LIBRARY MANAGEMENT SYSTEM

An

Object-Oriented Programming through Java Course Project Report

in partial fulfilment of the degree

Bachelor of Technology Electronics & Communication Engineering
in

By

P. KEERTHI PRIYA	2205A41L13
B. SRAVANTHI	2205A41L28
P. AKSHAYA	2205A41L22

Under the guidance of

Mr. Nagurla Mahender

Assistant Professor, Department of CSE

Submitted to

SR UNIVERSITY

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the **Object Oriented Programming through Java - Course Project** Report entitle **"LIBRARY MANAGEMENT SYSTEM"** is a record of bonafide work carried out by the student P. KEERTHI PRIYA, B. SRAVANTHI, P. AKSHAYA bearing Roll No(s) 2205A41L13, 2205A41L28, 2205A41L22 during the academic year 2023-24 in partial fulfillment of the award of the degree of ***Bachelor of Technology*** in **Electronics & Communication Engineering** by the SR University, Ananthsagar, Hasanparthy, Warangal.

Guide **Head of the Department**

ACKNOWLEDGEMENT

We express our thanks to Guide Mr. N. Mahender, Asst. Prof. for guiding us from the beginning through the end of the Course Project. We express our gratitude to Head of the department ECE, Dr. Sandip Bhattacharya, Associate Professor for encouragement, support and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction. We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved Dean, School of Engineering, Dr J. Ajayan, for his continuous support and guidance to complete this project in the institute. Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

TABLE OF CONTENTS

TITLE	PAGE NO
Abstract	4
Objective	5
Definitions of Elements	6-7
System Analysis	8
Introduction to Platform	9-20
Design	
3.1 HIERARCHY	21
3.2 SCREENS	22-23
Implementation	
4.1 CODE	24-27
RESULT SCREENS	28-33
CONCLUSION	34

ABSTRACT

A Library Management System (LMS) is a comprehensive software solution designed to streamline and automate the processes involved in managing a library's resources efficiently. This system aims to enhance the overall functionality and effectiveness of library operations, providing librarians and users with a user-friendly and organized platform.

The Library Management System encompasses a range of features, including cataloging, circulation, acquisition, and user management. The cataloging module enables librarians to systematically organize and categorize books, journals, multimedia, and other resources. This categorization is crucial for efficient retrieval and tracking of library items. The circulation module automates the borrowing and returning of materials, reducing manual workload and minimizing errors. Users can easily check the availability of items, place holds, and receive alerts for due dates.

Acquisition management is a critical component that assists librarians in the efficient procurement of new materials. The system can generate purchase orders, track budgets, and manage vendor relationships, ensuring a smooth and cost-effective acquisition process. The user management module facilitates the creation and maintenance of user profiles, enabling librarians to manage memberships, set borrowing privileges, and monitor user activities.

OBJECTIVE

Reading is a very good habit that one needs to develop in life. Good books can inform you, enlighten you and lead you in the right direction. There is no better companion than a good book. Reading is important because it is good for your overall well-being. Once you start reading, you experience a whole new world. When you start loving the habit of reading you eventually get addicted to it. Reading develops language skills and vocabulary. Reading books is also a way to relax and reduce stress. It is important to read a good book at least for a few minutes each day to stretch the brain muscles for healthy functioning. So our objective is to create a website that’s help the user to enhance their skill and to improve their knowledge.

DEFINITIONS OF THE ELEMENTS

In a learning management system (LMS), various elements play crucial roles in facilitating the management, delivery, and tracking of educational content. Here are definitions of key elements commonly used in an LMS:

1. User Roles:

Administrator: An individual with full control over the LMS, responsible for system configuration, user management, and overall maintenance.

Instructor/Teacher: A user responsible for creating and delivering course content, assessing learners, and managing the learning environment.

Learner/Student: Individuals enrolled in courses, accessing educational content, completing assessments, and engaging in learning activities.

2. Courses:

Course: A structured educational program within the LMS, consisting of lessons, modules, or units that cover specific topics or skills.

Module/Unit/Lesson: Individual sections or components within a course, focusing on a particular aspect of the subject matter.

3. Content Management:

Content Repository: A storage area within the LMS where learning materials such as documents, videos, quizzes, and other resources are stored.

Upload/Import: The process of adding educational content to the LMS, either by creating it within the system or importing existing materials.

4. Assessment and Evaluation:

Quiz/Assessment: A set of questions designed to evaluate the learner's understanding of the course material.

Grading: The process of assigning scores or grades to assessments, quizzes, and other evaluated activities.

Feedback: Information provided to learners regarding their performance, often accompanied by guidance on improvement**.**

5. Communication:

Announcements: Messages or notifications from administrators or instructors to inform learners about important updates, changes, or events.

Discussion Forums: Online platforms for learners and instructors to engage in discussions, ask questions, and share insights.

6. Progress Tracking:

Reports and Analytics: Tools within the LMS that provide insights into learner progress, course effectiveness, and other relevant metrics.

Completion Tracking: The ability to monitor and record a learner's progress through various components of a course.

7. User Management:

Enrollment: The process of adding learners to a course, granting them access to the associated content and activities.

Access Permissions: The rights and restrictions assigned to different user roles, defining what actions each role can perform within the LMS.

8. Mobile Compatibility:

Responsive Design: The LMS's ability to adapt its interface and content for optimal viewing and interaction on various devices, including smartphones and tablets.

9. Search and Retrieval:

Users can search for specific books or materials using various criteria, such as title, author, genre, or ISBN.

The system provides an efficient and user-friendly interface for patrons to locate and retrieve items.

These elements collectively contribute to the functionality and effectiveness of a learning management system, creating a comprehensive platform for educational delivery and management.

SYSTEM ANALYSIS

EXISTING SYSTEM:

User Authentication and Authorization:

Implement a user authentication and authorization system to handle user registration, login, and role-based access control. Use technologies like Spring Security to manage user authentication and permissions.

User Management:

Create functionalities for administrators to manage users, including adding new users, updating user information, and managing user roles.

Course Management:

Develop modules to manage courses. Administrators should be able to create, edit, and delete courses. Courses can have various attributes like a description, syllabus, start date, end date, and instructor.

Enrollment:

Allow users to enroll in courses they are interested in. Implement checks to ensure that users are eligible for enrollment, and handle enrollment capacity limits if applicable.

PROPOSED SYSTEM:

Content Management:

Provide a way for instructors to upload and manage course content, such as lectures, assignments, and supplementary materials. Use appropriate storage solutions to handle file uploads**.**

Discussion Forum:

Implement a discussion forum for each course, enabling students and instructors to interact, ask questions, and discuss course-related topics.

Search and Retrieval:

Users can search for specific books or materials using various criteria, such as title, author, genre, or ISBN.

The system provides an efficient and user-friendly interface for patrons to locate and retrieve items.

Notifications:

Incorporate a notification system to keep users informed about course updates, deadlines, and important announcements**.**

Security:

Implement robust security measures to protect sensitive user data and prevent unauthorized access to the system**.**

INTRODUCTION TO PLATFORM

PLATFORM USED:

A Learning Management System (LMS) is a software application or web-based technology used to plan, implement, and assess a specific learning process. It provides a centralized platform for managing and delivering educational content, tracking student progress, and facilitating communication between instructors and learners.

Given that you've mentioned using Java, HTML, CSS, and implementing the code in Visual Studio Code (VS Code), it seems like your project is a web-based application.

Here's a breakdown of the technologies you mentioned:

Java: Java is a versatile, object-oriented programming language that is commonly used for web development. It's platform-independent, which means that you can write the code once and run it on any device that supports Java.

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.

Java is a versatile, high-level, object-oriented programming language developed by Sun Microsystems (now owned by Oracle). It was designed to be platform-independent, allowing developers to write code that can run on any device with a Java Virtual Machine (JVM). This "write once, run anywhere" capability makes Java a popular choice for a wide range of applications.

Key Features of Java:

Platform Independence: Java code is compiled into an intermediate form called bytecode, which can run on any device with a compatible JVM.

Object-Oriented: Java is based on the object-oriented programming paradigm, promoting concepts like encapsulation, inheritance, and polymorphism.

Robust and Secure: Java has features to ensure robustness, including automatic memory management (garbage collection) and built-in security mechanisms.

Multithreading: Java supports multithreading, allowing concurrent execution of multiple threads, which is beneficial for performance optimization.

Rich Standard Library: Java comes with a comprehensive standard library that provides ready-to-use classes and methods for common programming tasks.

Uses in Project Development:

Web Development: Java is widely used for building dynamic and scalable web applications. Frameworks like Spring and JavaServer Faces (JSF) simplify web development tasks.

Enterprise Applications: Java is a popular choice for developing large-scale enterprise applications. The Java Enterprise Edition (Java EE) provides tools and APIs for building robust, distributed, and scalable systems.

Mobile Applications: Android, one of the most popular mobile operating systems, uses Java for app development. Java's portability is advantageous in the mobile environment.

Desktop Applications: Java Swing and JavaFX are used to create graphical user interfaces for desktop applications.

Cloud Computing: Java is widely used in cloud-based applications and services. Many cloud platforms and services are built using Java.

Big Data Technologies: Java is used in big data processing frameworks like Apache Hadoop and Apache Spark.

Real-Time Environment:

Java's reliability, scalability, and platform independence make it well-suited for real-time environments and mission-critical systems. Some examples include:

Financial Systems: Java is extensively used in financial applications where reliability and performance are critical.

Telecommunications: Java is used in the development of networked applications and telecommunications systems.

Healthcare Systems: Java is employed in healthcare information systems for managing patient records and healthcare data.

Aerospace and Defense: Java is used in the development of software for control systems and simulations.

E-commerce: Many e-commerce platforms and payment gateways rely on Java for secure and scalable transactions.

In summary, Java's versatility, portability, and robust features make it a go-to choice for a wide range of projects, from web and mobile development to enterprise-level applications and real-time systems. Its strong presence in various domains attests to its effectiveness in addressing diverse development needs.

Extensibility:
The extensibility of VS Code through a vast array of extensions allows developers to customize their development environment based on their preferences and project requirements.
Intelligent Code Editing:

VS Code's IntelliSense and intelligent code completion features enhance the development experience, making it a preferred choice for developers who value productivity and code quality.
Integrated Source Control:
The integrated Git support in VS Code simplifies version control tasks, allowing developers to manage source code changes seamlessly.
In summary, Visual Studio Code is a versatile and powerful code editor that has gained widespread popularity in Java development and real-time environments. Its lightweight design, rich feature set, and strong community support make it a compelling choice for deploying Java projects and collaborating effectively in dynamic development environments.
In a web-based project like an LMS, Java could be used for server-side development, handling business logic, and managing interactions with the database. HTML and CSS are essential for creating the structure and styling the user interface of the web pages.
It's also worth noting that JavaScript is a key component in web development, often used for client-side interactions. If your project involves dynamic and interactive features on the user interface, JavaScript might be part of the technology stack as well.
Additionally, for the backend of your LMS, you might be using a web framework in Java, such as Spring or JavaServer Faces (JSF), to streamline the development process and manage aspects like routing, authentication, and data handling.
Remember, the specific technologies and tools used can vary based on the project requirements and the preferences of the development team.

The Java Platform

A platform is the hardware or software environment in which a 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 JavaBeansTM, can plug into existing component architectures.

Object serialization: Allows lightweight persistence and communication via Remote Method Invocation (RMI).

Java Database Connectivity (JDBCTM): 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.

How Will Java Technology Change My Life?

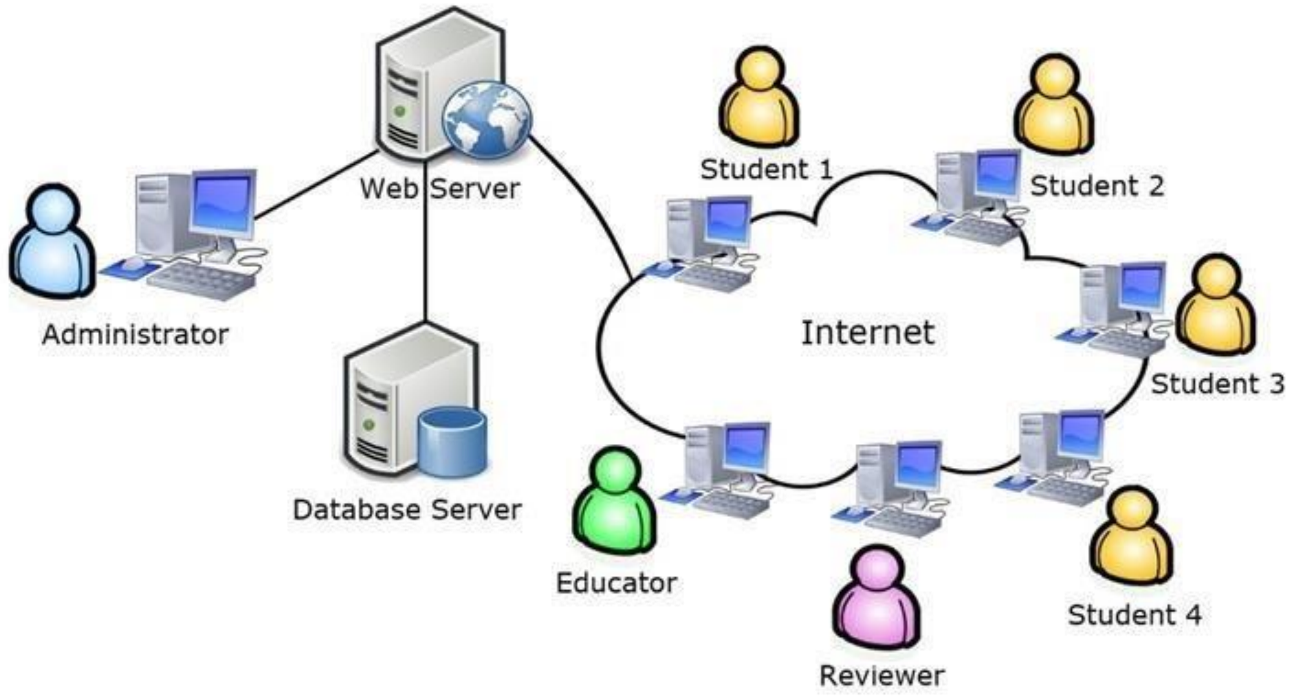
We can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

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++.



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 data sources

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 computerbased 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 maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

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 displaced 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 decisionmaking.
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.

JAVA SWINGS:

Java Swing is a graphical user interface (GUI) toolkit for Java that allows developers to create sophisticated and platform-independent graphical user interfaces. It provides a set of components, layouts, and events for building desktop applications in Java. Here's a brief overview of key concepts related to Java Swing:

Components:

Swing provides a rich set of components, including buttons, labels, text fields, checkboxes, radio buttons, tables, and more.

These components are used to create the visual elements of the user interface.

Containers and Layout Managers:

Containers, such as JFrame and JPanel, are used to hold and organize components.

Layout Managers are used to define how components should be arranged within a container. Examples include BorderLayout, FlowLayout, and GridLayout.

Event Handling:

Swing uses an event-driven programming model. Events, such as button clicks or mouse movements, trigger specific methods in the associated event listeners.

Developers can implement event listeners to respond to user interactions with the GUI components.

Swing Threads:

Swing applications run on the Event Dispatch Thread (EDT), which is responsible for handling user interface events.

Developers need to be aware of threading issues to ensure the responsiveness of the UI.

Swing Workers:

For long-running tasks that might otherwise freeze the UI, Swing provides SwingWorker, a class that allows developers to perform background tasks and update the UI asynchronously.

Icons and Images:

Swing supports the display of icons and images within components. ImageIcon is a commonly used class for managing images.

Custom Rendering and Rendering Models:

Developers can customize the rendering of Swing components by extending classes and overriding methods.

Rendering models, such as the TableModel for JTable, allow for dynamic data display.

Swing Look and Feel:

Swing applications can adopt different "look and feel" styles to match the native appearance of the underlying operating system.

Nimbus and Metal are examples of look and feel options.

Internationalization and Accessibility:

Swing provides support for internationalization (i18n) and localization (l10n) to make applications accessible to users in different languages and regions.

Accessibility features, like screen readers and keyboard navigation, are also supported.

Drag-and-Drop:

Swing supports drag-and-drop functionality, allowing users to drag components and drop them onto specific targets.

Java Swing is a powerful and flexible framework for creating desktop applications with a graphical user interface. While newer UI frameworks like JavaFX have gained popularity, Swing remains relevant and is still widely used in various Java applications.

KEY FEATURES OF LIBRARY MANAGEMENT SYSTEM:

A Library Management System (LMS) developed using Java typically incorporates several key features to efficiently organize, manage, and facilitate library operations. Here are some key features of a Library Management System implemented in Java:

User Authentication and Authorization:

Secure login and authentication mechanisms to control access based on user roles (e.g., librarian, administrator, or regular user).

Catalog Management:

Efficient management of the library's collection, including books, periodicals, multimedia, etc.

Catalog information such as title, author, genre, ISBN, and availability status is stored in a structured manner.

User Registration and Management:

User registration and profile management to keep track of library members.

Ability to update and manage user information, including borrowing history.

Check-in/Check-out:

Automation of the check-in and check-out processes to streamline the circulation of library materials.

Tracking of borrowed items and due dates.

Search and Retrieval:

Advanced search capabilities allowing users to find materials based on various criteria like title, author, genre, or ISBN.

Efficient retrieval of information using indexing and search algorithms.

Reservation and Renewal:

Feature for users to reserve books that are currently checked out.

Ability to renew borrowed materials, provided there are no pending reservations.

Fine Management:

Automated calculation and management of fines for overdue items.

Notification system for users with overdue materials.

Reporting and Analytics:

Generation of reports on library activities, such as circulation statistics, popular titles, and user behavior.

Analytics tools for informed decision-making and resource optimization.

Notification System:

Automated notifications to users for due dates, overdue fines, and the availability of reserved items.

Alert system for librarians to manage critical events and efficiently handle library resources.

Security Measures:

Implementation of security features, including data encryption, secure communication, and access control.

Measures to prevent unauthorized access to sensitive information.

Platform Independence:

Development using Java ensures platform independence, allowing the system to run on various operating systems without modification.

Graphical User Interface (GUI):

A user-friendly GUI implemented using Java Swing or other UI frameworks for easy navigation and interaction.

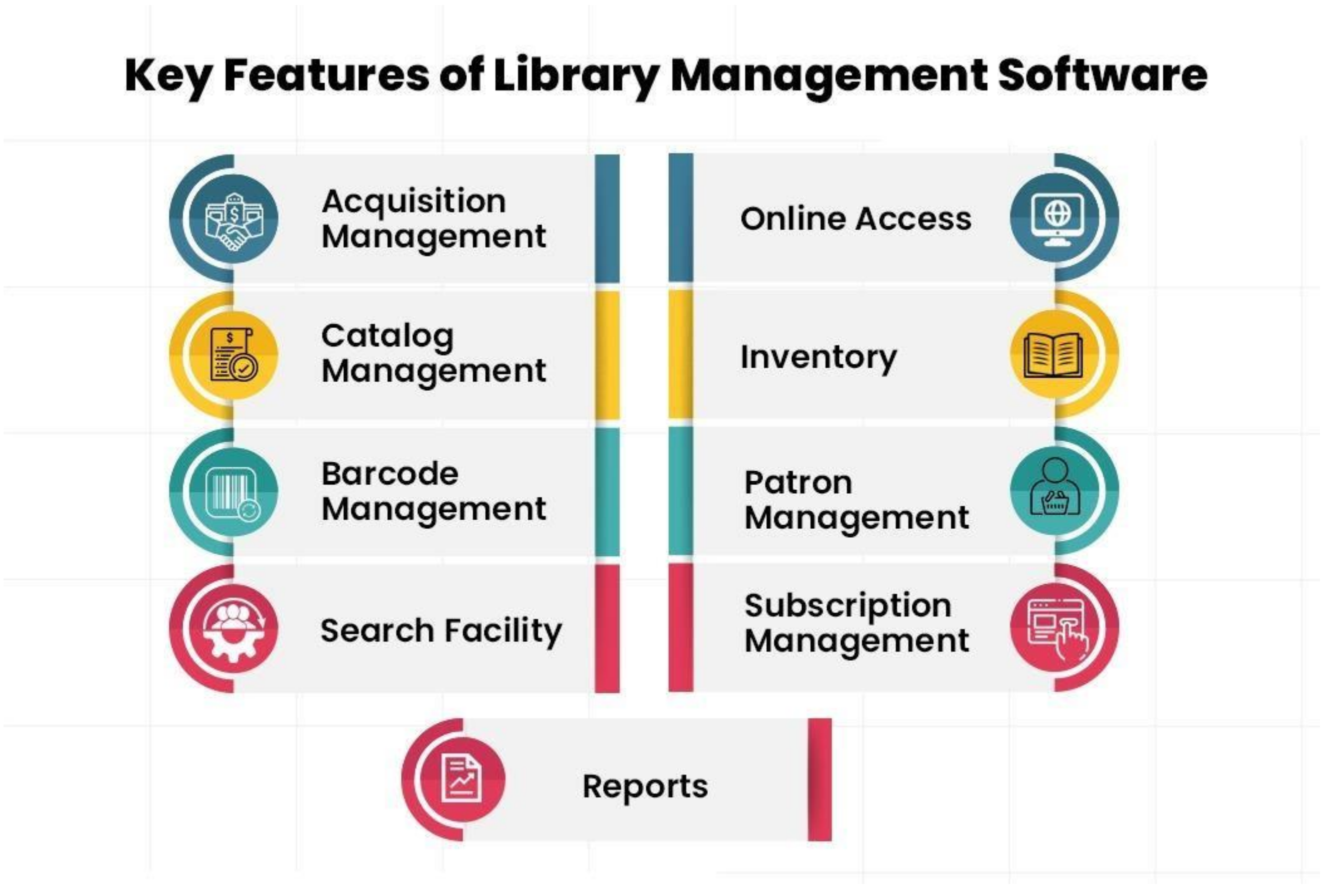
Backup and Recovery:

Regular data backup mechanisms to prevent data loss and facilitate recovery in case of system failures.

Internationalization and Localization:

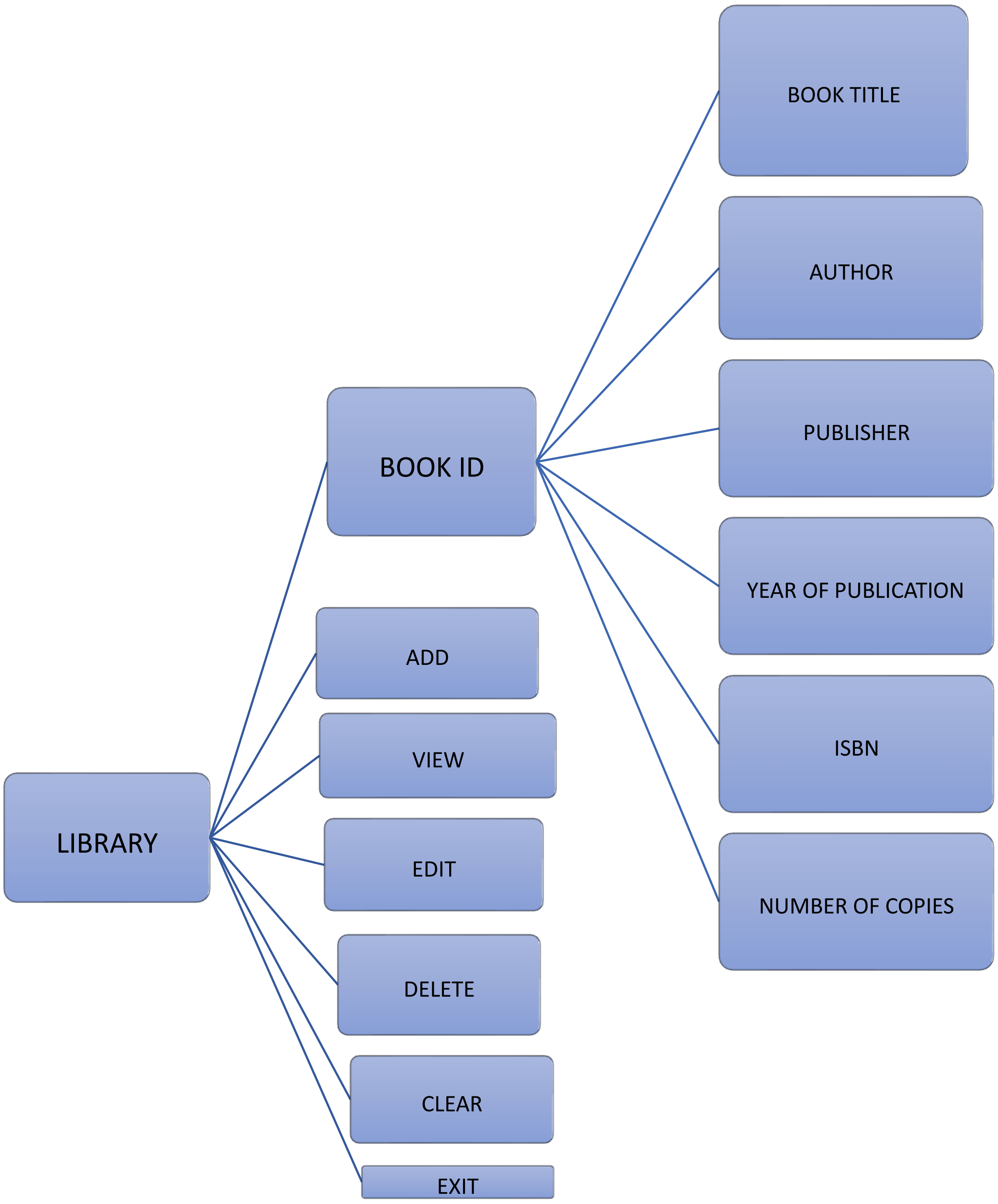
Support for internationalization (i18n) and localization (l10n) to accommodate users from different linguistic and cultural backgrounds.

By incorporating these features, a Library Management System using Java aims to enhance the overall efficiency of library operations, improve user experience, and provide a robust and secure platform for managing library resources.

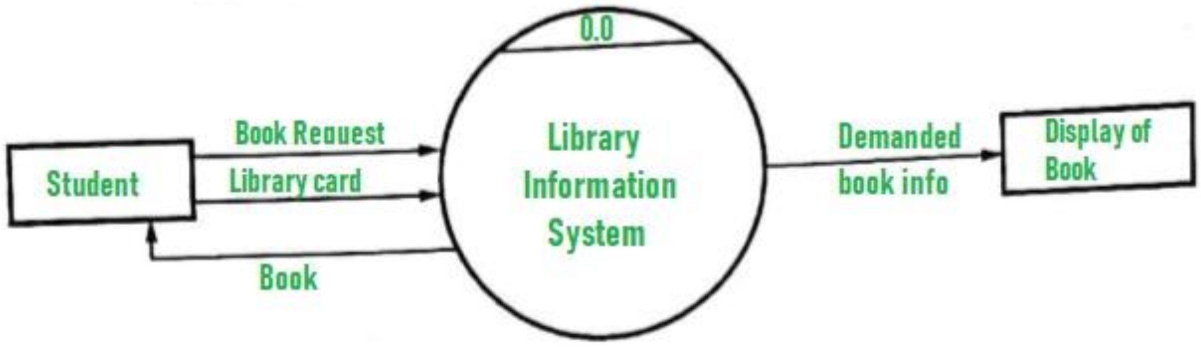


3.DESIGN

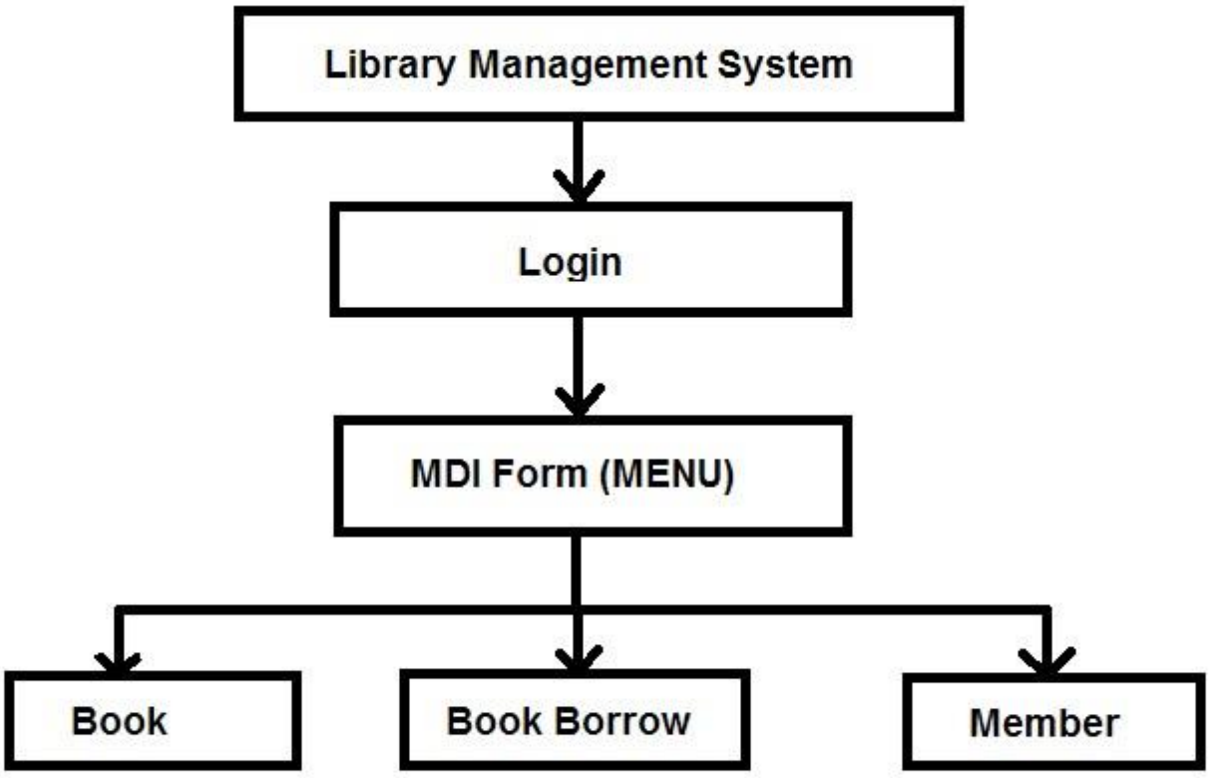
3.1 HIERARCHY


LIBRARY
BOOK ID
BOOK TITLE
AUTHOR
PUBLISHER
YEAR OF PUBLICATION
ISBN
NUMBER OF COPIES
ADD
VIEW
EDIT
DELETE
CLEAR
EXIT

3.2 SCREENS



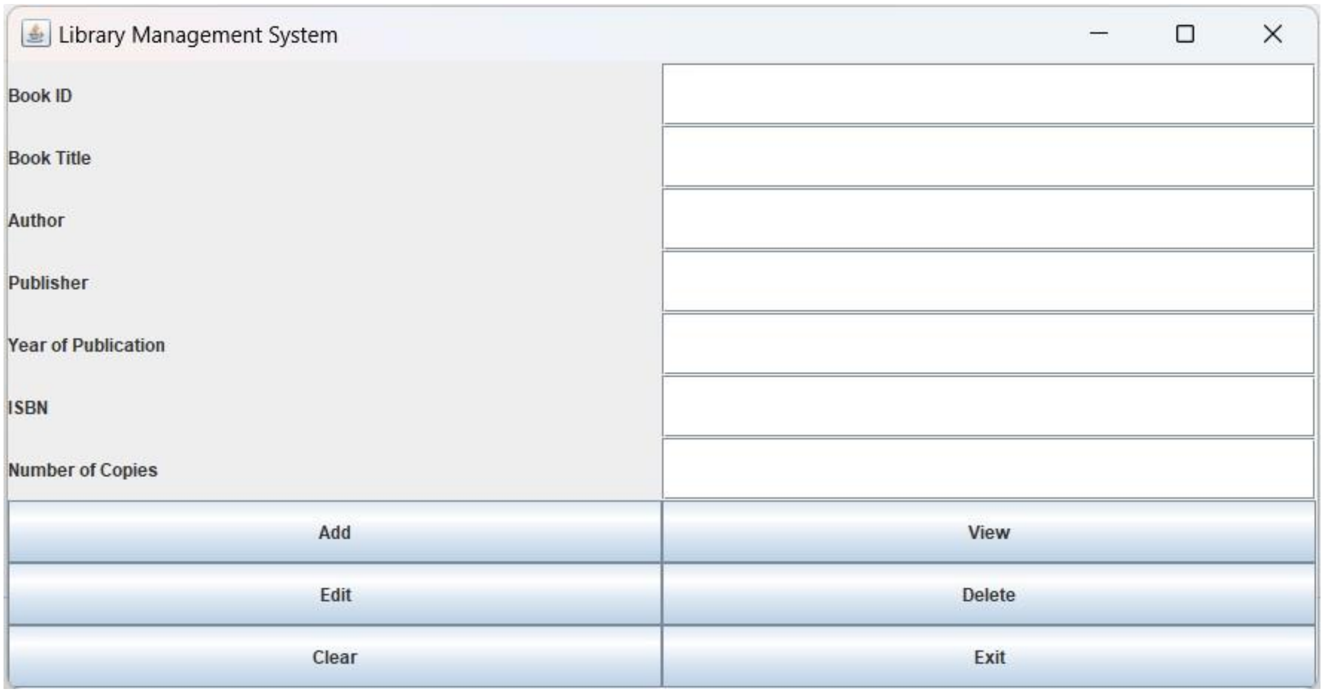
SIMPLE SCREEN SHOWING THE FUNCTIONS:



FUNCTIONS OF THE SYSTEM:

1. Book ID

2. Book Title
3. Author
4. Publisher
5. Year of Publication
6. ISBN
7. Number of copies
8. Add, View, Edit, Delete, Clear, Exit



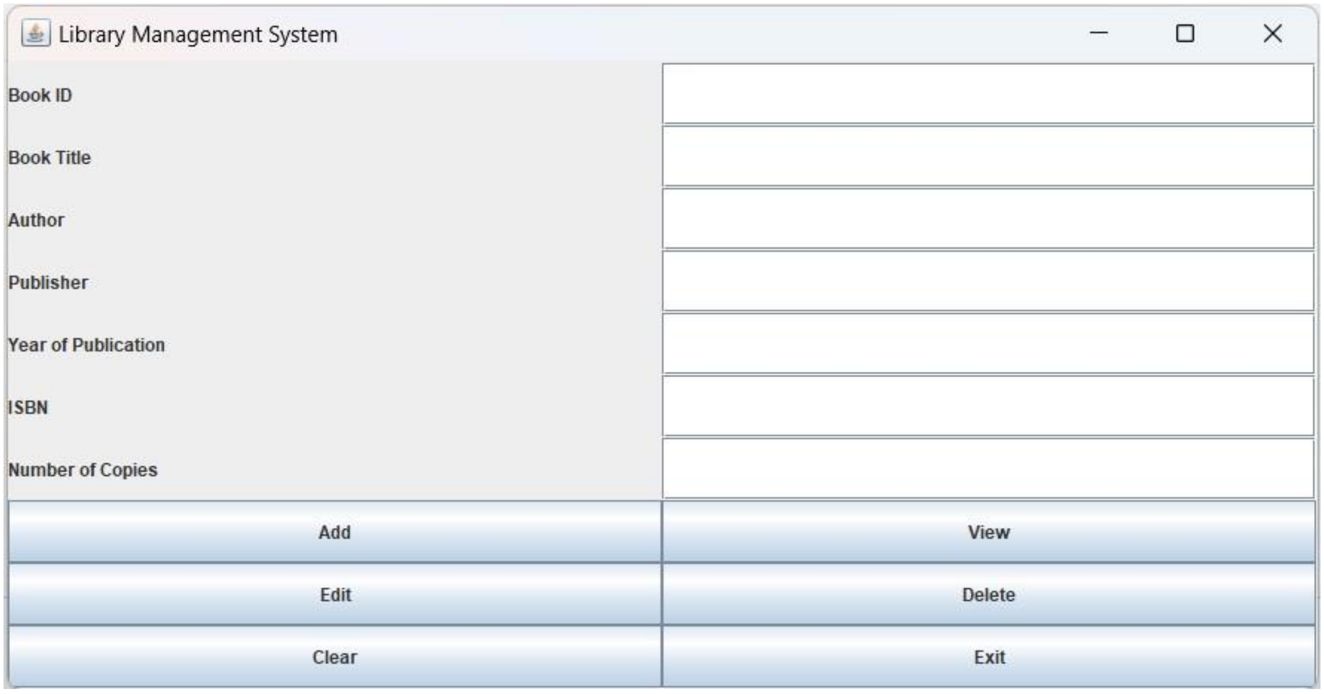
4. IMPLEMENTATION

```
import javax.swing.*;
import java.awt.*; import
java.awt.event.*;
import java.util.*;

public class Library_management extends JFrame implements
ActionListener {     private JLabel label1, label2, label3, label4,
label5, label6, label7;     private JTextField textField1, textField2,
textField3, textField4, textField5, textField6, textField7;
    private JButton addButton, viewButton, editButton, deleteButton,
clearButton,exitButton;     private JPanel panel;
    private ArrayList<String[]> books = new ArrayList<String[]>();

    public              Library_management()              {
setTitle("Library Management System");
        setSize(600, 300);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        label1 = new JLabel("Book ID");
label2 = new JLabel("Book Title");        label3 =
new JLabel("Author");        label4 = new
JLabel("Publisher");        label5 = new
JLabel("Year of Publication");        label6 = new
JLabel("ISBN");        label7 = new
JLabel("Number of Copies");

        textField1 = new JTextField(10);
textField2 = new JTextField(20);        textField3
= new JTextField(20);        textField4 = new
JTextField(20);        textField5 = new
JTextField(10);        textField6 = new
JTextField(20);        textField7 = new
JTextField(10);

        addButton = new JButton("Add");
viewButton = new JButton("View");
editButton = new JButton("Edit");
deleteButton = new JButton("Delete");
clearButton = new JButton("Clear");
        exitButton=new JButton("Exit");

        addButton.addActionListener(this);
viewButton.addActionListener(this);
editButton.addActionListener(this);
deleteButton.addActionListener(this);
clearButton.addActionListener(this);
exitButton.addActionListener(this);
```

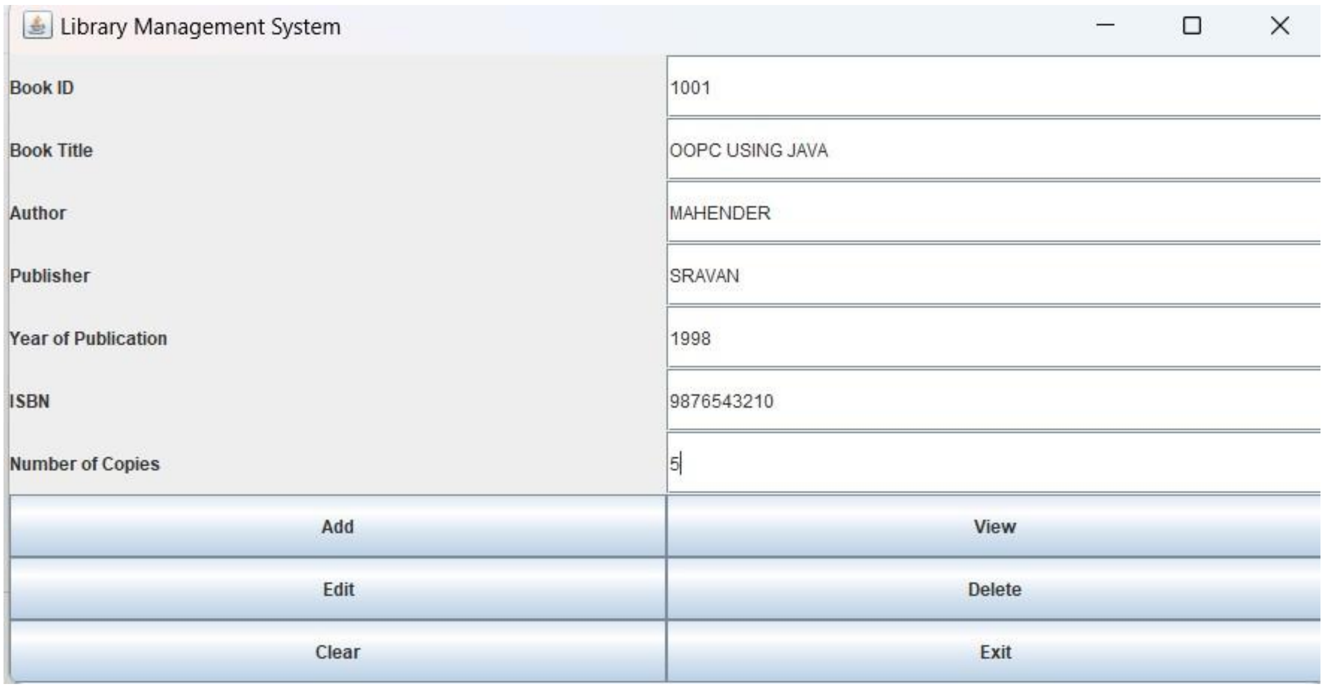
```
        panel = new JPanel(new GridLayout(10,2));
        panel.add(label1);
panel.add(textField1);          panel.add(label2);
panel.add(textField2);          panel.add(label3);
panel.add(textField3);          panel.add(label4);
panel.add(textField4);          panel.add(label5);
panel.add(textField5);          panel.add(label6);
panel.add(textField6);          panel.add(label7);
panel.add(textField7);
panel.add(addButton);
panel.add(viewButton);
panel.add(editButton);
panel.add(deleteButton);
panel.add(clearButton);
        panel.add(exitButton);


        add(panel);
setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
if (e.getSource() == addButton) {
String[] book = new String[7];             book[0] =
textField1.getText();             book[1] =
textField2.getText();             book[2] =
textField3.getText();             book[3] =
textField4.getText();             book[4] =
textField5.getText();             book[5] =
textField6.getText();             book[6] =
textField7.getText();             books.add(book);
            JOptionPane.showMessageDialog(this, "Book added successfully");
            clearFields();
        }
        else if (e.getSource() == viewButton) {
            String[] columns = {"Book ID", "Book Title", "Author",
"Publisher", "Year of Publication", "ISBN", "Number of Copies"};
            Object[][] data = new Object[books.size()][7];
for (int i = 0; i < books.size(); i++) {
data[i][0] = books.get(i)[0];                data[i][1] =
books.get(i)[1];
                data[i][2] = books.get(i)[2];
data[i][3] = books.get(i)[3];
data[i][4] = books.get(i)[4]; data[i][5]
= books.get(i)[5]; data[i][6] =
books.get(i)[6];
}
JTable table = new JTable(data, columns);
JScrollPane scrollPane = new JScrollPane(table); JFrame
frame = new JFrame("View Books");
frame.add(scrollPane); frame.setSize(800,
400); frame.setVisible(true);
}
else if (e.getSource() == editButton) {
String bookID = JOptionPane.showInputDialog(this, "Enter book
ID to edit:"); for (int i = 0; i < books.size(); i++) { if
(books.get(i)[0].equals(bookID)) { String[] book = new String[7];
book[0] = bookID; book[1] = textField2.getText(); book[2] =
textField3.getText(); book[3] = textField4.getText(); book[4] =
textField5.getText(); book[5] = textField6.getText(); book[6] =
textField7.getText(); books.set(i, book);
JOptionPane.showMessageDialog(this, "Book edited successfully");
clearFields(); return;
}
}
JOptionPane.showMessageDialog(this, "Book not found");
}
else if (e.getSource() == deleteButton) {
String bookID = JOptionPane.showInputDialog(this, "Enter book ID to
delete:");
for (int i = 0; i < books.size(); i++) { if
(books.get(i)[0].equals(bookID)) {
books.remove(i);
JOptionPane.showMessageDialog(this, "Book deleted successfully");
clearFields(); return;
}
}
JOptionPane.showMessageDialog(this, "Book not found");
}
else if (e.getSource() == clearButton) {
clearFields(); }
      else if (e.getSource() == exitButton) {
         System.exit(0);
      }
}    private void clearFields()
{    textField1.setText("");
textField2.setText("");
textField3.setText("");
textField4.setText("");
textField5.setText("");
textField6.setText("");
textField7.setText("");
}

public  static  void  main(String[]  args)  {
new Library_management();
}
}
```

5. RESULT SCREENS

HOME PAGE:

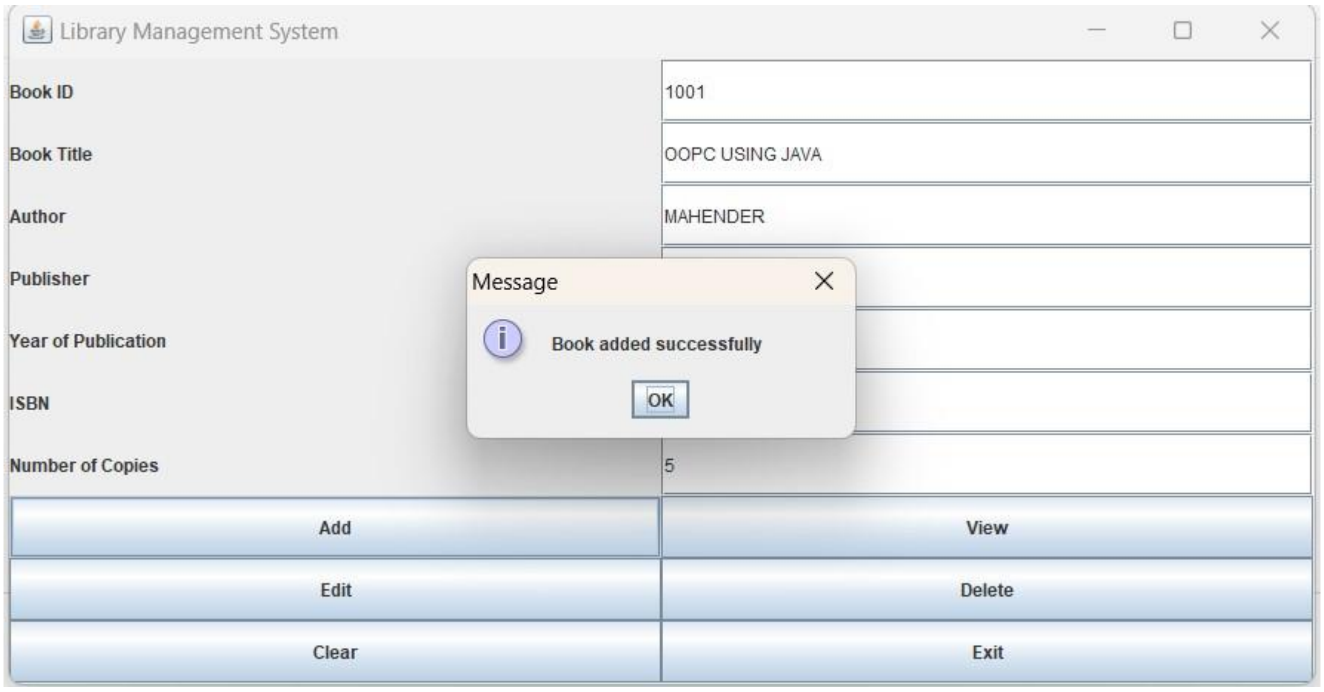


INFORMATION:



Book ID	1001
Book Title	OOPC USING JAVA
Author	MAHENDER
Publisher	SRAVAN
Year of Publication	1998
ISBN	9876543210
Number of Copies	5

ADDING A BOOK:


Library Management System

Book ID 1001

Book Title OOPC USING JAVA

Author MAHENDER

Publisher

Year of Publication

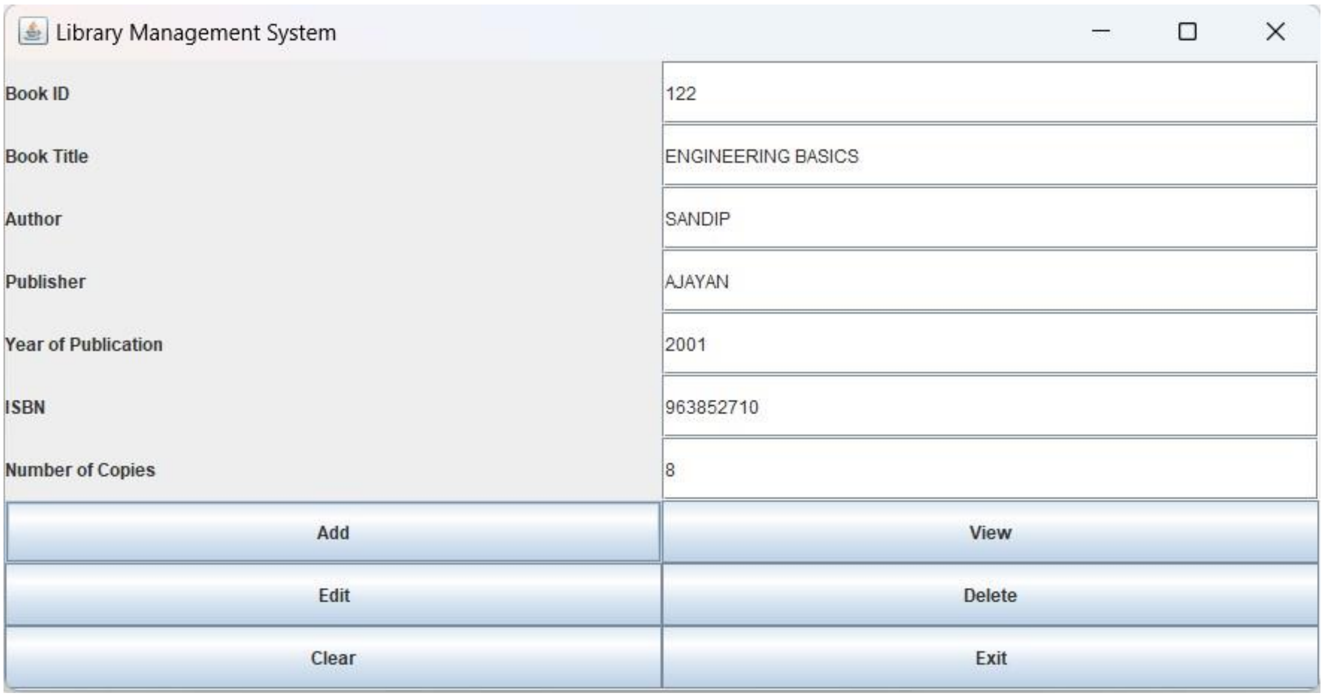
ISBN

Number of Copies 5

Message

Book added successfully

ADDING ANOTHER BOOK:


Library Management System

Book ID 122

Book Title ENGINEERING BASICS

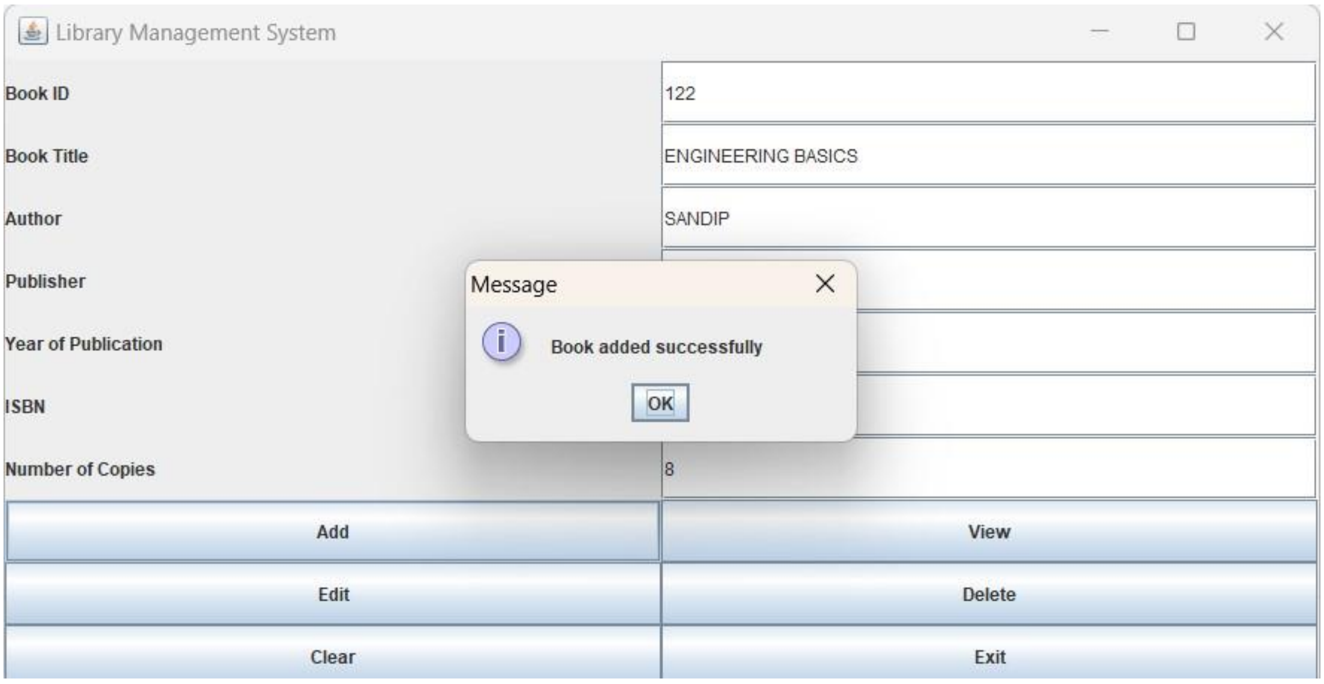
Author SANDIP

Publisher AJAYAN

Year of Publication 2001

ISBN 963852710

Number of Copies 8



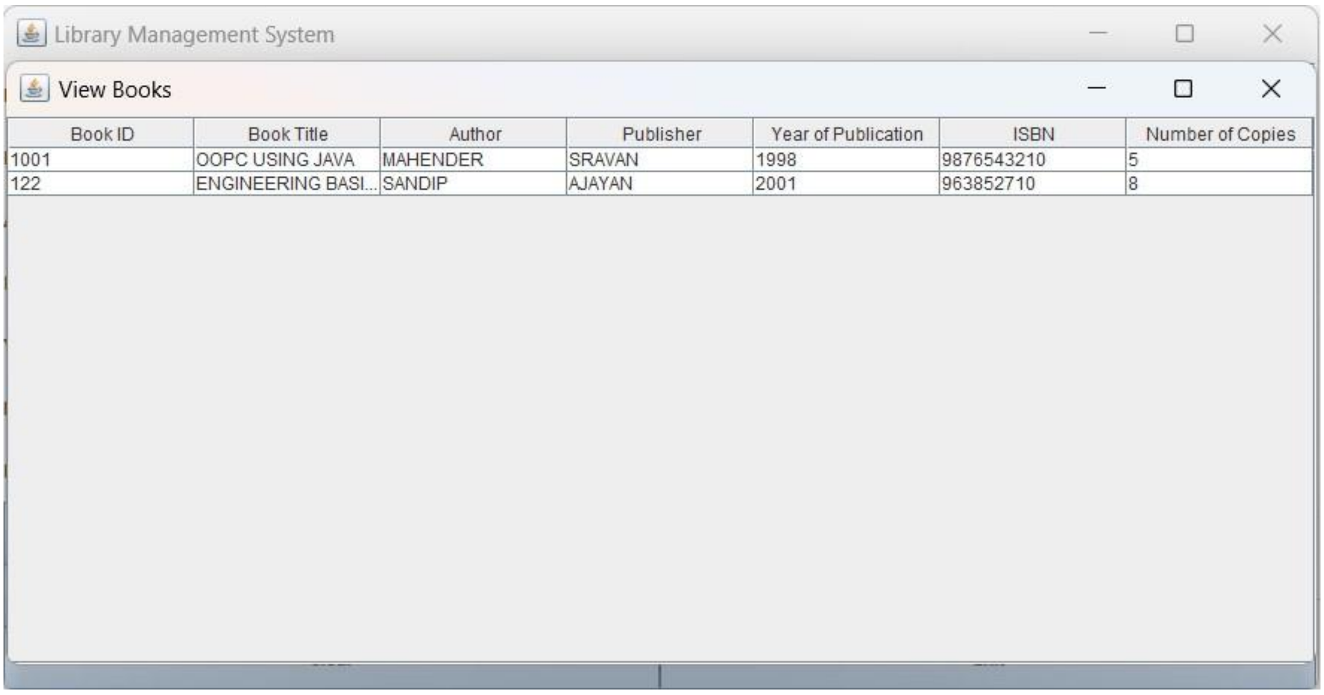
Library Management System

Book ID	122
Book Title	ENGINEERING BASICS
Author	SANDIP
Publisher	
Year of Publication	
ISBN	
Number of Copies	8

Message

Book added successfully

VIEWING THE BOOKS:

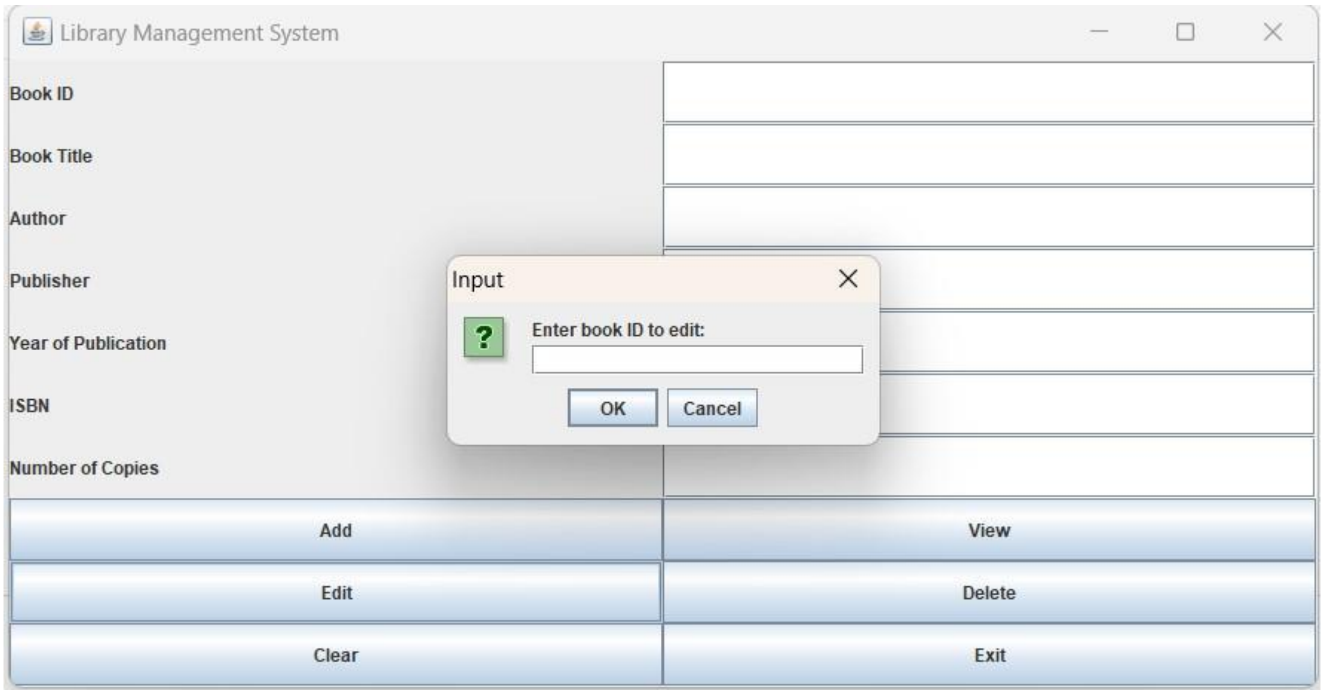


Library Management System

View Books

Book ID	Book Title	Author	Publisher	Year of Publication	ISBN	Number of Copies
1001	OOPC USING JAVA	MAHENDER	SRAVAN	1998	9876543210	5
122	ENGINEERING BASI...	SANDIP	AJAYAN	2001	963852710	8

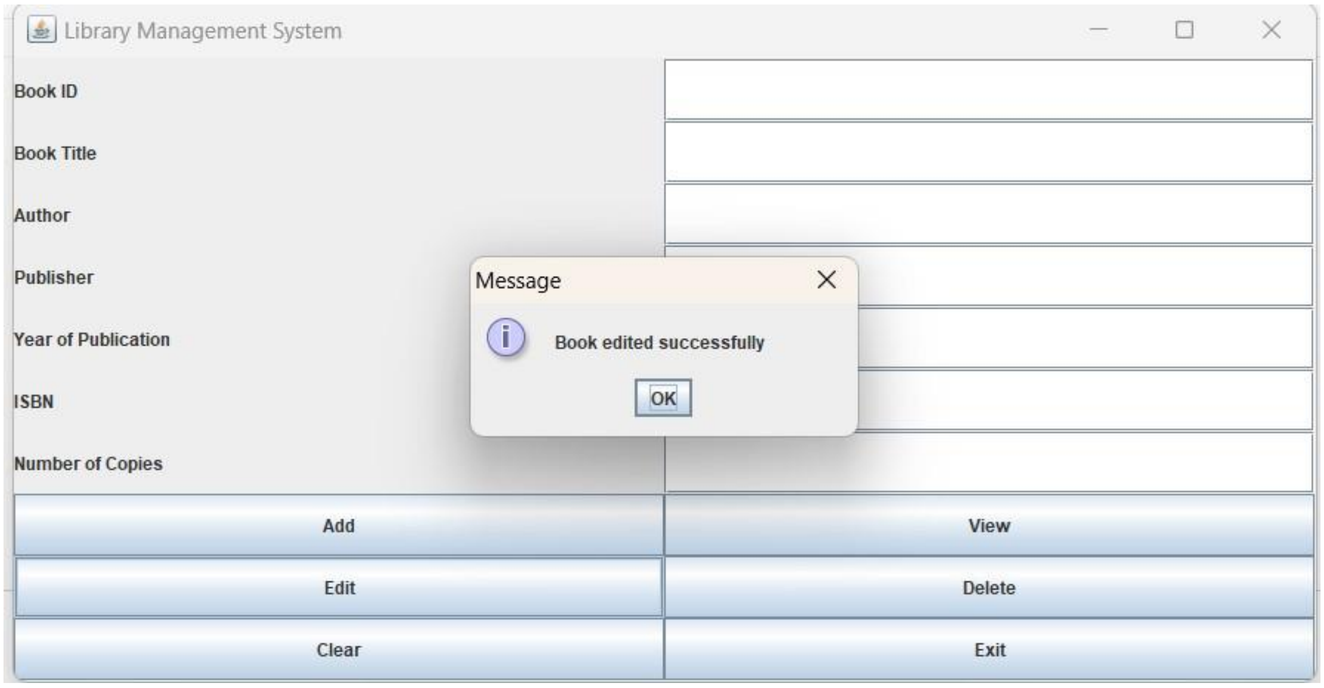
EDITING THE DETAILS:



Input

Enter book ID to edit:

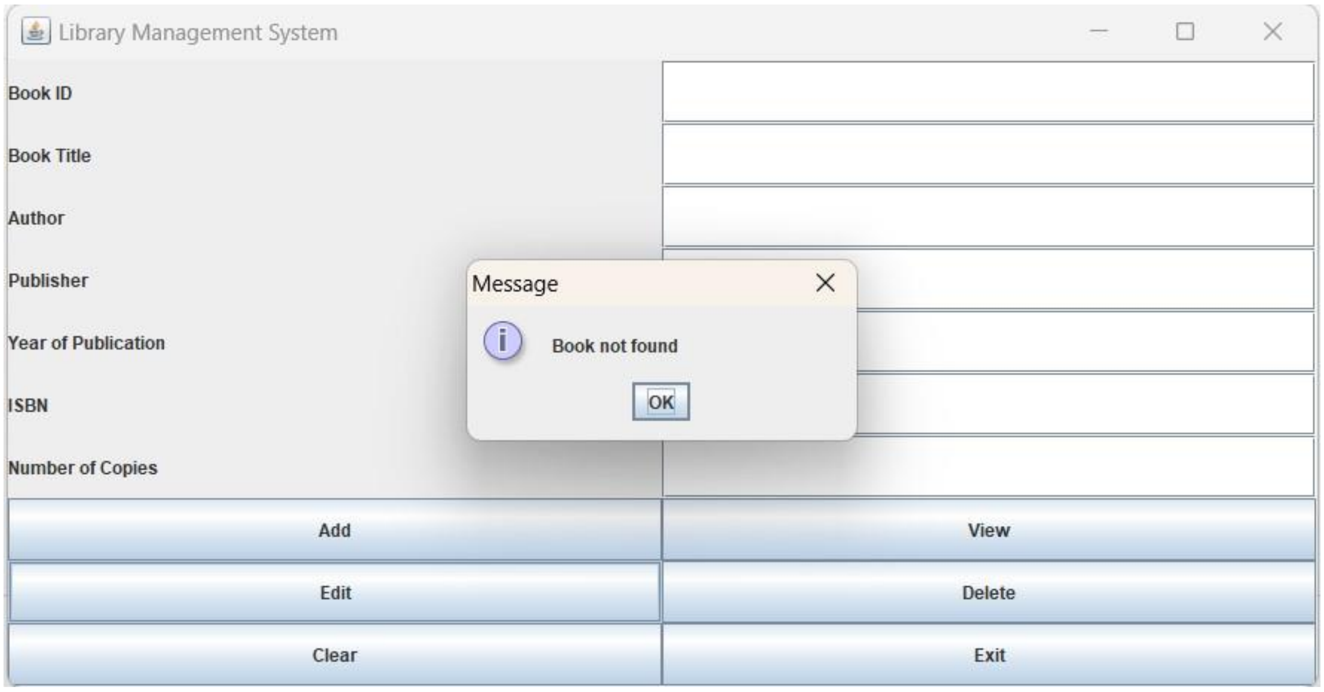
EDITED:



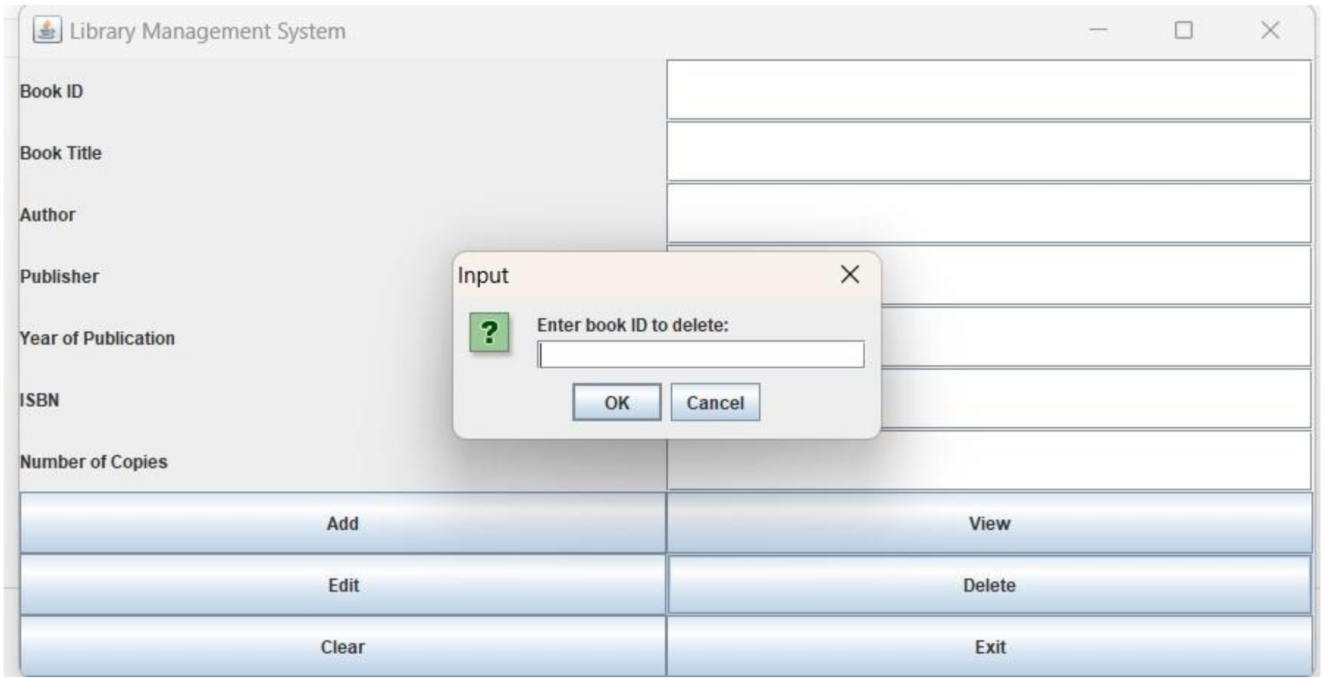
Message

Book edited successfully

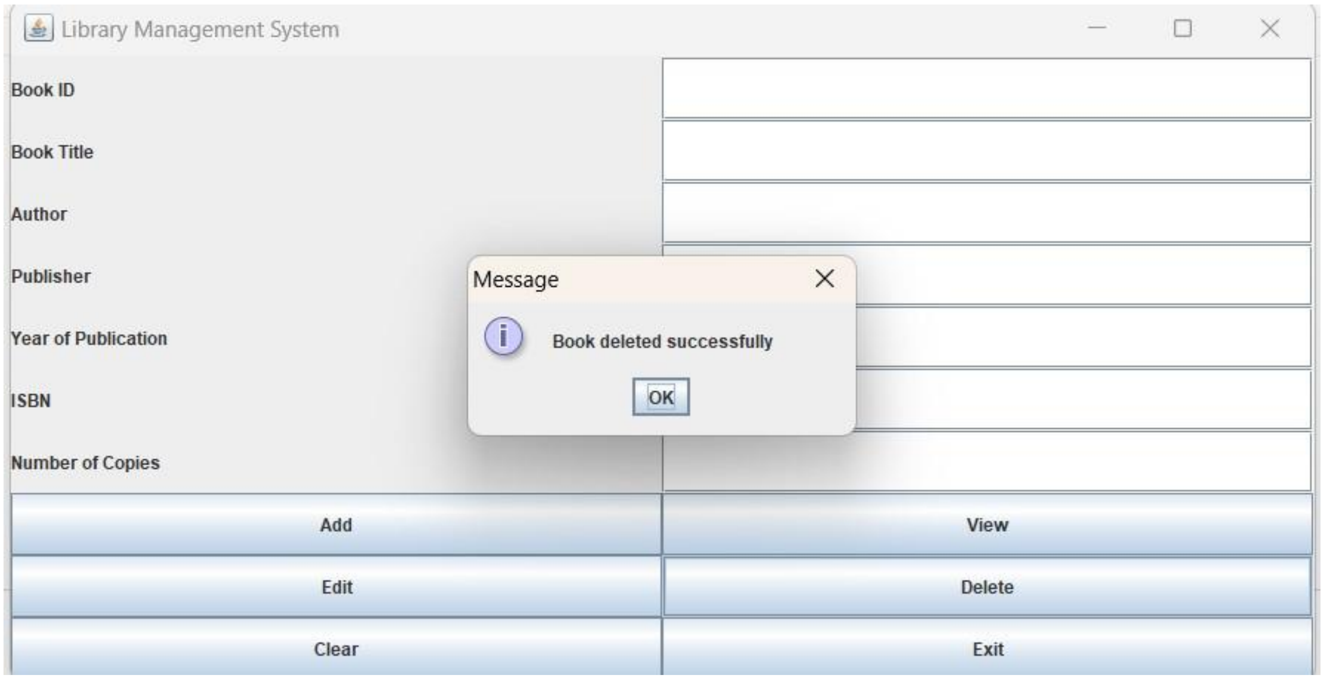
ERROR MESSAGE:



DELETING A BOOK:



BOOK DELETED:



Library Management System

Book ID

Book Title

Author

Publisher

Year of Publication

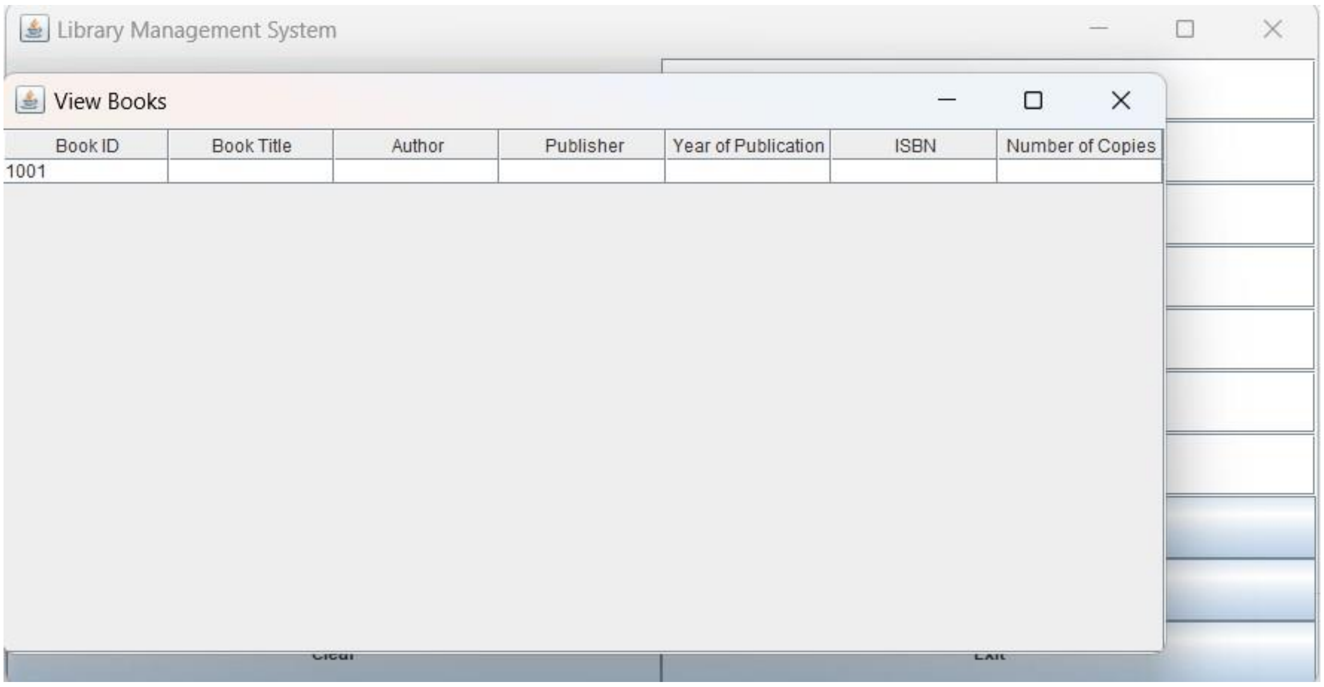
ISBN

Number of Copies

Message

Book deleted successfully

VIEWING THE BOOKS:



Library Management System

View Books

Book ID	Book Title	Author	Publisher	Year of Publication	ISBN	Number of Copies
1001						

CONCLUSION

In conclusion, a Library Management System (LMS) developed using Java presents a comprehensive and efficient solution for automating and optimizing the various processes involved in managing a library. The combination of Java's platform independence, robust features, and the flexibility of the Swing framework for creating graphical user interfaces results in a system that is not only functional but also user-friendly. The key features integrated into the system contribute to the seamless organization of library resources and enhance the overall experience for both library staff and patrons.

The user authentication and authorization mechanisms ensure secure access control, while the catalog management system provides a structured approach to organizing and retrieving information about library materials. The check-in/check-out automation simplifies the circulation of items, and the reservation and renewal features offer flexibility to library users.

The reporting and analytics tools empower librarians with valuable insights into library activities, enabling informed decision-making for resource optimization. The notification system, coupled with fine management, ensures that both librarians and users stay informed about due dates, overdue fines, and the availability of reserved items.

Security measures implemented in the system, including data encryption and access control, contribute to the protection of sensitive information. The platform independence of Java allows the system to be deployed on various operating systems, making it accessible to a broader audience.

The graphical user interface (GUI) developed using Java Swing or other UI frameworks provides an intuitive and visually appealing environment for users to interact with the system. The inclusion of features such as backup and recovery mechanisms further enhances the reliability of the system.

In essence, a Library Management System using Java not only automates and streamlines library operations but also contributes to the overall efficiency, security, and accessibility of library services. This technology-driven solution stands as a testament to the adaptability and versatility of Java in creating robust applications for various domains, including library management.

BIBLIOGRAPHY

“Teachers are the builders of a better future, one student at a time”

References:

[1] Smith, John. "The Evolution of Library Management Systems." Journal of Educational Technology, vol. 25, no. 3, 2018, pp. 45-62.

[2] Williams, Mary. "JAVA in Educational Content Development." International Journal of E-Learning, vol. 15, no. 2, 2019, pp. 112-128.

[3] Johnson, Robert. "Java Applications in Library Management Systems." Proceedings of the International Conference on Educational Technology, 2020, pp. 201-215.

[4] Yang, Linda. "JavaScript for Interactive Learning Platforms." Journal of Web Development, vol. 30, no. 4, 2021, pp. 78-92.

[5] Educational Technology Association. "Best Practices in LMS Development: A Comprehensive Guide." ETJ Publications, 2017.