CHITKARA UNIVERSITY

TRAINING REPORT

OF

SIX MONTHS INDUSTRIAL TRAINING, UNDERTAKEN

AT

iNautix Technologies India Pvt Limited

IN

Learning and Development

ON

AAYL Project

SUBMITTED IN PARTIAL FULFILLMENT OF THE DEGREE

OF

BE (CSE)

Under the Guidance of:
Name: Ms Payal
Department: CSE

Submitted By:
Name: Divya
University ID No.: B130010163

Acknowledgement

It is my pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behaviour and acts during the course of study.

The internship opportunity I had with iNautix Technologies India Pvt Ltd. was a great chance for learning and professional development. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it. I am also grateful for having a chance to meet so many wonderful people and professionals who led me though this internship period

Bearing in mind previous I am using this opportunity to express my deepest gratitude and special thanks to my manager Mr. Gangadhar who in spite of being extraordinarily busy with his duties, took time out to hear, guide and keep me on the correct path and allowing me to carry out my project at their esteemed organization and extending during the training.

I am thankful to “Ms Payal” for his support, cooperation, and motivation provided to me during the training for constant inspiration, presence and blessings.

Lastly, I would like to thank the almighty and my parents for their moral support and friends with whom I shared my day-to day experience and received lots of suggestions that improved my quality of work.

Divya
B130010003

PREFACE

My project "Blood Donation Management System" is innovated to make a working web application on a project in order to create an e-Information about the donor and organization that are related to donating the blood. Through this application any person who is interested in donating the blood can register himself in the same way if any organization wants to register itself with this site that can also register. Moreover if any general consumer wants to make request blood online he can also take the help of this site. Blood Bank is the main authority who can do addition, deletion, and modification if required. This web application is made with the help of JAVA, Spring, JSP and Oracle.

CONTENTS

1. Project Undertaken

1.1 Objective

The Blood Donation Management System project is programmed in order to help the humans or patients who are seeking blood at a particular location. This project is designed in such a way that it keeps detailed information as well as separate information of all the locations where the blood is available and what kind of blood is available and in how much quantity.

The Blood Donation Management System does not store blood but it stores the information about the blood or more precisely we can say it store the information or database of the blood available in the particular location. Because there was a time when some needs bloods in urgent, then this software proved to be his best friend and help the person finding the place nearby him very quickly.

The system is basically an E-information system for getting the database for the blood availability in any particular arena.

1.2 Need to choose the project

- To understand the basics of the JSP, Java and to help the job seekers for the jobs and the companies in their recruitment process.
- To understand how to make the front-end.
- To understand the working of HTML, CSS, JS.
- To understand the working of JDBC.
- How to make API's.
- How to work with the SQL in a live working project.

1.3 Industry Application

Our organization iNautix, A BNY Mellon company will get help by this project in various ways:

For Company:

- Can be used as a way of social service.
- Easy way to help society.
- A way to contribute in society on professional level.

For Employees:

- Can be used as a way of social service.
- Easy way to help society.
- They can recommend people in case of emergency.
- Very beneficial on personal level.

2. *Introduction to assigned job*

2.1 Purpose

The Blood Donation management System is to create an e-Information about the donor and organization that are related to donating the blood. Through this application any person who is interested in donating the blood can register himself in the same way if any organization wants to register itself with this site that can also register. Moreover if any general consumer wants to make request blood online he can also take the help of this site. Blood Bank is the main authority who can do addition, deletion, and modification if required.

Blood Donation management System project is aimed to developing an online Blood Donation Information. The entire Online Blood Donation management System project has been developed keeping in view of the distributed client server computing technology, in mind.

2.2 Scope

- **Existing System**

In existing Blood Donation Management System, not all users can get access to the information because of the low working of the application or is not able to access any site. Sometimes the information is not updated or available for a particular place. In existing system the security is less and latest updates and uploads are not so frequent

- **Proposed System**

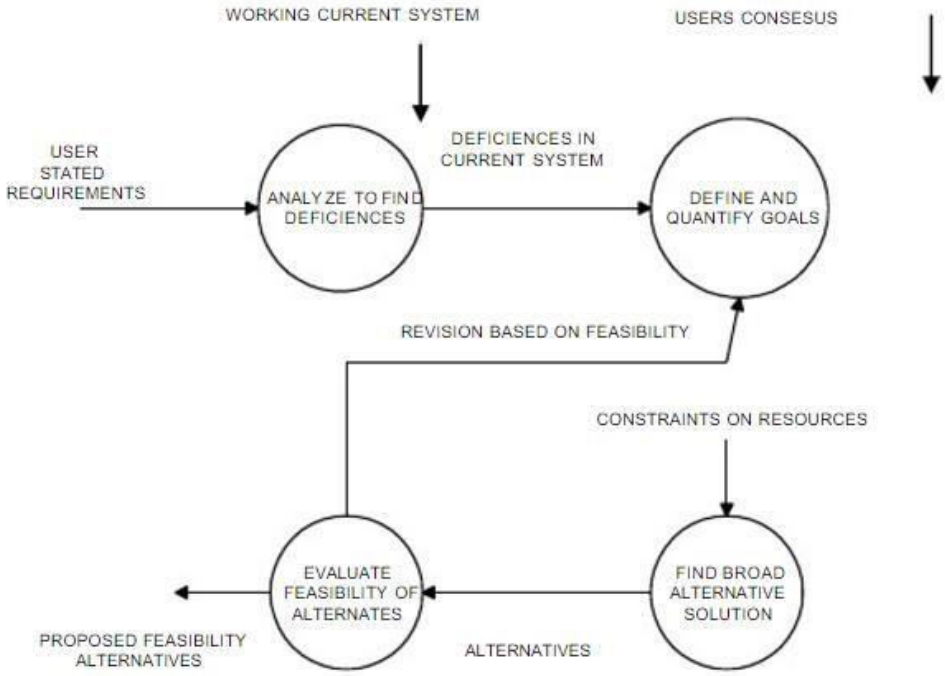
In the proposed Blood Donation Management System, in this software once the timer is being arranged, it put up updates and uploads automatically and does not need anyone to do so. Also it is easily available due to its speed and programming part and using it is quite an easy task and well as due to its speed the information which will be available by one or two clicks, will get available in few seconds only

2.3 Individual Job

As a fresher firstly they gave me an external training by an internal trainer on **Core Java**, **Advance Java (Servlet-JSP, Spring** and **Restful-API), HTML, CSS, JS, Angular, SQL** also they taught us basic of **JDBC** this training was of 3 months and they took our tests. After that they give us different projects iNautix projects. They tell us to use Eclipse IDE for that. They also giving us training of what type of models they used for service and what for development purpose. They are also giving us small trainings of Business, English, and Market training also.

3. ***Feasibility Study***

Feasibility study is the process of determination of whether or not a project is worth doing. Feasibility studies are undertaken within tight time constraints and normally culminate in a written and oral feasibility report. The contents and recommendations of this feasibility study helped us as a sound basis for deciding how to precede the project. It helped in taking decisions such as which software to use, hardware combinations, etc. The following is the process diagram for feasibility analysis. In the diagram, the feasibility analysis starts with the user set of requirements. With this, the existing system is also observed. The next step is to check for the deficiencies in the existing system. By evaluating the above points, a fresh idea is conceived to define and quantify the required goals. Besides that, a set of alternatives and their feasibility is also considered incise of any failure in the proposed system. Thus, feasibility study is an important part in software development.



PROCESS DIAGRAM FOR FEASIBILITY ANALYSIS

In the SDLC (Systems Development Life Cycle) of our project we maintained a number of feasibility checkpoints between the two phases of the SDLC. These checkpoints indicate that the management decision to be made after a phase is complete. The feasibility checkpoints in our project were as follows:

- Survey phase checkpoint
- Study phase checkpoint
- Selection phase checkpoint
- Acquisition phase checkpoint
- Design phase checkpoint

3.1 Technical Feasibility

Technical feasibility determines whether the work for the project can be done with the existing equipment, software technology and available personnel. Technical feasibility is concerned with specifying equipment and software that will satisfy the user requirement. This project is feasible on technical remarks also, as the proposed system is more beneficiary in terms of having a sound proof system with new technical components installed on the system. The proposed system can run on any machines supporting Windows and Internet services and works on the best software and hardware that had been used while designing the system so it would be feasible in all technical terms of feasibility. The technologies such as JAVA (JSP, Servlet), JavaScript and the compatible

H/W's are so familiar with the today's knowledge based industry that anyone can easily be compatible to the proposed environment

Technical Feasibility Addresses Three Major Issues:

- **Is the proposed Technology or Solution Practical?**
  The technologies used are matured enough so that they can be applied to our problems. The practicality of the solution we have developed is proved with the use of the technologies we have chosen. The technologies such as JAVA (JSP, Servlet), JavaScript and the compatible H/W's are so familiar with the today's knowledge based industry that anyone can easily be compatible to the proposed environment.

- **Do we currently possess the necessary technology?**
  We first make sure that whether the required technologies are available to us or nor. If they are available, then we must ask if we have the capacity. For instance,

  Will our current Printer be able to handle the new reports and forms required of a new system?

Do we possess the necessary Technical Expertise and is the Schedule reasonable?

This consideration of technical feasibility is often forgotten during Feasibility analysis. We may have the technology, but that doesn't mean we have the skills required to properly apply that technology. As far as our project is concerned we have the necessary expertise so that the proposed solution can be made feasible.

3.2 Economical Feasibility

Economic feasibility determines whether there are sufficient benefits increasing to make the cost acceptable, or is the cost of the system too high. As this signifies cost benefit analysis and savings. On the behalf of the cost-benefit analysis, the proposed system is feasible and is economical regarding its pre-assumed cost for making a system.

During the economical feasibility test we maintained the balance between the Operational and Economical feasibilities, as the two were the conflicting. For example, the solution that provides the best operational impact for the end-users may also be the most expensive and, therefore, the least economically feasible.

We classified the costs of Online job portal according to the phase in which they occur. As we know that the system development costs are usually one-time costs that will not recur after the project has been completed.

For calculating the Development costs, we evaluated certain cost categories viz.

- Personnel costs
- Computer usage
- Training
- Supply and equipment's costs
- Cost of any new computer equipment's and software.

In order to test whether the Proposed System is cost-effective or not we evaluated it through three techniques viz.

- Payback analysis
- Return on Investment
- Net Present value
- Cost-based study

It is important to identify cost and benefit factors, which can be categorized as follows: 1. Development costs; and 2. Operating costs. This is an analysis of the costs to be incurred in the system and the benefits derivable out of the system.

- Time-based study

This is an analysis of the time required to achieve a return on investments. The future value of a project is also a factor.

3.3 Behavioral feasibility

People are inherently resistant to change and computers have been known to facilitate change. There is always some reluctance among the users against the introduction of new system but they were told that this system would eliminate the unnecessary overhead of database migration and conversion, which presently had to be carried out on daily basis to facilitate transactions between the different departments. The objective this feasibility phase is to take the operational staff into confidence. As the success of a good system depends upon the willingness of the operating staff, they were taken into full confidence that the new proposed system would make their jobs easier, relieve them from the unnecessary overheads and reduce the possibility of errors creeping into the system.

3.4 Project

The Blood Donation management System is to create an e-Information about the donor and organization that are related to donating the blood. Through this application any person who is interested in donating the blood can register himself in the same way if any organization wants to register itself with this site that can also register. Moreover if any general consumer wants to make request blood online he can also take the help of this site. Blood Bank is the main authority who can do addition, deletion, and modification if required.

Blood Donation management System project is aimed to developing an online Blood Donation Information. The entire Online Blood Donation management System project has been developed keeping in view of the distributed client server computing technology, in mind. Through this Online Blood Donation management System application any person who is interested in donating the blood can register himself in the same way if any organization wants to register itself with this site that can also register. Moreover if any general consumer wants to make request blood online he can also take the help of this site. Blood Bank is the main authority who can do addition, deletion, and modification if required.

Blood Donation management System project is designed such that it follows the view of distributed architecture having centralized storage of the database part. By using the constructs of MS-SQL Server and all the user interfaces have been designed using the JSP servlet and spring mvc technologies. The database connectivity is planned using the "SQL Connection" methodology. The standards of security and data protective mechanism have been given a big choice for proper usage. The application takes care of different modules and their associated reports, which are produced as per the applicable strategies and standards that are put forwarded by the administrative staff.

4. *Requirement Analysis*

Systems analysis is the study of sets of interacting entities, including computer systems analysis. This field is closely related to operations research. It is also "an explicit formal inquiry carried out to help someone (referred to as the decision maker) identify a better course of action and make a better decision than he might otherwise have made. "Analysis is defined as the procedure by which we break down an intellectual or substantial whole into parts so that we can achieve our end goals. The development of a computer-based information system includes a systems analysis phase which produces or enhances the **data model** which itself is a precursor to creating or enhancing a **database**. There are a number of different approaches to system analysis. When a computer-based information system is developed, systems analysis would constitute the following

Steps:

1. The development of a feasibility study, involving determining whether a project is economically, socially, technologically and organizationally feasible.
2. Conducting fact-finding measures, designed to ascertain the requirements of the system's end-users. These typically span interviews, questionnaires, or visual observations of work on the existing system.
3. Gauging how the end-users would operate the system (in terms of general experience in using computer hardware or software), what the system would be used for etc.

Another view outlines a phased approach to the process. This approach breaks systems analysis into 5 phases:

- Scope definition
- Problem analysis
- Requirements analysis
- Logical design
- Decision analysis

Use case are a widely-used systems analysis modelling tool for identifying and expressing the functional requirements of a system. Each use case is a business scenario or event for which the system must provide a defined response. Use cases evolved out of object-oriented analysis.

4.1 Requirement specification

Information gathering is usually the first phase of the software development project. The purpose of this phase is to identify and document the exact requirements for the system. The user‘s request identifies the need for a new information system and on investigation re-defined the new problem to be based on MIS, which supports management. The objective is to determine whether the request is valid and feasible before recommendation is made to build a new or existing manual system continues.

The major steps are:

- Defining the user requirements.
- Studying the present system to verify the problem.
- Defining the performance expected by the candidate to use requirements.

4.2 S/W and H/W Requirement Specification

4.2.1 Hardware Requirements:

- Pentium IV 1.8 GHz and Above
- 128 MB DDRAM or More
- 40 GB HDD
- Printer
- Power Backup
- Internet Connection

4.2.2 Software Requirements:

1. JDK 1.7
   - Eclipse Neon JEE
2. Database
   - SQL Developer 10g
3. Web Server
   - Tomcat 7.0.144
4. Operating System
   - Windows 7 / Vista / XP sp3 /10

4.3 Technologies Used

1. Presentation Layer

❖ Web Interface
- HTML (Hypertext Markup Language)
- CSS (Cascading Style Sheet)
- JavaScript

2. Database Layer

 SQL

3. Business Layer

❖ Core Java Technologies
- Exception Handling
- Multithreading
- Collections Framework
- JDBC (Java Database Connectivity)
- Spring
- Restful-API
- Maven

❖ Web Components
- Servlets
- JSP (Java Server Pages)

❖ Enterprise Components
- J2EE (Java 2 Enterprise Edition)

4. Server Layer

 Tomcat

5. ***Modular description of the Job***

- To develop a powerful online programming environment for java programming.
- To manage all details of all users of the Web IDE.
- To manage programs (i.e. Java files) on server made by users using IDE.
- To provide support to users, so that users could share their problems with other users.
- To allow users to share their Java Programs with other users
- To understand the working of JDBC
- To understand how to make Front-End with the help of HTML, CSS, JS, Angular JS.
- To understand the need of SQL in daily life and in the project.
- To know about the business how it works.
- To understand the working of share market.

6. *Detailed Analysis of individual module*

HTML

❖ HTML is a language for describing web pages.

- HTML stands for **H**yper **T**ext **M**arkup **L**anguage
- HTML is not a programming language, it is a **markup language**
- A markup language is a set of **markup tags**
- HTML uses **markup tags** to describe web pages

❖ HTML markup tags are usually called HTML tags

- HTML tags are keywords surrounded by **angle brackets** like <html>
- HTML tags normally **come in pairs** like and
- The first tag in a pair is the **start tag,** the second tag is the **end tag** Start and end tags are also called **opening tags** and **closing tags**.

❖ HTML Documents

 HTML documents **describe web pages**

 HTML documents **contain HTML tags** and plain text
HTML documents are also **called web pages**

CSS

A few words about CSS

- **CSS** stands for **C**ascading Styl**e**sheets
- Styles define **how to display** HTML elements
- Styles are normally stored in **Style Sheets**
- Styles were added to HTML 4.0 **to solve a problem**
- **External Style Sheets** can save you a lot of work
- External Style Sheets are stored in **CSS files**
- Multiple style definitions will **cascade** into one

CSS provides means to customize inbuilt HTML tags HTML tags were originally designed to define the content of a document. They were supposed to say "This is a header", "This is a paragraph", "This is a table", by using tags like <h1>, <p>, <table>, and so on. The layout of the document was supposed to be taken care of by the browser, without using any formatting tags. As the two major browsers - Netscape and Internet Explorer - continued to add new HTML tags and attributes (like the <font> tag and the color attribute) to the original HTML

Specification, it became more and more difficult to create Web sites where the content of HTML documents was clearly separated from the document's presentation layout. To solve this problem, the World Wide Web Consortium (W3C) - the non-profit, standard setting consortium, responsible for standardizing HTML - created STYLES in addition to HTML 4.0. All major browsers support Cascading Style Sheets. Styles sheets define HOW HTML elements are to be displayed, just like the font tag and the color attribute in HTML 3.2. Styles are normally saved in external .css files. External style sheets enable you to change the appearance and layout of all the pages in your Web, just by editing one single CSS document.

JavaScript

JavaScript is used in millions of Web pages to improve the design, validate forms, detect browsers, create cookies, and much more. JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Firefox, and Opera.

A few words about JavaScript

- JavaScript was designed to add interactivity to HTML pages.
- JavaScript is a scripting language.
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

Purpose of using JavaScript

 JavaScript gives HTML designers a programming tool –
HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages

 JavaScript can put dynamic text into an HTML page –
A JavaScript statement like this: document. Write("<h1>" + name + "</h1>") can write a variable text into an HTML page

 JavaScript can react to events –
A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element

- **JavaScript can read and write HTML elements –**
  A JavaScript can read and change the content of an HTML element

- **JavaScript can be used to validate data –**
  A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing

- **JavaScript can be used to detect the visitor's browser –**
  A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser

- **JavaScript can be used to create cookies –**
  A JavaScript can be used to store and retrieve information on the visitor's computer

Where to Put the JavaScript

- **Scripts in the head section:**
  Scripts to be executed when they are called, or when an event is triggered, go in the head section. When you place a script in the head section, you will ensure that the script is loaded before anyone uses it.

- **Scripts in the body section:**
  Scripts to be executed when the page loads go in the body section. When you place a script in the body section it generates the content of the page.

- **Using an External JavaScript:**
  When you might want to run the same JavaScript on several pages, without having to write the same script on every page, then you can write a JavaScript in an external file. Save the external JavaScript file with a .js file extension. The external script cannot contain the <script> tag. To use the external script, point to the J1. Js file in the "src" attribute of the <script> tag:<script type="text/JavaScript" src="J1.js"></script>

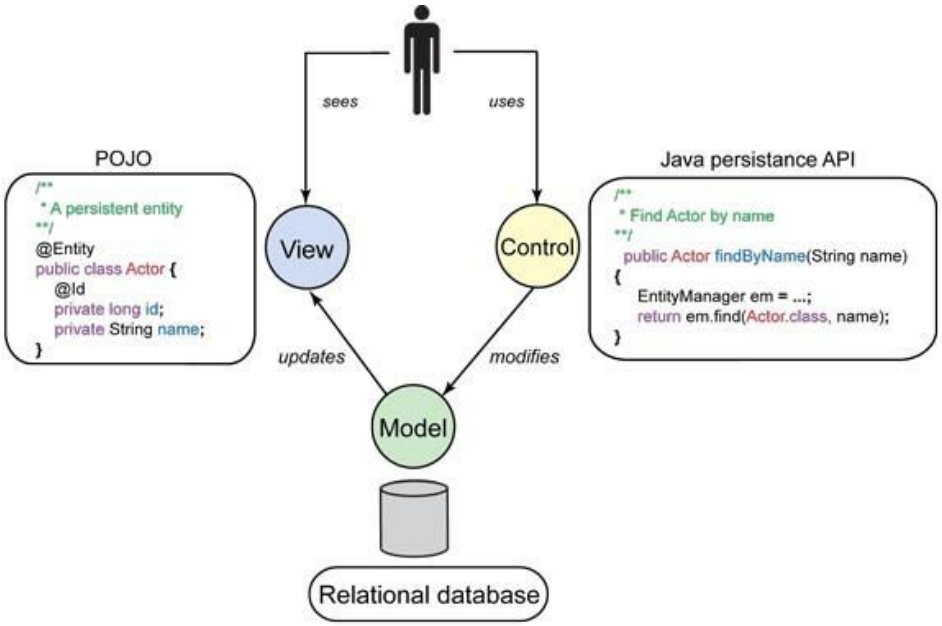
Angular JS

AngularJS (commonly referred to as "Angular.js" or "AngularJS 1.X") is a JavaScript based open-source front-end web application framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered

In developing single-page applications. The JavaScript components complement Apache Cordova, the framework used for developing cross-platform mobile apps. It aims to simplify both the development and the testing of such applications by providing a framework for client-side model–view–controller (MVC) and model–view–view model (MVVM) architectures, along with components commonly used in rich Internet applications. In 2014, the original AngularJS team began working on Angular (Application Platform).

The AngularJS framework works by first reading the HTML page, which has embedded into it additional custom tag attributes. Angular interprets those attributes as directives to bind input or output parts of the page to a model that is represented by standard JavaScript variables. The values of those JavaScript variables can be manually set within the code, or retrieved from static or dynamic JSON resources.

According to JavaScript analytics service Libscore, AngularJS is used on the websites of Wolfram Alpha, NBC, Walgreens, Intel, Sprint, ABC News, and approximately 12,000 other sites out of 1 million tested in October 2016. AngularJS is the 6th most starred project of all time on GitHub.



AngularJS is the frontend part of the MEAN stack, consisting of MongoDB database, Express.js web application server framework, Angular.js itself, and Node.js server runtime environment.

AngularJS is built on the belief that declarative programming should be used to create user interfaces and connect software components, while imperative programming is better suited to defining an application's business logic. The framework adapts and extends traditional HTML to present dynamic content through two-way data-binding that allows for the automatic synchronization of models and views. As a result, AngularJS de-emphasizes explicit DOM manipulation with the goal of improving testability and performance.

AngularJS's design goals include:

- To decouple DOM manipulation from application logic. The difficulty of this is dramatically affected by the way the code is structured.
- To decouple the client side of an application from the server side. This allows development work to progress in parallel, and allows for reuse of both sides.
- To provide structure for the journey of building an application: from designing the UI, through writing the business logic, to testing.

Angular implements the MVC pattern to separate presentation, data, and logic components. Using dependency injection, Angular brings traditionally server-side services, such as view-dependent controllers, to client-side web applications. Consequently, much of the burden on the server can be reduced.

AngularJS uses the term "scope" in a manner akin to the fundamentals of computer science.

Scope in computer science describes when in the program a particular binding is valid. The ECMA-262 specification defines scope as: a lexical environment in which a Function object is executed in client-side web scripts; akin to how scope is defined in lambda calculus.

As a part of the "MVC" architecture, the scope forms the "Model", and all variables defined in the scope can be accessed by the "View" as well as the "Controller". The scope behaves as a glue and binds the "View" and the "Controller".

In AngularJS, "scope" is a certain kind of object that itself can be in scope or out of scope in any given part of the program, following the usual rules of variable scope in JavaScript like any other object. When the term "scope" is used below, it refers to the Angular scope object and not the scope of a name binding.

SQL

SQL (**Structured Query Language**) is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS).

Originally based upon relational algebra and tuple relational calculus, SQL consists of a data definition language, data manipulation language, and data control language. The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control. Although SQL is often described as, and to a great extent is, a declarative language (4GL), it also includes procedural elements.

SQL was one of the first commercial languages for Edgar F. Codd's relational model, as described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks." Despite not entirely adhering to the relational model as described by Codd, it became the most widely used database language.

SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987. Since then, the standard has been revised to include a larger set of features. Despite the existence of such standards, most SQL code is not completely portable among different database systems without adjustments.

Design

SQL deviates in several ways from its theoretical foundation, the relational model and its tuple calculus. In that model, a table is a set of tuples, while in SQL, tables and query results are lists of rows: the same row may occur multiple times, and the order of rows can be employed in queries (e.g. in the LIMIT clause).

Critics argue that SQL should be replaced with a language that strictly returns to the original foundation

Language elements

The SQL language is subdivided into several language elements, including:

- *Clauses*, which are constituent components of statements and queries. (In some cases, these are optional.)
- *Expressions*, which can produce either scalar values, or tables consisting of columns and rows of data

- *Predicates*, which specify conditions that can be evaluated to SQL three-valued logic (3VL) (true/false/unknown) or Boolean truth values and are used to limit the effects of statements and queries, or to change program flow.
- *Queries*, which retrieve the data based on specific criteria. This is an important element of *SQL.*
- *Statements*, which may have a persistent effect on schemata and data, or may control transactions, program flow, connections, sessions, or diagnostics.
- SQL statements also include the semicolon (";") statement terminator. Though not required on every platform, it is defined as a standard part of the SQL grammar.
- *Insignificant whitespace* is generally ignored in SQL statements and queries, making it easier to format SQL code for readability.

Queries

The most common operation in SQL, the query, makes use of the declarative SELECT statement. SELECT retrieves data from one or more tables, or expressions. Standard SELECT statements have no persistent effects on the database. Some non-standard implementations of SELECT can have persistent effects, such as the SELECT INTO syntax provided in some databases.[19]

Queries allow the user to describe desired data, leaving the database management system (DBMS) to carry out planning, optimizing, and performing the physical operations necessary to produce that result as it chooses.

A query includes a list of columns to include in the final result, normally immediately following the SELECT keyword. An asterisk ("*") can be used to specify that the query should return all columns of the queried tables. SELECT is the most complex statement in SQL, with optional keywords and clauses that include:

FROM clause, which indicates the table(s) to retrieve data from. The FROM

- The JOIN clause can include optional subclauses to specify the rules for joining tables.
- The WHERE clause includes a comparison predicate, which restricts the rows returned by the query. The WHERE clause eliminates all rows from the result set where the comparison predicate does not evaluate to True.

- The GROUP BY clause projects rows having common values into a smaller set of rows. GROUP BY is often used in conjunction with SQL aggregation functions or to eliminate duplicate rows from a result set. The WHERE clause is applied before the GROUP BY clause.
- The HAVING clause includes a predicate used to filter rows resulting from the GROUP BY clause. Because it acts on the results of the GROUP BY clause, aggregation functions can be used in the HAVING clause predicate.
- The ORDER BY clause identifies which column[s] to use to sort the resulting data, and in which direction to sort them (ascending or descending). Without an ORDER BY clause, the order of rows returned by an SQL query is undefined. The DISTINCT keyword eliminates duplicate data.

Operator

Operator	Description	Example
=	Equal to	Author = 'Alcott'
	Not equal to (many DBMSs accept != in addition to <>)	Dept <> 'Sales'
>	Greater than	Hire_Date > '2012-01-31'

<	Less than	Bonus < 50000.00
>=	Greater than or equal	Dependents >= 2
<=	Less than or equal	Rate <= 0.05
BETWEEN	Between an inclusive range	Cost **BETWEEN** 100.00 **AND** 500.00
LIKE	Match a character pattern	First_Name **LIKE** 'Will%'
IN	Equal to one of multiple possible values	DeptCode **IN** (101, 103, 209)
IS **IS NOT** *or*	Compare to null (missing data)	Address **IS NOT NULL**
IS NOT DISTINCT FROM	Is equal to value or both are	Debt **IS NOT DISTINCT FROM** - Receivables

	nulls (missing data)	
AS	Used to change a field name when results	**SELECT** employee **AS** 'department1'

Java Technology

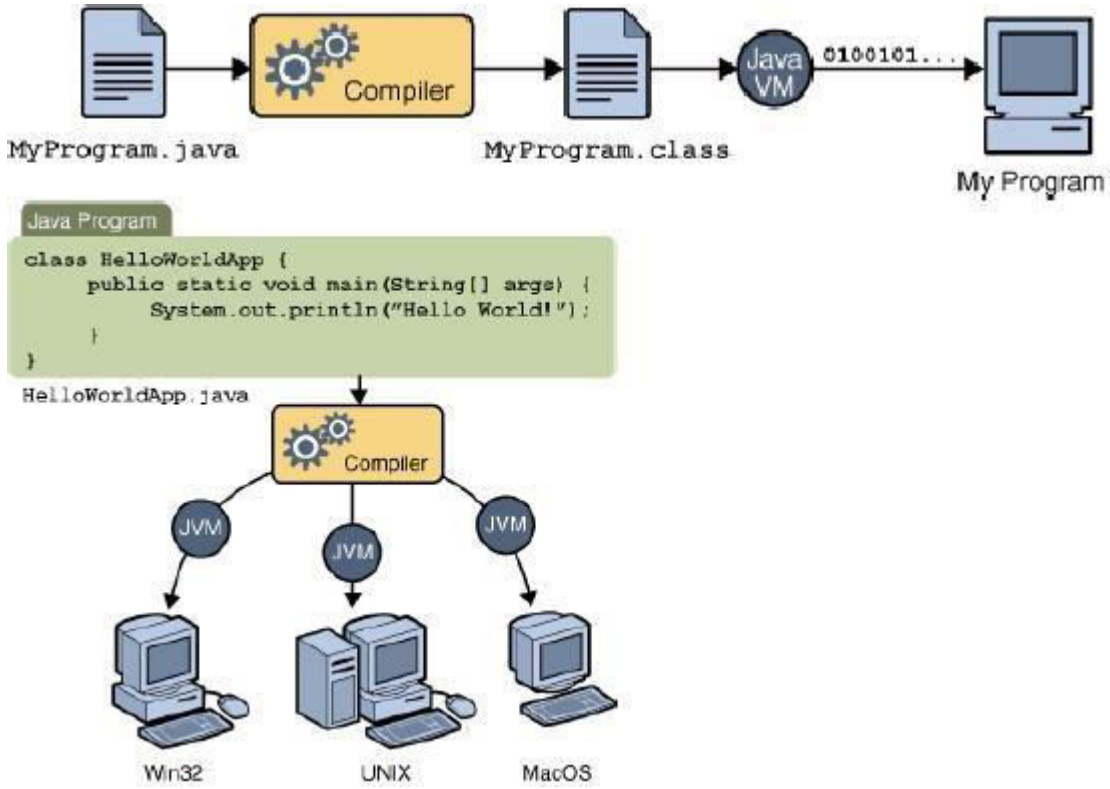
Java technology is both a programming language and a platform.

The Java Programming Language

The Java programming language is a high-level language that can be characterized by all of the following characteristics

- Simple
- Object oriented
- Distributed
- Multithreaded
- Architecture neutral
- Portable
- High performance
- Robust

In the Java programming language, all source code is first written in plain text files ending with the .java extension. Those source files are then compiled into .class files by the javac compiler. A .class file does not contain code that is native to your processor; it instead contains bytecodes the machine language of the Java Virtual Machine1 (Java VM). The java launcher
tool then runs your application with an instance of the Java Virtual Machine



An overview of the software development process.

An overview of the software development process. Because the Java VM is available on many different operating systems, the same. classfiles are capable of running on Microsoft Windows, the Solaris™ Operating System (Solaris OS), Linux, or Mac OS. Some virtual machines, such as the Java Hotspot virtual machine, perform additional steps at runtime to give your application a performance boost.

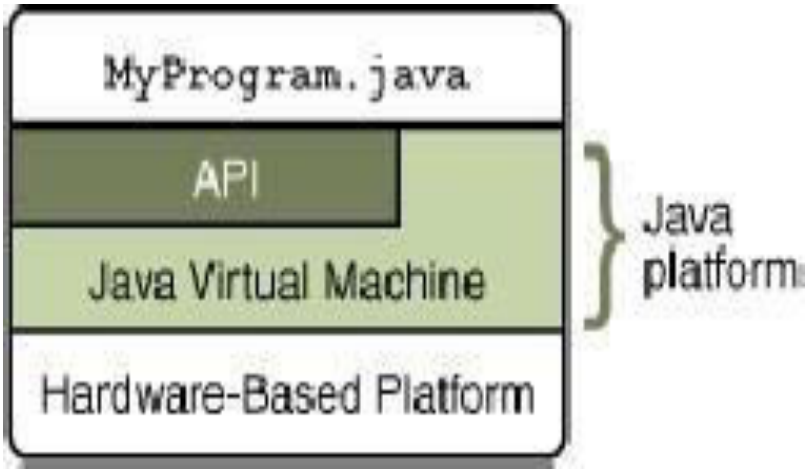
The Java Platform

A *platform* is the hardware or software environment in which a program runs. Most platforms can be described as a combination of the operating system and underlying hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- The Java Virtual Machine
- The Java Application Programming Interface (API)

Java Virtual Machine is the base for the Java platform and is ported onto various hardware-based platforms. The API is a large collection of ready-made software components that provide many useful capabilities. It is grouped into libraries of related classes and interfaces; these libraries are known as packages.



The API and Java Virtual Machine insulate the program from the underlying hardware.

Every full implementation of the Java platform gives you the following features:

- Development Tools:
  The development tools provide means for compiling, running, debugging, and documenting your applications. The main tools are the javac compiler, the java launcher, and the Javadoc documentation tool.

- Application Programming Interface (API):
  The API provides the core functionality of the Java programming language. It offers a wide array of useful classes ready for use in your own applications. It spans everything from basic objects, to networking and security, to XML generation and database access, etc.

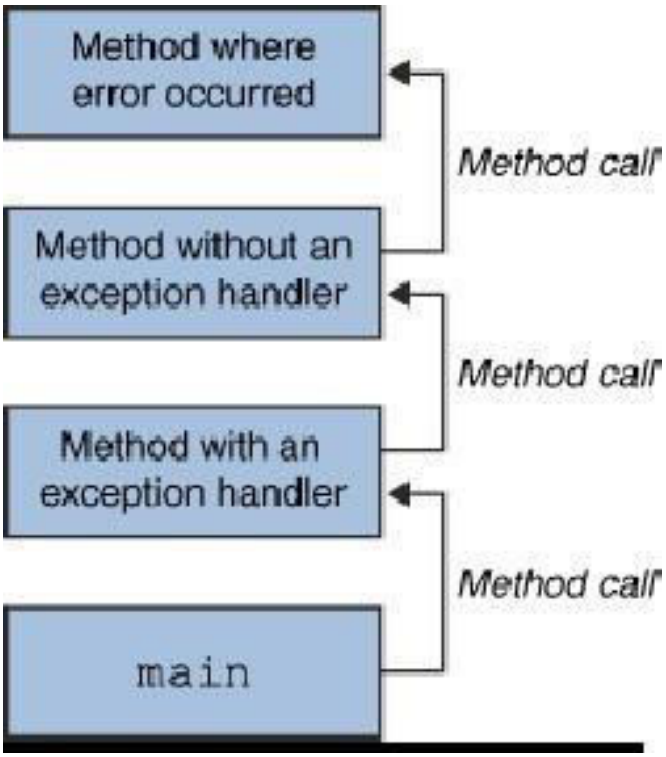
- Deployment Technologies:
  The JDK software provides standard mechanisms such as the Java Web Start software and Java Plug-In software for deploying your applications to end users.

- User Interface Toolkits:
  The Swing and Java 2D toolkits make it possible to create sophisticated Graphical User Interfaces (GUIs).

- Integration Libraries:
  Integration libraries such as the Java IDL, JDBC, JNDI, Java RMI, and Java Remote Method Invocation over Internet Inter-ORB Protocol Technology (Java RMI-IIOP Technology) enable database access and manipulation of remote objects.

Java - Exception Handling

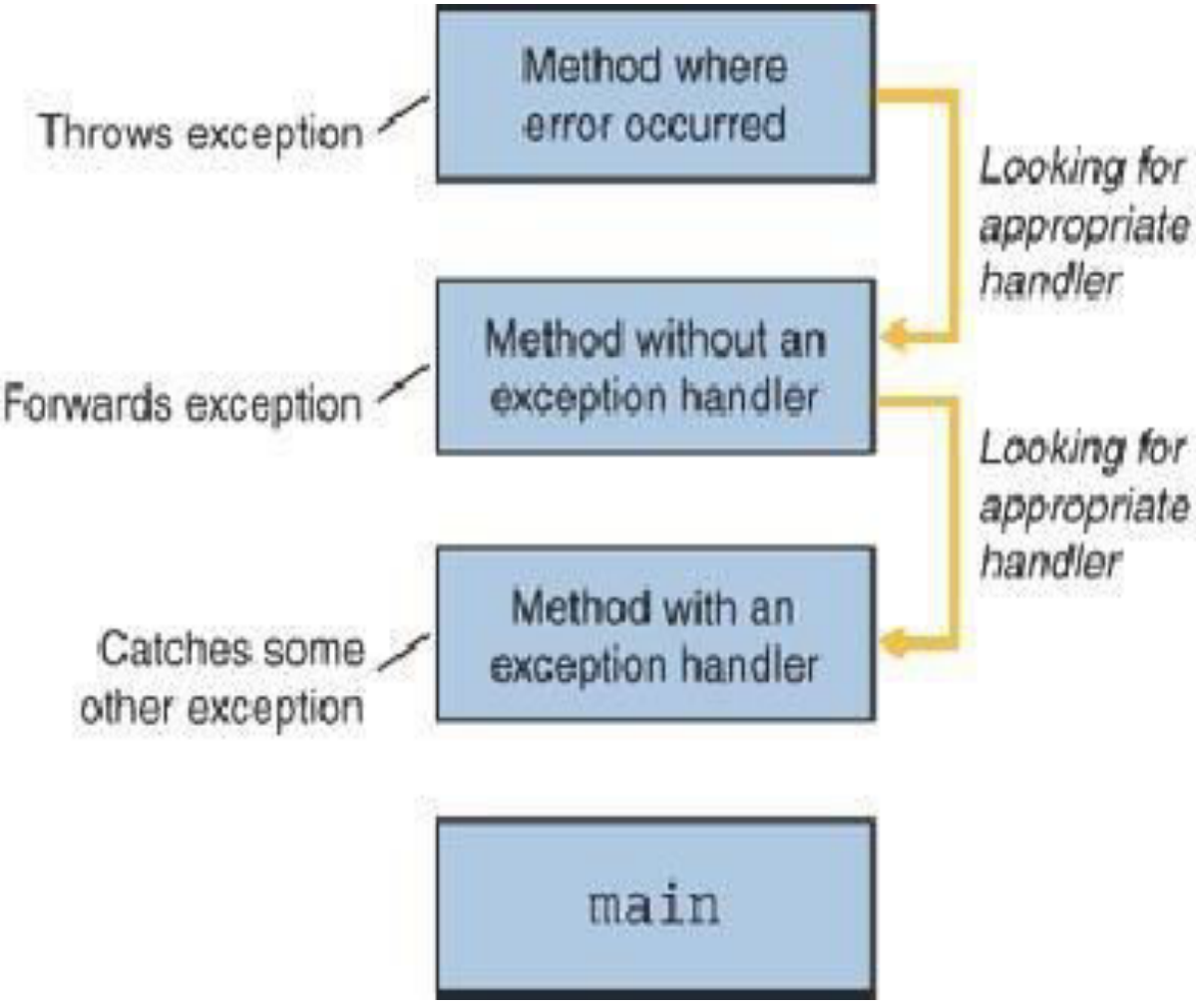
An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. When an error occurs within a method, the method creates an object and hands It off to the runtime system. The object, called an exception object, contains information about the error, including its type and the state of the program when the error occurred. Creating an exception object and handing it to the runtime system is called throwing an exception. After a method throws an exception, the runtime system attempts to find something to handle it. The set of possible "somethings" to handle the exception is the ordered list of methods that had been called to get to the method where the error occurred. The list of methods is known as the call stack.



The call stack

The runtime system searches the call stack for a method that contains a block of code that can handle the exception. This block of code is called an exception handler. These arch begins with the method in which the error occurred and proceeds through the call stack in the reverse order in which the methods were called. When an appropriate handler is found, the runtime system passes the exception to the handler. An exception handler is considered appropriate if the type of the exception object thrown matches the type that can be handled by the handler.

The exception handler chosen is said to catch the exception. If the runtime system exhaustively searches all the methods on the call stack without finding an appropriate exception handler, as shown in the next figure, the runtime system (and, consequently, the program) terminates.



Searching the call stack for the exception handler

A program can catch exceptions by using a combination of the try, catch, and finally blocks.

- The try block identifies a block of code in which an exception can occur.
- The catch block identifies a block of code, known as an exception handler, that can handle a particular type of exception.

The finally block identifies a block of code that is guaranteed to execute, and is the right place to close files, recover resources, and otherwise clean up after the code enclosed in the try block.

Java – Multithreading

Threads are called lightweight processes. Threads exist within a process — every process has at least one. Threads share the process's resources, including memory and open files. This makes for efficient, but potentially problematic, communication.

Multithreaded execution is an essential feature of the Java platform. Every application has at least one thread — or several, if you count "system" threads that do things like memory management and signal handling. But from the application programmer's point of view, you start with just one thread, called the main thread. This thread has the ability to create additional threads.

Each thread is associated with an instance of the class Thread. There are two basic strategies for using Thread objects to create a concurrent application.

- To directly control thread creation and management, simply instantiate Thread each time the application needs to initiate an asynchronous task.
- To abstract thread management from the rest of your application, pass the application's tasks to an executor.

Java-JDBC

JDBC stands for "Java Database Connectivity". It is an API (Application Programming Interface) which consists of a set of Java classes, interfaces and exceptions and a specification to which both JDBC driver vendors and JDBC developers adhere when developing applications. JDBC is a very popular data access standard. RDBMS (Relational Database Management Systems) or third-party vendors develop drivers which adhere to the JDBC specification. Other developers use these drivers to develop applications which access those databases. The JDBC API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database.

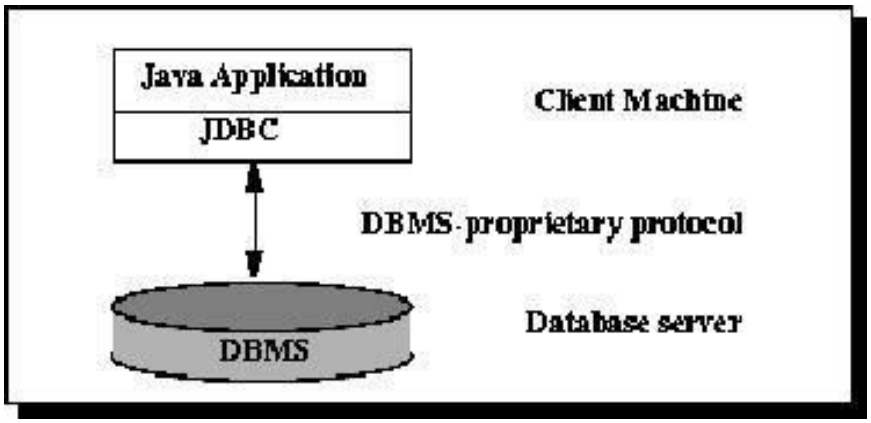
JDBC helps you to write java applications that manage these programming activities:

- Connect to a data source, like a database
- Send queries and update statements to the database

JDBC Architecture

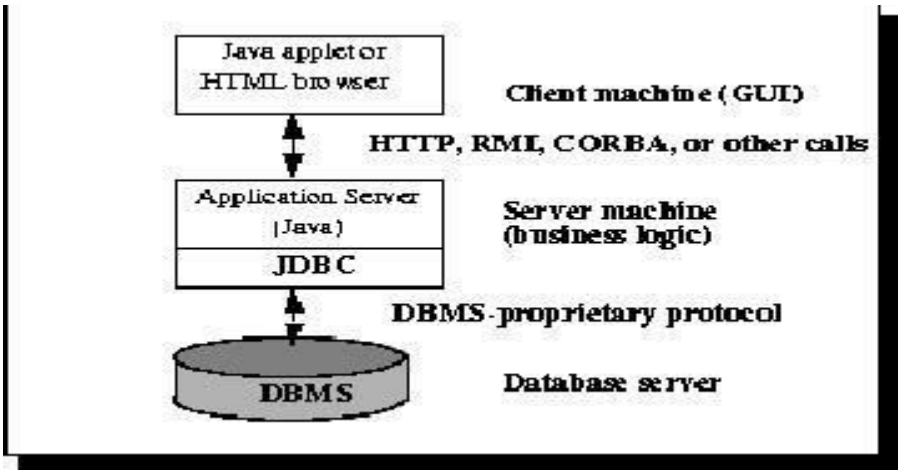
The JDBC API supports both two-tier and three-tier processing models for database access.

Two-tier Architecture for Data Access



In the two-tier model, a Java application talks directly to the data source. This requires a JDBC driver that can communicate with the particular data source being accessed. A user's commands are delivered to the database or other data source, and the results of those statements are sent back to the user. The data source may be located on another machine to which the user is connected via a network. This is referred to as a client/server configuration, with the user's machine as the client, and the machine housing the data source as the server. In the three-tier model, commands are sent to a "middle tier" of services, which then sends the commands to the data source. The data source processes the commands and sends the results back to the middle tier, which then sends them to the user. MIS directors find the three-tier model very attractive

Three-tier Architecture for Data Access



Spring Framework

The **Spring Framework** is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform. Although the framework does not impose any specific programming model, it has become popular in the Java community as an alternative to, replacement for, or even addition to the Enterprise JavaBeans (EJB) model. The Spring Framework is open source.

Inversion of control container (dependency injection)

Central to the Spring Framework is its inversion of control (IoC) container, which provides a consistent means of configuring and managing Java objects using reflection. The container is responsible for managing object lifecycles of specific objects: creating these objects, calling their initialization methods, and configuring these objects by wiring them together.

Objects created by the container are also called managed objects or beans. The container can be configured by loading XML files or detecting specific Java annotations on configuration classes. These data sources contain the bean definitions that provide the information required to create the beans.

Objects can be obtained by means of either dependency lookup or dependency injection. Dependency lookup is a pattern where a caller asks the container object for an object with a specific name or of a specific type. Dependency injection is a pattern where the container passes objects by name to other objects, via either constructors, properties, or factory methods.

In many cases one need not use the container when using other parts of the Spring Framework, although using it will likely make an application easier to configure and customize. The spring container provides a consistent mechanism to configure applications and integrates with almost all Java environments, from small-scale applications to large enterprise applications.

The container can be turned into a partially compliant EJB 3.0 container by means of the Pitchfork project. Some criticize the Spring Framework for not complying with standards.[14] However, Spring Source doesn't see EJB 3 compliance as a major goal, and claims that the Spring Framework and the container allow for more powerful programming models.[15] You do not create an object, but describe how they should be created, by defining it in the spring configuration file. You do not call services and components, but tell which services and components must be called, by defining them in the spring configuration files. This makes the code easy to maintain and easier to test through IoC.

Aspect-oriented programming framework

The Spring Framework has its own Aspect-oriented programming (AOP) framework that modularizes cross-cutting concerns in aspects. The motivation for creating a separate AOP framework comes from the belief that it would be possible to provide basic AOP features without too much complexity in either design, implementation, or configuration. The Spring AOP framework also takes full advantage of the spring container.

The Spring AOP framework is proxy pattern-based, and is configured at run time. This removes the need for a compilation step or load-time weaving. On the other hand, interception only allows for public method-execution on existing objects at a join point.

Compared to the AspectJ framework, Spring AOP is less powerful, but also less complicated. Spring 1.2 includes support to configure AspectJ aspects in the container. Spring 2.0 added more integration with AspectJ; for example, the pointcut language is reused and can be mixed with Spring AOP-based aspects. Further, Spring 2.0 added a Spring Aspects library that uses AspectJ to offer common Spring features such as declarative transaction management and dependency injection via AspectJ compile-time or load-time weaving. SpringSource also uses AspectJ AOP in other Spring projects such as Spring Roo and Spring Insight, with Spring Security also offering an AspectJ-based aspect library.

Spring AOP has been designed to make it able to work with cross-cutting concerns inside the Spring Framework. Any object which is created and configured by the container can be enriched using Spring AOP.

The Spring Framework uses Spring AOP internally for transaction management, security, remote access, and JMX.
Since version 2.0 of the framework, spring provides two approaches to the AOP configuration:

- schema-based approach and
- @AspectJ-based annotation style.

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="http://www.springframework.org/schema/beans
```

```
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc.xsd
http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop.xsd">
```

The Spring team decided not to introduce new AOP-related terminology; therefore, in the Spring reference documentation and API, terms such as aspect, join point, advice, pointcut, introduction, target object (advised object), AOP proxy, and weaving all have the same meanings as in most other AOP frameworks (particularly AspectJ).

Data access framework

Spring's data access framework addresses common difficulties developers face when working with databases in applications. Support is provided for all popular data access frameworks in Java: JDBC, iBatis/MyBatis, Hibernate, JDO, JPA, Oracle TopLink, Apache OJB, and Apache Cayenne, among others.

For all of these supported frameworks, Spring provides these features

- Resource management - automatically acquiring and releasing database resources
- Exception handling - translating data access related exception to a Spring data access hierarchy
- Transaction participation - transparent participation in ongoing transactions
- Resource unwrapping - retrieving database objects from connection pool wrappers
- Abstraction for BLOB and CLOB handling

All these features become available when using template classes provided by Spring for each supported framework. Critics have said these template classes are intrusive and offer no advantage over using (for example) the Hibernate API directly. In response, the Spring developers have made it possible to use the Hibernate and JPA APIs directly. This however requires transparent transaction management, as application code no longer assumes the responsibility to obtain and close database resources, and does not support exception translation.

Together with spring's transaction management, its data access framework offers a flexible abstraction for working with data access frameworks. The Spring Framework doesn't offer a common data access API; instead, the full power of the supported APIs is kept intact. The Spring Framework is the only framework available in Java that offers managed data access environments outside of an application server or container.

Transaction management framework

Spring's transaction management framework brings an abstraction mechanism to the Java platform. Its abstraction is capable of:

- working with local and global transactions (local transaction does not require an application server)
- working with nested transactions
- working with savepoints
- working in almost all environments of the Java platform

In comparison, JTA only supports nested transactions and global transactions, and requires an application server (and in some cases also deployment of applications in an application server).

The Spring Framework ships a Platform Transaction Manager for a number of transaction management strategies:

- Transactions managed on a JDBC Connection
- Transactions managed on Object-relational mapping Units of Work
- Transactions managed via the JTA Transaction Manager and User Transaction
- Transactions managed on other resources, like object databases

Next to this abstraction mechanism the framework also provides two ways of adding transaction management to applications:

- Programmatically, by using Spring's Transaction Template
- Confirmatively, by using metadata like XML or Java annotations (@Transactional, etc.)

Together with spring's data access framework — which integrates the transaction management framework — it is possible to set up a transactional system through configuration without having to rely on JTA or EJB. The transactional framework also integrates with messaging and caching engines.

Model–view–controller framework

The Spring Framework features its own MVC web application framework, which wasn't originally planned. The Spring developers decided to write their own Web framework as a reaction to what they perceived as the poor design of the (then) popular Jakarta Struts Web framework,[19] as well as deficiencies in other available frameworks. In particular, they felt there was insufficient separation between the presentation and request handling layers, and between the request handling layer and the model.[20]

Like Struts, Spring MVC is a request-based framework. The framework defines strategy interfaces for all of the responsibilities that must be handled by a modern request-based

framework. The goal of each interface is to be simple and clear so that it's easy for Spring MVC users to write their own implementations, if they so choose. MVC paves the way for cleaner front end code. All interfaces are tightly coupled to the Servlet API. This tight coupling to the Servlet API is seen by some as a failure on the part of the Spring developers to offer a high-level abstraction for Web-based applications. However, this coupling makes sure that the features of the Servlet API remain available to developers while offering a high abstraction framework to ease working with said API.

The Dispatcher Servlet class is the front controller[21] of the framework and is responsible for delegating control to the various interfaces during the execution phases of an HTTP request.

The most important interfaces defined by Spring MVC, and their responsibilities, are listed below:

- Controller: comes between Model and View to manage incoming requests and redirect to proper response. It acts as a gate that directs the incoming information. It switches between going into model or view.
- Handler Adapter: execution of objects that handle incoming requests
- Handler Interceptor: interception of incoming requests comparable, but not equal to Servlet filters (use is optional and not controlled by Dispatcher Servlet).
- Handler Mapping: selecting objects that handle incoming requests (handlers) based on any attribute or condition internal or external to those requests
- Locale Resolver: resolving and optionally saving of the locale of an individual user
- Multipart Resolver: facilitate working with file uploads by wrapping incoming requests
- View: responsible for returning a response to the client. Some requests may go straight to view without going to the model part; others may go through all three.
- View Resolver: selecting a View based on a logical name for the view (use is not strictly required)

Each strategy interface above has an important responsibility in the overall framework. The abstractions offered by these interfaces are powerful, so to allow for a set of variations in their implementations, Spring MVC ships with implementations of all these interfaces and together offers a feature set on top of the Servlet API. However, developers and vendors are free to write other implementations. Spring MVC uses the Java `java.util.Map` interface as a data-oriented abstraction for the Model where keys are expected to be string values.

The ease of testing the implementations of these interfaces seems one important advantage of the high level of abstraction offered by Spring MVC. Dispatcher Servlet is tightly coupled to the Spring inversion of control container for configuring the web layers of applications. However, web applications can use other parts of the Spring Framework—including the container—and choose not to use Spring MVC.

Restful API

Representational state transfer (**REST**) or **RESTful** Web services are one way of providing interoperability between computer systems on the Internet. REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations. Other forms of Web service exist, which expose their own arbitrary sets of operations such as WSDL and SOAP.

"Web resources" were first defined on the World Wide Web as documents or files identified by their URLs, but today they have a much more generic and abstract definition encompassing everything or entity that can be identified, named, addressed or handled, in any way whatsoever, on the Web. In a RESTful Web service, requests made to a resource's URI will elicit a response that may be in XML, HTML, JSON or some other defined format. The response may confirm that some alteration has been made to the stored resource, and it may provide hypertext links to other related resources or collections of resources. Using HTTP, as is most common, the kind of operations available include those predefined by the HTTP verbs GET, POST, PUT, DELETE and so on.

By making use of a stateless protocol and standard operations, REST systems aim for fast performance, reliability, and the ability to grow, by re-using components that can be managed and updated without affecting the system as a whole, even while it is running.

The term *representational state transfer* was introduced and defined in 2000 by Roy Fielding in his doctoral dissertation. Fielding used REST to design HTTP 1.1 and Uniform Resource Identifiers (URI). The term is intended to evoke an image of how a well-designed Web application behaves: it is a network of Web resources (a virtual state-machine) where the user progresses through the application by selecting /user/tom links, such as and operations such as GET or DELETE (state transitions), resulting in the next resource (representing the next state of the application) being transferred to the user for their use.

Architectural Properties

The architectural properties affected by the constraints of the REST architectural style are:[2][8]

- Performance - component interactions can be the dominant factor in user-perceived performance and network efficiency[9]
- Scalability to support large numbers of components and interactions among components. Roy Fielding, one of the principal authors of the HTTP specification, describes REST's effect on scalability as follows:
    - REST's client–server separation of concerns simplifies component implementation, reduces the complexity of connector semantics, improves the

effectiveness of performance tuning, and increases the scalability of pure server components.

- Layered system constraints allow intermediaries—proxies, gateways, and firewalls—to be introduced at various points in the communication without changing the interfaces between components, thus allowing them to assist in communication translation or improve performance via large-scale, shared caching.
- REST enables intermediate processing by constraining messages to be self-descriptive: interaction is stateless between requests, standard methods and media types are used to indicate semantics and exchange information, and responses explicitly indicate cache ability.

- Simplicity of a uniform Interface
- Modifiability of components to meet changing needs (even while the application is running)
- Visibility of communication between components by service agents
- Portability of components by moving program code with the data
- Reliability is the resistance to failure at the system level in the presence of failures within components, connectors, or data.

Apache Maven

Maven is a build automation tool used primarily for Java projects. The word *maven* means "accumulator of knowledge" in Yiddish.

Maven addresses two aspects of building software: first, it describes how software is built, and second, it describes its dependencies. Contrary to preceding tools like Apache Ant, it uses conventions for the build procedure, and only exceptions need to be written down. An XML file describes the software project being built, its dependencies on other external modules and components, the build order, directories, and required plug-ins. It comes with predefined targets for performing certain well-defined tasks such as compilation of code and its packaging.

Maven dynamically downloads Java libraries and Maven plug-ins from one or more repositories such as the Maven 2 Central Repository, and stores them in a local cache.[4] This local cache of downloaded artifacts can also be updated with artifacts created by local projects. Public repositories can also be updated.

Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. The Maven project is hosted by the Apache Software Foundation, where it was formerly part of the Jakarta Project.

Maven is built using a plugin-based architecture that allows it to make use of any application controllable through standard input. Theoretically, this would allow anyone to write plugins to interface with build tools (compilers, unit test tools, etc.) for any other language. In reality, support and use for languages other than Java has been minimal. Currently a plugin for the .NET framework exists and is maintained, and a C/C++ native plugin is maintained for Maven

2.

Alternative technologies like Gradle and sbt as build tools do not rely on XML, but keep the key concepts Maven introduced. With Apache Ivy, a dedicated dependency manager was developed as well that also supports Maven repositories.

Convention over Configuration, that is, Maven provides default values for the project's configuration. The directory structure of a normal idiomatic Maven project has the following directory entries:

Project Object Model

A Project Object Model (POM) provides all the configuration for a single project. General configuration covers the project's name, its owner and its dependencies on other projects. One can also configure individual phases of the build process, which are implemented as plugins. For example, one can configure the compiler-plugin to use Java version 1.5 for compilation, or specify packaging the project even if some unit tests fail.

Larger projects should be divided into several modules, or sub-projects, each with its own POM. One can then write a root POM through which one can compile all the modules with a single command. POMs can also inherit configuration from other POMs. All POMs inherit from the Super POM by default. The Super POM provides default configuration, such as default source directories, default plugins, and so on.

Plugins

Most of Maven's functionality is in plugins. A plugin provides a set of goals that can be executed using the following syntax:

```
mvn [plugin-name]:[goal-name]
```

For example, a Java project can be compiled with the compiler-plugin's compile-goal by running `mvn compiler:compile`.

There are Maven plugins for building, testing, source control management, running a web server, generating Eclipse project files, and much more.[10] Plugins are introduced and configured in a `pom.xml` <plugins>-section of a file. Some basic plugins are included in every project by default, and they have sensible default settings.

However, it would be cumbersome if the archetypical build sequence of building, testing and packaging a software project required running each respective goal manually:

```
mvn compiler:compile
mvn surefire:test  mvn
jar:jar
```

Maven's lifecycle concept handles this issue.

Plugins are the primary way to extend Maven. Developing a Maven plugin can be done by extending the org.apache.maven.plugin.AbstractMojo class. Example code and explanation for a Maven plugin to create a cloud-based virtual machine running an application server is given in the article *Automate development and management of cloud virtual machines*.

Build lifecycles

Build lifecycle is a list of named *phases* that can be used to give order to goal execution. One of Maven's standard lifecycles is the *default lifecycle*, which includes the following phases, in this order:

- Validate
- generate-sources
- process-sources
- generate-resources
- process-resources
- compile
- process-test-sources
- process-test-resources
- test-compile
- test
- package
- install
- deploy

Goals provided by plugins can be associated with different phases of the lifecycle. For example, by default, the goal "compiler:compile" is associated with the "compile" phase, while the goal "surefire:test" is associated with the "test" phase. Consider the following command:

```
mvn test
```

When the preceding command is executed, Maven runs all goals associated with each of the phases up to and including the "test" phase. In such a case, Maven runs the "resources:resources" goal associated with the "process-resources" phase, then "compiler:compile", and so on until it finally runs the "surefire:test" goal.

Maven also has standard phases for cleaning the project and for generating a project site. If cleaning were part of the default lifecycle, the project would be cleaned every time it was built. This is clearly undesirable, so cleaning has been given its own lifecycle.

Standard lifecycles enable users new to a project the ability to accurately build, test and install every Maven project by issuing the single command:

```
mvn install
```

By default, Maven packages the POM file in generated JAR and WAR files. Tools like diet4j can use this information to recursively resolve and run Maven modules at run-time without requiring an "uber"-jar that contains all project code.

Dependencies

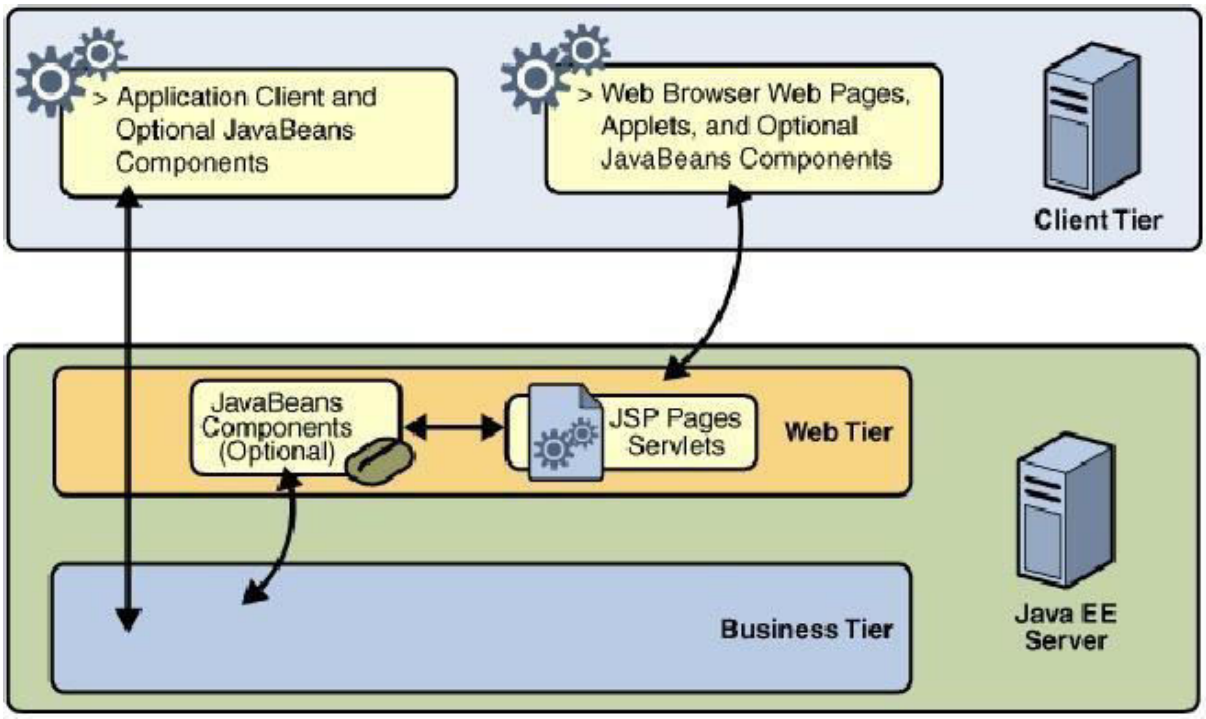
A central feature in Maven is dependency management. Maven's dependency-handling mechanism is organized around a coordinate system identifying individual artifacts such as software libraries or modules. The POM example above references the JUnit coordinates as a direct dependency of the project. A project that needs, say, the Hibernate library simply has to declare Hibernate's project coordinates in its POM. Maven will automatically download the dependency and the dependencies that Hibernate itself needs (called transitive dependencies) and store them in the user's local repository. Maven 2 Central Repository is used by default to search for libraries, but one can configure the repositories to be used (e.g., company-private repositories) within the POM.
There are search engines such as The Central Repository Search Engine[14] which can be used to find out coordinates for different open-source libraries and frameworks.

Projects developed on a single machine can depend on each other through the local repository. The local repository is a simple folder structure that acts both as a cache for downloaded dependencies and as a centralized storage place for locally built artifacts. The Maven command builds a project and places its binaries in the local repository. Then other projects can utilize this project by specifying its coordinates in their POMs.

Web Components

- Java web components are either servlets or pages created using JSP technology (JSP pages).
- Servlets are Java programming language classes that dynamically process requests and construct responses.
- JSP pages are text-based documents that execute as servlets but allow a more natural approach to creating static content.

The web tier might include a JavaBeans component to manage the user input and send that input to enterprise beans running in the business tier for processing, as shown:

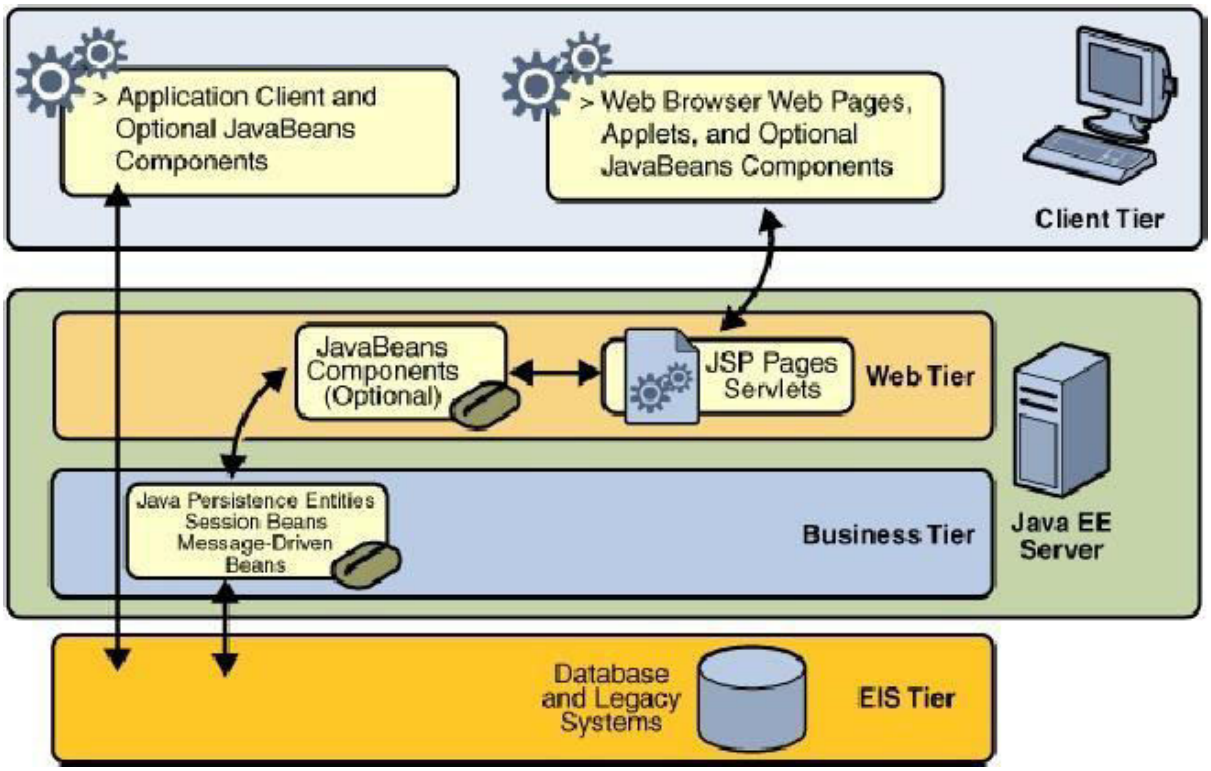


Business Components

Business code, which is logic that solves or meets the needs of a particular business domain is handled by enterprise beans running in the business tier.

Following figure shows how an enterprise bean receives data from client programs, processes it (if necessary), and sends it to the enterprise information system tier for storage.

An enterprise bean also retrieves data from storage, processes it (if necessary), and sends it back to the client program:



Enterprise JavaBeans Technology

An Enterprise JavaBeans (EJB) component, or enterprise bean, is a body of code having fields and methods to implement modules of business logic. You can think of an enterprise bean as a building block that can be used alone or with other enterprise beans to execute business logic on the Java EE server. There are two kinds of enterprise beans: session beans and message driven beans. A session bean represents a transient conversation with a client. When the client finishes executing, the session bean and its data are gone. A message-driven bean combines features of a session bean and a message listener, allowing a business component to receive messages asynchronously. Commonly, these are Java Message Service (JMS) messages.

Java Servlet Technology

Java servlet technology lets you define HTTP-specific servlet classes. A servlet class extends the capabilities of servers that host applications that are accessed by way of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers.

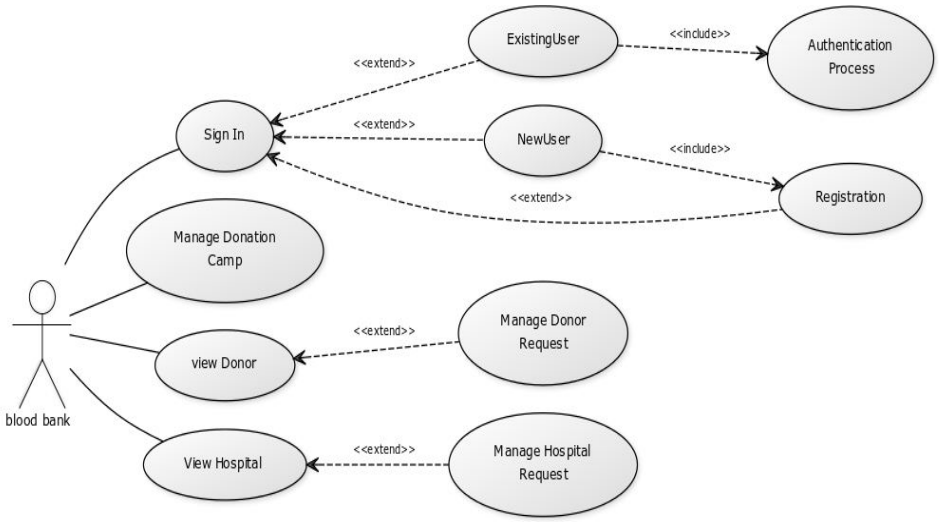
Java Server Pages Technology

Java Server Pages (JSP) technology lets you put snippets of servlet code directly into a text based document. A JSP page is a text-based document that contains two types of text: static data (which can be expressed in any text-based format such as HTML, WML, and XML) and JSP elements, which determine how the page constructs dynamic content.

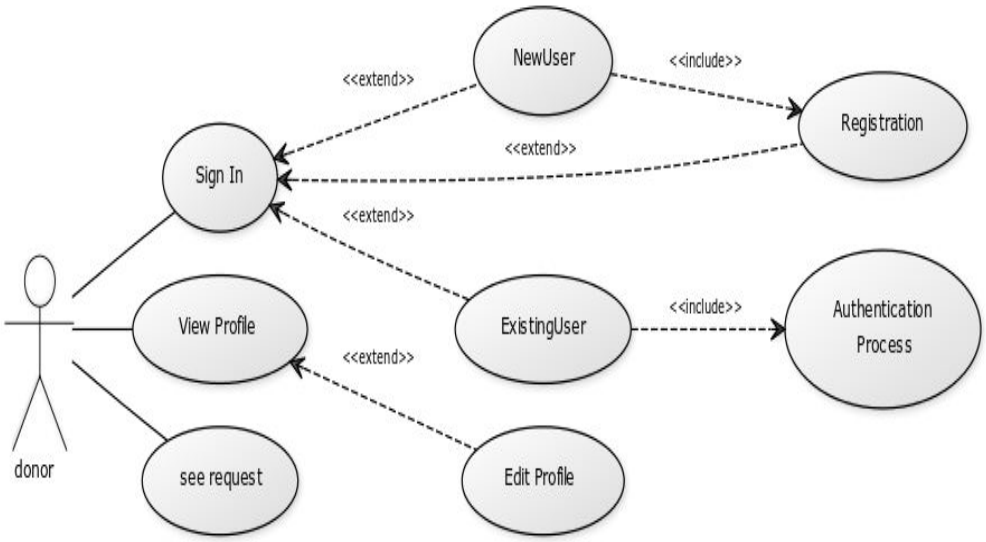
7. *Design*

UML Diagrams

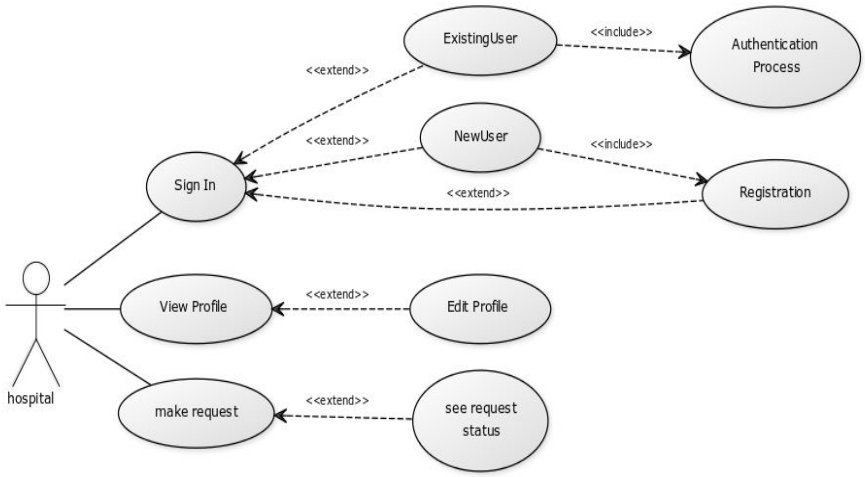
Blood Bank Use Case Diagram



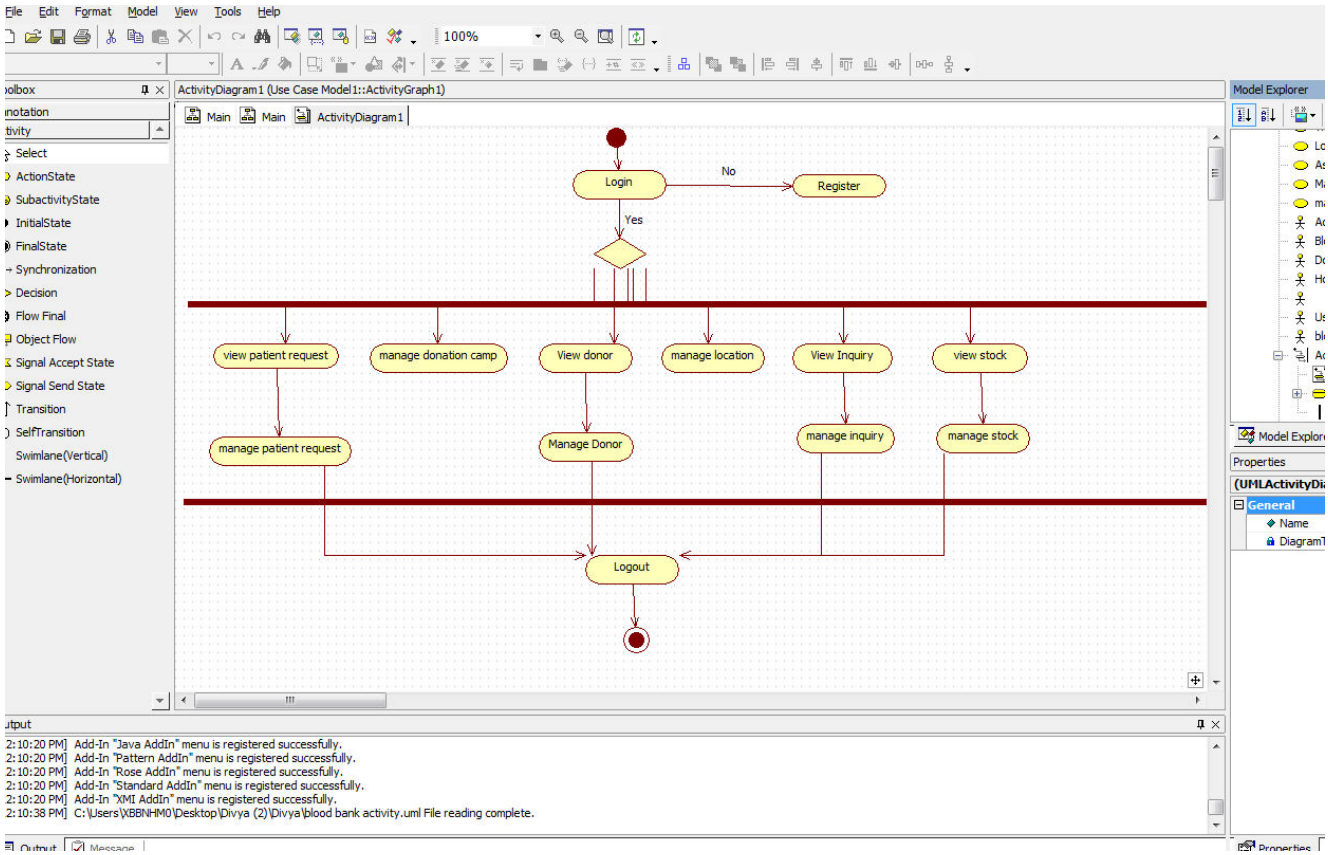
Donor Use Case Diagram



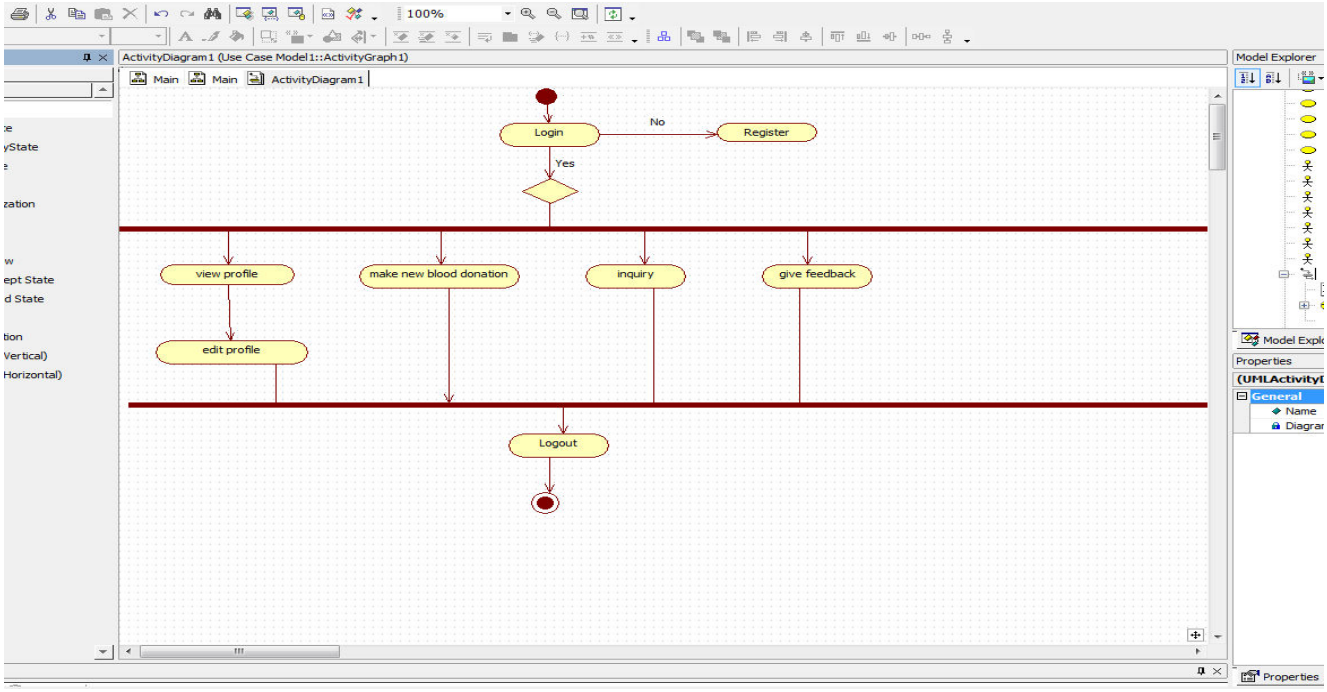
Hospital Use case Diagram



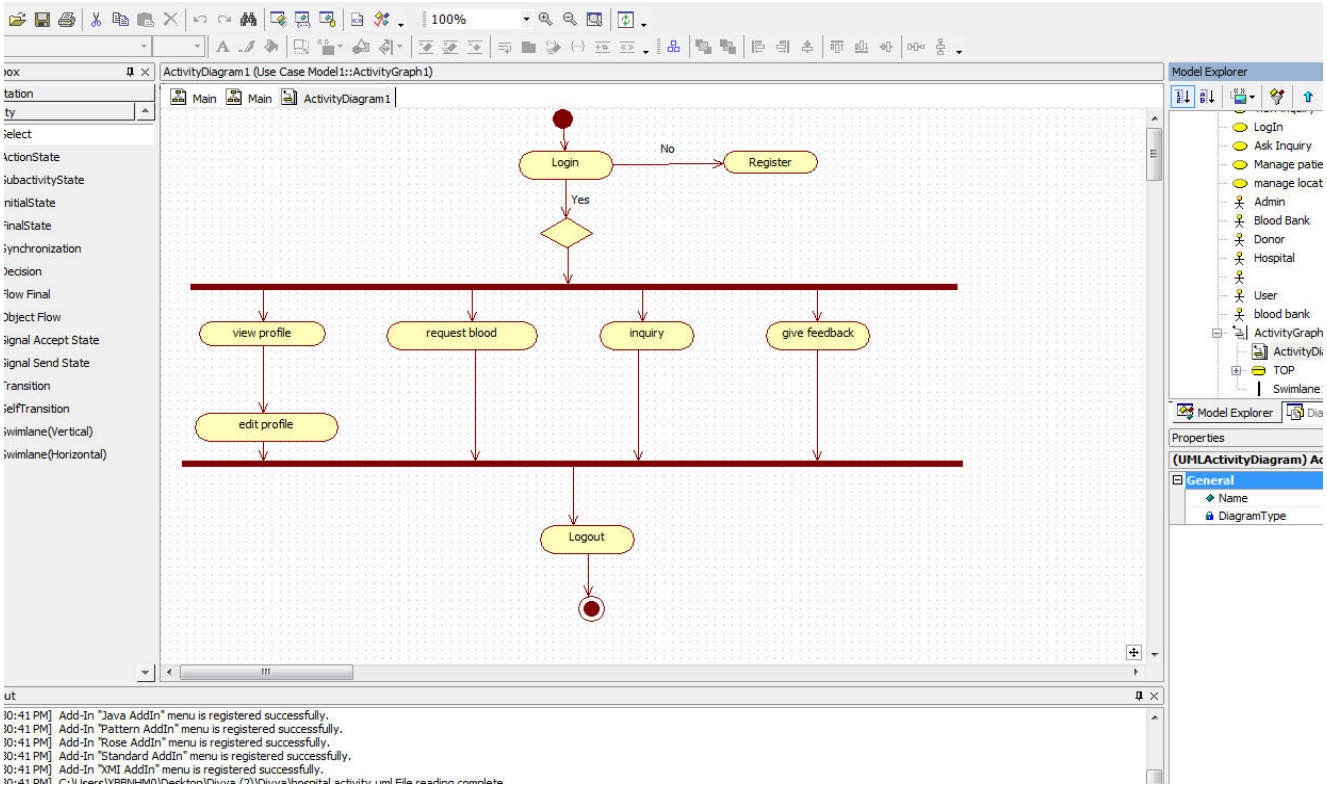
Blood Bank Activity Diagram



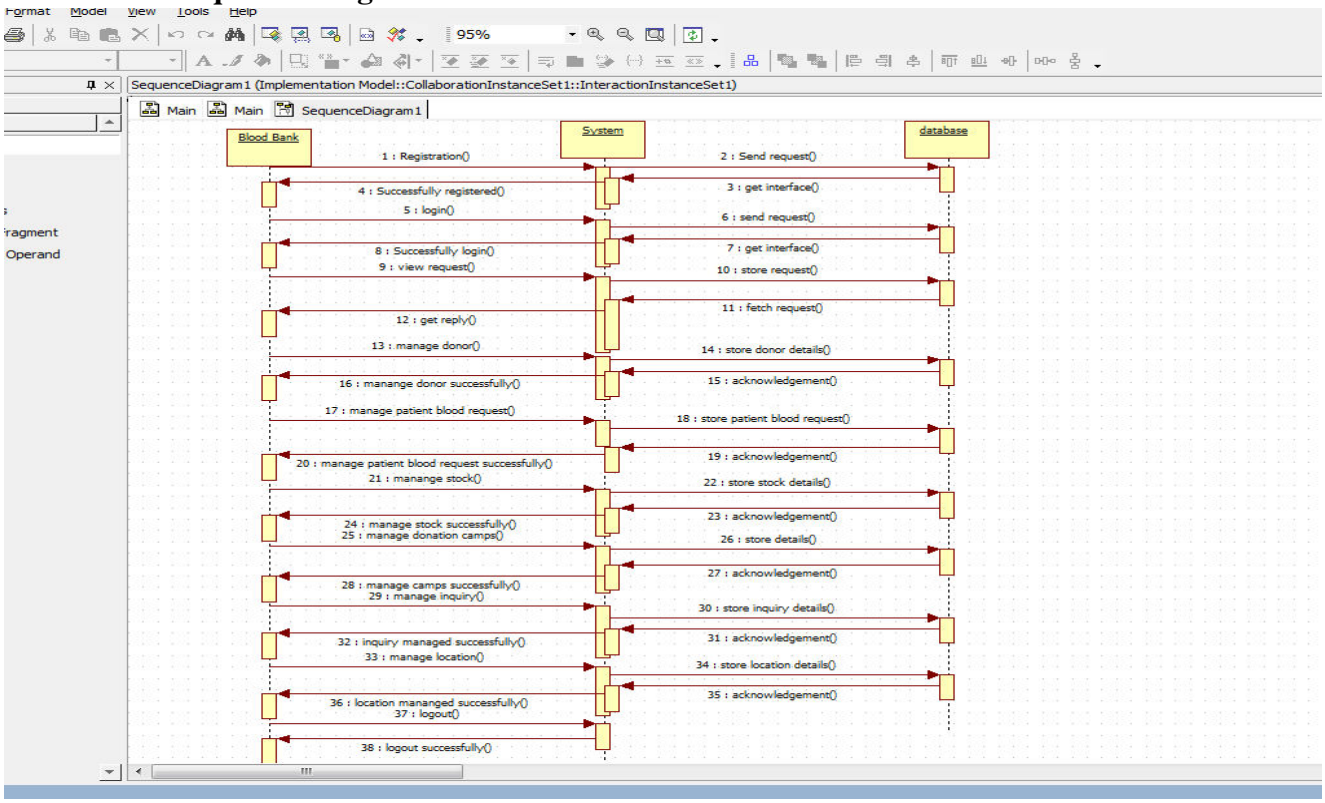
Donor Activity Diagram



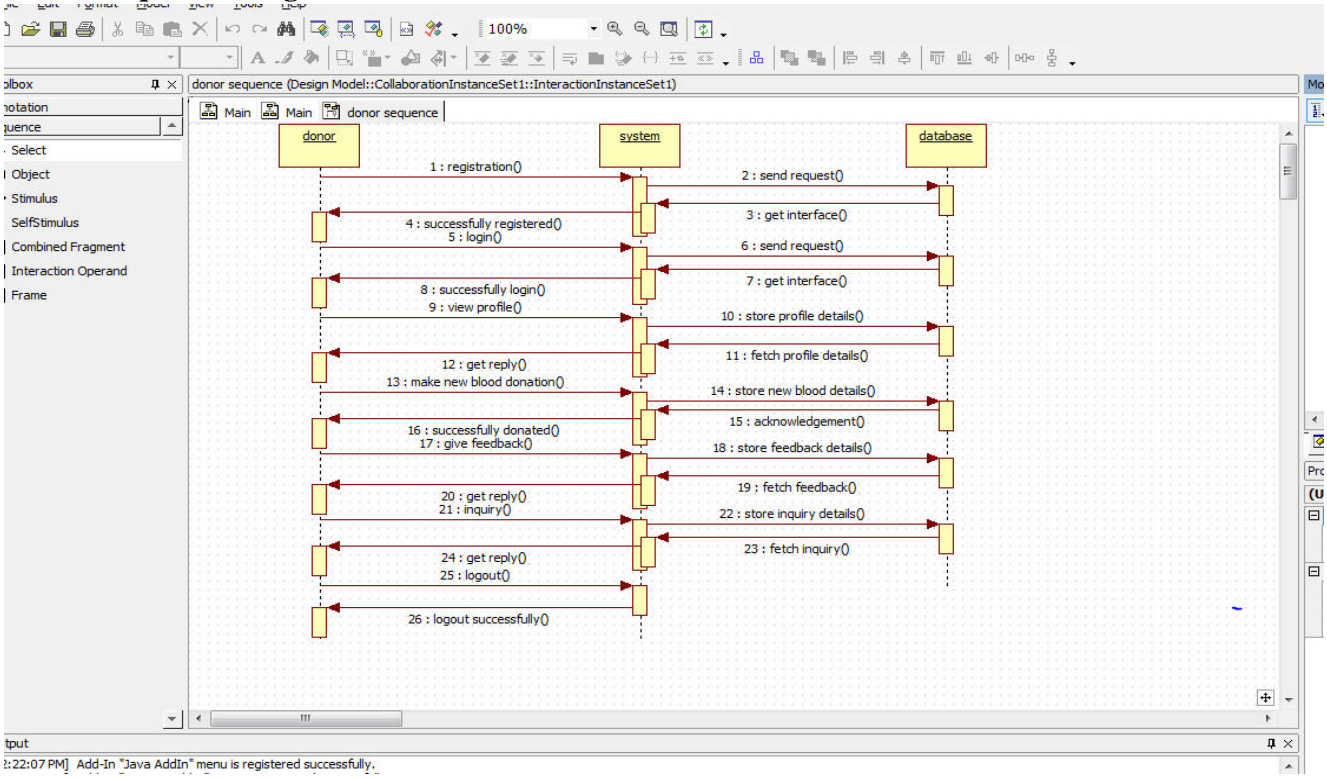
Hospital Activity Diagram



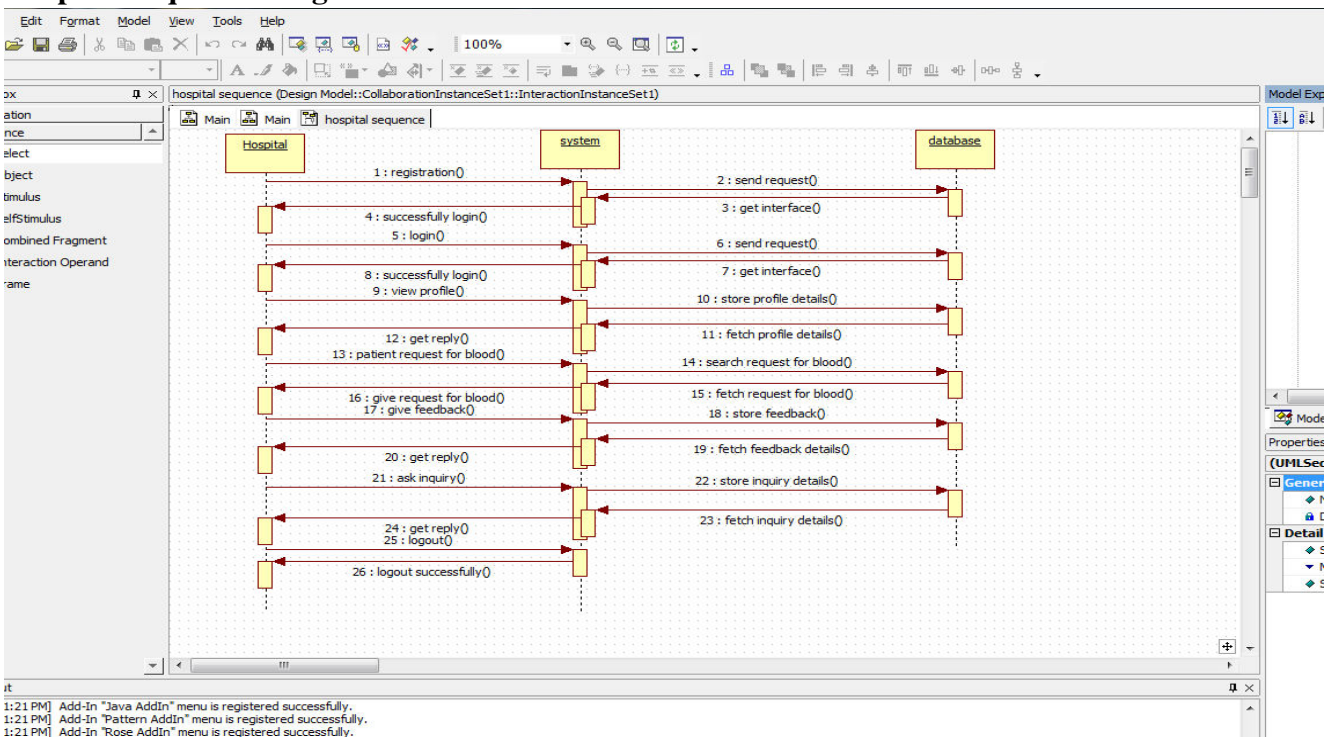
Blood Bank Sequence Diagram



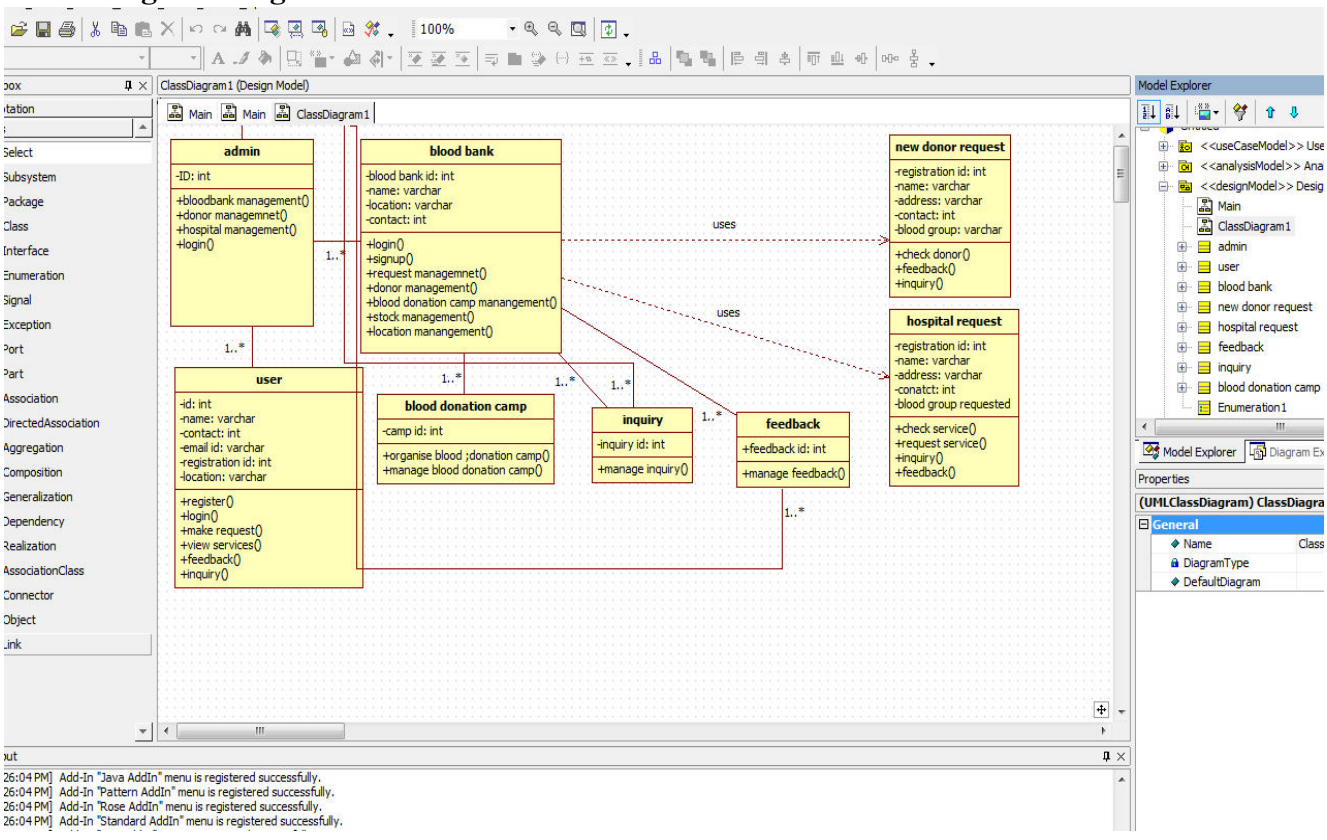
Donor Sequence Diagram



Hospital Sequence Diagram

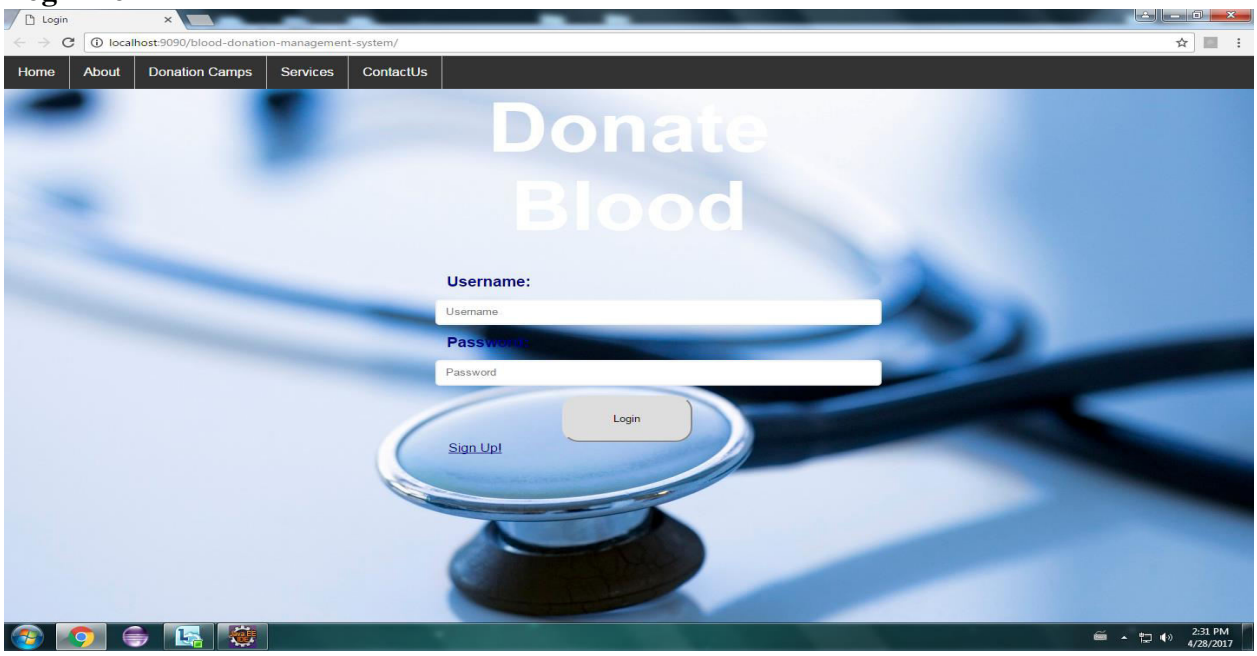


Class Diagram Diagram

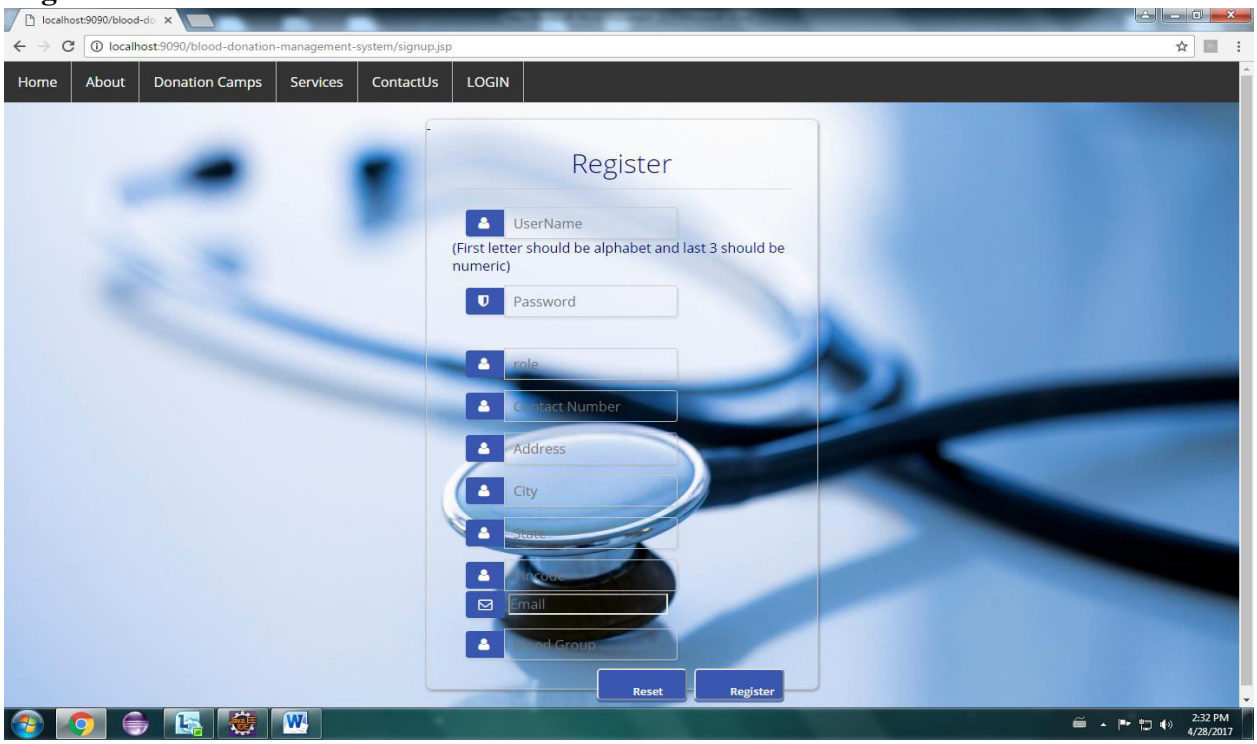


8. Screenshots

Login Form

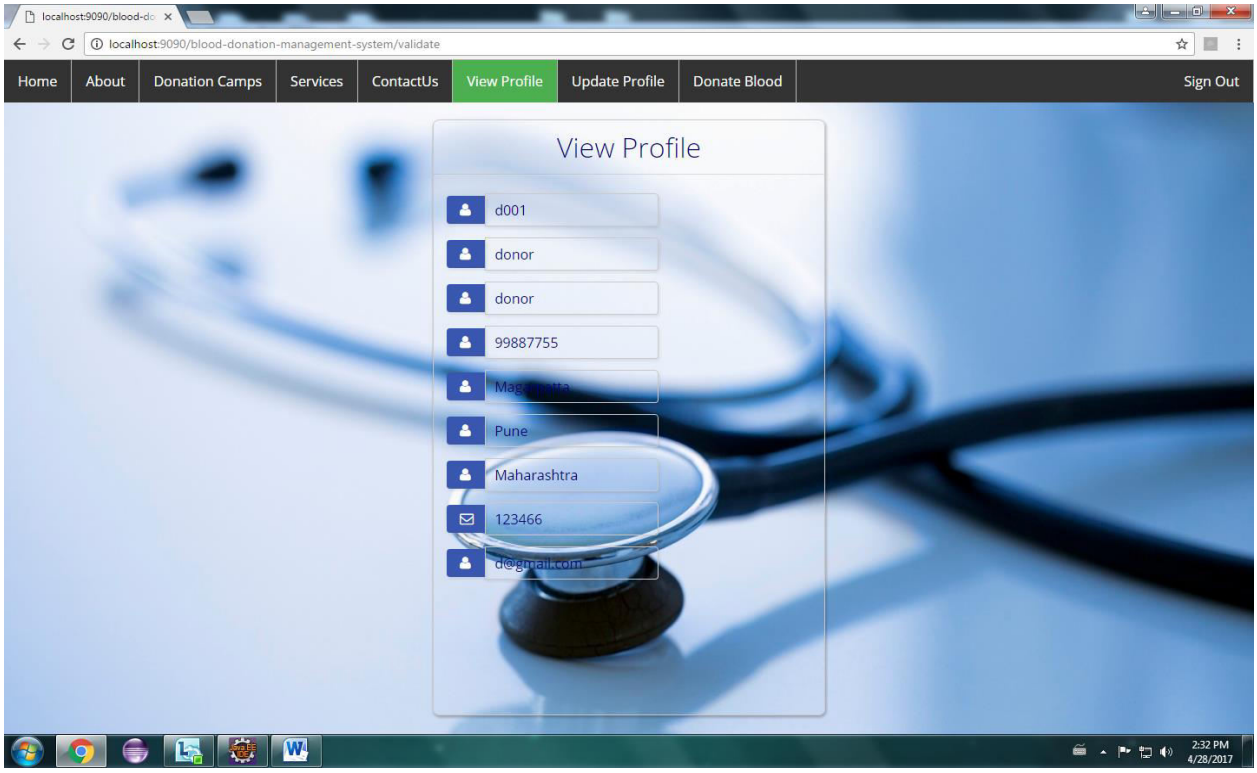


Registeration Form

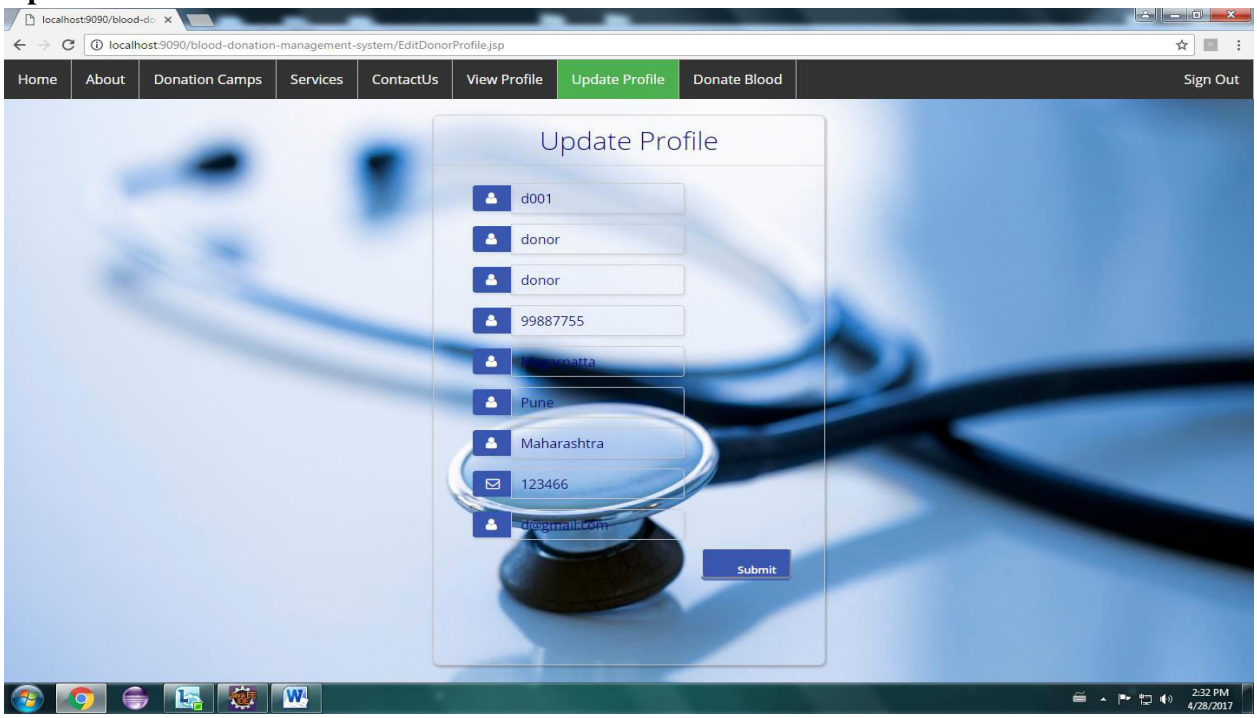


For Donor

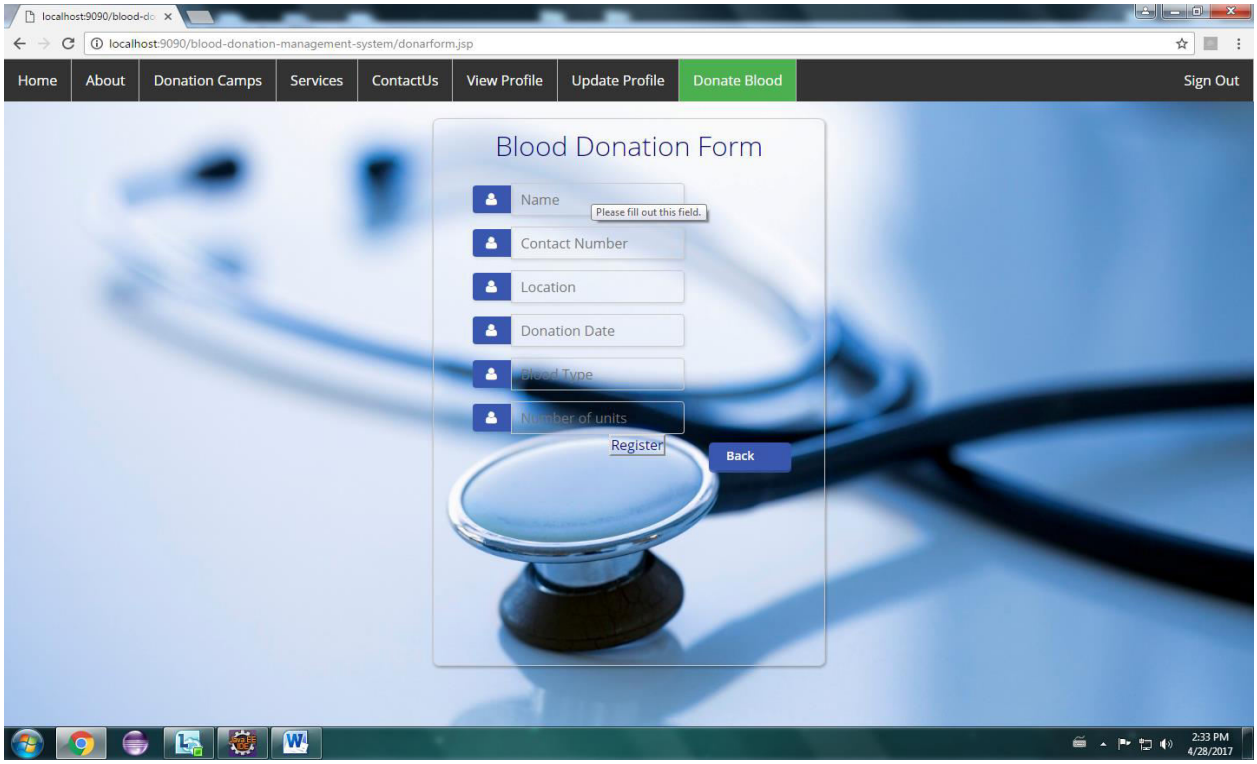
View Profile



Update Profile

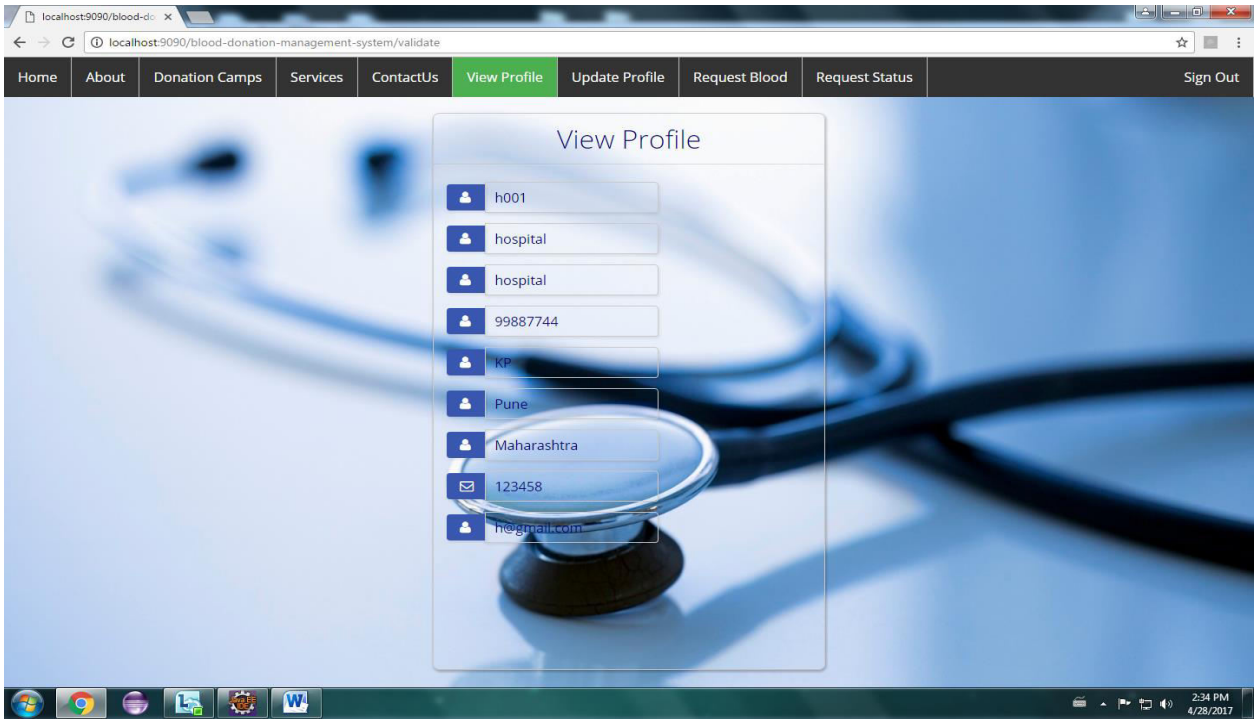


Blood Donation Form

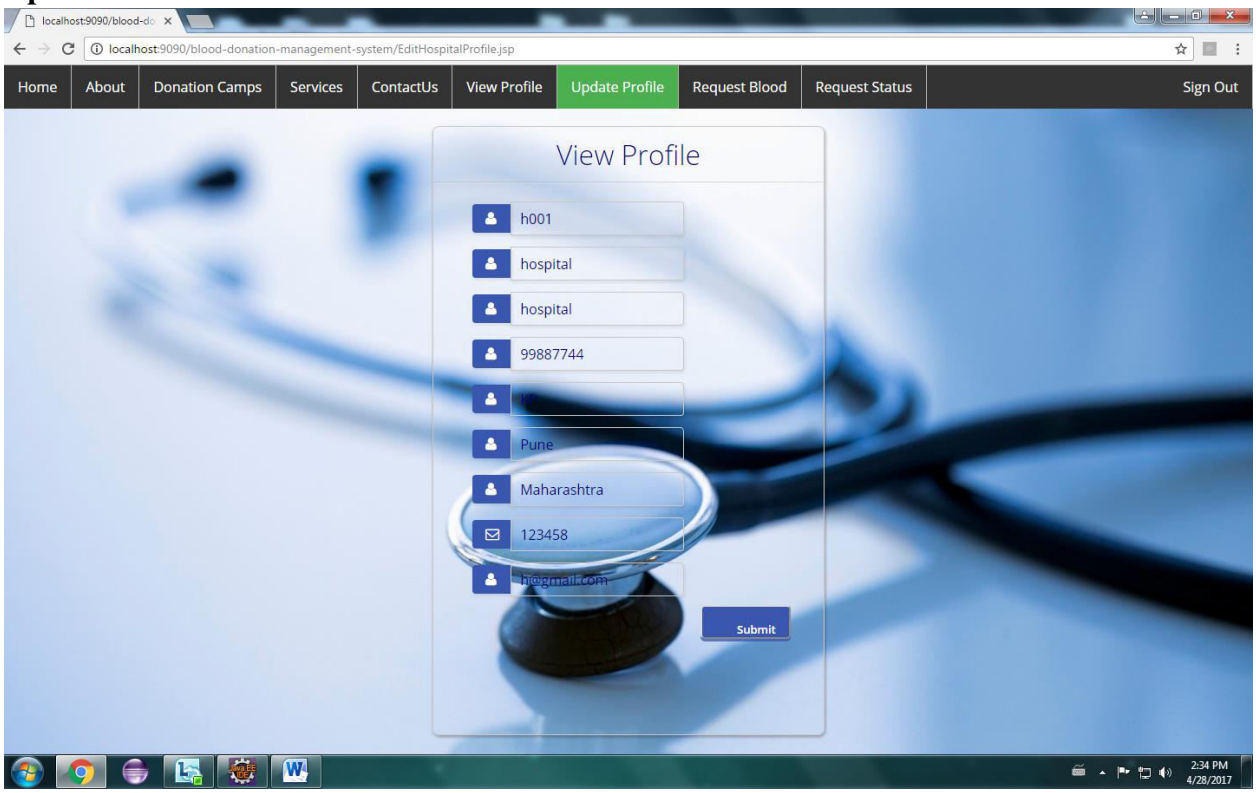


For Hospital

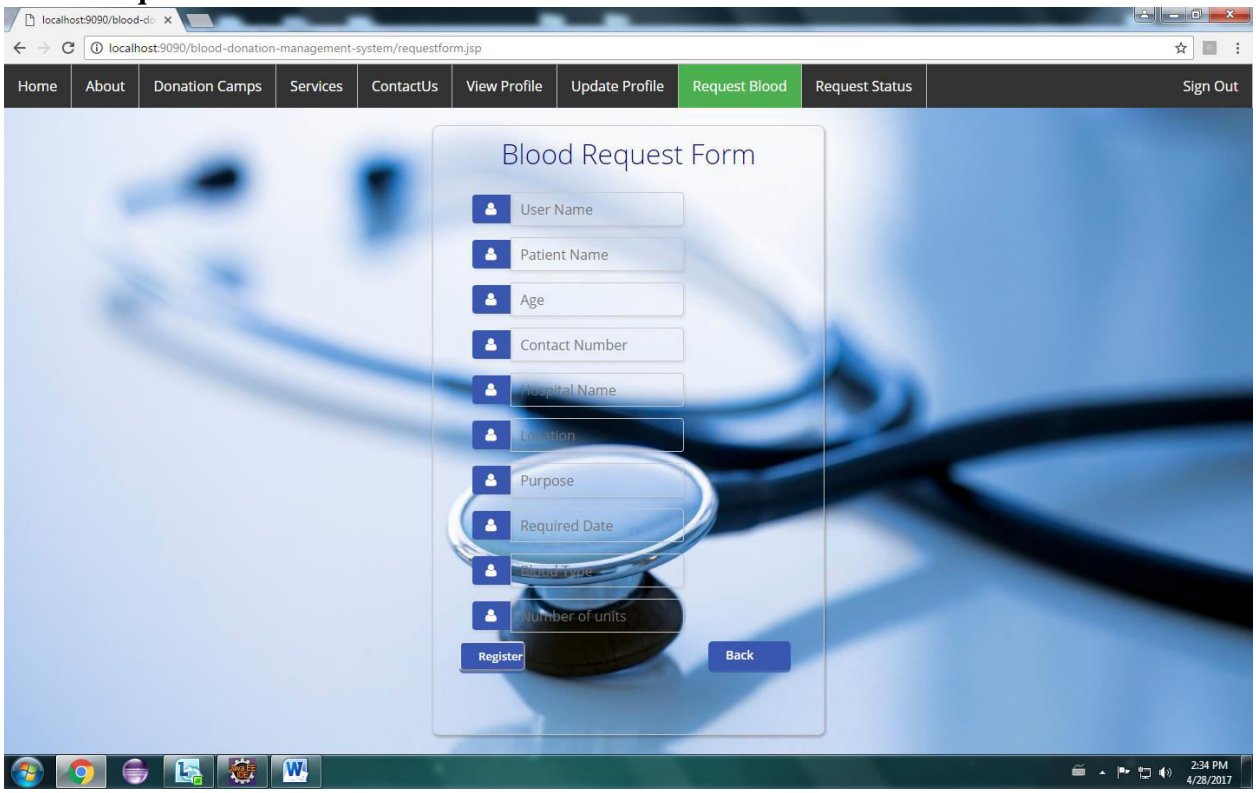
View Profile



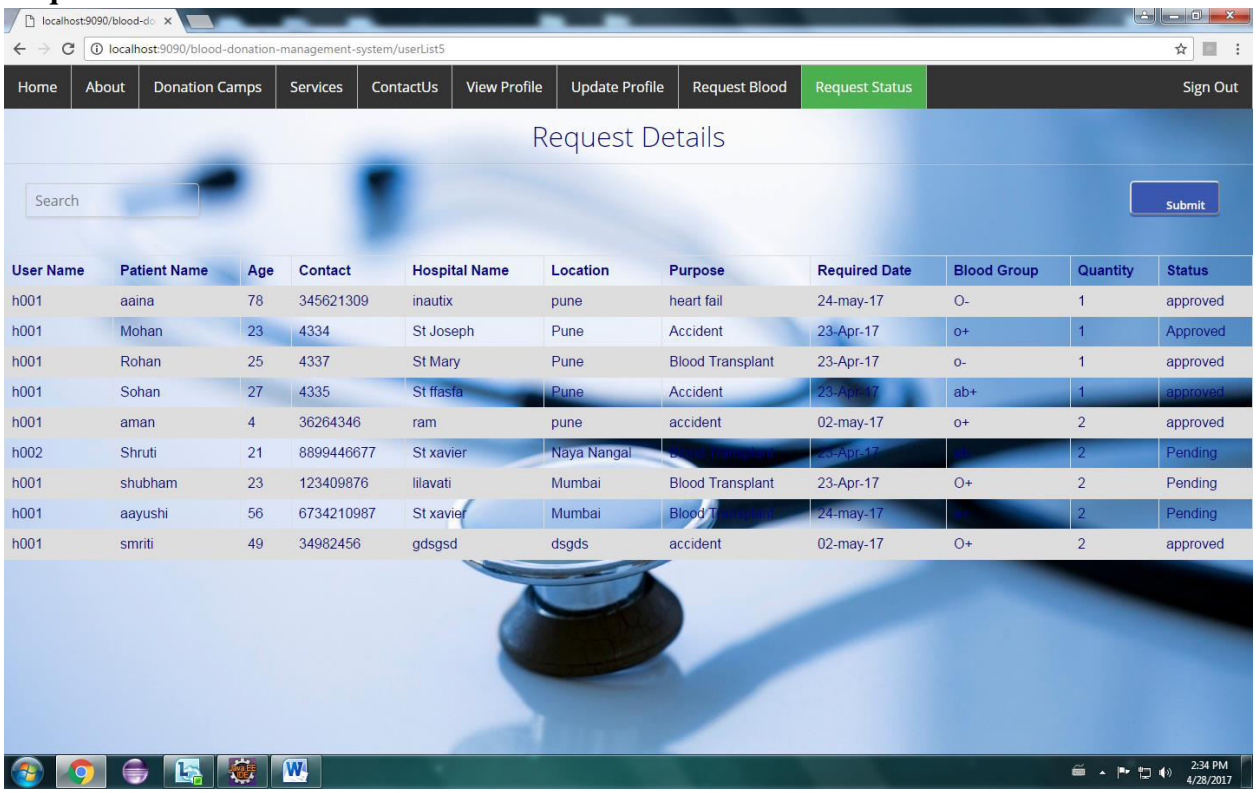
Update Profile

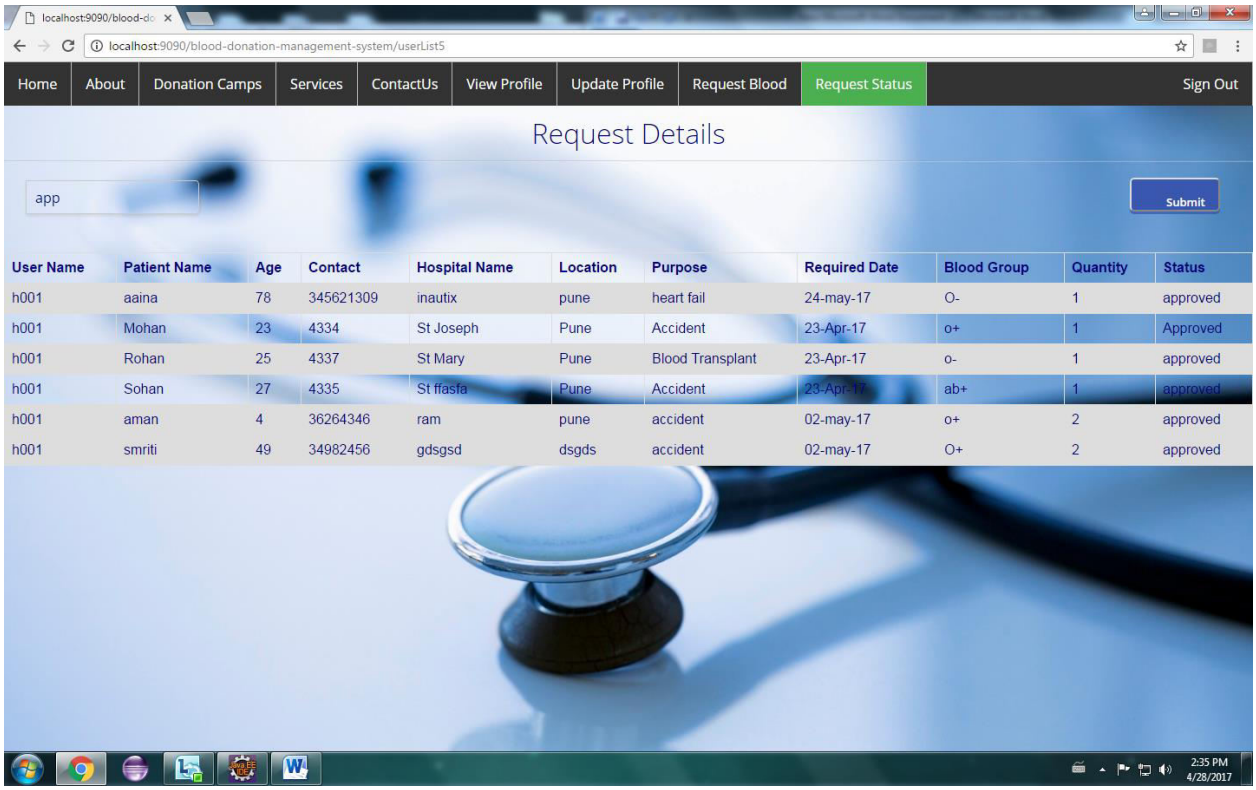


Blood Request Form



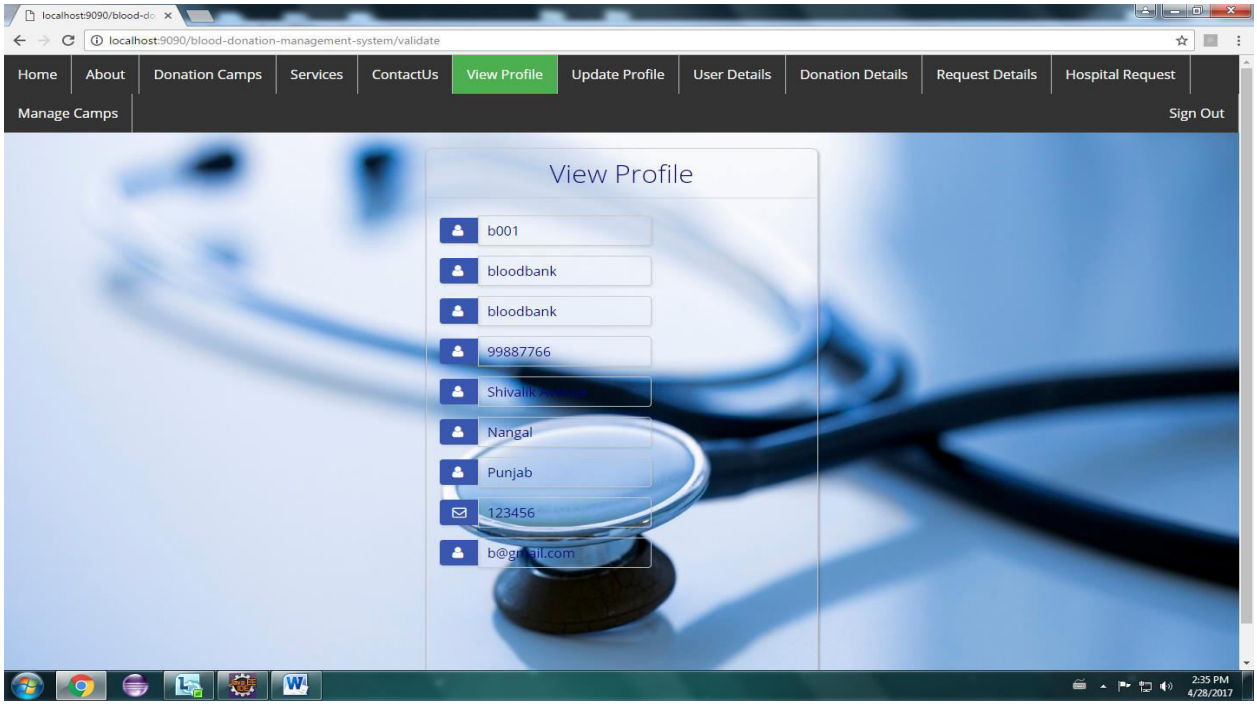
Request Status



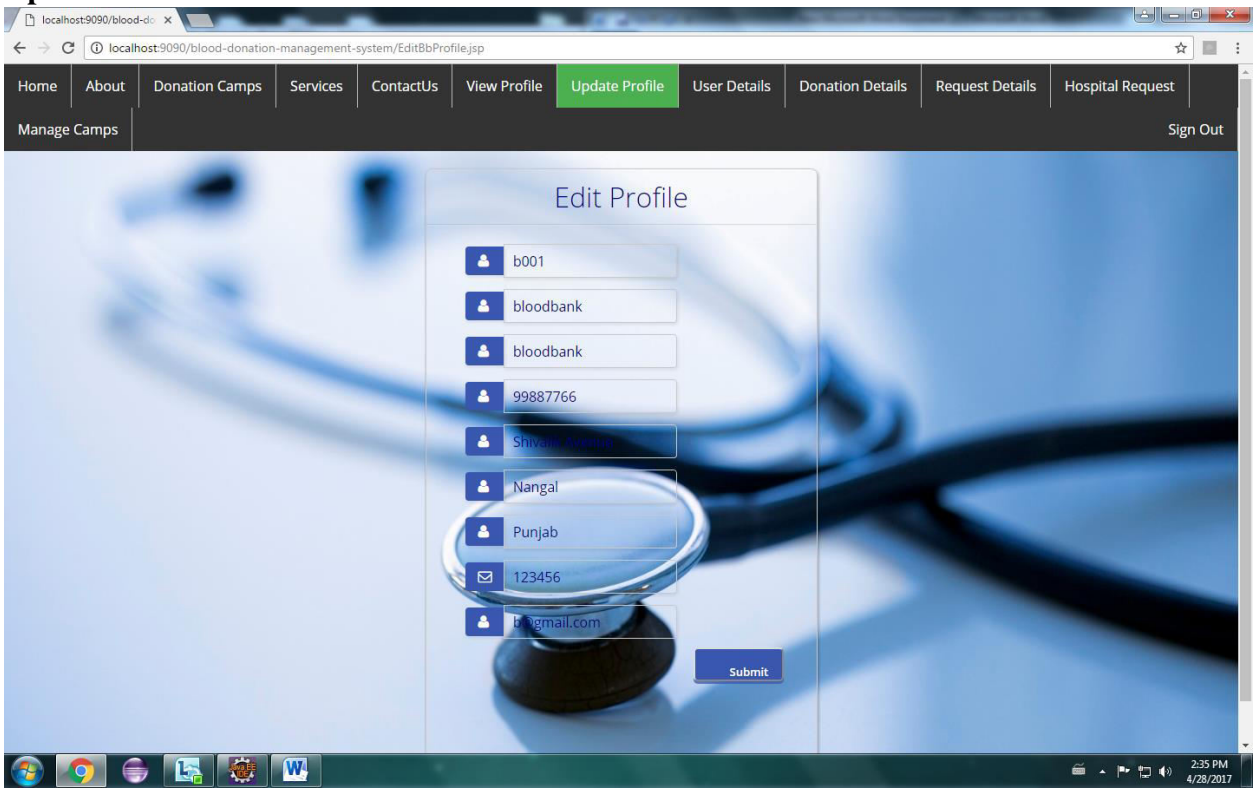


For Blood Bank

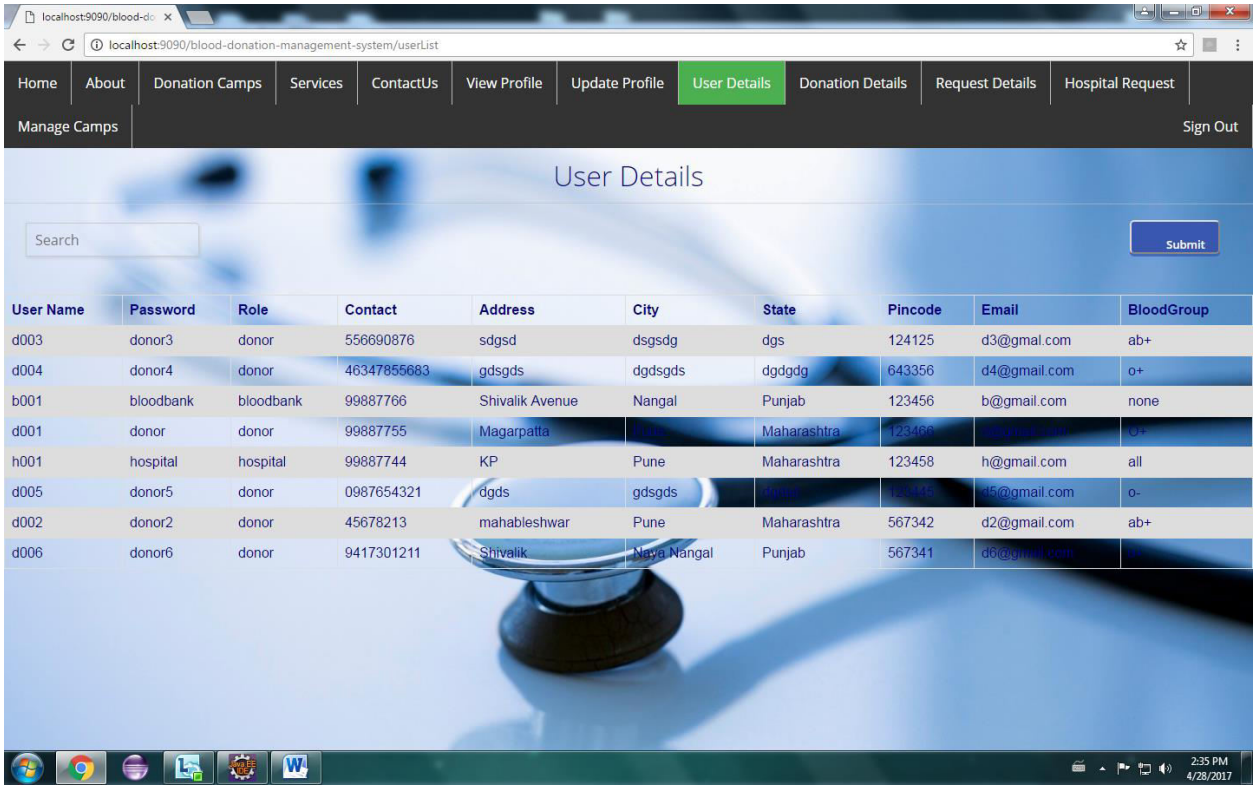
View Profile



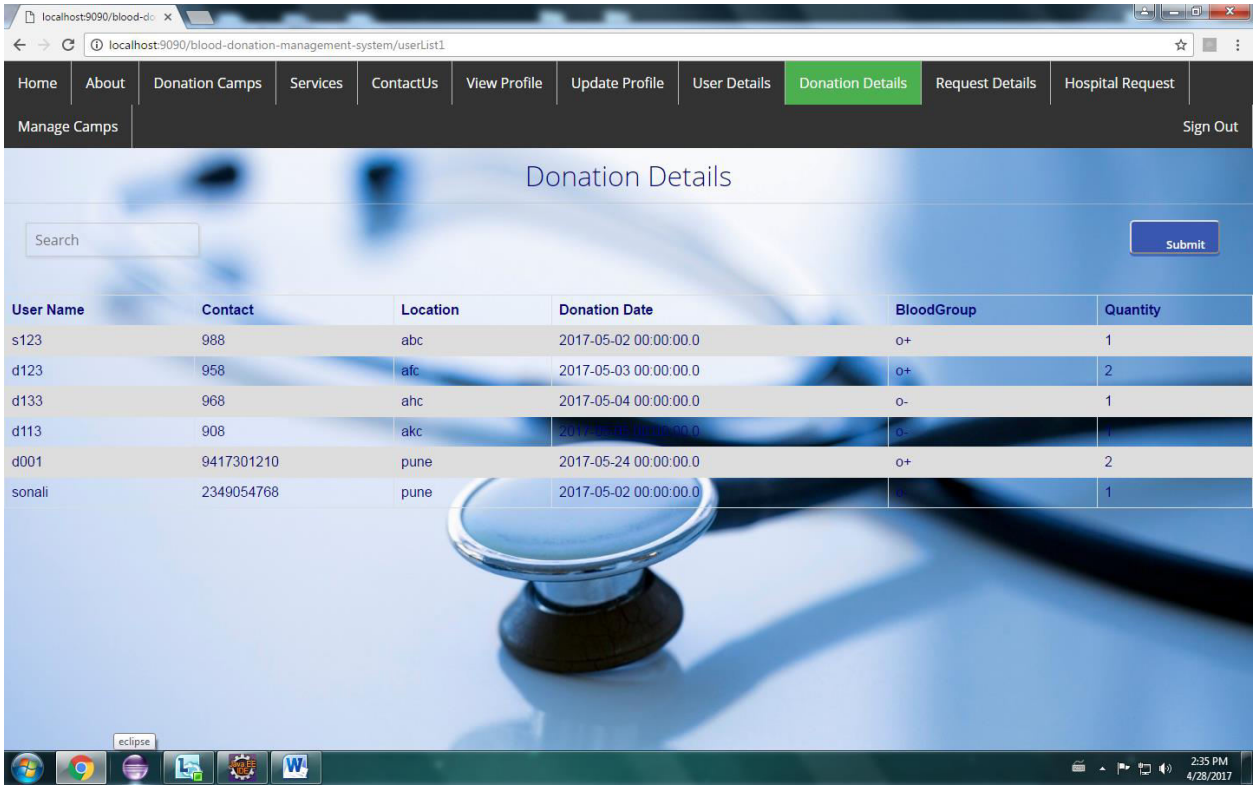
Update Profile

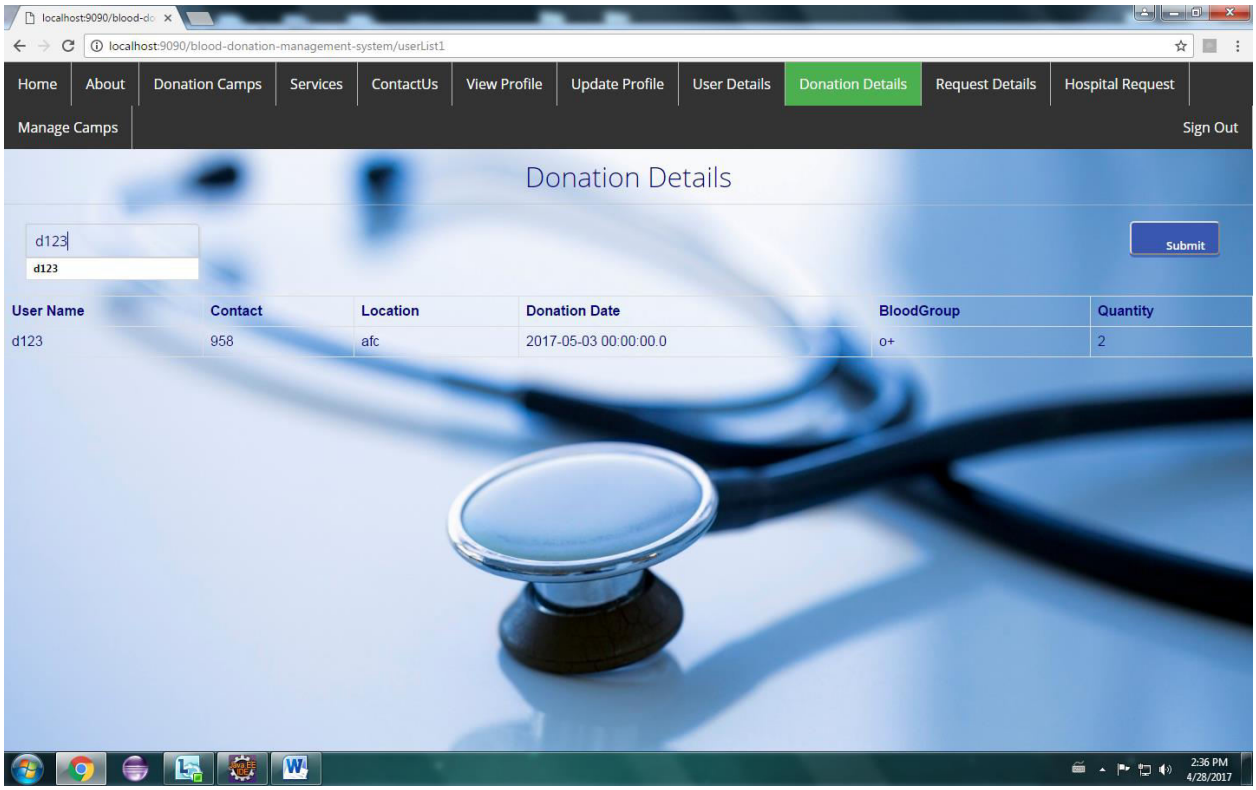


User Details

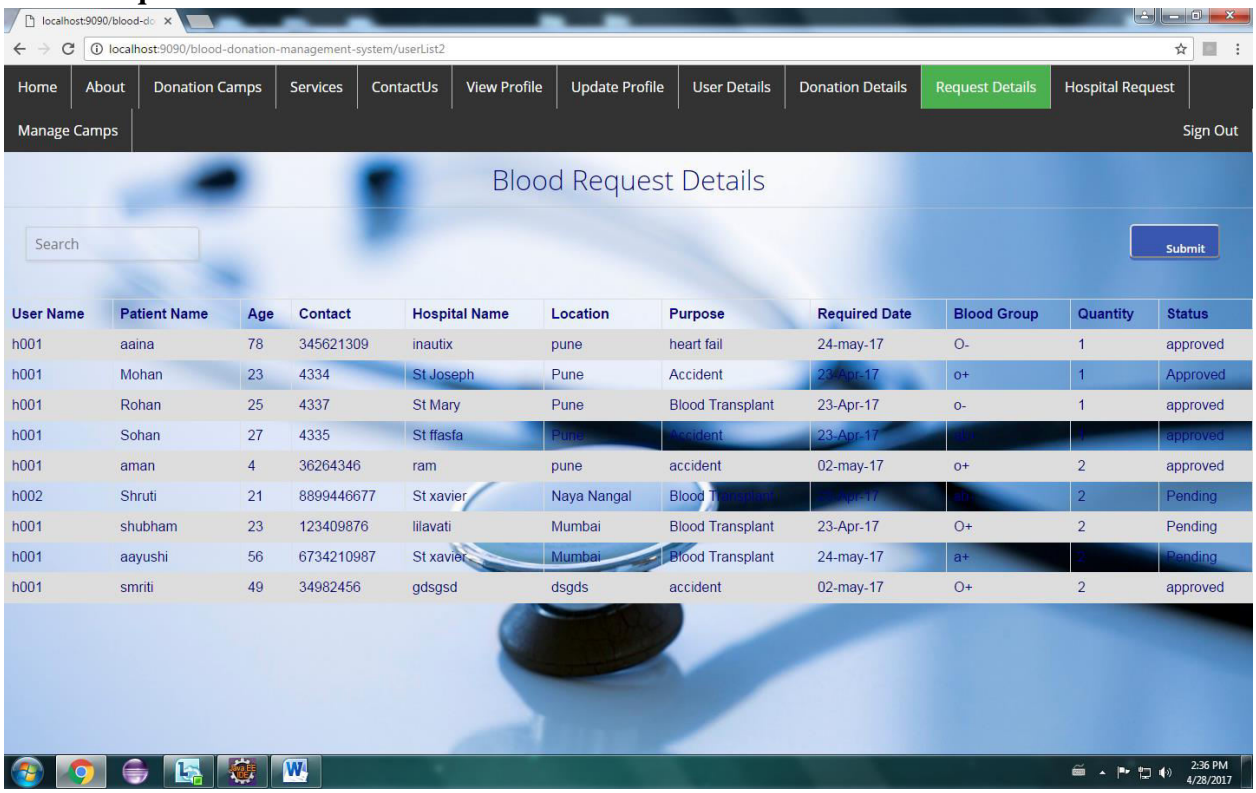


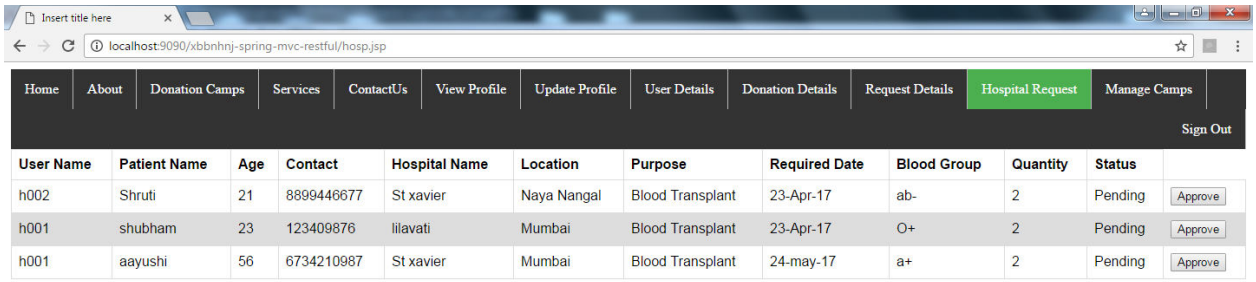
Blood Donation Details



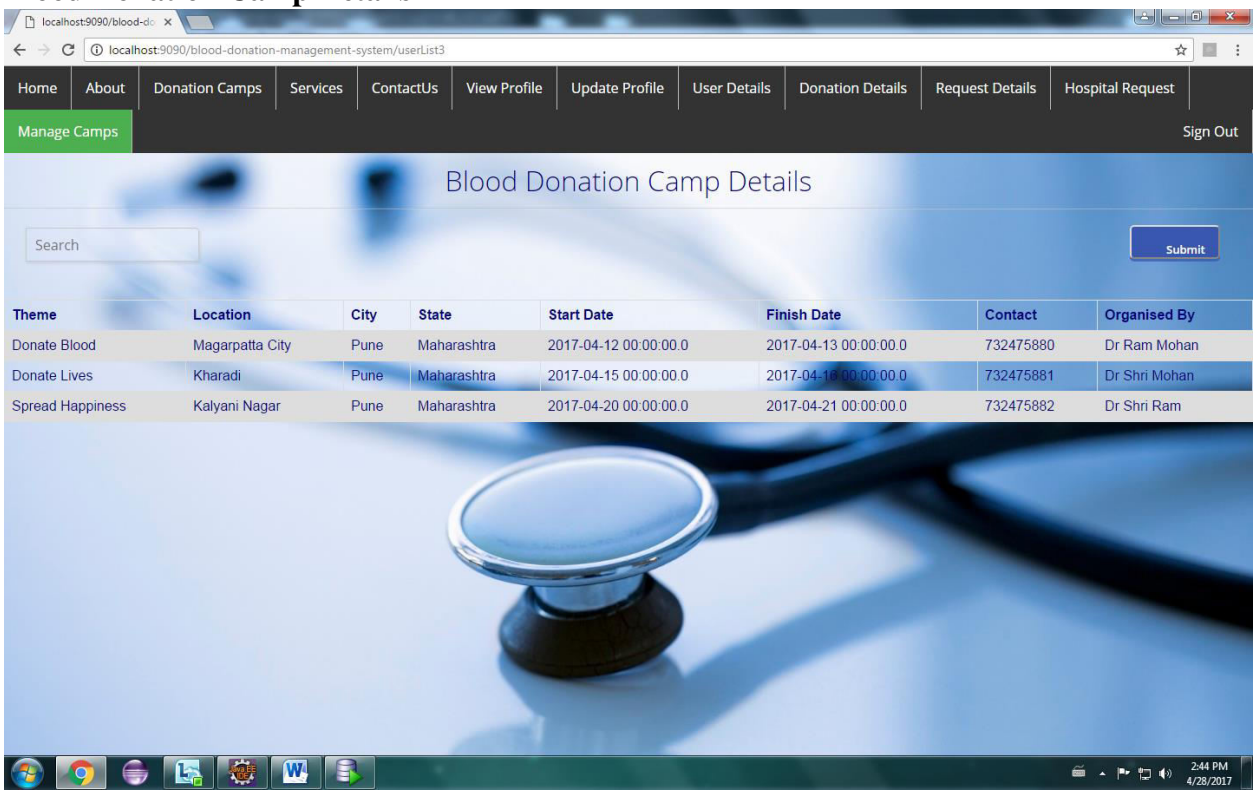


Blood Request Details





Blood Donation Camp Details



9. ***Future Enhancements***

- Future Enhancements will be making an Android application and IOS application.
- Also to include search option for patients.
- Making reports
- Blog to share ideas on projects
- Area oriented project.

10. Conclusion

- We got the experience of working in a great company iNautix A BNY Mellon company
- We got the practical experience on the latest technologies used in the companies
- We got to know the working of Java, Spring, RESTful-API, SQL
- We also the knowledge of the HTML, CSS, JS, Angular JS
- We also got some basic knowledge of business module like working of the stock market

11. Bibliography and References

- https://www.tutorialspoint.com/
- https://www.javatpoint.com/
- https://maven.apache.org/
- http://www.oracle.com/technetwork/database/databasetechnologies/sql/overview/index.html
- https://w3layouts.com/