LIBRARY MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

PRASHANTH REDDY C

NOVEMBER - 2024

ABSTRACT

The Library Management System is a console-based Java application that manages library operations such as adding and removing books, checking book availability, borrowing, and returning books. This system uses a simple text-based interface to facilitate user interaction. The project focuses on implementing key functionalities that ensure efficient and error-free library management. It employs core Java concepts, such as object-oriented programming (OOP), exception handling, and collections framework, to build a scalable and maintainable application. The system ensures robust error handling for invalid operations, such as attempting to borrow unavailable books or returning non-borrowed books.

The project is designed to provide a clear understanding of the basics of software design and development, demonstrating Java's capabilities in building functional and modular applications.

.

INTRODUCTION

Libraries are essential for educational institutions, public services, and private collections. Efficient management of library resources ensures that users can borrow and return books smoothly, and administrators can track book availability without hassle.

This Library Management System aims to simplify these tasks by providing an easy-to-use, console-based application. Users can interact with the system to perform common library operations such as adding new books, borrowing books, checking their availability, and managing borrower records.

By using Java as the programming language, this project showcases the use of fundamental programming principles and object-oriented design to solve real-world problems.

OBJECTIVE

The primary objectives of this project are:

1. To develop a system for efficient library management.
2. To provide functionalities for adding, removing, and searching for books.
3. To manage borrower records, including borrowing and returning books.
4. To ensure error-free operation through robust exception handling.
5. To design a modular, maintainable application using object-oriented principles.

LITERATURE REVIEW

Library management has been a vital area of software development, with numerous systems built to automate library operations. Traditional systems often relied on paper records, which were error-prone and inefficient. With the advent of computer systems, various digital library management solutions emerged.

Studies show that an effective library management system must incorporate:

1. Data Consistency: Ensuring accurate tracking of book and borrower records.
2. Error Handling: Graceful handling of invalid operations.
3. Scalability: The ability to handle a growing number of books and users.

The Library Management System developed in this project addresses these needs using Java's powerful features, such as the collections framework for data storage and exception handling for reliability.

SYSTEM DESIGN AND IMPLEMENTATION

Architecture

The system follows a three-tier architecture:

1. Data Layer: Stores book and borrower information using Java's HashMap and ArrayList.
2. Logic Layer: Handles operations such as adding, removing, borrowing, and returning books.
3. Presentation Layer: Provides a simple console-based interface for user interaction.

Components

1. Library: A class that manages the collection of books and borrower records.
2. Book: A class that represents individual books, including attributes such as title, author, and borrow status.
3. Main Program: A driver class that provides a menu-driven interface for user interaction.

Features

1. Add and Remove Books: Allows the librarian to manage the library's inventory.
2. Check Book Availability: Enables users to verify if book is available for borrowing.
3. Borrow and Return Books: Tracks borrowing and returning activities, ensuring accurate borrower records.
4. View All Books: Displays all books in the library with their details.
5. Error Handling: Handles invalid inputs and operations gracefully

CODE

LibraryManagementSystem.java

The main class provides a menu-based interface and calls appropriate methods from the Library class based on user input.

```java
import java.util.*;

public class LibraryManagementSystem {

    private static final Scanner scanner = new Scanner(System.in);
    private static final Library library = new Library();

    public static void main(String[] args) {
        System.out.println("Welcome to the Library Management System!");

        while (true) {
            showMenu();
            int choice = getUserChoice();
            handleUserChoice(choice);
        }
    }

    private static void showMenu() {
        System.out.println("\n==== Library Menu ====");
        System.out.println("1. Add a Book");
        System.out.println("2. Remove a Book");
        System.out.println("3. Check Book Availability");
        System.out.println("4. Borrow a Book");
        System.out.println("5. Return a Book");
        System.out.println("6. View All Books");
        System.out.println("7. Exit");
        System.out.println("======================");
        System.out.print("Enter your choice: ");
    }

    private static int getUserChoice() {
        try {
            return Integer.parseInt(scanner.nextLine());
        } catch (NumberFormatException e) {
            System.out.println("Invalid input. Please enter a number.");
            return -1;
        }
    }
    private static void handleUserChoice(int choice) {
        switch (choice) {
            case 1:
                addBook();
                break;
            case 2:
                removeBook();
                break;
            case 3:
                checkBookAvailability();
                break;
            case 4:
                borrowBook();
                break;
            case 5:
                returnBook();
                break;
            case 6:
                library.viewAllBooks();
                break;
            case 7:
                System.out.println("Exiting the system. Goodbye!");
                System.exit(0);
                break;
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    }

    private static void addBook() {
        System.out.print("Enter Book Title: ");
        String title = scanner.nextLine();
        System.out.print("Enter Author: ");
        String author = scanner.nextLine();
        library.addBook(new Book(title, author));
    }

    private static void removeBook() {
        System.out.print("Enter Book Title to Remove: ");
        String title = scanner.nextLine();
        library.removeBook(title);
    }
    private static void checkBookAvailability() {
        System.out.print("Enter Book Title to Check: ");
        String title = scanner.nextLine();
        library.checkBookAvailability(title);
    }
    private static void borrowBook() {
        System.out.print("Enter Book Title to Borrow: ");
        String title = scanner.nextLine();
        System.out.print("Enter Your Name: ");
        String borrower = scanner.nextLine();
        library.borrowBook(title, borrower);
    }

    private static void returnBook() {
        System.out.print("Enter Book Title to Return: ");
        String title = scanner.nextLine();
        library.returnBook(title);
    }
}
```

Library.java

The Library class manages the collection of books and their associated operations. It uses Java's HashMap for efficient data retrieval and ArrayList for maintaining borrower records.

```
import java.util.*;

class Library {
    private final Map<String, Book> books;
    public Library() {
        this.books = new HashMap<>();
    }

    public void addBook(Book book) {
        if (books.containsKey(book.getTitle())) {
            System.out.println("Book already exists in the library.");
        } else {
            books.put(book.getTitle(), book);
            System.out.println("Book added successfully!");
        }
    }
    public void removeBook(String title) {
        if (books.remove(title) != null) {
            System.out.println("Book removed successfully!");
        } else {
            System.out.println("Book not found in the library.");
        }
    }
```

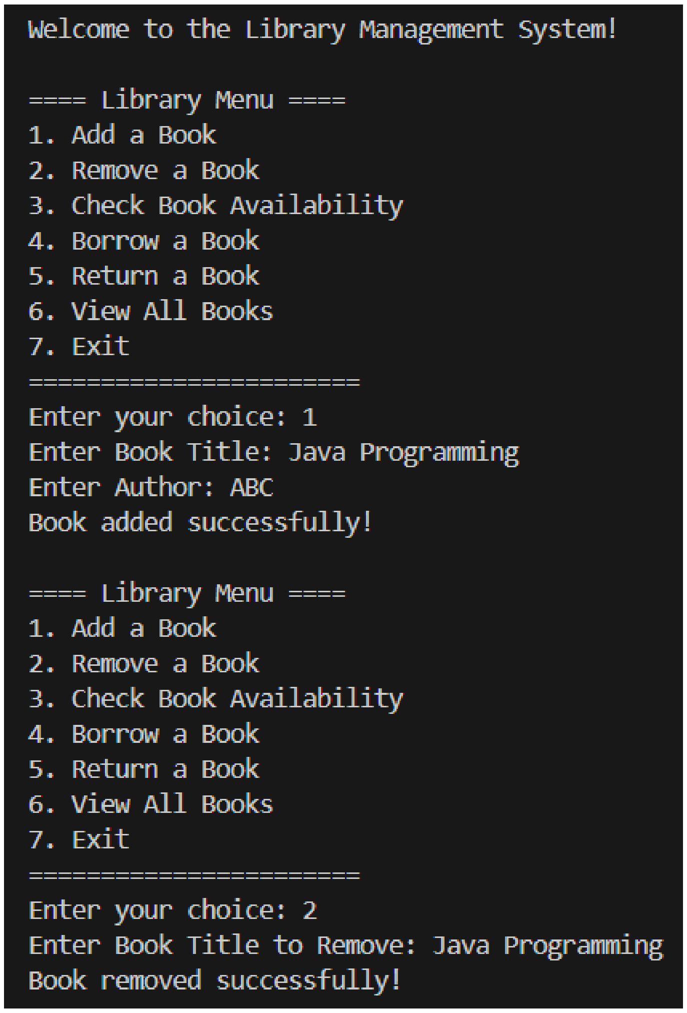
```java
    public void checkBookAvailability(String title) {
        if (books.containsKey(title)) {
            System.out.println("The book \"" + title + "\" is available.");
        } else {
            System.out.println("The book \"" + title + "\" is not available.");
        }
    }

    public void borrowBook(String title, String borrower) {
        Book book = books.get(title);
        if (book == null) {
            System.out.println("Book not found.");
        } else if (book.isBorrowed()) {
            System.out.println("Book is currently borrowed by " + book.getBorrower() + ".");
        } else {
            book.borrowBook(borrower);
            System.out.println("Book borrowed successfully by " + borrower + ".");
        }
    }

    public void returnBook(String title) {
        Book book = books.get(title);
        if (book == null) {
            System.out.println("Book not found.");
        } else if (!book.isBorrowed()) {
            System.out.println("This book was not borrowed.");
        } else {
            book.returnBook();
            System.out.println("Book returned successfully!");
        }
    }

    public void viewAllBooks() {
        if (books.isEmpty()) {
            System.out.println("No books available in the library.");
        } else {
            System.out.println("\n=== Books in the Library ===");
            for (Book book : books.values()) {
                System.out.println(book);
            }
        }
    }
}
```

Library.java

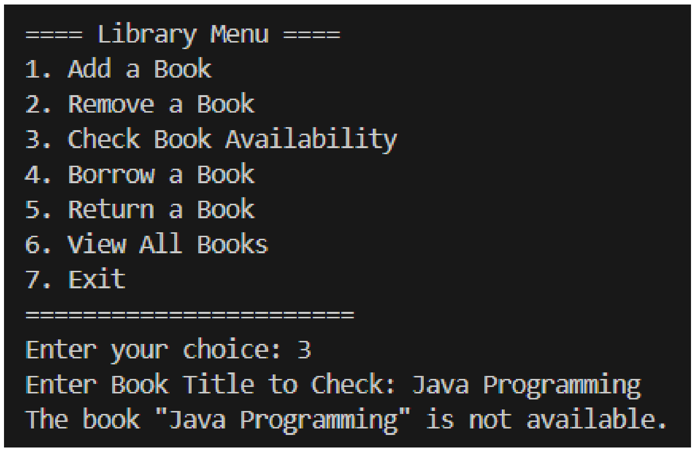
```
class Book {
    private final String title;
    private final String author;
    private boolean isBorrowed;
    private String borrower;

    public Book(String title, String author) {
        this.title = title;
        this.author = author;
        this.isBorrowed = false;
        this.borrower = null;
    }

    public String getTitle() {
        return title;
    }

    public boolean isBorrowed() {
        return isBorrowed;
    }

    public String getBorrower() {
        return borrower;
    }

    public void borrowBook(String borrower) {
        this.isBorrowed = true;
        this.borrower = borrower;
    }

    public void returnBook() {
        this.isBorrowed = false;
        this.borrower = null;
    }

    @Override
    public String toString() {
        return String.format("Title: %s | Author: %s | Borrowed: %s", title, author, isBorrowed ? "Yes, by "
+ borrower : "No");
    }
}
```

RESULTS

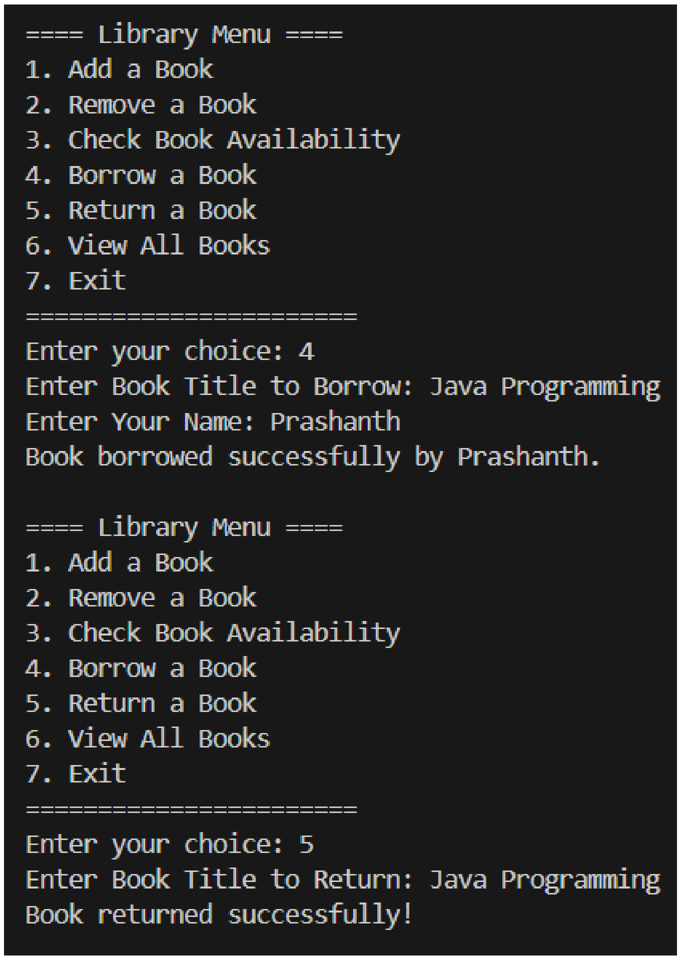
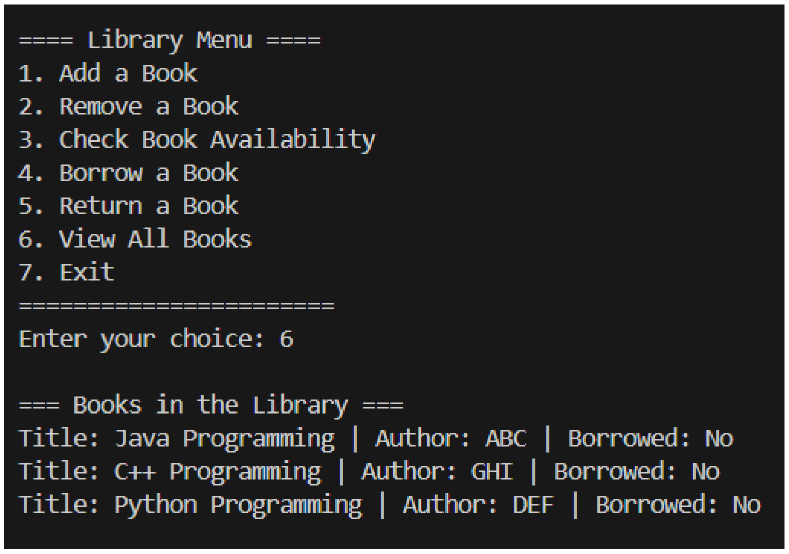
Screenshots:



```
Welcome to the Library Management System!

==== Library Menu ====
1. Add a Book
2. Remove a Book
3. Check Book Availability
4. Borrow a Book
5. Return a Book
6. View All Books
7. Exit
=======================
Enter your choice: 1
Enter Book Title: Java Programming
Enter Author: ABC
Book added successfully!

==== Library Menu ====
1. Add a Book
2. Remove a Book
3. Check Book Availability
4. Borrow a Book
5. Return a Book
6. View All Books
7. Exit
=======================
Enter your choice: 2
Enter Book Title to Remove: Java Programming
Book removed successfully!
```



```
==== Library Menu ====
1. Add a Book
2. Remove a Book
3. Check Book Availability
4. Borrow a Book
5. Return a Book
6. View All Books
7. Exit
=======================
Enter your choice: 3
Enter Book Title to Check: Java Programming
The book "Java Programming" is not available.
```


```
==== Library Menu ====
1. Add a Book
2. Remove a Book
3. Check Book Availability
4. Borrow a Book
5. Return a Book
6. View All Books
7. Exit
=======================
Enter your choice: 4
Enter Book Title to Borrow: Java Programming
Enter Your Name: Prashanth
Book borrowed successfully by Prashanth.

==== Library Menu ====
1. Add a Book
2. Remove a Book
3. Check Book Availability
4. Borrow a Book
5. Return a Book
6. View All Books
7. Exit
=======================
Enter your choice: 5
Enter Book Title to Return: Java Programming
Book returned successfully!
```

```
==== Library Menu ====
1. Add a Book
2. Remove a Book
3. Check Book Availability
4. Borrow a Book
5. Return a Book
6. View All Books
7. Exit
=======================
Enter your choice: 6

=== Books in the Library ===
Title: Java Programming | Author: ABC | Borrowed: No
Title: C++ Programming | Author: GHI | Borrowed: No
Title: Python Programming | Author: DEF | Borrowed: No
```

CONCLUSION

The Library Management System successfully demonstrates how core Java programming principles can be applied to solve real-world problems. The project provides a fully functional library system with all necessary operations and robust error handling. By focusing on modular design, the application is both scalable and maintainable, laying a foundation for future enhancements such as adding a graphical user interface (GUI) or database integration.

REFERENCES

[1] Oracle Documentation on Java.

[2] Effective Java, 3rd Edition by Joshua Bloch

[3] Java: The Complete Reference by Herbert Schildt