RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

LIBRARY MANAGEMENT SYSTEM

Submitted by

MONIKA J(231001118)
MONIGA K(231001117)

MINI PROJECT REPORT

DEPARTMENT OF INFORMATION TECHNOLOGY

RAJALAKSMI ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

Certified that this project report titled "LIBRAY MANAGEMENT SYSTEM" is the bonafide work of MONIKA J(231001118), MONIGA K (231001117)who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

P.Valarmathie
Head of The Department
Department of Information Technology
Rajalakshmi Engineering College

SIGNATURE

Mrs.S.Padimini
Assistant Professor,
Department of Information Technology
Rajalakshmi Engineering College

Submitted to Project Viva-Voce Examination held on 22 Nov 2024

ABSTRACT

The Library Management System (LMS) is a comprehensive software application designed to automate the core functionalities of a library, ensuring efficient management of books, users, and transactions. The system integrates a user-friendly front-end interface developed using Java, with a powerful backend powered by a Database Management System (DBMS) like MySQL or PostgreSQL.The front-end of the system is implemented using Java with a graphical user interface (GUI), built using JavaFX or Swing. The interface enables users (both library staff and patrons) to interact seamlessly with the system. Features include searching for books, viewing detailed book information, checking availability, borrowing, and returning books. Additionally, users can view their borrowing history, and library staff can manage user accounts, add new books, and track current loans. The intuitive interface ensures ease of use and provides dynamic responses to user actions.On the backend, the system uses a DBMS to store and manage all critical data, such as books, users, transactions, and overdue items. The database allows for efficient data retrieval and manipulation using SQL queries. Key operations include adding and updating book information, tracking borrowings, generating overdue reports, and ensuring data consistency across the system. The database also supports real-time updates and allows library staff to easily manage inventory and user records.The Library Management System automates several manual tasks, such as book issuance, return tracking, overdue notifications, and report generation. This significantly reduces administrative workload, minimizes human error, and enhances operational efficiency. It also provides real-time notifications to users regarding overdue books and due dates, ensuring smooth library operations.Overall, the Library Management System provides an effective solution to streamline library operations. It combines Java for the front-end user interface and a DBMS for the back-end data management, offering a robust and efficient tool to manage library resources effectively

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER - 1

INTRODUCTION

1.1 MOTIVATION

The motivation for developing the Library Management System (LMS) stems from the need to modernize and automate the operations of libraries, which are often burdened by inefficient manual processes. Traditional methods of managing book inventories, user records, and transactions are time-consuming and prone to errors, leading to administrative overhead and a subpar experience for both librarians and library patrons. By automating these tasks, the LMS aims to streamline core functions such as book borrowing, returns, cataloging, and tracking overdue items, improving efficiency and reducing human error.

Furthermore, the project seeks to enhance user experience by providing an intuitive, digital platform for users to search for books, manage their borrowing history, and access real-time information on book availability. With a database-driven backend, the LMS ensures accurate, up-to-date information, offering a scalable solution that can accommodate growing collections and user bases. The system also automates administrative tasks, freeing up librarians to focus on more value-added activities. Ultimately, the LMS is designed to create a more efficient, accessible, and user-friendly library management experience.

1.2 EXISTING SYSTEM

1. **Manual Library Management**:

Libraries often rely on paper-based systems, which are error-prone and inefficient, especially for tracking books, users, and overdue items.

2. **Desktop-Based Software**:

Some libraries use desktop applications like MS Access, offering basic cataloging and transaction features, but they lack real-time updates and advanced functionalities like notifications or online reservations.

3. **Open-Source Systems (Koha, PMB)**:

Open-source platforms provide features like cataloging and reporting but require technical expertise for installation, configuration, and maintenance. They may not integrate well with modern technologies.

4. **Cloud-Based Systems (Alma, Libsys)**:

Cloud-based systems offer advanced features and scalability but can be costly and less flexible for customization or integration with other systems.

1.3 PROJECT OBJECTIVES

1. **Automate Library Operations**: Streamline key tasks like book check-outs, check-ins, inventory management, and overdue tracking, reducing manual workload and errors.

2. **Real-Time Book Availability**: Enable users to instantly check book availability, ensuring accurate, up-to-date information and a smoother experience.

3. **Enhance User Experience**: Create an intuitive interface for both library staff and patrons, allowing easy book searches, transaction management, and automated overdue reminders.

4. **Improve Data Management and Reporting**: Organize library data in a database, offering powerful reporting tools to track books, users, and transactions efficiently.

5. **Ensure Scalability and Flexibility**: Build a system that can handle growing data and future enhancements, such as online reservations or mobile app integration.

1.4 PROPOSED SYSTEM

The proposed Library Management System (LMS) automates library operations, offering real-time book availability, user-friendly search, and efficient management of books and transactions. Key features include overdue tracking, notifications, report generation, and role-based security. Scalable and modular, it supports future enhancements like online reservations and mobile integration, ensuring a modern solution for libraries.

BENEFITS OF THE PROPOSED SYSTEM

1. Improved Efficiency: Automates tasks like check-ins, check-outs, and overdue tracking, reducing manual work and saving time.

2. Real-Time Information: Provides instant updates on book availability, improving user experience.

3. Enhanced User Experience: Easy-to-use interface for searching books, viewing borrowing history, and receiving overdue reminders.

4. Accurate Data Management: Centralized database ensures organized, consistent, and accessible data.

5. Scalability: The system grows with the library, handling increased books, users, and transactions.

CHAPTER 2

SYSTEM DESIGN

2.1 INTRODUCTION

A Library Management System (LMS) is a software solution designed to automate and streamline the management of library resources, including books, journals, and user transactions. It replaces traditional manual systems by providing a centralized platform for cataloging books, tracking check-ins and check-outs, managing user accounts, and handling overdue fines. The system offers real-time updates on book availability, simplifies administrative tasks, and ensures accurate data management, making it easier for library staff to operate efficiently. Additionally, it enhances the user experience by allowing patrons to search for books, check availability, and view their borrowing history. Overall, the LMS improves operational efficiency, reduces errors, and provides a more convenient and accessible way to manage library resources.

2.2 SYSTEM ARCHITECTURE

The architecture of a Library Management System (LMS) typically follows a client-server model, where various components work together to ensure smooth operation and efficient management of library resources. The system architecture can be broken down into the following layers:

1. User Interface Layer (Frontend):

- This layer interacts directly with the users (librarians and patrons). It provides an intuitive graphical user interface (GUI) where users can perform tasks like searching for books, checking availability, borrowing and returning books, and viewing their transaction history.

- Technologies: HTML, CSS, JavaScript, JavaFX (for desktop applications), or React (for web-based applications).

2. Application Logic Layer (Backend):

- This layer handles the business logic and processes requests from the front end. It manages user authentication, book transactions (check-ins, check-outs), and generates reports. It also handles the communication between the frontend and the database.

- Technologies: Java (Spring Boot, Java EE), Python (Django, Flask), or Node.js.

3. Database Layer:

- This layer stores all the data related to the library, including books, user records, transactions, and fines. It ensures data consistency, integrity, and efficient querying. The database is responsible for maintaining real-time updates on book availability and user transactions.

- Technologies: Relational Database Management Systems (RDBMS) like MySQL, PostgreSQL, or SQLite.

4. Communication Layer:

- This layer is responsible for communication between the frontend and backend components. It ensures that the user's actions (like searching for books or borrowing) are transmitted to the backend and results are sent back to the user interface.

- Technologies: RESTful APIs, JSON, or SOAP for data exchange.

5. Security Layer:

- This layer ensures the protection of data and privacy by implementing user authentication and authorization (role-based access control), ensuring that only authorized users (librarians, admins, and patrons) can perform specific actions. It also handles data encryption and secure communication.

6. Notification System (Optional):

- This subsystem sends automated reminders or notifications to users regarding overdue books, upcoming due dates, or book availability.

- Technologies: Email APIs (SMTP), SMS gateways, or push notifications.

Flow of Operation:

1. A user interacts with the user interface to search for books or perform transactions.

2. The frontend sends requests to the backend, which processes the logic (e.g., checking book availability, borrowing, or returning books).

3. The backend interacts with the database to update records or retrieve information.

4. The system communicates results back to the user interface, and any relevant notifications or reports are generated and sent to users.

2.3 SYSTEM REQUIREMENTS

HARDWARE SPECIFICATIONS:

PROCESSOR	:	Ryzen 5
MEMORY SIZE	:	4GB(Minimum)
HARD DISK	:	500 GB of free space

SOFTWARE SPECIFICATIONS:

PROGRAMMING LANGUAGE	:	Java, MySQL
FRONT-END	:	Java
BACK-END	:	MySQL
OPERATING SYSTEM	:	Windows 11

CHAPTER - 3

PROJECT DESCRIPTION

3.1 METHODOLOGIES

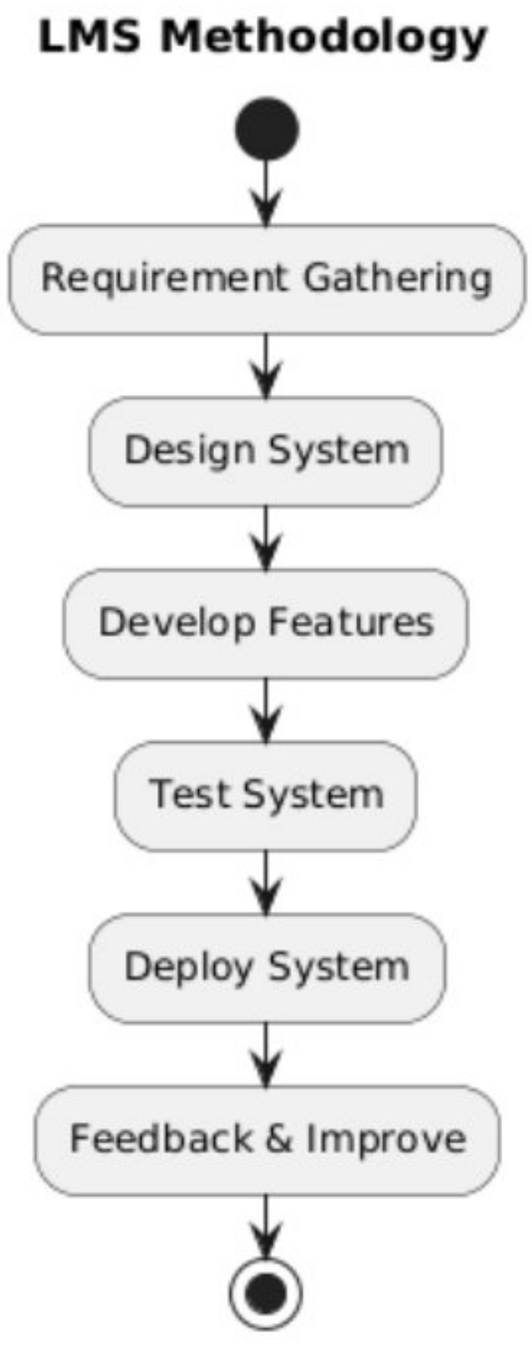


Table 3.1

3.2 CODING

1.LOGIN PAGE

```
import java.sql.*;

import javax.swing.*;


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() {


        jLabel1 = new javax.swing.JLabel();
```

```
        jLabel2 = new javax.swing.JLabel();

        jLabel3 = new javax.swing.JLabel();

        user = new javax.swing.JTextField();

        password = new javax.swing.JPasswordField();

        jButton1 = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

setResizable(false);


jLabel1.setFont(new java.awt.Font("Segoe UI", 1, 24)); // NOI18N

jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel1.setText("LOGIN");


jLabel2.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N

jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);

jLabel2.setText("PASSWORD");


jLabel3.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N

jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);

jLabel3.setText("USERNAME");


    user.addActionListener(new java.awt.event.ActionListener() {

        public void actionPerformed(java.awt.event.ActionEvent evt) {

            userActionPerformed(evt);
```

```
        }
    });


    jButton1.setFont(new java.awt.Font("Segoe UI", 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 layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addGap(0, 0, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel3,    javax.swing.GroupLayout.PREFERRED_SIZE,    93,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel2,    javax.swing.GroupLayout.PREFERRED_SIZE,    93,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(83, 83, 83)
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                .addComponent(user,        javax.swing.GroupLayout.DEFAULT_SIZE,        213,
Short.MAX_VALUE)
                .addComponent(password))
            .addGap(63, 63, 63))
        .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(173, 173, 173)
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 213,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGroup(layout.createSequentialGroup()
                .addGap(199, 199, 199)
                .addComponent(jButton1,        javax.swing.GroupLayout.PREFERRED_SIZE,
165, javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addContainerGap(199, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
      layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
          .addGap(14, 14, 14)
          .addComponent(jLabel1,      javax.swing.GroupLayout.PREFERRED_SIZE,      36,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addGap(59, 59, 59)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(user, javax.swing.GroupLayout.PREFERRED_SIZE, 43,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 43,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(54, 54, 54)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 38,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(password, javax.swing.GroupLayout.PREFERRED_SIZE, 38,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(45, 45, 45)
                .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 43,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(51, Short.MAX_VALUE))
        );


        pack();
    }// </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:jdbc:mysql://localhost:3306/?user=root
   String url="jdbc:mysql://localhost:3306/library?useSSL=false";
   String mysqluser="root";
   String mysqlpwd= "Monika@2006";
   String pswrd=new String(password.getPassword());
   String username=user.getText();
   String query=("select PASSWORD from admin where USER_ID ='"+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 wrong");
            }
        }
          else
              JOptionPane.showMessageDialog(this, "Wrong Username");


         }
        catch(Exception e)
        {   JOptionPane.showMessageDialog(this,e.getMessage());  }


}


/**
 * @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.logging.Level.S
EVERE, null, ex);
        } catch (InstantiationException ex) {

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

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

java.util.logging.Logger.getLogger(LoginPage.class.getName()).log(java.util.logging.Level.S
EVERE, 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.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JPasswordField password;
    private javax.swing.JTextField user;
    // End of variables declaration
}
```

2.DASHBOARD

```
public class Dashboard extends javax.swing.JFrame {

    /**
     * Creates new form Dashboard
     */
    public Dashboard() {
```

```
        initComponents();
        //setDefaultCloseOperation(Dashboard.DISPOSE_ON_CLOSE);
    }

    /**
     * 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() {

        jProgressBar1 = new javax.swing.JProgressBar();
        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();
        b5 = new javax.swing.JButton();
        b6 = new javax.swing.JButton();
        b7 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
```

```
        setPreferredSize(new java.awt.Dimension(585, 385));
        setResizable(false);


        jLabel1.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N
        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText("DASHBOARD");


        b1.setText("BOOKS AVAILABLE");
        b1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                b1ActionPerformed(evt);
            }
        });


        b2.setText("ADD BOOKS");
        b2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                b2ActionPerformed(evt);
            }
        });


        b3.setText("REMOVE BOOKS");
        b3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
            b3ActionPerformed(evt);
        }
    });

    b4.setText("STAFF DETAILS");
    b4.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            b4ActionPerformed(evt);
        }
    });

    b5.setText("ADD STAFF");
    b5.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            b5ActionPerformed(evt);
        }
    });

    b6.setText("REMOVE STAFF");
    b6.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            b6ActionPerformed(evt);
        }
    });
```

```
b7.setText("EDIT ADMIN");
b7.setPreferredSize(new java.awt.Dimension(113, 23));
b7.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        b7ActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
            .addGap(49, 49, 49)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                .addComponent(b1, javax.swing.GroupLayout.DEFAULT_SIZE, 169,
Short.MAX_VALUE)
                .addComponent(b2, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
```

```
                .addComponent(b3, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))

                .addGap(105, 105, 105)


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

                .addComponent(b5, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 169,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

                .addComponent(b4, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

                .addComponent(b6, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 169, Short.MAX_VALUE)))

                .addGap(58, 58, 58))

            .addGroup(layout.createSequentialGroup()

                .addGap(191, 191, 191)

                .addComponent(b7, javax.swing.GroupLayout.PREFERRED_SIZE, 169,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))

        );

        layout.setVerticalGroup(

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

            .addGroup(layout.createSequentialGroup()
```

```
                .addGap(16, 16, 16)

                .addComponent(jLabel1,      javax.swing.GroupLayout.PREFERRED_SIZE,      30,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,    60,
Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)

                    .addComponent(b1,          javax.swing.GroupLayout.DEFAULT_SIZE,          42,
Short.MAX_VALUE)

                    .addComponent(b4,                    javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

                .addGap(18, 18, 18)


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

                    .addComponent(b2,       javax.swing.GroupLayout.PREFERRED_SIZE,       42,
javax.swing.GroupLayout.PREFERRED_SIZE)

                    .addComponent(b5,       javax.swing.GroupLayout.PREFERRED_SIZE,       42,
javax.swing.GroupLayout.PREFERRED_SIZE))

                .addGap(18, 18, 18)


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

                    .addComponent(b6,       javax.swing.GroupLayout.PREFERRED_SIZE,       40,
javax.swing.GroupLayout.PREFERRED_SIZE)

                    .addComponent(b3,       javax.swing.GroupLayout.PREFERRED_SIZE,       42,
javax.swing.GroupLayout.PREFERRED_SIZE))

                .addGap(33, 33, 33)
```

```
            .addComponent(b7,        javax.swing.GroupLayout.PREFERRED_SIZE,        41,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(53, 53, 53))
        );

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

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

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

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

```
    }


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


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


    private void b5ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Add_Staff staff=new Add_Staff();
        staff.setVisible(true);
    }


    private void b6ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Remove_Staff staff=new Remove_Staff();
        staff.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.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
```

```
java.util.logging.Logger.getLogger(Dashboard.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (IllegalAccessException ex) {

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

java.util.logging.Logger.getLogger(Dashboard.class.getName()).log(java.util.logging.Level.S
EVERE, 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
```

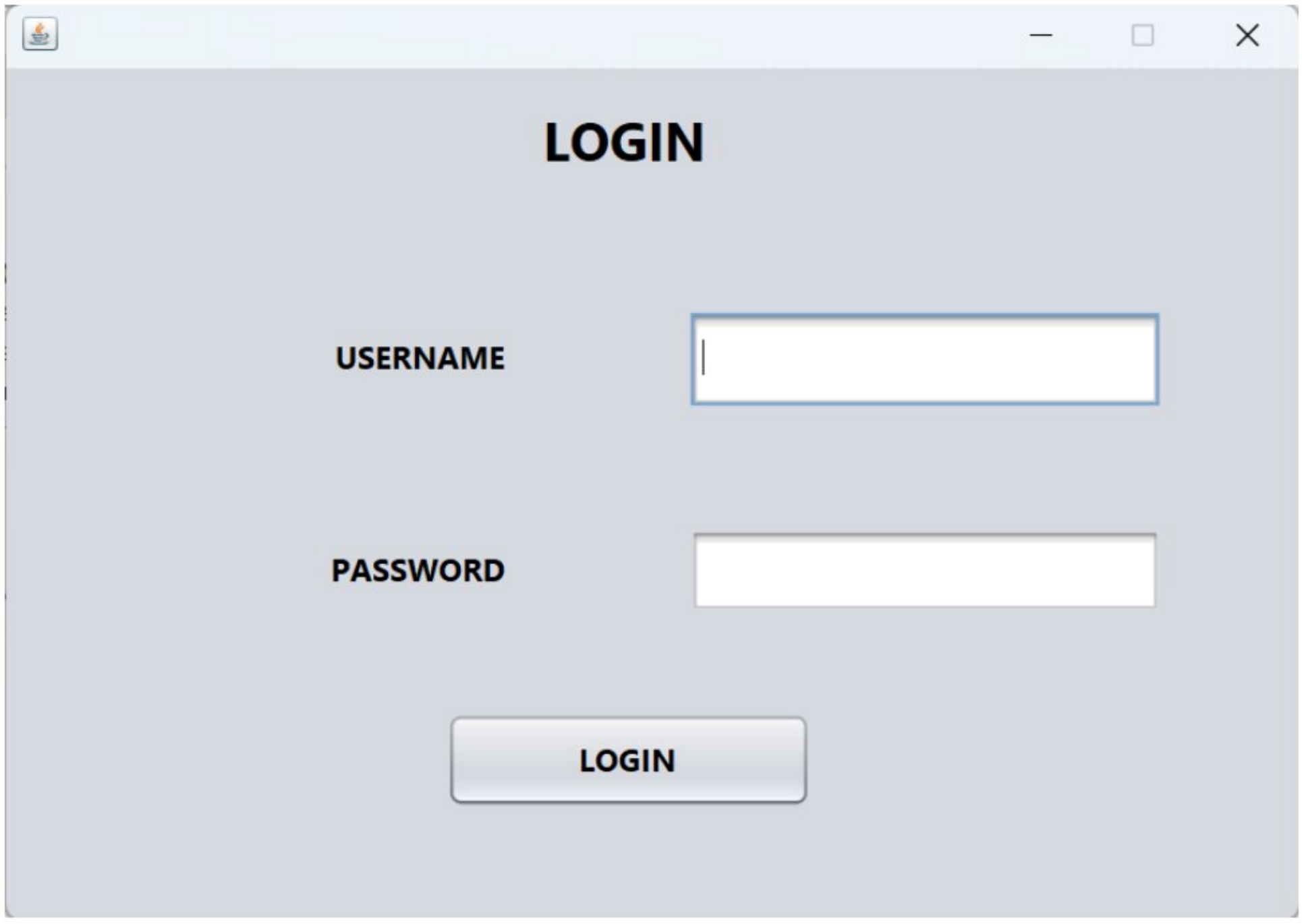
```
    private javax.swing.JButton b1;
    private javax.swing.JButton b2;
    private javax.swing.JButton b3;
    private javax.swing.JButton b4;
    private javax.swing.JButton b5;
    private javax.swing.JButton b6;
    private javax.swing.JButton b7;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JProgressBar jProgressBar1;
    // End of variables declaration
}
```

CHAPTER – 4

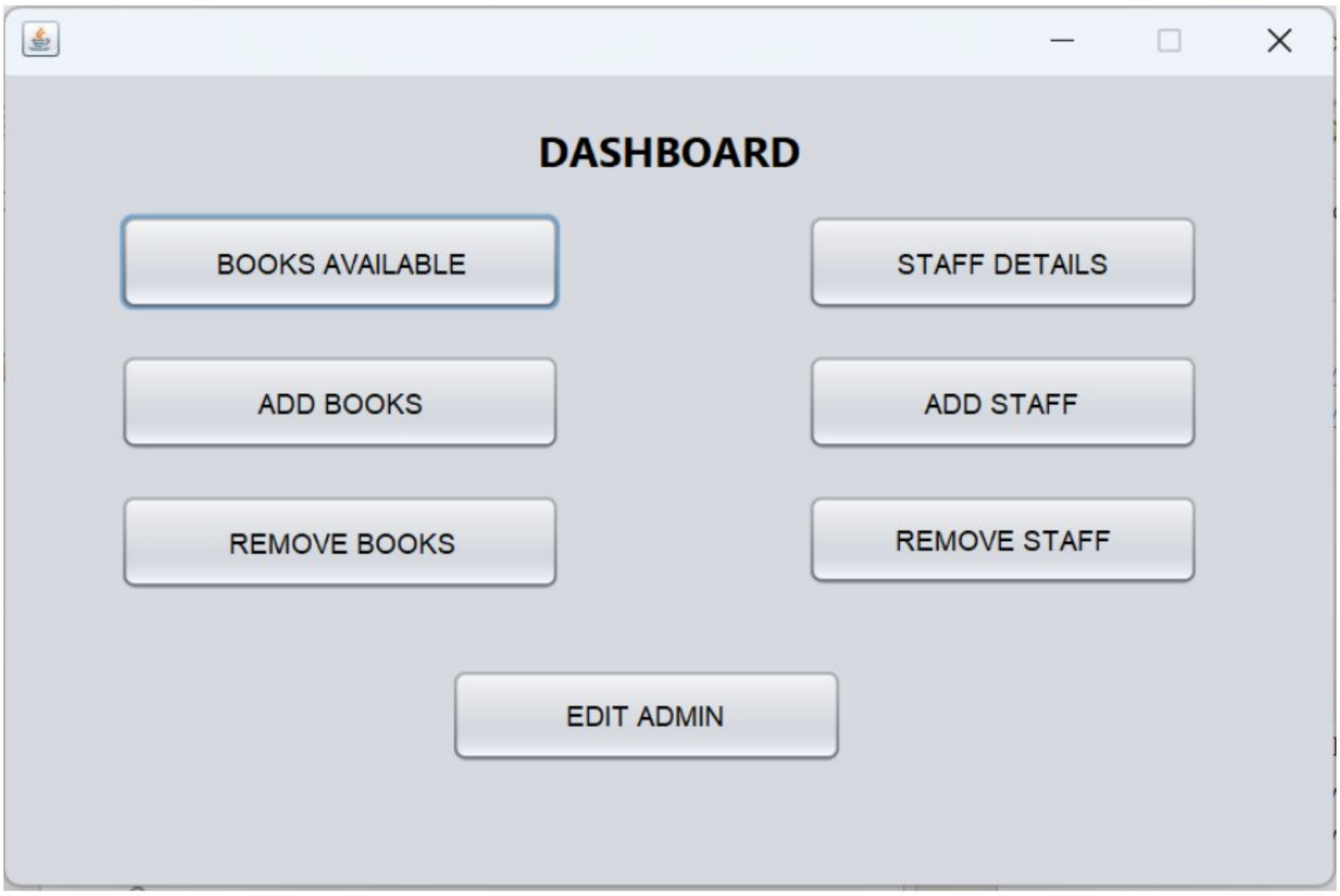
RESULTS

Output Images

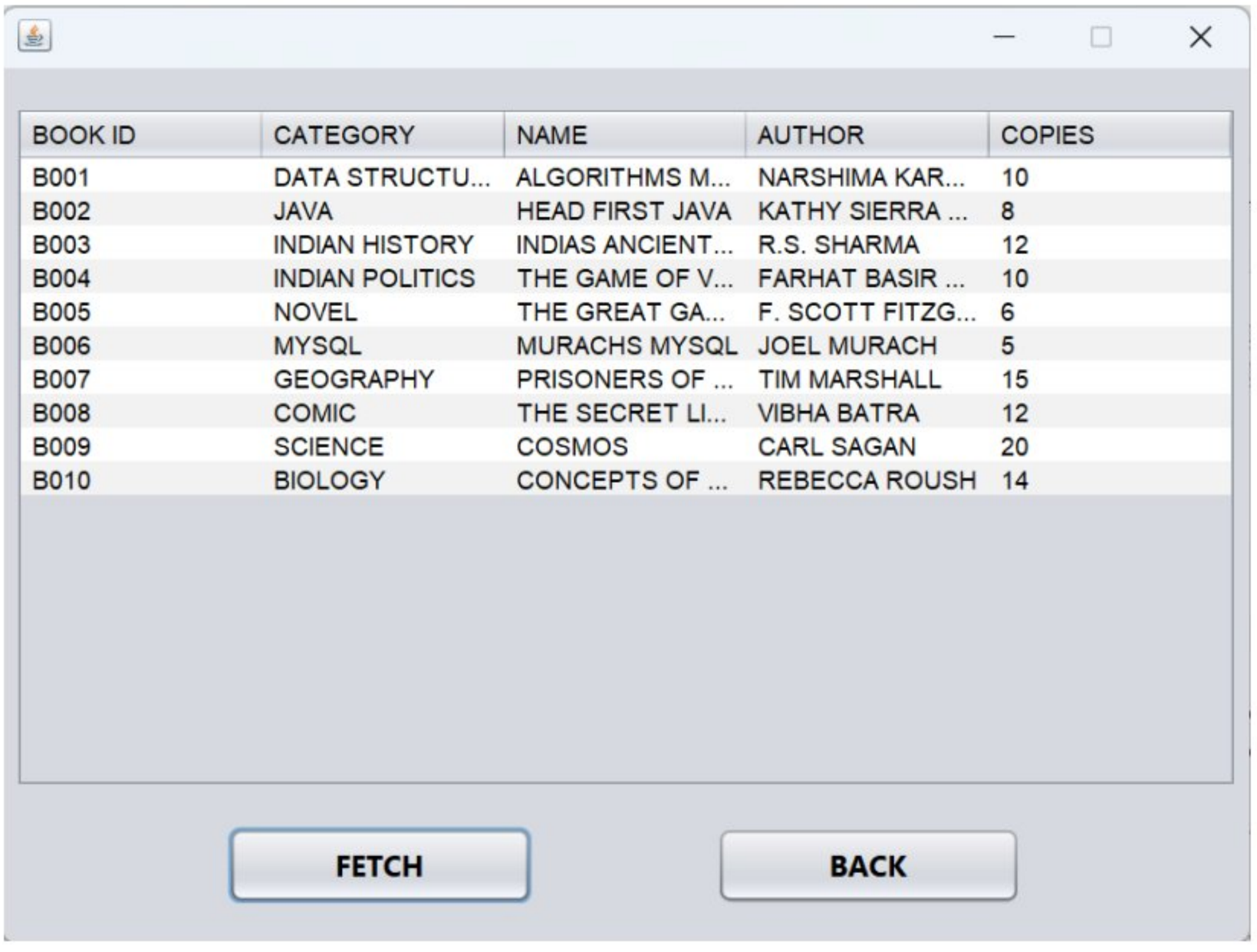
4.1 Login page



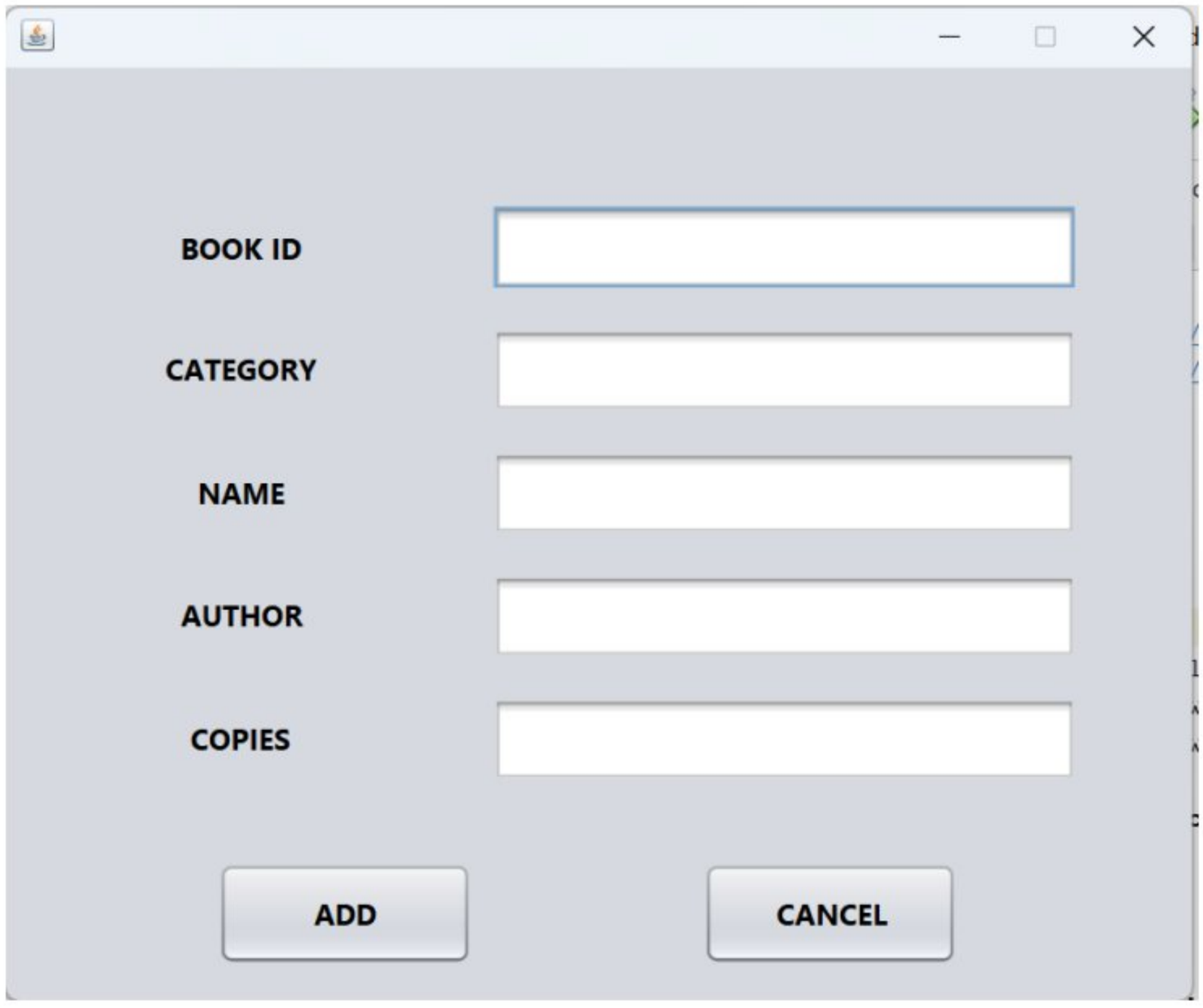
4.2 Dashboard



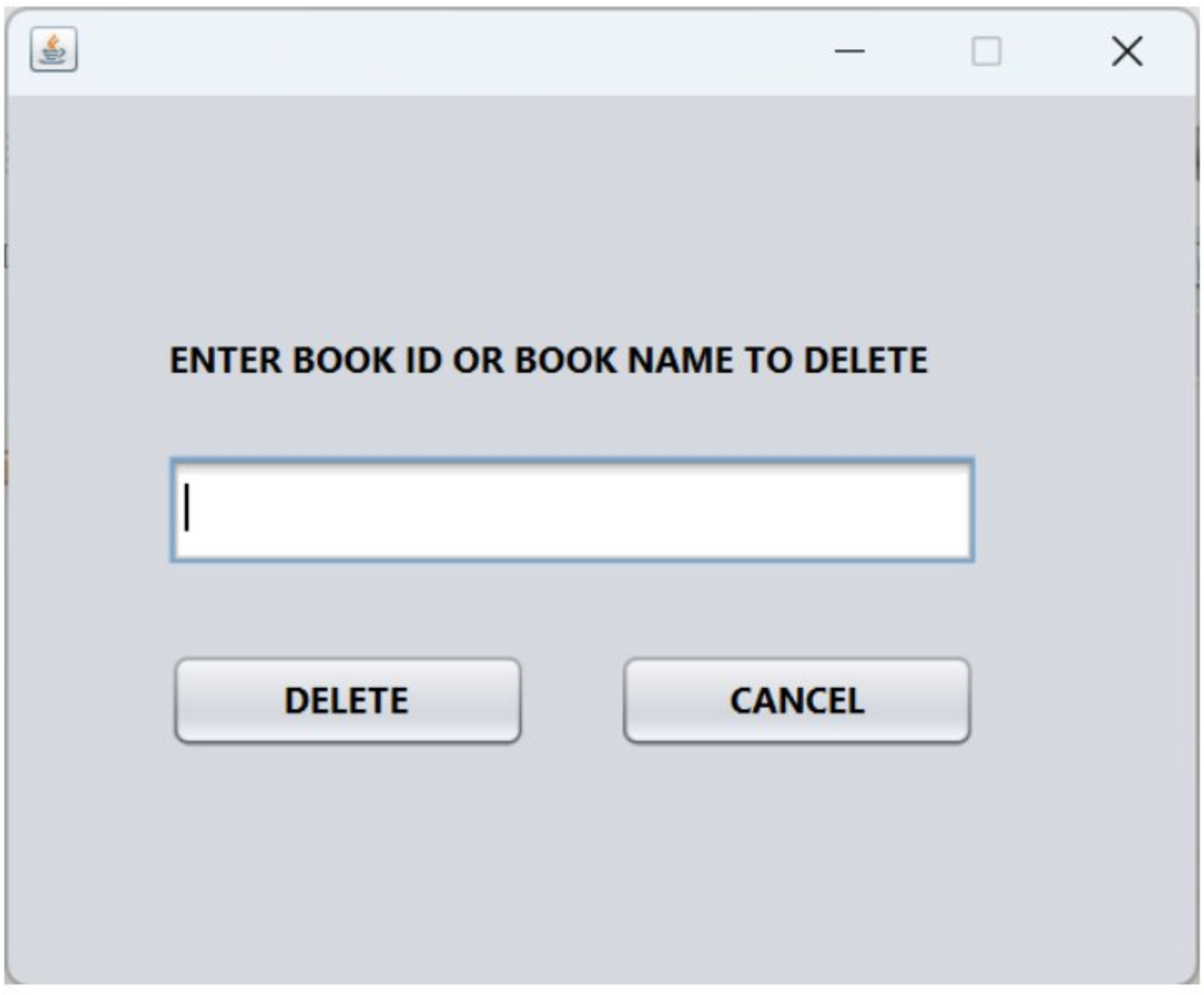
4.3 Books Available



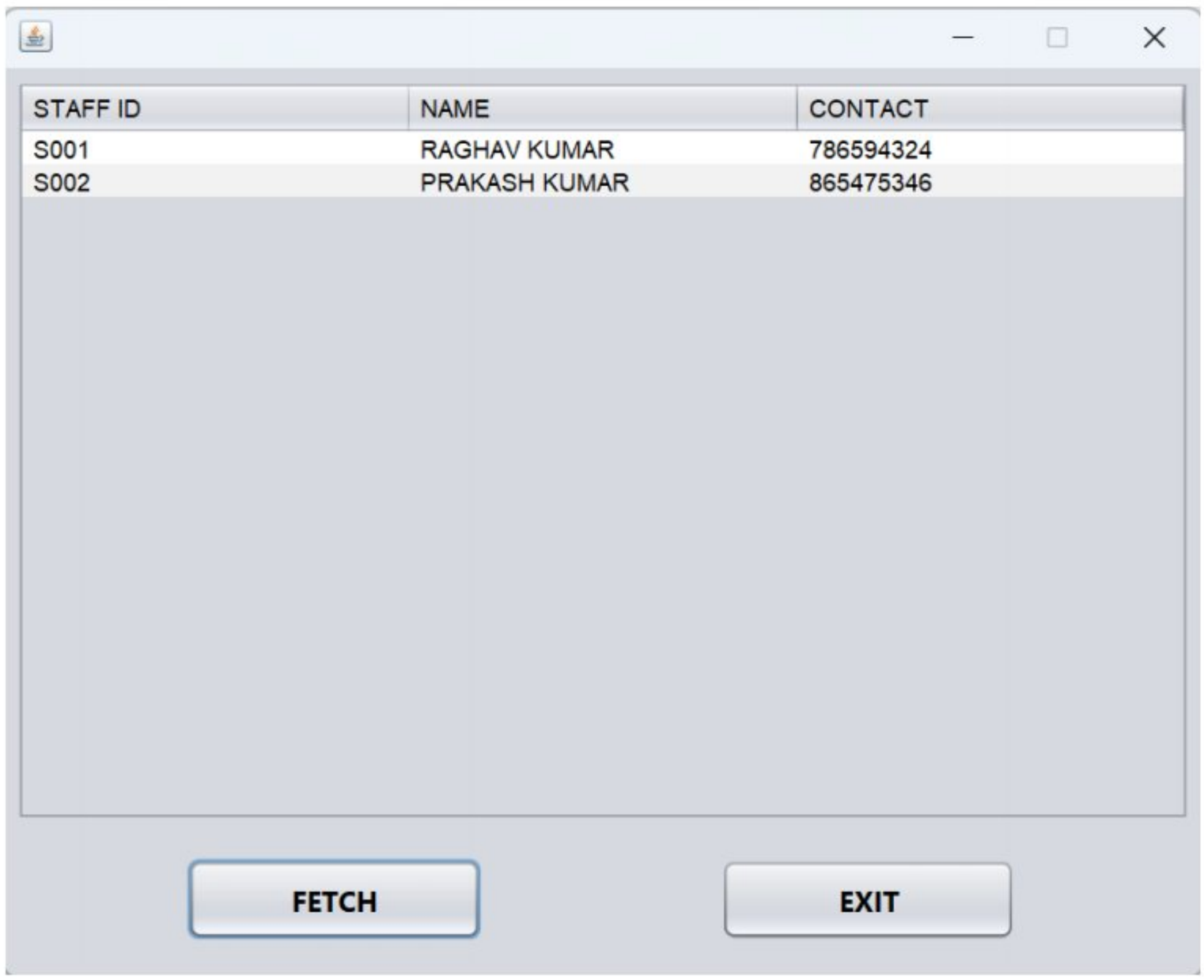
4.4 Add Books



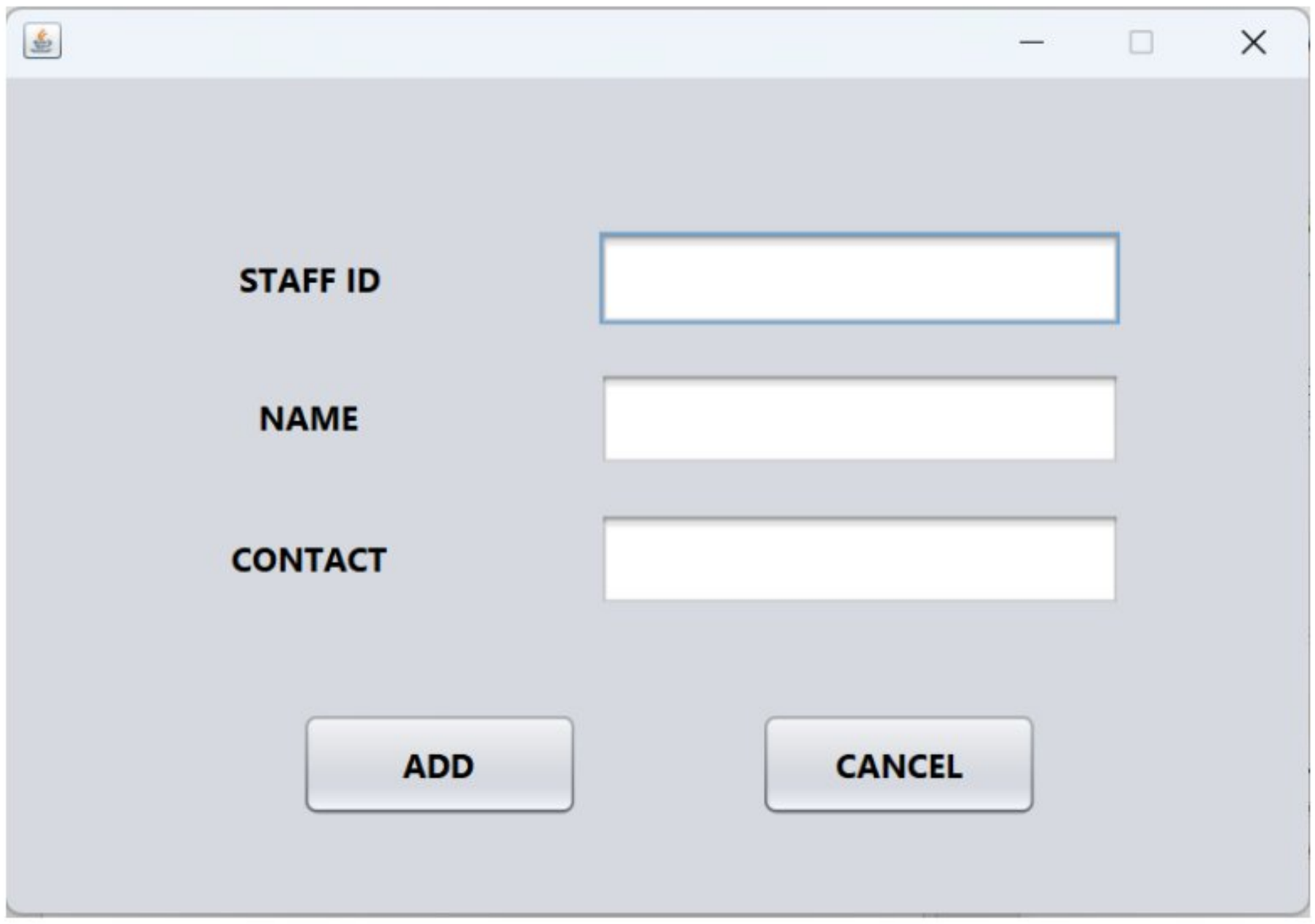
4.5 Delete Book



4.6 Staff Details

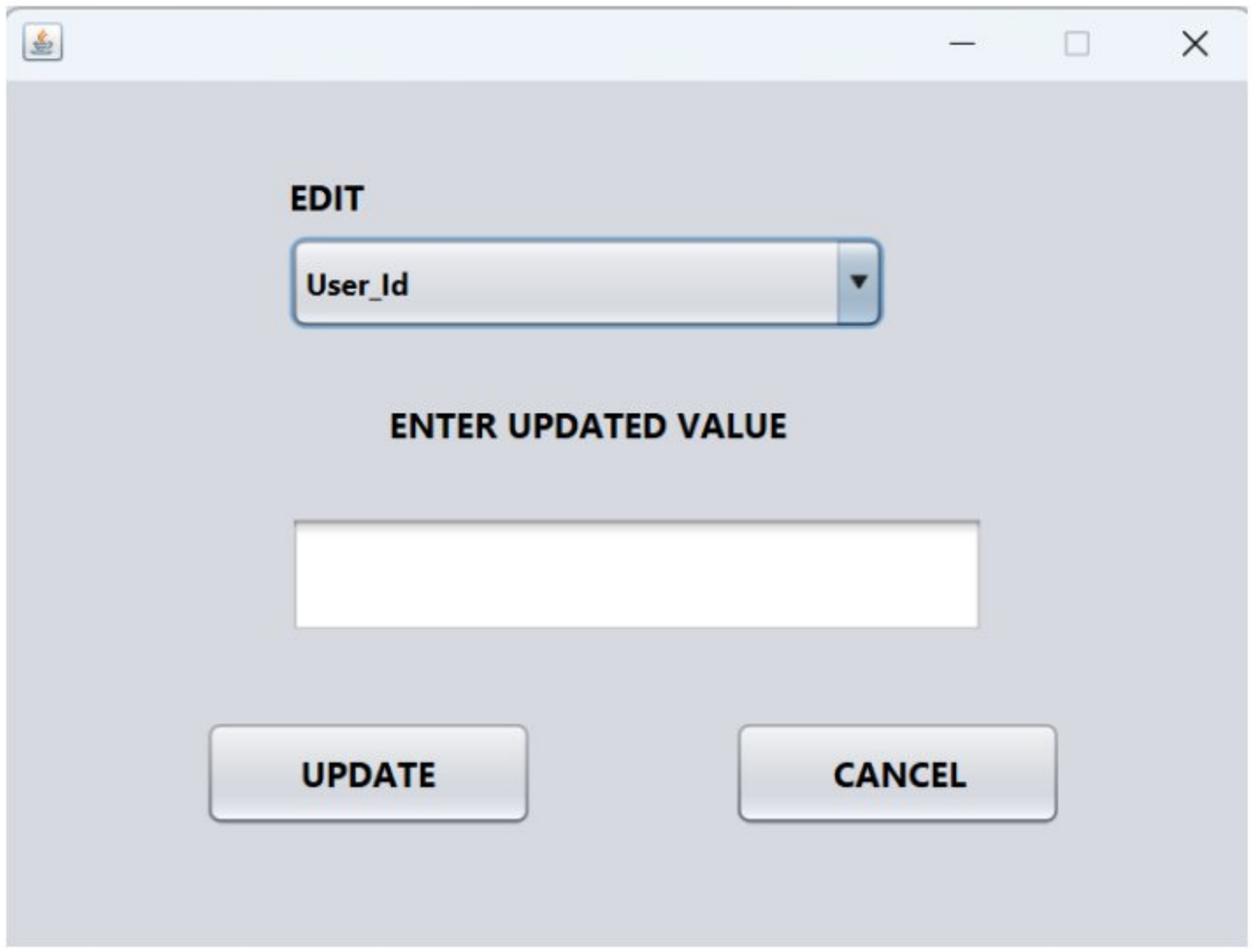


4.7 Add Staff



4.8 Remove Staff

4.9 Edit Admin



CONCLUSION

The Library Management System (LMS) aims to modernize and streamline the management of library resources, making operations more efficient for both staff and patrons. By automating tasks such as book check-ins, check-outs, overdue tracking, and user management, the LMS reduces manual work, minimizes errors, and ensures real-time updates of book availability. Its user-friendly interface enhances the overall experience for library users, enabling them to easily search for books, view borrowing history, and receive notifications.

Ultimately, the LMS provides a scalable, secure, and flexible solution to address the evolving needs of modern libraries. It offers improved data management, reduces administrative burdens, and provides a centralized platform for managing large volumes of books and users. With its potential for future enhancements, the LMS ensures that libraries can adapt to technological advancements while delivering better services to their patrons.