LIBRARY MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

SEVENHILLSVASA S (2303811724321100)

in partial fulfilment of requirements for the award of the course

CGB1201 – JAVA PROGRAMMING

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

DECEMBER, 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"LIBRARY MANAGEMENT SYSTEM"** is the bonafide work of **SEVENHILLSVASA S 2303811724321100** who carried out the project work during the academic year 2024 - 2025 under my supervision.

[illegible]

Signature

Dr. T. AVUDAIAPPAN M.E., Ph.D.,

HEAD OF THE DEPARTMENT,

Department of Artificial Intelligence,
K. Ramakrishnan College of Technology,
Samayapuram, Trichy-621 112.

S. Geetha

Signature

Mrs. S. GEETHA M.E.,

SUPERVISOR,

Department of Artificial Intelligence,
K. Ramakrishnan College of Technology,
Samayapuram, Trichy-621 112.

Submitted for the viva-voce examination held on 3.12.24.

S. Geetha

INTERNAL EXAMINER

[illegible]

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on "**LIBRARY MANAGEMENT SYSTEM**" is the result of original work done by me and best of my knowledge, similar work has not been submitted to "**ANNA UNIVERSITY CHENNAI**" for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfilment of the requirement of the award of the **CGB1201 – JAVA PROGRAMMING.**

[illegible]

Signature

SEVENHILLSVASA S

Place: Samayapuram

Date: 3/12/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and indebtedness to our institution, **“K. Ramakrishnan College of Technology (Autonomous)”,** for providing us with the opportunity to do this project.

I extend our sincere acknowledgment and appreciation to the esteemed and honourable Chairman, **Dr. K. RAMAKRISHNAN, B.E.,** for having provided the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding our project and offering an adequate duration to complete it.

I would like to thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.,** Principal, who gave the opportunity to frame the project to full satisfaction.

I thank **Dr.T. AVUDAIAPPAN, M.E., Ph.D.**, Head of the Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for providing her encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mrs. S. GEETHA M.E**., Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

I render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

MISSION OF THE INSTITUTION

- Be a centre of excellence for technical education in emerging technologies by exceeding the needs of industry and society.
- Be an institute with world class research facilities.
- Be an institute nurturing talent and enhancing competency of students to transform them as all- round personalities respecting moral and ethical values.

VISION AND MISSION OF THE DEPARTMENT

To excel in education, innovation and research in Artificial Intelligence and Data Science to fulfill industrial demands and societal expectations.

Mission 1: To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

Mission 2: To collaborate with industry and offer top-notch facilities in a conductive learning environment.

Mission 3: To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.

Mission 4: To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO 1: Compete on a global scale for a professional career in Artificial Intelligence and Data Science.

PEO 2: Provide industry-specific solutions for the society with effective communication and ethics

PEO 3: Hone their professional skills through research and lifelong learning initiatives.

PROGRAM OUTCOMES

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO 1:** Capable of working on data-related methodologies and providing industry focussed solutions.
- **PSO 2:** Capable of analysing and providing a solution to a given real-world problem by designing an effective program.

ABSTRACT

The Library Management System (LMS) is a cutting-edge software solution designed to modernize and streamline the complex operations of a library. Tailored to meet the needs of both librarians and members, the system automates critical tasks such as cataloguing books, registering users, and managing transactions including borrowing, returning, renewing, and reserving books. It provides a centralized platform that ensures efficiency, reduces manual labour, and improves overall user experience. Developed using Java, the LMS employs Object-Oriented Programming (OOP) principles to create a modular and extensible architecture. Key features include rolebased access, enabling librarians to manage book inventories and oversee user accounts, while members can search for books, check availability, and manage their borrowing activities. The use of advanced data structures and collections ensures fast and reliable processing of library data, even for large-scale implementations. This document outlines the system's conceptual design, the development methodology, and its modular structure. Each module is built to handle specific functionalities, from book management and user operations to transaction monitoring and reporting. The implementation phase highlights its success in reducing errors, enhancing user satisfaction, and providing scalability for future growth. The LMS thus serves as a transformative tool, fostering an efficient and user-friendly library experience while embracing the digital age.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Library Management System is a Java-based software solution designed to enhance the efficiency of library operations. By automating key tasks such as book lending, user management, and resource tracking, the system aims to reduce manual effort and improve the user experience. Its modular design ensures flexibility, while its scalability supports future enhancements to meet growing library demands. This report outlines the system's development process, key functionalities, and proposed advancements, emphasizing its role in modernizing library management.

1.2 OBJECTIVE

The Library Management System aims to offer an efficient solution to libraries by automating and optimizing various tasks traditionally managed manually. The system is designed to:

- Enhance library efficiency by automating tasks like catalog management, user registration, and book transactions.
- Facilitate book management by allowing librarians to easily add, edit, or remove books from the catalog.
- Streamline member interaction by enabling them to borrow, return, renew, and reserve books directly through the system.
- Introduce a reservation queue to ensure that members can request books that are currently unavailable, with automatic notification and issuance once the book is returned.

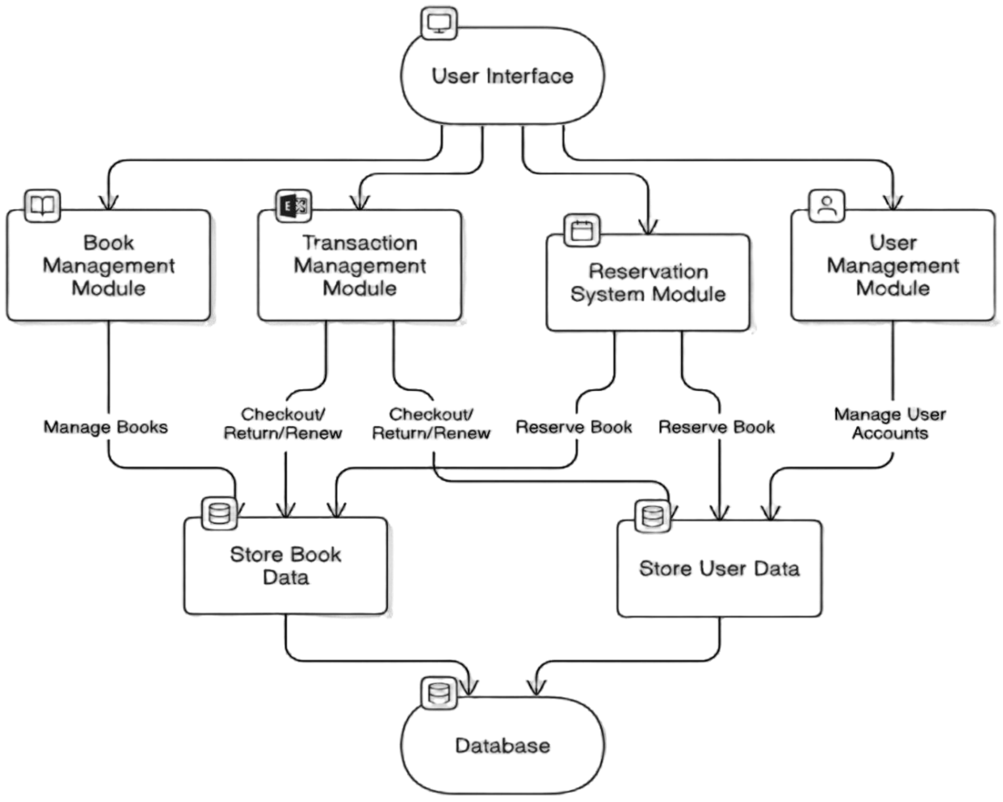
CHAPTER 2
PROJECT METHODOLOGY

2.1 PROPOSED WORK

The proposed work involves developing a Library Management System using Java to streamline the management of library operations. The system will feature user management to add and manage library members, along with role-based access control for security. It will also include an interactive book catalog for efficient book management and tracking, allowing users to search for, borrow, and return books.

The system's data will be handled using Array List and HashMap, with a graphical user interface (GUI) built using AWT. Future improvements may involve integrating a database, adding advanced search filters, and implementing overdue alerts for better management.

2.2 BLOCK DIAGRAM



CHAPTER 3
JAVA PROGRAMMING CONCEPTS

3.1 OBJECT OREIENTED PROGRAMMING

- Encapsulation: The use of private member variables like users, books, and borrowed Books to hide implementation details and provide controlled access through methods.
- Inheritance: Although not explicitly extending custom classes, the program uses inheritance by extending the Frame class from java.awt.
- Polymorphism: The ActionListener interface and its implementation in different buttons exhibit polymorphism.

2. Collections Framework

- HashMap<String, String> for managing users and borrowed books.
- Array List<String> for managing books and reserved books.

3. Event-Driven Programming

The program uses event handling (ActionListener and WindowAdapter) to respond to user interactions like button clicks and window events.

3.2 GUI COMPONENTS

The program heavily relies on Java AWT classes for creating graphical user interfaces.

Components used:

- Frame, Panel, Button, Label, Text Field
- Layout Managers:
- Flow Layout, Grid Layout, Border Layout

CHAPTER 4
MODULE DESCRIPTION

4.1 BOOK MANAGEMENT

The Book Management module provides functionalities for managing the library's catalog of books. Librarians can add new books to the catalog, remove outdated or unavailable books, and search for books based on their details (title, author, ISBN).

Key Methods:

- addBook (String title, String author, String isbn): Adds a new book to the system's catalog.
- removeBook (String isbn): Removes a book from the catalog using its ISBN.
- searchBooks (String query): Searches the catalog for books that match the provided query (e.g., title, author, ISBN).

4.2 USER MANAGEMENT

This module allows librarians to register new members or librarians, and it handles the management of user accounts. Each user has a specific role, which dictates their access rights.

Key Methods:

- registerUser(String userId, String name, Role role): Registers a new user, either as a member or a librarian.
- cancelMembership(String userId): Cancels a member's account if they are no longer active.

4.3 TRANSACTION MANAGEMENT

The Transaction Management module is responsible for tracking borrowing, returning, and renewing books. Members can borrow books as long as they are available and can return them once done. Librarians can renew books for members, ensuring that users can keep books for extended periods if needed.

Key Methods:

- checkOutBook(String userId, String isbn): Checks out a book for a member.
- returnBook(String userId, String isbn): Processes the return of a borrowed book.
- renewBook(String userId, String isbn): Renews a book, extending the borrowing period.

4.4 RESERVEATION SYSTEM

The Reservation System manages book reservations, ensuring that members can request books that are currently unavailable. Once the book is returned, the system automatically checks if there are any pending reservations, notifying the next member in the queue.

Key Methods:

- reserveBook(String userId, String isbn): Adds the member to the reservation queue for the specified book.

CHAPTER 5
CONCLUSION

The Library Management System is an efficient solution for managing books, users, and library operations. By incorporating features like online reservation, role based access, and streamlined transactions, it enhances the user experience and simplifies librarian tasks. Future Enhancements:

- Mobile Application: Develop a mobile app for managing reservations, transactions, and book searches on the go.
- Count Integration: Add a feature to track the total number of books, users, and active transactions in real time.
- Advanced Analytics: Implement analytics to provide insights on user preferences, book popularity, and library usage patterns.

REFERENCES:

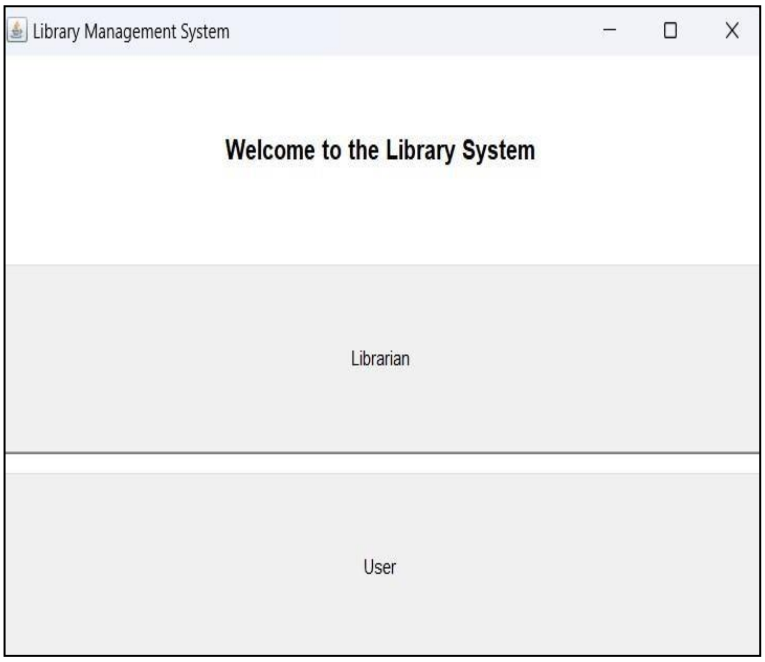
1. **Books**:
    - "Head First Java" by Kathy Sierra and Bert Bates.
    - "Java: The Complete Reference" by Herbert Schildt.
2. **Websites**:
    - Oracle Java Documentation
    - GeeksforGeeks
3. **YouTube**:
    - "Java Programming Tutorials" by Programming with Mosh.

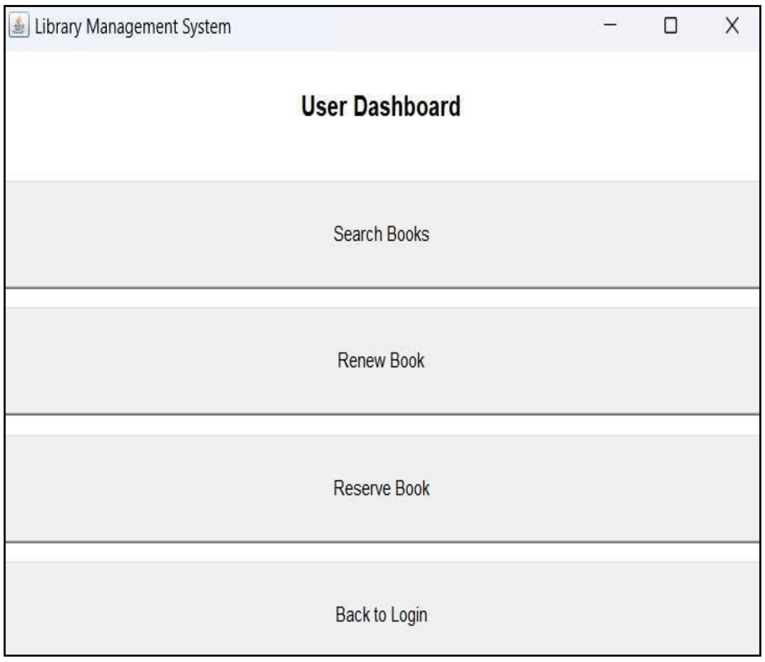
APPENDICES APPENDIX A – SOURCE CODE

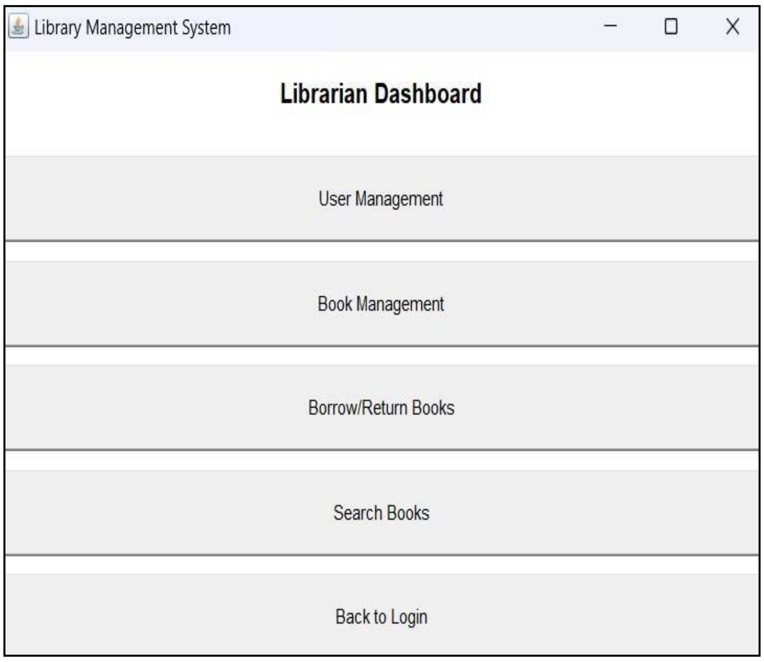
```
import java.awt.*; import java.awt.event.*;
import java.util.ArrayList; import
java.util.HashMap;
import javax.swing.JOptionPane;

public class LibraryManagementSystem extends Frame {

   private CardLayout cardLayout = new CardLayout(); // For switching between screens
private Panel mainPanel = new Panel(); // Central container for all screens     private
HashMap<String, String> users = new HashMap<>(); // For User Management     private
ArrayList<String> books = new ArrayList<>(); // For Book Management      private
HashMap<String, String> borrowedBooks = new HashMap<>(); // For Borrowing System
private ArrayList<String> reservedBooks = new ArrayList<>(); // For Reserved Books

   public LibraryManagementSystem() {
setTitle("Library Management System");
       setSize(600, 400);
       setLayout(new BorderLayout());

       mainPanel.setLayout(cardLayout);
add(mainPanel, BorderLayout.CENTER);

       createLoginPage();
createLibrarianDashboard();        createUserDashboard();
createUserManagementPage();
createBookManagementPage();
       createBorrowingSystemPage();

       // Adding dummy data to the system         addDummyData();

       addWindowListener(new WindowAdapter() {
public void windowClosing(WindowEvent we) {
               System.exit(0);
           }
       });

       setVisible(true);
   }

  private void addDummyData() {        // Add
some dummy users        users.put("admin",
"admin123");        users.put("john_doe",
"password");        users.put("alice_smith",
"alicepass"); // Add some dummy books
books.add("Java
Programming");      books.add("Data      Structures      and
Algorithms");            books.add("Artificial  Intelligence:  A
Modern  Approach");              books.add("Introduction  to
Databases");             books.add("Machine  Learning  with
Python");
   }

   private void createLoginPage() {        Panel
loginPanel = new Panel();
loginPanel.setLayout(new GridLayout(3, 1, 10, 10));

       Label lblWelcome = new Label("Welcome to the Library System", Label.CENTER);
lblWelcome.setFont(new Font("Arial", Font.BOLD, 16));

       Button btnLibrarian = new Button("Librarian");
       Button btnUser = new Button("User");

       btnLibrarian.addActionListener(e -> cardLayout.show(mainPanel, "LibrarianDashboard"));
btnUser.addActionListener(e -> cardLayout.show(mainPanel, "UserDashboard"));

       loginPanel.add(lblWelcome);
loginPanel.add(btnLibrarian);        loginPanel.add(btnUser);

       mainPanel.add(loginPanel, "LoginPage");
   }

   private void createLibrarianDashboard() {        Panel
librarianPanel = new Panel();
librarianPanel.setLayout(new GridLayout(6, 1, 10, 10));

       Label lblTitle = new Label("Librarian Dashboard", Label.CENTER);
lblTitle.setFont(new Font("Arial", Font.BOLD, 16));

       Button userMgmtBtn = new Button("User Management");
       Button bookMgmtBtn = new Button("Book Management");
       Button borrowMgmtBtn = new Button("Borrow/Return Books");
       Button searchBtn = new Button("Search Books");
       Button backBtn = new Button("Back to Login");

       userMgmtBtn.addActionListener(e -> cardLayout.show(mainPanel, "UserManagement"));
bookMgmtBtn.addActionListener(e -> cardLayout.show(mainPanel, "BookManagement"));
borrowMgmtBtn.addActionListener(e -> cardLayout.show(mainPanel, "BorrowingSystem"));
searchBtn.addActionListener(e -> searchBookLibrarian());
       backBtn.addActionListener(e -> cardLayout.show(mainPanel, "LoginPage"));

       librarianPanel.add(lblTitle);
librarianPanel.add(userMgmtBtn);
librarianPanel.add(bookMgmtBtn);
librarianPanel.add(borrowMgmtBtn);
librarianPanel.add(searchBtn);        librarianPanel.add(backBtn);
mainPanel.add(librarianPanel, "LibrarianDashboard"); }

   private void createUserDashboard() {
       Panel userPanel = new Panel();         userPanel.setLayout(new
GridLayout(5, 1, 10, 10));

       Label lblTitle = new Label("User Dashboard", Label.CENTER);         lblTitle.setFont(new
Font("Arial", Font.BOLD, 16));

       Button searchBtn = new Button("Search Books");
       Button renewBtn = new Button("Renew Book");
       Button reserveBtn = new Button("Reserve Book");
       Button backBtn = new Button("Back to Login");

       searchBtn.addActionListener(e -> searchBookUser());
renewBtn.addActionListener(e -> renewBook());         reserveBtn.addActionListener(e
-> reserveBook());          backBtn.addActionListener(e ->
cardLayout.show(mainPanel, "LoginPage"));

       userPanel.add(lblTitle);
userPanel.add(searchBtn);
userPanel.add(renewBtn);
userPanel.add(reserveBtn);         userPanel.add(backBtn);

       mainPanel.add(userPanel, "UserDashboard");
   }

   private void createUserManagementPage() {         Panel
userMgmtPanel = new Panel();
userMgmtPanel.setLayout(new FlowLayout());

       Label lblTitle = new Label("User Management", Label.CENTER);         lblTitle.setFont(new
Font("Arial", Font.BOLD, 16));

       Label lblName = new Label("Name:");
       Label lblPass = new Label("Password:");
       Label lblEmail = new Label("Email ID:");
       TextField txtName = new TextField(20);
       TextField txtPass = new TextField(20);         TextField
txtEmail = new TextField(20);
txtPass.setEchoChar('*');
       Button addMemberBtn = new Button("Add Member");
       Button deleteMemberBtn = new Button("Delete Member");
       Button backBtn = new Button("Back to Librarian Dashboard");

       addMemberBtn.addActionListener(e -> {
users.put(txtName.getText(), txtPass.getText());
         showMessage("Member Added: " + txtName.getText());
      });
      deleteMemberBtn.addActionListener(e -> {
if (users.remove(txtName.getText()) != null) {
            showMessage("Member Deleted: " + txtName.getText());
        } else {
             showMessage("Member Not Found: " + txtName.getText());
         } });

       backBtn.addActionListener(e -> cardLayout.show(mainPanel, "LibrarianDashboard"));

       userMgmtPanel.add(lblTitle);
userMgmtPanel.add(lblName);        userMgmtPanel.add(txtName);
userMgmtPanel.add(lblPass);        userMgmtPanel.add(txtPass);
userMgmtPanel.add(lblEmail);       userMgmtPanel.add(txtEmail);
userMgmtPanel.add(addMemberBtn);
userMgmtPanel.add(deleteMemberBtn);
userMgmtPanel.add(backBtn);

       mainPanel.add(userMgmtPanel, "UserManagement");
   }

   private void createBookManagementPage() {        Panel
bookMgmtPanel = new Panel();
bookMgmtPanel.setLayout(new FlowLayout());

       Label lblTitle = new Label("Book Management", Label.CENTER);
lblTitle.setFont(new Font("Arial", Font.BOLD, 16));

       Label lblBook = new Label("Book Name:");
       TextField txtBook = new TextField(20);
       Button addBtn = new Button("Add Book");
       Button deleteBtn = new Button("Delete Book");
       Button backBtn = new Button("Back to Librarian Dashboard");

       addBtn.addActionListener(e -> {
books.add(txtBook.getText());
          showMessage("Book Added: " + txtBook.getText());
       });

       deleteBtn.addActionListener(e -> {
if (books.remove(txtBook.getText())) {
            showMessage("Book Deleted: " + txtBook.getText());
         } else {
            showMessage("Book Not Found: " + txtBook.getText());
         }
       });

       backBtn.addActionListener(e -> cardLayout.show(mainPanel, "LibrarianDashboard"));
bookMgmtPanel.add(lblTitle);
bookMgmtPanel.add(lblBook);        bookMgmtPanel.add(txtBook);
bookMgmtPanel.add(addBtn);
bookMgmtPanel.add(deleteBtn);
bookMgmtPanel.add(backBtn); mainPanel.add(bookMgmtPanel,
"BookManagement"); }

   private void createBorrowingSystemPage() {         Panel
borrowPanel = new Panel();
borrowPanel.setLayout(new FlowLayout());

       Label lblTitle = new Label("Borrow/Return Books", Label.CENTER);
lblTitle.setFont(new Font("Arial", Font.BOLD, 16));

       Label lblBook = new Label("Book Name:");
       Label lblUser = new Label("Username:");
       TextField txtBook = new TextField(20);
       TextField txtUser = new TextField(20);

       Button borrowBtn = new Button("Borrow");
       Button returnBtn = new Button("Return");
       Button backBtn = new Button("Back to Librarian Dashboard");

       borrowBtn.addActionListener(e -> {
borrowedBooks.put(txtBook.getText(), txtUser.getText());            showMessage("Book
Borrowed: " + txtBook.getText());
       });

       returnBtn.addActionListener(e -> {             if
(borrowedBooks.remove(txtBook.getText()) != null) {
showMessage("Book Returned: " + txtBook.getText());
           } else {
               showMessage("Book Not Found: " + txtBook.getText());
           }
       });

       backBtn.addActionListener(e -> cardLayout.show(mainPanel, "LibrarianDashboard"));

       borrowPanel.add(lblTitle);
borrowPanel.add(lblBook);        borrowPanel.add(txtBook);
borrowPanel.add(lblUser);        borrowPanel.add(txtUser);
borrowPanel.add(borrowBtn);        borrowPanel.add(returnBtn);
       borrowPanel.add(backBtn);

       mainPanel.add(borrowPanel, "BorrowingSystem");
   }

   private void searchBookLibrarian() {
   String query = JOptionPane.showInputDialog("Enter book name to search:");
  if (query != null && !query.isEmpty()) {
boolean found = false;
```

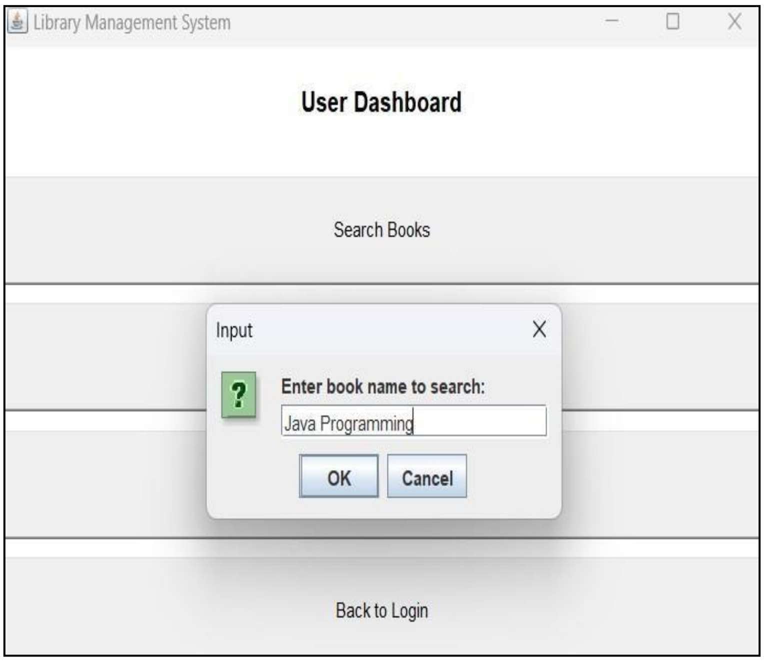
```
       // Convert query to lowercase for case-insensitive comparison         query
= query.trim().toLowerCase();
      // Loop through the books list to check for a match         for
(String book : books) {
           if (book.toLowerCase().contains(query)) {
        found = true;          break; // Exit the loop if a
        match is found
            }
       }

        if (found) {              showMessage("Book
found: " + query);
        } else {
           showMessage("Book not found.");
        }
    }
}

private void searchBookUser() {
    String query = JOptionPane.showInputDialog("Enter book name to search:");     if
(query != null && !query.isEmpty()) {
        boolean found = false;
        // Convert query to lowercase for case-insensitive comparison          query
= query.trim().toLowerCase();

        // Loop through the books list to check for a match          for
(String book : books) {
           if (book.toLowerCase().contains(query)) {
found = true;
              break; // Exit the loop if a match is found
           }
        }

        if (found) {
           showMessage("Book found: " + query);
        } else {
           showMessage("Book not found.");
        }
    }
}

    private void renewBook() {
showMessage("Book Renewed.");
    }

    private void reserveBook() {
        String bookName = JOptionPane.showInputDialog("Enter book name to reserve:");
if (bookName != null && !bookName.isEmpty()) {            reservedBooks.add(bookName);
           showMessage("Book Reserved: " + bookName);
        }
    }

  private void showMessage(String message) {        JOptionPane.showMessageDialog(this,
message);    }

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

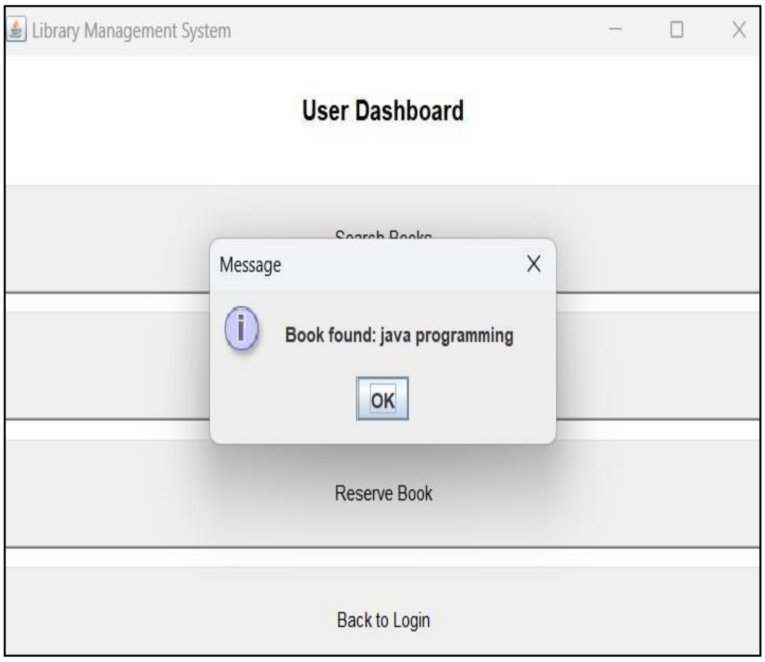
APPENDIX B - SCREENSHOTS



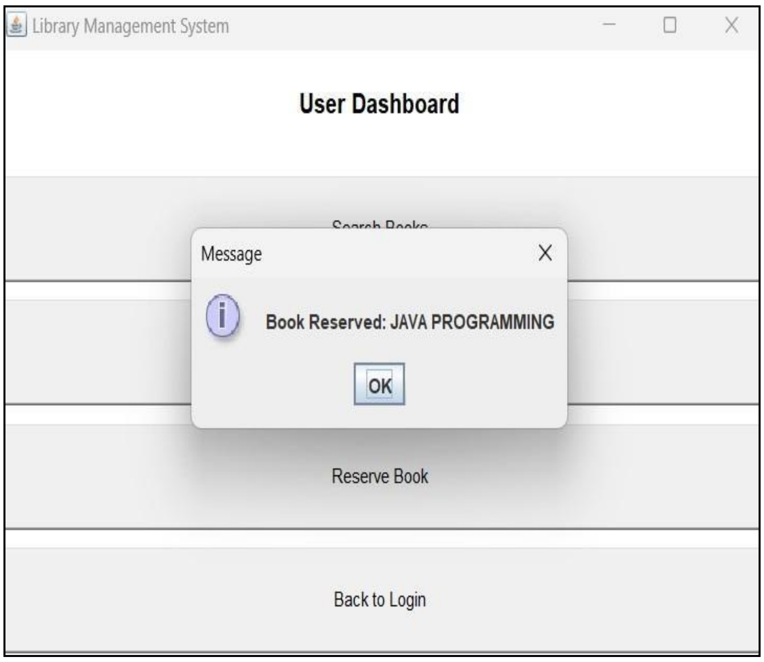



Library Management System
Librarian Dashboard
User Management
Book Management
Borrow/Return Books
Search Books
Back to Login


Library Management System
User Dashboard
Search Books
Input
Enter book name to search:
Java Programming
OK
Cancel
Back to Login



Book found: java programming



Book Reserved: JAVA PROGRAMMING