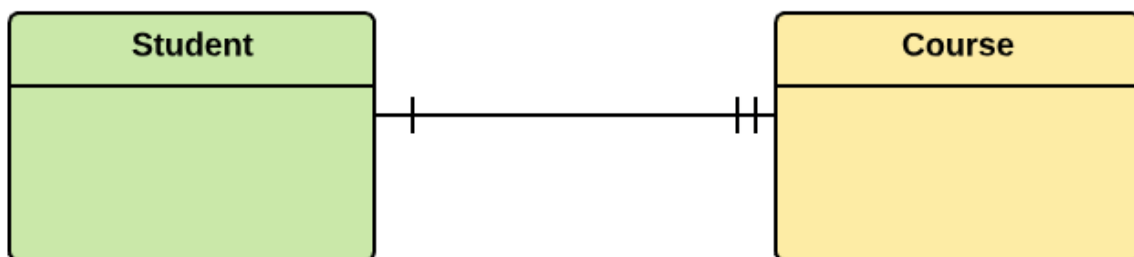
- One-to-One Relationships
- One-to-Many Relationships
- May to One Relationships
- Many-to-Many Relationships

Relationship cardinality



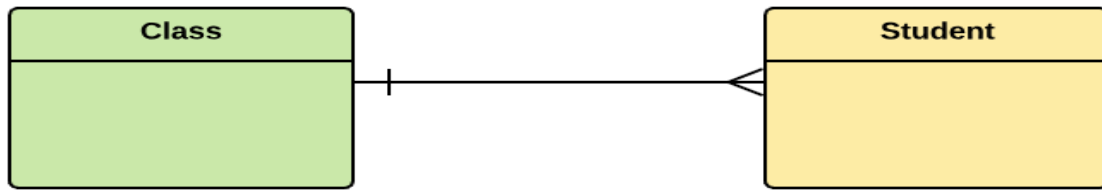
1. One-to-one:

One entity from entity set X can be associated with at most one entity of entity set Y and vice versa. Example: One student can register for numerous courses. However, all those courses have a single line back to that one student.



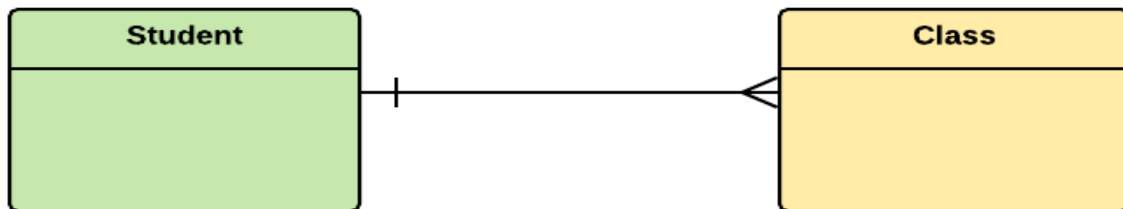
2. One-to-many:

One entity from entity set X can be associated with multiple entities of entity set Y, but an entity from entity set Y can be associated with at least one entity. For example, one class is consisting of multiple students.



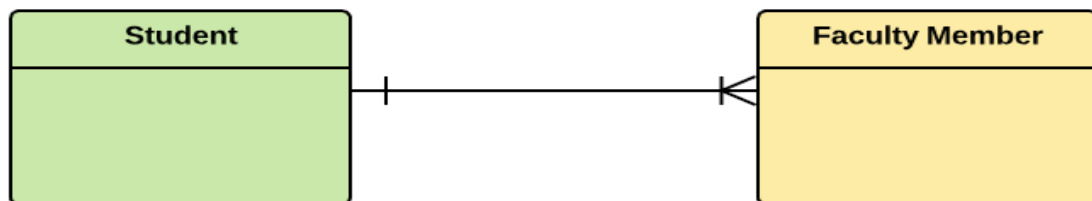
3. Many to One

More than one entity from entity set X can be associated with at most one entity of entity set Y. However, an entity from entity set Y may or may not be associated with more than one entity from entity set X. For example, many students belong to the same class.

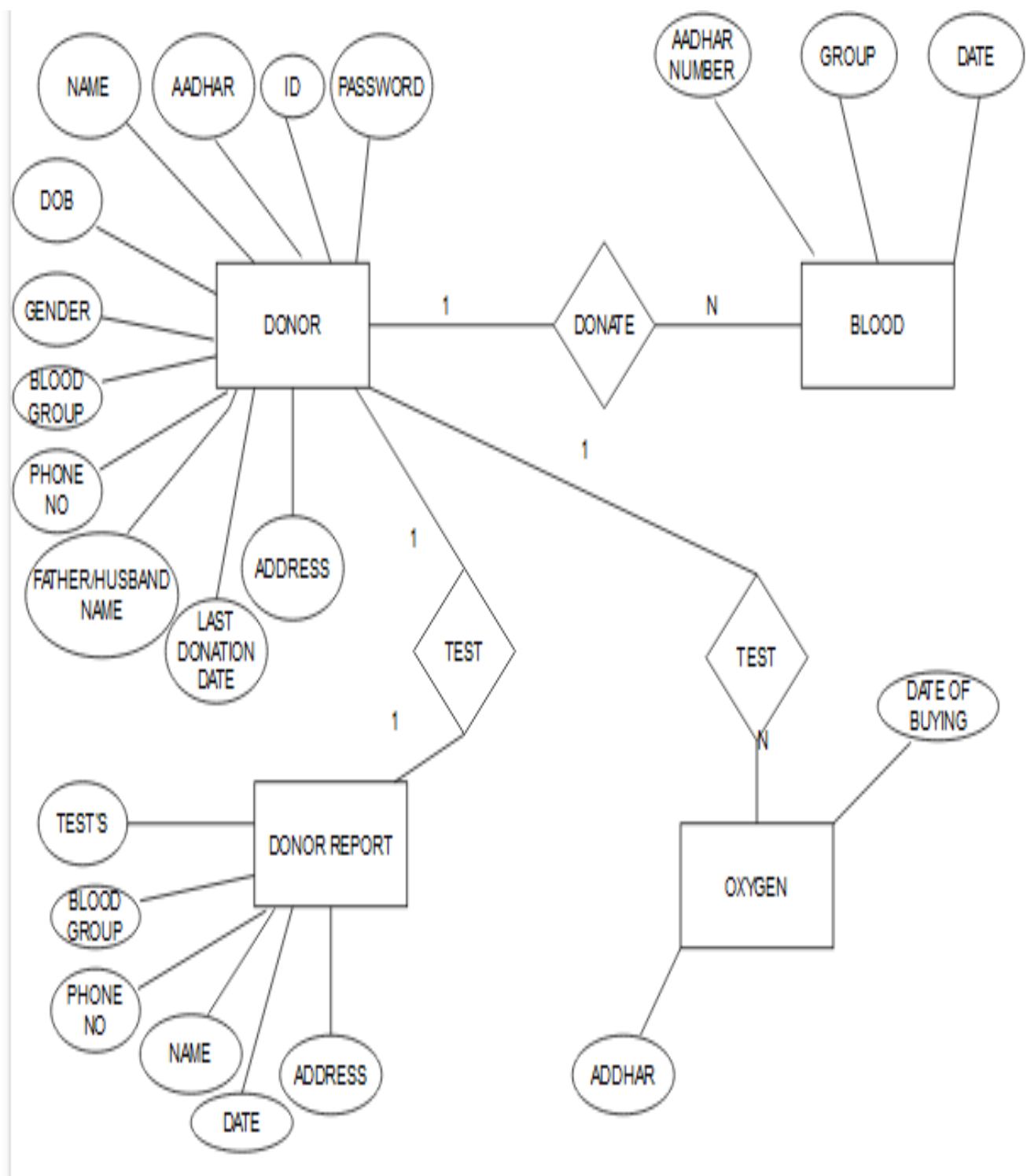


4. Many to Many:

One entity from X can be associated with more than one entity from Y and vice versa. For example, Students as a group are associated with multiple faculty members, and faculty members can be associated with multiple students.



ER Diagram:



12. Coding:

Connection Class:

```
import java.awt.Component;

import java.sql.*;

import javax.swing.JOptionPane;

public class DBconnect {

    public Connection con;

    public ResultSet rs;

    public DBconnect()throws SQLException
    {

        try

        {

            Class.forName("com.mysql.cj.jdbc.Driver");

            con=
DriverManager.getConnection("jdbc:mysql://localhost:3306/blood_oxygen_bank","root","");

        }

        catch (ClassNotFoundException | SQLException e)

        {

            JOptionPane.showMessageDialog(((Component) this.con,e);

        }

    }

    public int Insert(PreparedStatement pt)

    {

        int i=0;

        try
```

```

{
    i=pt.executeUpdate();
}
catch(SQLException e)
{
    JOptionPane.showMessageDialog((Component) this.con,e);
}
return i;
}
public int Update(PreparedStatement pt)
{
    int i=0;
    try
    {
        // JOptionPane.showMessageDialog((Component) this,"upd");
        i=pt.executeUpdate();
    }
    catch(SQLException e)
    {
        JOptionPane.showMessageDialog((Component) this.con,e);
    }
    return i;
}
public int Delete(PreparedStatement pt)
{

```

```

    int i=0;

    try
    {
        i=pt.executeUpdate();
    }

    catch(SQLException e)
    {
        JOptionPane.showMessageDialog((Component) this.con,e);
    }

    return i;
}

public int View(PreparedStatement pt)
{
    try
    {
        rs=pt.executeQuery();

    }

    catch(SQLException e)
    {
        JOptionPane.showMessageDialog((Component)this.con,e);
    }

    return 1;
}

```



```
public ResultSet view(PreparedStatement pt)
```

```
{
```

```
    try
```

```
    {
```

```
        rs=pt.executeQuery();
```

```
    }
```

```
    catch(SQLException e)
```

```
    {
```

```
        JOptionPane.showMessageDialog((Component)this.con,e);
```

```
    }
```

```
    return rs;
```

```
}
```

```
void close() {
```

```
    throw new UnsupportedOperationException("Not supported yet."); // Generated from  
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
```

```
}
```

```
}
```

```
Login Page:
```

```
try
```

```
{
```

```
    DBconnect connect=new DBconnect();
```

```
    i=txtid.getText();
```

```

Pattern p=Pattern.compile("[^a-zA-Z0-9]");
Pattern pat=Pattern.compile("[A-Za-z]++$");
Matcher m= p.matcher(i);
Matcher mt= pat.matcher(i);
if(mt.find() || m.find())
{
    pt=connect.con.prepareStatement("select aadhar_number, NAME from registration_table
where id=? and Password=?");

    pt.setString(1,i);
    pt.setString(2,txtpass.getText());
    rs=connect.view(pt);
    if(rs.next()==true)
    {
        aadhar=Long.parseLong(rs.getString("aadhar_number"));
        name=rs.getString("NAME");
        aa=String.valueOf(aadhar);
        txtaadharval.setText(aa);
        JOptionPane.showMessageDialog(this, "welcome "+name);
        this.hide();
        MenuBar b=new MenuBar();
        b.txtaadharval1.setText(txtaadharval.getText());
        b.setVisible(true);
    }
}
else if(txtid.getText().equals("") && txtpass.getText().equals(""))

```

```

{
    JOptionPane.showMessageDialog(this, "Field is empty.");
    txtid.requestFocus();
}
else
{
    aadhar= Long.parseLong(i);

    pt=connect.con.prepareStatement("select    name    from    registration_table    where
aadhar_number=? and Password=?");

    pt.setLong(1,aadhar);
    pt.setString(2,txtpass.getText());
    rs=connect.view(pt);
    if(rs.next()==true)
    {
        name=rs.getString("name");
        txtaadharval.setText(String.valueOf(aadhar));
        JOptionPane.showMessageDialog(this, "welcome "+name);
        this.hide();

        MenuBar b=new MenuBar();
        b.txtaadharval1.setText(txtaadharval.getText());
        b.setVisible(true);
    }
}

connect.con.close();
}

```

```
catch(HeadlessException | NumberFormatException e)
```

```
{
```

```
    JOptionPane.showMessageDialog(this, e.getMessage());
```

```
}
```

```
catch (SQLException ex)
```

```
{
```

```
    Logger.getLogger(Log.class.getName()).log(Level.SEVERE, null, ex.getMessage());
```

```
}
```

Registration:

```
try
```

```
{
```

```
    String r_val =  
"0123456789@#$%*?"+"ABCDEFGHIJKLMNOPQRSTUVWXYZ#@*?"+"0123456789@#$%&  
*?"+"abcdefghijklmnopqrstuvwxyz#@*?";    Random rand = new Random();
```

```
    StringBuilder b = new StringBuilder();
```

```
    while (b.length() < 8)
```

```
{
```

```
    b.append(r_val.charAt((int)(rand.nextDouble()*r_val.length())));
```

```
}
```

```
String id=b.toString();
```

```
DBconnect connect=new DBconnect();
```

```
PreparedStatement pt=connect.con.prepareStatement("insert into registration_table  
values(?,?,?,?,?,?,?,?,?,?)");
```

```
pt.setString(1, id);
```

```
pt.setString(2, txtname.getText());
```

```
pt.setString(3, txtfather_husbandname.getText());
```

```

Pattern p = Pattern.compile("(0/91)?[7-9][0-9]{9}");
Matcher m = p.matcher(txtnumber.getText());
if (m.matches())
{
    pt.setLong(4, Long.parseLong(txtnumber.getText()));
}
else
{
    JOptionPane.showMessageDialog(this,"Enter correct Phone Number");
}
Pattern pd = Pattern.compile("^[2-9]{1}[0-9]{3}[0-9]{4}[0-9]{4}$");
Matcher mc = pd.matcher(txtaadhar.getText());
if (mc.matches())
{
    pt.setLong(5, Long.parseLong(txtaadhar.getText()));
}
else
{
    JOptionPane.showMessageDialog(this,"Enter correct Aadhaar Number");
}
SimpleDateFormat sd;
sd = new SimpleDateFormat("yyyy-MM-dd");
String da= sd.format(txtdob.getDate());
DateTimeFormatter dt=DateTimeFormatter.ofPattern("yyyy-MM-dd");
LocalDateTime ld= LocalDateTime.now();

```

```

String d=dt.format(ld);
LocalDate Ld= LocalDate.parse(da);
LocalDate LdL= LocalDate.parse(d);
int x=Ld.getYear();
int y=LdL.getYear();
if((y-x)>=18)
{
    pt.setString(6,da);
}
else
{
    JOptionPane.showMessageDialog(this,"18+ Only.");
}
if(txtpass.getText().equals(txtconpass.getText()))
{
    pt.setString(7,txtpass.getText());
}
else
{
    JOptionPane.showMessageDialog(this, "Password Missmatch.");
}
if(checkmark.isSelected())
{
    pt.setDate(8,java.sql.Date.valueOf("1947-08-15"));
}

```

```

else
{
    pt.setString(8,sd.format(txtlastdate.getDate()));
}

pt.setString(9, txtaddress.getText());
if(Radiomale.isSelected())
{
    pt.setString(10,Radiomale.getText());
}
else if(Radiofemale.isSelected())
{
    pt.setString(10,Radiofemale.getText());
}
else
{
    pt.setString(10,Radioother.getText());
}

if(Combogroup.getSelectedIndex()>=1)
{
    pt.setString(11,Combogroup.getItemAt(Combogroup.getSelectedIndex()));
}

int r= connect.Insert(pt);
if(r==1)
{
    JOptionPane.showMessageDialog(this, ""

```

Registration Successfull.

Your Id: """+id);

txtname.setText(null);

txtfather_husbandname.setText(null);

txtnumber.setText(null);

txtaadhar.setText(null);

txtdob.setCalendar(null);

txtpass.setText(null);

txtconpass.setText(null);

txtlastdate.setCalendar(null);

txtaddress.setText(null);

if(Radiomale.isSelected())

{

Radiogroup.clearSelection();

}

else if(Radiofemale.isSelected())

{

Radiogroup.clearSelection();

}

else

{

Radiogroup.clearSelection();

}

Combogroup.setSelectedIndex(0);

this.hide();


```

    Login ob=new Login();

    ob.txtid.setText(id);

    ob.setVisible(true);

    connect.con.close();

}

else

{

    JOptionPane.showMessageDialog(this, "Registration Unsuccessfull.");

}

connect.con.close();

}

catch(HeadlessException | NumberFormatException | SQLException e)

{

    JOptionPane.showMessageDialog(this, e.getMessage());

}

```

Search:

```

try

{

    DBconnect connect=new DBconnect();

    pt=connect.con.prepareStatement("select      *      from      registration_table      where
aadhar_number=?");

    pt.setLong(1, Long.parseLong(txtaadhar.getText()));

    rs = connect.view(pt);

    if(rs.next()==true)

    {

```

```

txtname.setText(rs.getString(2));
txtfat_husName.setText(rs.getString(3));
txtnumber.setText(rs.getString(4));
date=new SimpleDateFormat("yyyy-MM-dd");
try
{
    ds = (Date) date.parse(rs.getString(6));
    da = (Date) date.parse(rs.getString(8));
    txtdob.setDate(ds);
    txtlastdate.setDate(da);
}
catch (ParseException ex)
{
    Logger.getLogger(Update.class.getName()).log(Level.SEVERE, null, ex.getMessage());
}
String gender=rs.getString(10);
switch (gender)
{
    case "MALE" -> Radiomale.setSelected(true);
    case "FEMALE" -> Radiofemale.setSelected(true);
    case "OTHER" -> Radioother.setSelected(true);
    default ->
    {
    }
}

```

```

        txtpass.setText(rs.getString(7));

        txtaddress.setText(rs.getString(9));

        String b=rs.getString(11);

        txtgroup.setText(b);
    }

    connect.con.close();

}

catch(NumberFormatException | SQLException ex)

{

    JOptionPane.showMessageDialog(this, ex.getMessage());

}

```

Update:

try

```

{

    DBconnect connect=new DBconnect();

    String          q="update          registration_table          set
name=?,FATHER_HUSBAND_NAME=?,PASSWORD=?,PHONE_NUMBER=?,DATE_OF_BIR
TH=?,LAST_DONATION_DATE=?,ADDRESS=?,GENDER=?,BLOOD_GROUP=?          where
aadhar_number=?";

    pt = connect.con.prepareStatement(q);

    pt.setString(1,txtname.getText());

    pt.setString(2,txtfat_husName.getText());

    pt.setString(3,txtpass.getText());

    pt.setLong(4,Long.parseLong(txtnumber.getText()));

    Date d=txtdob.getDate();

    DateFormat dx=new SimpleDateFormat("yyyy-MM-dd");

```

```

pt.setString(5,dx.format(d));
Date dg=txtlastdate.getDate();
pt.setString(6,dx.format(dg));
pt.setString(7,txtaddress.getText());
if(Radiomale.isSelected())
{
    pt.setString(8,Radiomale.getText());
}
else if(Radiofemale.isSelected())
{
    pt.setString(8,Radiofemale.getText());
}
else
{
    pt.setString(8,Radioother.getText());
}
pt.setString(9,txtgroup.getText());
pt.setLong(10,Long.parseLong(txtaadhar.getText()));

```

```

int x=JOptionPane.showConfirmDialog(this, "Are you sure for Updation
?","Select",JOptionPane.YES_NO_OPTION);
if(x==0)
{
    int f=connect.Update(pt);
    if(f==1)

```

```

{
    JOptionPane.showMessageDialog(this, f+" Record Updated.");
    txtname.setText(null);
    txtfat_husName.setText(null);
    txtnumber.setText(null);
    txtaadhar.setText(null);
    txtdob.setCalendar(null);
    if(Radiomale.isSelected())
    {
        Radiogroup.clearSelection();
    }
    else if(Radiofemale.isSelected())
    {
        Radiogroup.clearSelection();
    }
    else
    {
        Radiogroup.clearSelection();
    }
    txtpass.setText(null);
    txtgroup.setText(null);
    txtlastdate.setCalendar(null);
    txtaddress.setText(null);
    Combogroup.setSelectedIndex(0);
    txtaadhar.requestFocus();
}

```

```

    }
    }

    connect.con.close();

}

catch(NumberFormatException | SQLException ex)
{
    JOptionPane.showMessageDialog(this, ex.getMessage());
}

```

Oxygen:

```

try
{
    DBconnect connect=new DBconnect();

    PreparedStatement pt=connect.con.prepareStatement("insert into oxygen values(?,?)");

    pt.setLong(1,Long.parseLong(txtaadhar.getText()));

    SimpleDateFormat sd;

    sd = new SimpleDateFormat("yyyy-MM-dd");

    pt.setString(2,sd.format(txtdate.getDate()));

    int x=connect.Insert(pt);

    if(x==1)
    {
        JOptionPane.showMessageDialog(this," Thanks for buying.");
    }

    txtaadhar.setText(null);

    txtdate.setCalendar(null);
}

```

```

    }

    catch(NumberFormatException | SQLException e)
    {
        JOptionPane.showMessageDialog(this, e.getMessage());
    }

```

Report:

```

    try
    {
        DBconnect connect=new DBconnect();

        pt=connect.con.prepareStatement("select
GENDER,BLOOD_GROUP,DATE_OF_BIRTH      from      registration_table      NAME,
aadhar_number=?");                                where

        pt.setLong(1, Long.parseLong(txtaadhar.getText()));

        rs=connect.view(pt);

        if(rs.next()==true)
        {
            Labname.setText(rs.getString(1));

            Labgender.setText(rs.getString(2));

            Labbloodgroup.setText(rs.getString(3));

            LabDR.setText("Devesh Gupta");

            Labsample.setText("Dev Lab's");

            Labsugar.setText("0.7 mmol/L");

            Labacetone.setText("2.05 mmol/L");

            Labfasting.setText("5.5 mmol/L");

            /*Code for current date */

```

```

    DateTimeFormatter dft=DateTimeFormatter.ofPattern("yyyy-MM-dd");

    LocalDateTime lfd= LocalDateTime.now();

    String dax=dft.format(lfd);

    Labdate.setText(dax);

    /*Code for current date */

    try
    {
        //holding dob year value

        //holding date in date type variable

        Date da = new SimpleDateFormat("yyyy-MM-dd").parse(rs.getString(4));

        String fd=new SimpleDateFormat("yyyy-MM-dd").format(da);//checking format and
convert into string

        LocalDate LdL= LocalDate.parse(fd);

        int g = LdL.getYear();

        //JOptionPane.showMessageDialog(this, g);

        //holding dob year value

        //holding current year value

        DateTimeFormatter dt=DateTimeFormatter.ofPattern("yyyy-MM-dd");

        LocalDateTime ld= LocalDateTime.now();

        String de=dt.format(ld);

        LocalDate LdLg= LocalDate.parse(de);

        int y=LdLg.getYear();

        //JOptionPane.showMessageDialog(this, y);

        //holding current year value

```



```

        Labage.setText( String.valueOf(y-g));
    }
    catch(HeadlessException | SQLException | ParseException ex)
    {
        JOptionPane.showMessageDialog(this, ex.getMessage());
    }
}
}
catch(NumberFormatException | SQLException ex)
{
    JOptionPane.showMessageDialog(this, ex.getMessage());
}

```

Stock:

```

try
{
    DBconnect connect=new DBconnect();

    pt=connect.con.prepareStatement("SELECT      COUNT(BLOOD_GROUP)      FROM
registration_table where BLOOD_GROUP=?");

    pt.setString(1, txtgroup.getText());

    rs = connect.view(pt);

    if(rs.next()==true)
    {
        JOptionPane.showMessageDialog(this, rs.getString(1));
    }
}

```

```

catch(NumberFormatException | SQLException e)
{
    JOptionPane.showMessageDialog(this, e.getMessage());
}

```

Moving page:

```

MenuBar mb=new MenuBar();

```

```

    this.hide();

    mb.setVisible(true);

```

Donation:

```

try {
    DBconnect connect=new DBconnect();

    String q=" select * from registration_table where aadhar_number=?";

    pt = connect.con.prepareStatement(q);

    n=Long.parseLong(txtaadhardonate.getText());

    pt.setLong(1,n);

    rs= connect.view(pt);

    if(rs.next()==false)
    {
        JOptionPane.showMessageDialog(this, "First register your self.");

        Registration reg=new Registration();

        reg.setVisible(true);
    }

    else
    {
        String w="insert into donation_table values(?,?,?)";

```

```

    pt = connect.con.prepareStatement(w);
    pt.setLong(1,n);
    if(Combodonatetype.getSelectedIndex()>=1)
    {
        pt.setString(2,Combodonatetype.getItemAt(Combodonatetype.getSelectedIndex()));
    }
    SimpleDateFormat sd;
    sd = new SimpleDateFormat("yyyy-MM-dd");
    String da= sd.format(Donationdate.getDate());
    pt.setString(3,da);
    x=connect.Insert(pt);
    if(x==1)
    {
        JOptionPane.showMessageDialog(this, "Thanks for your donation.");
        try
        {
            String q1="update  registration_table  set  LAST_DONATION_DATE=?  where
aadhhar_number='"+n+"'";
            pt = connect.con.prepareStatement(q1);
            //pt.setLong(1,Long.parseLong(txtaadhardonate.getText()));
            Date d=Donationdate.getDate();
            DateFormat dx=new SimpleDateFormat("yyyy-MM-dd");
            pt.setString(1,dx.format(d));
            connect.Update(pt);
        }
    }

```

```

        catch(NumberFormatException | SQLException e)
        {
            JOptionPane.showMessageDialog(this, e.getMessage());
        }

        txtaadhardonate.setText(null);

        Combodonatetype.setSelectedIndex(0);

        Donationdate.setCalendar(null);
    }
}

catch (HeadlessException | NumberFormatException | SQLException ex)
{
    JOptionPane.showMessageDialog(this, ex.getMessage());
}

```

Search donor:

```

try
{
    DBconnect connect= new DBconnect();

    pt=connect.con.prepareStatement("select * from registration_table where
AADHAR_NUMBER=?");

    pt.setLong(1, Long.parseLong(txtaadhar.getText()));

    rs=connect.view(pt);

    tabledata.setModel(DbUtils.resultSetToTableModel(rs));

    //SELECT *, DATE_FORMAT(FROM_DAYS(DATEDIFF(NOW(), date_of_birth)), '%Y') +
0 AS age FROM registration_table;

    connect.con.close();
}

```

```

    }

    catch(SQLException e)

    {

        JOptionPane.showMessageDialog(this, e.getMessage());

    }

}

```

Menu bar:

```

import javax.swing.JOptionPane;

public class MenuBar extends javax.swing.JFrame {

    public MenuBar()

    {

        initComponents();

    }

    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {

        DonorReport dr=new DonorReport();

        dr.txtaadhar.setText(txtaadharval1.getText());

        this.hide();

        dr.setVisible(true);

    }

    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {

        Registration ob=new Registration();

        this.hide();

        ob.setVisible(true);

    }

    private void jMenuItem5ActionPerformed(java.awt.event.ActionEvent evt) {

```

```

    BloodStock bs=new BloodStock();

    this.hide();

    bs.setVisible(true);

}

private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {

    Update up=new Update();

    up.txtaadhar.setText(txtaadharval1.getText());

    this.hide();

    up.setVisible(true);

}

private void jMenuItem7ActionPerformed(java.awt.event.ActionEvent evt) {

    BloodDonation bd=new BloodDonation();

    this.hide();

    bd.setVisible(true);

}

private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {

    DonorDetails dd=new DonorDetails();

    dd.txtaadhar.setText(txtaadharval1.getText());

    this.hide();

    dd.setVisible(true);

}

private void formWindowActivated(java.awt.event.WindowEvent evt) {

    txtaadharval1.setVisible(false);

}

```

```

private void jMenuItem6MouseClicked(java.awt.event.MouseEvent evt) {
    int x=JOptionPane.showConfirmDialog(this, "Are you sure
?", "Select",JOptionPane.YES_NO_OPTION);
    if(x==0)
    {
        System.exit(0);
    }
}

private void jMenuItem9ActionPerformed(java.awt.event.ActionEvent evt) {
    BloodDonation bd=new BloodDonation();
    this.hide();
    bd.setVisible(true);
}

private void jMenuItem8ActionPerformed(java.awt.event.ActionEvent evt) {
    BloodDonation bd=new BloodDonation();
    this.hide();
    bd.setVisible(true);
}

private void jMenuItem6ActionPerformed(java.awt.event.ActionEvent evt) {
    Oxygen ox=new Oxygen();
    this.hide();
    ox.setVisible(true);
}

public static void main(String args[]) {
    try {

```

```

        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MenuBar.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
    catch (InstantiationException ex)
    {

java.util.logging.Logger.getLogger(MenuBar.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MenuBar.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MenuBar.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
}
//</editor-fold>

```



```

    /* Create and display the form */

    java.awt.EventQueue.invokeLater() -> {
        new MenuBar().setVisible(true);
    });
}

// Variables declaration - do not modify

private javax.swing.JLabel jLabel3;
private javax.swing.JMenu jMenuItem1;
private javax.swing.JMenu jMenuItem2;
private javax.swing.JMenu jMenuItem3;
private javax.swing.JMenu jMenuItem6;
private javax.swing.JMenuBar jMenuItemBar1;
private javax.swing.JMenuItem jMenuItem1;
private javax.swing.JMenuItem jMenuItem2;
private javax.swing.JMenuItem jMenuItem3;
private javax.swing.JMenuItem jMenuItem4;
private javax.swing.JMenuItem jMenuItem5;
private javax.swing.JMenuItem jMenuItem6;
private javax.swing.JMenuItem jMenuItem7;
private javax.swing.JMenuItem jMenuItem8;
private javax.swing.JMenuItem jMenuItem9;
public javax.swing.JTextField txtaadharval1;

```

```
// End of variables declaration
```

```
}
```

13. Future Scope:

BLOOD BANK and OXYGEN BANK MANAGEMENT is a software application to build 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 avoided through adding some message sending procedure this will help to find proper blood donor in time. This will provide availability of blood in time.

14. Bibliography:

Java: <https://www.javatpoint.com/java-tutorial>

Java Swing: <https://www.javatpoint.com/java-swing> , <https://www.javatpoint.com/awt-and-swing-in-java>

SQL: <https://www.w3schools.com/sql/>

Data Structure: <https://www.javatpoint.com/data-structure-tutorial>

	BOOK NAME	WRITE NAME
JAVA	The Complete Reference	Herbert Schildt
JAVA SWING	Java Swing Complete comprehensive guide	Amro Solima
DATABASE	Database Management Systems	Raghu Ramakrishnan and Johannes Gehrke
DATA STRUCTURE	Data Structures Through C	G.S Baluja
SQL	The Complete Reference	GROFF and JAMES