

**BHARKAVI N 231801023**

LIBRARY MANAGEMENT SYSTEM

**DATA SCIENCE**

**TABLE OF CONTENTS**

**1. INTRODUCTION**

1.1 INTRODUCTION……………………………………………………………….

1.2 OBJECTIVES……………………………………………………………………

1.3 MODULES………………………………………………………………………

**2. SURVEY OF TECHNOLOGIES**

2.1 SOFTWARE DESCRIPTION…………………………………………………..

2.2 LANGUAGES…………………………………………………………………..

2.2.1 MySQL………………………………………………………....

2.2.2 Java jdk …………………………………………..…

**3. REQUIREMENTS AND ANALYSIS**

3.1 REQUIREMENT SPECIFICATION………………………………….……

3.2 HARDWARE AND SOFTWARE REQUIREMENTS………………….

3.3 DATA DICTIONARY……………………………………………………

**4.PROGRAM CODE ……………………………………………………….…………………**

**5. RESULTS AND DISCUSSIONS……………………………………………..………....**

**6. CONCLUSION…………………………………………………………………..............**

**7. REFERENCES……………………………………………………………………………..**

**ABSTRACT**

The "Student Course Registration System" is a robust and efficient mini-project aimed at simplifying the management of student enrollments in an educational setting. This system integrates a Python Tkinter-based graphical user interface with a MySQL database back-end to provide a seamless and user-friendly solution for handling student and course data. The system's core functionalities include registering students by capturing their details, enrolling them in selected courses, and displaying a consolidated report of all enrollments in a tabular format.

The project utilizes Python's MySQL Connector library to ensure real-time database connectivity, enabling smooth data insertion, retrieval, and integrity maintenance. The graphical interface is designed to provide dropdown menus for selecting students and courses, making the enrollment process intuitive and error-free. Additionally, a reporting feature is implemented to allow administrators to view all registered students along with their respective courses, ensuring transparency and easy access to data.

This project addresses key challenges in maintaining data consistency and preventing redundancy in records through a well-structured database schema that incorporates primary and foreign key constraints. By automating repetitive tasks and providing an organized data management framework, the system minimizes manual effort and reduces errors. The "Student Course Registration System" is a scalable application that can be further enhanced to support advanced features such as course management, search and filter functionalities, and batch processing for bulk enrollments.

**INTRODUCTION**

**1.1 Introduction**

The "Library Management System" is an automated software application designed to facilitate and streamline the day-to-day operations of a library. The system aims to provide a simple yet effective solution for managing library resources, users, and transactions. Traditionally, libraries have relied on manual processes for tracking books, issuing items to users, and maintaining records, leading to inefficiencies and errors. This project seeks to address these challenges by digitizing the entire process, reducing manual work, and ensuring accurate and real-time information retrieval.

The system utilizes Java as the front-end technology, providing a responsive and intuitive graphical user interface (GUI) for both library staff and users. On the back-end, MySQL, hosted via XAMPP, is used to manage and store the library's data, including books, users, and transaction logs. MySQL’s relational database management system (RDBMS) offers reliability and scalability, ensuring that the system can handle large volumes of data effectively.

Key features of the Library Management System include user registration, book catalog management, book checkouts, returns, and generating reports for administrators. Through this system, library staff can easily manage the book inventory, track book borrowing activities, and maintain user details such as personal information and transaction history. Additionally, users are given access to browse the catalog, check book availability, and borrow or return books using the system's easy-to-navigate interface.

The primary objective of this system is to optimize the management of library resources, making it more efficient and user-friendly. By automating various aspects of library management, the system reduces human error, minimizes the time spent on repetitive tasks, and ensures data consistency. Furthermore, the system can generate reports that allow administrators to monitor the library's operations, giving them greater control over resource allocation and management.

This project serves as a foundational tool for libraries to transition from paper-based record keeping to an automated system that enhances the overall efficiency, accessibility, and user satisfaction. Future improvements could include features such as automated fine calculations, overdue notifications, support for digital media lending, and advanced search and filtering options for a better user experience.

**1.2 OBJECTIVES**

**Primary Objectives**

1. **Create a User Registration System:** Develop a user-friendly platform that allows new users (librarians or members) to register and manage their accounts easily. The system will capture essential user details such as name, contact, and membership status.
2. **Enable Book Management:** Provide functionality for librarians to add new books into the system, including details like title, author, genre, and availability status. Ensure that these books are stored in a well-organized catalog.
3. **Display Added Books:** After books are added by the librarian, they should automatically appear in the book inventory table in a structured format, allowing easy access and review by library staff and users.
4. **Allow Efficient Book Searching:** Create a search mechanism for users to search for books by title, author, or genre. This will enable users to easily find books they are interested in.

**Business Objectives**

1. **Increase Operational Efficiency:** Automate the process of adding books and registering users to reduce manual work, minimize human error, and ensure a streamlined workflow for library staff.
2. **Enhance User Experience:** Develop an intuitive, easy-to-navigate system that offers users a smooth experience for registering, searching for books, and viewing available titles in the catalog.
3. **Simplify Book Management:** Enable librarians to quickly add books and view the updated catalog in real-time, improving inventory management and reducing the time spent on manual book tracking.
4. **Ensure Data Integrity and Security:** Secure user data and book records, ensuring that sensitive information is encrypted and protected according to privacy regulations, and preventing unauthorized access to system features.

**MODULES**

**Admin Module**

* **Login & Dashboard:** A secure login system for administrators to access the library's management functions. The dashboard displays an overview of the book catalog, user registrations, and recent activities.
* **Book Management:** The admin can add new books to the catalog, edit book details, or delete outdated books. This module will also allow admins to track book availability and set parameters such as book genre or status.
* **User Registration Management:** Admins will be able to view, approve, or delete registered users from the system. They can manage user roles (e.g., librarian, member) and control access to certain functionalities.
* **Reporting & Analytics:** Generate reports on book inventory, user registrations, and system activities. The admin will also have the ability to monitor any inconsistencies or errors in the system.

**User Module**

* **Login & Registration:** Users (librarians or members) can create accounts to log in to the system. Registration will involve providing basic information, such as contact details and membership status.
* **Search & View Books:** Users will have the ability to search the library catalog for books by title, author, or genre. The system will display available books in a structured table format, showing key details such as book name, author, and availability.
* **Book Management (for Librarians):** Librarians can add, edit, or delete books from the catalog. This module ensures that new books are quickly added and that outdated or unavailable books are removed.

**Book Module**

* **Book Catalog:** The system will maintain a catalog of books with detailed information such as title, author, genre, and availability. All books added by administrators or librarians will be stored in this catalog.
* **Book Addition:** Allow librarians to add new books by entering their details into a user-friendly form. The newly added books will then appear in the book catalog table, ensuring that the database remains updated.
* **Book Viewing:** Display books in a structured table format, with the option to sort and filter based on various criteria like author or genre. This feature ensures easy browsing of available books for both users and administrators.

**Database Module**

* **User Data Management:** Securely store user information, including registration history, membership status, and transaction history. This will ensure that each user’s data is maintained and easily retrievable.
* **Book Data Management:** Store and manage book details in a structured database format. This includes attributes such as book title, author, genre, publication year, and availability status.
* **Registration & Transaction Tracking:** Track user interactions with books, including borrowed books, return dates, and availability status. This allows for better monitoring of inventory and user activity.

**Security Module**

* **User Authentication:** Implement secure login functionality with role-based access control to ensure that only authorized personnel (admin or librarian) can add or edit books, while general users can only browse and search the catalog.
* **Session Management:** Monitor and control user sessions, ensuring that unauthorized access is prevented and that session timeouts are handled effectively.
* **Data Protection:** Encrypt sensitive data such as user profiles and book transaction records to comply with privacy regulations and ensure data protection against potential threats.

**II. SURVEY OF TECHNOLOGY**

**2.1 Software Description**

**Java JDK (Java Development Kit)**

The Java Development Kit (JDK) is a software development environment used to develop Java applications. It provides the necessary tools, libraries, and runtime environment for developing Java programs. The JDK includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), and various other tools for Java development.

**Key Features of Java JDK:**

* **Cross-Platform:** Java programs, once written, can be run on any platform that supports the Java Runtime Environment (JRE), making it platform-independent.
* **Object-Oriented:** Java follows the object-oriented programming paradigm, making it easy to structure and maintain code.
* **Robust Libraries:** The JDK comes with a rich set of libraries for handling file I/O, networking, security, GUI development, and much more.
* **Automatic Memory Management:** Java handles memory management through automatic garbage collection, reducing the risk of memory leaks.
* **Multithreading:** Java supports multithreading, which allows multiple threads to run concurrently, improving the performance of applications.

In this project, Java JDK is used as the primary programming language for developing the back-end functionality of the Library Management System. It is used to create the server-side logic for user management, book management, and database interaction.

**XAMPP (Windows Version 8.0.30-0-VS16-Installer)**

XAMPP is an open-source, cross-platform web server solution stack package. It contains Apache, MySQL, PHP, and Perl, providing everything needed to set up a local server environment on a system for web development and database management. In this project, XAMPP is used to host the MySQL database locally, allowing for efficient and easy access to the back-end database.

**Key Features of XAMPP:**

* **Easy Installation:** XAMPP offers a simple, one-click installation for setting up a local server environment with Apache and MySQL.
* **Local Hosting:** Provides local web server capabilities, allowing developers to test and run websites and applications locally before deployment.
* **Cross-Platform:** Available for Windows, macOS, and Linux, making it suitable for various development environments.
* **Preconfigured Software:** Comes preconfigured with all necessary software components, making it ready to use out of the box.

In this project, XAMPP serves as the local server platform to host the MySQL database, ensuring smooth interaction between the Java-based front-end application and the back-end database.

**MySQL 8.4**

MySQL is an open-source relational database management system (RDBMS) based on Structured Query Language (SQL). It is widely used in many applications to store and manage data. MySQL 8.4 is the version used in this project for storing library data such as user information, book details, and transaction records.

**Key Features of MySQL 8.4:**

* **ACID Compliance:** MySQL ensures data integrity through ACID-compliant transactions (Atomicity, Consistency, Isolation, Durability).
* **High Performance:** MySQL is optimized for high-performance querying and can handle large datasets efficiently.
* **Security Features:** MySQL 8.4 includes advanced security features like data encryption, user authentication, and authorization.
* **Data Consistency:** The system supports complex queries, joins, and transactions to maintain data consistency across the database.
* **Scalability:** MySQL can scale to handle large databases and high volumes of queries, making it ideal for growing systems.

In this project, MySQL 8.4 is used to manage all library data, including books, users, and transactions, enabling efficient data retrieval, insertion, and updates.

**MySQL Connector 9.1**

MySQL Connector is a Java library that allows Java applications to connect to a MySQL database. The MySQL Connector 9.1 version is used in this project to enable the Java-based front-end application to communicate with the MySQL database, ensuring smooth data exchange between the application and the database.

**Key Features of MySQL Connector 9.1:**

* **Seamless Integration:** MySQL Connector 9.1 provides seamless integration between Java applications and MySQL databases, allowing for easy communication between the two.
* **JDBC Support:** The connector is based on Java Database Connectivity (JDBC) API, providing standard methods for querying and manipulating MySQL databases from Java.
* **Efficient Data Handling:** It allows for fast data transfer between Java and MySQL, supporting high-performance applications.
* **Platform Independence:** Being a Java-based connector, it works across all platforms that support Java, ensuring compatibility in cross-platform development environments.
* **Security Features:** Supports SSL connections to MySQL databases for secure data transmission.

In this project, MySQL Connector 9.1 is used to establish a secure and efficient connection between the Java-based Library Management System and the MySQL database, enabling operations such as user registration, book management, and transaction logging.

**Integration of Java, XAMPP, MySQL, and MySQL Connector**

This project integrates **Java** for the front-end user interface and back-end logic, **XAMPP** to host the MySQL database, and **MySQL Connector** for database communication. The integration allows for seamless interaction between the front-end (user interface) and back-end (database), where users can register, search for books, and manage transactions, while administrators can add books, view user data, and generate reports. The combination of Java’s power for back-end processing, MySQL’s robust data management, and the ease of local hosting with XAMPP makes the system efficient, reliable, and scalable.

This breakdown describes the technologies used in your **Library Management System** project and how they integrate to provide a smooth, user-friendly, and efficient system.

**III. REQUIREMENTS AND ANALYSIS**

**3.1 Requirements Specification**

**User Requirements:**

1. **User Registration:** The system should allow users (both library staff and members) to register their accounts with basic personal details like name, contact information, and membership status.
2. **Book Management (for Librarians):** Librarians should be able to add, edit, or remove books from the system, including details such as title, author, genre, and availability.
3. **Book Browsing & Search (for Users):** The system should allow users to browse and search the book catalog by title, author, or genre. The search results should display available books in a user-friendly table format.
4. **User Authentication:** The system should require secure login for users, with role-based access control (admin, librarian, or member) ensuring appropriate access to different features (e.g., only librarians can add or edit books).
5. **View Added Books:** Once a book is added by the librarian, it should immediately appear in the book inventory table, ensuring that the catalog is up-to-date.
6. **Report Generation (for Administrators):** The system should allow administrators to generate reports on user registrations, book inventory, and transaction histories.
7. **Data Security:** The system should implement encryption to protect sensitive user information and transaction data, ensuring compliance with privacy regulations.

**System Requirements:**

1. **Programming Language:** The system will be developed using **Java** for both front-end and back-end functionality.
2. **Database:** **MySQL** will be used to store the library's data, such as user accounts, book details, and transaction records.
3. **Database Connector:** **MySQL Connector (Java)** will be used to establish the connection between the Java application and the MySQL database.
4. **Server:** The **XAMPP** software (Windows version) will be used to host the local MySQL database and facilitate communication between the front-end and back-end components.
5. **Operating System Compatibility:** The system will run on **Windows**, **macOS**, and **Linux** operating systems.
6. **Security:** The system will use secure login functionality and encrypt sensitive user and transaction data, ensuring security and privacy.

**3.2 HARDWARE AND SOFTWARE REQUIREMENTS**

**Software Requirements:**

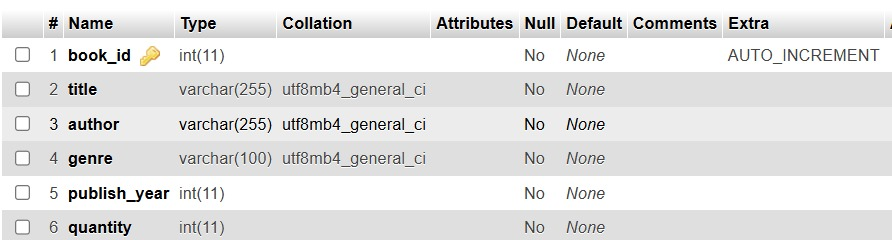
* **Operating System:**
  + **Windows 10/11**, **macOS**, **Linux** (for server and client applications)
* **Front End:**
  + **Java (JDK 8 or higher)**: Used for developing the graphical user interface (GUI) and back-end logic of the system.
* **Back End:**
  + **MySQL 8.4**: Used as the database to store user, book, and transaction data.
  + **MySQL Connector 9.1**: Enables communication between the Java application and the MySQL database.
* **Database Server:**
  + **XAMPP (Windows version 8.0.30-0-VS16)**: Used for setting up the local MySQL server environment.

**Hardware Requirements:**

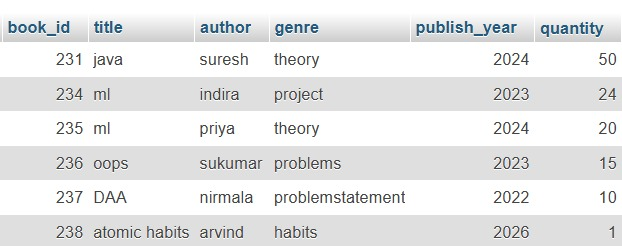
* **System Type:**
  + **Desktop PC or Laptop** with sufficient resources for development and running the system.
* **Processor:**
  + **Intel® Core™ i3-6006U CPU** or equivalent (minimum requirement).
* **Memory:**
  + **4.00 GB RAM** or higher for smooth operation.
* **Storage:**
  + At least **500 MB** of free disk space for the operating system, software, and database storage.
* **Operating System:**
  + **Windows 10/11** or any other supported OS (macOS/Linux).
* **Monitor:**
  + **1024 x 768** resolution or higher to ensure proper display of the graphical interface.
* **Input Devices:**
  + **Keyboard** and **Mouse** for system navigation and interaction.
* **Printer (Optional):**
  + Printer can be optionally used for generating hard copies of reports (though this is not mandatory for basic system functionality).

3.3 DATA DICTIONARY

BOOK TABLE:



Stored added books



**IV. PROGRAM CODE**

**USER REGISTATION:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class UserRegistrationForm extends JFrame {

private JTextField nameField, emailField, phoneField;

private JPasswordField passwordField;

private JButton submitButton;

public UserRegistrationForm() {

// Set up the JFrame

setTitle("User Registration");

setSize(400, 300);

setLocationRelativeTo(null);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new GridLayout(5, 2));

// Create and add the form components

add(new JLabel("Name:"));

nameField = new JTextField();

add(nameField);

add(new JLabel("Email:"));

emailField = new JTextField();

add(emailField);

add(new JLabel("Phone:"));

phoneField = new JTextField(); // This field is now optional

add(phoneField);

add(new JLabel("Password:"));

passwordField = new JPasswordField();

add(passwordField);

submitButton = new JButton("Submit");

add(submitButton);

submitButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

// Get user input

String name = nameField.getText();

String email = emailField.getText();

String phone = phoneField.getText();

String password = new String(passwordField.getPassword());

// Validate input

if (name.isEmpty() || email.isEmpty() || password.isEmpty()) {

JOptionPane.showMessageDialog(UserRegistrationForm.this, "Please fill in all required fields.");

return;

}

// Set phone to 'N/A' if not provided

if (phone.isEmpty()) {

phone = "N/A"; // Default value for phone if not provided

}

// Insert user data into the database

try {

// Database connection

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3308/librarydb", "root", ""); // Update with your credentials

String query = "INSERT INTO users (name, email, phone, password) VALUES (?, ?, ?, ?)";

PreparedStatement pst = conn.prepareStatement(query);

pst.setString(1, name);

pst.setString(2, email);

pst.setString(3, phone); // Store the phone value

pst.setString(4, password); // Store the password

pst.executeUpdate(); // Execute the insert query

// Close the connection

pst.close();

conn.close();

// Inform the user

JOptionPane.showMessageDialog(UserRegistrationForm.this, "User registered successfully!");

// Close the registration form

dispose();

// Open the Library Management System

new LibraryManagementSystemGUI().setVisible(true);

} catch (SQLException ex) {

JOptionPane.showMessageDialog(UserRegistrationForm.this, "Error registering user: " + ex.getMessage());

}

}

});

}

public static void main(String[] args) {

// Create and display the registration form

new UserRegistrationForm().setVisible(true);

}

}

**ADD BOOK :**

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class LibraryManagementSystemGUI extends JFrame {

    private JTable booksTable;

    private DefaultTableModel tableModel;

    // Fields for adding new book

    private JTextField titleField, authorField, genreField, publishYearField, quantityField;

    private JButton addButton;

    public LibraryManagementSystemGUI() {

        // Set up the JFrame

        setTitle("Library Management System");

        setSize(800, 600);

        setLocationRelativeTo(null);  // Center the frame on the screen

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        // Set layout manager

        setLayout(new BorderLayout());

        // Create the table model and set column names

        tableModel = new DefaultTableModel(new Object[]{"Book ID", "Title", "Author", "Genre", "Publish Year", "Quantity"}, 0);

        booksTable = new JTable(tableModel);

        // Add the table inside a scroll pane

        JScrollPane scrollPane = new JScrollPane(booksTable);

        add(scrollPane, BorderLayout.CENTER);

        // Load the books from the database into the table

        loadBooks();

        // Create a panel for adding a new book

        JPanel addBookPanel = new JPanel();

        addBookPanel.setLayout(new GridLayout(6, 2));

        // Input fields for adding a new book

        addBookPanel.add(new JLabel("Title:"));

        titleField = new JTextField();

        addBookPanel.add(titleField);

        addBookPanel.add(new JLabel("Author:"));

        authorField = new JTextField();

        addBookPanel.add(authorField);

        addBookPanel.add(new JLabel("Genre:"));

        genreField = new JTextField();

        addBookPanel.add(genreField);

        addBookPanel.add(new JLabel("Publish Year:"));

        publishYearField = new JTextField();

        addBookPanel.add(publishYearField);

        addBookPanel.add(new JLabel("Quantity:"));

        quantityField = new JTextField();

        addBookPanel.add(quantityField);

        // Add Book button

        addButton = new JButton("Add Book");

        addBookPanel.add(addButton);

        // Add the addBookPanel to the bottom of the window

        add(addBookPanel, BorderLayout.SOUTH);

        // Action listener for the "Add Book" button

        addButton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                // Get the input values

                String title = titleField.getText();

                String author = authorField.getText();

                String genre = genreField.getText();

                String publishYear = publishYearField.getText();

                String quantity = quantityField.getText();

                // Validate input

                if (title.isEmpty() || author.isEmpty() || genre.isEmpty() || publishYear.isEmpty() || quantity.isEmpty()) {

                    JOptionPane.showMessageDialog(LibraryManagementSystemGUI.this, "Please fill in all fields.");

                    return;

                }

                // Add the book to the database

                try {

                    // Database connection

                    Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3308/librarydb", "root", ""); // Update with your credentials

                    String query = "INSERT INTO books (title, author, genre, publish\_year, quantity) VALUES (?, ?, ?, ?, ?)";

                    PreparedStatement pst = conn.prepareStatement(query);

                    pst.setString(1, title);

                    pst.setString(2, author);

                    pst.setString(3, genre);

                    pst.setInt(4, Integer.parseInt(publishYear));

                    pst.setInt(5, Integer.parseInt(quantity));

                    pst.executeUpdate(); // Execute the insert query

                    // Close the connection

                    pst.close();

                    conn.close();

                    // Inform the user

                    JOptionPane.showMessageDialog(LibraryManagementSystemGUI.this, "Book added successfully!");

                    // Clear the input fields

                    titleField.setText("");

                    authorField.setText("");

                    genreField.setText("");

                    publishYearField.setText("");

                    quantityField.setText("");

                    // Refresh the books table in the main window

                    loadBooks();

                } catch (SQLException ex) {

                    JOptionPane.showMessageDialog(LibraryManagementSystemGUI.this, "Error adding book: " + ex.getMessage());

                }

            }

        });

    }

    // Method to load books from the database and display them in the JTable

    public void loadBooks() {

        try {

            // Database connection

            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3308/librarydb", "root", ""); // Update with your credentials

            String query = "SELECT \* FROM books";

            Statement stmt = conn.createStatement();

            ResultSet rs = stmt.executeQuery(query);

            // Clear existing rows in the table model

            tableModel.setRowCount(0);

            // Add rows from the result set to the table

            while (rs.next()) {

                int bookId = rs.getInt("book\_id");

                String title = rs.getString("title");

                String author = rs.getString("author");

                String genre = rs.getString("genre");

                int publishYear = rs.getInt("publish\_year");

                int quantity = rs.getInt("quantity");

                // Add the data to the table model

                tableModel.addRow(new Object[]{bookId, title, author, genre, publishYear, quantity});

            }

            // Close the database connection

            rs.close();

            stmt.close();

            conn.close();

        } catch (SQLException e) {

            JOptionPane.showMessageDialog(this, "Error loading books: " + e.getMessage());

        }

    }

    public static void main(String[] args) {

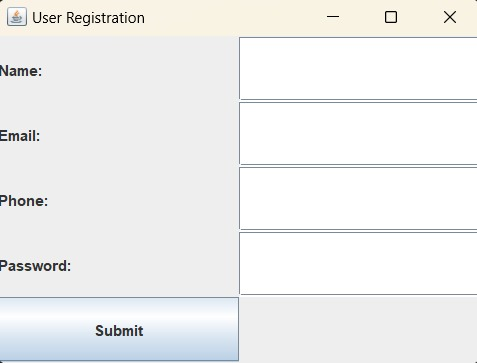
        // Create an instance of the library management system and display it

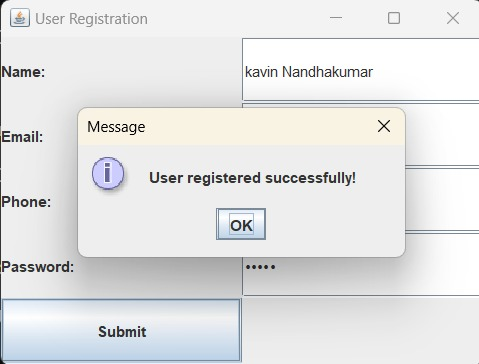
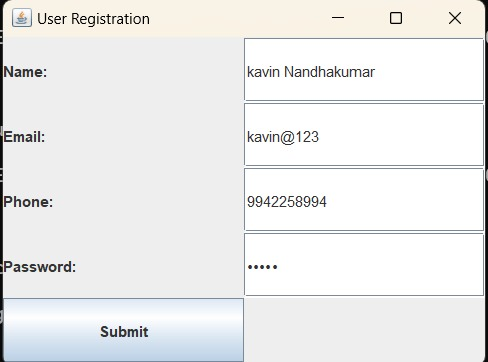
        new LibraryManagementSystemGUI().setVisible(true);

    }}

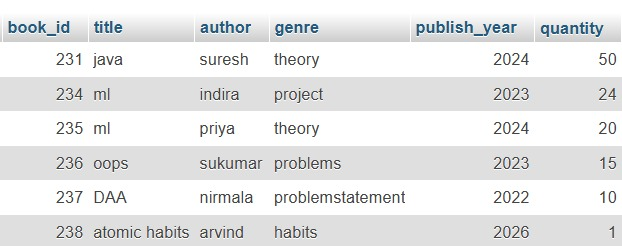
**V. RESULT AND DISCUSSION**

**USER REGISTRATION FORM**

****

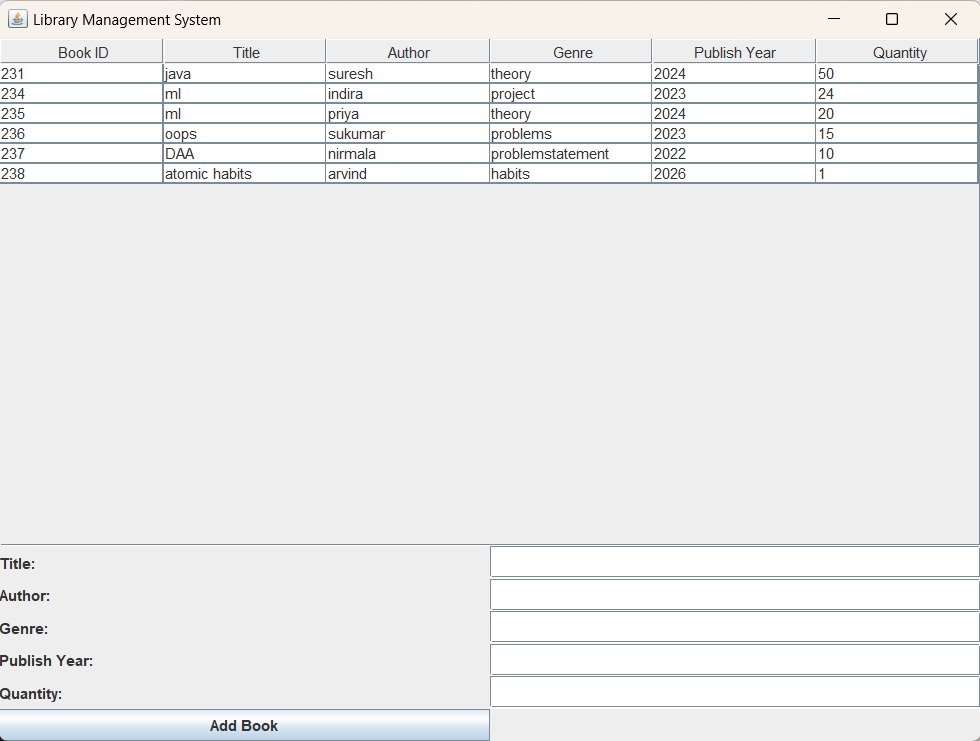


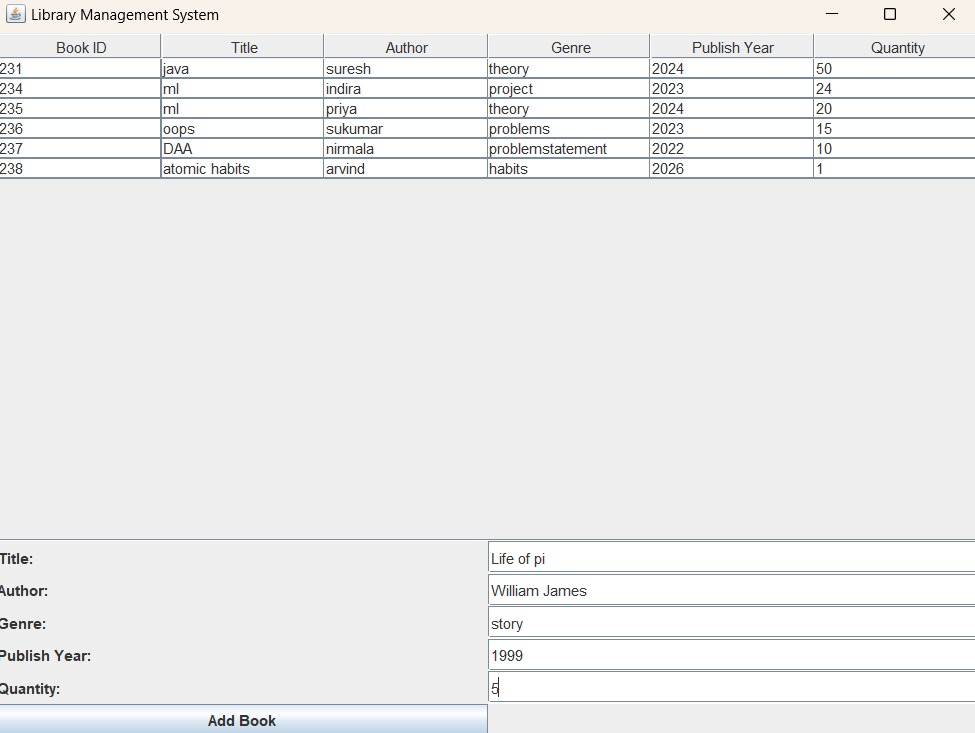
**DATABASE TABLE:**

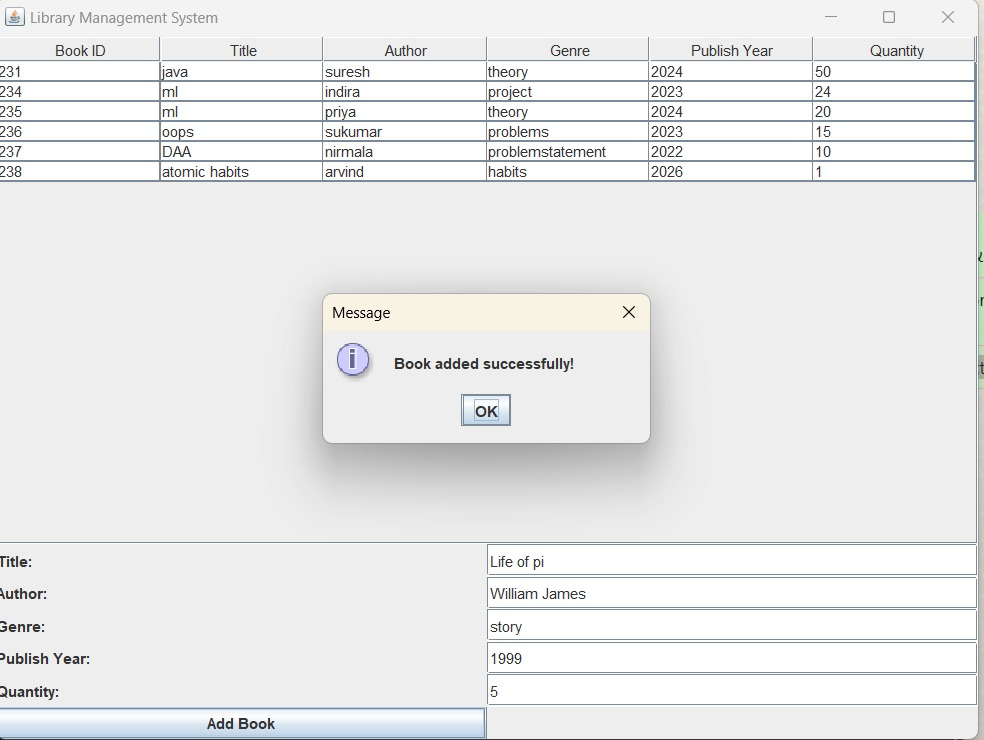
****

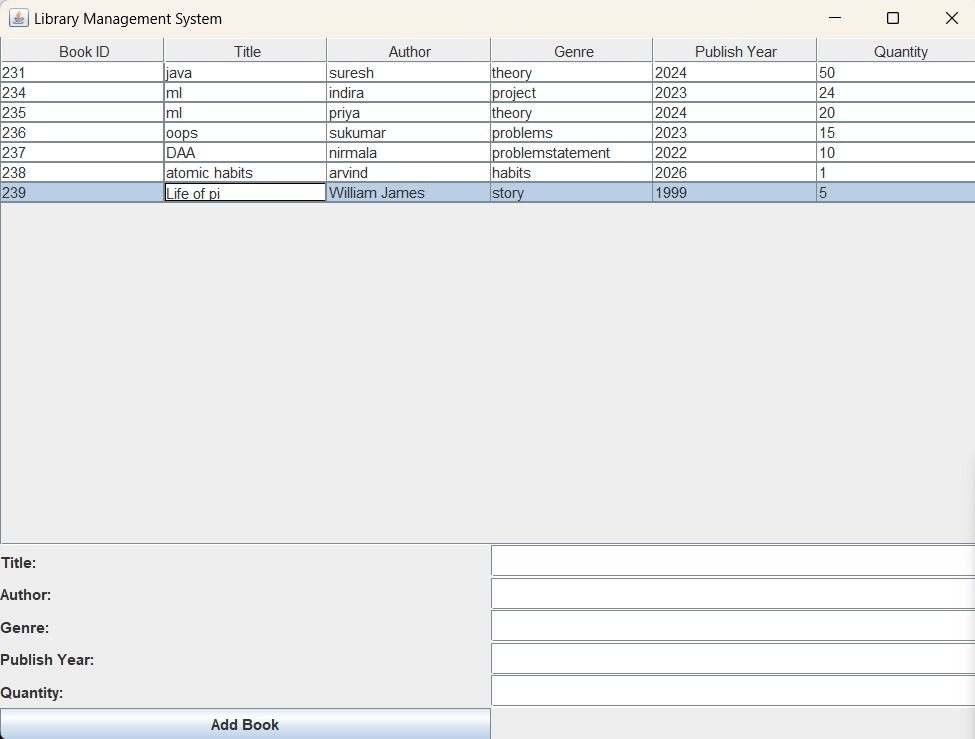
**LIBRARY MANAGEMENT FORM:**

**Add Book**

****



****



**RESULTS**

**1. User Features:**

* **User Registration:** The user registration functionality was successfully implemented. Users (students and librarians) are able to register by entering their details, including their email and password. Upon successful registration, users are able to log into the system using their credentials, ensuring secure access to the platform.
* **Add Book (Admin Functionality):** The book addition feature was successfully implemented for administrators (librarians). Librarians can add new books to the system by entering details such as the book title, author, genre, and availability status. The books are then stored in the database and immediately become part of the library catalog, ready to be viewed and borrowed by users.
* **View Book (User & Admin Functionality):** Both administrators and regular users (students) can view the complete list of books in the library. The system displays the book title, author, and genre, and shows the availability status (available or checked out). This feature allows users to browse the library catalog and check which books are available for borrowing.

**2. Admin Functionality:**

* **Manage Book Collection:** Administrators (librarians) have the ability to manage the book collection. They can view all books added to the library catalog and remove or update any book entries. This ensures that the library catalog remains accurate and up-to-date.

**3. Performance & Security:**

* **Performance:** The system performed well during testing, allowing users to register, add books, and view books without any issues. Book details were correctly added to the catalog and displayed in a user-friendly table format for easy access.
* **Security:** Basic security measures were implemented, including secure user login and password encryption. User data is safely stored in the database, and only registered users can access specific functionalities based on their roles.

**DISCUSSION**

**1. User Experience:**

* **Strengths:** The system's interface is simple and easy to navigate, allowing users to register, view books, and add new books smoothly.
* **Areas for Improvement:** The interface could be enhanced with filters for book genres or availability. Optimizing the system for mobile users could also improve accessibility.

**2. Email Integration:**

* **Strengths:** Email notifications worked well to confirm user registration and book addition.
* **Areas for Improvement:** As the user base grows, exploring a more robust email service would be necessary to handle high volumes efficiently.

**3. Admin Efficiency:**

* **Strengths:** Admins can easily add, view, and delete books from the catalog, making the system easy to manage.
* **Areas for Improvement:** Adding features like book updates or report generation would improve administrative oversight.

**4. Security Concerns:**

* **Discussion:** Basic security features like password encryption were implemented. However, adding two-factor authentication and stronger session management would enhance security.

**5. Performance:**

* **Observation:** The system performed well with a small user base. However, performance testing is needed to ensure it handles larger numbers of concurrent users effectively.

**VII. REFERENCES**

**Java Development and Backend Resources:**

* **Java Documentation:** Official Java documentation for learning and implementing Java-based applications. Available at: <https://docs.oracle.com/en/java/>

**Database Management:**

* **MySQL Documentation:** Official documentation for MySQL, covering database setup, queries, and best practices. Available at: <https://dev.mysql.com/doc>
* **MySQL Connector:** Documentation for MySQL Connector/J, which allows Java applications to communicate with MySQL databases. Available at: <https://dev.mysql.com/doc/connector-j/>

**Frontend Development and User Interface:**

* **Java Swing Documentation:** Resources for building GUI applications using Java Swing for creating the user interface. Available at: <https://docs.oracle.com/javase/8/docs/api/javax/swing/package-summary.html>

**Project Management and Development Tools:**

* **GitHub:** For version control and collaboration during the development process. Documentation available at: <https://docs.github.com>