



**RAJALAKSHMI
ENGINEERING COLLEGE**

An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

MINI PROJECT REPORT

**BOOTHALINGESH N -230701056
ADHAVAN BALAJI N M-230701012**

LIBRARY MANAGEMENT SYSTEM

**BACHELOR OF ENGINEERING IN COMPUTER
SCIENCE ENGINEERING
RAJALAKSHMI ENGINEERING COLLEGE
(AUTONOMOUS) THANDALAM
CHENNAI-602105
2024 - 2025**

BONAFIDE CERTIFICATE

Certified that this project report

“HOSTEL MANAGEMENT SYSTEM”

Is the bonafide work of

“BOOTHALINGESH N (230701056),

ADHAVAN BALAJI N M (230701012)”

Who carried out the project work under my supervision.

Submitted for the Practical Examination held on.

23.11.2024_____

Signature of faculty in-charge

ABSTRACT

The **Library Management System** is a software solution designed to automate the core operations of a library, facilitating efficient management of books, members, and transaction records. The system streamlines key processes such as cataloging, member registration, book issue and return, and fine calculation. With this system, library staff can easily manage an extensive collection of resources, including books, journals, and digital media, and members can efficiently access, reserve, and borrow items. The project uses a MySQL database for data storage and retrieval, ensuring secure and organized handling of library data.

System Components

1. Books and Resources

- Manages information on books, journals, and digital media in the library.
- Tracks each item's availability, status, and lending history.

2. Library Members

- Registers library users, including students, faculty, and community members.
- Stores member details and borrowing privileges for easy tracking.

3. Library Staff

- Library staff members manage the system, including updating catalog information, handling book requests, and overseeing the fine collection process.
- Employee details, roles, and privileges are stored in the database.

Key Features and Functionalities

1. Book Catalog Management

- Allows library staff to catalog books and resources by title, author, ISBN, and category.
- Enables search, sort, and filter options for easy retrieval of resources by members and staff.

2. Member Registration and Management

- Stores member information, including personal details, membership ID, and borrowing history.
- Tracks active loans, overdue items, and calculated fines.

3. Book Issue and Return

- Streamlines the process of issuing and returning books.
- Automatically updates book availability, due dates, and fines, if applicable.

4. Transaction Management

- Manages transactions, including book checkouts, renewals, and returns.
- Calculates overdue fines and records payments.

5. Database Storage (MySQL)

- A MySQL database stores information on books, members, employees, and transaction logs.
- Ensures data consistency, secure access, and quick retrieval through optimized queries.

Database Schema Overview

- **Books Table:** Stores data on each resource, including title, author, category, and availability status.
- **Members Table:** Manages member information such as ID, contact details, and borrowing history.
- **Employees Table:** Tracks library staff roles and permissions for managing library resources.
- **Transactions Table:** Logs all book issues, renewals, and returns with associated dates and fine calculations.

Benefits

- **Efficiency:** Automation reduces manual work, minimizes errors, and allows staff to focus on customer service.
- **Centralized Data Management:** A MySQL database securely stores all data, allowing real-time access and updates.
- **Improved User Experience:** Members can easily search for resources, check availability, and reserve books online.
- **Scalability:** The system can accommodate an expanding catalog and membership base, supporting the library's growth.

The **Library Management System** offers a streamlined and reliable solution for managing library operations, enhancing both the administrative efficiency and user experience of the library.

Language Used - Java Core

Concept Used - Swing

IDE Used - JDBC

Database Used - MySQL

CONTENTS

CHAPTERS		PAGE NO
Chapter 1	Introduction	
	1.1 Problem Definition	1
	1.2 Need	2
Chapter 2	Requirements	
	2.1 Software Requirement Specifications	3
	2.2 Hardware Requirement Specifications	3
Chapter 3	Entity Relationship Diagram	4
	3.1 Entity relationship diagram	5
Chapter 4	Schema Diagram	6
	4.1 Schema diagram	7
Chapter 5	Implementation	
	5.1 Backend Implementation	8
	5.2 Frontend implemenatation	9

	5.3 Creating mainframe class	10 - 13
Chapter 6	Snapshots	14 - 19
	Conclusion	
	References	

CHAPTER 1

INTRODUCTION

The **Library Management System** is a software application developed to streamline and automate the daily operations of a library. This system allows libraries to manage resources efficiently, including the cataloging of books, tracking of inventory, registration of members, and processing of book issues and returns. By integrating data management and user interaction, the system enables libraries to offer a seamless experience for both staff and members, reducing manual effort and enhancing accessibility to library resources.

In the Library Management System, members can browse the catalog, reserve or check out items, and track their borrowing history. Meanwhile, library staff can manage collections, oversee membership records, and monitor transactions with ease. Using a structured database, this system provides a reliable platform for handling large volumes of information, ensuring quick data retrieval, accuracy, and secure access. Designed for scalability, the Library Management System can accommodate growing collections and user bases, making it an essential tool for modern libraries.

1.1 Problem Definition

The **problem definition** for the Library Management System is to address the inefficiencies and challenges associated with managing a library's resources, memberships, and transactions manually. Traditional methods, which often rely on paper records or basic spreadsheets, lead to difficulties in tracking book availability, managing member information, processing book issues and returns, and calculating fines for overdue items. These manual processes are time-consuming, prone to errors, and limit access to accurate, real-time information.

The goal of this system is to provide an automated solution that centralizes and streamlines library operations, allowing efficient handling of book cataloging, inventory management, and

member services. By integrating all library functions into a single, user-friendly platform, the Library Management System will improve resource accessibility, reduce administrative workload, minimize human error, and provide better service to library users.

1.2 Need

The need for a **Library Management System** arises from the demand for a more efficient, organized, and user-friendly way to handle the daily operations of a library. As libraries expand their collections and serve growing numbers of members, traditional methods of managing books, transactions, and member information become insufficient. Here are some key reasons that highlight the need for such a system:

1. **Improved Efficiency:** Automating cataloging, book issuance, and return processes reduces manual workload for library staff, allowing them to serve more members with greater speed and accuracy.
2. **Accurate Record Keeping:** A digital system minimizes human errors, ensuring accurate tracking of book inventory, member information, and transaction history, which is crucial for library operations.
3. **Better Resource Accessibility:** Members can easily browse the library's catalog, check book availability, and make reservations online, improving user experience and resource utilization.
4. **Real-Time Updates:** The system provides real-time information on book status (checked out, available, reserved), membership records, and fine calculations, offering instant updates and reducing delays.
5. **Enhanced Security and Data Integrity:** The system maintains secure records with role-based access controls, protecting member and transaction data from unauthorized access and ensuring data integrity.
6. **Scalability:** As libraries grow, a digital system can easily accommodate an increasing volume of books, members, and transaction data without performance issues.
7. **Streamlined Reporting:** The system enables easy generation of reports on book circulation, popular titles, overdue books, and more, supporting data-driven decision-making for library management.

Overall, a Library Management System provides an essential infrastructure to modernize library services, ensuring efficient, secure, and accessible management of resources for both staff and members

CHAPTER 2

REQUIREMENTS

2.1 Software Requirement Specifications

Operating System Front End Back End Server Documentation : Windows 10

Frontend Software:Java JDBC 8.2 :JDK 8

Backend Software: MySQL

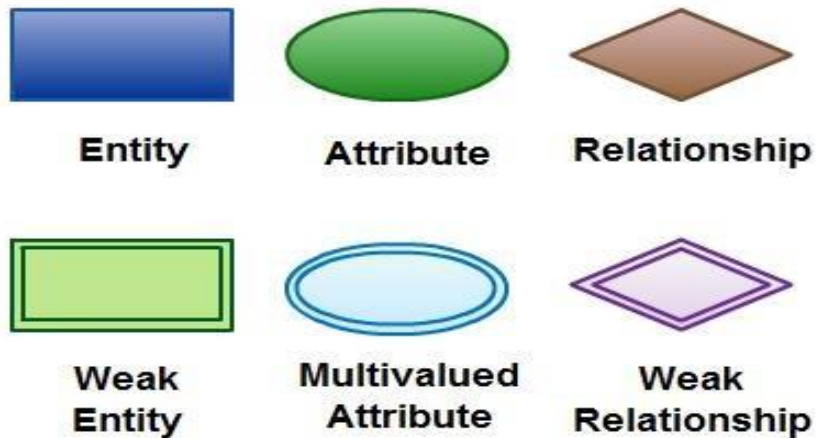
2.2 Hardware Requirement Specifications

Computer Processor Core i3 Processor Speed 2.3 GHz Processor Hard Disk 400 GB or more RAM
Min 2GB

CHAPTER 3

ENTITY RELATIONSHIP DIAGRAM

An entity-relationship (ER) diagram is a specialized graphic that illustrates the interrelationships between entities in a database. ER diagrams often use symbols to represent three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes. If the application is primarily a database application, the entity-relationship approach can be used effectively for modeling some parts of the problem. The main focus in ER modeling is the Data Items in the system and the relationship between them. It aims to create conceptual scheme for the Data from the user's perspective. The model thus created is independent of any database model. The ER models are frequently represented as ER diagram. Here we present the ER diagram of the above mentioned project.



CHAPTER 4

SCHEMA DIAGRAM

4.1 SCHEMA DIAGRAM

A database schema is the skeleton structure that represents the logical view of the entire database. A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.

A database schema can be divided broadly into two categories –

- **Physical Database Schema** – This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.

Logical Database Schema – This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

CHAPTER 5

IMPLEMENTATION

5.1 Backend Implementation

MYSQL

MySQL is an open-source relational database management system (RDBMS). A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.

```
CREATE DATABASE LibraryDB;
```

```
USE LibraryDB;
```

```
CREATE TABLE Books (  
    book_id INT AUTO_INCREMENT PRIMARY KEY,  
    title VARCHAR(255) NOT NULL,  
    author VARCHAR(255) NOT NULL,  
    genre VARCHAR(100) NOT NULL,  
    available BOOLEAN DEFAULT TRUE  
);
```

5.2 Frontend Implementation

Java Core

Core Java is the part of Java programming language that is used for creating or developing a general-purpose application. It uses only one tier architecture that is why it is called as 'stand alone' application. Core java programming covers the swings, socket, awt, thread concept, collection object and classes.

Swings

Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs.

Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

5.3 Creating mainframe class

```

import javax.swing.

import java.awt.*;

import java.awt.event.*;

import java.sql.*;

public class App extends JFrame {
    private JTextField titleField, authorField, genreField,
bookIdField;
    private JButton addButton, listBooksButton, updateButton,
deleteButton;

    public App() {
        setTitle("Library Management System");
        setSize(500, 400);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.insets = new Insets(5, 5, 5, 5);

        titleField = new JTextField(15);
        authorField = new JTextField(15);
        genreField = new JTextField(15);
        bookIdField = new JTextField(5); // Used for updating or
deleting
        addButton = new JButton("Add Book");
        listBooksButton = new JButton("List Books");
        updateButton = new JButton("Update Book");
        deleteButton = new JButton("Delete Book");

        // Add components with improved layout
        addComponent(new JLabel("Book ID (for Update/Delete):"),
bookIdField, gbc, 0);
        addComponent(new JLabel("Title:"), titleField, gbc, 1);
        addComponent(new JLabel("Author:"), authorField, gbc, 2);
        addComponent(new JLabel("Genre:"), genreField, gbc, 3);

        gbc.gridx = 0;
        gbc.gridy = 4;
        gbc.gridwidth = 2;
        add(addButton, gbc);
        gbc.gridy++;
        add(listBooksButton, gbc);

```

```

        gbc.gridy++;
        add(updateButton, gbc);
        gbc.gridy++;
        add(deleteButton, gbc);

        addButton.addActionListener(e -> addBook());
        listBooksButton.addActionListener(e -> listBooks());
        updateButton.addActionListener(e -> updateBook());
        deleteButton.addActionListener(e -> deleteBook());

        setVisible(true);
    }

    private void addComponent(JLabel label, JTextField field,
        GridBagConstraints gbc, int y) {
        gbc.gridx = 0;
        gbc.gridy = y;
        add(label, gbc);
        gbc.gridx = 1;
        add(field, gbc);
    }

    private void addBook() {
        String title = titleField.getText();
        String author = authorField.getText();
        String genre = genreField.getText();

        if (title.isEmpty() || author.isEmpty() || genre.isEmpty()) {
            showError("All fields (Title, Author, Genre) are
required!");
            return;
        }

        executeUpdate("INSERT INTO Books (title, author, genre) VALUES
(?, ?, ?)",
            "Book added successfully!",
            title, author, genre);
    }

    private void listBooks() {
        try (Connection conn = getConnection()) {
            String query = "SELECT * FROM Books";
            PreparedStatement stmt = conn.prepareStatement(query);
            ResultSet rs = stmt.executeQuery();

            StringBuilder books = new StringBuilder();

```

```

        while (rs.next()) {
            books.append("ID: ").append(rs.getInt("book_id"))
                .append(", Title: ")
            ").append(rs.getString("title"))
                .append(", Author: ")
            ").append(rs.getString("author"))
                .append(", Genre: ")
            ").append(rs.getString("genre"))
                .append(", Available: ")
            ").append(rs.getBoolean("available"))
                .append("\n");
        }
        JOptionPane.showMessageDialog(this, books.toString());
    } catch (SQLException e) {
        showError("Error fetching books.");
    }
}

private void updateBook() {
    String bookIdText = bookIdField.getText();
    String title = titleField.getText();
    String author = authorField.getText();
    String genre = genreField.getText();

    if (bookIdText.isEmpty() || title.isEmpty() ||
author.isEmpty() || genre.isEmpty()) {
        showError("All fields (Book ID, Title, Author, Genre) are
required!");
        return;
    }

    executeUpdate("UPDATE Books SET title = ?, author = ?, genre =
? WHERE book_id = ?",
        "Book updated successfully!",
        title, author, genre, bookIdText);
}

private void deleteBook() {
    String bookIdText = bookIdField.getText();

    if (bookIdText.isEmpty()) {
        showError("Book ID is required for deletion!");
        return;
    }
}

```

```

        executeUpdate("DELETE FROM Books WHERE book_id = ?", "Book
deleted successfully!", bookIdText);
    }

    private void executeUpdate(String query, String successMessage,
String... params) {
        try (Connection conn = getConnection();
            PreparedStatement stmt = conn.prepareStatement(query)) {

            for (int i = 0; i < params.length; i++) {
                if (i == params.length - 1 &&
query.contains("book_id")) {
                    stmt.setInt(i + 1, Integer.parseInt(params[i]));
                } else {
                    stmt.setString(i + 1, params[i]);
                }
            }

            int rowsAffected = stmt.executeUpdate();
            if (rowsAffected > 0) {
                JOptionPane.showMessageDialog(this, successMessage);
            } else {
                JOptionPane.showMessageDialog(this, "No book found
with the given ID.");
            }
        } catch (SQLException e) {
            showError("Database error.");
        } catch (NumberFormatException e) {
            showError("Invalid Book ID.");
        }
    }

    private Connection getConnection() throws SQLException {
        String url = "jdbc:mysql://localhost:3306/LibraryDB";
        String user = "root";
        String password = "12345678";
        return DriverManager.getConnection(url, user, password);
    }

    private void showError(String message) {
        JOptionPane.showMessageDialog(this, message, "Error",
JOptionPane.ERROR_MESSAGE);
    }

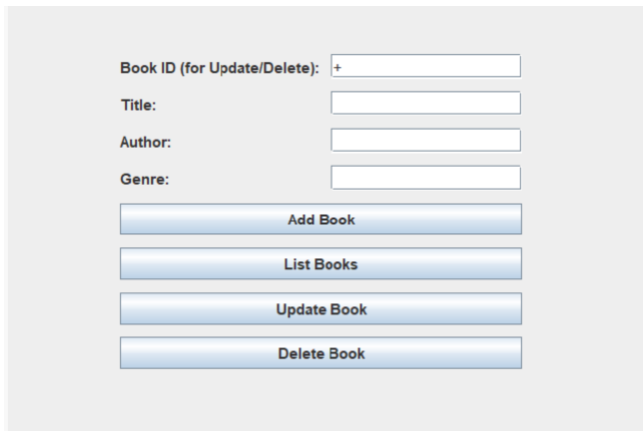
    public static void main(String[] args) {
        SwingUtilities.invokeLater(App::new);
    }

```


} }

CHAPTER 6

SNAPSHOTS



The screenshot shows a web application interface for managing books. It features a light gray background with a white form area. The form contains the following elements:

- A label "Book ID (for Update/Delete):" followed by a text input field containing a "+" sign.
- A label "Title:" followed by a text input field.
- A label "Author:" followed by a text input field.
- A label "Genre:" followed by a text input field.
- Four blue buttons with white text, stacked vertically:
 - "Add Book"
 - "List Books"
 - "Update Book"
 - "Delete Book"

Fig:6.1 Input page

Book ID (for Update/Delete):

Title:

Author:

Genre:

Fig:6.2 Entering details

Book ID (for Update/Delete):

Title:

Author:

Genre:

Message

Book added successfully!

Fig:6.3 Adding Book

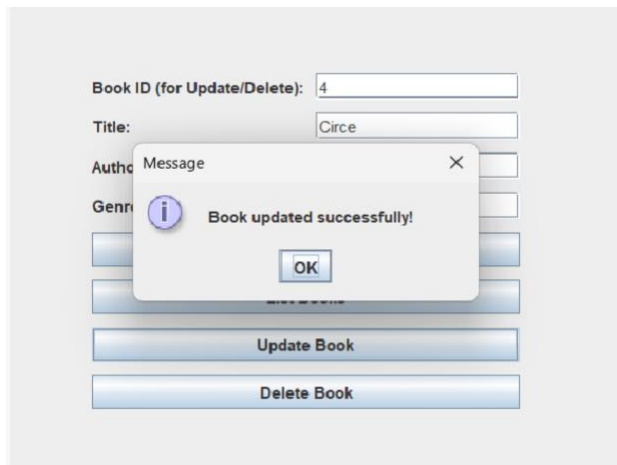


Fig:6.4 Updating book

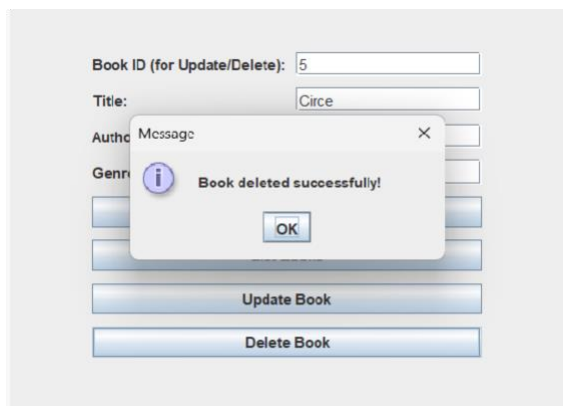


Fig:6.5 Deleting book

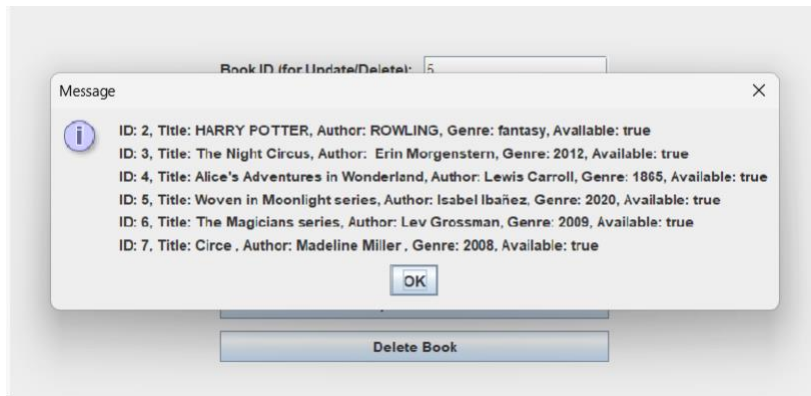


Fig:6.6 Listing books

CONCLUSION

The **Library Management System** built using **Swing** and **JDBC** in Java provides a robust, user-friendly solution for managing the daily operations of a library. By automating tasks such as adding, updating, deleting, and listing books, the system significantly reduces the administrative burden and enhances efficiency. With an intuitive graphical user interface (GUI) developed using Swing components, users can easily interact with the system to manage book records and perform essential operations.

The use of **JDBC** allows seamless interaction with a **MySQL database**, ensuring that all library data is stored securely and can be retrieved or updated with ease. This integration also ensures the system's scalability, allowing it to grow as the library's collection and user base expand.

Overall, this system demonstrates how Java's powerful Swing toolkit and JDBC can be combined to create a functional, efficient, and easy-to-use application for managing library resources. Future improvements could include features like user authentication, book borrowing/return functionality, and more advanced search options, making this Library Management System adaptable to different library needs and scalable for larger environments.

REFERENCES

1. <https://koha-community.org/>
2. <https://evergreen-ils.org/>

