**CHAPTER 1**

**INTRODUCTION**

The population of the world is multiplying with each coming year and so are the diseases and health issues. With an increase in the population there is an increase in the need of blood. The growing population of the world results in a lot of potential blood donors. But in spite of this not more than 10% of the total world population participates in blood donation. With the growing population and the advancement in medical science the demand for blood has also increased. Due to the lack of communication between the blood donors and the blood recipients, most of the patients in need of blood do not get blood on time and hence lose their lives. There is a dire need of synchronization between the blood donors and hospitals and the blood banks. This improper management of blood leads to wastage of the available blood inventory. Improper communication and synchronization between the blood banks and hospitals leads to wastage of the blood available. These problems can be dealt with by automating the existing manual blood bank management system. A high-end, efficient, highly available and scalable system has to be developed to bridge the gap between the donors and the recipients and to reduce the efforts required to search for blood donors.

The number of persons who are in need of blood are increasing in large number day by day. In order to help people who are in need of blood, this Online Blood Bank can be used effectively for getting details of blood donors having the same blood group and within the same city. People who are interested in donating blood can register themselves online.

At present, the public can only know about the blood donation events through conventional media means such as radio, newspaper or television advertisements. There is no information regarding the blood donation programs available on any of the portal. The current system that is using by the blood bank is manual system. With the manual system, there are problems in managing the donors' records. The records of the donor might not be kept safely and there might be missing of donor's records due to human error or disasters. Besides that, errors might occur when the staff keeps more than one record for the same donor. There is no centralized database of volunteer donors. So, it becomes really tedious for a person to search blood in case of emergency. The only option is to manually search and match donors and then make phone calls to every donor. There is also no centralized database used to keep the donors' records. Each bank is having their own records of donors. If a donor makes donation in different hospital, no previous records can be traced except if the donor brings along the donation certificate. Hence, the donor is considered to be a first-timer if they make blood donation in a new place. Without an automated management system, there are also problems in keeping track of the actual amount of each and every blood type in the blood bank. In addition, there is also no alert available when the blood quantity is below its par level or when the blood in the bank has expired.

**CHAPTER 2**

**LITERATURE REVIEW**

**A STUDY ON BLOOD BANK MANAGEMENT**

A.Clemen Teena, K.Sankar and S.Kannan, IDOSI Publications, 2014

Blood Bank Information System will be an information management system which helps to manage the records of donors and patients at a blood bank. The system will allow the authorized blood bank officer to login using a secret password and easily manage the records of the blood donors and the patients in need of blood. The key features of the system are centralized database architecture, access to the system secured by login, search facility for finding blood donors based of various search criteria, search facility for finding Patients (acceptors) based of various search criteria, easy addition and updating of donor's and acceptors details.

**BLOOD BANK MANAGEMENT SYSTEM**

Prathamesh Raut, Prachi Parab, Yogesh Suthar, Sumeet Narwani

IRF International Conference, July 2015

A high-end system to bridge the gap between the blood donors and the people in need for blood. Application for Blood Bank Management System is a way to synchronize Blood banks and Hospitals with the help of Internet. It is a Web Application through which Registered Hospitals can check the availability of required Blood and can send Request for blood to the nearest blood bank or donor matching with blood requirement and can be ordered online as and when required. Blood bank can also send a request to another blood bank for unavailable blood. Person willing to donate blood can find out nearest blood banks using Blood Bank Management Android Application. The location of the blood bank can also be traced using maps. The Android application can be accessed only by the donors to search the blood donation centers and the requesting blood banks and hospitals to search the nearest blood banks and donors.

**BLOOD BANK MANAGEMENT INFORMATION SYSTEM**

Vikas Kulshreshtha, Sharad Maheswari, International Journal of Engineering Research and Applications, May 2016

A blood bank is a bank of blood or blood components, gathered as a result of blood donation, stored and preserved for later use in blood transfusion. To provide web based communication there are numbers of online web based blood bank management system exists for communicating between department of blood centers and hospitals, to satisfy blood necessity, to buy, sale and stock the blood, to give information about this blood. Manual systems as compared to Computer Based Information Systems are time consuming, laborious, and costly. This paper introduces the review of the main features, merits and demerits provided by the existing Web-Based Information System for Blood Banks. This study shows the comparison of various existing system and provide some more idea for improve the existing system.

**BENEFITS OF MANAGEMENT INFORMATION SYSTEM IN BLOOD BANK**

Vikas Kulshreshtha, Sharad Maheswari, International Journal of Engineering Research and Applications, December 2013

This paper discuss about the benefits of management information system in blood bank. Management information system plays an important role in the development of the projects. In today’s world of information most of the systems are transforming into the management information system. The paper is basically focused on the blood bank management information system. It discuss about the beneficiaries of the blood bank management information system. It plays a vital role in the blood bank as blood is the necessity of everyone.

**A SURVEY ON BLOOD BANK MANAGEMENT SYSTEM**

Animesh Tayal, Harshad Gahare, Akshay Patel, Sagar Jog, IJRTER 2016

In this paper they propose a new and efficient way to overcome such scenarios with our project. These records contain the information like Donor Name, Blood Group, Email Address, etc. After that your contact details will appear in alphabetical order on the screen; the urgent time of a blood requirement, you can quickly check for contacts matching a particular or related blood group and reach out to them through the Blood Bank Website. Blood Bank Management System provides list of donors in your city/area. Use this app in case of emergency. Since almost everyone carries a mobile phone, it ensures instant location tracking and communication. The location-based app, operational on android platform, will help users easily find donors of matching blood groups in their location and access their mobile numbers for instant help.

**A SURVEY PAPER ON E-BLOOD BANK**

Tushar Pandit, Satish Niloor, A.S. Shinde, International Journal of Computer Applications, March 2015

There are number of web based blood banks which are available for communication between blood bank and hospitals. None of the online blood bank offers the direct contact between donor and blood bank. This is the major drawback of the existing system. Existing systems are time consuming require more man power and it is costly. This paper introduces comparison between existing system and improved system. The new idea will improve the existing system and it will move from conventional desktop system to mobile system.

**CHAPTER 3**

**PROBLEM DESCRIPTION**

The existing blood bank management systems are window based application. These applications are time consuming because they involve use of one machine and also the records are stored one by one. With the manual system, there are problems in managing the donors' records. The records of the donor might not be kept safely and there might be missing of donor's records due to human error or disasters. Besides that, errors might occur when the staff keeps more than one record for the same donor. There is no centralized database of volunteer donors. So, it becomes really tedious for a person to search blood in case of emergency. The only option is to manually search and match donors and then contact them. To overcome this problem Online Blood Bank is proposed. The system keeps the record of all the donors, recipients over a centralized database. This system also has the ability to keep track of the donor's donation records and the blood stock in the blood bank. This project intends to computerize the blood and donor management system in a blood bank in order to improve the record management efficiency due to the grown size of records of data.

**CHAPTER 4**

**METHODOLOGY**

**4.1 DOTNET**

.NET is a general-purpose software enhancement platform, similar to Java. At its base is a virtual machine that turns intermediate language (IL) into machine code. High-level language compilers for C#, VB.NET, and C++ are provided to turn source code into IL. C# is a new programming language, very similar to Java. An extensive class library is included, featuring all the functionality one might expect from a contemporary development platform - windows GUI development (Windows Forms), database access (ADO.NET), web development (ASP.NET), web services, XML etc. It has five primary principles:

* Easier and Quick Programming.
* Reduced amount of code.
* Supports Software Lifecycle.
* It executes with high performance.
* Larger class library.

**4.2 .NET FRAMEWORK**

A set of technologies for developing and using the components to create web forms, web services, and window applications within the Microsoft environment by .NET framework. The framework offers a fundamental shift in Microsoft strategy which moves application development from client-centric to server-centric. Dot net languages do not comply with machine code. They are compiled to an Intermediate Language (IL).

**4.3 ASP .NET**

ASP.NET is a web development platform, which provides a programming model, a comprehensive software infrastructure and various services required to build up robust web applications for PC, as well as mobile devices.

ASP.NET works on top of the HTTP protocol, and uses the HTTP commands and policies to set a browser-to-server bilateral communication and cooperation.

ASP.NET is a part of Microsoft .Net platform. ASP.NET applications are compiled codes, written using the extensible and reusable components or objects present in .Net framework. These codes can use the entire hierarchy of classes in .Net framework.

ASP.NET is used to produce interactive, data-driven web applications over the internet. It consists of a large number of controls such as text boxes, buttons, and labels for assembling, configuring, and manipulating code to create HTML pages.

**4.3.1 ASP.NET WEB FORMS MODEL**

ASP.NET web forms extend the event-driven model of interaction to the web applications. The browser submits a web form to the web server and the server returns a full markup page or HTML page in response. All client side user activities are forwarded to the server for stateful processing. The server processes the output of the client actions and triggers the reactions. Now, HTTP is a stateless protocol. ASP.NET framework helps in storing the information regarding the state of the application, which consists of:

* Page state
* Session state

The page state is the state of the client, i.e., the content of various input fields in the web form. The session state is the collective information obtained from various pages the user visited and worked with, i.e., the overall session state. ASP.NET session state and server side infrastructure keeps track of the information collected globally over a session.

The ASP.NET runtime carries the page state to and from the server across page requests while generating ASP.NET runtime codes, and incorporates the state of the server side components in hidden fields. This way, the server becomes aware of the overall application state and operates in a two-tiered connected way.

**4.3.2 ASP.NET COMPONENT MODEL**

The ASP.NET component model provides various building blocks of ASP.NET pages. Basically it is an object model, which describes:

* Server side counterparts of almost all HTML elements or tags, such as <form> and <input>.
* Server controls, which help in developing complex user-interface. For example, the Calendar control or the Grid view control.

ASP.NET is a technology, which works on the .Net framework that contains all web-related functionalities. The .Net framework is made of an object-oriented hierarchy. An ASP.NET web application is made of pages. When a user requests an ASP.NET page, the IIS delegates the processing of the page to the ASP.NET runtime system. The ASP.NET runtime transforms the .aspx page into an instance of a class, which inherits from the base class page of the .Net framework. Therefore, each ASP.NET page is an object and all its components i.e., the server-side controls are also objects.

**4.4 COMPONENTS OF .NET FRAMEWORK**

**Common Language Runtime or CLR**

It performs memory management, exception handling, debugging, security checking, thread execution, code execution, code safety, verification, and compilation. The code that is directly managed by the CLR is called the managed code. When the managed code is compiled, the compiler converts the source code into a CPU independent intermediate language (IL) code. A Just In Time (JIT) compiler compiles the IL code into native code, which is CPU specific.

**.Net Framework Class Library**

It contains a huge library of reusable types. classes, interfaces, structures, and enumerated values, which are collectively called types.

**Common Language Specification**

It contains the specifications for the .Net supported languages and implementation of language integration.

**Common Type System**

It provides guidelines for declaring, using, and managing types at runtime, and cross-language communication.

**Metadata and Assemblies**

Metadata is the binary information describing the program, which is either stored in a portable executable file (PE) or in the memory. Assembly is a logical unit consisting of the assembly manifest, type metadata, IL code, and a set of resources like image files.

**Windows Forms**

Windows Forms contain the graphical representation of any window displayed in the application.

**ASP.NET and ASP.NET AJAX**

ASP.NET is the web development model and AJAX is an extension of ASP.NET for developing and implementing AJAX functionality. ASP.NET AJAX contains the components that allow the developer to update data on a website without a complete reload of the page.

**ADO.NET**

It is the technology used for working with data and databases. It provides access to data sources like SQL server, OLE DB, XML etc. The ADO.NET allows connection to data sources for retrieving, manipulating, and updating data.

**Windows Workflow Foundation (WF)**

It helps in building workflow-based applications in Windows. It contains activities, workflow runtime, workflow designer, and a rules engine.

**Windows Presentation Foundation**

It provides a separation between the user interface and the business logic. It helps in developing visually stunning interfaces using documents, media, two and three dimensional graphics, animations, and more.

**Windows Communication Foundation (WCF)**

It is the technology used for building and executing connected systems.

**Windows CardSpace**

It provides safety for accessing resources and sharing personal information on the internet.

**LINQ**

It imparts data querying capabilities to .Net languages using a syntax which is similar to the tradition query language SQL.

**4.5 ASP.NET APPLICATION LIFE CYCLE**

The application life cycle has the following stages:

* User makes a request for accessing application resource, a page. Browser sends this request to the web server.
* A unified pipeline receives the first request and the following events take place:
* An object of the class ApplicationManager is created.
* An object of the class HostingEnvironment is created to provide information regarding the resources.
* Top level items in the application are compiled.
* Response objects are created. The application objects such as HttpContext, HttpRequest and HttpResponse are created and initialized.
* An instance of the HttpApplication object is created and assigned to the request.
* The request is processed by the HttpApplication class. Different events are raised by this class for processing the request.

**4.6 ASP.NET PAGE LIFE CYCLE**

When a page is requested, it is loaded into the server memory, processed, and sent to the browser. Then it is unloaded from the memory. At each of these steps, methods and events are available, which could be overridden according to the need of the application. The Page class creates a hierarchical tree of all the controls on the page. All the components on the page, except the directives, are part of this control tree.

The page life cycle phases are:

* Initialization
* Instantiation of the controls on the page
* Restoration and maintenance of the state
* Execution of the event handler codes
* Page rendering

Following are the different stages of an ASP.NET page:

* **Page request** - When ASP.NET gets a page request, it decides whether to parse and compile the page, or there would be a cached version of the page; accordingly the response is sent.
* **Starting of page life cycle** - At this stage, the Request and Response objects are set. If the request is an old request or post back, the IsPostBack property of the page is set to true. The UICulture property of the page is also set.
* **Page initialization** - At this stage, the controls on the page are assigned unique ID by setting the UniqueID property and the themes are applied. For a new request, postback data is loaded and the control properties are restored to the view-state values.
* **Page load** - At this stage, control properties are set using the view state and control state values.
* **Validation** - Validate method of the validation control is called and on its successful execution, the IsValid property of the page is set to true.
* **Postback event handling** - If the request is a postback (old request), the related event handler is invoked.
* **Page rendering** - At this stage, view state for the page and all controls are saved. The page calls the Render method for each control and the output of rendering is written to the OutputStream class of the Response property of page.
* **Unload** - The rendered page is sent to the client and page properties, such as Response and Request, are unloaded and all cleanup done.

**4.7 ASP.NET PAGE LIFE CYCLE EVENTS**

At each stage of the page life cycle, the page raises some events, which could be coded. An event handler is basically a function or subroutine, bound to the event, using declarative attributes such as Onclick or handle.

Following are the page life cycle events:

* **PreInit** - PreInit is the first event in page life cycle. It checks the IsPostBack property and determines whether the page is a postback. It sets the themes and master pages, creates dynamic controls, and gets and sets profile property values. This event can be handled by overloading the OnPreInit method or creating a Page\_PreInit handler.
* **Init** - Init event initializes the control property and the control tree is built. This event can be handled by overloading the OnInit method or creating a Page\_Init handler.
* **InitComplete** - InitComplete event allows tracking of view state. All the controls turn on view-state tracking.
* **LoadViewState** - LoadViewState event allows loading view state information into the controls.
* **LoadPostData** - During this phase, the contents of all the input fields are defined with the <form> tag are processed.
* **PreLoad** - PreLoad occurs before the post back data is loaded in the controls. This event can be handled by overloading the OnPreLoad method or creating a Page\_PreLoad handler.
* **Load** - The Load event is raised for the page first and then recursively for all child controls. The controls in the control tree are created. This event can be handled by overloading the OnLoad method or creating a Page\_Load handler.
* **LoadComplete** - The loading process is completed, control event handlers are run, and page validation takes place. This event can be handled by overloading the OnLoadComplete method or creating a Page\_LoadComplete handler
* **PreRender** - The PreRender event occurs just before the output is rendered. By handling this event, pages and controls can perform any updates before the output is rendered.
* **PreRenderComplete** - As the PreRender event is recursively fired for all child controls, this event ensures the completion of the pre-rendering phase.
* **SaveStateComplete** - State of control on the page is saved. Personalization, control state and view state information is saved. The HTML markup is generated. This stage can be handled by overriding the Render method or creating a Page\_Render handler.
* **UnLoad** - The UnLoad phase is the last phase of the page life cycle. It raises the UnLoad event for all controls recursively and lastly for the page itself. Final cleanup is done and all resources and references, such as database connections, are freed. This event can be handled by modifying the OnUnLoad method or creating a Page\_UnLoad handler.

**4.8 MVC FRAMEWORK**

The **Model-View-Controller (MVC)** is an architectural pattern that separates an application into three main logical components: the **model**, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

**4.8.1 MVC COMPONENTS**

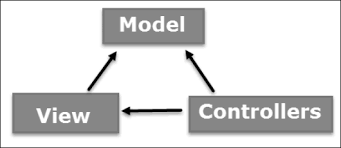


Figure 4.1 MVC Components

### Model

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

### View

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

### Controller

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

### 4.8.2 ASP.NET MVC FEATURES

ASP.NET MVC provides the following features

* Ideal for developing complex but lightweight applications.
* Provides an extensible and pluggable framework, which can be easily replaced and customized. For example, if you do not wish to use the in-built Razor or ASPX View Engine, then you can use any other third-party view engines or even customize the existing ones.
* Utilizes the component-based design of the application by logically dividing it into Model, View, and Controller components. This enables the developers to manage the complexity of large-scale projects and work on individual components.
* MVC structure enhances the test-driven development and testability of the application, since all the components can be designed interface-based and tested using mock objects.
* Does not use the concept of View State (which is present in ASP.NET). This helps in building applications, which are lightweight and gives full control to the developers.

**4.8.3 MVC FLOW DIAGRAM**

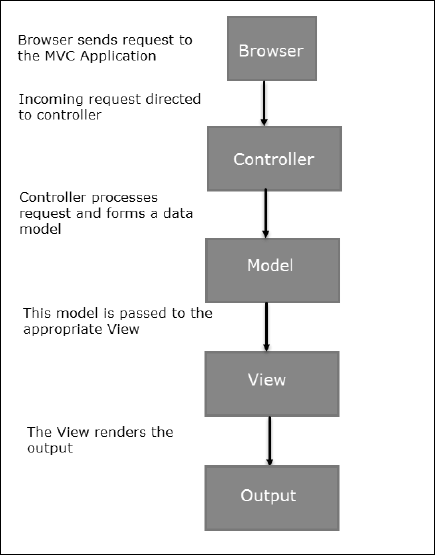


Figure 4.2 MVC Flow Diagram

**4.8.4 MVC FLOW STEPS**

* The client browser sends request to the MVC Application.
* Global.ascx receives this request and performs routing based on the URL of the incoming request using the RouteTable, RouteData, UrlRoutingModule and MvcRouteHandler objects.
* This routing operation calls the appropriate controller and executes it using the IControllerFactory object and MvcHandler object's Execute method.
* The Controller processes the data using Model and invokes the appropriate method using ControllerActionInvoker object
* The processed Model is then passed to the View, which in turn renders the final output.

**4.9 MICROSOFT SQL SERVER**

MS SQL Server is a relational database management system (RDBMS) developed by Microsoft. This product is built for the basic function of storing retrieving data as required by other applications. It can be run either on the same computer or on another across a network.

**4.9.1 USAGE OF SQL SERVER**

* To create databases.
* To maintain databases.
* To analyze the data through SQL Server Analysis Services (SSAS).
* To generate reports through SQL Server Reporting Services (SSRS).
* To carry out ETL operations through SQL Server Integration Services (SSIS).

## 4.9.2 SQL SERVER COMPONENTS

SQL Server works in client-server architecture, hence it supports two types of components − (a) Workstation and (b) Server.

* **Workstation components** are installed in every device/SQL Server operator’s machine. These are just interfaces to interact with Server components. Example: SSMS, SSCM, Profiler, BIDS, SQLEM etc.
* **Server components** are installed in centralized server. These are services. Example: SQL Server, SQL Server Agent, SSIS, SSAS, SSRS, SQL browser, SQL Server full text search etc.

**4.9.3 ARCHITECTURE OF SQL SERVER**

Thearchitecture of SQL Server into the following parts

* General architecture
* Memory architecture
* Data file architecture
* Log file architecture

## GENERAL ARCHITECTURE

**Client** − Where the request initiated.

**Query** − SQL query which is high level language.

**Logical Units** − Keywords, expressions and operators, etc.

**N/W Packets** − Network related code.

**Protocols** − In SQL Server we have 4 protocols.

* Shared memory (for local connections and troubleshooting purpose).
* Named pipes (for connections which are in LAN connectivity).
* TCP/IP (for connections which are in WAN connectivity).
* VIA-Virtual Interface Adapter (requires special hardware to set up by vendor and also deprecated from SQL 2012 version).

**Server** − Where SQL Services got installed and databases reside.

**Relational Engine** − This is where real execution will be done. It contains Query parser, Query optimizer and Query executor.

**Query Parser (Command Parser) and Compiler (Translator)** − This will check syntax of the query and it will convert the query to machine language.

**Query Optimizer** − It will prepare the execution plan as output by taking query, statistics and Algebrizer tree as input.

**Execution Plan** − It is like a roadmap, which contains the order of all the steps to be performed as part of the query execution.

**Query Executor** − This is where the query will be executed step by step with the help of execution plan and also the storage engine will be contacted.

**Storage Engine** − It is responsible for storage and retrieval of data on the storage system (disk, SAN, etc.,), data manipulation, locking and managing transactions.

**SQL OS** − This lies between the host machine (Windows OS) and SQL Server. All the activities performed on database engine are taken care of by SQL OS. SQL OS provides various operating system services, such as memory management deals with buffer pool, log buffer and deadlock detection using the blocking and locking structure.

**Checkpoint Process** − Checkpoint is an internal process that writes all dirty pages (modified pages) from Buffer Cache to Physical disk. Apart from this, it also writes the log records from log buffer to physical file. Writing of Dirty pages from buffer cache to data file is also known as Hardening of dirty pages.

It is a dedicated process and runs automatically by SQL Server at specific intervals. SQL Server runs checkpoint process for each database individually. Checkpoint helps to reduce the recovery time for SQL Server in the event of unexpected shutdown or system crash\Failure.

## MEMORY ARCHITECTURE

One of the primary design goals of all database software is to minimize disk I/O because disk reads and writes are among the most resource-intensive operations. Memory in windows can be called with Virtual Address Space, shared by Kernel mode (OS mode) and User mode (Application like SQL Server).SQL Server "User address space" is broken into two regions: MemToLeave and Buffer Pool. Size of MemToLeave (MTL) and Buffer Pool (BPool) is determined by SQL Server during startup.

**Buffer management** is a key component in achieving I/O highly efficiency. The buffer management component consists of two mechanisms: the buffer manager to access and update database pages, and the buffer pool to reduce database file I/O.The buffer pool is further divided into multiple sections. The most important ones being the buffer cache (also referred to as data cache) and procedure cache.

Buffer cache holds the data pages in memory so that frequently accessed data can be retrieved from cache. The alternative would be reading data pages from the disk. Reading data pages from cache optimizes performance by minimizing the number of required I/O operations which are inherently slower than retrieving data from the memory. Procedure cache keeps the stored procedure and query execution plans to minimize the number of times that query plans have to be generated.

## DATA FILE ARCHITECTURE

Data File architecture has the following components

### File Groups

Database files can be grouped together in file groups for allocation and administration purposes. No file can be a member of more than one file group. Log files are never part of a file group. Log space is managed separately from data space. There are two types of file groups in SQL Server, Primary and User-defined. Primary file group contains the primary data file and any other files not specifically assigned to another file group. All pages for the system tables are allocated in the primary file group. User-defined file groups are any file groups specified using the file group keyword in create database or alter database statement.

One file group in each database operates as the default file group. When SQL Server allocates a page to a table or index for which no file group was specified when they were created, the pages are allocated from default file group. To switch the default file group from one file group to another file group, it should have db\_owner fixed db role.By default, primary file group is the default file group. User should have db\_owner fixed database role in order to take backup of files and file groups individually.

### Files

Databases have three types of files - Primary data file, Secondary data file, and Log file. Primary data file is the starting point of the database and points to the other files in the database.

Every database has one primary data file. We can give any extension for the primary data file but the recommended extension is **.**mdf. Secondary data file is a file other than the primary data file in that database. Some databases may have multiple secondary data files. Some databases may not have a single secondary data file. Recommended extension for secondary data file is .ndf.

Log files hold all of the log information used to recover the database. Database must have at least one log file. We can have multiple log files for one database. The recommended extension for log file is .ldf.

The location of all the files in a database are recorded in both master database and the primary file for the database. Most of the time, the database engine uses the file location from the master database.

### Events

Extents are basic unit in which space is allocated to tables and indexes. An extent is 8 contiguous pages or 64KB. SQL Server has two types of extents - Uniform and Mixed. Uniform extents are made up of only single object. Mixed extents are shared by up to eight objects.

### Pages

It is the fundamental unit of data storage in MS SQL Server. The size of the page is 8KB. The start of each page is 96 byte header used to store system information such as type of page, amount of free space on the page and object id of the object owning the page.

## LOG FILE ARCHITECTURE

The SQL Server transaction log operates logically as if the transaction log is a string of log records. Each log record is identified by Log Sequence Number (LSN). Each log record contains the ID of the transaction that it belongs to.

Log records for data modifications record either the logical operation performed or they record the before and after images of the modified data. The before image is a copy of the data before the operation is performed; the after image is a copy of the data after the operation has been performed.

The steps to recover an operation depend on the type of log record

* Logical operation logged.
  + To roll the logical operation forward, the operation is performed again.
  + To roll the logical operation back, the reverse logical operation is performed.
* Before and after image logged.
  + To roll the operation forward, the after image is applied.
  + To roll the operation back, the before image is applied.

Different types of operations are recorded in the transaction log. These operations include

* The start and end of each transaction.
* Every data modification (insert, update, or delete). This includes changes by system stored procedures or data definition language (DDL) statements to any table, including system tables.
* Every extent and page allocation or de allocation.
* Creating or dropping a table or index.

Rollback operations are also logged. Each transaction reserves space on the transaction log to make sure that enough log space exists to support a rollback that is caused by either an explicit rollback statement or if an error is encountered. This reserved space is freed when the transaction is completed.

The section of the log file from the first log record that must be present for a successful database-wide rollback to the last-written log record is called the active part of the log, or the active log. This is the section of the log required to a full recovery of the database. No part of the active log can ever be truncated. LSN of this first log record is known as the minimum recovery LSN (Min LSN).

The SQL Server Database Engine divides each physical log file internally into a number of virtual log files. Virtual log files have no fixed size, and there is no fixed number of virtual log files for a physical log file.

The Database Engine chooses the size of the virtual log files dynamically while it is creating or extending log files. The Database Engine tries to maintain a small number of virtual files. The size or number of virtual log files cannot be configured or set by administrators. The only time virtual log files affect system performance is if the physical log files are defined by small size and growth\_increment values.

The size value is the initial size for the log file and the growth\_increment value is the amount of space added to the file every time new space is required. If the log files grow to a large size because of many small increments, they will have many virtual log files. This can slow down database startup and also log backup and restore operations.

**CHAPTER 5**

**PRESENT WORK**

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well-running system. Systems design could see it as application of systems theory to product development. There is some overlap with the discipline of the system analysis, architecture and engineering. Conceptual design is not the end of design process. It serves as the basis for a brief design. System design implies a systematic and rigorous approach to the design as demanded by complexity of many system problems.

**5.1 UNIFIED MODELING LANGUAGE**

UML can be described as the successor of object oriented analysis and design. An object contains both data and methods that control the data. The data represents the state of the object. A class describes an object and they also form hierarchy to model real world system. The hierarchy is represented as inheritance and the classes can also be associated in different manners as per the requirement. The objects are the real world entities that exist around us and the basic concepts like abstraction, encapsulation, inheritance, polymorphism all can be represented using UML. So UML is powerful enough to represent all the concepts exists in object oriented analysis and design. UML diagrams are representation of object oriented concept.

**5.1.1 USE CASE DIAGRAM**

Use case diagrams are a set of use cases, actors and their relationships. They represent the use case view of a system. A use case represents a particular functionality of a system. So use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors.

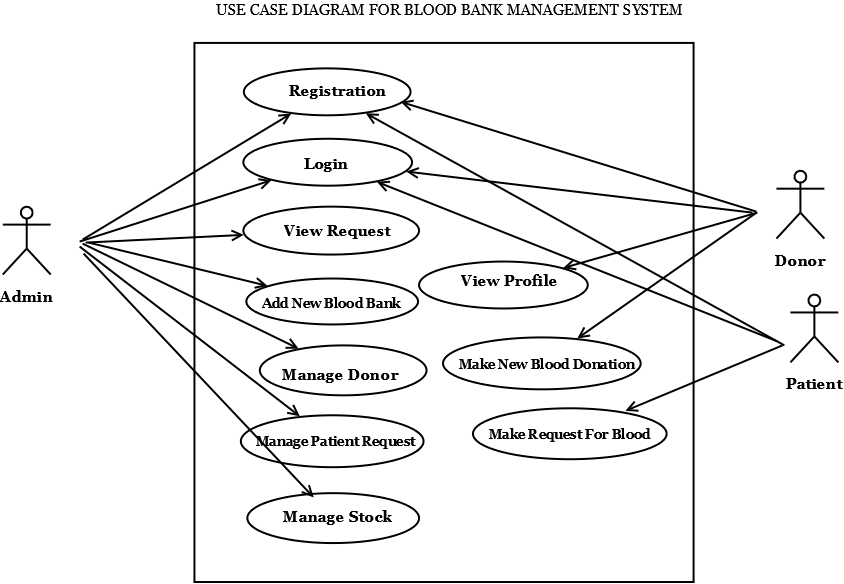
****

Figure 5.1 Use Case Diagram for Blood Bank Management System

**5.1.2** **ACTIVITY DIAGRAM**

Activity diagram describes the flow of control in a system. So it consists of activities and links. The flow can be sequential, concurrent or branched. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system. Activity diagrams are used to visualize the flow of controls in a system. This is prepared to have an idea of how the system will work when executed.

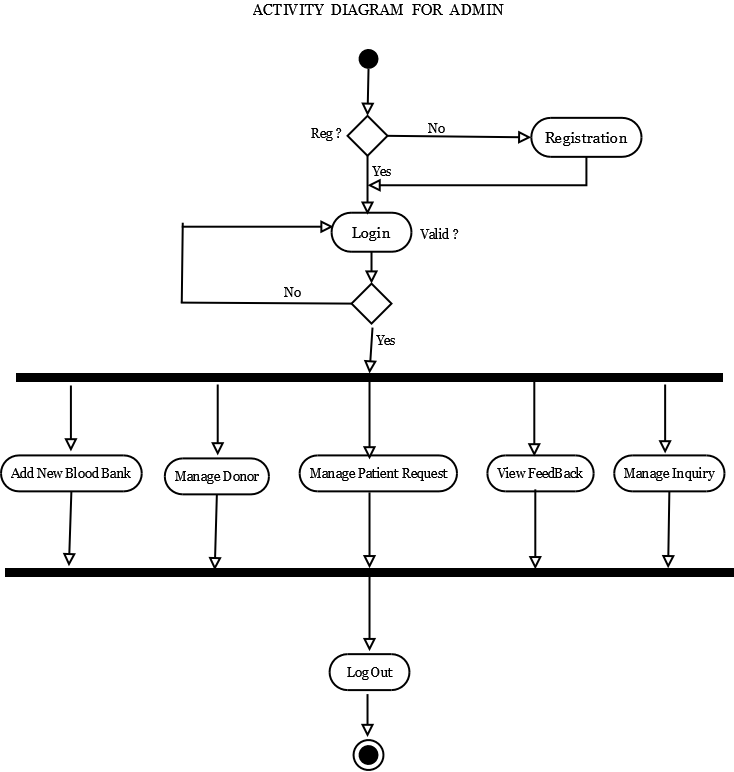


Figure 5.2 Activity Diagram for Admin

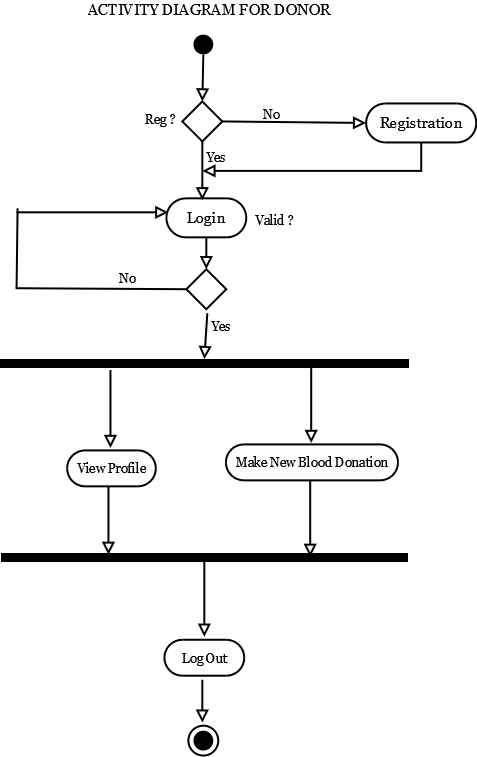


Figure 5.3 Activity Diagram for Donor

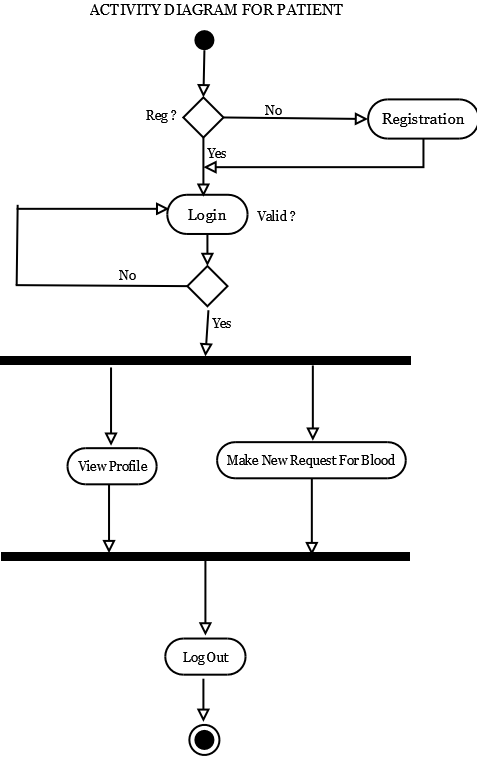
****

Figure 5.4 Activity Diagram for Patient

**5.1.3 CLASS DIAGRAM**

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations and collaboration. Class diagrams basically represent the object oriented view of a system which is static in nature. Active class is used in a class diagram to represent the concurrency of the system. Class diagram represents the object orientation of a system. So it is generally used for development purpose. This is the most widely used diagram at the time of system construction.

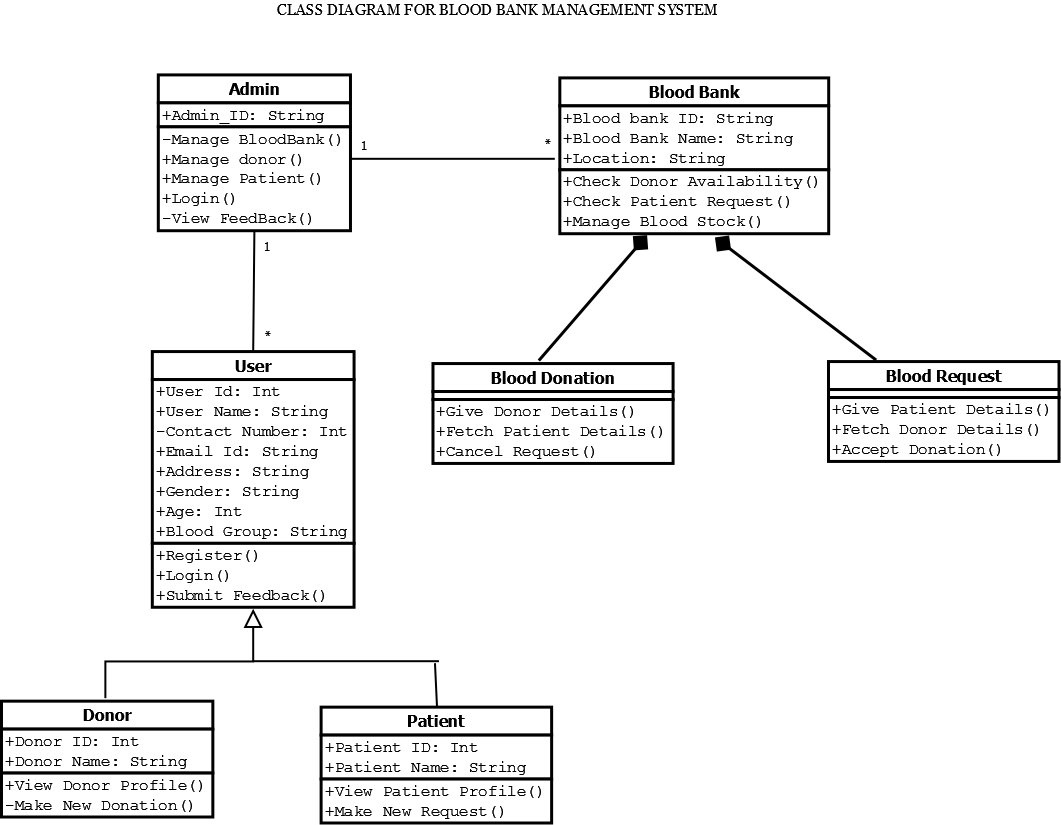


Figure 5.5 Class Diagram for Blood Bank Management System

**5.1.4 SEQUENCE DIAGRAM**

A sequence diagram is an interaction diagram. From the name it is clear that the diagram deals with some sequences, which are the sequence of messages flowing from one object to another. Interaction among the components of a system is very important from implementation and execution perspective. So Sequence diagram is used to visualize the sequence of calls in a system to perform a specific functionality.

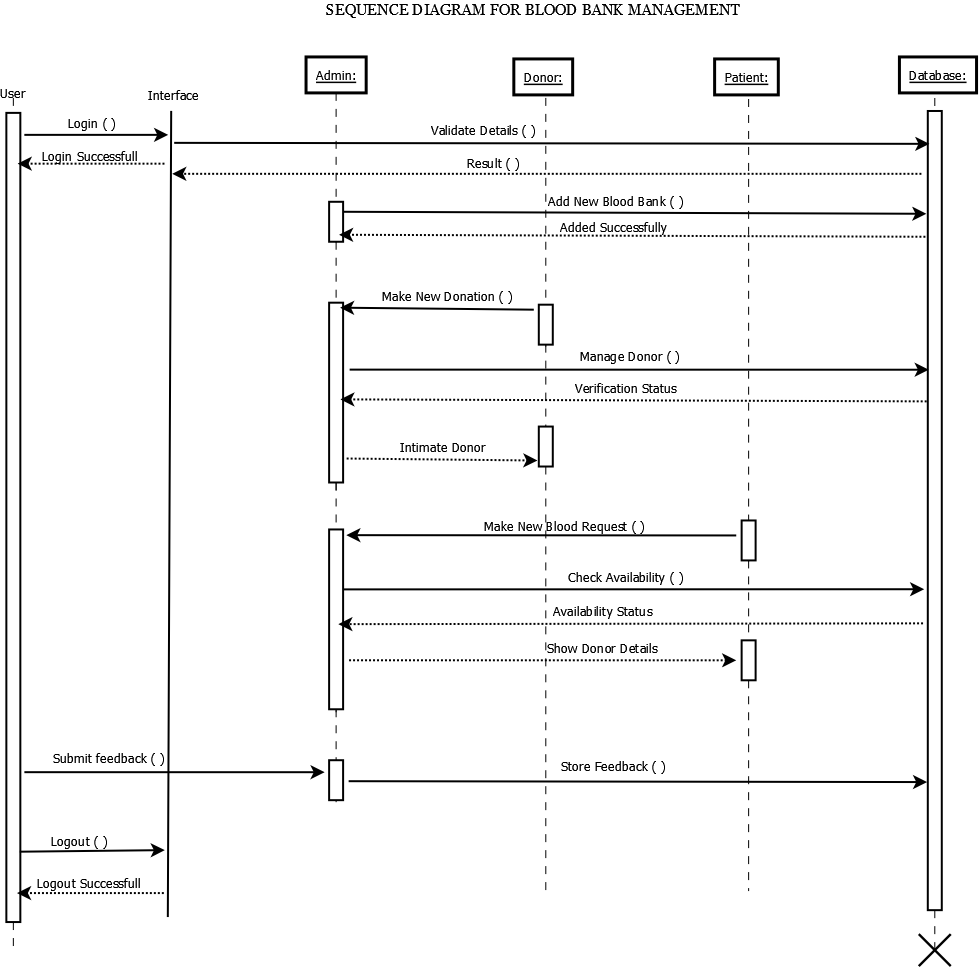


Figure 5.6 Sequence Diagram for Blood Bank Management System

**5.2 ENTITY RELATIONSHIP DIAGRAM**

Entity-relationship (ER) diagram is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems.

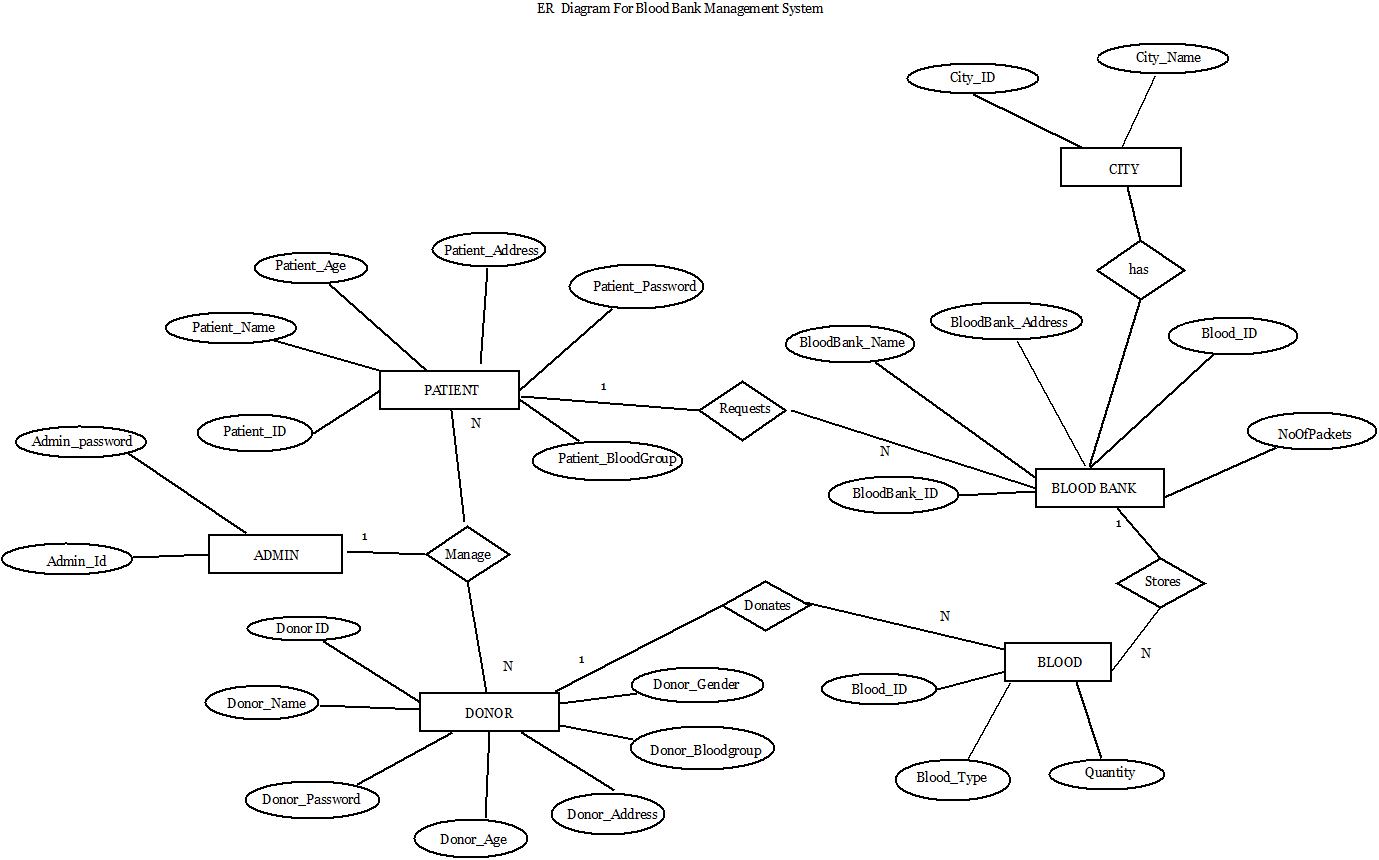
****

Figure 5.7 ER Diagram for Blood Bank Management System

**CHAPTER 6**

**RESULTS AND DISCUSSIONS**

****

Figure 6.1 Home Page

****

Figure 6.2 Blood Tips

****

Figure 6.3 Search Donor

****

Figure 6.4 Searched Donor Results

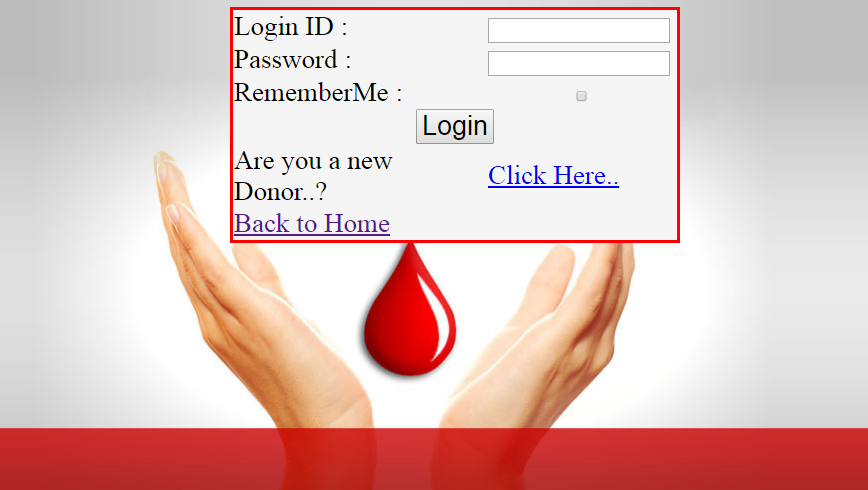
****

Figure 6.5 Donor Login

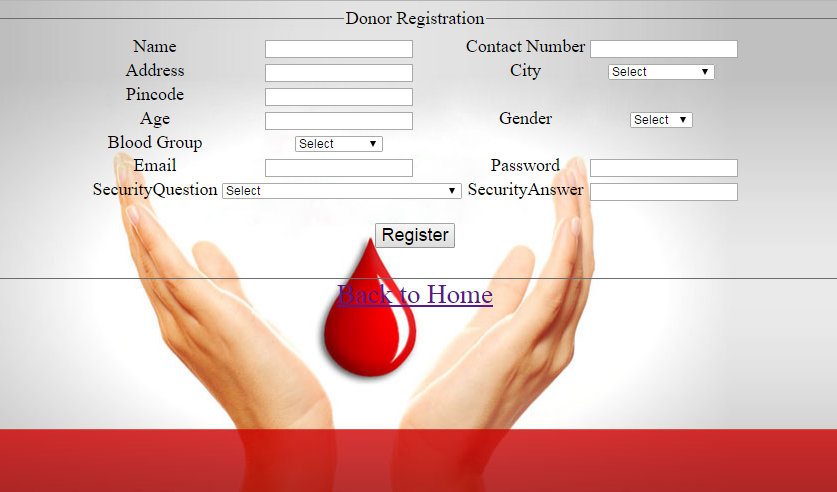
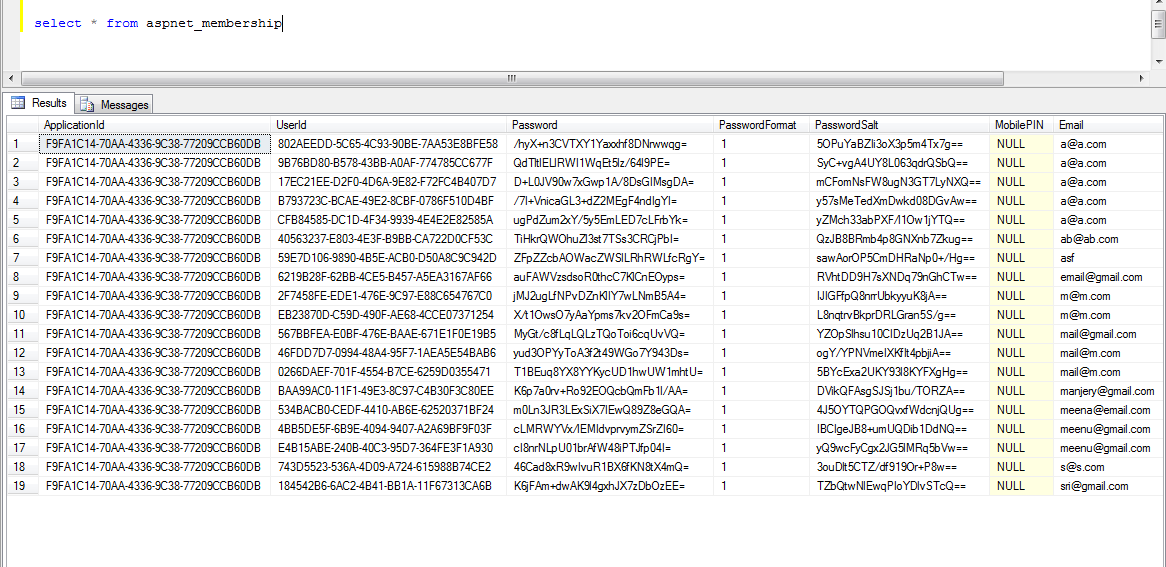
****

Figure 6.6 Donor Registration

****

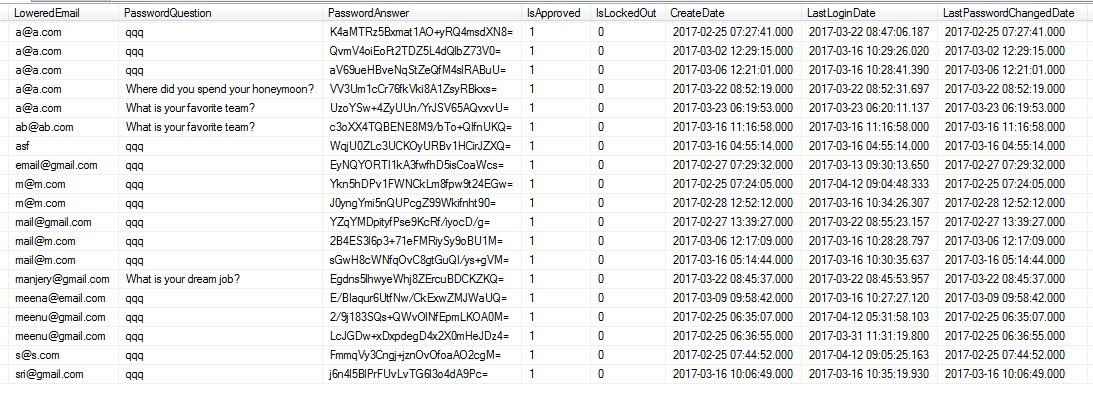
****

Figure 6.7 Membership Table

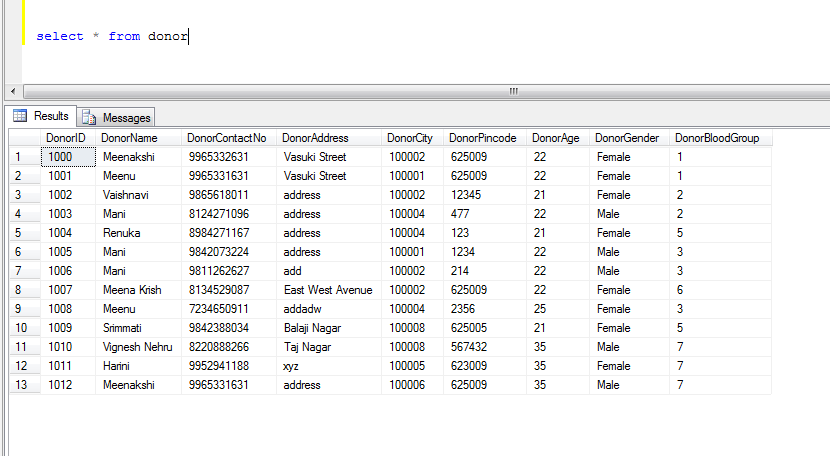
****

Figure 6.8 Donor Table

****

Figure 6.9 Donor My Profile

****

Figure 6.10 Donor Make Donation

****

Figure 6.11 Donor My Donation

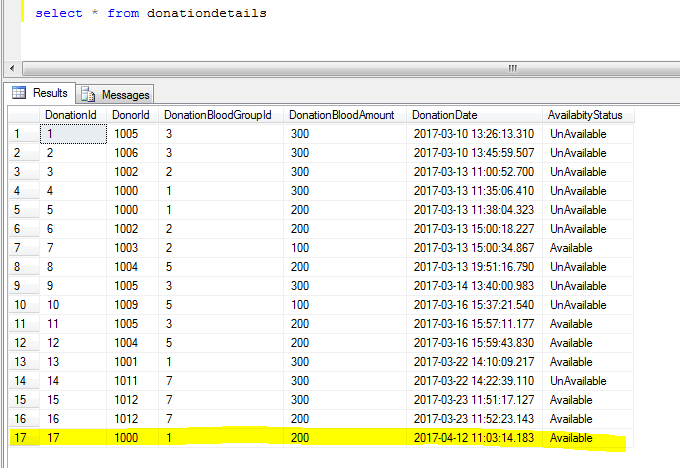
****

Figure 6.12 Donation Details Table

****

Figure 6.13 Hospital Login

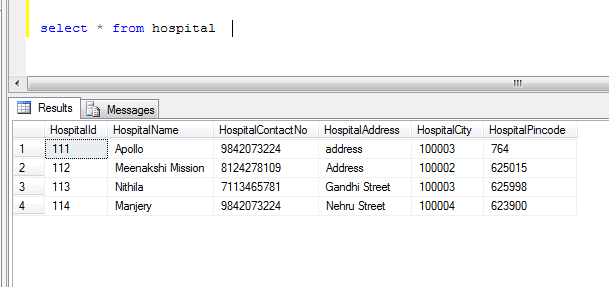
****

Figure 6.14 Hospital Table

****

Figure 6.15 Hospital My Profile

****

Figure 6.16 Hospital Make Request

****

Figure 6.17 Hospital View Request Made

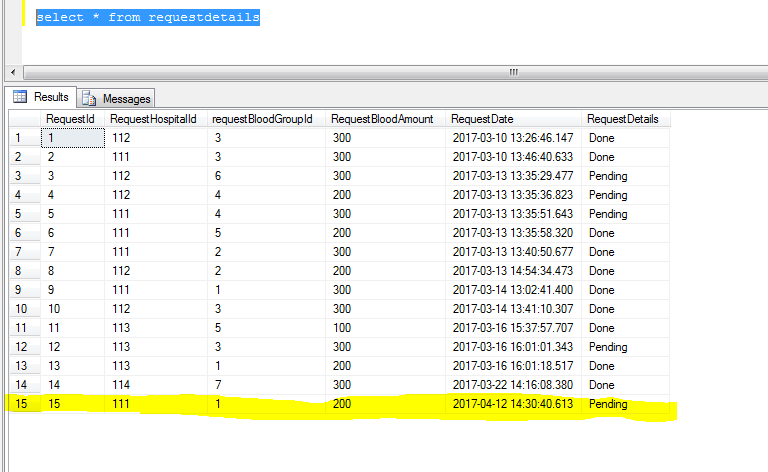
****

Figure 6.18 Request Details Table

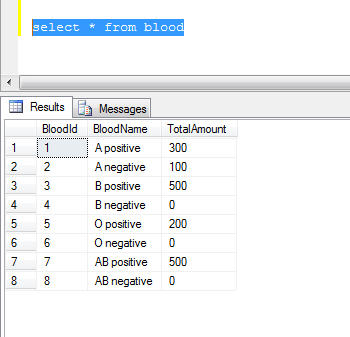
****

Figure 6.19 Blood Table

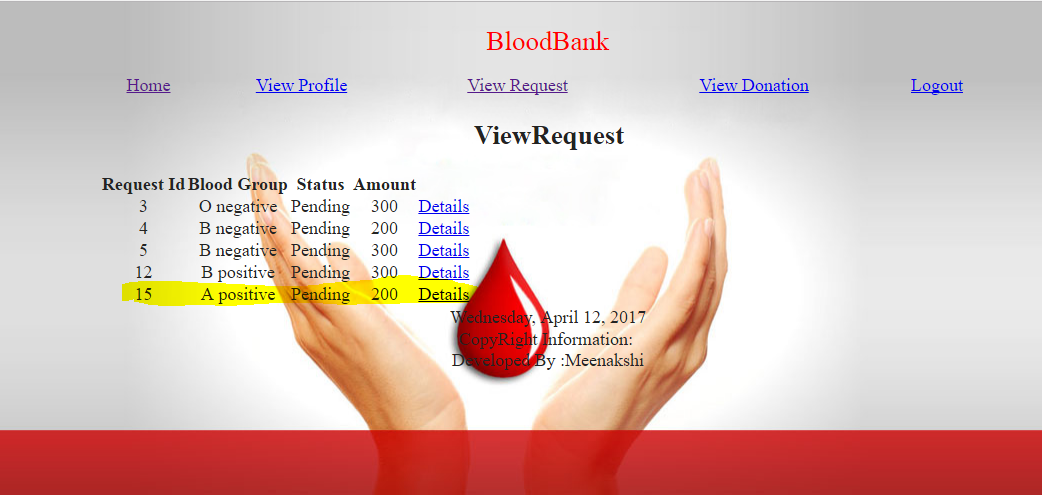
****

Figure 6.20 Blood Bank View Request

****

Figure 6.21 Blood Bank View Request Details

****

Figure 6.22 Blood Bank Check Availability

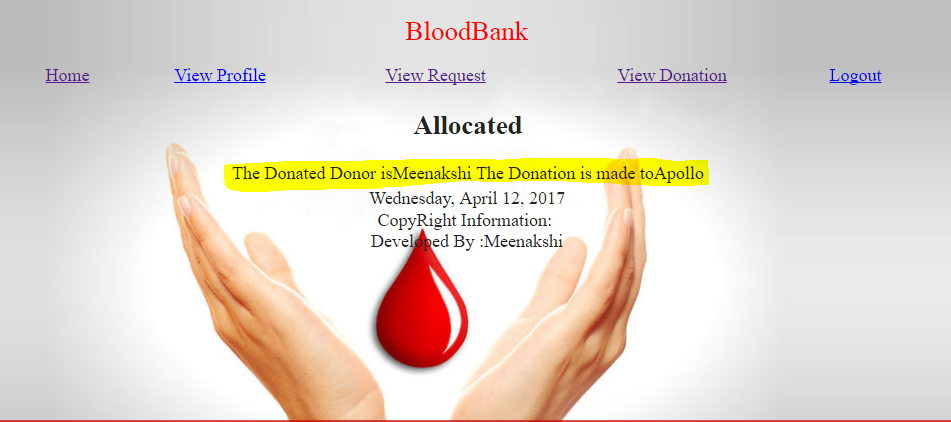
****

Figure 6.23 Blood Bank Allocation

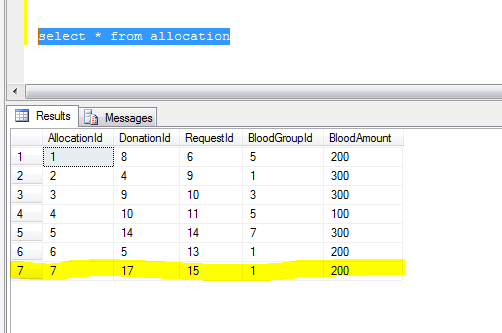
****

Figure 6.24 Allocation Table

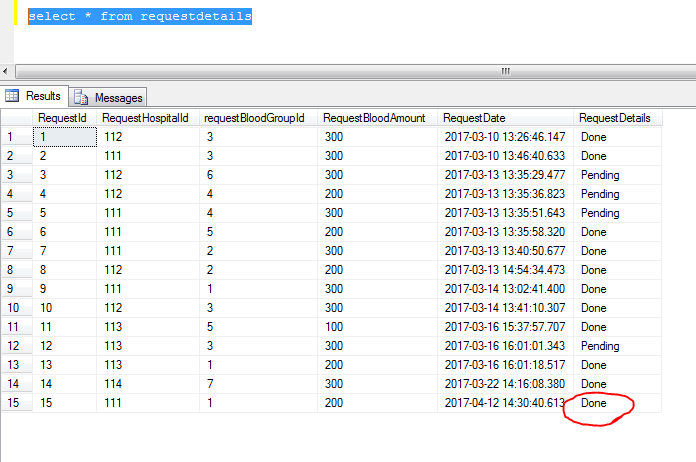
****

Figure 6.25 Triggered Request Details Table

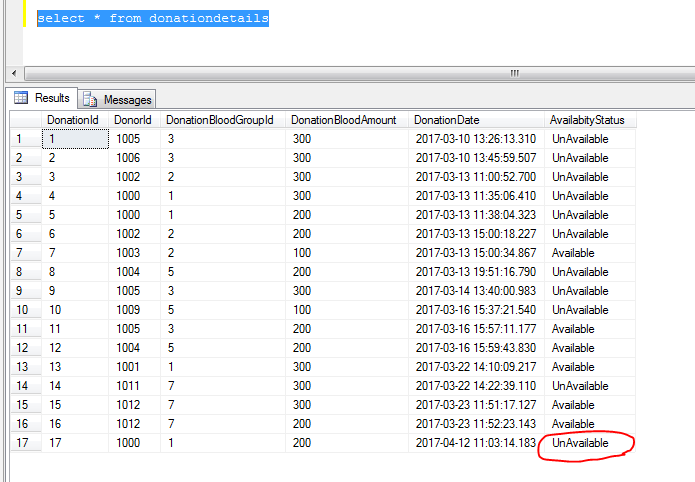
****

Figure 6.26 Triggered Donation Details Table

**CHAPTER 7**

**CONCLUSION AND FUTURE WORK**

The Online Blood Bank bridges the gap between the blood donors and the people in need for blood. This provides a way to synchronize Blood banks and Hospitals with the help of Internet. This web based system simplifies and automates the process of searching for blood in case of emergency and maintain the records of blood donors, recipients, blood donation programs and blood stocks in the bank. It provides reliable security measures which protect the data and the package from accidental of deliberate threats that could cause unauthorized modifications, disclosures of destruction of the data and protection of the information system by the use of password. This online blood bank reduces the time required to deliver required blood to the needy in cases of emergency. In future, more security features can be incorporated in the application to make it more secure and trustworthy.

**REFERENCES**

[1] A.Clemen Teena, K.Sankar and S.Kannan, A study on blood bank management, IDOSI Publications, 2014

[2] Animesh Tayal, Harshad Gahare, Akshay Patel, Sagar Jog, A Survey on Blood Bank Management System, IJRTER 2016

[3] Prathamesh Raut, Prachi Parab, Yogesh Suthar, Sumeet Narwan, Blood Bank Management System, IRF International Conference, July 2015

[4] Tushar Pandit, Satish Niloor, A.S. Shinde, A Survey Paper on E-Blood Bank , International Journal of Computer Applications, March 2015

[5] Vikas Kulshreshtha, Sharad Maheswari, Blood Bank Management Information System, International Journal of Engineering Research and Applications, May 2016

[6] Vikas Kulshreshtha, Sharad Maheswari, Benefits Of Management Information System in Blood Bank, International Journal of Engineering Research and Applications, December 2013

**WEB REFERENCES**