LIBRARY MANAGEMENT SYSTEM USING JAVAFX

A MINI PROJECT REPORT

Submitted by

Samitha K – 953624244051

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND BUSINESS SYSTEMS

RAMCO INSTITUTE OF TECHNOLOGY,

RAJAPALAYAM

NOVEMBER 2025

CS3391 – Object Oriented Programming

ABSTRACT

The Library Management System is a JavaFX-based application designed to simplify the process of managing library operations. It allows administrators to add, view, issue, and return books using a graphical interface. The program uses JavaFX for front-end development and core Java collections for backend data handling. It provides functionalities for secure login, book management, and user transaction tracking in an efficient and user-friendly environment.

INTRODUCTION

This project aims to develop a simple Library Management System using JavaFX. The system provides an easy-to-use interface for the librarian to manage books, including adding new books, viewing available books, issuing books to students, and returning books.

The system ensures smooth handling of library records without using any external database.

PROJECT DESCRIPTION

OVERVIEW

The Library Management System is designed to maintain records of books available in the library.

It also keeps track of issued and returned books. The program ensures secure login for admin access and enables library operations through simple button-based navigation.

MODULES

1. **Login Module** – Handles secure admin authentication.
2. **Add Book Module** – Allows adding new books to the system.
3. **View Books Module** – Displays all books and their statuses.
4. **Issue Book Module** – Issues a selected book to a student.
5. **Return Book Module** – Updates returned book information.
6. **Exit Module** – Exits the application.

TECHNOLOGY STACK

- **Programming Language**: Java

- **Framework**: JavaFX
- **IDE**: NetBeans
- **Libraries Uses**: javafx.application, javafx.scene, javafx.stage, javafx.control, javafx.geometry, java.util

PACKEGES USED

1. **javafx.application** - Used to launch the JavaFX application and handle the main application lifecycle.
2. **javafx.geometry** – Provides layout geometry classes such as Insets for spacing and padding.
3. **javafx.scene** – Contains core classes for creating GUI elements and scenes.
4. **javafx.scene.control** – Provides user interface controls like buttons, labels, text fields, and password fields.
5. **javafx.scene.layout** – Used to create layouts such as VBox, HBox, and GridPane for arranging GUI components.
6. **javafx.stage** – Handles the main window and scene management.
7. **java.util** – Includes data structures like ArrayList and HashMap used for storing and managing book details.

CODING

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.stage.Stage;
import java.util.*;
public class LibraryManagementSystem extends Application {
private final String ADMIN_USER = "admin";
private final String ADMIN_PASS = "1234";
private List<Book> books = new ArrayList<>();
private Map<String, String> issuedBooks = new HashMap<>();
@Override
public void start(Stage primaryStage) {
primaryStage.setTitle("Library Management System");
Label userLabel = new Label("Username:");
TextField userField = new TextField();
Label passLabel = new Label("Password:");
PasswordField passField = new PasswordField();
Button loginButton = new Button("Login");
```

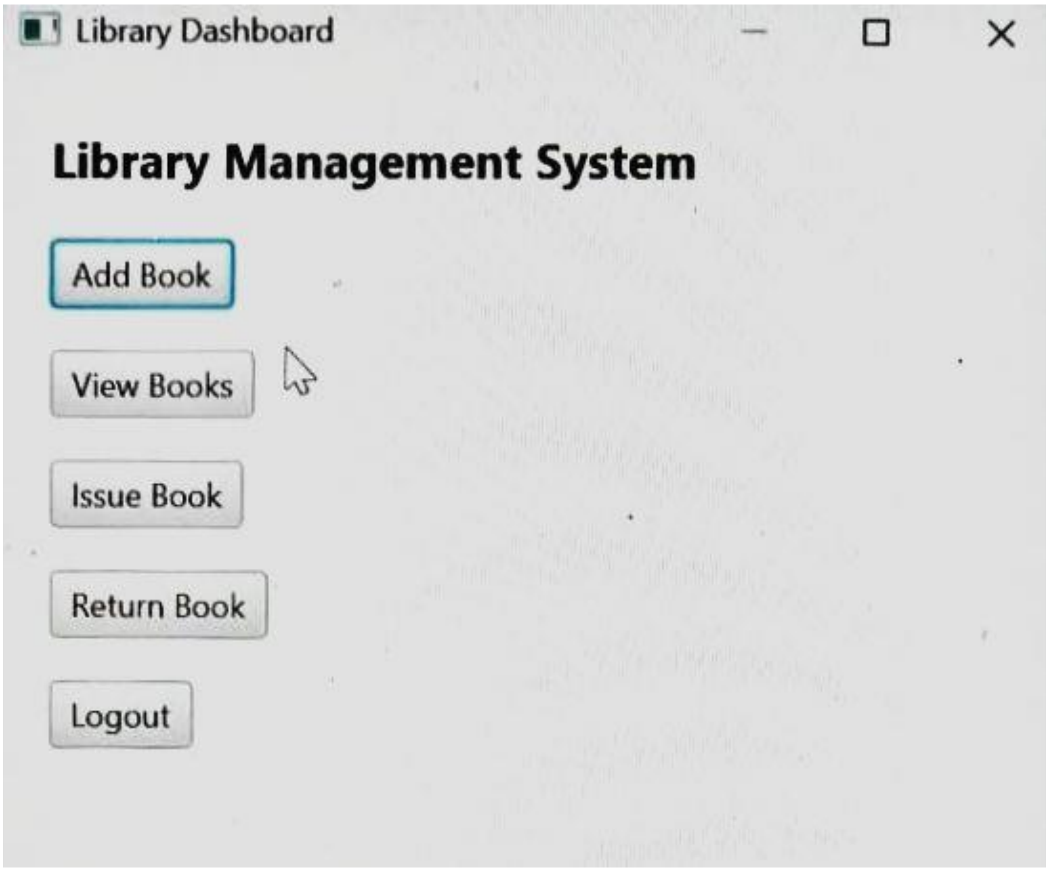
```
Label loginStatus = new Label();
VBox loginLayout = new VBox(10, userLabel, userField, passLabel, passField,
loginButton, loginStatus);
loginLayout.setPadding(new Insets(20));
Scene loginScene = new Scene(loginLayout, 300, 250);
Button addBtn = new Button("Add Book");
Button viewBtn = new Button("View Books");
Button issueBtn = new Button("Issue Book");
Button returnBtn = new Button("Return Book");
Button exitBtn = new Button("Exit");
VBox mainLayout = new VBox(10, addBtn, viewBtn, issueBtn, returnBtn,
exitBtn);
mainLayout.setPadding(new Insets(20));
Scene mainScene = new Scene(mainLayout, 300, 250);
loginButton.setOnAction(e -> {
String user = userField.getText();
String pass = passField.getText();
if (user.equals(ADMIN_USER) && pass.equals(ADMIN_PASS)) {
loginStatus.setText("Login Successful!");
primaryStage.setScene(mainScene);
} else {
loginStatus.setText("Invalid Credentials!");}});
addBtn.setOnAction(e -> addBookWindow());
viewBtn.setOnAction(e -> viewBooksWindow());
issueBtn.setOnAction(e -> issueBookWindow());
returnBtn.setOnAction(e -> returnBookWindow());
exitBtn.setOnAction(e -> primaryStage.close());
primaryStage.setScene(loginScene);
primaryStage.show();}
private void addBookWindow() {
Stage stage = new Stage();
stage.setTitle("Add Book");
TextField idField = new TextField();
idField.setPromptText("Book ID");
TextField titleField = new TextField();
titleField.setPromptText("Title");
TextField authorField = new TextField();
authorField.setPromptText("Author");
Button addButton = new Button("Add");
Label msg = new Label();
addButton.setOnAction(e -> {
String id = idField.getText();
String title = titleField.getText();
String author = authorField.getText();
if (!id.isEmpty() && !title.isEmpty() && !author.isEmpty()) {
books.add(new Book(id, title, author, "Available"));
msg.setText("Book added successfully!");
idField.clear(); titleField.clear(); authorField.clear();
} else {
msg.setText("Please fill all fields.");}});
VBox layout = new VBox(10, idField, titleField, authorField, addButton, msg);
layout.setPadding(new Insets(15));
stage.setScene(new Scene(layout, 300, 250));
stage.show();}
private void viewBooksWindow() {
Stage stage = new Stage();
stage.setTitle("View Books");
TextArea bookArea = new TextArea();
bookArea.setEditable(false);
StringBuilder sb = new StringBuilder();
if (books.isEmpty()) {
sb.append("No books available.");
} else {
for (Book b : books) {
sb.append("ID: ").append(b.id)
.append(" | Title: ").append(b.title)
.append(" | Author: ").append(b.author)
.append(" | Status: ").append(b.status).append("\n");}}
bookArea.setText(sb.toString());
VBox layout = new VBox(10, bookArea);
layout.setPadding(new Insets(10));
stage.setScene(new Scene(layout, 400, 300));
stage.show();}
private void issueBookWindow() {
Stage stage = new Stage();
stage.setTitle("Issue Book");
TextField bookIdField = new TextField();
bookIdField.setPromptText("Book ID");
TextField studentField = new TextField();
studentField.setPromptText("Student Name");
Button issueBtn = new Button("Issue");
Label msg = new Label();
issueBtn.setOnAction(e -> {
String bookId = bookIdField.getText();
String student = studentField.getText();
for (Book b : books) {
if (b.id.equals(bookId) && b.status.equals("Available")) {
b.status = "Issued";
issuedBooks.put(bookId, student);
msg.setText("Book issued to " + student);
return;}}
msg.setText("Book not found or already issued.");});
```

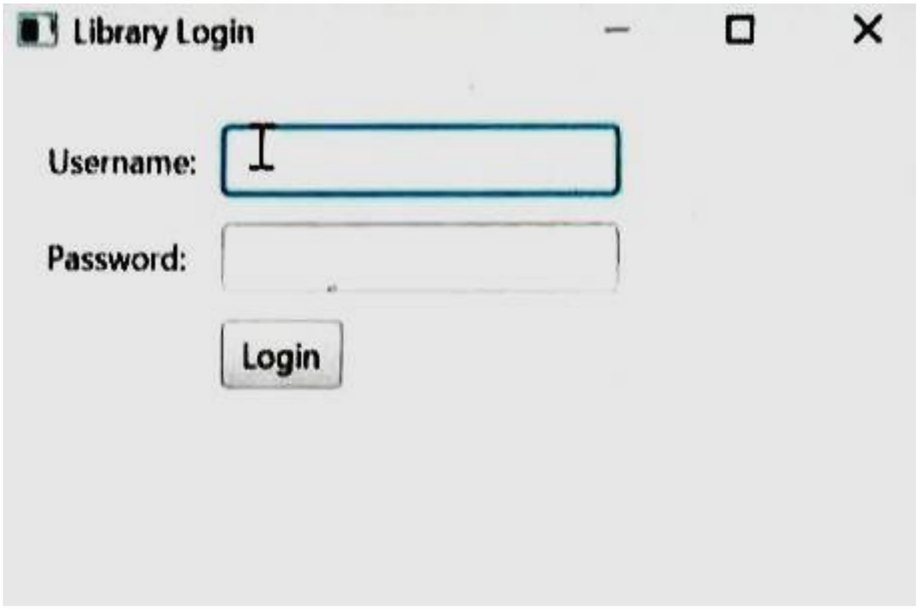
```
VBox layout = new VBox(10, bookIdField, studentField, issueBtn, msg);
layout.setPadding(new Insets(15));
stage.setScene(new Scene(layout, 300, 200));
stage.show();}
private void returnBookWindow() {
Stage stage = new Stage();
stage.setTitle("Return Book");
TextField bookIdField = new TextField();
bookIdField.setPromptText("Book ID");
Button returnBtn = new Button("Return");
Label msg = new Label();
returnBtn.setOnAction(e -> {
String bookId = bookIdField.getText();
if (issuedBooks.containsKey(bookId)) {
issuedBooks.remove(bookId);
for (Book b : books) {
if (b.id.equals(bookId)) {
b.status = "Available";}}
msg.setText("Book returned successfully!");
} else {
msg.setText("Book not issued or invalid ID.");}});
VBox layout = new VBox(10, bookIdField, returnBtn, msg);
layout.setPadding(new Insets(15));
stage.setScene(new Scene(layout, 300, 200));
stage.show();}
class Book {
String id, title, author, status;
Book(String id, String title, String author, String status) {
this.id = id;
this.title = title;
this.author = author;
this.status = status;}}
public static void main(String[] args) {
launch(args);}}
```

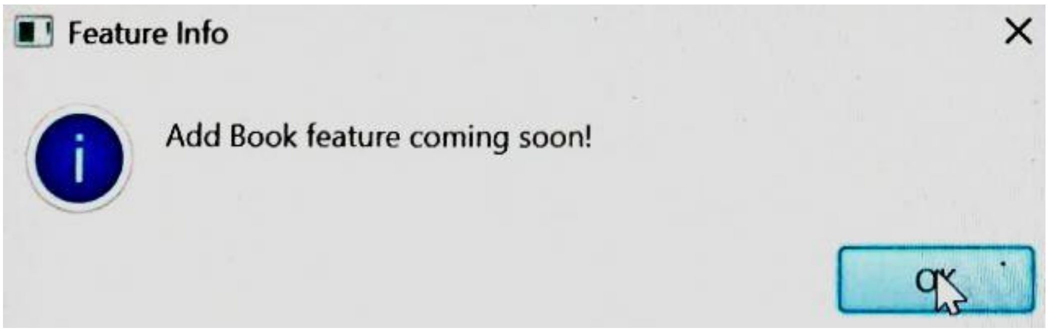
OUTPUT

1. **Login Page** – Allows admin to login with username and password.
2. **Main Menu** – Displays options: Add Book, View Books, Issue Book, Return Book, and Exit.
3. **Add Book** – Adds a new book to the system.
4. **View Books** – Lists all books with their status.
5. **Issue Book** – Issues a book to a student.
6. **Return Book** – Returns a book and updates its status to Available.

SCREENSHOT


Library Dashboard
Library Management System
Add Book
View Books
Issue Book
Return Book
Logout


Library Login
Username:
Password:
Login


Feature Info

Add Book feature coming soon!

CONCLUSION

The Library Management System using JavaFX is an efficient application for managing library operations.

It provides an interactive graphical interface for adding, viewing, issuing, and returning books.

The use of JavaFX ensures that the application is responsive and user-friendly.

This project enhances understanding of GUI-based application development and basic data management using Java collections.