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

LIBRARY MANAGEMENT SYSTEM

MINI PROJECT REPORT

SUBMITTED

R.R.RENITHJOEL 230701263

M.PRAVEEN 230701243

In partial fulfilment for the award of the degree

BACHELOR OF ENGINEERING

IN COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE(AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024- 2025

BONAFIDE CERTIFICATE

Certified that this project report

"LIBRARY MANAGEMENT SYSTEM

is the bonafide work of

“R.R.RENITHJOEL (230701263),

M. PRAVEEN (230701243)”

who carriedout the project work under my
supervision.

Submitted for the Practical Examination held on 23.11.2024_____

Signature of faculty in-charge

ABSTRACT

The Library Management System (LMS) is a software application designed to automate and manage the day-to-day activities of a library, including book management, user registration, borrowing, returning, and cataloging. This system utilizes Java as the programming language and MySQL as the database management system to handle the storage and retrieval of data efficiently.

The system allows users to search for books, check availability, and manage their borrowing records, while library administrators can manage books, track inventory, issue books, and generate reports. The use of Java provides a robust and scalable platform for implementing the core functionality, ensuring a smooth and responsive user experience. MySQL, on the other hand, offers a reliable and structured database to store information about books, users, transactions, and other relevant data, with support for efficient queries and data integrity

- User Management
- Borrowing and Returning Books
- Maintaining Library Records

Each of these accesses a database schema with corresponding tables to support their operations.

Language Used:- Java Core

Concept Used:- Swing

IDE Used:- Apache NetBeans
Database Used:- MySQL

CONTENTS

CHAPTER 1

INTRODUCTION

The proposed Library Management System, developed using Java and MySQL, offers an efficient and reliable platform for managing library activities. Java, a widely-used and versatile programming language, provides a powerful foundation for building a scalable, platform-independent system. It allows for the development of robust and interactive user interfaces and backend logic to handle operations like book cataloging, user management, and transaction processing. MySQL, a popular relational database management system, ensures the secure and efficient storage of all data, including user profiles, book details, and borrowing history.The proposed Library Management System, developed using Java and MySQL, offers an efficient and reliable platform for managing library activities. Java, a widely-used and versatile programming language, provides a powerful foundation for building a scalable, platform-independent system. It allows for the development of robust and interactive user interfaces and backend logic to handle operations like book cataloging, user management, and transaction processing.

1.1 Problem Definition

Libraries, whether public, academic, or private, play a crucial role in providing access to knowledge, information, and resources. However, managing a library's collection and operations manually can lead to several challenges, including inefficiency, errors, and difficulty in tracking resources. Traditional manual systems for managing books, users, and transactions.Libraries, whether public, academic, or private, play a crucial role in providing access to knowledge, information, and resources. However, managing a library's collection and operations manually can lead to several challenges, including inefficiency, errors, and difficulty in tracking resources. Traditional manual systems for managing books, users, and transactions

1.2 Need

In today’s fast-paced, technology-driven world, libraries are evolving to meet the growing demands for efficient, accessible, and user-friendly services.

The traditional manual methods of managing library resources and operations are no longer sufficient to handle the complexity and scale of modern libraries. A Library Management System (LMS) built using Java and MySQL addresses these challenges by automating and streamlining library operations, improving overall efficiency, accuracy, and user experience. Below are key reasons why there is a strong need for such a system:

Key reasons for developing the system include:

1. Faster system
2. Accuracy
3. Reliability
4. Informative
5. Scalability and Flexibility

CHAPTER 2

REQUIREMENTS

2.1 Software Requirement Specifications

Operating System Front End Back End Server Documentation :Window 11

Frontend Software: Java NetBeans 8.2 : JDK 8

Backend Software: MySQL

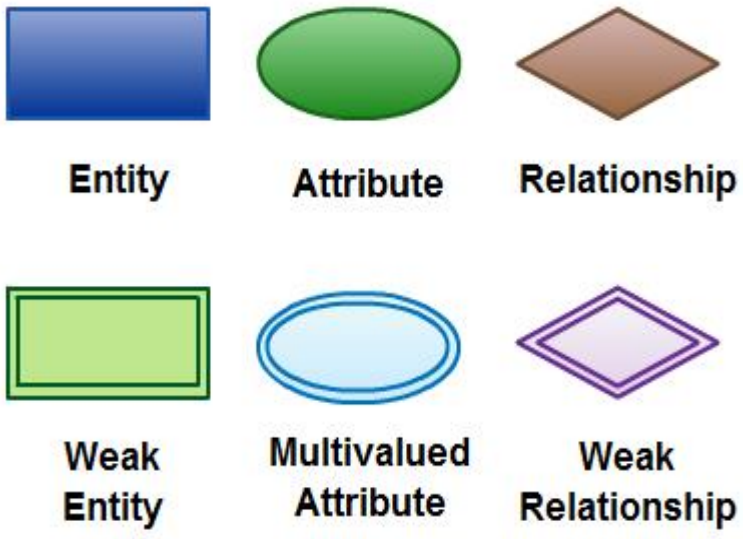
2.2 Hardware Requirement Specifications

Computer Processor Core i3 Processor Speed 2.3 GHz Processor Hard Disk 400 GB or more RAM Min 2GB

CHAPTER 3

ENTITY RELATIONSHIP DIAGRAM

An entity-relationship (ER) diagram is a specialized graphic that illustrates the interrelationships between entities in a database. ER diagrams often use symbols to represent three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes. If the application is primarily a database application, the entity-relationship approach can be used effectively for modeling some parts of the problem. The main focus in ER modeling is the Data Items in the system and the relationship between them. It aims to create conceptual scheme for the Data from the user's perspective. The model thus created is independent of any database model. The ER models are frequently represented as ER diagram. Here we present the ER diagram of the above mentioned project.



CHAPTER 4

SCHEMA DIAGRAM

4.1 SCHEMA DIAGRAM

A Schema Diagram for the database of a Library Management System is a graphical representation of the structure of the database, including tables, fields, relationships, and constraints. Based on the Entity-Relationship Diagram (ERD) provided earlier, I will outline the schema design for the system. This diagram helps in visualizing how data will be stored in the MySQL database and how different tables will interact with each other.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.

A database schema can be divided broadly into two categories −

- Physical Database Schema − This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
- Logical Database Schema − This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

CHAPTER 5

IMPLEMENTATION

5.1 Backend Implementation

MYSQL

MySQL is an open-source relational database management system (RDBMS). A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.

Table Library:

```
CREATE TABLE Library (
BOOK_ID INT PRIMARY KEY,
CATEGORY VARCHAR(30),
NAME VARCHAR(50) NOT NULL,
AUTHOR VARCHAR(30) NOT NULL,
COPIES NUMBER(8)

);
```

5.2 Frontend Implementation

Java Core

Core Java is the part of Java programming language that is used for creating or developing a general-purpose application. It uses only one tier architecture that is why it is called as 'stand alone' application.Core java programming covers the swings, socket, awt, thread concept, collection object and classess.

Swings

Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs.

Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

CREATING LOGIN CLASS:

```
import java.awt.HeadlessException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
public class LOGINPAGE extends javax.swing.JFrame {
```

```
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"
>//GEN-BEGIN:initComponents
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jPasswordField1 = new javax.swing.JPasswordField();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();

        setDefaultCloseOperation
(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Tempus Sans ITC", 1, 18)); // NOI18N
        jLabel1.setForeground(new java.awt.Color(255, 51, 51));
        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText("LOGIN");
        jLabel1.setBorder(new javax.swing.border.MatteBorder(null));

        jLabel2.setFont(new java.awt.Font("Yu Gothic", 1, 14)); // NOI18N
        jLabel2.setForeground(new java.awt.Color(51, 51, 255));
        jLabel2.setText("USERNAME");
        jLabel2.setBorder(javax.swing.BorderFactory.createMatteBorder(1, 1, 1, 1,
new java.awt.Color(153, 0, 0)));

        jLabel3.setFont(new java.awt.Font("Yu Gothic UI", 3, 14)); // NOI18N
        jLabel3.setForeground(new java.awt.Color(51, 51, 255));
        jLabel3.setText("PASSWORD");
        jLabel3.setBorder(javax.swing.BorderFactory.createMatteBorder(1, 1, 1, 1,
new java.awt.Color(204, 0, 0)));

        jTextField1.setFont(new java.awt.Font("Segoe UI Emoji", 2, 14)); //
NOI18N
        jTextField1.setForeground(new java.awt.Color(255, 102, 102));
        jTextField1.setBorder(new javax.swing.border.MatteBorder(null));
```

```
jPasswordField1.setFont(new java.awt.Font("Segoe UI Historic", 3, 14)); //
NOI18N
        jPasswordField1.setForeground(new java.awt.Color(204, 153, 0));
        jPasswordField1.setBorder
(javax.swing.BorderFactory.createEtchedBorder());

        jButton1.setFont(new java.awt.Font("Tempus Sans ITC", 3, 14)); //
NOI18N
        jButton1.setText("LOGIN");
        jButton1.setBorder(new javax.swing.border.SoftBevelBorder
(javax.swing.border.BevelBorder.RAISED, new java.awt.Color(255, 51, 51),
new java.awt.Color(153, 153, 0), new java.awt.Color(153, 102, 0), new
java.awt.Color(255, 0, 0)));
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setFont(new java.awt.Font("Yu Gothic UI Semilight", 3, 14)); //
NOI18N
        jButton2.setForeground(new java.awt.Color(0, 51, 255));
        jButton2.setText("CLOSE");
        jButton2.setBorder(new javax.swing.border.SoftBevelBorder
(javax.swing.border.BevelBorder.RAISED, new java.awt.Color(255, 153,
153), new java.awt.Color(102, 102, 0), new java.awt.Color(255, 0, 51), new
java.awt.Color(255, 0, 0)));

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout
(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(133, 133, 133)
                        .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 142,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(28, 28, 28)
                        .addGroup(layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(jLabel3,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(jLabel2,
javax.swing.GroupLayout.DEFAULT_SIZE, 107, Short.MAX_VALUE))
```

```
.addGroup(layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(jLabel3,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabel2,
javax.swing.GroupLayout.DEFAULT_SIZE, 107, Short.MAX_VALUE))
                .addGap(81, 81, 81)
                .addGroup(layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(jTextField1,
javax.swing.GroupLayout.DEFAULT_SIZE, 226, Short.MAX_VALUE)
                    .addComponent(jPasswordField1))))
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        .addGroup(layout.createSequentialGroup()
            .addGap(54, 54, 54)
            .addComponent(jButton1,
javax.swing.GroupLayout.PREFERRED_SIZE, 111,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap
(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 111,
Short.MAX_VALUE)
            .addComponent(jButton2,
javax.swing.GroupLayout.PREFERRED_SIZE, 116,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(56, 56, 56))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(22, 22, 22)
            .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 31,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(77, 77, 77)
            .addGroup(layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.TRAILING)
                .addComponent(jLabel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(51, 51, 51)
            .addGroup(layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 36,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jPasswordField1,
```

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton1ActionPerformed
        String url="jdbc:mysql://localhost:3307/library?
useSSL=false&allowPublicKeyRetrieval=true";
        String mysqluser="root";
        String mysqlpwd="";
        String pswrd=new String(jPasswordField1.getPassword());
        String username=jTextField1.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 (HeadlessException | SQLException e) {
            JOptionPane.showMessageDialog(this, e.getMessage());
        }



    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 | InstantiationException |
    IllegalAccessException | javax.swing.UnsupportedLookAndFeelException
    ex) {
            java.util.logging.Logger.getLogger(LOGINPAGE.class.getName
    ()).log(java.util.logging.Level.SEVERE, null, ex);
        }
```

DASHBOARD:

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

    private static final String DB_URL = "jdbc:mysql://localhost:
3307/library"; // Database URL
    private static final String USER = "root"; // Default MySQL username in
XAMPP
    private static final String PASS = ""; // Default password (blank in
XAMPP)

    public Dashboard() {
        initComponents();
        // Establish connection to the database when the form is initialized
        connectToDatabase();
    }

    private Connection connectToDatabase() {
        Connection conn = null;
        try {
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            System.out.println("Connection successful!");
            JOptionPane.showMessageDialog(this, "Connected to the
database successfully!");
        } catch (SQLException e) {
            System.err.println("Connection failed: " + e.getMessage());
            JOptionPane.showMessageDialog(this, "Failed to connect to the
database: " + e.getMessage());
        }
        return conn;
    }



private void fetchBooks() {
        Connection conn = connectToDatabase();
        if (conn != null) {
            try {
                String query = "SELECT * FROM books"; // Assuming a
table 'books' exists
                Statement stmt = conn.createStatement();
                ResultSet rs = stmt.executeQuery(query);

                // Fetch all books and print their titles (you can display this in a
JTable, etc.)
                while (rs.next()) {
                    String bookTitle = rs.getString("book_title"); // Assuming
there's a column 'book_title'
                    System.out.println(bookTitle); // Print to console or display in UI
```

```
public static void main(String args[]) {
        // Create and display the form
        java.awt.EventQueue.invokeLater(() -> new Dashboard().setVisible
(true));
    }
```

ADD BOOKS:

```
 private void b1ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_b1ActionPerformed
         // TODO add your handling code here:
        String url="jdbc:mysql://localhost/library";
        String user="root";
        String pwd= "";
        String query="insert into books values(?,?,?,?,?);";
        String id=t1.getText();
        String category=t2.getText();
        String name=t3.getText();
        String author=t4.getText();
        int copies=Integer.parseInt(t5.getText());
        String checkquery="update books set copies=copies+"+copies+"
where name='"+name+"' and category='"+category+"' and author='"
+author+"';";
        try {
            Connection conn= (Connection) DriverManager.getConnection
(url,user,pwd);
            Statement stmnt=conn.createStatement();
            int rows=stmnt.executeUpdate(checkquery);
            if(rows>0)
            {
                JOptionPane.showMessageDialog(this,"One record added
successfully");
            }
            else
            {
                PreparedStatement stm = conn.prepareCall(query);
                stm.setString(1,id);
                stm.setString(2,category);
                stm.setString(3,name);
                stm.setString(4,author);
                stm.setInt(5,copies);
                stm.execute();
                JOptionPane.showMessageDialog(this,"One record added
successfully");
            }
```

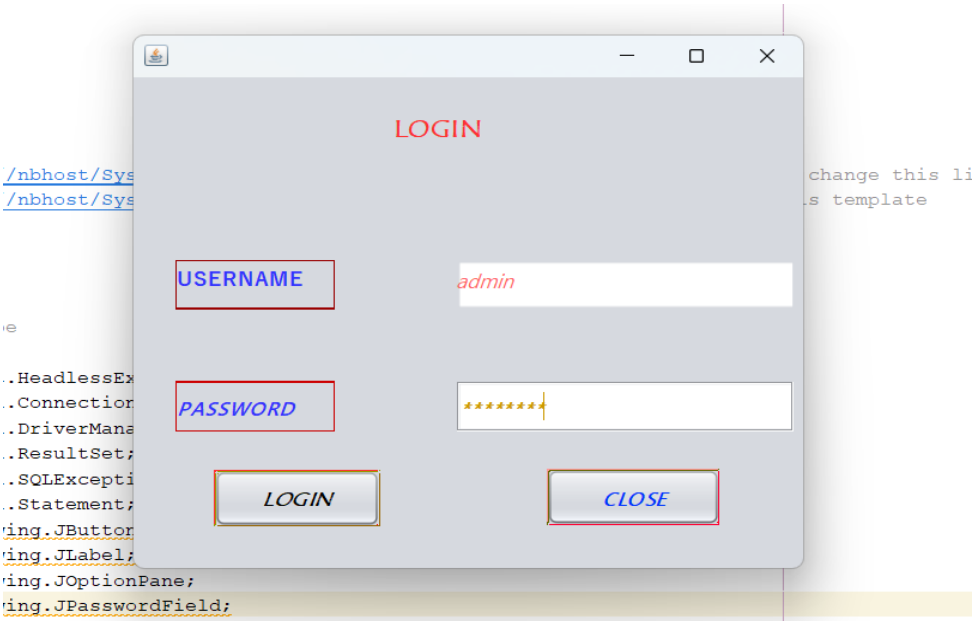
```
t1.setText(null);
            t2.setText(null);
            t3.setText(null);
            t4.setText(null);
            t5.setText(null);
        }
        catch(SQLException e)  {
         JOptionPane.showMessageDialog(this, e);
        }

    }
```

BOOKS AVAILABLE:

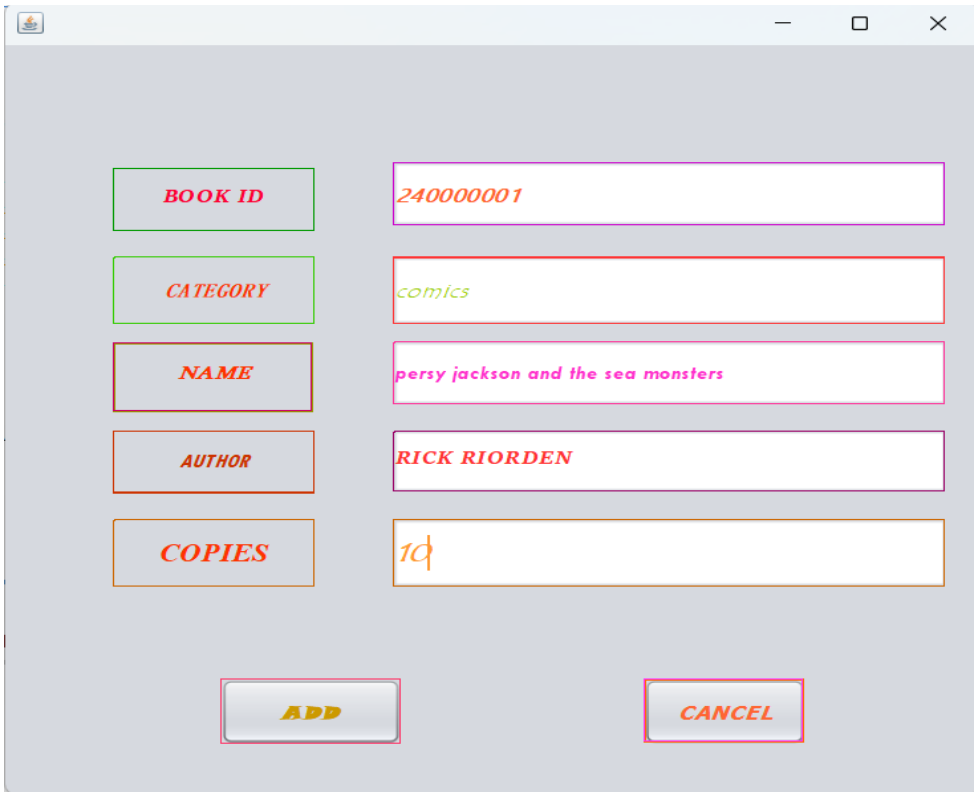
```
private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();
        jButton2 = new javax.swing.JButton();
        jButton1 = new javax.swing.JButton();

        setDefaultCloseOperation
(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jTable1.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null, null, null},
                {null, null, null, null, null},
                {null, null, null, null, null},
                {null, null, null, null, null}
            },
            new String [] {
                "BOOK ID", "CATEGORY", "NAME", "AUTHOR", "COPIES"
            }
        ));
        jScrollPane1.setViewportView(jTable1);


  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 {
```

CHAPTER-6

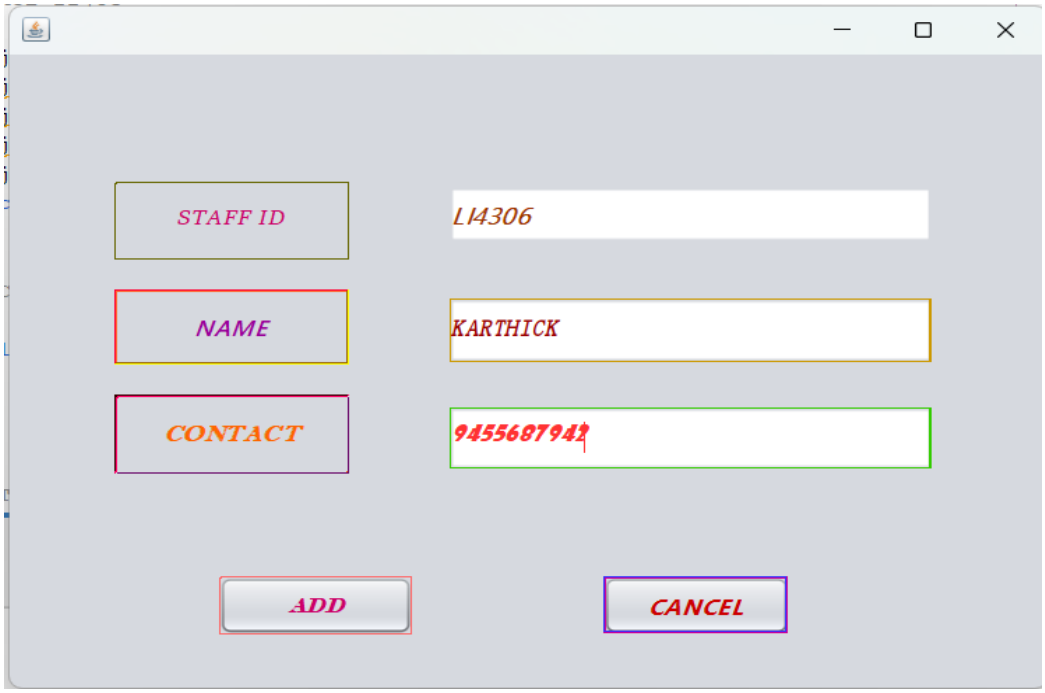
SNAPSHOTS



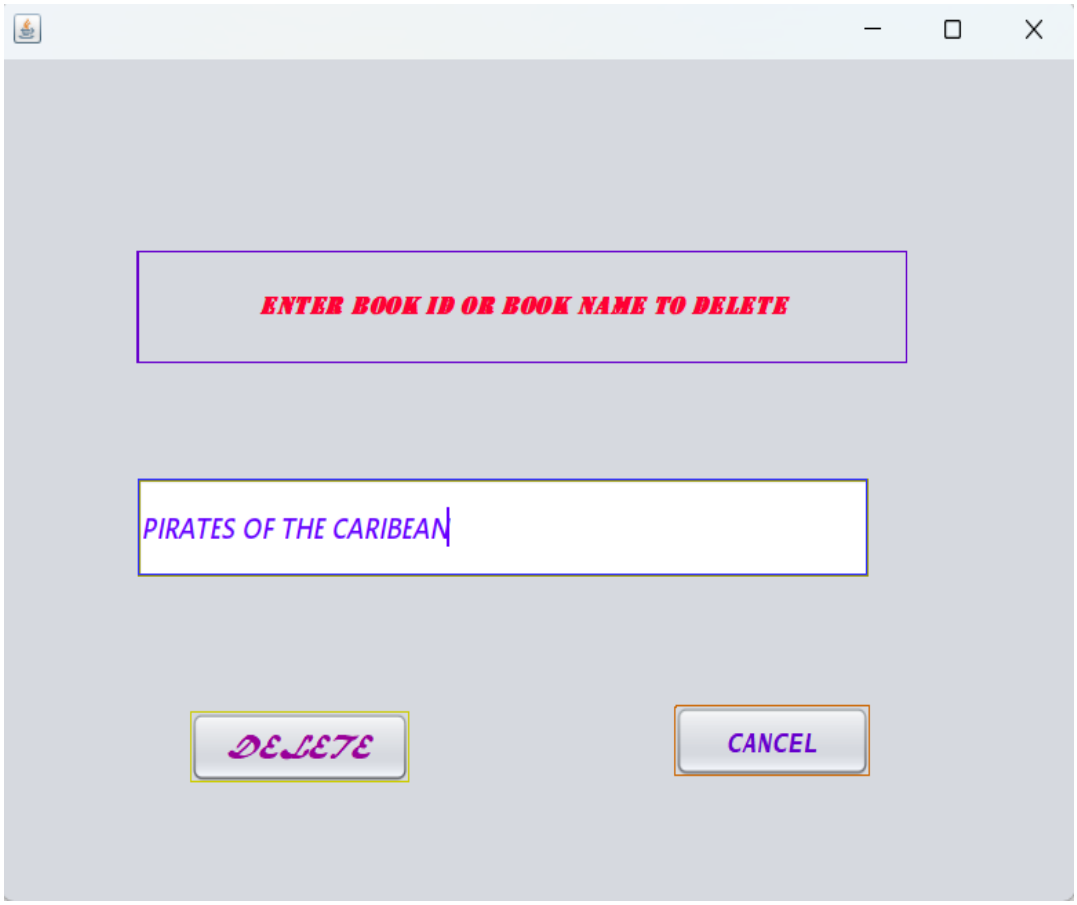
LOGIN PAGE



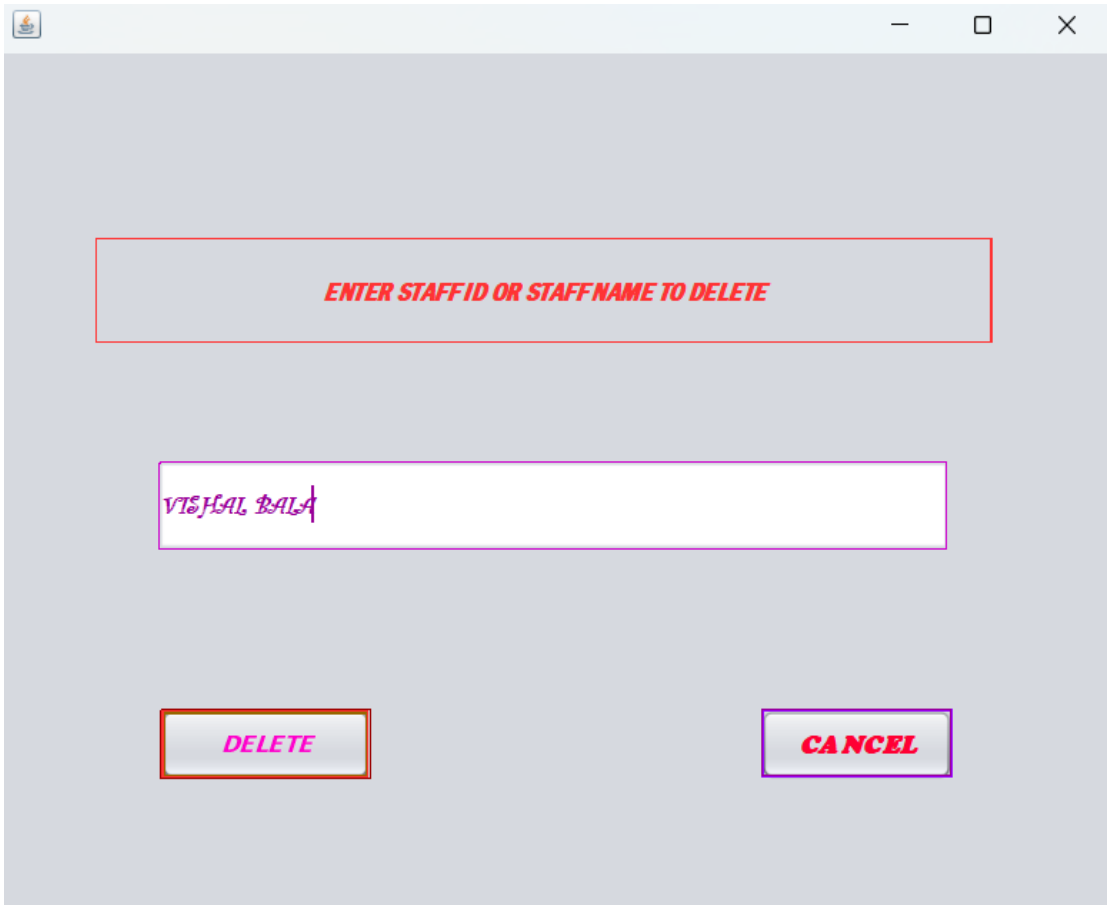
BOOK DETAILS



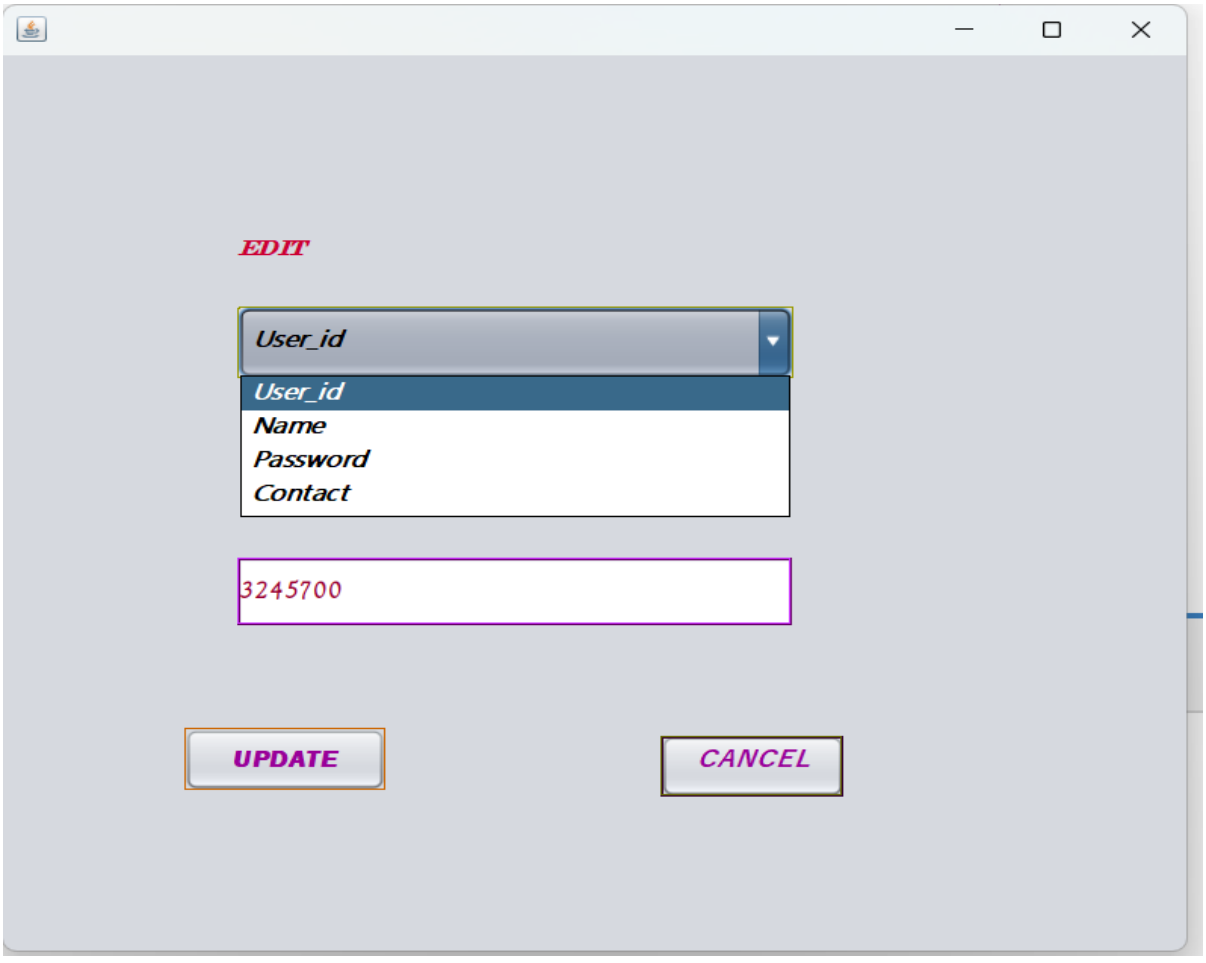
STAFF DETAILS



REMOVE BOOKS



REMOVE STAFF



ADMIN EDIT

ADMIN EDIT:

```
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 | InstantiationException |
IllegalAccessException | javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(EditAdmin.class.getName()).log
(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(() -> {
            new EditAdmin().setVisible(true);
        });
    }
```

CHAPTER - 7

BACKEND(MYSQL):

```
mysql> select * from books;
+---------+---------------------------+---------------------------+-----------------------+--------+
| BOOK_ID | category                  | NAME                      | AUTHOR                | COPIES |
+---------+---------------------------+---------------------------+-----------------------+--------+
| B001    | DATA STRUCTURE & ALGORITHMS | ALGORITHMS MADE EASY      | NARSHIMA KARUMANCHI   |     10 |
| B002    | JAVA                      | HEAD FIRST JAVA           | KATHY SIERRA BERT BATES |      8 |
| B003    | INDIAN HISTORY            | INDIAS ANCIENT PAST       | R.S SHARMA            |     12 |
| B004    | INDIAN POLITICS           | THE GAME OF VOTES         | FARHAT BASIR KHAN     |     10 |
| B005    | NOVEL                     | THE GREAT GATSBY          | F.SCOTT FITZGERALD    |      6 |
| B006    | MYSQL                     | MURACHS MYSQL             | JOEL MURACH           |      5 |
| B007    | GEOGRAPHY                 | PRISONERS OF GEOGRAPHY    | TIM MARSHALL          |      7 |
| B008    | COMIC                     | THE SECRET LIFE OF DEBBIE G. | VIBHA BATRA        |     15 |
| B009    | SCIENCE                   | COSMOS                    | CARL SAGAN            |     20 |
| B010    | BIOLOGY                   | CONCEPTS OF BIOLOGY       | REBECCE ROUSH         |     14 |
+---------+---------------------------+---------------------------+-----------------------+--------+
10 rows in set (0.00 sec)
```

UPDATING AND INSERTING RECORDS

CONCLUSION

the Library Management System built using Java and MySQL is a powerful tool that addresses the challenges of managing modern libraries. It streamlines operations, ensures data accuracy, and improves user satisfaction while providing a scalable and secure platform for future growth. By automating core functions, the system enhances the efficiency of library management and allows library staff to focus on more meaningful tasks, such as supporting users and enhancing library services. The project demonstrates the potential of Java and MySQL in creating a robust, efficient, and easy-to-use system for managing library resources, making it a valuable solution for libraries seeking to improve their operations in the digital age.

REFERENCES

Websites and Blogs:

- GeeksforGeeks - Java Programming - https://www.geeksforgeeks.org/java/
- Stack Overflow for troubleshooting Java and SQL-related queries - https://stackoverflow.com

YouTube Tutorials:

- Java and JDBC Tutorial: Java JDBC Tutorial for Beginners
- MySQL Tutorials: MySQL Database Management System Tutorials