LIBRARY MANAGEMENT SYSTEM

Desktop Application for Library Operations

A MINI-PROJECT REPORT

Submitted by

AKKSHAT N-240701028

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project "LIBRARY MANAGEMENT SYSTEM **- Desktop Application for Library Operations**" is the Bonafide work of "AKKSHAT N" who carried out the project work under my supervision.

SIGNATURE

Jananee V

ASSISTANT PROFESSOR SG

Dept. of Computer Science and Eng,

Rajalakshmi Engineering College

Chennai

This mini project report is submitted for the viva voce examination to be held on __________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The "Library Management System" is a desktop-based application designed to streamline and automate library operations, replacing traditional manual record-keeping with an efficient digital solution. The project aims to provide librarians with a centralized platform to manage books, students, and book transactions while ensuring data accuracy and accessibility.

The core functionality includes user authentication for librarians, comprehensive book management (add, view, update, delete), student registration and management, book issue and return operations with automatic fine calculation, and detailed transaction tracking. The application's frontend is built using **Java Swing**, providing an intuitive graphical user interface, while the backend leverages **MySQL database** for persistent data storage.

A key technical feature of this project is the implementation of **JDBC (Java Database Connectivity) API** to establish a robust connection between the Java application and MySQL database. All data operations, including creating, reading, updating, and deleting records (CRUD), are managed efficiently through PreparedStatements, ensuring data integrity, preventing SQL injection, and maintaining system reliability. This platform successfully demonstrates a practical library management solution that enhances operational efficiency and provides real-time book availability tracking.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable chairman **MR. S. MEGANATHAN** and the chairperson DR. **M. THANGAM MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by our

Head of The Department **Dr. E.M. MALATHY** and our Deputy Head of The Department

Dr. J. MANORANJINI for being ever supporting force during our project work

We also extend our sincere and hearty thanks to our internal guide **DR.V.JANANEE ,** for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

1. ***AKKSHAT N***

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Libraries are essential institutions for knowledge dissemination and learning. However, traditional library management using manual record-keeping is time-consuming, error-prone, and inefficient. This project addresses the need for a computerized system that can automate library operations, providing real-time tracking of books, managing student records, and handling book transactions seamlessly.

The Library Management System is a desktop application that simplifies library operations through an intuitive graphical interface. It enables librarians to efficiently manage the complete lifecycle of library resources—from acquisition to circulation—while maintaining accurate records and generating reports.

1.2 SCOPE OF THE WORK

The scope of this project covers the development of a fully functional desktop application using Java Swing and MySQL. The key deliverables are:

1. **User Authentication Module:** Secure login system for librarians with role-based access control.
2. **Book Management Module:** Complete CRUD operations for books including adding new books, viewing book catalog with search functionality, updating book details, and deleting books from the system.
3. **Student Management Module:** Registration and management of student records including enrollment details, contact information, and academic data.
4. **Book Issue Module:** Interface to issue books to students with automatic stock reduction and due date calculation (default 14 days).
5. **Book Return Module:** Process book returns with automatic fine calculation based on overdue days (₹5 per day) and stock restoration.
6. **Transaction Tracking:** Comprehensive view of all issued and returned books with filtering capabilities.
7. **Database Integration:** Implementation of MySQL database with JDBC for persistent data storage and retrieval.

1.3 PROBLEM STATEMENT

Traditional library management systems face several critical challenges:

- **Manual Record Keeping:** Paper-based systems are prone to errors, loss of data, and difficult to maintain.
- **Limited Accessibility:** Physical registers cannot be accessed simultaneously by multiple staff members.
- **Time-Consuming Operations:** Searching for book availability, student records, and transaction history is slow.
- **Inventory Tracking:** Difficult to maintain accurate real-time book availability status.
- **Fine Calculation:** Manual calculation of overdue fines is error-prone and time-consuming.
- **Report Generation:** Creating reports on book circulation, popular books, or defaulters is nearly impossible with manual systems.

There is a clear need for an automated solution that digitalizes library operations, ensures data accuracy, provides quick access to information, and generates meaningful insights.

1.4 AIM AND OBJECTIVES OF THE PROJECT

Aim

The primary aim of the "Library Management System" project is to **develop a desktop-based application that automates library operations**, creating an efficient, accurate, and user-friendly system for managing books, students, and transactions.

Objectives

To achieve the project's aim, the following objectives will be met:

- **To design and develop a desktop application** with Java Swing providing an intuitive graphical user interface.
- **To implement secure user authentication** for librarian access control.
- **To create a comprehensive book management system** allowing complete CRUD operations with real-time availability tracking.
- **To develop a student registration module** for maintaining detailed student records.
- **To build a book transaction system** handling both issue and return operations with automatic stock management.
- **To implement automatic fine calculation** based on overdue days for returned books.
- **To establish robust database connectivity using JDBC** with MySQL to manage all application data persistently.
- **To ensure proper transaction management** with rollback capabilities for maintaining data consistency.
- **To provide search and filter capabilities** across all modules for quick information retrieval.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Component	Detailed Specification
Processor	2.0 GHz Dual-Core or higher
RAM	4 GB DDR3/DDR4 Minimum
Storage	500 MB free disk space
Display	1280x720 resolution minimum
Network	Not required (standalone application)

2.2 SOFTWARE SPECIFICATIONS

Component	Detailed Specification
Operating System	Windows, macOS, or Linux
Programming Language	Java SE Development Kit (JDK) 8 or higher
GUI Library	javax.swing and java.awt
Database Management	MySQL Server (v5.7 or later)
JDBC Driver	MySQL Connector/J (mysql-connector-java-8.x.x.jar)
Database Utility	MySQL Workbench or phpMyAdmin
IDE (Optional)	Eclipse, IntelliJ IDEA, or NetBeans

CHAPTER 3

MODULE DESCRIPTION

The application is structured into four primary modules: Database Layer, Authentication Module, Management Modules, and Transaction Modules.

3.1 DATABASE LAYER MODULE

DatabaseConnection.java: Manages MySQL database connectivity.

- **Core Function:** Provides singleton pattern connection to ensure single database connection throughout application lifecycle.
- **Methods Implemented:** `getConnection()` - returns active Connection object, `closeConnection()` - safely closes database connection.
- **Connection Management:** Uses prepared connection URL with MySQL JDBC driver (com.mysql.cj.jdbc.Driver).

Database Schema:

- **users table:** Stores librarian authentication credentials (user_id, username, password, full_name, role).
- **books table:** Contains book information (book_id, book_name, author, publisher, isbn, category, quantity, available_quantity, rack_number).
- **students table:** Maintains student records (student_id, enrollment_no, student_name, course, branch, year, phone, email).
- **issue_books table:** Tracks book transactions (issue_id, book_id, student_id, issue_date, due_date, return_date, fine_amount, status).

3.2 AUTHENTICATION MODULE

LoginFrame.java: Handles user authentication.

- **Security Features:** Username and password validation against database, password masking using JPasswordField.
- **Methods Implemented:** `authenticateUser()` - validates credentials using PreparedStatement to prevent SQL injection.
- **Navigation:** On successful login, opens DashboardFrame and closes login window.

3.3 MANAGEMENT MODULES

3.3.1 Dashboard Module

DashboardFrame.java: Main navigation hub.

- **Layout:** Grid-based layout with categorized buttons for different operations.
- **Categories:**
  - Book Management: Add Book, View Books
  - Student Management: Add Student, View Students
  - Transaction Management: Issue Book, Return Book, View Issued Books
- **Features:** Logout functionality, user greeting display.

3.3.2 Book Management Module

AddBookFrame.java: Interface for adding new books.

- **Input Fields:** Book Name, Author, Publisher, ISBN, Category, Quantity, Rack Number.
- **Validation:** Ensures mandatory fields (Book Name, Author) are filled, validates quantity as positive integer.
- **Database Operation:** Uses PreparedStatement for INSERT operation with proper null handling for optional fields.
- **Features:** Clear form functionality, duplicate ISBN detection.

ViewBooksFrame.java: Displays all books in JTable.

- **Features:**
  - Search functionality across book name, author, and category.
  - Displays all book details including availability status.
  - Delete functionality with confirmation dialog.
  - Refresh button to reload data.
- **Table Configuration:** Non-editable cells, custom column widths, styled headers.

3.3.3 Student Management Module

AddStudentFrame.java: Interface for student registration.

- **Input Fields:** Enrollment No, Student Name, Course, Branch, Year, Phone, Email.
- **Validation:** Mandatory enrollment and name validation, year validation (1-5 range).
- **Database Operation:** Proper handling of NULL values for optional fields using Types.INTEGER.
- **Features:** Duplicate enrollment detection, clear form functionality.

ViewStudentsFrame.java: Displays all students in JTable.

- **Features:**
  - Multi-field search (enrollment, name, course, branch).
  - Complete student information display.
  - Delete student with confirmation.
  - Auto-refresh capability.

3.4 TRANSACTION MODULES

3.4.1 Issue Book Module

IssueBookFrame.java: Handles book issuing to students.

- **Features:**
  - Dropdown selection for available books (shows only books with stock > 0).
  - Dropdown selection for registered students.
  - Auto-populated issue date (current date).
  - Auto-calculated due date (14 days from issue date).
  - Dynamic detail display for selected book and student.
- **Database Operations:**
  - Transaction-based operation with rollback support.
  - INSERT into issue_books table.
  - UPDATE books table to decrement available_quantity.
  - Both operations committed together or rolled back on error.

3.4.2 Return Book Module

ReturnBookFrame.java: Processes book returns.

- **Features:**
  - Dropdown showing only currently issued books.
  - Displays complete issue details (issue date, due date).
  - Auto-populated return date (current date, editable).
  - Automatic fine calculation (₹5 per overdue day).
  - Calculate Fine button for manual recalculation.
- **Fine Calculation Logic:**

```
days_late = return_date - due_date
if days_late > 0:
   fine = days_late × 5.0
else:
   fine = 0.0
```

- **Database Operations:**
  - Transaction-based operation.
  - UPDATE issue_books table (return_date, fine_amount, status='returned').
  - UPDATE books table to increment available_quantity.

3.4.3 View Issued Books Module

ViewIssuedBooksFrame.java: Displays transaction history.

- **Features:**
  - Complete transaction details in JTable.
  - Status filter (All, Issued, Returned).
  - Shows book name, student details, dates, fine amount, and status.
  - Dynamic record count display.
  - Custom column widths for optimal display.

CHAPTER 4

SAMPLE CODING

DATABASE CONNECTION CODE:

java

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
```

```
public class DatabaseConnection {

    private static final String URL = "jdbc:mysql://localhost:3306/library_db";
    private static final String USERNAME = "root";
    private static final String PASSWORD = "your_password";

    private static Connection connection = null;

    public static Connection getConnection() {
        try {
            if (connection == null || connection.isClosed()) {
                Class.forName("com.mysql.cj.jdbc.Driver");
                connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
                System.out.println("Database connected successfully!");
            }
        } catch (ClassNotFoundException e) {
            System.out.println("MySQL JDBC Driver not found!");
            e.printStackTrace();
        } catch (SQLException e) {
            System.out.println("Connection failed!");
            e.printStackTrace();
        }
        return connection;
    }

    public static void closeConnection() {
        try {
            if (connection != null && !connection.isClosed()) {
                connection.close();
                connection = null;
                System.out.println("Database connection closed!");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

LOGIN FRAME CODE:

java

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

public class LoginFrame extends JFrame implements ActionListener {

    private JTextField usernameField;
    private JPasswordField passwordField;
    private JButton loginButton, clearButton;

    public LoginFrame() {
        setTitle("Library Management System - Login");
        setSize(450, 350);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setResizable(false);

        JPanel mainPanel = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                Graphics2D g2d = (Graphics2D) g;
                Color color1 = new Color(41, 128, 185);
                Color color2 = new Color(109, 213, 250);
                GradientPaint gp = new GradientPaint(0, 0, color1, 0, getHeight(), color2);
                g2d.setPaint(gp);
                g2d.fillRect(0, 0, getWidth(), getHeight());
            }
        };
        mainPanel.setLayout(null);

        JLabel titleLabel = new JLabel(" LIBRARY SYSTEM");
        titleLabel.setBounds(80, 30, 300, 40);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 28));
        titleLabel.setForeground(Color.WHITE);
        mainPanel.add(titleLabel);

        usernameField = new JTextField();
        usernameField.setBounds(70, 130, 300, 35);
        mainPanel.add(usernameField);

        passwordField = new JPasswordField();
        passwordField.setBounds(70, 205, 300, 35);
        mainPanel.add(passwordField);

        loginButton = new JButton("LOGIN");
        loginButton.setBounds(70, 260, 140, 35);
        loginButton.addActionListener(this);
        mainPanel.add(loginButton);

        add(mainPanel);
        setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == loginButton) {
            authenticateUser();
        }
    }

    private void authenticateUser() {
        String username = usernameField.getText().trim();
        String password = new String(passwordField.getPassword());

        try {
            Connection conn = DatabaseConnection.getConnection();
            String query = "SELECT * FROM users WHERE username = ? AND password = ?";
            PreparedStatement pst = conn.prepareStatement(query);
            pst.setString(1, username);
            pst.setString(2, password);

            ResultSet rs = pst.executeQuery();
```

```
            if (rs.next()) {
                String fullName = rs.getString("full_name");
                JOptionPane.showMessageDialog(this,
                    "Welcome, " + fullName + "!",
                    "Login Successful", JOptionPane.INFORMATION_MESSAGE);

                this.dispose();
                new DashboardFrame(fullName);
            } else {
                JOptionPane.showMessageDialog(this,
                    "Invalid credentials!",
                    "Login Failed", JOptionPane.ERROR_MESSAGE);
            }

            rs.close();
            pst.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new LoginFrame());
    }
}
```

ISSUE BOOK TRANSACTION CODE:

java

```
private void issueBook() {
    int bookId = getSelectedBookId();
    int studentId = getSelectedStudentId();
    String issueDate = issueDateField.getText();
    String dueDate = dueDateField.getText();

    Connection conn = null;
    PreparedStatement pst1 = null;
    PreparedStatement pst2 = null;
    try {
        conn = DatabaseConnection.getConnection();
        conn.setAutoCommit(false); // Start transaction

        // Insert issue record
        String insertQuery = "INSERT INTO issue_books (book_id, student_id, issue_date, due_date,
status) VALUES (?, ?, ?, ?, 'issued')";
        pst1 = conn.prepareStatement(insertQuery);
        pst1.setInt(1, bookId);
        pst1.setInt(2, studentId);
        pst1.setDate(3, Date.valueOf(issueDate));
        pst1.setDate(4, Date.valueOf(dueDate));
        pst1.executeUpdate();

        // Update book stock
        String updateQuery = "UPDATE books SET available_quantity = available_quantity - 1 WHERE
book_id = ?";
        pst2 = conn.prepareStatement(updateQuery);
        pst2.setInt(1, bookId);
        pst2.executeUpdate();

        conn.commit(); // Commit transaction
        conn.setAutoCommit(true);

        JOptionPane.showMessageDialog(this,
            "Book issued successfully!",
            "Success", JOptionPane.INFORMATION_MESSAGE);

    } catch (SQLException ex) {
        try {
            if (conn != null) {
                conn.rollback(); // Rollback on error
                conn.setAutoCommit(true);
            }
        } catch (SQLException e2) {
            e2.printStackTrace();
        }
        JOptionPane.showMessageDialog(this,
            "Error: " + ex.getMessage(),
            "Error", JOptionPane.ERROR_MESSAGE);
    } finally {
        try {
            if (pst1 != null) pst1.close();
            if (pst2 != null) pst2.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```

CHAPTER 5

SCREENSHOTS

Fig 5.1 LOGIN FRAME

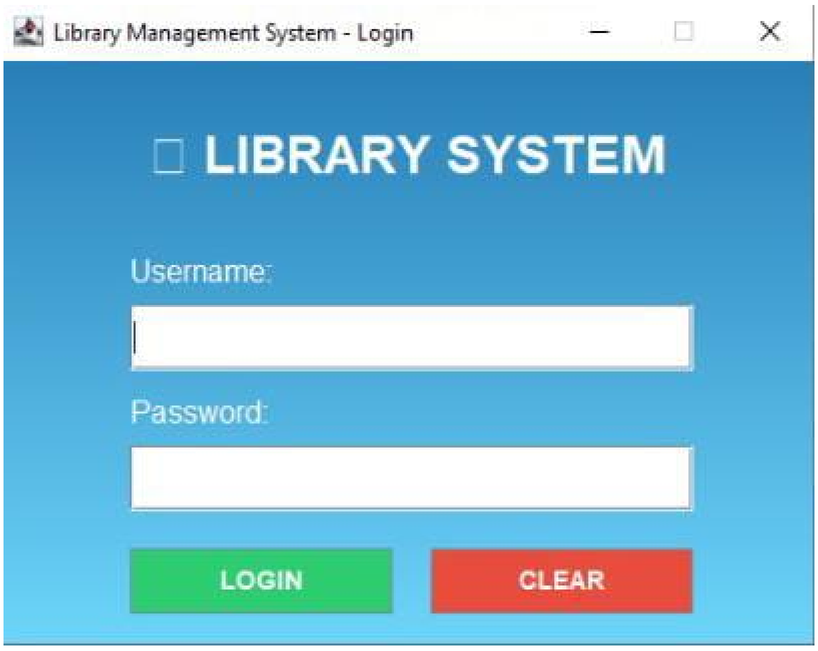


Fig 5.2 DASHBOARD FRAME

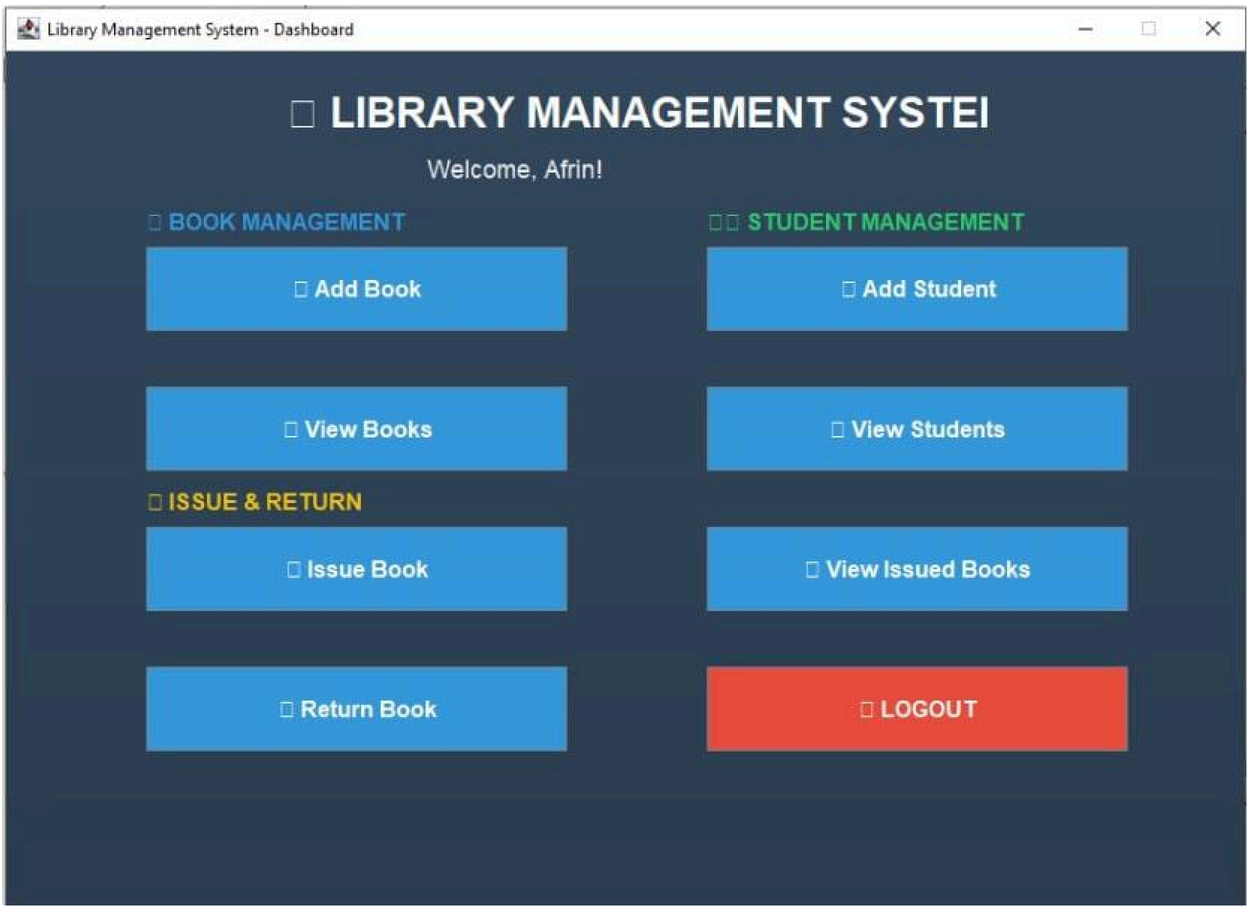


Fig 5.3 ADD NEW BOOK

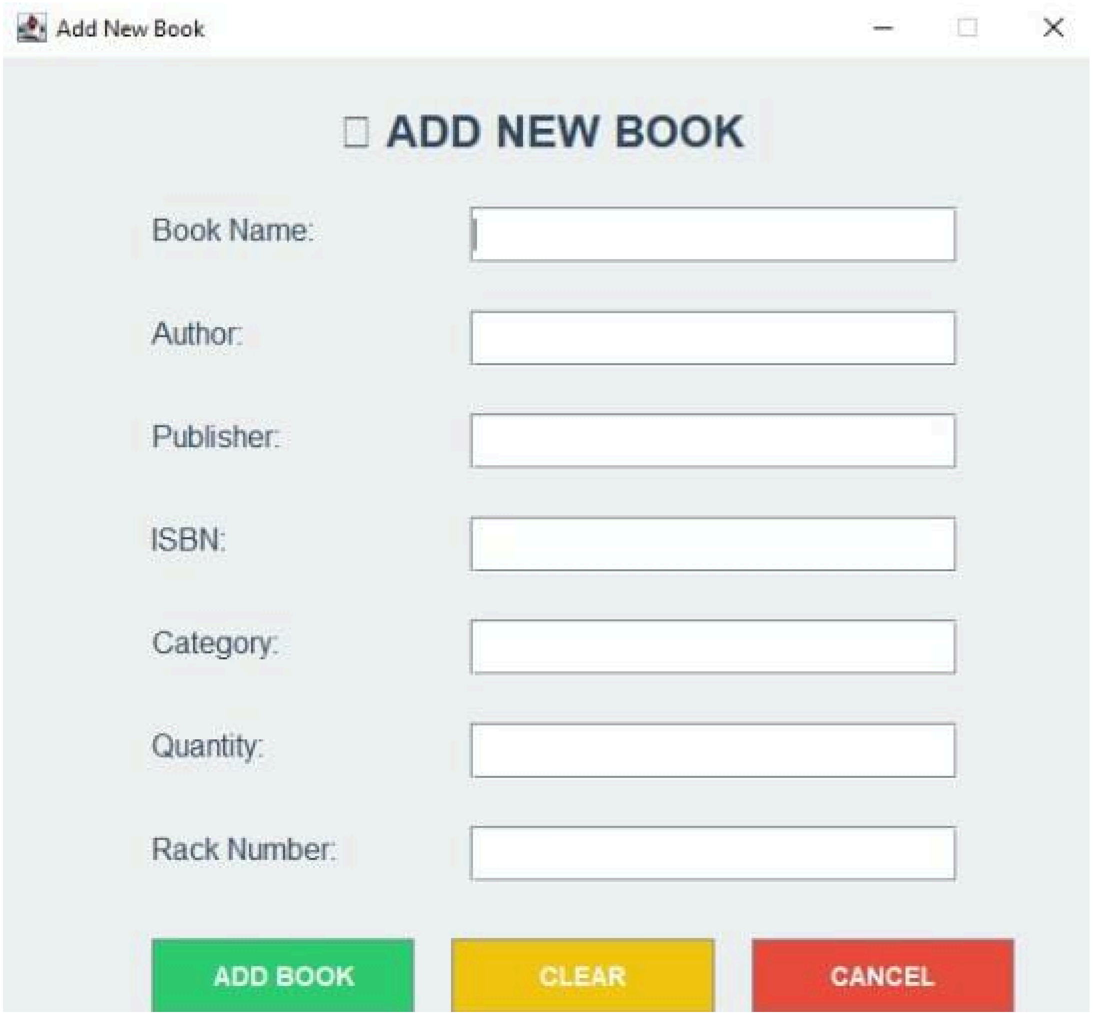
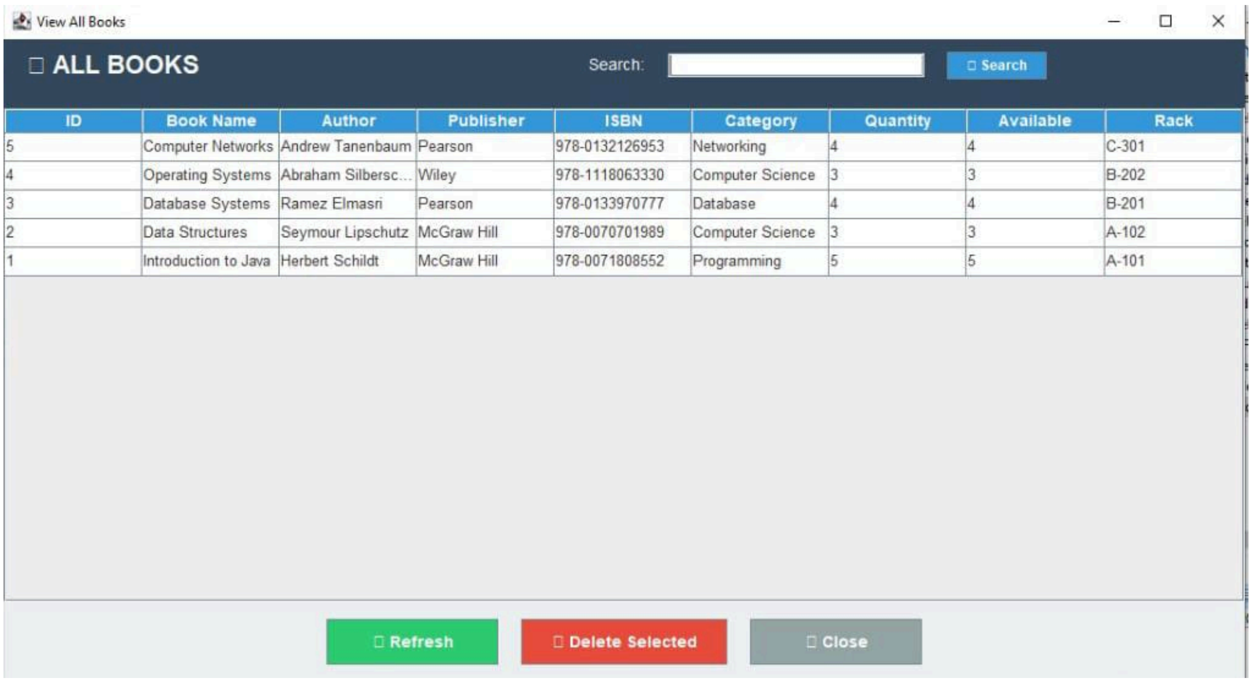


Fig 5.4 VIEW BOOKS FRAME

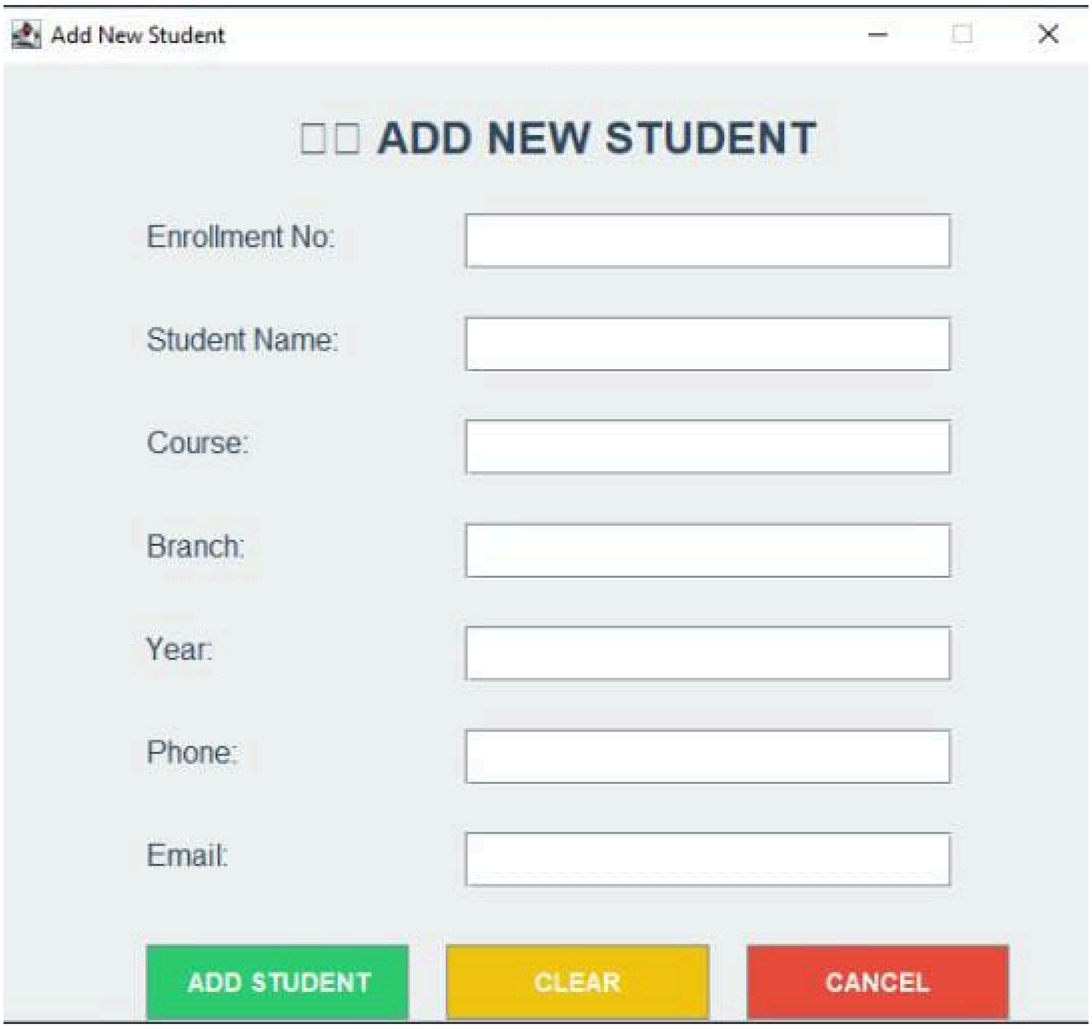

View All Books

□ ALL BOOKS

Search: □ Search

ID	Book Name	Author	Publisher	ISBN	Category	Quantity	Available	Rack
5	Computer Networks	Andrew Tanenbaum	Pearson	978-0132126953	Networking	4	4	C-301
4	Operating Systems	Abraham Silbersc...	Wiley	978-1118063330	Computer Science	3	3	B-202
3	Database Systems	Ramez Elmasri	Pearson	978-0133970777	Database	4	4	B-201
2	Data Structures	Seymour Lipschutz	McGraw Hill	978-0070701989	Computer Science	3	3	A-102
1	Introduction to Java	Herbert Schildt	McGraw Hill	978-0071808552	Programming	5	5	A-101

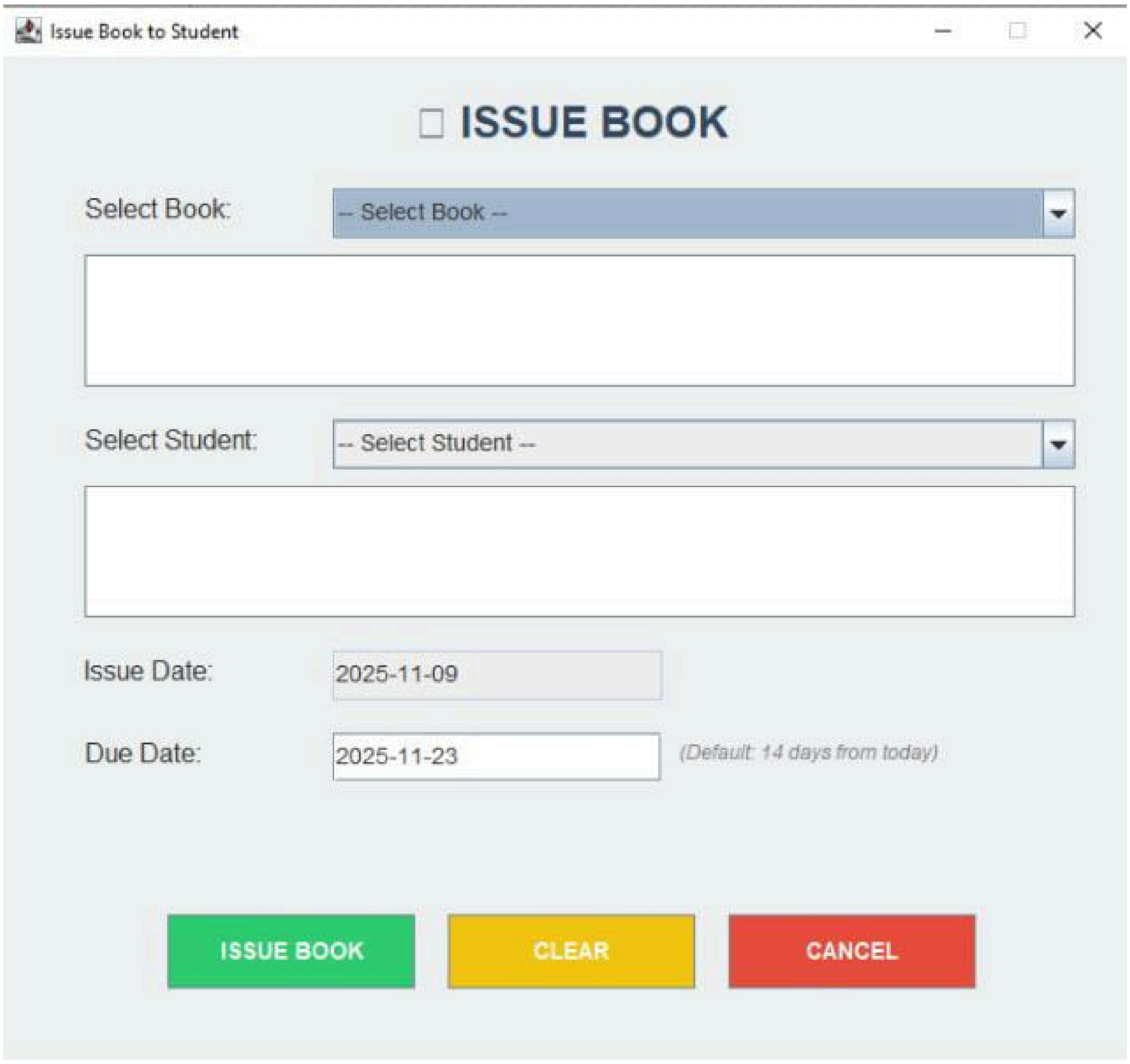
Fig 5.5 ADD STUDENT FRAME


Add New Student

□□ ADD NEW STUDENT

Enrollment No:	
Student Name:	
Course:	
Branch:	
Year:	
Phone:	
Email:	

Fig 5.6 ISSUE BOOK FRAME


Issue Book to Student

☐ ISSUE BOOK

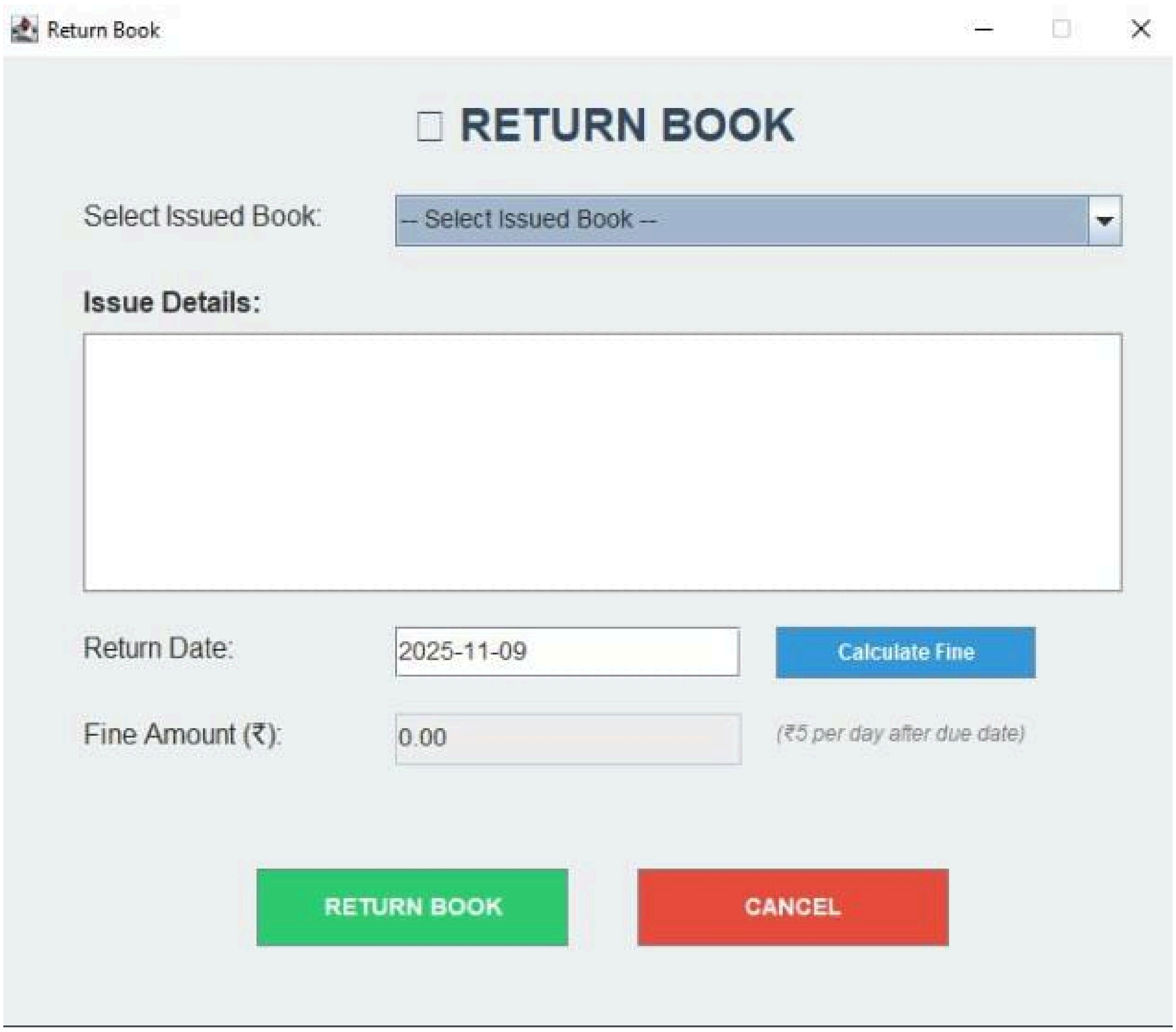
Select Book: -- Select Book --

Select Student: -- Select Student --

Issue Date: 2025-11-09

Due Date: 2025-11-23 *(Default: 14 days from today)*

Fig 5.7 RETURN BOOK FRAME


Return Book

□ RETURN BOOK

Select Issued Book: -- Select Issued Book --

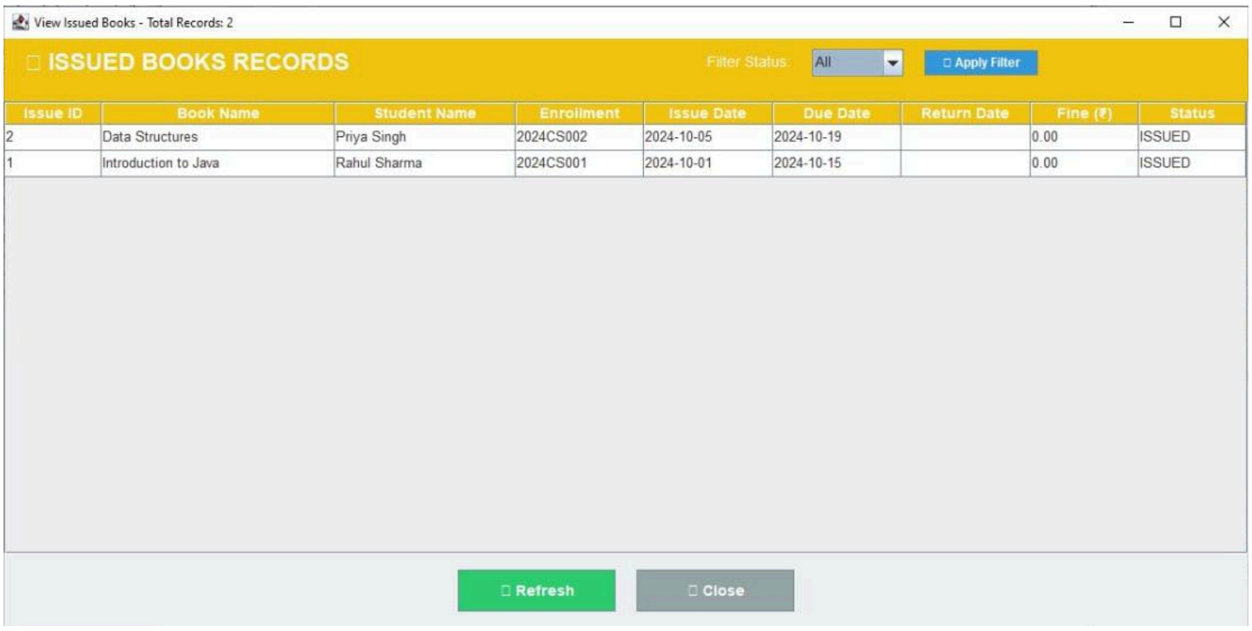
Issue Details:

Return Date: 2025-11-09

Fine Amount (₹): 0.00

(₹5 per day after due date)

Fig 5.8 VIEW ISSUED BOOKS FRAME



Issue ID	Book Name	Student Name	Enrollment	Issue Date	Due Date	Return Date	Fine (₹)	Status
2	Data Structures	Priya Singh	2024CS002	2024-10-05	2024-10-19		0.00	ISSUED
1	Introduction to Java	Rahul Sharma	2024CS001	2024-10-01	2024-10-15		0.00	ISSUED

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION

The Library Management System successfully achieves all objectives, providing a comprehensive, database-driven desktop solution for automating library operations. The use of JDBC with MySQL ensures robust data persistence, while transaction management guarantees data consistency across operations. The separation of concerns through distinct modules (Authentication, Book Management, Student Management, Transactions) makes the system maintainable and scalable.

Key achievements include:

- Intuitive GUI using Java Swing with gradient backgrounds and styled components
- Secure authentication system preventing unauthorized access
- Complete CRUD operations for books and students
- Automated stock management with real-time availability tracking
- Transaction-based book issuing and returning with rollback capability
- Automatic fine calculation for overdue returns
- Comprehensive search and filter capabilities across all modules
- Proper resource management with try-catch-finally blocks

The system demonstrates practical implementation of core software engineering principles including MVC architecture, database normalization, and exception handling.

FUTURE ENHANCEMENT

To evolve this project into a more comprehensive library management platform, the following future enhancements are recommended:

1. **Advanced Reporting Module:** Generate reports including:
   - Most issued books
   - Defaulter list (students with pending fines)
   - Category-wise book distribution
   - Monthly/yearly transaction statistics
   - Low stock alerts
2. **User Role Management:**
   - Multiple librarian accounts with different privilege levels
   - Admin user for system configuration
   - Read-only access for library assistants
3. **Book Reservation System:**
   - Allow students to reserve books that are currently issued
   - Notification system when reserved books become available
4. **Fine Payment Tracking:**
   - Record fine payments
   - Generate receipts
   - Outstanding fine reports
5. **Book Request Module:**
   - Students can request new book additions
   - Librarians can review and approve requests
6. **SMS/Email Notifications:**
   - Send due date reminders to students
   - Notify about overdue books
   - Alert on successful book issue/return
7. **Barcode Integration:**
   - Generate ISBN barcodes for books
   - Implement barcode scanner support for quick book identification
8. **Export Functionality:**
   - Export reports to PDF format
   - Export data to Excel for analysis
9. **Book Review System:**
   - Allow students to rate and review books
   - Display popular books based on ratings
10. **Multi-Language Support:**
    - Interface translation for regional languages
    - Unicode support for non-English book titles
11. **Web-Based Version:**
    - Convert to web application for remote access
    - Online catalog browsing for students
    - Mobile-responsive design
12. **Database Backup:**
    - Automated daily database backup

- Restore functionality

CHAPTER 7

REFERENCES

1. Oracle Java Swing Documentation
   https://docs.oracle.com/javase/tutorial/uiswing/
2. Java JDBC Tutorial
   https://docs.oracle.com/javase/tutorial/jdbc/
3. MySQL Official Documentation
   https://dev.mysql.com/doc/refman/8.0/en/
4. MySQL Connector/J Documentation
   https://dev.mysql.com/doc/connector-j/8.0/en/
5. Java PreparedStatement Documentation
   https://docs.oracle.com/javase/8/docs/api/java/sql/PreparedStatement.html
6. "Head First Java" by Kathy Sierra and Bert Bates
   O'Reilly Media, 2nd Edition
7. "Java: The Complete Reference" by Herbert Schildt
   McGraw Hill Education, 11th Edition
8. GeeksforGeeks - Java Swing Tutorial
   https://www.geeksforgeeks.org/java-swing-tutorial/
9. JavaTpoint - JDBC Tutorial
   https://www.javatpoint.com/java-jdbc
10. Stack Overflow - Java and MySQL Integration
    https://stackoverflow.com/questions/tagged/java+mysql