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

LIBRARY MANAGEMENT SYSTEM

MINI PROJECT REPORT

Submitted by

JISHANTHI R	**231801072**
MADHUSHA HARINI **MANIKANTAN GEETHA**	**231801091**

CS23333 OBJECT ORIENTED PROGRAMMING

USING JAVA

Department of Artificial Intelligence and Data Science

Rajalakshmi Engineering College Thandalam

BONAFIDE CERTIFICATE

Certified that this project report "**LIBRARY MANAGEMENT SYSTEM**" is Bonafide work of "**JISHANTHI R** (231801072), **MADHUSHA HARINI MANIKANTAN GEETHA** (231801091)"who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE

MrS. RENUKA DEVI
Assistant Professor,Artificial Intelligence Science,Rajalakshmi Engineering College (Autonomous),Thandalam,Channai-602105

SIGNATURE

Dr.GNANASEKAR J M
Head of the Department,Artificial intelligence and data Science,Rajalakshmi Engineering College(Autonomous),Chennai-602105

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

The Library Management System (LMS) is a simple yet effective console-based application developed in Java that facilitates the management of a library's operations. The primary goal of the system is to streamline the process of managing books, users, and transactions related to borrowing and returning books. The system allows for the creation of a collection of books and users, supports book borrowing and return functionalities, and provides an easy way to search and display available books and registered users. The project implements object-oriented programming (OOP) principles by utilizing classes such as `Book`, `User`, and `Library`, each encapsulating their respective attributes and methods. The `Library` class acts as the core of the system, managing books and users, while `Book` and `User` classes model the entities. The program provides a menu-driven interface that enables users to interact with the system, performing actions like viewing available books, borrowing and returning books, and managing user details. This project demonstrates key concepts such as encapsulation, data abstraction, and collections management (using ArrayLists) and is designed for scalability, with potential enhancements like database integration, user authentication, or a graphical user interface (GUI). The system provides an educational foundation for learning Java programming and system design,

APPENDIX

LIST OF CONTENT

CHAPTER – 1

INTRODUCTION

1.1 OBJECTIVE

The objective of the Library Management System is to develop a robust and efficient console based application in Java that automates and simplifies essential library operations, such as the management of books and users. This system is designed to provide a seamless experience for library administrators and users by offering key functionalities, including adding books, borrowing and returning books, and tracking user transactions. By implementing object-oriented programming (OOP) principles, the system encapsulates core features within distinct classes (e.g., `Book`, `User`, `Library`), allowing for modular, maintainable, and scalable code.

The application aims to streamline the day-to-day activities of a library, enabling administrators to manage the library's collection easily and users to interact with the system through a straightforward interface. It facilitates accurate tracking of book availability, reducing errors in managing borrowed items and providing real-time status of the library's inventory. Additionally, the system is built to be extensible, with the potential for future upgrades such as database integration for persistent storage, user authentication for secure access, and the development of a graphical user interface (GUI) for enhanced usability.

This project also serves as an educational tool for understanding the fundamentals of Java programming, demonstrating key concepts such as inheritance, encapsulation, and collections management through `ArrayList`. It provides a practical approach to solving real-world problems, offering a solid foundation for learning system design, data abstraction, and command-line interaction. The objective is not only to solve the immediate library management problem but also to provide a flexible and scalable system that can evolve with the needs of its users and developers**.**

CHAPTER – 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

In many libraries today, management of books and user activities is often done using either manual paper-based methods or basic digital tools such as spreadsheets. In manual systems, librarians record transactions like book borrowing and returning in physical logbooks or cards. This approach is prone to human error, labour-intensive, and inefficient when tracking book availability or maintaining user histories. It also creates challenges in keeping records accurate, leading to problems like lost or misplaced books, as well as difficulty in retrieving information about a particular book or user. The lack of automation also means that tasks like checking book availability, adding new entries, or updating borrowed books must be done manually, consuming significant time and effort. On the other hand, basic digital systems, such as those using simple spreadsheets, partially improve over manual methods but still have several limitations. While they offer some level of data organization, they require frequent manual updates, leading to a high risk of input errors. Searching for books or user details in large spreadsheets can be slow and cumbersome. Additionally, these systems do not offer real-time tracking, meaning that the status of books is often outdated, which can create confusion regarding book availability. These systems also lack centralized control, making it difficult for multiple users to access or update information simultaneously. As libraries grow in size, both manual and basic digital systems become inefficient and unable to handle the increased volume of data and transactions**.**

2.2 PROPOSED SYSTEM

The proposed Library Management System (LMS) is designed to overcome the inefficiencies of existing manual and basic digital systems by providing an automated, streamlined, and scalable solution using Java. This system allows for real-time management of books and users, automating key tasks such as adding books, borrowing, returning, and tracking book availability. With a centralized in-memory data structure, all book and user transactions are updated instantly, ensuring accurate and current information. The system's automated nature reduces human error, speeds up transactions, and provides quick access to data through efficient search functionalities. In addition to real-time tracking, the proposed system offers scalability, making it capable of handling a growing number of books and users without performance issues. Its modular design makes it extensible, allowing for future integration of advanced features like database support, user authentication, or graphical user interfaces (GUIs) for enhanced user interaction. This means that the system can evolve with the library's needs, providing a long-term solution that enhances productivity and reduces manual labor. Furthermore, the system's ability to provide backups and maintain data integrity ensures that records are preserved and easily recoverable, mitigating the risks of data loss common in manual systems. Overall, the proposed system significantly improves efficiency, accuracy, and user experience in library management**.**

CHAPTER – 3

SYSTEM REQUIREMENT

3.1 HARDWARE REQUIREMENT

Hard Disk : 10GB and Above

RAM : 512MB and Above

Processor : Core i5

Speed : 2.2GHz

3.2 SOFTWARE REQUIREMENT

Platform : JAVA

Front End Tools : Visual Studio 2010

Back End Database : SQL Server 2008

CHAPTER – 4

SYSTEM IMPLEMENTATION

4.1 MODULE LIST

1. Book Management Module
2. Student Module
3. Library Module
4. Main UI Module

4.2 MODULE DESCRIPTION

4.2.1 BOOK MANAGEMENT MODULE

The Book module represents individual books in the library. It stores details like book ID, title, author, and the book's availability status (issued or not). This module provides methods to issue or return a book and display book details

4.2.2 STUDENT MODULE

The student module manages student information, such as their ID, name, and the list of books they have issued. It allows students to issue or return books and keeps track of all books issued to a particular student

4.2.3 LIBRARY MODULE

The library module is the core of the system, managing the library's collection of books and handling book-related operations such as adding, removing, viewing, issuing, and returning books. It also keeps track of students and integrates the functionality of the other modules.

4.24 MAIN UI MODULE

The Main module provides a simple console-based user interface for interacting with the library system. It allows users to perform tasks like adding books and students, viewing available books, issuing and returning books, all through a menu-driven system.

CHAPTER – 5

TESTING

5.1 TESTING OBJECTIVE

The objective of testing the Library Management System is to ensure that the application functions correctly, adheres to specified requirements, and provides a reliable and user-friendly experience for both library administrators and users. This involves verifying the functionality of all modules, such as user registration, login, book management, and borrowing processes, to ensure they work without errors. Usability assessments will evaluate the intuitiveness of the user interface, while performance evaluations will confirm the system's capability to handle multiple simultaneous users effectively. Security testing will identify vulnerabilities and ensure that user data is protected against unauthorized access.

5.2 TEST CASES

TEST CASE NAME	TEST CASE DESCRIPTION	EXPECTED OUTCOME	ACTUAL OUTCOME
Add a New Book	Add a book with ID 15002, Title "To Kill a Mockingbird," and author "Harper Lee"	The book should be added to the library and visible in the list of books.	Book "To Kill s Mockingbird" was added and appears in the list.
View All Books	View the list of all available books in the library.	All added books should be displayed with their details (ID, title, author).	All added books are displayed correctly with appropriate details.
Issue a Book	issue book ID 15002 to student ID 001.	Book 15002 should be issued to student 001, and its status updated.	Book 15002 was successfully issued to student 001.
Return a Book	Return the issued book 15002 by student 001.	Book 15002 should be returned, and its availability status updated.	Book 15002 was returned and is available for issuing again.
Add a student	Add a student with ID 101 to the library.	Student with ID 101 should be added to the library.	Student with ID 101 was successfully added.

CHAPTER – 6

CONCLUSION

In conclusion, the Library Management System (LMS) project represents a comprehensive and innovative solution to the challenges faced by traditional library management methods, which often rely on manual processes or basic digital tools. By leveraging modern technology and a modular design approach, the LMS effectively streamlines various library operations, enhancing efficiency and user experience. The system's key functionalities, including user registration, secure authentication, book management, and transaction tracking, create a cohesive environment where librarians can easily manage resources while providing library members with a straightforward way to borrow and return books. Moreover, the LMS facilitates real-time tracking of book availability, significantly reducing the risk of errors associated with manual record-keeping and improving the accuracy of user borrowing histories. Comprehensive testing has been conducted to ensure the system's reliability, usability, and security, validating that it meets the diverse needs of its users. The incorporation of features such as error handling, data validation, and user role differentiation further enhances its functionality, making it suitable for both librarians and library patrons. Additionally, the design allows for future scalability and enhancements, ensuring that the system can evolve alongside changing library demands and technological advancements. The successful implementation of the LMS not only empowers libraries to better serve their communities but also positions them to adapt to the increasingly digital landscape of information management. By adopting this system, libraries can enhance their operational capabilities, improve user satisfaction, and ultimately foster a more engaging and accessible environment for learning and discovery. The LMS stands as a testament to the potential of technology in transforming traditional practices, paving the way for modern libraries to thrive in the 21st century and beyond.

APPENDIX 1

CODING

Book.java

```
public class Book {
    private int bookId;
    private String title;
    private String author;
    private boolean isIssued;
    public Book(int bookId, String title, String author) {
        this.bookId = bookId;
        this.title = title;
        this.author = author;
        this.isIssued = false;
    }
    public String getTitle() { return title; }
    public String getAuthor() { return author; }
    public int getBookId() { return bookId; }
    public boolean isIssued() { return isIssued; }

    public void issueBook() { this.isIssued = true; }
    public void returnBook() { this.isIssued = false; }
    public String toString() {
        return "ID: " + bookId + ", Title: " + title + ", Author: " + author + ", Issued: " + isIssued;
    }
}
```

Student.java

```
import java.util.ArrayList;
public class Student {
    private int studentId;
    private String name;
    private ArrayList<Book> issuedBooks;
    public Student(int studentId, String name) {
        this.studentId = studentId;
        this.name = name;
        this.issuedBooks = new ArrayList<>();
    }
    public String getName() { return name; }
    public int getStudentId() { return studentId; }
    public void issueBook(Book book) {
        issuedBooks.add(book);
        book.issueBook();
    }
    public void returnBook(Book book) {
        issuedBooks.remove(book);
        book.returnBook();
    }
    public ArrayList<Book> getIssuedBooks() {
        return issuedBooks;
    }
}
```

Library.java

```
import java.util.ArrayList;
import java.util.HashMap;
public class Library {
    private ArrayList<Book> books;
    private HashMap<Integer, Student> students;
    public Library() {
        books = new ArrayList<>();
        students = new HashMap<>();
    }
    public void addBook(Book book) {
        books.add(book);
    }
      public void removeBook(int bookId) {
        books.removeIf(book -> book.getBookId() == bookId);
    }
    public void viewBooks() {
        for (Book book : books) {
            System.out.println(book);
        }
    }
    public void issueBook(int studentId, int bookId) {
        Student student = students.get(studentId);
        for (Book book : books) {
            if (book.getBookId() == bookId && !book.isIssued()) {
                student.issueBook(book);
                System.out.println("Book issued to student: " + student.getName());
                return;
```

```
            }
        }
        System.out.println("Book is either not available or already issued.");
    }
    public void returnBook(int studentId, int bookId) {
        Student student = students.get(studentId);
        for (Book book : student.getIssuedBooks()) {
            if (book.getBookId() == bookId) {
                student.returnBook(book);
                System.out.println("Book returned by student: " + student.getName());
                return;
            }
        }
        System.out.println("This book is not issued to the student.");
    }
    public void addStudent(Student student) {
        students.put(student.getStudentId(), student);
    }
}
```

Main.java

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Library library = new Library();
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("\nLibrary Management System");
            System.out.println("1. Add Book");
            System.out.println("2. View Books");
```

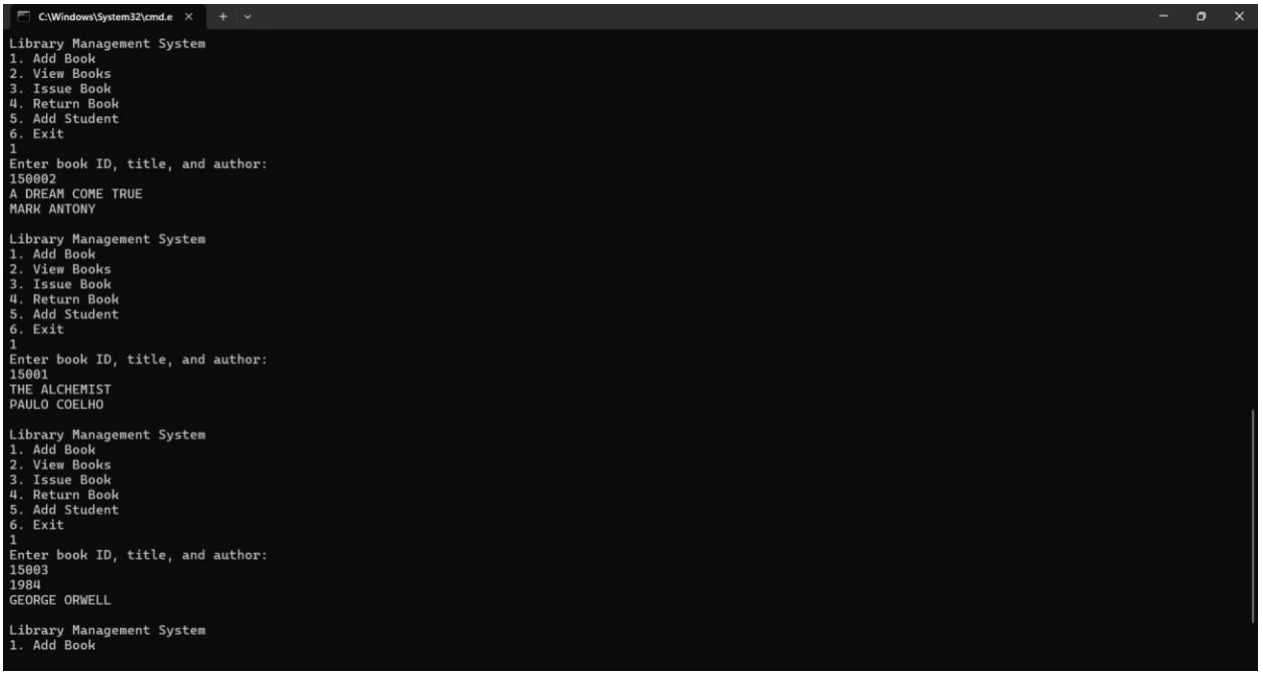
```
System.out.println("3. Issue Book");
System.out.println("4. Return Book");
System.out.println("5. Add Student");
System.out.println("6. Exit");
int choice = scanner.nextInt();
scanner.nextLine();
switch (choice) {
    case 1:
        System.out.println("Enter book ID, title, and author:");
        int bookId = scanner.nextInt();
        scanner.nextLine();
        String title = scanner.nextLine();
        String author = scanner.nextLine();
        library.addBook(new Book(bookId, title, author));
        break;
    case 2:
        library.viewBooks();
        break;
    case 3:
        System.out.println("Enter student ID and book ID:");
        int studentId = scanner.nextInt();
        int issueBookId = scanner.nextInt();
        library.issueBook(studentId, issueBookId);
        break;
    case 4:
        System.out.println("Enter student ID and book ID:");
        int returnStudentId = scanner.nextInt();
        int returnBookId = scanner.nextInt();
        library.returnBook(returnStudentId, returnBookId);
```

```
                break;
            case 5:
                System.out.println("Enter student ID and name:");
                int newStudentId = scanner.nextInt();
                scanner.nextLine();
                String name = scanner.nextLine();
                library.addStudent(new Student(newStudentId, name));
                break;
            case 6:
                System.out.println("Exiting...");
                return;
            default:
                System.out.println("Invalid choice!");
        }
    }
}
}
```

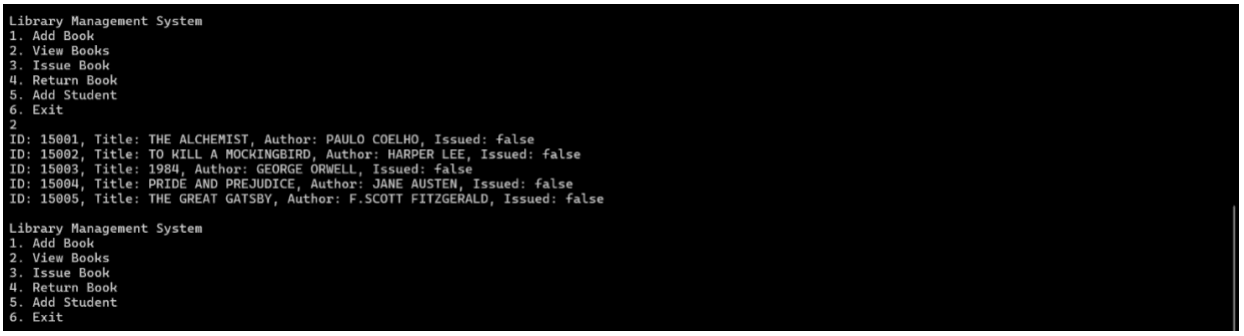
APPENDIX 2

SCREENSHOT

A2.1 ADD BOOK



A2.2 VIEW BOOKS



A2.3 ADD STUDENT

```
3. Issue Book
4. Return Book
5. Add Student
6. Exit
5
Enter student ID and name:
001
AATHISUDHAN

Library Management System
1. Add Book
2. View Books
3. Issue Book
4. Return Book
5. Add Student
6. Exit
5
Enter student ID and name:
002
ABESHEK

Library Management System
1. Add Book
2. View Books
3. Issue Book
4. Return Book
5. Add Student
6. Exit
5
Enter student ID and name:
003
ARAVIND

Library Management System
1. Add Book
2. View Books
3. Issue Book
4. Return Book
5. Add Student
6. Exit
```

A2.4 ISSUE BOOK

```
Library Management System
1. Add Book
2. View Books
3. Issue Book
4. Return Book
5. Add Student
6. Exit
3
Enter student ID and book ID:
001
15003
Book issued to student: AATHISUDHAN

Library Management System
1. Add Book
2. View Books
3. Issue Book
4. Return Book
5. Add Student
6. Exit
3
Enter student ID and book ID:
003
15002
Book issued to student: ARAVIND

Library Management System
1. Add Book
2. View Books
3. Issue Book
4. Return Book
5. Add Student
6. Exit
```

A2.5 RETURN BOOK

REFERENCES

[1]. Adeyemo, A. B., & Folorunso, O. (2013). Development of an Automated Library Management System for Academic Institutions. International Journal of Digital Library Systems (IJDLS), 3(1).
[DOI: 10.4018/ijdls.2013010101](https://doi.org/10.4018/ijdls.2013010101)

[2]. Schildt, H.(2018). Java: The Complete Reference, Eleventh Edition. McGraw-Hill Education.
[DOI: 10.1036/0071808558](https://doi.org/10.1036/0071808558)

[3] Mishra, M., & Mishra, S. (2017). Library Management System Using Java and MySQL. International Journal of Scientific & Engineering Research (IJSER), 8(4).
[DOI: 10.14299/00000008](https://doi.org/10.14299/00000008)