

# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI-590014



A DBMS Mini Project Report On

## "LIBRARY MANAGEMENT SYSTEM(LMS)"

*Submitted in the partial fulfilment of the requirements for the award of the Degree of **Bachelor of Engineering in Information Science and Engineering***

Submitted by,

**Divya Bharathi R (10X18IS020)**

**Deepthi A (10X18IS019)**

**Ganavi (10X18IS024)**

*Under the guidance of*

**Sandhya Rani Asst. professor at ISE**



**Department of Information Science and Engineering**

**The Oxford College of Engineering, Bommanahalli, Bangalore-68**

**2020-2021**

# THE OXFORD COLLEGE OF ENGINEERING

Bommanahalli, Hosur Road, Bangalore – 560068

(Affiliated to Visvesvaraya Technological University, Belgaum)

Department of Information Science and Engineering



## CERTIFICATE

Certified that the project work entitled "**LIBRARY MANAGEMENT SYSYTEM**" carried out by **Divyabharathi (10X18IS020)**, **Deepthi (10X18IS019)**, and **Ganavi (10X18IS024)** Bonifide students of The Oxford College of Engineering, Bangalore in partial fulfilment for the award of the Degree of **Bachelor of Engineering in Information Science and Engineering of Visvesvaraya Technological University, Belgaum** during the year 2019-2020 . The Mini project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

**Mrs.SANDHYA RANI**

**Dr.KANAGAVALLI**

**Dr. A.S.ARAVIND**

**Project Guide**

**H.O.D,Dept. of ISE**

**Principal, TOCE**

### External Viva

**Name of the Examiners**

**Signature with Date**

**1.** \_\_\_\_\_

\_\_\_\_\_

**2.** \_\_\_\_\_

\_\_\_\_\_

# **Abstract**

The Library Management System(**LMS**) is used to manage the library through automated process and very minimal manual intervention is required. Because the system is taking care of library life cycle such as issue book, return book automatically. Similarly it is taking care of the library managements such as adding a book, delete and modifying the books. In addition to this system is providing high security for the system as it allows only authorized user can subscribe the books and authorized employee can manage the library. The registered customer can only view existing library books but cannot subscribe it. It can be done with help of employee. The registered employee can manage the library and can register the customer on behalf of them.

## **ACKNOWLEDGEMENT**

A project is a job of great enormity and it can't be accomplished by an individual all by them. Eventually, we are grateful to a number of individuals whose professional guidance, assistance and encouragement have made it a pleasant endeavor to undertake this project.

It gives us great pleasure in expressing our deep sense of gratitude to our respected Founder Chairman **Shri S. Narasa Raju** and to the respected Chairman **Shri S.N.V.L Narasimha Raju** for having provided us with great infrastructure and well-furnished labs.

We take this opportunity to express our profound gratitude to our respected Principal Dr.A.S. Aravind for his support.

We are graceful to the Head of the Department **Dr. Kanagavalli** for her unfailing encouragement and suggestion given to us in the course of our project work.

Guidance and deadlines play a very important role in successful completion of the project on time. We convey our gratitude to **Mrs. Sandhya Rani** Project Guide for having constantly guided and monitored the development of the project.

A note of thanks to the Department of Information science and Engineering, both teaching and non-teaching staff for their co-operation extended to us.

We thank our parents for their constant support and encouragement. Last, but not the least, we would like to thank our peers and friends.

**DIVYA BHARATHI R**

**DEEPTHI A**

**GANAV**

## Table of Contents

- 1 Introduction**
  - 1.1 Problem Statement
  - 1.2 Proposed Solution
- 2 Analysis and System Requirements**
  - 2.1 System U.I Screen
  - 2.2 Analysis survey
  - 2.3 System Requirements
- 3 *System Design and Modelling***
  - 3.1 Architecture of the Application
  - 3.2 Technical Architecture
  - 3.3 ER Diagram
  - 3.4 Normalization
    - 3.4.1 Database Table
- 4 *Implementation***
  - 4.1 Frontend
  - 4.2 Middleware
  - 4.3 Backend
  - 4.4 Pseudocode
- 5 *Testing***
  - 5.1 Introduction to Testing
- 6 *Conclusion***
- 7 *References***

# **1.INTRODUCTION**

## **1.1 Problem Statement**

The most of the librarians are struggling to manage the library because it has many manual activities such as issue the books, returning the book, and adding the books. Many times there is a possibility of losing books because the librarian is not able to maintain the return history. At the Beginning of the examination more students subscribe the book, during the vacation, more customer subscribe the book so during this time, it is indeed very difficult to maintain the library manually.

## **1.2 Proposed Solution**

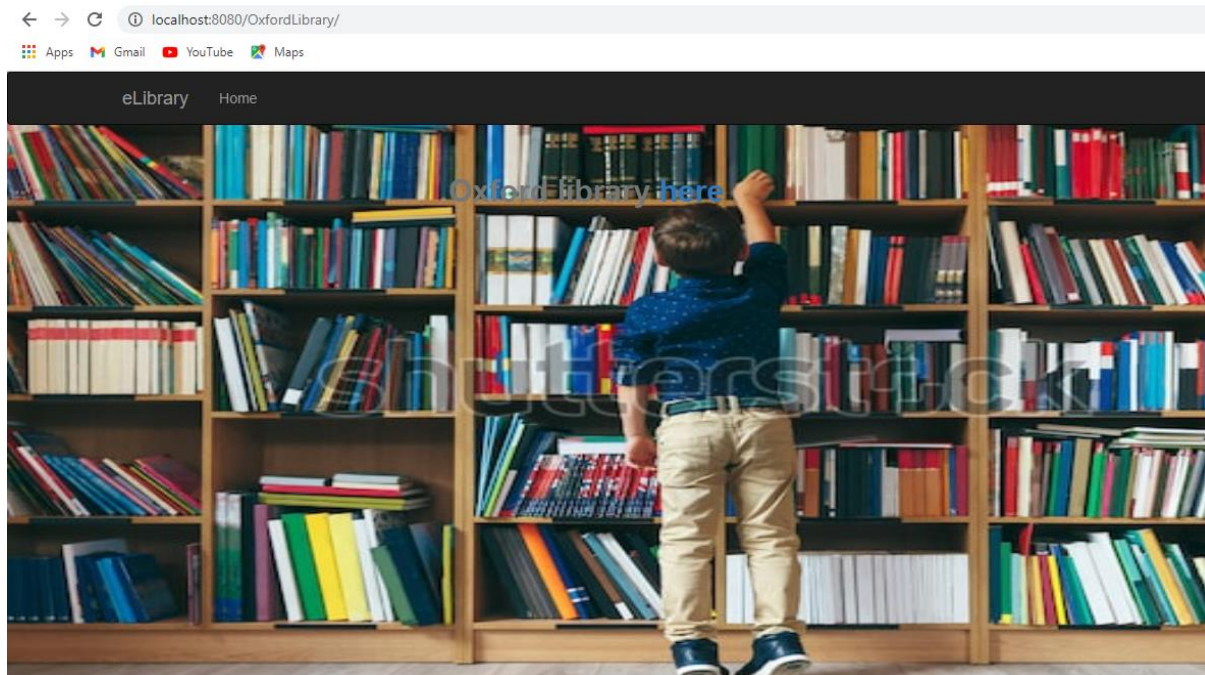
The Library Management System(**LMS**) is used to solve the many of the issues occurred in the library over manual process. It is required very minimal manual intervention. Because the system is taking care of library life cycle such as issue the book, return the book automatically. Similarly it is taking care of the library managements such as adding a book, delete and modifying the books. It is also providing high security for the system by providing the customer registration process so that it will allow only authorized customer into the system. It supports below features:

- Customer/Employee Login
- Customer/Employee Registration
- Add new books arrived in the library.
- Modify the books details
- Delete the books if it is not valid.
- View existing books in the library
- Issue the book to customer who is already subscribed.
- Return the book by customer.
- View list of due dated books and subscribed books

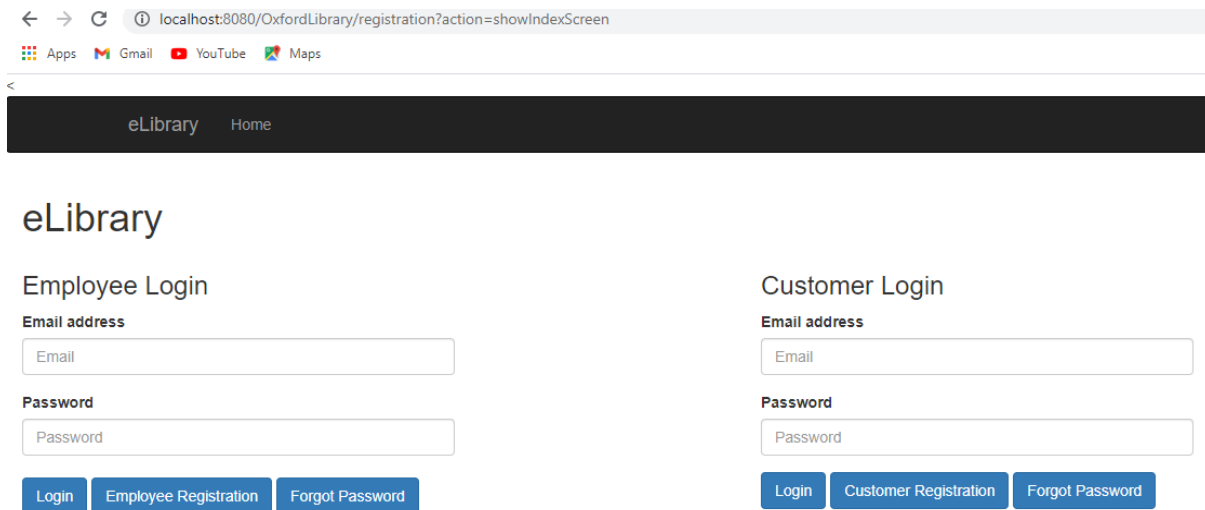
## 2. Analysis and System Requirements

### *2.1 System UI screens*

#### Home Page:



#### Login Page

A screenshot of the eLibrary Login Page. The browser's address bar shows 'localhost:8080/OxfordLibrary/registration?action=showIndexScreen'. The page has a dark navigation bar with 'eLibrary' and 'Home' links. The main content area is divided into two sections: 'Employee Login' and 'Customer Login'. Each section has a form with 'Email address' and 'Password' fields, and three buttons: 'Login', 'Registration', and 'Forgot Password'.

## Registration Page

# Customer Registration Form

First Name \*

Last Name

Password\*

Confirm Password \*

E-mail \*

Mobile \*

Address \*

Gender \*

male ☐ female ☐

Register

Reset

Home

## Library History Page

← → ↻ ⓘ localhost:8080/OxfordLibrary/userauthentication?action=E 🔍 ☆ 📱 🌐 🏠 👤 ⋮

📱 Apps 📧 Gmail 📺 YouTube 🗺 Maps

eLibrary Home

## Library Management System

isbn	book_title	category	rental_price	author	publisher		
1018	chanakya	mythology	30	ashok	hema publication	Edit	Delete
1019	mahabharatha secret	mythology	30	christopher	hema publication	Edit	Delete
1020	the blue umbrella	novel	30	ruskin bond	rupa publication	Edit	Delete
1021	india after gandhi	history	30	ramchandhar guha	haper collins	Edit	Delete
1022	discovery of india	history	30	jawaharlal nehru	som	Edit	Delete
1023	three men in boat	comedy	30	pegasus	ram publications	Edit	Delete
1024	stories of birbal	comedy	30	ananth pai	lakshmi publications	Edit	Delete
1025	kalki	mythology	30	kelvin missal	deepa publication	Edit	Delete
1026	the last mughal	novel	30	william datrymple	sonu publication	Edit	Delete
1030	wings of fire	novel	30	abdul kalam	arun publications	Edit	Delete
1031	dd	test1	33	test1	test1test1test1	Edit	Delete
1032	chemistry	science fiction	30	123	123	Edit	Delete

View of Due Dated Books

View of Subscribed Books

Return books

Add Book

Issue books

Customer Registration

Logout



## AddBook Page

[←](#) [→](#) [↻](#) [localhost:8080/OxfordLibrary/registration?action=showAddBookPage](#)

[Apps](#) [Gmail](#) [YouTube](#) [Maps](#)

Home

### Add Book

**Title**

**Category**

**Rental\_price**

**Author**

**Publisher**

[Add book](#) [Reset](#) [Home](#)

## Issue Books

[←](#) [→](#) [↻](#) [localhost:8080/OxfordLibrary/librarymanagement?action=fwdAssgnbookPage](#)

[Apps](#) [Gmail](#) [YouTube](#) [Maps](#)

eLibrary Home Employee Librarian

# eLibrary

## Assign Book

Isbn

Phone Number

Assign

Reset

Home

## List of Subscribed Books

[←](#) [→](#) [↻](#) [localhost:8080/OxfordLibrary/librarymanagement?action=viewsubsbook](#) [☆](#) [📄](#) [👤](#) [⋮](#)

[Apps](#) [Gmail](#) [YouTube](#) [Maps](#)

eLibrary Home

## Library Management System

isbn	book_title	category	rental_price	author	publisher	Subscriber Name	Subscriber mobile	Subscriber Mail-Jd	
1025	kalki	mythology	30	kelvin missal	deepa publication	divya999	9898	mahendran_raja2003@rediffmail.com	<a href="#">Return</a>
1021	india after gandhi	history	30	ramchandhar guha	haper collins	Mahendran	9686	mahendranhp@gmail.com	<a href="#">Return</a>

View of Due Dated Books

View of Subscribed Books

Return books

Add Book

Issue books

Customer Registration

Logout

## 2.2 Analysis Survey

- The feasibility study is a major factor which contributes to the analysis and development of the system. The decision of the system analyst (whether to design a particular system or not) depends on its feasibility study.
- Study of requirement analysis is done through different feasibility studies.
- Feasibility studies are undertaken whenever a requirement to improve the existing system or designing new system is necessary. Feasibility study helps to meet user requirements.
- It enables us to determine the potential of existing systems and helps in improving the existing system. It helps to develop a technically feasible system. It helps to know what should be embedded in the new system and helps develop a cost-effective system.

The project concept is feasible because of the following:

- **Technical Feasibility:**

Technical feasibility is the study of hardware and software requirements on which the application is to be used upon. It is important to consider the budget of the system and overall cost. This system is technically feasible as it has been developed with the help of available technology. The proposed system requires Net Beans and backend tool as MySQL for storing and maintain the database. No expert man power is required for this purpose.

- **Economic Feasibility:**

This area is concerned with cost for the development and implementation of the system. We have looked into the maintenance of the system post development. So, the actual cost of the system is important before designing a new system which was perfectly estimated by us. All this is studied in economic feasibility study. The designed system will provide tangible as well as intangible benefits to the organization.

- **Operation Feasibility:**

Automation makes lives easy. The proposed system is very friendly and the user is easily able to interact with the system. Therefore, the users will readily accept the system. Data entry and making queries can be done easily.

## **2.3 System Requirements**

**Definition of Software:** Computer software or just software is any set of machine-readable instructions that directs a computer's processor to perform specific operations. The term is used to construct with computer hardware, the physical objects (processor and related devices) that carry out the instructions. Computer hardware and software require each other and neither can be realistically used without the other.

### **Software Specification:**

Operating System: Windows 10

Front End: HTML, cascading style sheet(CSS), java script, and Java server page (JSP).

Middleware : Tomcat 7.0

Rear End: MYSQL

**Definition of Hardware:** Computer hardware is the collection of physical elements that constitute a computer system. Computer hardware refers to the physical components of a computer such as the monitor, mouse, keyboard, computer data storage, hard drive disk, system unit (graphic cards, memory, motherboard and chips) etc. all of which are physical objects that can be touched. In contrast, software is instructions that can be stored and run by hardware.

## Hardware Specification:

Processor: AMD E2 900, 1.5 GHZ

RAM 4GB

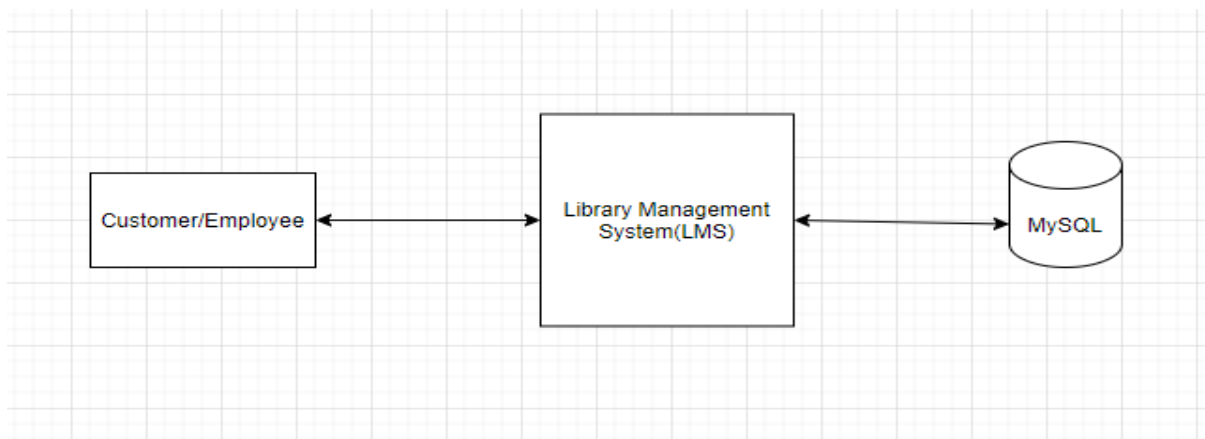
Monitor: VGA/SVGA

Keyboard: 104 keys standard

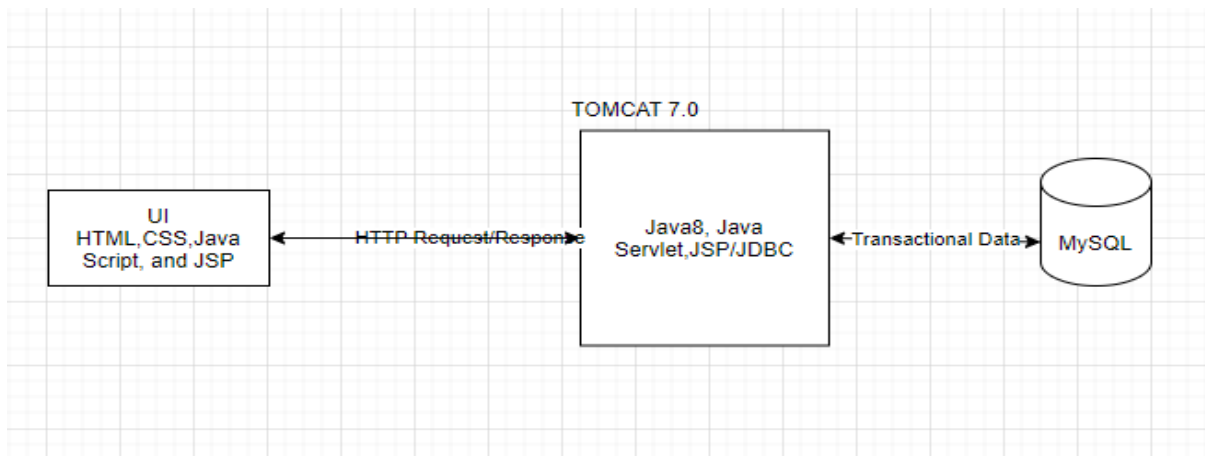
Mouse: 2/3 button. Optical/Mechanical.

## **3.System Design & Modelling**

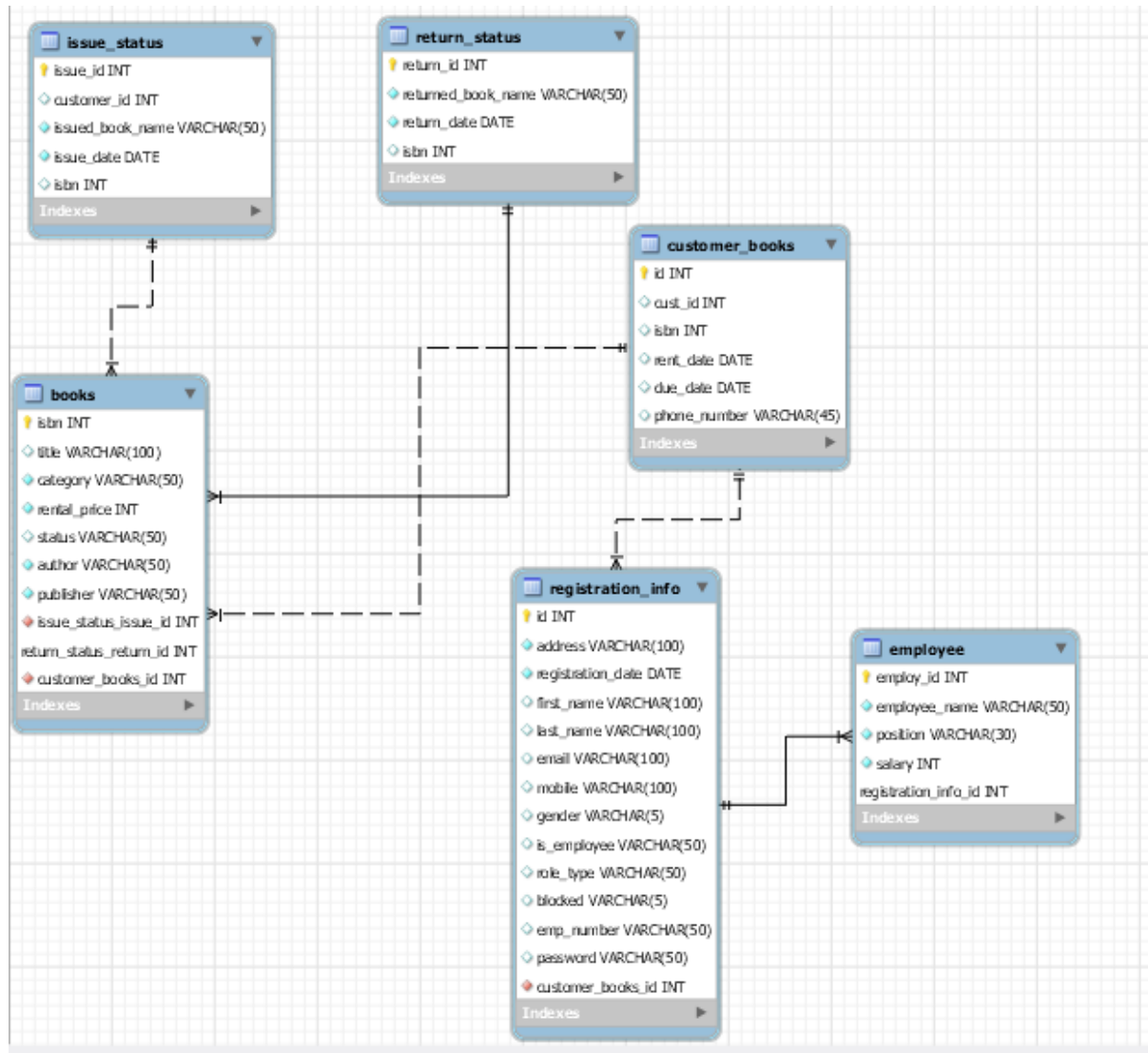
### **3.1 Architecture of the Application**



### **3.2 Technical Architecture**



### 3.1.2 ER Diagram



### 3.4 Normalization

#### 3.4.1 Database Tables

Table1:Books

Field	Type	Null	Key	Default	Extra
isbn	int	NO	PRI	NULL	
book_title	varchar(50)	NO		NULL	
category	varchar(50)	NO		NULL	
rental_price	int	NO		NULL	
status	varchar(50)	YES		NULL	
author	varchar(50)	NO		NULL	
publisher	varchar(50)	NO		NULL	

7 rows in set (0.62 sec)

Table2:Employee

Field	Type	Null	Key	Default	Extra
employ_id	int	NO	PRI	NULL	
employ_name	varchar(50)	NO		NULL	
position	varchar(30)	NO		NULL	
salary	int	NO		NULL	

4 rows in set (0.22 sec)

Table3:customer

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
cust_id	int	YES		NULL	
isbn	int	YES		NULL	
rent_date	date	YES		NULL	
due_date	date	YES		NULL	
phone_number	varchar(45)	YES		NULL	

6 rows in set (0.38 sec)

Table4:Issue Status

Field	Type	Null	Key	Default	Extra
issue_id	int	NO	PRI	NULL	
customer_id	int	YES	MUL	NULL	
issued_book_name	varchar(50)	NO		NULL	
issue_date	date	NO		NULL	
isbn	int	YES	MUL	NULL	

5 rows in set (0.07 sec)

Table 5:Registration Info

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
address	varchar(100)	NO		NULL	
registration_date	date	NO		NULL	
first_name	varchar(100)	YES		NULL	
last_name	varchar(100)	YES		NULL	
email	varchar(100)	YES		NULL	
mobile	varchar(100)	YES		NULL	
gender	varchar(5)	YES		NULL	
is_employee	varchar(50)	YES		NULL	
role_type	varchar(50)	YES		NULL	
blocked	varchar(5)	YES		NULL	
emp_number	varchar(50)	YES		NULL	
password	varchar(50)	YES		NULL	

13 rows in set (0.06 sec)

Table 6:Return status

Field	Type	Null	Key	Default	Extra
return_id	int	NO	PRI	NULL	
customer_id	int	YES	MUL	NULL	
returned_book_name	varchar(50)	NO		NULL	
return_date	date	NO		NULL	
isbn	int	YES		NULL	

5 rows in set (0.05 sec)



## **4.Implementation**

### **4.1 Frontend**

The front end technologies such as HTML, CSS, and java scripts are used to develop the user interface (UI) that are rendered in the browser, is used to make the static html pages into more dynamic for user interaction.

The java scripts is used to validate the HTML fields like customer name , mail id and phone number. It ensures that customer or employee is providing the correct information for example, if user does not provide the valid email id then system used pop up the error message to correct it and it ensures that user cannot proceed to next level until they provide a valid information.

### **4.2 Middleware:**

The middleware used for LMS is Java8, Servlet, JSP and Tomcat.

The middleware technologies is used to process the HTTP transactions like it receive the HTTP request and send to corresponding service class to proceed the business process. It returns the HTTP responses after processing the request. The tomcat middleware server is used to host the java web based application.

### **A Basic introduction to Java**

Java is a simple and yet powerful object-oriented programming language and it is in many respects similar to C++. Java originated at Sun Microsystems, Inc. in 1991.

Java was designed with a concept of 'write once and run everywhere'. Java Virtual Machine plays the central role in this concept. The JVM is the environment in which Java programs execute. It is a software that is implemented on top of real hardware and operating system. When the source code (.java files) is compiled, it is translated into byte codes and then placed

into (.class) files. The JVM executes these bytecodes. So, Java byte codes can be thought of as the machine language of the JVM. A JVM can either interpret the bytecode one instruction at a time or the bytecode can be compiled further for the real microprocessor using what is called a just-in-time compiler. The JVM must be implemented on a particular platform before compiled programs can run on that platform.

### **Types of Java Applications:**

- **Web Application** - Java is used to create server-side web applications. Currently, Servlet, JSP, Struts, JSF, etc. technologies are used.
- **Standalone Application** - It is also known as the desktop application or window-based application. An application that we need to install on every machine or server such as media player, antivirus, etc. AWT and Swing are used in java for creating standalone applications.
- **Enterprise Application** - An application that is distributed in nature, such as banking applications, etc. It has the advantage of the high-level security, load balancing, and clustering. In Java, EJB is used for creating enterprise applications.
- **Mobile Application** - Java is used to create application software for mobile devices. Currently, Java ME is used for building applications for small devices, and also Java is a programming language for Google Android application development.

### **Features of Java:**

- **Object-Oriented** - Java supports the features of object-oriented programming. Its object model is simple and easy to expand.
- **Platform independent** - C and C++ are platform dependency languages hence the application programs written in one Operating system cannot run in any other Operating system, but in platform independence language like Java application programs written in one Operating system can able to run on any Operating system.
- **Simple** - Java has included many features of C / C ++, which makes it easy to understand.

- **Secure** - Java provides a wide range of protection from viruses and malicious programs. It ensures that there will be no damage and no security will be broken.
- **Portable** - Java provides us the concept of portability. Running the same program with Java on different platforms is possible.
- **Robust** - During the development of the program, it helps us to find possible mistakes as soon as possible.
- **Multi-threaded** - The multithreading programming feature in Java allows you to write a program that performs several different tasks simultaneously.
- **Distributed** - Java is designed for distributed Internet environments as it manages the TCP/IP protocol.

## 4.3 Backend

### *A Basic introduction to MYSQL*

MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. This tutorial will give you a quick start to MySQL and make you comfortable with MySQL programming.

- Tables to store data.
- Queries to find and retrieve specific data of interest.
- Forms to view, add, and update data in tables.
- Reports to analyses or print data in a specific layout.
- Data access pages to view or update, the data.

In MySQL, data is stored once in one table, but can be viewed from multiple locations. When the data is updated in a Table, Query or Form, it is automatically updated everywhere it appears.

## **Establishment of MySQL database Connection**

A connection is a computer science facility that allows the user to connect with the database server software. A user can connect with the database server, whether on the same machine or remote locations. Therefore, if we want to work with the database server to send commands and receive answers in the form of a result set, we need connections. In this article, we are going to learn how we can connect to MySQL Server in various ways.

### MySQL Connection Types

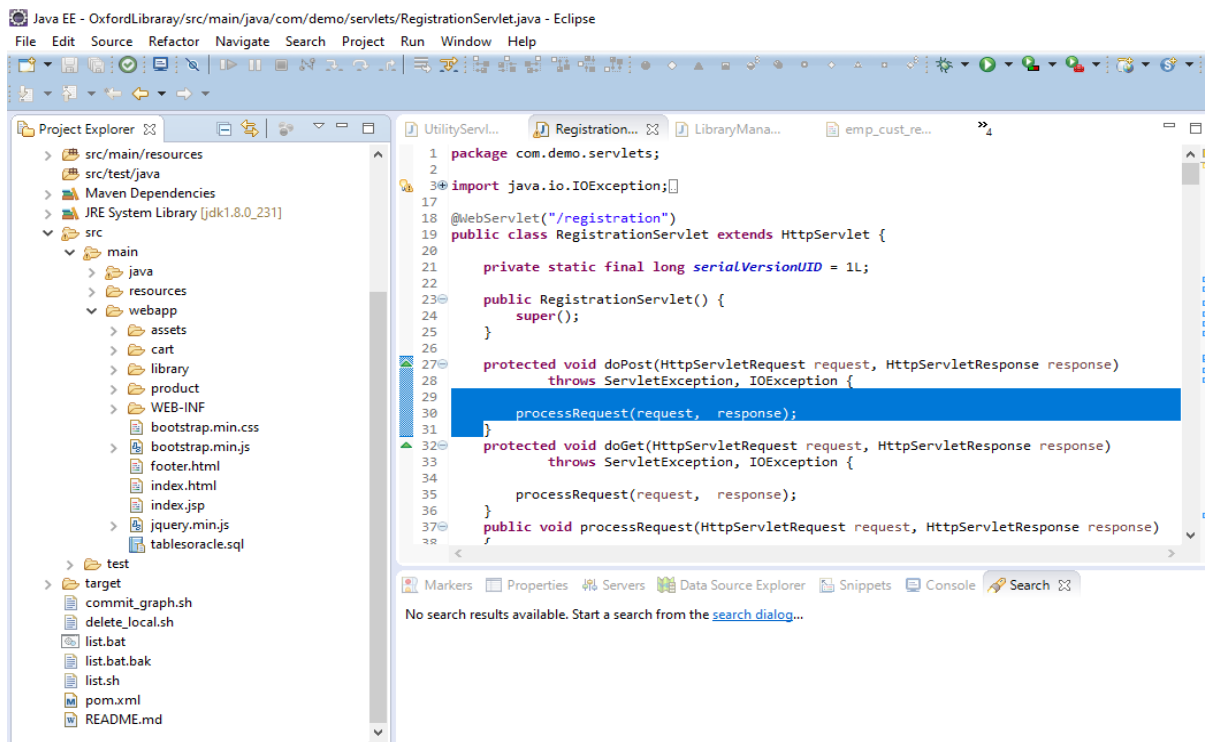
MySQL provides various ways to connect with the database server. Once we have installed the MySQL server, we can connect it using any of the client programs that are listed below:

Command-line client

MySQL Workbench

PHP Script.

## **Application Folder Structure**



## 4.4 Pseudocode

Front End:

```
<html>
<head>
<meta charset="ISO-8859-1">
<title>Customer Registration</title>
<link rel="stylesheet" href="bootstrap.min.css"/>
</head>
<script type = "text/javascript">
function validateForm() {

    var x = document.forms[0]["pass"].value;
    var y = document.forms[0]["cpass"].value;
    /* alert(x);
    alert(y); */

    if (x != "" && x != "") {

        //alert("inside 1"+x);
        var ans = x.localeCompare(y);

        if( ans != 0)
        {
            alert("Password and confirm password should be same");
            document.forms[0]["cpass"].value="";
            document.forms[0]["cpass"].focus();
            return false;
        }

    }

}
```

```

</script>
<body>

<nav class="navbar navbar-inverse">
  <div class="container">
    <!-- Brand and toggle get grouped for better mobile display -->
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="index.html">eLibrary</a>
    </div>

    <!-- Collect the nav links, forms, and other content for toggling -->
    <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
      <ul class="nav navbar-nav">
        <li><a href="./registration?action=home" class="active">Home</a></li>
      </ul>

    </div><!-- /.navbar-collapse -->
  </div><!-- /.container-fluid -->
</nav>

<form action="./registration?action=empcustregistration" method="post" >

  <table border="0" align="center" width="500" bgcolor="#CCCCCC" >

```

```

    <h2><b><%=request.getAttribute("errorinfo") !=
null?request.getAttribute("errorinfo"): ""%></b></h2>

<caption><h2><b>Customer Registration On Behalf of Customer</b></h2></caption>

<tr>

<th>First Name*<h4 style="color:red;"></h4></th>

<td><input type="text" name="fn" id="fn1" maxlength="100" required/></td>

</tr>

<tr>

<th>Last Name</th>

<td><input type="text" name="ln"/></td>

</tr>

<tr>

<th>Password*</th>

<td><input type="password" name="pass" maxlength="100" required/></td>

</tr>

<tr>

<th>Confirm Password*</th>

<td><input type="password" name="cpass" maxlength="100" required/></td>

</tr>

<tr>

<th>E-mail*</th>

<td><input type="email" name="email" maxlength="100" required/></td>

</tr>

<tr>

<th>Mobile*</th>

<td><input type="number" name="mobile" maxlength="100" required/></td>

</tr>

<tr>

<th>Address*</th>

<td><textarea rows="8" cols="20" name="address" maxlength="100"
required></textarea></td>

</tr>

```





## Java Code

RegistrationServlet.java

```
package com.demo.servlets;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.demo.entities.Books;
import com.demo.entities.Employee;
import com.demo.entities.Item;
import com.demo.entities.RegistrationInfo;
import com.demo.models.ProductModel;
import com.demo.services.RegistrationService;

@WebServlet("/registration")
public class RegistrationServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    public RegistrationServlet() {
        super();
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }
}
```

```

protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}

public void processRequest(HttpServletRequest request, HttpServletResponse
response)
{
    String action = request.getParameter("action");
    System.out.println("++++++++++"+action);
    if (action.equalsIgnoreCase("empregistration"))
    {
        empRegistration(request,response);
    }
    if (action.equalsIgnoreCase("showCustRegistrationPage"))
    {
        new UtilityServlet().returnToCustRegisterPage(request,response);
    }
    if (action.equalsIgnoreCase("custregistration"))
    {
        userRegistration(request,response);
    }
    if (action.equalsIgnoreCase("empcustregistration"))
    {
        custRegistrationOnBehalfOfEmployee(request,response);
    }
    if (action.equalsIgnoreCase("showAddBookPage"))
    {
        showAddBookPage(request,response);
    }
}

```

```

        if (action.equalsIgnoreCase("procesForgotPassword"))
        {
            procesForgotPassword(request,response);
        }

        if (action.equalsIgnoreCase("custregistrationbehalfemp"))
        {
            userRegistrationOnBehalfOfEmployee(request,response);
        }
        if (action.equalsIgnoreCase("empverification"))
        {
            empValidation(request,response);
        }
        if (action.equalsIgnoreCase("home"))
        {
            homePage(request,response);
        }
        if (action.equalsIgnoreCase("allhistoryPage"))
        {
            new UtilityServlet().returnToAllBookHistory(request, response);
        }

    }

    private void empValidation(HttpServletRequest request, HttpServletResponse
response)
    {
        RegistrationService registrationService = new RegistrationService();

        Employee employee = new Employee();
        employee.setEmployeeId(request.getParameter("empid"));

```

```

        employee.setEmployeeName(request.getParameter("empname"));
        System.out.println("emp id "+request.getParameter("empid"));
        System.out.println("empname "+request.getParameter("empname"));
        boolean isEmpValid = registrationService.isValidEmployee(employee);
        System.out.println("isEmpValid "+isEmpValid);
        if(isEmpValid)
        {
            try {
                request.getSession().setAttribute("empinfo",employee);

                request.getRequestDispatcher("library/EmployeeRegister.jsp").forward(request,
response);
            } catch (ServletException | IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        else
        {
            try
            {
                request.setAttribute("errorinfo","Please provide valid
employee details");

                request.getRequestDispatcher("library/employee_info.jsp").forward(request,
response);
            } catch (ServletException | IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}

```

```

    }

    private void showEmployeeInfoPage(HttpServletRequest request,
    HttpServletResponse response)
    {
        try {
            request.getRequestDispatcher("library/employee_info.jsp").forward(request,
            response);

                } catch (ServletException | IOException e) {
                    e.printStackTrace();
                }

        }

        private void showAddBookPage(HttpServletRequest request, HttpServletResponse
        response)
        {
            try {
                request.getRequestDispatcher("library/add_book.jsp").forward(request, response);

                    } catch (ServletException | IOException e) {
                        e.printStackTrace();
                    }

        }

        private void showForgotPasswordPage(HttpServletRequest request,
        HttpServletResponse response)
        {
            try {

                request.getRequestDispatcher("library/forgot_password.jsp").forward(request,
                response);

                    } catch (ServletException | IOException e) {
                        e.printStackTrace();
                    }

        }

```

```

    }

    private void showIndexScreen(HttpServletRequest request, HttpServletResponse
response)
    {
        try {
            request.getRequestDispatcher("library/index.html").forward(request,
response);
        } catch (ServletException | IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    private void procesForgotPassword(HttpServletRequest request,
HttpServletResponse response)
    {
        RegistrationService registrationService = new RegistrationService();
        String fname = request.getParameter("fname");
        String mail = request.getParameter("mail");
        String mobile = request.getParameter("mobile");
        String password =
registrationService.fetchForgotPassword(fname,mail,mobile);
        System.out.println("isExist "+password);

        if(password != null && password.length()>0)
        {
            new UtilityServlet().showPasswordPage(request, response,password);
        }
        else
        {

```

```
String message= "There is no user available for first name = "+fname+",  
mail id ="+mail+", and mobile #: "+mobile +" So please provide valid details to get the  
password..";
```

```
request.setAttribute("forgotpassworderrmsg", message);
```

```
try {
```

```
request.getRequestDispatcher("library/forgot_password.jsp").forward(request,  
response);
```

```
    } catch (ServletException e) {
```

```
        // TODO Auto-generated catch block
```

```
        e.printStackTrace();
```

```
    } catch (IOException e) {
```

```
        // TODO Auto-generated catch block
```

```
        e.printStackTrace();
```

```
    }
```

```
}
```

```
}
```

```
private void showCustregistrationScreen(HttpServletRequest request,  
HttpServletResponse response)
```

```
{
```

```
    try {
```

```
request.getRequestDispatcher("library/emp_cust_register.jsp").forward(request,  
response);
```

```
    } catch (ServletException | IOException e) {
```

```
        // TODO Auto-generated catch block
```

```
        e.printStackTrace();
```

```
    }
```

```
}
```

```
private void homePage(HttpServletRequest request, HttpServletResponse response)
```

```

{

    try {

        request.getRequestDispatcher("library/index.html").forward(request,
response);

    } catch (ServletException | IOException e) {

        // TODO Auto-generated catch block
        e.printStackTrace();

    }

}

private void userRegistration(HttpServletRequest request, HttpServletResponse
response)
{

    RegistrationService registrationService = new RegistrationService();

    RegistrationInfo registrationInfo = new RegistrationInfo();
    registrationInfo.setFirstName(request.getParameter("fn"));
    registrationInfo.setLastName(request.getParameter("ln"));
    registrationInfo.setPassword(request.getParameter("pass"));
    registrationInfo.setEmail(request.getParameter("email"));
    registrationInfo.setGender(request.getParameter("gender"));
    registrationInfo.setAddress(request.getParameter("address"));
    registrationInfo.setDob(request.getParameter("dob"));
    registrationInfo.setCountry(request.getParameter("country"));
    registrationInfo.setMobile(request.getParameter("mobile"));

    String mobile = request.getParameter("mobile");
    String email = request.getParameter("email");
    boolean isExist = registrationService.isCustomerRegistered(mobile,email);

```



```

        System.out.println("isExist "+isExist);

        if(!isExist)
        {
            registrationService.registerUser(registrationInfo);

            String message = "Customer "+request.getParameter("fn")+" is
successfully registered .";

            new UtilityServlet().returnToCustRegSuccessPage(request,
response,message);
        }
        else
        {
            new UtilityServlet().returnToCusttRegisterPagewithError(request,
response, email,mobile);// change the jsp url
        }

    }

    private void custRegistrationOnBehalfofEmployee(HttpServletRequest request,
    HttpServletResponse response)
    {
        RegistrationService registrationService = new RegistrationService();

        RegistrationInfo registrationInfo = new RegistrationInfo();
        registrationInfo.setFirstName(request.getParameter("fn"));
        registrationInfo.setLastName(request.getParameter("ln"));
        registrationInfo.setPassword(request.getParameter("pass"));
        registrationInfo.setEmail(request.getParameter("email"));
        registrationInfo.setGender(request.getParameter("gender"));
        registrationInfo.setAddress(request.getParameter("address"));
        registrationInfo.setDob(request.getParameter("dob"));
        registrationInfo.setCountry(request.getParameter("country"));
    }

```

```

        registrationInfo.setMobile(request.getParameter("mobile"));

String mobile = request.getParameter("mobile");
String email = request.getParameter("email");
boolean isExist = registrationService.isCustomerRegistered(mobile,email);
System.out.println("isExist "+isExist);

if(!isExist)
{
    registrationService.registerUser(registrationInfo);
    String message = "Customer "+request.getParameter("fn")+" is
successfully registered .";
    new UtilityServlet().returnToSuccessPage(request, response,
message);
}
else
{
    new
UtilityServlet().returnToCustttRegisterPagewithEmpwithError(request, response,
email,mobile);// change the jsp url
}

}

private void userRegistrationOnBehalfOfEmployee(HttpServletRequest request,
HttpServletResponse response)
{
    RegistrationService registrationService = new RegistrationService();

String mobile = request.getParameter("mobile");

```

```

String email = request.getParameter("email");

boolean isExist = registrationService.isCustomerRegistered(mobile,email);

System.out.println("isExist "+isExist);


if(!isExist)
{
    RegistrationInfo registrationInfo = new RegistrationInfo();
    registrationInfo.setFirstName(request.getParameter("fn"));
    registrationInfo.setLastName(request.getParameter("ln"));
    registrationInfo.setPassword(request.getParameter("pass"));
    registrationInfo.setEmail(request.getParameter("email"));
    registrationInfo.setGender(request.getParameter("gender"));
    registrationInfo.setAddress(request.getParameter("address"));
    registrationInfo.setDob(request.getParameter("dob"));
    registrationInfo.setCountry(request.getParameter("country"));
    registrationInfo.setMobile(request.getParameter("mobile"));
    registrationService.registerUser(registrationInfo);


    ProductModel productModel = new ProductModel();
    request.setAttribute("books", productModel.findAll());
    new UtilityServlet().returnToHome(request, response);
}
else
{
    new
UtilityServlet().returnToCustttRegisterPagewithEmpwithError(request, response,
email,mobile);// change the jsp url
}
}

private void empRegistration(HttpServletRequest request, HttpServletResponse
response)

```

```

{
    RegistrationService registrationService = new RegistrationService();

    String mobile = request.getParameter("mobile");
    String email = request.getParameter("email");
    boolean isExist = registrationService.isCustomerRegistered(mobile,email);
    System.out.println("isExist "+isExist);

    if(!isExist)
    {
        RegistrationInfo registrationInfo = new RegistrationInfo();
        registrationInfo.setFirstName(request.getParameter("fn"));
        registrationInfo.setLastName(request.getParameter("ln"));
        registrationInfo.setPassword(request.getParameter("pass"));
        registrationInfo.setEmail(request.getParameter("email"));
        registrationInfo.setGender(request.getParameter("gender"));
        registrationInfo.setAddress(request.getParameter("address"));
        registrationInfo.setDob(request.getParameter("dob"));
        registrationInfo.setCountry(request.getParameter("country"));
        registrationInfo.setMobile(request.getParameter("mobile"));
        registrationInfo.setIsEmployee("Y");

        Employee employee =
(Employee)request.getSession().getAttribute("empinfo");

        System.out.println("employee "+employee);

        System.out.println("employee.getId()
"+employee.getId());

        System.out.println("employee.getEmployeeType()
"+employee.getEmployeeType());

        registrationInfo.setIsEmployee("Y");
        registrationInfo.setEmpNumber(employee.getId());
    }
}

```

```

        registrationInfo.setIsAdmin(employee.getEmployeeType().equalsIgnoreCase("A")?"Y
        ":"N");

        registrationService.registerEmployee(registrationInfo);

        String message = "Employee "+request.getParameter("fn")+" is
        successfully registered .";

        new UtilityServlet().returnToSuccessPage(request, response,
        message);
    }
    else
    {
        new UtilityServlet().returnToCustRegisterPagewithError(request,
        response, email,mobile);
    }
    /*try {

        request.getRequestDispatcher("library/library_managment.html").forward(request,
        response);

        } catch (ServletException | IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }*/
    }

}

```

## **RegistrationDAOImpl.java**

```

package com.demo.model.dao;

```

```

import java.sql.Connection;

```

```

import java.sql.PreparedStatement;

```

```

import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import com.demo.entities.Books;
import com.demo.entities.Employee;
import com.demo.entities.RegistrationInfo;
import com.demo.models.DatabaseConnection;

public class RegistrationDAOImpl {

    public RegistrationDAOImpl() {

    }

    public List findAll ()
    {
        Connection con = null;

        Statement st = null;
        ResultSet rs = null;
        List<Books> li = new ArrayList<Books>();
        try
        {
            // Initialize the database
            con = DatabaseConnection.getConnection();
            st = con.createStatement();
            rs = st.executeQuery("select * from libprjct.books");

```

```

while (rs.next()) {

    System.out.println(rs.getString(1));
    Books book = new Books();
    book.setIsbn(rs.getInt("isbn"));
    book.setAuthor(rs.getString("author"));
    book.setBook_title(rs.getString("book_title"));
    book.setCategory(rs.getString("category"));
    book.setPublisher(rs.getString("publisher"));
    book.setRental_price(rs.getInt("rental_price"));
    book.setStatus(rs.getString("status"));
    li.add(book);
}

```

```

    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    finally
    {

        try {
            //con.close();
            rs.close();
            st.close();
        } catch (SQLException e) {

```

```

        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
System.out.println("====Total size :"+li.size());
return li;
}
public boolean isValidEmployee(Employee employee)
{
    int i=0;
    Connection con = null;
    // PreparedStatement ps = null;
    Statement st = null;
    ResultSet rs = null;
    try
    {
        System.out.print("Start of user registration");
        // Initialize the database
        con = DatabaseConnection.getConnection();
        System.out.print("getEmployeeId.." + employee.getEmployeeId());
        System.out.print("getEmployeeName.." + employee.getEmployeeName());
        String query = "SELECT position FROM libprjct.employee where
employ_id="+employee.getEmployeeId() + " and
employee_name='"+employee.getEmployeeName()+"'";
        System.out.print("query.." + query);
        st = con.createStatement();
        rs = st.executeQuery(query);

        if(rs.next())
        {

```



```

        System.out.print("valid employee..");

employee.setEmployeeType(rs.getString("position").equalsIgnoreCase("manager")?"M":"L")
;

        return true;
    }

```

```

        System.out.print("You are successfully verified...");
return false;
    }
    catch(Exception e)
    {
        System.out.print("Exception occurred"+e.getMessage());
        e.printStackTrace();
    }
    finally
    {
        try {
            rs.close();
            st.close();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    return false;
}

```

```

public String fetchForgotPassword(String fname,String mail,String mobile)

```

```

{

    Connection con = null;
    Statement st = null;
    ResultSet rs = null;
    String password="";
    try
    {
        System.out.print("Start of user fetchForgotPassword");
        // Initialize the database
        con = DatabaseConnection.getConnection();
        System.out.print("mobile.." +mobile);

        String query = "select password from libprjct.registration_info where
upper(first_name)='"+fname.toUpperCase()+" and upper(email)='"+mail.toUpperCase()+"
and mobile='"+mobile+"'";

        System.out.print("query.." +query);
        st = con.createStatement();
        rs = st.executeQuery(query);

        if(rs.next())
        {
            password = rs.getString("password");
            System.out.print("valid employee.." +password);
            return password;
        }
        System.out.print("Customer are successfully verified...");
    }
    return password;
}
catch(Exception e)

```

```

        {
            System.out.print("Exception occurred"+e.getMessage());
            e.printStackTrace();
        }
        finally
        {
            try {
                rs.close();
                st.close();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        return password;
    }

    public boolean isCustomerRegistered(String mobile, String email)
    {
        Connection con = null;
        Statement st = null;
        ResultSet rs = null;
        try
        {
            System.out.print("Start of user isCustomerRegistered");
            // Initialize the database

            con = DatabaseConnection.getConnection();
            System.out.print("mobile.." + mobile);
            System.out.print("email.." + email);

            String query = "select email,mobile from libprjct.registration_info where mobile
            ='" + mobile + "' and upper(email)='" + email.toUpperCase() + "'";

```

```

System.out.print("query.." + query);

st = con.createStatement();
rs = st.executeQuery(query);

        if(rs.next())
        {
            System.out.print("valid employee.." + rs.getString("mobile"));
            return true;
        }
        System.out.print("Customer are successfully verified...");
return false;
    }
    catch(Exception e)
    {
        System.out.print("Exception occurred" + e.getMessage());
        e.printStackTrace();
    }
    finally
    {
        try {
            rs.close();
            st.close();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    return false;
}

```

```

public int registerUser(RegistrationInfo registrationInfo)
{
    int i=0;
    Connection con = null;
    //PreparedStatement ps = null;
    Statement st = null;
    ResultSet rs = null;
    try
    {
        System.out.print("Start of user registration");
        // Initialize the database
        con = DatabaseConnection.getConnection();

        st = con.createStatement();
        rs = st.executeQuery("SELECT max(id) as id FROM libprjct.registration_info");
        int id = 0;
        if(rs.next())
        {
            id = rs.getInt("id");
        }
        id = id+1;
        System.out.print("id..." + id);

        registrationInfo.setId(id);

        PreparedStatement ps = con.prepareStatement("insert into
libprjct.registration_info(id,first_name,last_name,mobile,address,registration_date,email,gender,dob,country,role_type,password) values (?,?,?,?,?,?,?,?,?,?,?)");
        ps.setInt(1,id );
        ps.setString(2, registrationInfo.getFirstName());
    }
}

```

```

ps.setString(3, registrationInfo.getLastName());
ps.setString(4, registrationInfo.getMobile());
ps.setString(5, registrationInfo.getAddress());

java.sql.Date sqlDate = new java.sql.Date(new java.util.Date().getTime());
// java.sql.Date sqlDate = new java.sql.Date(myDate.getTime());

ps.setDate(6, sqlDate);
ps.setString(7, registrationInfo.getEmail());
ps.setString(8, registrationInfo.getGender());
ps.setString(9, registrationInfo.getDob());
ps.setString(10, registrationInfo.getCountry());
ps.setString(11, "C");
ps.setString(12, registrationInfo.getPassword());

i=ps.executeUpdate();
if(i>0)
    System.out.print("You are successfully registered...");

}
catch(Exception e)
{
    System.out.print("Exception occurred"+e.getMessage());
    e.printStackTrace();
}
finally
{
    try {
        //con.close();
    }
}

```

```

        rs.close();
        st.close();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
return i;
}

```

```

public int registerEmployee(RegistrationInfo registrationInfo)
{
    int i=0;
    Connection con = null;
    PreparedStatement ps = null;
    Statement st = null;
    ResultSet rs = null;
    try
    {
        System.out.print("Start of user registration");
        // Initialize the database
        con = DatabaseConnection.getConnection();

        st = con.createStatement();
        rs = st.executeQuery("SELECT max(id) as id FROM libprjct.registration_info");
        int id = 0;
        if(rs.next())
        {

```

```

        id = rs.getInt("id");
    }

```

```

System.out.print("id..." + id);

```

```

        ps = con.prepareStatement("insert into
libprjct.registration_info(id,first_name,last_name,mobile,address,registration_date,email,gender,dob,country,role_type,password,is_employee,emp_number) values
(?,?,?,?,?,?,?,?,?,?,?,?,?)");

```

```

        ps.setInt(1, id+1);

```

```

        ps.setString(2, registrationInfo.getFirstName());

```

```

        ps.setString(3, registrationInfo.getLastName());

```

```

        ps.setString(4, registrationInfo.getMobile());

```

```

        ps.setString(5, registrationInfo.getAddress());

```

```

        java.sql.Date sqlDate = new java.sql.Date(new java.util.Date().getTime());

```

```

// java.sql.Date sqlDate = new java.sql.Date(myDate.getTime());

```

```

        ps.setDate(6, sqlDate);

```

```

        ps.setString(7, registrationInfo.getEmail());

```

```

        ps.setString(8, registrationInfo.getGender());

```

```

        ps.setString(9, registrationInfo.getDob());

```

```

        ps.setString(10, registrationInfo.getCountry());

```

```

        ps.setString(11, "E");

```

```

        ps.setString(12, registrationInfo.getPassword());

```

```

        ps.setString(13, registrationInfo.isEmployee());

```

```

        ps.setString(14, registrationInfo.getEmpNumber());

```

```

        i=ps.executeUpdate();

```

```

        if(i>0)

```

```

            System.out.print("Employee is successfully registered...");

```



```

    }
    catch(Exception e)
    {
        System.out.print("Exception occurred"+e.getMessage());
        e.printStackTrace();
    }
    finally
    {

        try {
            //con.close();
            ps.close();
            rs.close();
            st.close();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    return i;
}

public int issueBook(Books book)
{
    int i=0;
    Connection con = null;
    PreparedStatement ps = null;
    // Statement st = null;
    ResultSet rs = null;

```

```

        try
        {

            // Initialize the database

            con = DatabaseConnection.getConnection();

            ps = con.prepareStatement("insert into
libprjct.books(isbn,book_title,category,rental_price,status,author,publisher) values
(?,?,?,?,?,?,?)");

            ps.setInt(1, book.getIsbn());
            ps.setString(1, book.getBook_title());
            ps.setString(1, book.getCategory());
            ps.setInt(1, book.getRental_price());
            ps.setString(1, "available");
            ps.setString(1, book.getAuthor());
            ps.setString(1, book.getPublisher());

            i=ps.executeUpdate();
            if(i>0)
            System.out.print("You are successfully registered...");

        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        finally
        {

            try {

```

```

        //con.close();
        ps.close();
        rs.close();

    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
return i;
}
}

```

## **5. Testing**

### **5.1 Introduction to Testing**

Software Testing is a process of verifying a computer system/program to decide whether it meets the specified requirements and produces the desired results. As a result, you identify bugs in software product/project

Software testing is an activity to check whether the actual results match the expected results and to ensure that the software system is defect free. It involves execution of a software component or system component to evaluate one or more properties of interest.

Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. It can be either done manually or using automated tools.

## **6.Conclusion**

The project entitled “**Library Management SYSTEM(LMS)s**” was completed on time with total satisfaction after testing with possible sample data. The performance was found to be efficient and error free. This is a user-friendly packaged application which is very easy to access and understand. Anyone with Knowledge of Computers will find it very easy to use this software and perform various operations on it.

In this project, first an attempt has been made to find the need of the system. To fulfil the needs, a detailed study had been conducted to find the various requirements of the system. This particular system has been designed in an attractive manner, so that even a user with minimum knowledge can be able to operate the system easily.

This software combines the best of both the world i.e.; programming language (JAVA), and database (MYSQL) providing easy accessibility and security. It was developed to benefit the organizations and the customers. Finally the system was tested with real data and everything worked successfully. Thus the system has fulfilled all the objectives identified and is able to replace the existing system.

## **References**

### **WEBSITE:**

[www.google.co.in](http://www.google.co.in)

[www.w3schools.com](http://www.w3schools.com)

[www.schoolprojects.com](http://www.schoolprojects.com)

[www.m.etrain.info.com](http://www.m.etrain.info.com)

[www.indiamike.com](http://www.indiamike.com)

<https://www.javatpoint.com/>