

# **E-LIBRARY MANAGEMENT SYSTEM**

Submitted in partial fulfillment of the requirements for the award of  
Bachelor of Engineering Degree in Computer Science and Engineering

By

**SUVEETHA J (Reg. No. 39290107)**

**THAMIZHSELVI K (Reg. No. 39290110)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
SCHOOL OF COMPUTING**

**SATHYABAMA  
INSTITUTE OF SCIENCE AND TECHNOLOGY  
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,  
CHENNAI– 600119, TAMILNADU**

**MAY 2022**



# SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC  
(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI- 600119

[www.sathyabama.ac.in](http://www.sathyabama.ac.in)



---

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

### BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **SUVEETHA J (Reg. No. 39290107)** and **THAMIZHSELVI K (Reg. No. 39290110)** who carried out the project entitled "**E-LIBRARY MANAGEMENT SYSTEM**" under my supervision from November 2021 to April 2022.

Internal Guide Dr. MAHESWARI M.,

Head of the Department Dr. S. VIGNESHWARI, M.E., Ph.D.,

---

Submitted for Viva voce Examination held on \_\_\_\_\_

Internal Examiner

External Examiner

## DECLARATION

I, **SUVEETHA J (Reg. No. 39290107) & THAMIZHSELVI K (Reg. No. 39290110)** hereby declare that the Project Report entitled “**E-LIBRARY MANAGEMENT SYSTEM**” done by me under the guidance of **Dr. MAHESWARI.M** is submitted in partial fulfillment of the requirements for the award of Bachelor of Science degree in Computer Science.

**DATE:**

**PLACE: CHENNAI**

**SIGNATURE OF THE CANDIDATE**

## **ACKNOWLEDGEMENT**

I am pleased to acknowledge my sincere thanks to the Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D.**, Dean, School of Computing and **Dr. S. VIGNESHWARI, M.E., Ph.D.**, and **Dr. L. LAKSHMANAN, M.E., Ph.D.**, Head of the Department, Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. MAHESWARI.M.**, for his valuable guidance, suggestions, and constant encouragement that paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science and Engineering who were helpful in many ways For the Completion of the project.

## **ABSTRACT**

Book Store Management System is the web application to automate all kinds of operations in the book shop. The purpose of this software is to manage the books in the book store. This project focuses attention on designing efficient and reliable software which controls the transactions of a bookshop. In the real world, it tends to associate with automated systems as they provide many benefits than doing the same thing manually. As above mentioned, here we have introduced a system which can be used to maintain a bookshop. When we are concerning the manual process of a bookshop, the major problem is the waste of time. A customer has to waste his/her valuable time when he needs to buy a book as all the events such as searching, purchasing, E-links are done by members of the Web application.

## TABLE OF CONTENTS

| Chapter No. | TITLE                                                             | Page No.   |
|-------------|-------------------------------------------------------------------|------------|
|             | <b>ABSTRACT</b>                                                   | <b>v</b>   |
|             | <b>LIST OF FIGURES</b>                                            | <b>vii</b> |
| 1           | <b>INTRODUCTION</b>                                               | <b>1</b>   |
|             | 1.1 OVERVIEW OF THE PROJECT                                       | 1          |
| 2           | <b>LITERATURE SURVEY</b>                                          | <b>2</b>   |
| 3           | <b>AIM AND SCOPE OF PRESENT INVESTIGATION</b>                     | <b>5</b>   |
|             | 3.1 PROBLEM STATEMENT                                             | 5          |
|             | 3.2 REQUIREMENT ANALYSIS                                          | 6          |
| 4           | <b>EXPERIMENTAL OR MATERIALS AND METHODS:<br/>ALGORITHMS USED</b> | <b>26</b>  |
|             | 4.1 INPUT DESIGN                                                  | 26         |
|             | 4.2 OUTPUT DESIGN                                                 | 27         |
|             | 4.3 RESOURCE REQUIREMENT                                          | 28         |
|             | 4.4 DESIGN PROCESS                                                | 28         |
|             | 4.5 SYSTEM ARCHITECTURE                                           | 29         |
|             | 4.6 MODULE DESCRIPTION                                            | 29         |
|             | 4.7 UML DIAGRAMS                                                  | 31         |
|             | 4.8 SYSTEM TESTING                                                | 31         |
|             | 4.9 UNIT TESTING                                                  | 33         |
|             | 4.10 INTEGRATION TESTING                                          | 34         |
|             | 4.11 ACCEPTANCE TESTING                                           | 34         |
|             | 4.12 SYSTEM STUDY                                                 | 35         |
| 5           | <b>IMPLEMENTATION AND RESULTS</b>                                 | <b>37</b>  |
| 6           | <b>SUMMARY AND CONCLUSION</b>                                     | <b>46</b>  |
|             | 6.1 CONCLUSION                                                    | 46         |
|             | 6.2 REFERENCES                                                    | 46         |
|             | <b>APPENDIX</b>                                                   | <b>48</b>  |
|             | A. SOURCE CODE                                                    | 48         |

## LIST OF FIGURES

| FIGURE<br>NO | FIGURE NAME               | PAGE<br>NO |
|--------------|---------------------------|------------|
| 3.1          | Java VM                   | 7          |
| 3.2          | Java IDE                  | 10         |
| 3.3          | TCP/IP Stack              | 17         |
| 3.4          | Total Address             | 19         |
| 3.5          | General J2ME Architecture | 22         |
| 4.1          | System Architecture       | 29         |
| 4.2          | Use case Diagram          | 31         |

# **CHAPTER-1**

## **1. INTRODUCTION**

### **1.1 OVERVIEW OF THE PROJECT**

In the world of software development, there is lots of improvement in the area of Architectural design and principles. The philosophies and implementation details are changing as the people guide the development of the application. In this fantastic and yet sometimes complex world of software development there are some tried and true architecture patterns and software development guidelines employed by most architects. Also, your design must have an ability to turn towards innovation instead of lending itself to common practices. Web services are one such area where architects must lean on their creative side and hope that their solutions are still successful. In this report, we will explain an exciting voyage down the road of Web services application. From requirements to use cases, to database design, to component frameworks, to user interfaces, we will cover every aspect of system design required to build an application with collaborative Web services. The reason why we selected the online Bookstore web service is everybody walking down the street has some idea about bookstores. The objective of this project is to develop an e- book store where books can be bought from the comfort of home through the Internet. An online book store is a virtual store on the Internet where customers can browse the catalog and select books of interest. The selected books may be collected in a shopping cart. At checkout time, the items in the shopping cart will be presented as an order. At that time, more information will be needed to complete the transaction. Usually, the customer will be asked to fill or select a billing address, a shipping address, a shipping option, and payment information such as a credit card number. E-mail notification is sent to the customer as soon as the order is placed.

## **CHAPTER-2**

### **2. LITERATURE SURVEY**

**TITLE:** Online Book Store

**AUTHOR:** Prasad NK, Varun Kishore, Omprakash

**ABSTRACT:**

The Internet by far plays a major role in people's life. It has drastically improved the quality of life and the standard of living of so many people. It has widened its branches into many different levels and areas. The e-commerce industry is one such branch that has come into the spotlight in recent years. The online bookstore system has eased the life of so many book lovers by making it easy for them to purchase books online. It is not always feasible to access a traditional bookstore, it is limited by its operation time, availability of a particular book, its location, and most importantly its capacity and the space required to store numerous books. Such drawbacks have led to the evolution of e-commerce industries related to bookstores. Our project is one such simple e-commerce website that houses various books of different categories for a consumer to purchase online.

**TITLE:** A Framework for Discovering Similar Products from Online Bookstore

**AUTHOR:** Mingxi Zhang

**ABSTRACT:**

Online bookstores have attracted millions of people and helped provide them with hopeful books. Similarity search over online book stores mainly focuses on finding the top-K most similar products for a given query. In this paper, we discuss how to find similar products for a given query product, and propose a framework for finding similar products from online bookstores. We firstly build the co-purchasing network by using the relationships between different types of products and then compute the similarity between products using Sim Rank. Finally, we give some experimental results by implementing this method on the Amazon dataset, which demonstrates that the proposed method can find the underlying results over the real dataset.

TITLE: Opportunities and Challenges of Building a Books-as-a-Service Platform

AUTHOR: Justo N. Hidalgo

ABSTRACT:

The concept of reading is by no means static. Technologies have undoubtedly affected the ways that we consume and produce written content, the tools that we use, our reading style, and even how we organize our thoughts as we read. Digital technologies seem to have finally taken the book to its next stage. The vision of building a product from which different stakeholders can build book-related services and applications more quickly and efficiently is not new, as academia and other publishing experts have already done some thinking about it. However, no commercial tool still exists that enables access to general book-related content. This article will introduce the concept of a “Book as a Service” platform, a software product that enables access to digital books and related functions such as text mining, reading analytics, and metadata so that other companies can create innovative services and products on top of it. The opportunities and challenges of this type of platform from technical, legal, and business standpoints will also be discussed, as well as its implications when redefining the possibilities of the written word.

TITLE: Online bookstore - A new trend in textbook sales management for services marketing

AUTHOR: Prathamesh Muzumdar

ABSTRACT:

The research study explores the phenomenon of an increase in the use of online mediums for textbook sales. The study uses the basics of commerce transactions, incorporating them into a model to explain their effects on different mediums of textbook sale. The study uses a theoretical approach in explaining the influence of advantages and disadvantages of each medium on transactions occurring through it. The study designs two different models each representing a single medium. The method of flowchart analysis is used to describe the various processes occurring in

each medium and transactions are incorporated at the point of sales in both models. Overall the study concludes by explaining the theory behind the growth of online mediums when compared to other mediums.

TITLE: Inventory Management Book Store System Using WID (IMBS)

AUTHOR: Norashikin Binti Mohamed Kamil

ABSTRACT:

Inventory Management Book Store System Using RFID (IMBS) is a system in the category of desktop tools applications. It focuses on managing book stores with the use of WID. WID is a device that scans the quantity and detailed information about a variety of books at the book stores. The area of study is at Bestari Book Store. The book store provides a variety of books and magazines for customers. This system will be developed using Microsoft Visual Basic.Net, Microsoft SQL Server to build a database, Microsoft Project, Microsoft Word, Adobe Photoshop as the software. The hardware that be used to build this system is RFID Reader, RFID Tag, and personal computer. Another device that we used is Window XP as an operating system. During the development of IMBS, the methodology being used is the Prototyping model. Each process during the development process is followed by each phases in Prototyping model. The system consists four modules includes Book Stock Managing Module, Order Module, Book stock Information Searching Module and Administration Management Module. The groups of user that had been identified to use the system are Administrator and employee at Bestari Book Store. IMBS also will develop to help admin and employee manage data of book store with quickly and efficient. Besides that, it can save time and cost.

## **CHAPTER 3**

### **3. AIM AND SCOPE OF PRESENT INVESTIGATION**

The project aims and objectives that will be achieved after completion of this project are discussed in this subchapter.

The aims and objectives are as follows:

- Online book reading.
- A search column to search availability of books.
- Facility to download required book.
- Video tutorial for students.
- An Admin login page where admin can add books, videos or page sources.
- Open link for Learning Websites

#### **3.1 PROBLEM STATEMENT**

An online bookstore software projects that acts as a central database containing various books in stock along with their title, author and cost. This project is a website that acts as a central book store. The MySQL database stores various book related details. A user visiting the website can see a wide range of books arranged in respective categories. The user may select desired book and view its price. The user may even search for specific books on the website. Once the user selects a book, he then has to fill in a form and the book is booked for the user.

#### **DIS-ADVANTAGE:**

1. search in product is not so efficient in existing System
2. un-available products are not notified

## **3.2 REQUIREMENT ANALYSIS:**

### **Software Environment**

#### **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 buzzwords:
  - Simple
  - Architecture neutral
  - Object oriented
  - Portable
  - Distributed
  - High performance
  - Interpreted
  - Multithreaded
  - Robust
  - Dynamic
  - Secure
- With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte

codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works. You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make “write once, run anywhere” possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.

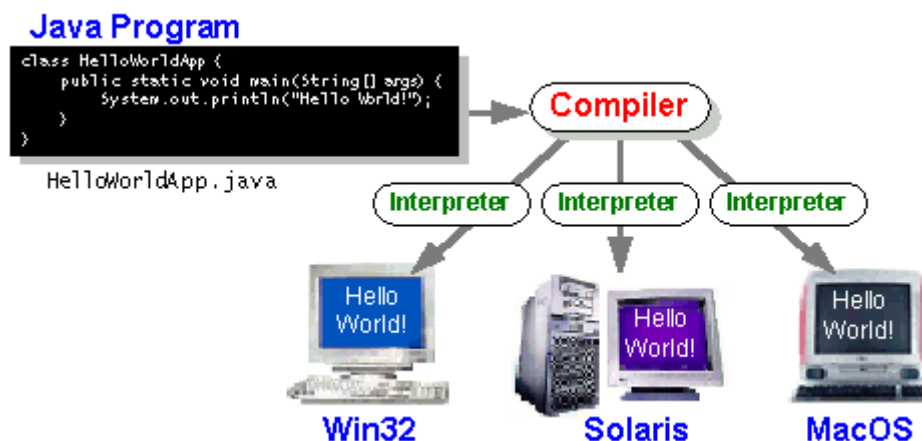


Fig No: 3.1 Java VM

## The Java Platform

- A *platform* is the hardware or software environment in which program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and 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* (Java VM)
- The *Java Application Programming Interface* (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as *packages*. The next section, *What Can Java Technology Do?* Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

## **What Can Java Technology Do?**

The most common types of programs written in the Java programming language are *applets* and *applications*. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java

programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a **server** serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a **servlet**. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server. How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

- **The essentials:** Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- **Applets:** The set of conventions used by applets.
- **Networking:** URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
- **Internationalization:** Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security:** Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software components:** Known as JavaBeans™, can plug into existing component architectures.

- **Object serialization:** Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC™):** Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.

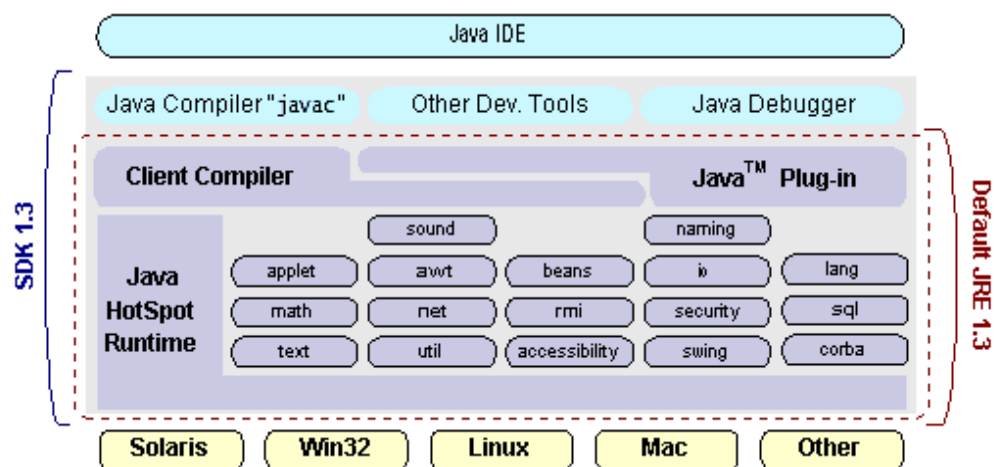


Fig No:3.2 Java IDE

### How Will Java Technology Change My Life?

We can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

- **Get started quickly:** Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.
- **Write less code:** Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java

programming language can be four times smaller than the same program in C++.

- **Write better code:** The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.
- **Develop programs more quickly:** Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
- **Avoid platform dependencies with 100% Pure Java:** You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure Java™ Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
- **Write once, run anywhere:** Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
- **Distribute software more easily:** You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded "on the fly," without recompiling the entire program.

## ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a de facto standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers

have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change. Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even

Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

## **JDBC**

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or drivers. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC's framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

## **JDBC Goals**

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

### **1. SQL Level API**

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

### **2. SQL Conformance**

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query

statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

### **3. JDBC must be implemental on top of common database interfaces**

The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

#### **4. Provide a Java interface that is consistent with the rest of the Java system**

Because of Java's acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

#### **5. Keep it simple**

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

#### **6. Use strong, static typing wherever possible**

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

#### **7. Keep the common cases simple**

Because more often than not, the usual SQL calls used by the programmer are simple SELECT's, INSERT's, DELETE's and UPDATE's, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible. Finally, we decided to proceed the implementation using Java Networking. And for dynamically updating the cache table we go for MS Access database. Java has two things: a programming language and a platform.

Java is a high-level programming language that is all of the following

1. Simple
2. Architecture-neutral
3. Object-oriented
4. Portable
5. Distributed

6. High-performance
7. Interpreted
8. Multithreaded
9. Robust
10. Dynamic
11. Secure

Java is also unusual in that each Java program is both compiled and interpreted. With a compiler you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer.

Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.

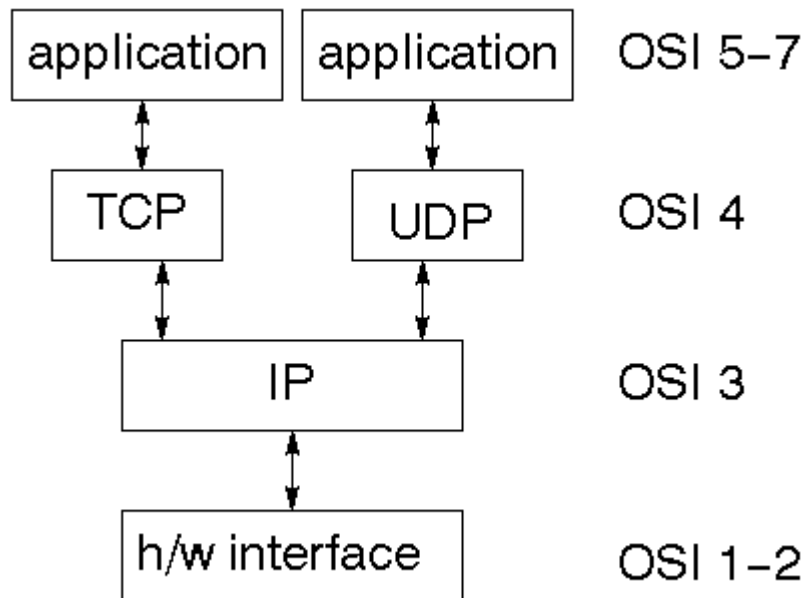
You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make "write once, run anywhere" possible. You can compile your Java program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. For example, the same Java program can run on Windows NT, Solaris, and Macintosh.

## **Networking**

### **TCP/IP stack:**

The TCP/IP stack is shorter than the OSI one.



**Fig No: 3.3 TCP/IP stack**

TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

### **IP datagram's**

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an

Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

## **UDP**

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

## **TCP**

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

## **Internet addresses**

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

### **Network address**

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

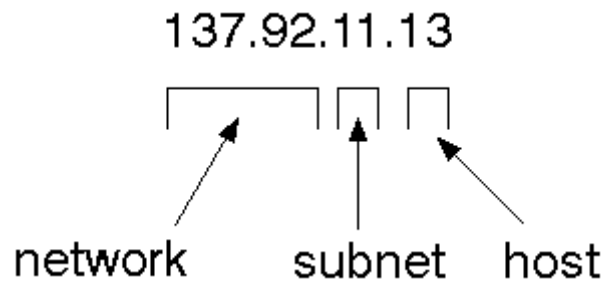
### **Subnet address**

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

### **Host address**

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

## Total address



**Fig No: 3.4 Total Address**

The 32 bit address is usually written as 4 integers separated by dots.

## Port addresses

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

## Sockets

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call `socket`. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with Read File and Write File functions.

```
#include <sys/types.h>
```

```
#include <sys/socket.h>
```

```
int socket(int family, int type, int protocol);
```

Here "family" will be `AF_INET` for IP communications, protocol will be zero, and type will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

## **JFree Chart**

FreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFreeChart's extensive feature set includes:

A consistent and well-documented API, supporting a wide range of chart types;

A flexible design that is easy to extend, and targets both server-side and client-side applications;

Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

JFreeChart is "open source" or, more specifically, [free software](#). It is distributed under the terms of the [GNU Lesser General Public Licence](#) (LGPL), which permits use in proprietary applications.

### **1. Map Visualizations**

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:

Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas);

Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XYPlot class in JFreeChart;

Testing, documenting, testing some more, documenting some more.

### **2. Time Series Chart Interactivity**

Implement a new (to JFreeChart) feature for interactive time series charts --  
- to display a separate control that shows a small version of ALL the time series

data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

### **3. Dashboards**

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFreeChart chart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

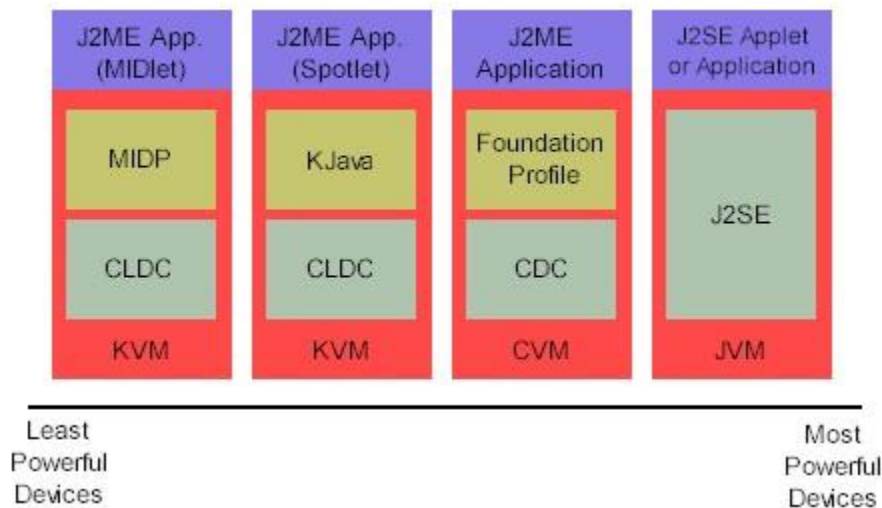
### **4. Property Editors**

The property editor mechanism in JFreeChart only handles a small subset of the properties that can be set for charts. Extend (or reimplement) this mechanism to provide greater end-user control over the appearance of the charts.

### **J2ME (Java 2 Micro edition):-**

Sun Microsystems defines J2ME as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems." Announced in June 1999 at the JavaOne Developer Conference, J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing mobile wireless devices to share applications. With J2ME, Sun has adapted the Java platform for consumer products that incorporate or are based on small computing devices.

## 1. General J2ME architecture



**Fig No:3.5 General J2ME Architecture**

J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes. The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. We'll discuss configurations in detail in the The profile defines the application; specifically, it adds domain-specific classes to the J2ME configuration to define certain uses for devices. We'll cover profiles in depth in the The following graphic depicts the relationship between the different virtual machines, configurations, and profiles. It also draws a parallel with the J2SE API and its Java virtual machine. While the J2SE virtual machine is generally referred to as a JVM, the J2ME virtual machines, KVM and CVM, are subsets of JVM. Both KVM and CVM can be thought of as a kind of Java virtual machine -- it's just that they are shrunken versions of the J2SE JVM and are specific to J2ME.

## 2. Developing J2ME applications

Introduction In this section, we will go over some considerations you need to keep in mind when developing applications for smaller devices. We'll take a look at the way the compiler is invoked when using J2SE to compile J2ME applications. Finally, we'll explore packaging and deployment and the role preverification plays in this process.

### **3.Design considerations for small devices**

Developing applications for small devices requires you to keep certain strategies in mind during the design phase. It is best to strategically design an application for a small device before you begin coding. Correcting the code because you failed to consider all of the "gotchas" before developing the application can be a painful process. Here are some design strategies to consider:

- \* Keep it simple. Remove unnecessary features, possibly making those features a separate, secondary application.

Smaller is better. This consideration should be a "no brainer" for all developers. Smaller applications use less memory on the device and require shorter installation times. Consider packaging your Java applications as compressed Java Archive (jar) files.

- \* Minimize run-time memory use. To minimize the amount of memory used at run time, use scalar types in place of object types. Also, do not depend on the garbage collector. You should manage the memory efficiently yourself by setting object references to null when you are finished with them. Another way to reduce run-time memory is to use lazy instantiation, only allocating objects on an as-needed basis. Other ways of reducing overall and peak memory use on small devices are to release resources quickly, reuse objects, and avoid exceptions.

### **4.Configurations overview**

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. Currently, two configurations exist for J2ME, though others may be defined in the future:

\* **Connected Limited Device Configuration (CLDC)** is used specifically with the KVM for 16-bit or 32-bit devices with limited amounts of memory. This is the configuration (and the virtual machine) used for developing small J2ME applications. Its size limitations make CLDC more interesting and challenging (from a development point of view) than CDC. CLDC is also the configuration that we will use for developing our drawing tool application. An example of a small wireless device running small applications is a Palm hand-held computer.

\* **Connected Device Configuration (CDC)** is used with the C virtual machine (CVM) and is used for 32-bit architectures requiring more than 2 MB of memory. An example of such a device is a Net TV box.

## 5.J2ME profiles

### What is a J2ME profile?

As we mentioned earlier in this tutorial, a profile defines the type of device supported. The Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J2ME and are built upon CLDC: KJava and MIDP. Both KJava and MIDP are associated with CLDC and smaller devices. Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations.

A skeleton profile upon which you can create your own profile, the Foundation Profile, is available for CDC.

### Profile 1: KJava

KJava is Sun's proprietary profile and contains the KJava API. The KJava profile is built on top of the CLDC configuration. The KJava virtual machine, KVM, accepts the same byte codes and class file format as the classic J2SE virtual machine.

KJava contains a Sun-specific API that runs on the Palm OS. The KJava API has a great deal in common with the J2SE Abstract Windowing Toolkit (AWT). However, because it is not a standard J2ME package, its main package is `com.sun.kjava`. We'll learn more about the KJava API later in this tutorial when we develop some sample applications.

## **Profile 2: MIDP**

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like KJava, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed dynamically on end user devices. MIDP is a common, industry-standard profile for mobile devices that is not dependent on a specific vendor. It is a complete and supported foundation for mobile application

development. MIDP contains the following packages, the first three of which are core CLDC packages, plus three MIDP-specific packages.

- \* `java.lang`
- \* `java.io`
- \* `java.util`
- \* `javax.microedition.io`
- \* `javax.microedition.lcdui`
- \* `javax.microedition.midlet`
- \* `javax.microedition.rms`

## **CHAPTER 4**

### **4 EXPERIMENTAL OR MATERIALS AND METHODS: ALGORITHMS USED**

#### **4.1 INPUT DESIGN:**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

#### **OBJECTIVES**

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in a maze of instant. Thus, the objective of input design is to create an input layout that is easy to follow

## **4.2 OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displayed for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- ❖ Convey information about past activities, current status or projections of the
- ❖ Future.
- ❖ Signal important events, opportunities, problems, or warnings.
- ❖ Trigger an action.
- ❖ Confirm an action.

### **4.3 RESOURCE REQUIREMENT**

#### **SYSTEM REQUIREMENTS:**

#### **HARDWARE REQUIREMENTS:**

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 4 GB

#### **SOFTWARE REQUIREMENTS:**

- Operating system : Windows 7/10.
- Coding Language : JAVA
- Data Base : MYSQL
- IDE : NetBeans 8.0.1

### **4.4 DESIGN PROCESS**

#### **PROPOSED SYSTEM**

Aiming at this goal, this project is to develop an e- book store where books can be bought from the comfort of home through the Internet. An online book store is a virtual store on the Internet where student can browse the catalog and select books of interest. The proposed method can find the underlying results over real dataset.

For example, Student Search/feedback to the book in e-book .

## 4.5 SYSTEM ARCHITECTURE

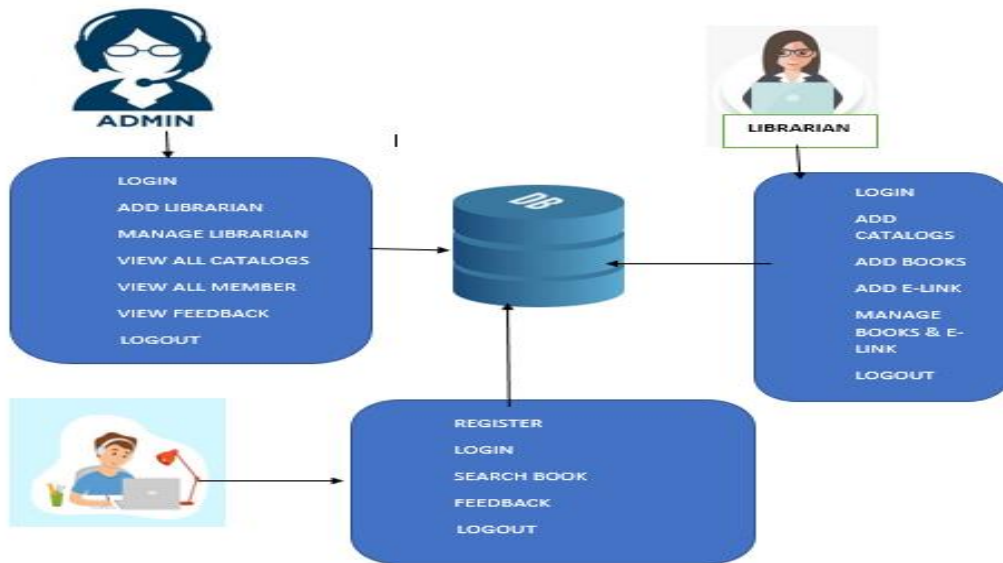


Fig No:4.1 System Architecture

## 4.6 MODULE DESCRIPTION:

Modules:

- 1.Admin
- 2.Librarian
- 3.Student

### ADMIN

1. LOGIN
2. ADD LIBRARIAN
3. MANAGE ALL LIBRARIAN
4. VIEW ALL CATALOGS
5. VIEW ALL MEMBERS
6. VIEW ALL FEEDBACKS GIVEN BY STUDENT

7. LOGOUT

## **LIBRARIAN**

1. LOGIN
2. ADD CATALOGS
3. ADD BOOKS
4. ADD E-LINK
5. MANAGE ALL BOOKS & E-LINK
6. LOGOUT

## **STUDENT**

1. REGISTER
2. LOGIN
3. SEARCH BOOK
4. FEEDBACK GIVING TO THE SEARCHED BOOK
5. LOGOUT

## 4.7 UML DIAGRAMS

Use case diagram:

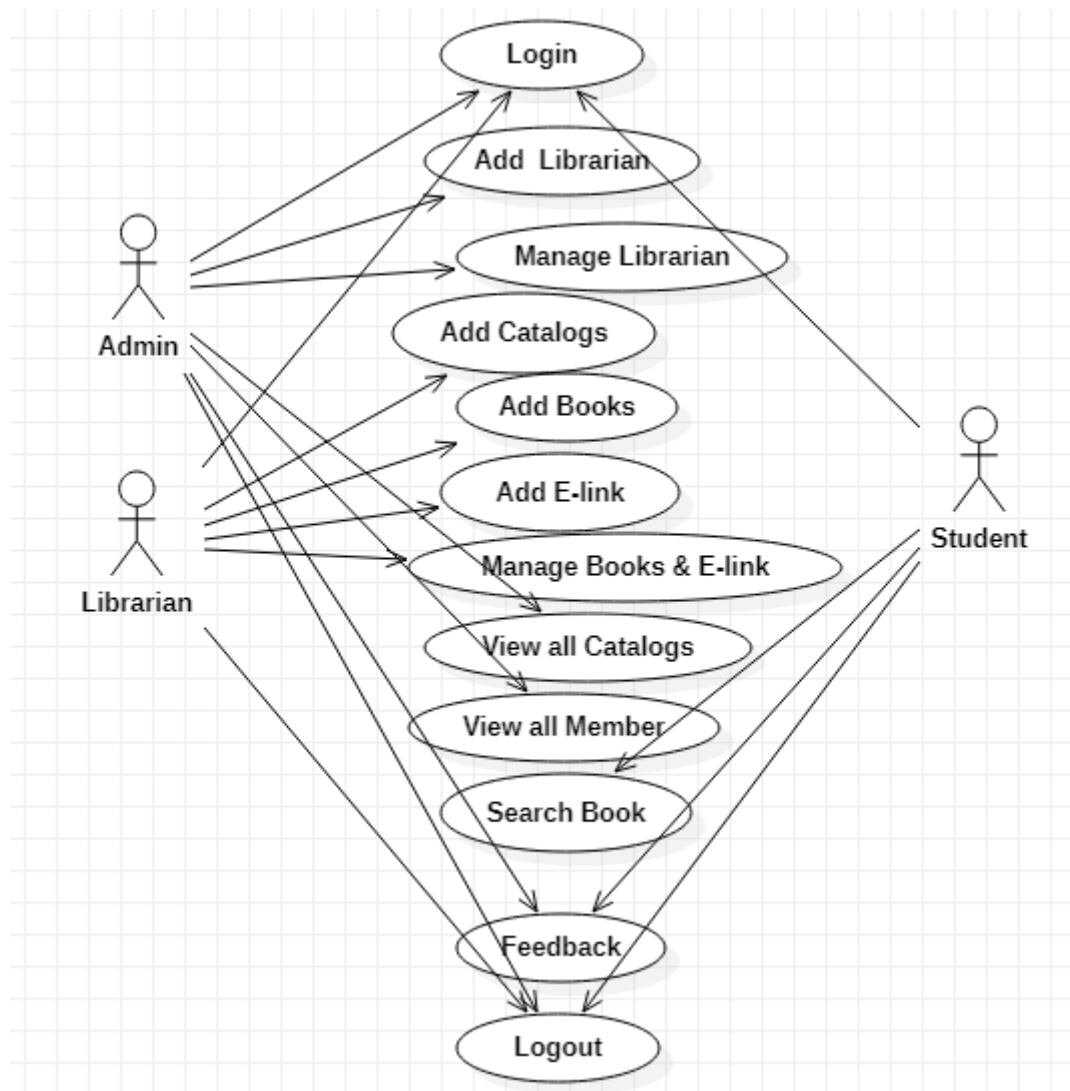


Fig No: 4.2 Use case diagram

## 4.8 SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

## **TYPES OF TESTS**

### **Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration.

Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

### **Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

### **Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

### **System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

### **White Box Testing**

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

### **Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

### **4.9 Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

## **Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

## **Test objectives**

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

## **Features to be tested**

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

### **4.10 Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

### **4.11 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

## **4.12 SYSTEM STUDY**

### **FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ◆ **ECONOMICAL FEASIBILITY**
- ◆ **TECHNICAL FEASIBILITY**
- ◆ **SOCIAL FEASIBILITY**
- ◆ **ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

#### **◆ TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

#### **◆ SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system

efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

#### **APPLICATION AND FUTURE ENHANCEMENT:**

A quality online bookstore will have a good product selection, an easy-to-use -yet comprehensive- website, a variety of shipping options, a number of payment options, excellent customer support and a strong return policy. A good product selection is critical to any online bookstore.

#### **Future Scope of Bookstore Management System**

In the future, we will try to cover all the states and cities till now we have covered it at a very small scale. We will also introduce new features such as Author videos and reviews so that users can get more satisfaction about his deal.

## CHAPTER 5

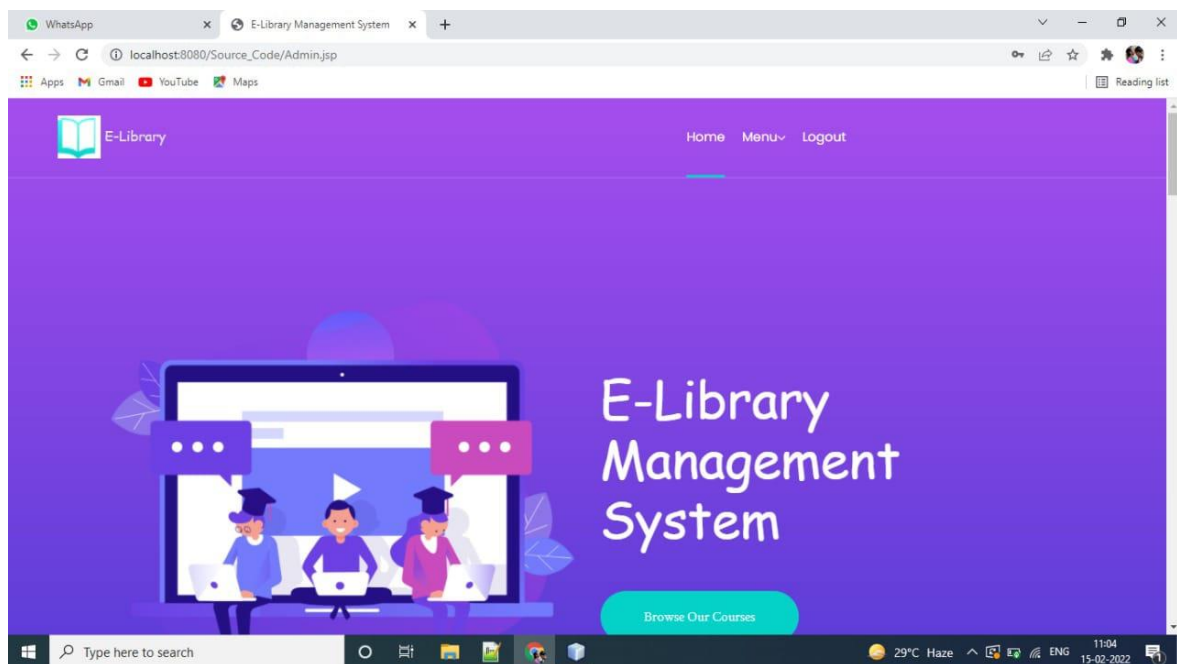
### IMPLEMENTATION AND RESULTS:

This chapter covers the system tests and results. This are demonstrated using a number of screen shots. Library Book Management System aimed at accomplishing a number of tasks. The system met its accomplishments and the following section looks at this tasks and how they are accomplished.

LBMS has been designed and developed with high security standards. A number of security levels have been designed to limit privileges according to a user type. Users can access a page depending on the security levels given to them. These levels determine the various rights and operations a user can carry out on the system. The levels include; the librarian access levels and the members access levels.

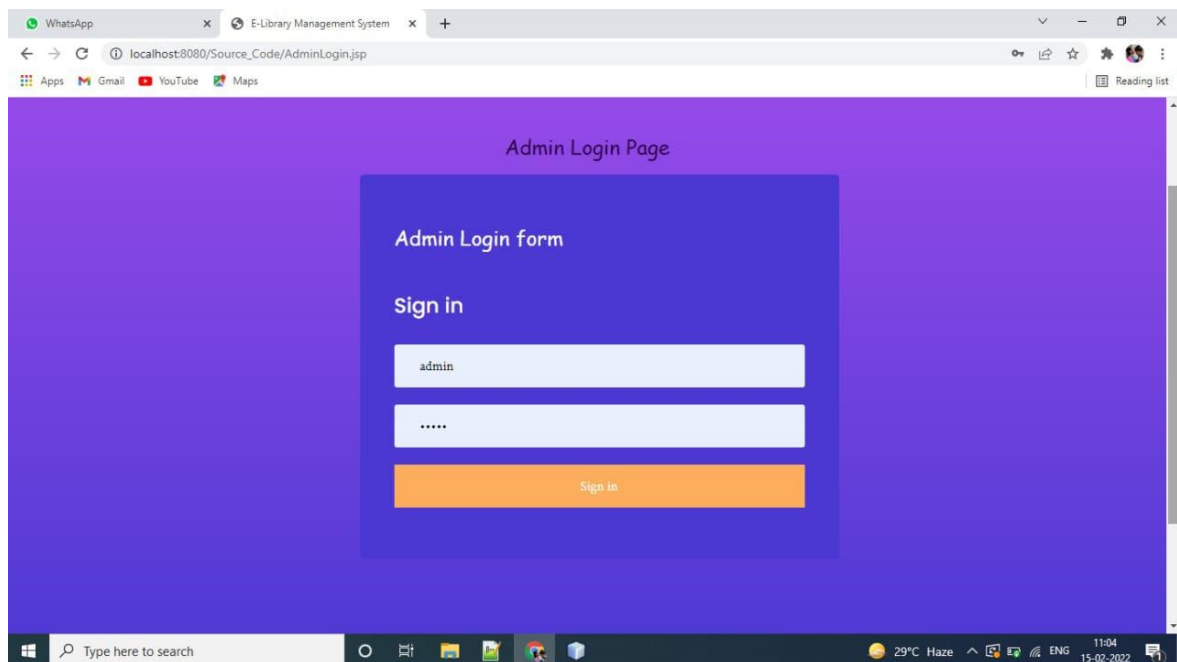
Thus shows the Library Book Management System's homepage. All the users login to the system from this page. A user clicks on the sign in button where the login form appears enabling them to carry on with this process.

### HOME PAGE



## ADMIN MODULE

### LOGIN



The screenshot shows a web browser window with the URL `localhost:8080/Source_Code/AdminLogin.jsp`. The page has a purple gradient background and is titled "Admin Login Page". In the center, there is a dark blue box containing the "Admin Login form". The form includes a "Sign in" heading, a text input field with the value "admin", a password input field with masked characters ".....", and an orange "Sign in" button. The browser's taskbar at the bottom shows the Windows logo, a search bar, and system icons for temperature (29°C Haze), language (ENG), and date/time (11:04 15-02-2022).

Admin Login Page

Admin Login form

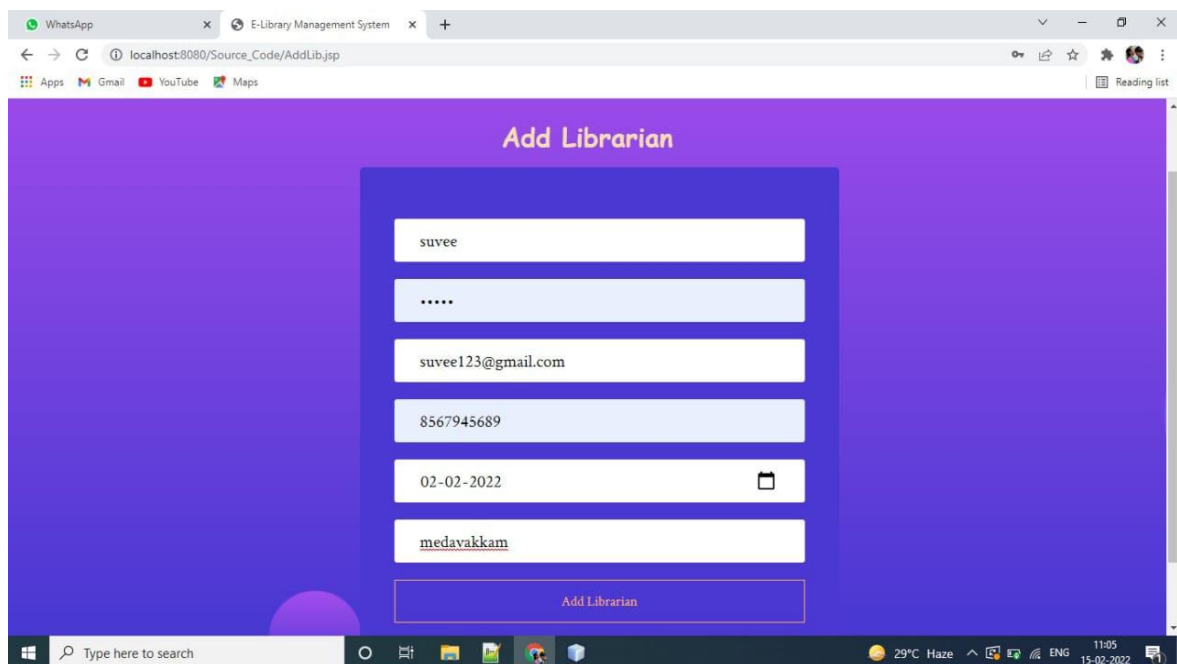
Sign in

admin

.....

Sign in

### ADD LIBRARIAN



The screenshot shows a web browser window with the URL `localhost:8080/Source_Code/AddLib.jsp`. The page has a purple gradient background and is titled "Add Librarian". In the center, there is a dark blue box containing a form with several input fields: a text field with "suvée", a password field with masked characters ".....", an email field with "suvée123@gmail.com", a phone number field with "8567945689", a date field with "02-02-2022" and a calendar icon, and a text field with "medavakkam". At the bottom of the box is a blue "Add Librarian" button. The browser's taskbar at the bottom shows the Windows logo, a search bar, and system icons for temperature (29°C Haze), language (ENG), and date/time (11:05 15-02-2022).

Add Librarian

suvée

.....

suvée123@gmail.com

8567945689

02-02-2022

medavakkam

Add Librarian

## MANAGE ALL LIBRARIAN

E-Library Home Menu Logout

### View & Manage Librarian

| ID | Name  | Password | Mail               | DOB        | Phone No   | Address    | Action |
|----|-------|----------|--------------------|------------|------------|------------|--------|
| 9  | suvee | suvee    | suve@gmail.com     | 2022-01-04 | 8567945942 | chennai    |        |
| 10 | suvee | admin    | suvee123@gmail.com | 2022-02-02 | 8567945689 | medavakkam |        |

Type here to search 29°C Haze ENG 11:05 15-02-2022

## VIEW ALL CATALOG

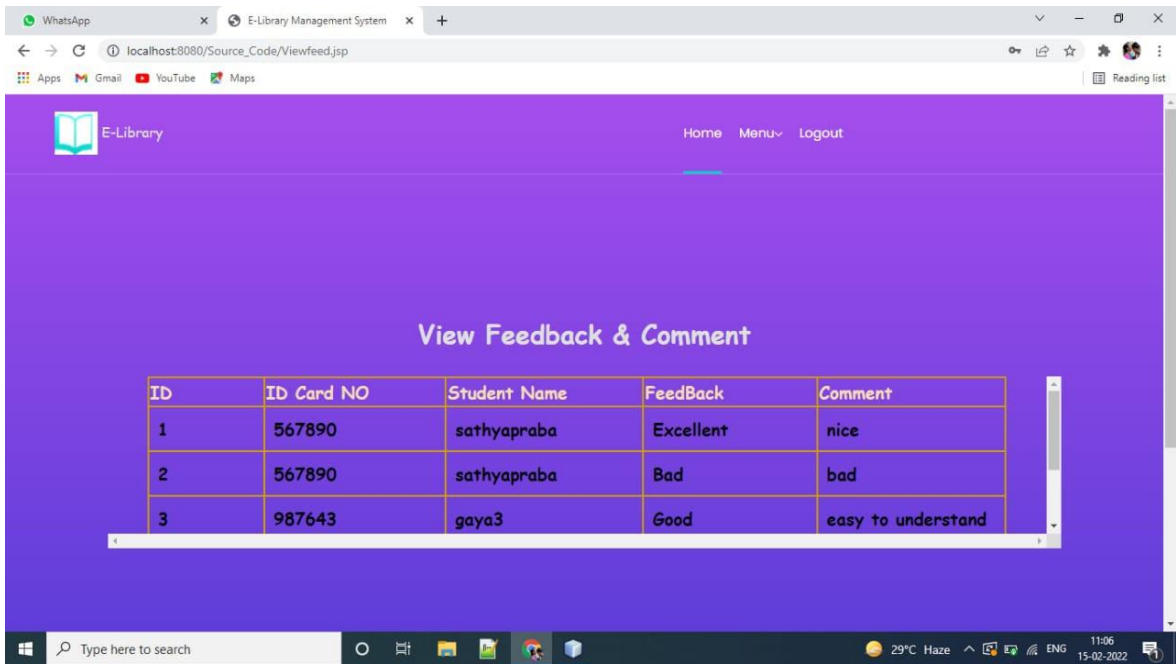
E-Library Home Menu Logout

### View all Catalog details

| ID | Lib_Id | Lib_Name | Book Id | Book Name | B_Description                      | Author        | Pages    | Publish Date |
|----|--------|----------|---------|-----------|------------------------------------|---------------|----------|--------------|
| 1  | null   | null     | 001     | html      | good                               | balaguruswamy | 678      | 2020-05-14   |
| 2  |        |          | 002     | kid story | kid are like very for that stroies | sathya        | 50 to 60 | 2020-05-01   |

Type here to search 29°C Haze ENG 11:06 15-02-2022

## VIEW ALL FEEBACKS GIVEN BY STUDENTS

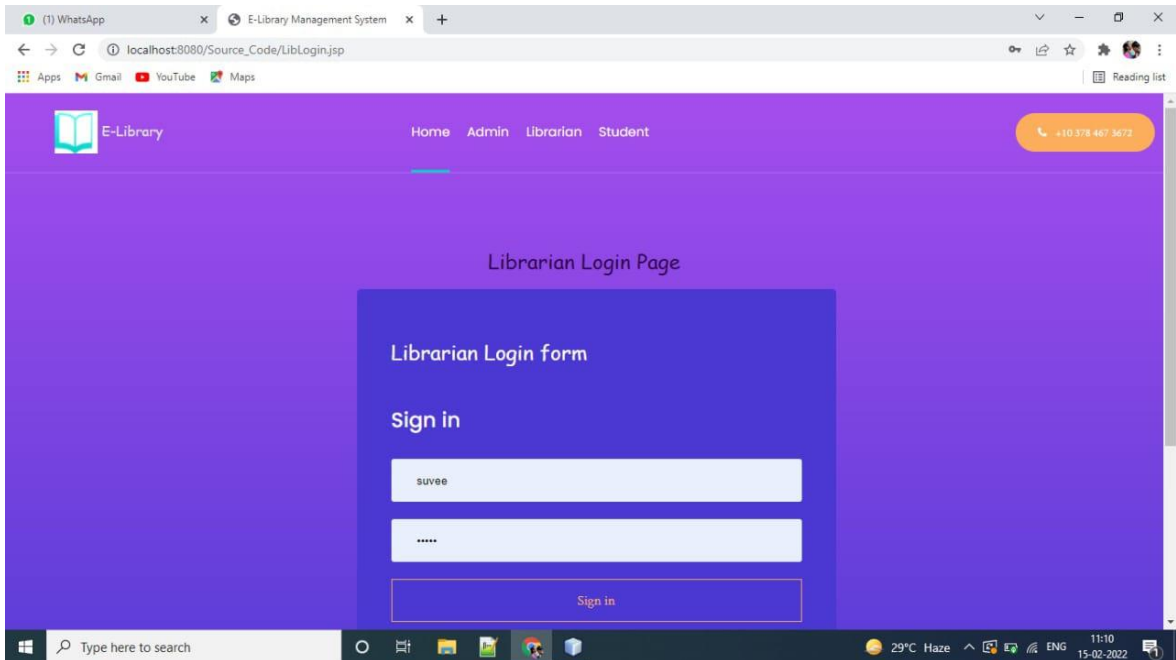


The screenshot shows a web browser window with the URL `localhost8080/Source_Code/Viewfeed.jsp`. The page has a purple header with the 'E-Library' logo and navigation links: Home, Menu, and Logout. The main content area is titled 'View Feedback & Comment' and contains a table with the following data:

| ID | ID Card NO | Student Name | FeedBack  | Comment            |
|----|------------|--------------|-----------|--------------------|
| 1  | 567890     | sathyapraba  | Excellent | nice               |
| 2  | 567890     | sathyapraba  | Bad       | bad                |
| 3  | 987643     | gaya3        | Good      | easy to understand |

## LIBRIRAN MODULE

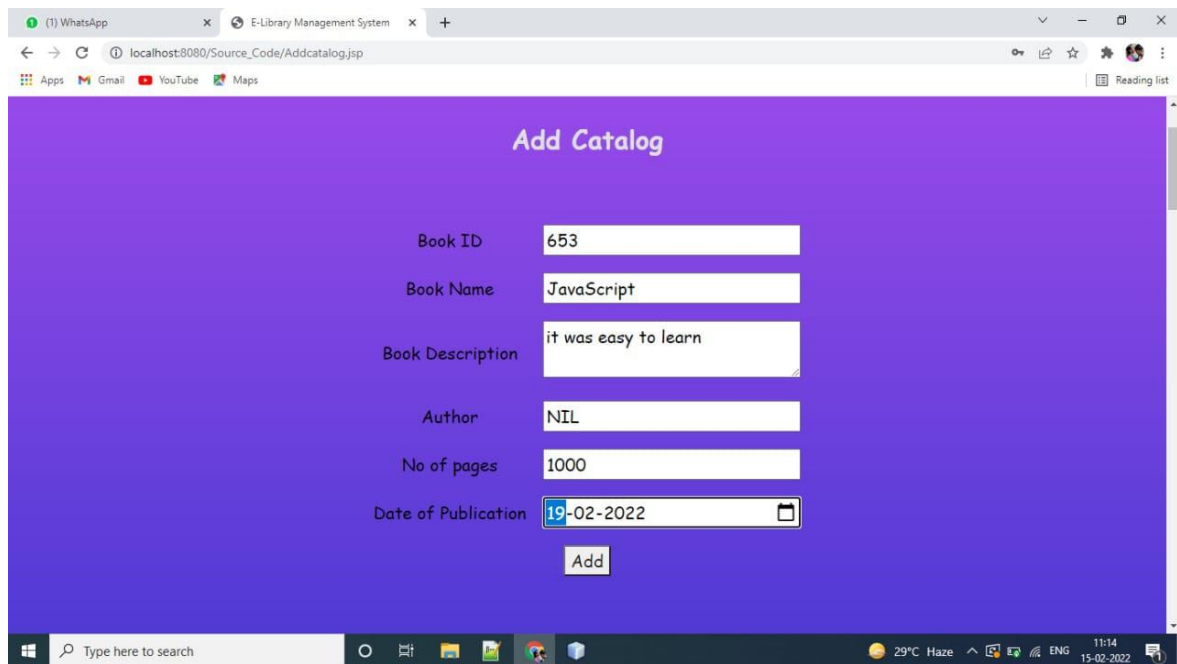
### LOGIN PAGE



The screenshot shows a web browser window with the URL `localhost8080/Source_Code/LibLogin.jsp`. The page has a purple header with the 'E-Library' logo and navigation links: Home, Admin, Librarian, and Student. A contact number `+91 378 447 3672` is displayed in the top right corner. The main content area is titled 'Librarian Login Page' and contains a 'Librarian Login form' with the following fields:

- Sign in
- Username field (containing 'suvie')
- Password field (containing '.....')
- Sign in button

## ADD CATALOG



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/Source\_Code/Addcatalog.jsp'. The page has a purple gradient background and is titled 'Add Catalog'. It contains a form with the following fields: 'Book ID' (653), 'Book Name' (JavaScript), 'Book Description' (it was easy to learn), 'Author' (NIL), 'No of pages' (1000), and 'Date of Publication' (19-02-2022). A calendar icon is next to the date field. An 'Add' button is located below the form. The Windows taskbar at the bottom shows the search bar, task view button, and several application icons. The system tray on the right indicates a temperature of 29°C, haze, and the date/time as 11:14 on 15-02-2022.

**Add Catalog**

Book ID: 653

Book Name: JavaScript

Book Description: it was easy to learn

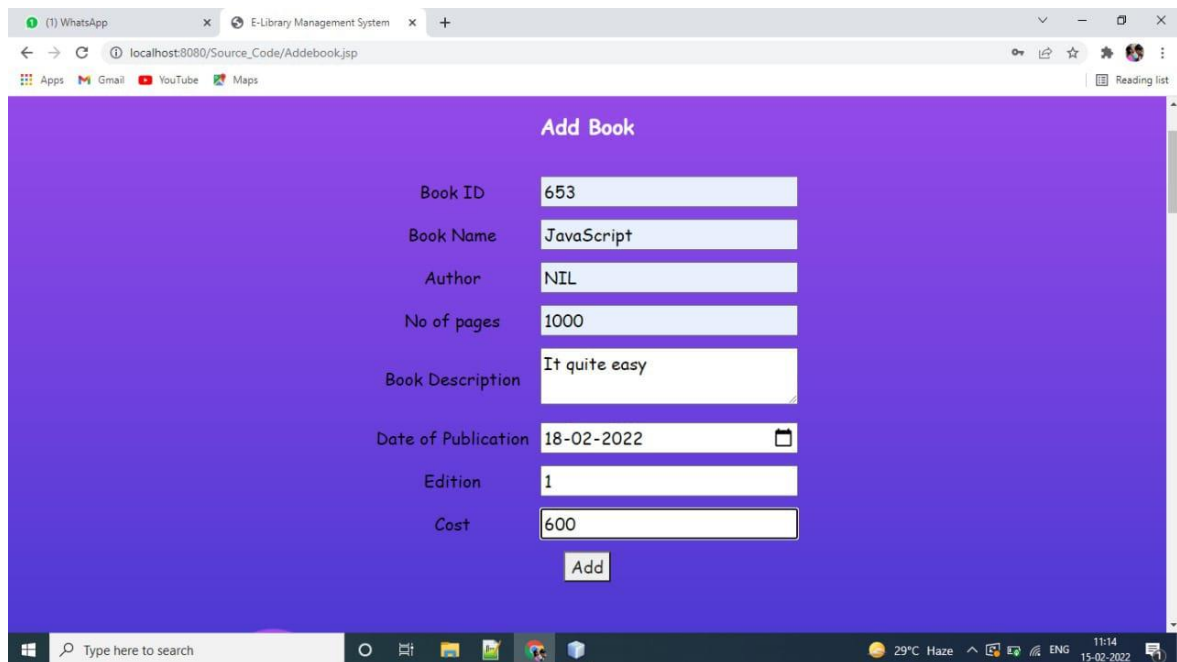
Author: NIL

No of pages: 1000

Date of Publication: 19-02-2022

Add

## ADD BOOKS



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/Source\_Code/Addebook.jsp'. The page has a purple gradient background and is titled 'Add Book'. It contains a form with the following fields: 'Book ID' (653), 'Book Name' (JavaScript), 'Author' (NIL), 'No of pages' (1000), 'Book Description' (It quite easy), 'Date of Publication' (18-02-2022), 'Edition' (1), and 'Cost' (600). A calendar icon is next to the date field. An 'Add' button is located below the form. The Windows taskbar at the bottom shows the search bar, task view button, and several application icons. The system tray on the right indicates a temperature of 29°C, haze, and the date/time as 11:14 on 15-02-2022.

**Add Book**

Book ID: 653

Book Name: JavaScript

Author: NIL

No of pages: 1000

Book Description: It quite easy

Date of Publication: 18-02-2022

Edition: 1

Cost: 600

Add

## ADD BOOK E-LINK

The screenshot shows a web browser window with the URL `localhost:8080/Source_Code/Addelink.jsp`. The page has a purple header with the 'E-Library' logo and navigation links for 'Home', 'Menu', and 'Logout'. The main content area is titled 'Add Book E-Link' and contains a form with the following fields:

- Book ID:
- Book Name:
- Author:
- Book E-Link:

Below the fields is an 'Add' button. The Windows taskbar at the bottom shows the search bar, task icons, and system tray with a temperature of 29°C and the date 15-02-2022.

## MANAGE ALL BOOKS & E-LINKS

The screenshot shows a web browser window with the URL `localhost:8080/Source_Code/Mancatalog.jsp`. The page has a purple header with the 'E-Library' logo and navigation links for 'Home', 'Menu', and 'Logout'. The main content area is titled 'View & Manage Book details' and contains a table with the following data:

| ID | Book Id | Book Name  | B_Description | Author | Pages | Publish Date | Cost | Edition | Action                                                                                                                                                                      |
|----|---------|------------|---------------|--------|-------|--------------|------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6  | 653     | JavaScript | It quite easy | NIL    | 1000  | 2022-02-18   | 600  | 1       |   |

The Windows taskbar at the bottom shows the search bar, task icons, and system tray with a temperature of 29°C and the date 15-02-2022.

## STUDENT MODULE

The screenshot shows a web browser window with the URL `localhost:8080/Source_Code/stuAction.jsp`. The page has a purple header with the "E-Library" logo and navigation links: Home, Admin, Librarian, and Student. A contact number `+10 578 447 5673` is displayed in an orange button. The main content area is titled "Student Login Page" and contains a "Student Login form". The form has a "Sign in" heading, two input fields (both containing "Niran"), and a "Sign in" button. The browser's taskbar at the bottom shows the Windows search bar and system tray with a temperature of 29°C and the date 15-02-2022.

Student Login Page

Student Login form

Sign in

Niran

Niran

Sign in

## SEARCH BOOKS & E-LINKS

The screenshot shows a web browser window with the URL `localhost:8080/Source_Code/Search.jsp`. The page has a purple header with the "E-Library" logo and navigation links: Home, Menu, and Logout. The main content area is titled "Search Book & E-Link" and contains a search form with a text input field (containing "javascript") and a "Search" button. The browser's taskbar at the bottom shows the Windows search bar and system tray with a temperature of 29°C and the date 15-02-2022.

Search Book & E-Link

javascript Search

## VIEW SEARCH BOOK DETAILS

The screenshot displays the 'View Search Book Result :-' page of an E-Library Management System. The page has a purple header with the 'E-Library' logo and navigation links for 'Home', 'Menu', and 'Logout'. The main content area is white and contains two tables. The first table, 'View Search Book Result :-', lists search results with columns for ID, Book Id, Book Name, and View details. The second table, 'View Search E-Link Result :-', lists search results with columns for ID, Book Id, Book Name, Author, and E-Link. Below the tables is a 'Click to back' link.

**View Search Book Result :-**

| ID | Book Id | Book Name  | View details               |
|----|---------|------------|----------------------------|
| 6  | 653     | JavaScript | <a href="#">Click Here</a> |

**View Search E-Link Result :-**

| ID | Book Id | Book Name  | Author | E-Link                                                            |
|----|---------|------------|--------|-------------------------------------------------------------------|
| 8  | 653     | JavaScript | NIL    | <a href="https://www.w3schools.com">https://www.w3schools.com</a> |

[Click to back](#)

## FEEDBACK

The screenshot displays a web browser window with the following elements:

- Browser Tabs:** (2) WhatsApp, E-Library Management System, javascript - Google Search.
- Address Bar:** localhost8080/Source\_Code/givefeed.jsp
- Page Header:** E-Library logo, Home, Menu, Logout.
- Form Title:** Give Feedback
- Form Fields:**
  - FeedBack:** A dropdown menu with the value 'Good' selected.
  - Comment:** A text area containing the text 'very usefull one! Thank you'.
- Form Button:** An 'ORDER' button located below the comment field.
- Windows Taskbar:** Includes a search bar ('Type here to search'), task view icon, and system tray showing '29°C Haze', 'ENG', and '11:20 15-02-2022'.

## CHAPTER 6

### 6SUMMARY AND CONCLUSION

#### 6.1 CONCLUSION:

The Internet has become a major resource in modern business, thus electronic shopping has gained significance not only from the entrepreneur's but also from the customer's point of view. The project to provide the user with easy navigation, retrieval of data and necessary feedback as much as possible. In this project, the user is provided with an e-commerce web site that can be used to buy books online. The shopping cart application described in this project provides a number of features that are designed to make the customer more comfortable. This project helps in understanding the creation of an interactive web page and the technologies used to implement it.

#### 6.2 REFERENCES:

- 1.G. Jeh and J. Widom. Sim Rank: A measure of structural-context similarity. In SIGKDD, 2002.
2. H. Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *Journal of the American Society for Information Science*, 1973, 24,pp. 265–269.
- 3.M. M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 1963, 14, pp.10–25.
- 4.Zhao, P., Han, J., and Sun, Y., “P-rank: a comprehensive structural similarity measure over information networks”[C]. In *Proc. of CIKM*, 2009, pp. 553–562.
5. Wensi Xi, Edward A. Fox, Weiguo Fan, Benyu Zhang, Zheng Chen, Jun Yan, Dong Zhuang. SimFusion: measuring similarity using unified relationship matrix. *SIGIR 2005*: 130-137.
- 6.Fogaras D., Racz B., “Scaling link-based similarity search”[C]. In *Proc. of WWW*, 2005: 641-650.
7. Y.Z. Cai, G. Cong, X. Jia, H.Y. Liu, J. He., “Efficient Algorithms for Computing Link based Similarity in Real World Networks”.

8.Ms. Pragati Bagmare<sup>1</sup> , Ms. Shraddha Girhepunje<sup>2</sup> , Ms. Priya Bisen, “Research Paper on Online Bookshop Management System”, International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 5., Issue 4., 2017, page no. 115-117.

9.Fatin Najwa Binti Abdullah Sani<sup>1</sup> , Hani Malini binti Majek<sup>2</sup> , Umairah binti Ahmad Khairudin<sup>3</sup> , Abdul Rahman bin Ahmad Dahlan<sup>4</sup>, “e-Bookstore: Opening Door to the Garden of Knowledge”, International Journal of Scientific and Research Publications, Volume 7, Issue 6, June 2017 ,page no. 2250-3153

10.Vamsi Krishna Mummaneni ,A Report Submitted in partial fulfillment of the requirements of the degree of Master of Software Engineering.

## APPENDIX

### SAMPLE CODING

#### ADMIN LOGIN

```
public static void loginFn() {  
    //Creating Login Frame  
    JFrame loginFrame = new JFrame("Login");  
    //Creating label Username  
    JLabel l1 = new JLabel("Username", SwingConstants.CENTER);  
    //Creating label Password  
    JLabel l2 = new JLabel("Password", SwingConstants.CENTER);  
    //Setting up opaque so that label component paints every pixel within its bounds.  
    l1.setOpaque(true);  
    //Setting up the background color of the label.  
    l1.setBackground(new Color(51, 35, 85));  
    //Setting up the foreground color of the label.  
    l1.setForeground(Color.white);  
    //Setting up opaque so that label component paints every pixel within its bounds.  
    l2.setOpaque(true);  
    //Setting up the background color of the label.  
    l2.setBackground(new Color(51, 35, 85));  
    //Setting up the foreground color of the label.  
    l2.setForeground(Color.white);  
    //Create textfield Username  
    JTextField usernameTF = new JTextField();  
    //Setting up the background color of the textfield.  
    usernameTF.setBackground(new Color(51, 35, 85));  
    //Setting up the foreground color of the textfield.  
    usernameTF.setForeground(Color.white);  
    //Create textfield Password  
    JPasswordField passwordTF = new JPasswordField();  
    //Setting up the background color of the textfield.  
    passwordTF.setBackground(new Color(51, 35, 85));  
    //Setting up the foreground color of the textfield.  
    passwordTF.setForeground(Color.white);  
    //Create button Login  
    JButton loginBtn = new JButton("Login");  
    //Setting up the background color of the button.  
    loginBtn.setBackground(new Color(124, 85, 227));  
    //Setting up the foreground color of the button.  
    loginBtn.setForeground(Color.white);  
}
```

```

//Create button cancel
JButton cancelBtn = new JButton("Cancel");
//Setting up the background color of the button.
cancelBtn.setBackground(new Color(124, 85, 227));
//Setting up the foreground color of the button.
cancelBtn.setForeground(Color.white);
//Performing action on button.
loginBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String username = usernameTF.getText();
        String password = passwordTF.getText();
        //If username is empty
        if (username.isEmpty()) {
            JOptionPane.showMessageDialog(null, "Please enter username"); //Display
            dialog box with the message
        } //If password is empty
        else if (password.isEmpty()) {
            JOptionPane.showMessageDialog(null, "Please enter password"); //Display dialog
            box with the message
        } //If both the fields are present then to login the user, check whether the user
        exists already
        else {
            //Connect to the database
            Connection connection = connect();
            try {
                Statement stmt = connection.createStatement();
                String st = ("SELECT * FROM USERS WHERE USERNAME='" + username + "'
                AND PASSWORD='" + password + "'"); //Retrieve username and passwords from
                users
                ResultSet rs = stmt.executeQuery(st); //Execute query
                if (rs.next() == false) { //Move pointer below
                    JOptionPane.showMessageDialog(null, "Invalid Username/Password!"); //Display
                    Message
                } else {
                    loginFrame.dispose();
                    rs.beforeFirst(); //Move the pointer above
                    while (rs.next()) {
                        String admin = rs.getString("user_type"); //user is admin
                        System.out.println(admin);
                        String UID = rs.getString("UID"); //Get user ID of the user
                        if (admin.equals("1")) { //If boolean value 1
                            //Redirecting to Librarian Frame

```

```

librarian_frame();
} else {
//Redirecting to User Frame for that user ID
user_frame(UID);
}
}
}
} catch (Exception ex) {
ex.printStackTrace();
}
}
}
});
cancelBtn.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e) {
loginFrame.dispose();
}
});
//Adding all login components in the login frame of java library management
system.
loginFrame.add(l1);
loginFrame.add(usernameTF);
loginFrame.add(l2);
loginFrame.add(passwordTF);
loginFrame.add(loginBtn);
loginFrame.add(cancelBtn);
//Setting size of frame (width, height)
loginFrame.setSize(330, 180);//400 width and 500 height
//Setting layout of the frame
loginFrame.setLayout(new GridLayout(3, 2));
//Setting frame visible to the user
loginFrame.setVisible(true);
//Setting frame non-resizable
loginFrame.setResizable(false);
}

```

## LIBRIRAN MODULE

```
public static void librarian_frame() {

    //Creating Librarian Frame
    JFrame librarianFrame = new JFrame("Librarian Functions");

    //Creating Button
    JButton view_books_btn = new JButton("View Books");

    //Setting up the background color of the button.
    view_books_btn.setBackground(new Color(51, 35, 85));

    //Setting up the foreground color of the button.
    view_books_btn.setForeground(Color.white);

    //Performing actions on button.
    view_books_btn.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {

            //Creating frame.
            JFrame viewBooksFrame = new JFrame("Books Available");

            //Connection to Database
            Connection connection = connect();

            //Query for retrieving data from database
            String sql = "select * from BOOKS";
            try {
                //Creating Statement
                Statement stmt = connection.createStatement();

                //Executing query
                ResultSet rs = stmt.executeQuery(sql);

                //Creating Table for to data will be in table format
                JTable book_list = new JTable();
                String[] bookColumnNames = {"Book ID", "Book ISBN", "Book Name",
                "Book Publisher", "Book Edition", "Book Genre", "Book price", "Book Pages"};

                //Creating model for the table
                DefaultTableModel bookModel = new DefaultTableModel();

                //Setting up the columns names of the model
                bookModel.setColumnIdentifiers(bookColumnNames);

                //Adding model to the table component
                book_list.setModel(bookModel);
```

```

//Setting background colour of the table
book_list.setBackground(new Color(51, 35, 85));

//Setting foreground colour of the table
book_list.setForeground(Color.white);

//Setting up table auto-resizable

book_list.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
book_list.setFillsViewportHeight(true);
book_list.setFocusable(false);

//Creating scrollbars for table
JScrollPane scrollBook = new JScrollPane(book_list);

scrollBook.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

scrollBook.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

while (rs.next()) {
    //Fetching the data from mysql database
    int book_id = rs.getInt(1);
    String book_isbn = rs.getString(2);
    String book_name = rs.getString(3);
    String book_publisher = rs.getString(4);
    String book_edition = rs.getString(5);
    String book_genre = rs.getString(6);
    int book_price = rs.getInt(7);
    int book_pages = rs.getInt(8);
    //Adding fetched data in model
    bookModel.addRow(new Object[]{book_id, book_isbn, book_name,
book_publisher, book_edition, book_genre, book_price, book_pages});
}

//Adding scrollbars in the frame
viewBooksFrame.add(scrollBook);

//Setting up the size of the frame (width,height)
viewBooksFrame.setSize(800, 400);

//Setting up frame visible for user
viewBooksFrame.setVisible(true);
} catch (Exception e1) {
    //Creating Dialog box to show any error if occurred!
    JOptionPane.showMessageDialog(null, e1);
}
}

```

```

});

//Creating button
JButton view_users_btn = new JButton("View Users");

//Setting Background color of the button.
view_users_btn.setBackground(new Color(51, 35, 85));

//Setting Foreground color of the button.
view_users_btn.setForeground(Color.white);

//Performing actions on the button.
view_users_btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        //Creating frame.
        JFrame viewUsersFrame = new JFrame("Users List");

        //Connection to database
        Connection connection = connect();

        //Query for retrieving data from database
        String sql = "select * from users";
        try {
            //Creating Statement
            Statement stmt = connection.createStatement();

            //Executing query
            ResultSet rs = stmt.executeQuery(sql);

            //Creating Table for to data will be in table format
            JTable users_list = new JTable();
            String[] userColumnNames = {"User ID", "User Name", "User Type"};

            //Creating model for the table
            DefaultTableModel userModel = new DefaultTableModel();

            //Setting up the columns names of the model
            userModel.setColumnIdentifiers(userColumnNames);

            //Adding model to the table component
            users_list.setModel(userModel);

            //Setting up table auto-resizable
            users_list.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
            users_list.setFillsViewportHeight(true);

            //Setting background colour of the table.

```

```

users_list.setBackground(new Color(51, 35, 85));

//Setting foreground colour of the table.
users_list.setForeground(Color.white);

//Creating scrollbars for table
JScrollPane scrollUser = new JScrollPane(users_list);

scrollUser.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_
_AS_NEEDED);

scrollUser.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_
NEEDED);

while (rs.next()) {
    //Fetching the data from mysql database
    int uid = rs.getInt(1);
    String user_name = rs.getString(2);
    int user_type = rs.getInt(4);
    if (user_type == 1) {
        //Checking if it is 1 then it is admin
        userModel.addRow(new Object[]{uid, user_name, "ADMIN"});
    } else {
        //Else it will be user
        userModel.addRow(new Object[]{uid, user_name, "USER"});
    }
}

//Adding scrollbars in the frame
viewUsersFrame.add(scrollUser);

//Setting up the size of the frame (width,height)
viewUsersFrame.setSize(800, 400);

//Setting up frame visible for user
viewUsersFrame.setVisible(true);

} catch (Exception e1) {
    //Creating Dialog box to show any error if occurred!
    JOptionPane.showMessageDialog(null, e1);
}

}
});

//Creating button
JButton view_issued_books_btn = new JButton("View Issued Books");

//Setting background colour of the button.

```

```

view_issued_books_btn.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the button.
view_issued_books_btn.setForeground(Color.white);

//Performing actions on button.
view_issued_books_btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        //Creating button
        JFrame issuedBooksFrame = new JFrame("Issued Books List");

        //Connection to database
        Connection connection = connect();

        //Query for retrieving data from database
        String sql = "select * from issued_books";
        try {
            //Creating Statement
            Statement stmt = connection.createStatement();

            //Executing query
            ResultSet rs = stmt.executeQuery(sql);

            //Creating Table for to data will be in table format
            JTable issue_book_list = new JTable();
            String[] issueBookColumnNames = {"Issue ID", "User ID", "Book ID",
"Issue Date", "Period"};

            //Creating model for the table
            DefaultTableModel issuedBookModel = new DefaultTableModel();

            //Setting up the columns names of the model
            issuedBookModel.setColumnIdentifiers(issueBookColumnNames);

            //Adding model to the table component
            issue_book_list.setModel(issuedBookModel);

            //Setting up table auto-resizable
            issue_book_list.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
            issue_book_list.setFillsViewportHeight(true);
            issue_book_list.setFocusable(false);

            //Setting background colour of the table
            issue_book_list.setBackground(new Color(51, 35, 85));

            //Setting foreground colour of the table
            issue_book_list.setForeground(Color.white);

```

```

        //Creating scrollbars for table
        JScrollPane scrollIssuedBook = new JScrollPane(issue_book_list);

scrollIssuedBook.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

scrollIssuedBook.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

        while (rs.next()) {
            //Fetching the data from mysql database
            int iid = rs.getInt(1);
            int uid = rs.getInt(2);
            int bid = rs.getInt(3);
            String issue_date = rs.getString(4);
            int period = rs.getInt(5);
            //Adding fetched data in model
            issuedBookModel.addRow(new Object[]{iid, uid, bid, issue_date,
period});
        }

        //Adding scrollbars in the frame
        issuedBooksFrame.add(scrollIssuedBook);

        //Setting up the size of the frame (width,height)
        issuedBooksFrame.setSize(800, 400);

        //Setting up frame visible for user
        issuedBooksFrame.setVisible(true);

    } catch (Exception e1) {
        //Creating Dialog box to show any error if occurred!
        JOptionPane.showMessageDialog(null, e1);
    }
});

//Creating button
JButton view_returned_books_btn = new JButton("View Returned Books");

//Setting Background Colour of the button.
view_returned_books_btn.setBackground(new Color(51, 35, 85));

//Setting Foreground Colour of the button.
view_returned_books_btn.setForeground(Color.white);

//Performing actions on the button.
view_returned_books_btn.addActionListener(new ActionListener() {
    @Override

```

```

public void actionPerformed(ActionEvent e) {

    //Creating button.
    JFrame returnedBooksFrame = new JFrame("Returned Books List");

    //Connection between database and java library management system.
    Connection connection = connect();

    //Query for retrieving data from database.
    String sql = "select * from returned_books";
    try {
        //Creating Statement.
        Statement stmt = connection.createStatement();

        //Executing query.
        ResultSet rs = stmt.executeQuery(sql);

        //Creating Table for data will be in table format.
        JTable returned_book_list = new JTable();

        String[] returnBookColumnNames = {"Return ID", "Book ID", "User
ID", "Return Date", "Fine"};

        //Creating a model for the table.
        DefaultTableModel returnBookModel = new DefaultTableModel();

        //Setting up the column names of the model.
        returnBookModel.setColumnIdentifiers(returnBookColumnNames);

        //Adding model to the table component.
        returned_book_list.setModel(returnBookModel);

        //Setting up the table auto-resizable.

returned_book_list.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
        returned_book_list.setFillsViewportHeight(true);
        returned_book_list.setFocusable(false);

        //Setting background colour of the table.
        returned_book_list.setBackground(new Color(51, 35, 85));

        //Setting the foreground colour of the table.
        returned_book_list.setForeground(Color.white);

        //Creating scrollbars for tables.
        JScrollPane scrollReturnedBook = new
JScrollPane(returned_book_list);

scrollReturnedBook.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SC
ROLLBAR_AS_NEEDED);

```

```
scrollReturnedBook.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
```

```
        while (rs.next()) {
            //Fetching the data from the mysql database.
            int rid = rs.getInt(1);
            int bid = rs.getInt(2);
            int uid = rs.getInt(3);
            String returned_date = rs.getString(4);
            int fine = rs.getInt(5);

            //Adding fetched data in model.
            returnBookModel.addRow(new Object[]{rid, bid, uid, returned_date,
fine});
        }
```

```
        //Adding scrollbars in the frame.
        returnedBooksFrame.add(scrollReturnedBook);
```

```
        //Setting up the size of the frame (width,height)
        returnedBooksFrame.setSize(800, 400);
```

```
        //Setting up frames visible for the user.
        returnedBooksFrame.setVisible(true);
```

```
    } catch (Exception e1) {
        //Creating Dialog box to show any error if occurred!
        JOptionPane.showMessageDialog(null, e1);
    }
}
});
```

```
//Creating button
JButton add_user_btn = new JButton("Add User");
```

```
//Setting Background Colour of the button.
add_user_btn.setBackground(new Color(51, 35, 85));
```

```
//Setting Foreground Colour of the button.
add_user_btn.setForeground(Color.white);
```

```
//Performing actions on buttons.
add_user_btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
```

```
        //Creating frame
        JFrame add_user_frame = new JFrame("Enter User Details"); //Frame
to enter user details
```

```

//Creating label
JLabel l1 = new JLabel("Username", SwingConstants.CENTER);

//Setting up opaque so that label component paints every pixel within its
bounds.
l1.setOpaque(true);

//Setting Background Colour of the label.
l1.setBackground(new Color(51, 35, 85));

//Setting Foreground Colour of the label.
l1.setForeground(Color.white);

//Creating label
JLabel l2 = new JLabel("Password", SwingConstants.CENTER);

//Setting up opaque so that label component paints every pixel within its
bounds.
l2.setOpaque(true);

//Setting Background Colour of the label.
l2.setBackground(new Color(51, 35, 85));

//Setting Foreground Colour of the label.
l2.setForeground(Color.white);

//Creating textfield
JTextField add_username_tf = new JTextField();

//Setting Background Colour of the textfield.
add_username_tf.setBackground(new Color(51, 35, 85));

//Setting Foreground Colour of the textfield.
add_username_tf.setForeground(Color.white);

//Creating textfield
JPasswordField add_password_tf = new JPasswordField();

//Setting Background Colour of the textfield.
add_password_tf.setBackground(new Color(51, 35, 85));

//Setting Foreground Colour of the textfield.
add_password_tf.setForeground(Color.white);

//Creating radio button
JRadioButton user_type_radio1 = new JRadioButton("Admin");

//Aligning center
user_type_radio1.setHorizontalAlignment(SwingConstants.CENTER);

```

```

//Setting Background Colour of the radiobutton.
user_type_radio1.setBackground(new Color(51, 35, 85));

//Setting Foreground Colour of the radiobutton.
user_type_radio1.setForeground(Color.white);

//Creating radio button
JRadioButton user_type_radio2 = new JRadioButton("User");

//Aligning center
user_type_radio2.setHorizontalAlignment(SwingConstants.CENTER);

//Setting Background Colour of the radiobutton.
user_type_radio2.setBackground(new Color(51, 35, 85));

//Setting Foreground Colour of the radiobutton.
user_type_radio2.setForeground(Color.white);

//Adding radio buttons in buttongroup
ButtonGroup user_type_btn_grp = new ButtonGroup();
user_type_btn_grp.add(user_type_radio1);
user_type_btn_grp.add(user_type_radio2);

//Creating button.
JButton create_btn = new JButton("Create");

//Setting Background Colour of the button.
create_btn.setBackground(new Color(124, 85, 227));

//Setting Foreground Colour of the button.
create_btn.setForeground(Color.white);

//Creating button.
JButton user_entry_cancel_btn = new JButton("Cancel");

//Setting Background Colour of the button.
user_entry_cancel_btn.setBackground(new Color(124, 85, 227));

//Setting Foreground Colour of the button.
user_entry_cancel_btn.setForeground(Color.white);

//Performing actions on the button.
create_btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        //Getting data from the textfield.
        String username = add_username_tf.getText();
        String password = add_password_tf.getText();
    }
});

```

```

//Connection to database.
Connection connection = connect();

try {
    //Creating statement
    Statement stmt = connection.createStatement();

    //Check if radio1 is click or not
    //If radio1 is click then it is added as admin, else normal student
    if (user_type_radio1.isSelected()) {
        //Query to insert inside in the table
        stmt.executeUpdate("INSERT INTO
USERS(USERNAME,PASSWORD,USER_TYPE) VALUES ('" + username + "','" +
password + "','" + "1" + "')");
        //Creating Dialog Box to display message.
        JOptionPane.showMessageDialog(null, "Admin added!");
        add_user_frame.dispose();
    } else {
        //Query to insert inside in the table.
        stmt.executeUpdate("INSERT INTO
USERS(USERNAME,PASSWORD,USER_TYPE) VALUES ('" + username + "','" +
password + "','" + "0" + "')");
        //Creating Dialog Box to display message.
        JOptionPane.showMessageDialog(null, "User added!");
        add_user_frame.dispose();
    }

} catch (Exception e1) {
    //Creating Dialog box to show any error if occurred!
    JOptionPane.showMessageDialog(null, e1);
}

});

//Performing actions on button.
user_entry_cancel_btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        add_user_frame.dispose();
    }
});

//Adding components in frame.
add_user_frame.add(l1);
add_user_frame.add(add_username_tf);
add_user_frame.add(l2);
add_user_frame.add(add_password_tf);
add_user_frame.add(user_type_radio1);
add_user_frame.add(user_type_radio2);

```

```

        add_user_frame.add(create_btn);
        add_user_frame.add(user_entry_cancel_btn);

        //Setting up the size of the frame (width,height)
        add_user_frame.setSize(350, 200);

        //Setting up layout of the frame
        add_user_frame.setLayout(new GridLayout(4, 2));

        //Setting up the frame visible
        add_user_frame.setVisible(true);

        //Setting up the table auto-resizable.
        add_user_frame.setResizable(false);
    }
});

//Creating button.
JButton add_book_btn = new JButton("Add Book");

//Setting Background Colour of the button.
add_book_btn.setBackground(new Color(51, 35, 85));

//Setting Foreground Colour of the button.
add_book_btn.setForeground(Color.white);

//Performing actions on button.
add_book_btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        //Creating Frame.
        JFrame book_frame = new JFrame("Enter Book Details");

        //Creating labels
        JLabel l1, l2, l3, l4, l5, l6, l7;

        l1 = new JLabel("ISBN", SwingConstants.CENTER);

        //Setting up opaque so that label component paints every pixel within its
        bounds.
        l1.setOpaque(true);

        //Setting background colour of the label.
        l1.setBackground(new Color(51, 35, 85));

        //Setting the foreground colour of the label.
        l1.setForeground(Color.white);
    }
});

```

```

l2 = new JLabel("Name", SwingConstants.CENTER);

//Setting up opaque so that label component paints every pixel within its
bounds.
l2.setOpaque(true);

//Setting background colour of the label.
l2.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the label.
l2.setForeground(Color.white);

l3 = new JLabel("Publisher", SwingConstants.CENTER);

//Setting up opaque so that label component paints every pixel within its
bounds.
l3.setOpaque(true);

//Setting background colour of the label.
l3.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the label.
l3.setForeground(Color.white);

l4 = new JLabel("Edition", SwingConstants.CENTER);

//Setting up opaque so that label component paints every pixel within its
bounds.
l4.setOpaque(true);

//Setting background colour of the label.
l4.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the label.
l4.setForeground(Color.white);

l5 = new JLabel("Genre", SwingConstants.CENTER);

//Setting up opaque so that label component paints every pixel within its
bounds.
l5.setOpaque(true);

//Setting background colour of the label.
l5.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the label.
l5.setForeground(Color.white);

l6 = new JLabel("Price", SwingConstants.CENTER);

```

```

//Setting up opaque so that label component paints every pixel within its
bounds.
l6.setOpaque(true);

//Setting background colour of the label.
l6.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the label.
l6.setForeground(Color.white);

l7 = new JLabel("Pages", SwingConstants.CENTER);

//Setting up opaque so that label component paints every pixel within its
bounds.
l7.setOpaque(true);

//Setting background colour of the label.
l7.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the label.
l7.setForeground(Color.white);

//Creating textfield.
JTextField book_isbn_tf = new JTextField();

//Setting background colour of the textfield.
book_isbn_tf.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the textfield.
book_isbn_tf.setForeground(Color.white);

//Creating textfield
JTextField book_name_tf = new JTextField();

//Setting background colour of the textfield.
book_name_tf.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the textfield.
book_name_tf.setForeground(Color.white);

//Creating textfield.
JTextField book_publisher_tf = new JTextField();

//Setting background colour of the textfield.
book_publisher_tf.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the textfield.
book_publisher_tf.setForeground(Color.white);

//Creating textfield.

```

```

JTextField book_edition_tf = new JTextField();

//Setting background colour of the textfield.
book_edition_tf.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the textfield.
book_edition_tf.setForeground(Color.white);

//Creating textfield.
JTextField book_genre_tf = new JTextField();

//Setting background colour of the textfield.
book_genre_tf.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the textfield.
book_genre_tf.setForeground(Color.white);

//Creating textfield.
JTextField book_price_tf = new JTextField();

//Setting background colour of the textfield.
book_price_tf.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the textfield.
book_price_tf.setForeground(Color.white);

//Creating textfield.
JTextField book_pages_tf = new JTextField();

//Setting background colour of the textfield.
book_pages_tf.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the textfield.
book_pages_tf.setForeground(Color.white);

//Creating button.
JButton create_btn = new JButton("Submit");

//Setting background colour of the button.
create_btn.setBackground(new Color(124, 85, 227));

//Setting the foreground colour of the button.
create_btn.setForeground(Color.white);

//Creating button.
JButton add_book_cancel_btn = new JButton("Cancel");

//Setting background colour of the button.
add_book_cancel_btn.setBackground(new Color(124, 85, 227));

```

```

//Setting the foreground colour of the button.
add_book_cancel_btn.setForeground(Color.white);

//Performing actions on the button.
create_btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        //Getting data from the textfield.
        String book_isbn = book_isbn_tf.getText();
        String book_name = book_name_tf.getText();
        String book_publisher = book_publisher_tf.getText();
        String book_edition = book_edition_tf.getText();
        String book_genre = book_genre_tf.getText();

        //Converting bookprice and bookpages to integer from string.
        int book_price = Integer.parseInt(book_price_tf.getText());
        int book_pages = Integer.parseInt(book_pages_tf.getText());

        //Connection to database.
        Connection connection = connect();

        try {
            //Creating statement
            Statement stmt = connection.createStatement();

            //Query to insert in the table.
            stmt.executeUpdate("INSERT INTO
BOOKS(book_isbn,book_name,book_publisher,book_edition,book_genre,book_pr
ice,book_pages)"
                + " VALUES (" + book_isbn + "," + book_name + "," +
book_publisher + "," + book_edition + "," + book_genre + "," + book_price + ","
+ book_pages + ")");

            //Creating Dialog Box to display message.
            JOptionPane.showMessageDialog(null, "Book added!");
            book_frame.dispose();
        } catch (Exception e1) {
            //Creating Dialog box to show any error if occurred!
            JOptionPane.showMessageDialog(null, e1);
        }
    }
});

//Performing actions on the button.
add_book_cancel_btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        book_frame.dispose();
    }
}

```

```

});

//Adding components in the frame.
book_frame.add(l1);
book_frame.add(book_isbn_tf);
book_frame.add(l2);
book_frame.add(book_name_tf);
book_frame.add(l3);
book_frame.add(book_publisher_tf);
book_frame.add(l4);
book_frame.add(book_edition_tf);
book_frame.add(l5);
book_frame.add(book_genre_tf);
book_frame.add(l6);
book_frame.add(book_price_tf);
book_frame.add(l7);
book_frame.add(book_pages_tf);
book_frame.add(create_btn);
book_frame.add(add_book_cancel_btn);

//Setting up the size of the frame (width,height)
book_frame.setSize(800, 500);

//Setting up layout of the frame
book_frame.setLayout(new GridLayout(8, 2));

//Setting up the frame visible
book_frame.setVisible(true);

//Setting up the table auto-resizable.
book_frame.setResizable(false);
}
});

//Creating button
JButton add_issue_book_btn = new JButton("Issue Book");

//Setting background colour of the button.
add_issue_book_btn.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the button.
add_issue_book_btn.setForeground(Color.white);

//Performing actions on the button.
add_issue_book_btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        //Creating frame.
        JFrame issue_book_frame = new JFrame("Enter Details");

```

```

//Creating panel.
JPanel pickerPanel = new JPanel();

//Creating a datepicker.
JXDatePicker picker = new JXDatePicker();

//Setting up current date in datepicker
picker.setDate(Calendar.getInstance().getTime());

//Formatting datepicker.
picker.setFormats(new SimpleDateFormat("dd.MM.yyyy"));

//Adding datepicker in the panel.
pickerPanel.add(picker);

//Setting background colour of the panel.
pickerPanel.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the panel.
pickerPanel.setForeground(Color.white);

//Creating labels
JLabel l1, l2, l3, l4;
l1 = new JLabel("Book ID", SwingConstants.CENTER);

//Setting up opaque so that label component paints every pixel within its
bounds.
l1.setOpaque(true);

//Setting background colour of the label.
l1.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the label.
l1.setForeground(Color.white);

l2 = new JLabel("User/Student ID", SwingConstants.CENTER);

//Setting up opaque so that label component paints every pixel within its
bounds.
l2.setOpaque(true);

//Setting background colour of the label.
l2.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the label.
l2.setForeground(Color.white);

l3 = new JLabel("Period(days)", SwingConstants.CENTER);

```

bounds. //Setting up opaque so that label component paints every pixel within its

```
l3.setOpaque(true);
```

```
//Setting background colour of the label.  
l3.setBackground(new Color(51, 35, 85));
```

```
//Setting the foreground colour of the label.  
l3.setForeground(Color.white);
```

```
l4 = new JLabel("Issued Date(DD-MM-YYYY)",  
SwingConstants.CENTER);
```

bounds. //Setting up opaque so that label component paints every pixel within its

```
l4.setOpaque(true);
```

```
//Setting background colour of the label.  
l4.setBackground(new Color(51, 35, 85));
```

```
//Setting the foreground colour of the label.  
l4.setForeground(Color.white);
```

```
//Creating textfield.  
JTextField bid_tf = new JTextField();
```

```
//Setting background colour of the textfield.  
bid_tf.setBackground(new Color(51, 35, 85));
```

```
//Setting the foreground colour of the textfield.  
bid_tf.setForeground(Color.white);
```

```
//Creating textfield.  
JTextField uid_tf = new JTextField();
```

```
//Setting background colour of the textfield.  
uid_tf.setBackground(new Color(51, 35, 85));
```

```
//Setting the foreground colour of the textfield.  
uid_tf.setForeground(Color.white);
```

```
//Creating textfield.  
JTextField period_tf = new JTextField();
```

```
//Setting background colour of the textfield.  
period_tf.setBackground(new Color(51, 35, 85));
```

```
//Setting the foreground colour of the textfield.  
period_tf.setForeground(Color.white);
```

```

//Creating button.
JButton create_btn = new JButton("Submit");

//Setting background colour of the button.
create_btn.setBackground(new Color(124, 85, 227));

//Setting the foreground colour of the button.
create_btn.setForeground(Color.white);

//Creating button.
JButton issue_book_cancel_btn = new JButton("Cancel");

//Setting background colour of the button.
issue_book_cancel_btn.setBackground(new Color(124, 85, 227));

//Setting the foreground colour of the button.
issue_book_cancel_btn.setForeground(Color.white);

//Performing actions on the button.
create_btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        //Getting data from textfield.
        int uid = Integer.parseInt(uid_tf.getText());
        int bid = Integer.parseInt(bid_tf.getText());
        String period = period_tf.getText();
        Date oDate = picker.getDate();

        //Formatting date.
        DateFormat oDateFormat = new SimpleDateFormat("dd-MM-
yyyy");

        String issued_date = oDateFormat.format(oDate);

        //Converting period from string to integer.
        int period_int = Integer.parseInt(period);

        //Connection to the database
        Connection connection = connect();

        try {
            //Creating Statement
            Statement stmt = connection.createStatement();

            //Query to insert data in the table.
            stmt.executeUpdate("INSERT INTO
issued_books(UID,BID,ISSUED_DATE,PERIOD) VALUES ('" + uid + "','" + bid +
',' + issued_date + "','" + period_int + "')");

            //Creating Dialog Box to display message.

```

```

        JOptionPane.showMessageDialog(null, "Book Issued!");
        issue_book_frame.dispose();

    } catch (Exception e1) {
        //Creating Dialog box to show any error if occurred!
        JOptionPane.showMessageDialog(null, e1);
    }
}
});

issue_book_cancel_btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        issue_book_frame.dispose();
    }
});

//Adding components in the frame
issue_book_frame.add(l1);
issue_book_frame.add(bid_tf);
issue_book_frame.add(l2);
issue_book_frame.add(uid_tf);
issue_book_frame.add(l3);
issue_book_frame.add(period_tf);
issue_book_frame.add(l4);
issue_book_frame.getContentPane().add(pickerPanel);
issue_book_frame.add(create_btn);
issue_book_frame.add(issue_book_cancel_btn);

//Setting up the size of the frame (width,height)
issue_book_frame.setSize(600, 300);

//Setting up frame layout
issue_book_frame.setLayout(new GridLayout(5, 2));

//Setting up the frame visible
issue_book_frame.setVisible(true);

//Setting up table auto-resizable.
issue_book_frame.setResizable(false);
}
});

//Creating button.
JButton add_return_book_btn = new JButton("Return Book");

//Setting background colour of the button.
add_return_book_btn.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the button.

```

```

add_return_book_btn.setForeground(Color.white);

//Performing actions on the button.
add_return_book_btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        //Creating frame.
        JFrame returnBookFrame = new JFrame("Enter Details");

        //Creating the labels.
        JLabel l1 = new JLabel("Book ID", SwingConstants.CENTER);

        //Setting up opaque so that label component paints every pixel within its
bounds.
        l1.setOpaque(true);

        //Setting background colour of the label.
        l1.setBackground(new Color(51, 35, 85));

        //Setting the foreground colour of the label.
        l1.setForeground(Color.white);

        JLabel l2 = new JLabel("User ID", SwingConstants.CENTER);

        //Setting up opaque so that label component paints every pixel within its
bounds.
        l2.setOpaque(true);

        //Setting background colour of the label.
        l2.setBackground(new Color(51, 35, 85));

        //Setting the foreground colour of the label.
        l2.setForeground(Color.white);

        //Creating labels.
        JLabel l3 = new JLabel("Return Date(DD-MM-YYYY)",
SwingConstants.CENTER);

        //Setting up opaque so that label component paints every pixel within its
bounds.
        l3.setOpaque(true);

        //Setting background colour of the label.
        l3.setBackground(new Color(51, 35, 85));

        //Setting the foreground colour of the label.
        l3.setForeground(Color.white);

        JLabel l4 = new JLabel("Fine", SwingConstants.CENTER);

```

bounds.  
//Setting up opaque so that label component paints every pixel within its

```
l4.setOpaque(true);
```

```
//Setting background colour of the label.  
l4.setBackground(new Color(51, 35, 85));
```

```
//Setting the foreground colour of the label.  
l4.setForeground(Color.white);
```

```
//Creating textfield.  
JTextField bid_tf = new JTextField();
```

```
//Setting background colour of the textfield.  
bid_tf.setBackground(new Color(51, 35, 85));
```

```
//Setting the foreground colour of the textfield.  
bid_tf.setForeground(Color.white);
```

```
//Creating textfield.  
JTextField uid_tf = new JTextField();
```

```
//Setting background colour of the textfield.  
uid_tf.setBackground(new Color(51, 35, 85));
```

```
//Setting the foreground colour of the textfield.  
uid_tf.setForeground(Color.white);
```

```
//Creating panel for date.  
JPanel pickerPanel = new JPanel();
```

```
//Creating a datepicker.  
JXDatePicker picker = new JXDatePicker();
```

```
//Getting and Setting up the current time of the date.  
picker.setDate(Calendar.getInstance().getTime());
```

```
//Setting up the format of the date picker.  
picker.setFormats(new SimpleDateFormat("dd.MM.yyyy"));
```

```
//Creating textfield.  
JTextField fine_tf = new JTextField();
```

```
//Setting background colour of the textfield.  
fine_tf.setBackground(new Color(51, 35, 85));
```

```
//Setting the foreground colour of the textfield.  
fine_tf.setForeground(Color.white);
```

```

//Adding datepicker in panel.
pickerPanel.add(picker);

//Setting background colour of the panel.
pickerPanel.setBackground(new Color(51, 35, 85));

//Setting the foreground colour of the panel.
pickerPanel.setForeground(Color.white);

//Creating button.
JButton return_book_btn = new JButton("Return");

//Setting background colour of the button.
return_book_btn.setBackground(new Color(124, 85, 227));

//Setting the foreground colour of the button.
return_book_btn.setForeground(Color.white);

//Creating button.
JButton cancel_book_btn = new JButton("Cancel");

//Setting background colour of the button.
cancel_book_btn.setBackground(new Color(124, 85, 227));

//Setting the foreground colour of the button.
cancel_book_btn.setForeground(Color.white);

//Performing actions on button.
return_book_btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        //Getting data from text fields.
        int bid = Integer.parseInt(bid_tf.getText());
        int uid = Integer.parseInt(uid_tf.getText());
        int fine = Integer.parseInt(fine_tf.getText());
        Date oDate = picker.getDate();
        //Formatting Date.
        DateFormat oDateFormat = new SimpleDateFormat("dd-MM-
yyyy");

        String return_date = oDateFormat.format(oDate);

        try {
            //Connection to database
            Connection connection = connect();

            //Creating Statement
            Statement stmt = connection.createStatement();

            //Querying to insert in the table.

```

```

        stmt.executeUpdate("INSERT INTO
returned_books(bid,uid,return_date,fine) VALUES ('" + bid + "','" + uid + "','" +
return_date + "','" + fine + "')");

```

```

        //Creating Dialog Box to display message.
        JOptionPane.showMessageDialog(null, "Book Returned!");
        returnBookFrame.dispose();

```

```

    } catch (Exception e1) {
        //Creating Dialog box to show any error if occurred!
        JOptionPane.showMessageDialog(null, e1);
    }
}
});

```

```

//Performing actions on the button.
cancel_book_btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        returnBookFrame.dispose();
    }
});

```

```

//Adding all return book components in the frame
returnBookFrame.add(l1);
returnBookFrame.add(bid_tf);
returnBookFrame.add(l2);
returnBookFrame.add(uid_tf);
returnBookFrame.add(l3);
returnBookFrame.getContentPane().add(pickerPanel);
returnBookFrame.add(l4);
returnBookFrame.add(fine_tf);
returnBookFrame.add(return_book_btn);
returnBookFrame.add(cancel_book_btn);

```

```

//Setting up the size of the frame
returnBookFrame.setSize(600, 300);

```

```

//Setting up the layout of the frame
returnBookFrame.setLayout(new GridLayout(5, 2));

```

```

//Setting up the frame visible
returnBookFrame.setVisible(true);

```

```

//Setting up frame non-resizable
returnBookFrame.setResizable(false);

```

```

    }
});

```

```

//Setting the layout of Librarian Frame

```

```

librarianFrame.setLayout(new GridLayout(2, 4));

//Adding Librarian components in the Librarian Frame
librarianFrame.add(add_user_btn);
librarianFrame.add(add_book_btn);
librarianFrame.add(add_issue_book_btn);
librarianFrame.add(add_return_book_btn);
librarianFrame.add(view_users_btn);
librarianFrame.add(view_books_btn);
librarianFrame.add(view_issued_books_btn);
librarianFrame.add(view_returned_books_btn);

//Setting size of the frame (width,height)
librarianFrame.setSize(800, 200);

//Setting up the frame visible to the user
librarianFrame.setVisible(true);

//Setting up frame non-resizable
librarianFrame.setResizable(false);

}

```

## STUDENT MODULE

```

public static void user_frame(String UID) {

    //Creating Frame for Student
    JFrame studentFrame = new JFrame("Student Functions");

    //Creating button
    JButton view_books_btn = new JButton("View Books");

    //Setting Background Colour of the button.
    view_books_btn.setBackground(new Color(51, 35, 85));

    //Setting Foreground Colour of the button.
    view_books_btn.setForeground(Color.white);

    view_books_btn.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {

            //Creating Frame.
            JFrame viewBooksUserFrame = new JFrame("Books Available");

            //Connection to database.
            Connection connection = connect();

```

```

//Query for retrieving data from database.
String sql = "select * from books";
try {
    //Creating Statement.
    Statement stmt = connection.createStatement();

    //Executing query.
    ResultSet rs = stmt.executeQuery(sql);

    //Creating Table for data will be in table format.
    JTable book_list = new JTable();
    String[] bookColumnNames = {"Book ID", "Book ISBN", "Book Name",
    "Book Publisher", "Book Edition", "Book Genre", "Book price", "Book Pages"};

    //Creating a model for the table.
    DefaultTableModel bookModel = new DefaultTableModel();

    //Setting up the column names of the model.
    bookModel.setColumnIdentifiers(bookColumnNames);

    //Adding model to the table component.
    book_list.setModel(bookModel);

    //Setting background colour of the table.
    book_list.setBackground(new Color(51, 35, 85));

    //Setting the foreground colour of the table.
    book_list.setForeground(Color.white);

    //Setting up the table auto-resizable.

book_list.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
    book_list.setFillViewportHeight(true);
    book_list.setFocusable(false);

    //Creating scrollbars for table.
    JScrollPane scrollBook = new JScrollPane(book_list);

scrollBook.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_
_AS_NEEDED);

scrollBook.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_
NEEDED);

    while (rs.next()) {
        //Fetching the data from mysql database
        int book_id = rs.getInt(1);
        String book_isbn = rs.getString(2);
        String book_name = rs.getString(3);
        String book_publisher = rs.getString(4);
    }
}

```

```

        String book_edition = rs.getString(5);
        String book_genre = rs.getString(6);
        int book_price = rs.getInt(7);
        int book_pages = rs.getInt(8);
        //Adding fetched data in model
        bookModel.addRow(new Object[]{book_id, book_isbn, book_name,
book_publisher, book_edition, book_genre, book_price, book_pages});
    }

    //Adding scrollbars in the frame.
    viewBooksUserFrame.add(scrollBook);

    //Setting up the size of the frame. (width,height)
    viewBooksUserFrame.setSize(800, 400);

    //Setting up frame visible for the user.
    viewBooksUserFrame.setVisible(true);

    } catch (Exception e1) {
        //Creating Dialog box to show any error if occurred!
        JOptionPane.showMessageDialog(null, e1);
    }
}
);

//Creating Button.
JButton view_user_issued_books_btn = new JButton("Issued Books");

//Setting Background color of the button.
view_user_issued_books_btn.setBackground(new Color(51, 35, 85));

//Setting Foreground color of the button.
view_user_issued_books_btn.setForeground(Color.white);

//Performing action on the button.
view_user_issued_books_btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        //Creating frame
        JFrame viewUserIssuedBooksFrame = new JFrame("My Issued Books");

        //Storing userid
        int userid = Integer.parseInt(UID);

        //Connection to database
        Connection connection = connect();

        //Database Query

```

```

        String sql = "select issued_books.iid as iid, issued_books.bid as bid,
issued_books.uid as uid,"
            + " books.book_isbn as book_isbn, books.book_name as
book_name, books.book_publisher as book_publisher, "
            + "books.book_edition as book_edition, books.book_genre as
book_genre, books.book_price as book_price,"
            + " books.book_pages as book_pages, issued_books.issued_date
as issued_date, issued_books.period as period from books,"
            + "issued_books where books.bid=issued_books.bid and
issued_books.uid=" + userid;

        try {
            //Creating statement
            Statement stmt = connection.createStatement();

            //Executing query
            ResultSet rs = stmt.executeQuery(sql);

            //Creating Table for to data will be in table format
            JTable issued_book_list = new JTable();

            String[] issuedBookColumnNames = {"Issue ID", "Book ID", "User ID",
"Book ISBN", "Book Name", "Book Publisher", "Book Edition", "Book Genre",
"Book Price", "Book Pages", "Issued Date", "Period"};

            //Creating model for the table
            DefaultTableModel bookModel = new DefaultTableModel();

            //Setting up the columns names of the model
            bookModel.setColumnIdentifiers(issuedBookColumnNames);

            //Adding model to the table component
            issued_book_list.setModel(bookModel);

            //Setting background colour of the table
            issued_book_list.setBackground(new Color(51, 35, 85));

            //Setting foreground colour of the table
            issued_book_list.setForeground(Color.white);

            //Setting up table auto-resizable
            issued_book_list.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
            issued_book_list.setFillViewportHeight(true);
            issued_book_list.setFocusable(false);

            //Creating scrollbars for table
            JScrollPane scrollIssuedBook = new JScrollPane(issued_book_list);

```

```
scrollIssuedBook.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
```

```
scrollIssuedBook.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
```

```
        while (rs.next()) {
            //Fetching the data from mysql database
            int iid = rs.getInt(1);
            int bid = rs.getInt(2);
            int uid = rs.getInt(3);
            String book_isbn = rs.getString(4);
            String book_name = rs.getString(5);
            String book_publisher = rs.getString(6);
            String book_edition = rs.getString(7);
            String book_genre = rs.getString(8);
            int book_price = rs.getInt(9);
            int book_pages = rs.getInt(10);
            String issued_date = rs.getString(11);
            int period = rs.getInt(12);
            //Adding fetched data in model
            bookModel.addRow(new Object[]{iid, bid, uid, book_isbn,
book_name, book_publisher, book_edition, book_genre, book_price, book_pages,
issued_date, period});
        }
```

```
        //Adding scrollbars.
```

```
        viewUserIssuedBooksFrame.add(scrollIssuedBook);
```

```
        //Setting up the dimensions of the frame.
```

```
        viewUserIssuedBooksFrame.setSize(1200, 600);
```

```
        //Setting up the frame visible.
```

```
        viewUserIssuedBooksFrame.setVisible(true);
```

```
        //Setting up the frame non-resizable.
```

```
        viewUserIssuedBooksFrame.setResizable(false);
```

```
    } catch (Exception e1) {
```

```
        //Creating Dialog box to show any error if occurred!
```

```
        JOptionPane.showMessageDialog(null, e1);
```

```
    }
```

```
}
```

```
});
```

```
        //Creating Button
```

```
        JButton view_user_returned_books_btn = new JButton("My Returned Books");
```

```
        //Setting Background color of the button.
```

```

view_user_returned_books_btn.setBackground(new Color(51, 35, 85));

//Setting Foreground color of the button.
view_user_returned_books_btn.setForeground(Color.white);

//Performing action on the button.
view_user_returned_books_btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        //Creating frame
        JFrame viewUserReturnedBooksFrame = new JFrame("My Returned
Books");

        //Storing userid
        int userid = Integer.parseInt(UID);

        //Connection to database
        Connection connection = connect();

        //Query for retrieving java library management data from database
        String sql = "select returned_books.rid as rid, returned_books.bid as bid,
returned_books.uid as uid,"
            + "books.book_isbn as book_isbn, books.book_name as
book_name, books.book_publisher as book_publisher,"
            + "books.book_edition as book_edition, books.book_genre as
book_genre, books.book_price as book_price,"
            + "books.book_pages as book_pages, returned_books.return_date
as return_date, returned_books.fine as fine "
            + "from books, returned_books where
books.bid=returned_books.bid and returned_books.uid=" + userid;

        try {
            //Creating Statement
            Statement stmt = connection.createStatement();

            //Executing query
            ResultSet rs = stmt.executeQuery(sql);

            //Creating Table for to data will be in table format
            JTable returned_book_list = new JTable();
            String[] returnedBookColumnNames = {"Return ID", "Book ID", "User
ID", "Book ISBN", "Book Name", "Book Publisher", "Book Edition", "Book Genre",
"Book Price", "Book Pages", "Returned Date", "Fine"};

            //Creating model for the table
            DefaultTableModel bookModel = new DefaultTableModel();

            //Setting up the columns names of the model
            bookModel.setColumnIdentifiers(returnedBookColumnNames);

```

```

//Adding model to the table component
returned_book_list.setModel(bookModel);

//Setting background colour of the table
returned_book_list.setBackground(new Color(51, 35, 85));

//Setting foreground colour of the table
returned_book_list.setForeground(Color.white);

//Setting up table auto-resizable

returned_book_list.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
returned_book_list.setFillsViewportHeight(true);
returned_book_list.setFocusable(false);

//Creating scrollbars for table
JScrollPane scrollIssuedBook = new JScrollPane(returned_book_list);

scrollIssuedBook.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

scrollIssuedBook.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

while (rs.next()) {
    //Fetching the data from mysql database
    int rid = rs.getInt(1);
    int bid = rs.getInt(2);
    int uid = rs.getInt(3);
    String book_isbn = rs.getString(4);
    String book_name = rs.getString(5);
    String book_publisher = rs.getString(6);
    String book_edition = rs.getString(7);
    String book_genre = rs.getString(8);
    int book_price = rs.getInt(9);
    int book_pages = rs.getInt(10);
    String returned_date = rs.getString(11);
    int fine = rs.getInt(12);
    //Adding fetched library management data in model
    bookModel.addRow(new Object[]{rid, bid, uid, book_isbn,
book_name, book_publisher, book_edition, book_genre, book_price, book_pages,
returned_date, fine});
}

//Adding scrollbars.
viewUserReturnedBooksFrame.add(scrollIssuedBook);

//Setting up the dimensions of the frame. Params:(width,height)
viewUserReturnedBooksFrame.setSize(1200, 600);

```

```

        //Setting up the frame visible.
        viewUserReturnedBooksFrame.setVisible(true);

        //Setting up the frame is non-resizable.
        viewUserReturnedBooksFrame.setResizable(false);

    } catch (Exception e1) {
        //Creating Dialog box to show any error if occurred!
        JOptionPane.showMessageDialog(null, e1);
    }
}

});

//Setting Layout of the student frame.
studentFrame.setLayout(new GridLayout(3, 1));

//Adding all the components in the student frame.
studentFrame.add(view_books_btn);
studentFrame.add(view_user_issued_books_btn);
studentFrame.add(view_user_returned_books_btn);

//Setting size of the student frame (width,height)
studentFrame.setSize(500, 500);

//Setting up the frame visible
studentFrame.setVisible(true);

//Setting up frame non-resizable
studentFrame.setResizable(false);
}

```