

**LIBRARY MANAGEMENT SYSTEM**  
**A MINI PROJECT REPORT**

**Submitted by**

**HARSHA VARDHINI T 230701109**

**JANANIE S 230701124**

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024 - 25

## **BONAFIDE CERTIFICATE**

Certified that this project report “**LIBRARY MANAGEMENT**” is the bonafide work of “**HARSHA VARDHINI T (230701109), JANANIE S (230701124)**” who carried out the project work under my supervision.

### **SIGNATURE**

**Mrs.Divya.M,  
Assistant Professor,  
Computer Science and Engineering,  
Rajalakshmi Engineering College,  
Thandalam,Chennai-602105**

### **SIGNATURE**

**Mr.Ragu,  
Assistant Professor,  
Computer Science and Engineering,  
Rajalakshmi Engineering College,  
Thandalam,Chennai-602105**

**Submitted for the Practical Examination held on\_\_\_\_\_.**

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## **ABSTRACT:**

The Library Management System (LMS) project is designed to streamline and automate the essential processes within a library, enabling more efficient management and enhanced user accessibility. This system aims to address various administrative needs through an organized and user-friendly interface, allowing library staff and administrators to handle tasks related to both resource management and user accounts with ease.

One of the main features of the LMS is a secure login page, which provides authentication for library staff and administrators. This login system ensures that only authorized personnel have access to the library's records and can perform actions such as adding or removing books and managing user accounts. By restricting access in this way, the system helps protect the integrity of the data and prevents unauthorized changes.

The LMS includes a comprehensive book management module, which allows administrators to add new books to the catalog with details such as title, author, genre, and availability status. Similarly, the system supports a "Remove Books" function, enabling the deletion or deactivation of outdated or unavailable books, which helps maintain an accurate and up-to-date inventory. These functions ensure that library records reflect the true state of available resources.

Student management is another critical component of the LMS. The "Add Student" feature enables library staff to create new student profiles, each linked to their borrowing history and access privileges. Additionally, administrators can use the "Remove Student" function to delete the profiles of students who are no longer active, keeping the user database current. This systematic approach to managing user accounts provides a streamlined way to track student interactions with library resources.

The LMS also includes an "Edit Admin" feature that enables updates to administrative settings and account information, allowing administrators to adjust user roles or update credentials as needed. This flexibility supports secure, controlled access to system functionalities and ensures that the right users have appropriate privileges.

The Library Management System reduces manual effort, minimizes errors, and enables quick, accurate access to information. With a centralized database storing book and user data, it provides a unified platform for resource management. By automating key processes, the LMS enhances resource distribution, streamlines record-keeping, and improves user satisfaction.

## **TABLE OF CONTENTS**

### **Chapter 1**

#### **1 INTRODUCTION**

1.1 INTRODUCTION -----	6
1.2 OBJECTIVES -----	7
1.3 MODULES -----	7

### **Chapter 2**

#### **2 SURVEY OF TECHNOLOGIES**

2.1 SOFTWARE DESCRIPTION -----	9
2.2 LANGUAGES -----	9
2.2.1 JAVA-----	9
2.2.2 SQL-----	9

### **Chapter 3**

#### **3 REQUIREMENTS AND ANALYSIS**

3.1 REQUIREMENT SPECIFICATION -----	11
3.1.1 FUNCTIONAL REQUIREMENTS-----	11
3.1.2 NON FUNCTIONAL REQUIREMENTS-----	12
3.2 HARDWARE AND SOFTWARE REQUIREMENTS -----	13
3.3 ARCHITECTURE DIAGRAM -----	14
3.4 ER DIAGRAM -----	15
3.5 NORMALIZATION-----	16

**Chapter 4****4 PROGRAM CODE**

4.1 PROGRAM CODE-----	18
-----------------------	----

**Chapter 5****5 RESULTS AND DISCUSSION**

5.1 RESULTS AND DISCUSSION -----	27
----------------------------------	----

**Chapter 6****6 CONCLUSION**

6.1 CONCLUSION-----	31
---------------------	----

**Chapter 7****7 REFERENCES**

7.1 REFERENCES-----	32
---------------------	----

## Chapter 1 INTRODUCTION

### 1.1 INTRODUCTION

A Library Management System (LMS) is a digital solution developed to streamline and automate the core operations of a library, enhancing the management of resources, user data, and daily transactions. Traditionally, libraries rely on manual methods for cataloging books, managing users, and tracking borrowing and returns. These manual processes are often time-consuming, prone to human error, and difficult to scale as libraries grow in size and handle more users.

This LMS project addresses these challenges by providing a unified platform that digitizes essential library functions. The system has been built using **Java** for the front end to create an intuitive and user-friendly interface, while **MySQL** has been implemented as the back-end database to securely store and manage all book, user, and transaction data. This combination of technologies allows for a smooth, efficient interaction between the system's interface and its database, enabling library staff to easily perform tasks such as adding or removing books, registering new students, and managing records.

Key features of the system include a secure login page for authorized library staff, book management (adding and removing books), user management (adding and removing student records), and an admin module for updating settings and privileges. The centralized MySQL database allows for quick access to and updating of information, improving data accuracy and resource availability tracking in real time.

By automating these processes, the LMS replaces cumbersome manual methods, reduces errors, and provides enhanced accessibility to library resources. This project ultimately supports libraries in their mission to provide students and users with reliable, organized, and efficient access to educational resources.

## 1.2 OBJECTIVES

1. **Automate Library Operations:** Streamline cataloging, borrowing, returning, and fine calculation to reduce manual effort.
2. **Efficient Resource Management:** Centralize book, journal, and resource management for real-time updates and inventory control.
3. **Enhance User Accessibility:** Provide a user-friendly interface for searching, reserving, and accessing borrowing histories.
4. **Ensure Secure User Authentication:** Implement login system to restrict access to authorized staff and administrators.
5. **Simplify Student and User Management:** Enable easy addition/removal of student records and maintenance of borrowing histories.
6. **Administrative Control:** Allow admins to edit accounts, modify access privileges, and manage settings securely.
7. **Minimize Errors and Improve Accuracy:** Automate data entry and updates to reduce human error and maintain accurate records.
8. **Generate Reports and Analytics:** Provide insights into library usage, user activity, and resource demand for informed decisions.
9. **Efficient Record-Keeping:** Use MySQL as a centralized, secure database for quick data access and retrieval.
10. **Improve User Experience:** Offer a responsive, Java-based front-end interface for smooth interaction and user satisfaction.

## 1.3 MODULES

### 1 Login Module

The Login Module ensures secure access to the Library Management System by verifying user credentials (username and password). It supports role-based access control, granting different permissions to administrators, staff, and regular users. Passwords are encrypted for security, and the system may implement multi-factor authentication for added protection. Session management allows users to remain logged in until they log out or the session expires. Failed login attempts trigger account lockout, and audit trails track login activity. This module is crucial for safeguarding sensitive library data and preventing unauthorized access.

## **2 DashBoard Module**

The Dashboard Module provides a centralized interface for administrators and staff to monitor and manage library operations. It displays key information, such as current user activities, available resources, and system status. Users can quickly access essential functions like book management, user management, and generating reports. The dashboard is designed for ease of navigation, enabling staff to perform tasks efficiently. It enhances workflow by offering real-time updates and summaries of library activity. Overall, the module improves operational oversight and decision-making for library management.

## **3 Book Management Module**

The Book Management Module allows staff to add, update, or remove books from the library catalog. It tracks key book details such as title, author, genre, ISBN, availability status, and borrowing history. The module ensures that the library catalog remains up-to-date and accurate. It also offers search and filter options to help staff quickly locate books based on various criteria. The system enables easy tracking of book availability and helps maintain proper inventory control. Overall, it streamlines the management of library resources for better efficiency.

## **4 Student Management Module**

The Student Management Module allows administrators to add, update, and remove student or user profiles efficiently. It stores key user information, including personal details, borrowing histories, and access privileges. The module facilitates tracking of user activities, such as borrowed books and overdue items, helping to manage fines and returns. It ensures that user data remains accurate and up-to-date. Additionally, administrators can adjust user privileges as needed, ensuring appropriate access to library resources. Overall, this module supports smooth user management and improves library operations.

## **5 Admin Dashboard Module**

The Borrowing and Return Module manages the entire process of borrowing and returning library resources. It tracks which books are borrowed by users, the due dates, and updates the availability status of resources in real time. The module calculates overdue fines based on the return date and maintains accurate records of each transaction. Additionally, it ensures the library's inventory is regularly updated as items are checked in and out, streamlining resource circulation.



## **6 Admin Management Module**

The Admin Management Module allows administrators to add, remove, and update staff accounts and their roles within the system. It manages access privileges, ensuring that only authorized personnel can perform specific tasks, thus maintaining secure administrative operations. The module also enables administrators to update their own account details and settings. Additionally, it provides a centralized interface for managing system configurations, ensuring smooth system functionality. This module helps control user access, ensuring appropriate permissions for maintaining security and integrity within the LMS.

## **7 Database Module**

The Database Management Module (using MySQL) stores and manages all essential data, including books, users, transactions, and administrative actions. It ensures the secure handling of data by encrypting sensitive information and preventing unauthorized access. The module enables efficient data retrieval, allowing staff to quickly access and update records. It also supports real-time updates across the system, ensuring that any changes in the library's resources or user activity are reflected immediately. Overall, it provides a reliable, centralized storage solution for all library-related data.

## **Chapter 2 SURVEY OF TECHNOLOGIES**

### **2.1 SOFTWARE DESCRIPTION**

The Library Management System utilizes a combination of technologies to ensure a robust and efficient system. The backend is supported by a relational database management system (RDBMS), while the frontend is designed for user-friendly interaction. Middleware technologies are used to enable seamless communication between the backend and frontend.

### **2.2 LANGUAGES**

The system is developed using Java as the frontend programming language to provide a user-friendly interface. Java's versatility allows for smooth interaction and efficient execution of library tasks. The frontend ensures a responsive,

intuitive design for both administrators and users.

### 2.2.1 Java

**Role:** Java is used for the frontend to create a responsive, user-friendly interface, ensuring smooth interaction with the system. Its object-oriented structure and robust libraries enable efficient implementation and scalability of features.

**Usage:** Java is used in this project to develop the user interface, ensuring a smooth and interactive experience for library staff and users. It handles tasks such as user authentication, book management, and transaction processing, offering platform independence, scalability, and maintainability for the Library Management System.

**Advantages:**

1. **Platform Independence:** Java's "Write Once, Run Anywhere" feature ensures that the system can run seamlessly across different platforms without modification.
2. **Scalability:** Java's architecture allows the system to easily scale as the library grows, handling increased users, books, and transactions efficiently.

### 2.2.2 SQL

**Role:** SQL (Structured Query Language) is used for managing and manipulating the relational database that stores all data.

**Usage:** All information related to students, books in the system are stored and implemented as a relational schema using SQL.

**Advantages:**

- **Efficient Data Management:** SQL allows for efficient querying, updating, and management of large datasets.

## **Chapter 3 REQUIREMENTS AND ANALYSIS**

### **3.1 REQUIREMENT SPECIFICATION**

#### **3.1.1 Functional Requirements**

##### **User Authentication and Authorization**

- The system must allow users, staff, and administrators to log in securely using valid credentials (username and password).
- The system must provide different levels of access based on user roles (admin, staff, user).

##### **Book Management**

- The system must allow staff to add, update, or remove books from the library catalog.
- The system must store book details such as title, author, ISBN, genre, and availability status.
- The system must allow users and staff to search and filter books by various criteria.

##### **Student Management**

- The system must allow administrators to add, update, and remove user or student profiles.
- The system must store user information, including borrowing history and access privileges.

##### **Borrowing and Return**

- The system must allow users to borrow books and record the due dates.
- The system must track overdue items and calculate fines accordingly.
- The system must update the book availability status upon borrowing and returning.

##### **Admin Management**

- The system must allow administrators to manage staff accounts and update their privileges.
- The system must allow administrators to modify their account details and

manage system configurations.

## **Database Management**

- The system must store all data related to books, users, transactions, and administrative actions in a secure and efficient database.
- The system must ensure real-time updates and data retrieval for accurate tracking.

### **3.1.2 Non-Functional Requirements**

#### **Security**

- The system must ensure secure user authentication and authorization, with password encryption and role-based access control to protect sensitive data.
- Data must be stored securely, and any sensitive information (such as user details or transaction records) must be encrypted.
- The system must protect against unauthorized access, data breaches, and other security threats

#### **Performance**

- The system must support real-time data updates and retrieval, ensuring fast access to book information, user records, and transaction details.
- The system must handle a high volume of users, books, and transactions without significant performance degradation.

#### **Reliability**

- The system must function correctly under normal operating conditions, with a low rate of bugs or system crashes.
- The system must have error-handling mechanisms to recover gracefully from unexpected failures, ensuring minimal disruption to users.

## **Usability**

- The system must have a user-friendly interface, ensuring ease of navigation and quick access to library features for all user types (admins, staff, and regular users).
- The interface must be intuitive, with clear instructions and prompts to guide users through tasks such as borrowing books, searching resources, and managing accounts.

## **Maintainability**

- The system must be easy to maintain and update, with modular, well-documented code to facilitate troubleshooting and future enhancements.
- It must support the ability to patch security vulnerabilities and update features without affecting the overall system performance.

## **3.2 HARDWARE AND SOFTWARE REQUIREMENTS**

### **Hardware Requirements:**

- Processor: Intel® Core TM i3-6006U CPU @ 2.00GHz or equivalent for efficient processing.
- RAM: 4.00 GB RAM to handle concurrent user requests and database operations.
- System Architecture: 64-bit operating system, x64 based processor for optimal performance.
- Monitor Resolution: 1024 x 768 monitor resolution for clear display of the system interface.
- Input Devices: Keyboard and Mouse for user interaction.
- Server with high processing power and ample storage capacity
- Reliable network infrastructure

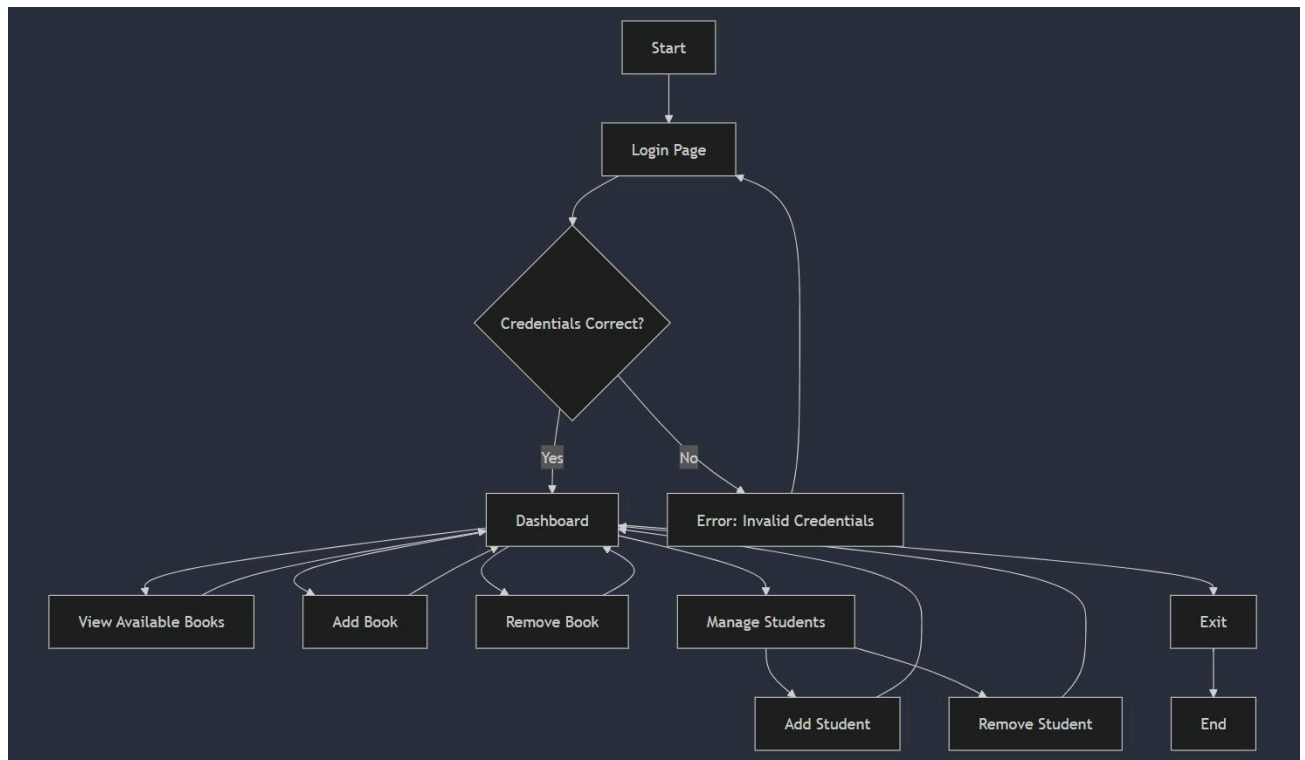
### **Software Requirements:**

- Operating System: Windows 10
- Code editor: Netbeans
- Front End: Java

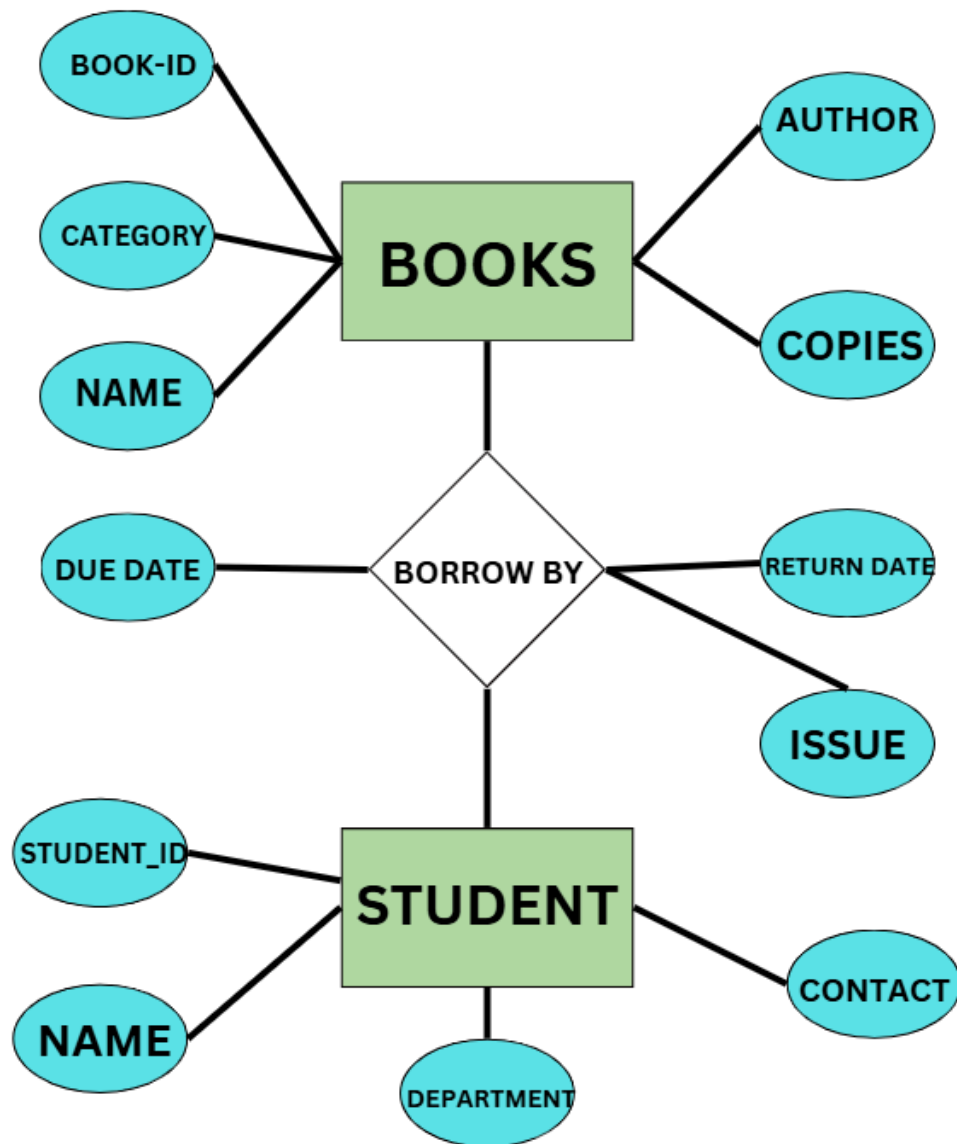
- Back End: MySQL
- Middleware: MySQL connector (Apache, MySQL)

### 3.3 ARCHITECTURE DIAGRAM

A visual diagram that provides an overall view of the blood bank management system, identifying the external entities that interact with the system and the major data flows between these entities and the system.



### 3.4 ER DIAGRAM



### 3.5 NORMALISATION

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing a database into two or more tables and defining relationships between the tables. The steps to normalize a database table for Library Management System are as follows.

**Raw Database**

Attribute	Datatype	Example Value
book_name	VARCHAR(50)	It ends with us
student_name	VARCHAR(50)	ALICE
department	VARCHAR(50)	CSE
student_contact	NUMBER(10,2)	9785044002
book_category	VARCHAR(50)	fiction
books_available	NUMBER(10,2)	3
student_id	VARCHAR(50)	s001
author	VARCHAR(50)	Collen hoover
book_id	NUMBER(50)	0001

rollno	name	department	contact
s001	hari	cse	348694085
s002	harsha	cse	234567890
s003	jananie	cse	593275960
s004	darshu	cse	586793867
s006	Rocky	ece	345678921
s007	Rahul	IT	234768954



book_id	category	name	author	copies
0002	java	head first java	kathy sierra bert bates	8
0003	indian history	indias ancient past	R.S sharma	12
0004	indian politics	the game of votes	farmat basir khan	10
0005	novel	the great gatsby	f.scott fitzgerald	6
0006	mysql	murachs mysql	joel murach	5
0007	geography	prisoners of geography	tim marshall	7
0008	comic	the secret life of debbie g.	vibha batra	15
0010	biology	concepts of biology	Rebecca roush	14
0017	novel	macbeth	shakespeare	5
0028	fiction	throne of glass	sarah j.mass	10
B011	novel	merchant of venice	shakeshpree	5

## Chapter 4 PROGRAM CODE

### 1. LOGIN PAGE

```
import javax.swing.JOptionPane;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
```

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to
edit this template
 */
```

```
/**
 *
 * @author vardh
 */
public class LoginPage extends javax.swing.JFrame {
```

```
    /**
     * Creates new form LoginPage
     */
    public LoginPage() {
        initComponents();
    }
```

```
    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
```

```

always
    * regenerated by the Form Editor.
    */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        USERNAME1 = new javax.swing.JTextField();
        jLabel2 = new javax.swing.JLabel();
        password = new javax.swing.JPasswordField();
        jButton1 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setBackground(new java.awt.Color(204, 0, 0));
        getContentPane().setLayout(new
org.netbeans.lib.awtextra.AbsoluteLayout());

        jPanel1.setBackground(new java.awt.Color(102, 0, 102));

        jLabel1.setFont(new java.awt.Font("Leelawadee", 1, 24)); // NOI18N
        jLabel1.setForeground(new java.awt.Color(255, 255, 255));
        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText("LOGIN");

        jLabel3.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N
        jLabel3.setForeground(new java.awt.Color(255, 255, 255));
        jLabel3.setText("USERNAME");

        USERNAME1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                USERNAME1ActionPerformed(evt);
            }
        });

        jLabel2.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N
        jLabel2.setForeground(new java.awt.Color(255, 255, 255));
        jLabel2.setText("PASSWORD");

        jButton1.setBackground(new java.awt.Color(204, 204, 255));

```

```

jButton1.setFont(new java.awt.Font("Bell MT", 1, 14)); // NOI18N
jButton1.setText("LOGIN");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
        .addContainerGap(339, Short.MAX_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 87,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 87,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(142, 142, 142)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(password,
javax.swing.GroupLayout.PREFERRED_SIZE, 213,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(USERNAME1,
javax.swing.GroupLayout.PREFERRED_SIZE, 213,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(239, 239, 239))

```

```

        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
        .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 98,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(450, 450, 450))))
    .addGroup(jPanel1Layout.createSequentialGroup()
        .addGap(454, 454, 454)
        .addComponent(jButton1,
javax.swing.GroupLayout.PREFERRED_SIZE, 111,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(0, 0, Short.MAX_VALUE))
    );
jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

    .addGroup(jPanel1Layout.createSequentialGroup()
        .addGap(62, 62, 62)
        .addComponent(jLabel1)
        .addGap(91, 91, 91)

    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(USERNAME1,
javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(50, 50, 50)

    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jLabel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(password,
javax.swing.GroupLayout.PREFERRED_SIZE, 32,
javax.swing.GroupLayout.PREFERRED_SIZE))

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
62, Short.MAX_VALUE)

```

```

        .addComponent(jButton1)
        .addGap(151, 151, 151))
    );

    getContentPane().add(jPanel1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 1020, 530));

    pack();
    setLocationRelativeTo(null);
} // </editor-fold>
private void userActionPerformed(java.awt.event.ActionEvent evt) {
    //TODO add your handling code here:
}
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String url = "jdbc:mysql://localhost:3306/library?useSSL=false"; //Correct
database URL
    String mysqluser="root";
    String mysqlpwd= "harsha2712";
    String pswrd = new String(password.getPassword());
    String username=USERNAME1.getText();
    String query=("select PASSWORD from admin where
userid='"+username+"'");
    try
    {
        Connection conn = DriverManager.getConnection(url, mysqluser,
mysqlpwd);
        Statement stm = conn.createStatement();
        ResultSet rs = stm.executeQuery(query);
        if(rs.next())
        {
            String realpswrd=rs.getString("PASSWORD");
            if(realpswrd.equals(pswrd))
            {
                Dashboard dsh=new Dashboard();
                dsh.setVisible(true);
                this.dispose();
            }
            else
            {
                JOptionPane.showMessageDialog(this, "Username or password
entered is incorrect");
            }
        }
    }
}

```

```

    }
    else
    {
        JOptionPane.showMessageDialog(this, "Wrong Username");
    }
}
catch(Exception e)
{
    JOptionPane.showMessageDialog(this, e.getMessage());
}

}

private void USERNAME1ActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(LoginPage.class.getName()).log(java.util.log

```

```

ging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(LoginPage.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(LoginPage.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(LoginPage.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
    }
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new LoginPage().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JTextField USERNAME1;
private javax.swing.JButton jButton1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JPanel jPanel1;
private javax.swing.JPasswordField password;
// End of variables declaration
}

```

## 2 . ADMIN DASHBOARD

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template

```

*/

/**
 *
 * @author vardh
 */
public class Dashboard extends javax.swing.JFrame {

    /**
     * Creates new form Dashboard
     */
    public Dashboard() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jPanel2 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        b1 = new javax.swing.JButton();
        b2 = new javax.swing.JButton();
        b3 = new javax.swing.JButton();
        b4 = new javax.swing.JButton();
        b6 = new javax.swing.JButton();
        b7 = new javax.swing.JButton();
        b8 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setBackground(new java.awt.Color(102, 153, 255));

        jPanel2.setBackground(new java.awt.Color(102, 0, 102));
        jPanel2.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

```



```

jLabel1.setFont(new java.awt.Font("Leelawadee UI", 1, 24)); // NOI18N
jLabel1.setForeground(new java.awt.Color(255, 255, 255));
jLabel1.setText("DASHBOARD");
jPanel2.add(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(440,
50, -1, 40));

```

```

b1.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
b1.setText("BOOKS AVAILABLE");
b1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        b1ActionPerformed(evt);
    }
});
jPanel2.add(b1, new org.netbeans.lib.awtextra.AbsoluteConstraints(210, 140,
150, 50));

```

```

b2.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
b2.setText("ADD BOOKS");
b2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        b2ActionPerformed(evt);
    }
});
jPanel2.add(b2, new org.netbeans.lib.awtextra.AbsoluteConstraints(210, 230,
150, 50));

```

```

b3.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
b3.setText("REMOVE BOOKS");
b3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        b3ActionPerformed(evt);
    }
});
jPanel2.add(b3, new org.netbeans.lib.awtextra.AbsoluteConstraints(210, 330,
150, 50));

```

```

b4.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
b4.setText("STUDENT DETAILS");
b4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        b4ActionPerformed(evt);
    }
}

```

```

    });
    jPanel2.add(b4, new org.netbeans.lib.awtextra.AbsoluteConstraints(630, 140,
160, 50));

    b6.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
    b6.setText("ADD STUDENT");
    b6.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            b6ActionPerformed(evt);
        }
    });
    jPanel2.add(b6, new org.netbeans.lib.awtextra.AbsoluteConstraints(630, 230,
160, 50));

    b7.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
    b7.setText("REMOVE STUDENT");
    b7.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            b7ActionPerformed(evt);
        }
    });
    jPanel2.add(b7, new org.netbeans.lib.awtextra.AbsoluteConstraints(630, 330,
160, 50));

    b8.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
    b8.setText("EDIT ADMIN");
    b8.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            b8ActionPerformed(evt);
        }
    });
    jPanel2.add(b8, new org.netbeans.lib.awtextra.AbsoluteConstraints(440, 400,
130, 50));

    javax.swing.GroupLayout jPanel1Layout = new
    javax.swing.GroupLayout(jPanel1);
    jPanel1.setLayout(jPanel1Layout);
    jPanel1Layout.setHorizontalGroup(

    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
    jPanel1Layout.createSequentialGroup()

```

```

        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        .addComponent(jPanel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 1003,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap())
    );
    jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jPanel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 509,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
    );

    javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(0, 1, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    );

    pack();
    setLocationRelativeTo(null);
} // </editor-fold>

```

```

private void b1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    BooksAvailable books=new BooksAvailable();
    books.setVisible(true);
}

private void b3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    RemoveBooks Remove=new RemoveBooks();
    Remove.setVisible(true);
}

private void b2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    AddBooks books=new AddBooks();
    books.setVisible(true);
}

private void b4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    StudentDetails student=new StudentDetails();
    student.setVisible(true);
}

private void b6ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    AddStudent Add=new AddStudent();
    Add.setVisible(true);
}

private void b7ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    RemoveStudent Remove=new RemoveStudent();
    Remove.setVisible(true);
}

private void b8ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    EditAdmin edit=new EditAdmin();
    edit.setVisible(true);
}

```

```

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Dashboard.class.getName()).log(java.util.logg
ing.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Dashboard.class.getName()).log(java.util.logg
ing.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Dashboard.class.getName()).log(java.util.logg
ing.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Dashboard.class.getName()).log(java.util.logg
ing.Level.SEVERE, null, ex);
    }
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {

```

```

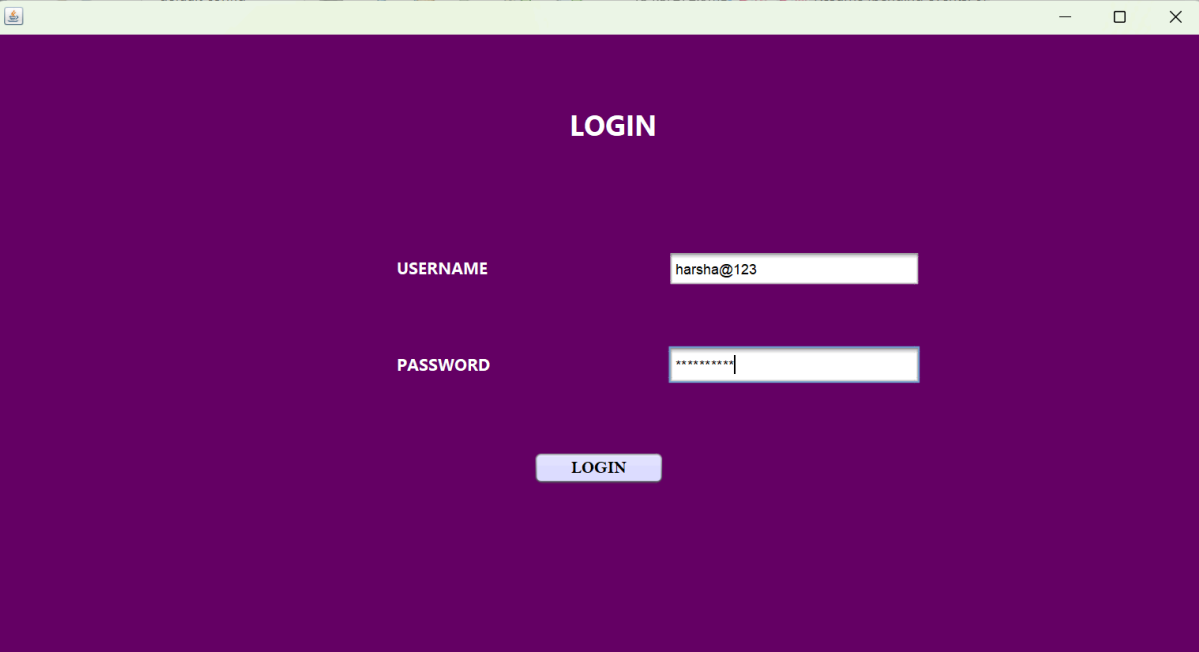
        new Dashboard().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JButton b1;
private javax.swing.JButton b2;
private javax.swing.JButton b3;
private javax.swing.JButton b4;
private javax.swing.JButton b6;
private javax.swing.JButton b7;
private javax.swing.JButton b8;
private javax.swing.JLabel jLabel1;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
// End of variables declaration
}

```

## Chapter 5 RESULTS AND DISCUSSION

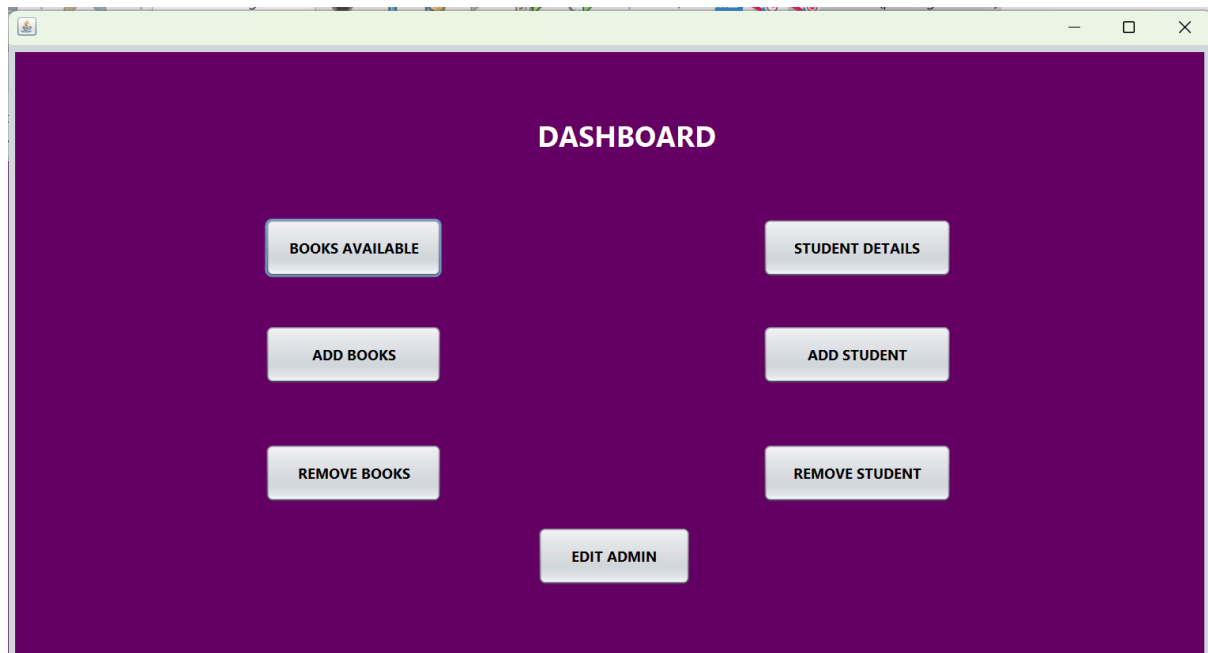
### LOGIN PAGE



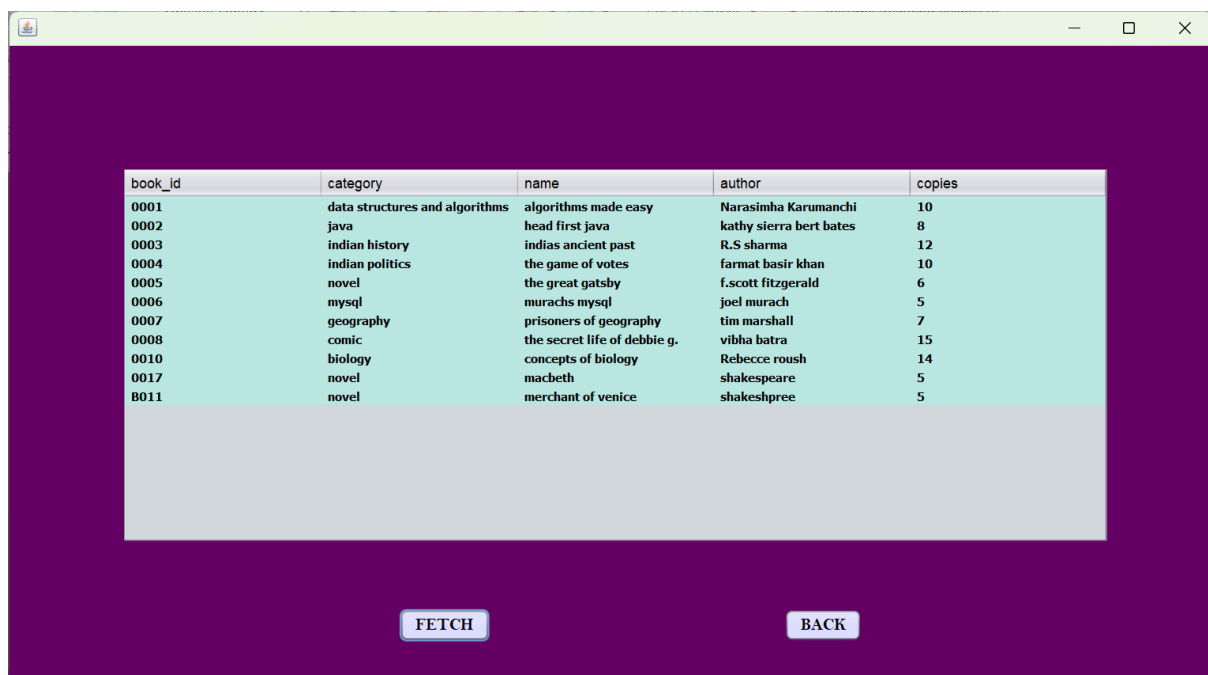
The screenshot shows a Java Swing window titled "LOGIN" with a purple background. The window contains the following elements:

- USERNAME:** A text field containing the value "harsha@123".
- PASSWORD:** A text field containing a series of asterisks "\*\*\*\*\*".
- LOGIN:** A button located at the bottom center of the window.

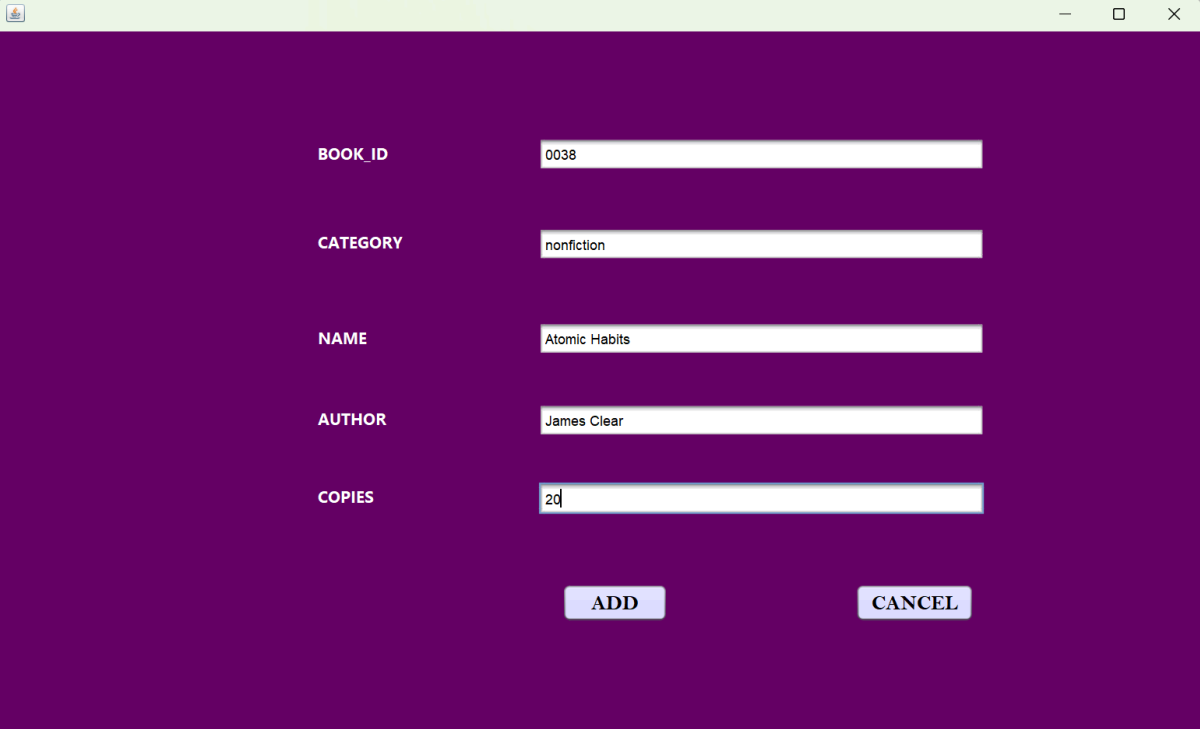
## DASHBOARD



## BOOKS AVAILABLE



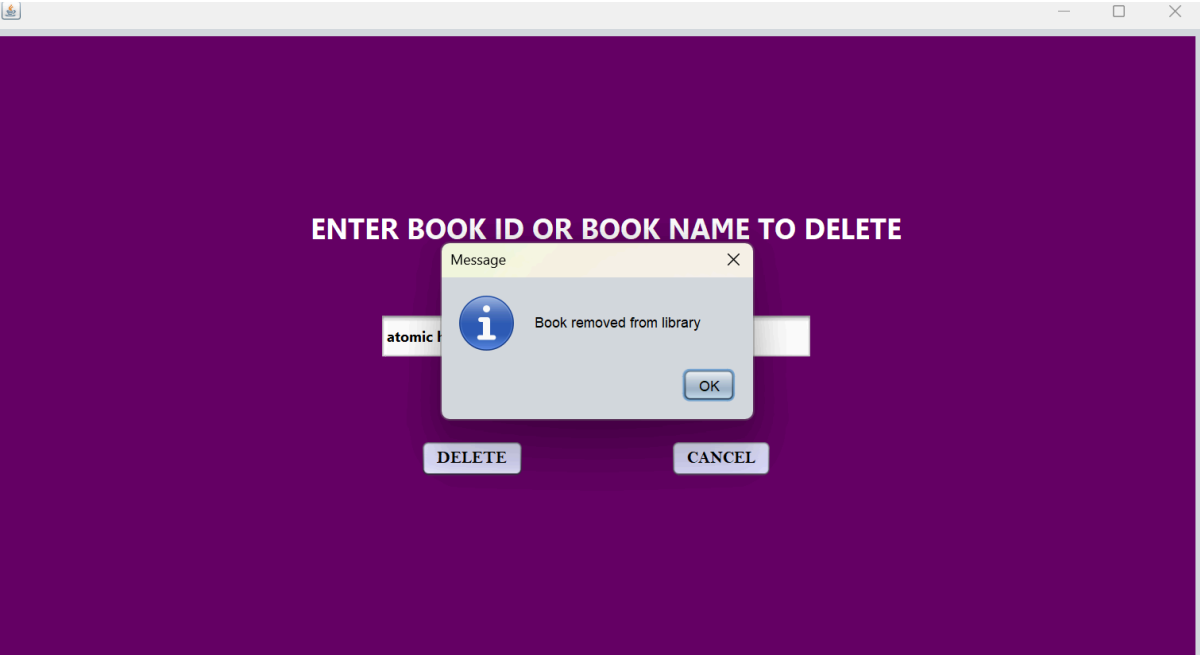
## BOOK DETAILS



BOOK_ID	0038
CATEGORY	nonfiction
NAME	Atomic Habits
AUTHOR	James Clear
COPIES	20

ADD CANCEL

## DELETE BOOK



ENTER BOOK ID OR BOOK NAME TO DELETE

atomic h

Message

Book removed from library

OK

DELETE CANCEL



## STUDENT DETAILS

student_id	name	department	contact
s001	hari	cse	348694085
s002	harsha	cse	234567890
s003	jananie	cse	593275960
s004	darshu	cse	586793867
s006	Rocky	ece	345678921
s007	Rahul	IT	234768954

**FETCH** **EXIT**

**ROLL NO**

**NAME**

**DEPARTMENT**

**CONTACT**

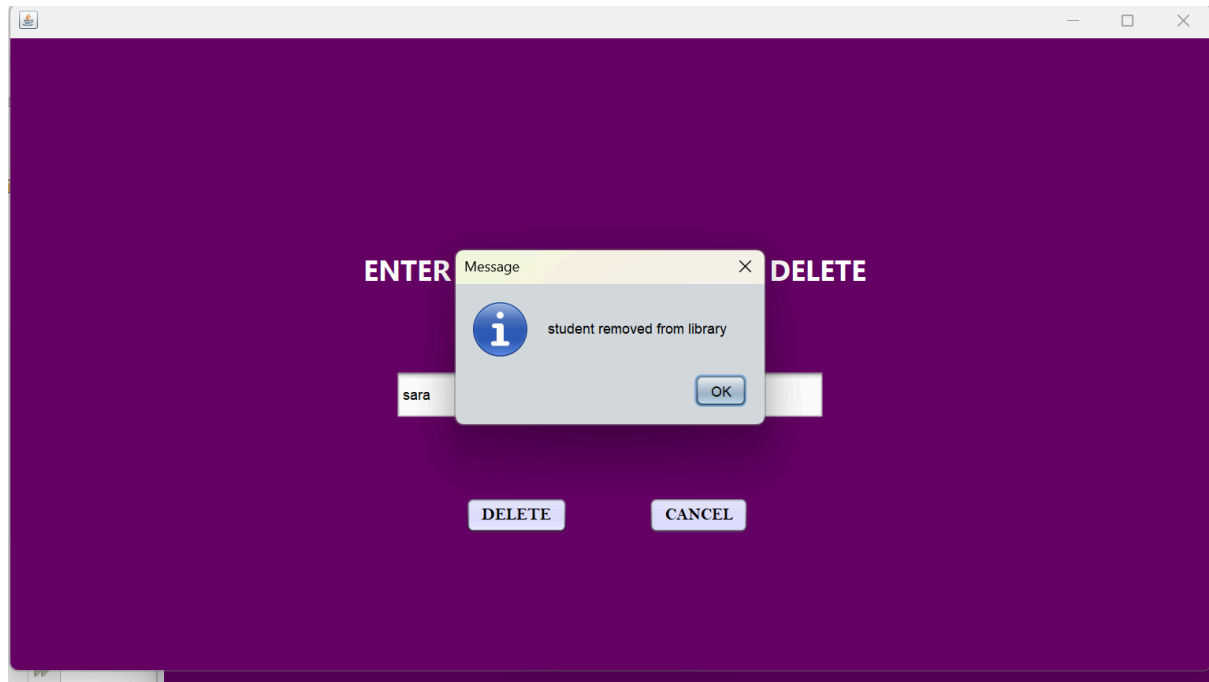
**ADD** **CANCEL**

Message

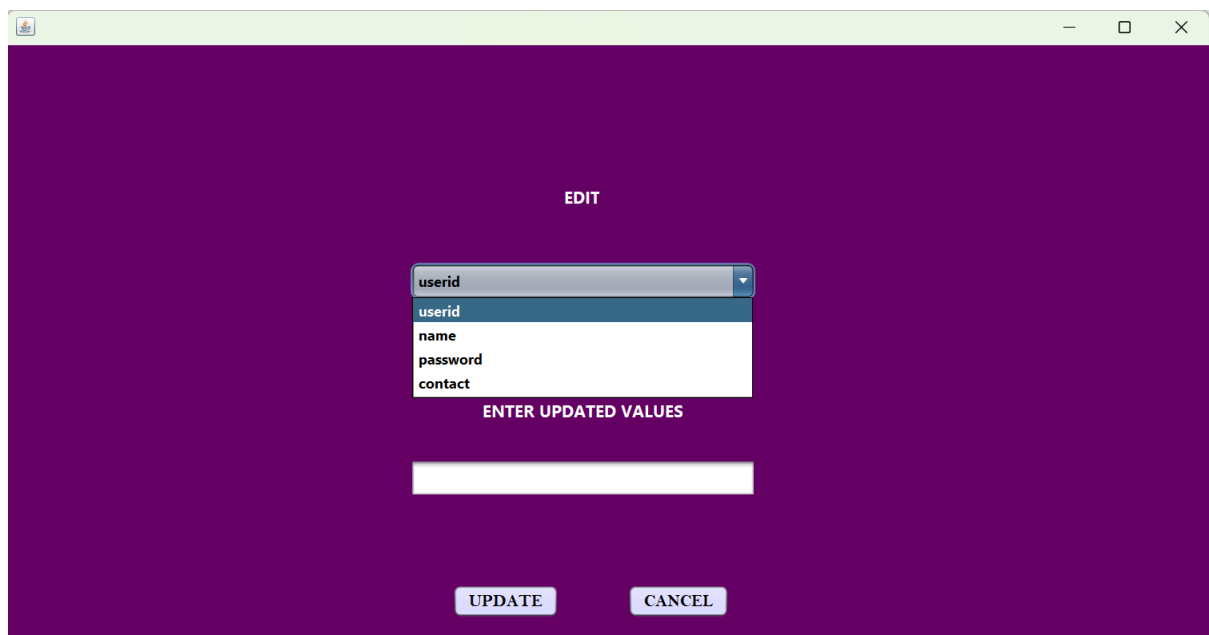
**i** One student added successfully

**OK**

## DELETE STUDENT



## EDIT ADMIN



## **Chapter 6 CONCLUSION**

The Library Management System database structure designed here provides an organized and efficient way to manage essential data, supporting key operations like tracking students, books, and administrative functions. By defining clear entities and relationships, this structure aims to streamline the management and retrieval of information within a library.

Firstly, Administrator Management is a crucial part of this design. The system includes secure authentication, allowing only authorized administrators to log in with unique credentials. This authentication ensures that only verified users can perform critical tasks, such as adding, editing, or removing records related to students and books, enhancing the security and integrity of the library's data.

Secondly, Student and Book Records are organized into dedicated tables. The student table captures information such as student names and contact details, while the book table includes attributes like book titles, authors, and genres. This organization makes it easy to maintain comprehensive and up-to-date records, allowing for quick searches and updates as needed. It supports a smooth operation in managing a library's core assets: its users and its inventory.

Finally, the system includes Action Tracking tables, which log each action administrators perform on student and book records. This feature adds an additional layer of accountability, as each modification is recorded with details about the action type (such as "Add" or "Remove") and timestamped. This tracking capability provides a valuable audit trail, allowing the library to monitor all changes made within the system, reinforcing data integrity and providing insights into administrative activities.

In summary, this database design supports a scalable and secure library management system. It balances ease of use with robust security and tracking capabilities, ensuring that the system can meet the operational demands of a library while preserving data accuracy and accountability.

## Chapter 7 REFERENCES

### 7.1 REFERENCES

**[1]** MySql and netbeans

connectivity:<https://youtu.be/d-VMtilnLJs?si=2iyduR8bSohz60td>

**[2]**<https://github.com/search?q=library%20management%20system&type=repositories>

**[3]**Setting Up MySQL in NetBeans Connect MySQL with

**NetBeans-javatpoint-** Step-by-step guide to configure MySQL with NetBeans, essential for building a Java-based backend.

Building a Java GUI in NetBeans

**[4]** (Swing/JavaFX) Java Swing GUI Builder in NetBeans -

NetBeans Documentation This guide shows how to use NetBeans'

GUI builder for Java Swing, making it easier to design a

frontentend interface for your LMS

**[5]** JDBC Connection in NetBeans Using JDBC with MySQL in

**NetBeans-GeeksforGeeks** Covers how to connect Java with MySQL using JDBC in NetBeans, crucial for executing SQL commands from your Java code.