**NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY**

**MILITARY COLLEGE OF SIGNALS**

****

Data Structures & Algorithms

(CS-250)

**OPEN-ENDED LAB**

**Submitted by:** **MUHAMMAD AHMAD SULTAN**

**CMS ID**: **408709**

**R**ANK: **NC**

**C**OURSE: **BESE-28**

**S**ECTION: **C**

**Submitted to:** **Mam Muntaha Noor**

***D****ated:*08-01-2024

Catalog of Open-Ended Lab

Table of Contents

* Title of the Open-Ended Lab03
* About Me04
* Introduction (Abstract)05
* History of Library Management Systems05
* Objective05
* Scope05
* Functionality & Implementation06
* Java Classes10
* MySQL Database Queries18
* Data Structures and Algorithms22
* Input and Output Specifications23
* OOP Principles & Concepts24
* Future Enhancements25
* Execution & Output 26
* Display27
* Potential Questions35
* Conclusion35
* Reference Citations35

**DEPARTMENT OF COMPUTER SOFTWARE ENGINEERING**

**Military College of Signals**

**National University of Sciences and Technology**

[**www.mcs.nust.edu.pk**](http://www.mcs.nust.edu.pk)

****

**T**ITLEOF **T**HE **O**PEN**-ENDED L**AB

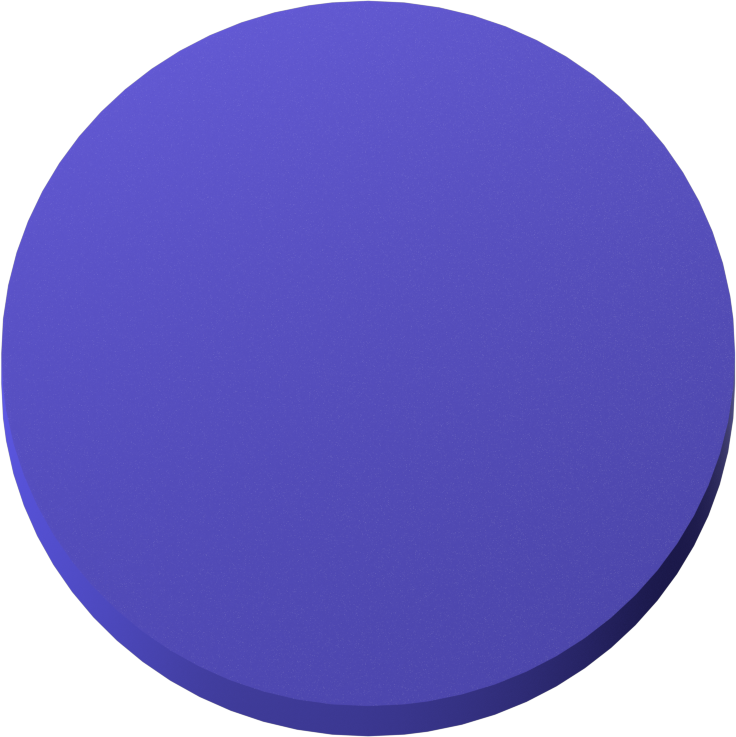
**BookTrack Nexus**

**Library**

**Management System**

***A Comprehensive Library Management System in Java, integrating Data Structures & Algorithms, Object-Oriented Programming, and Database Management with MySQL, ensuring efficient and secure operations.***

About me



Allow me to introduce myself. I am *Muhammad Ahmad Sultan*, a highly motivated second-year student at the ***Military College of Signals***, currently pursuing a **Bachelor's degree in Software Engineering**. With unwavering determination, I am committed to achieving my goals and making significant strides in my academic journey.

**Muhammad Ahmad Sultan**

Developer

* Abstract:

The Java-based Library Management System (LMS) utilizes the Swing framework and MySQL Workbench for efficient resource, user, and transaction management. Rooted in Object-Oriented Programming, the system integrates data structures and algorithms to create a comprehensive solution. With robust search and sorting capabilities, users can quickly locate books, while streamlined borrowing processes and transaction history tracking enhance user interactions. The integrated return mechanism ensures accurate inventory and user record updates, contributing to a user-friendly and organized library environment. This system aims to efficiently manage library resources, user interactions, and transactions. It incorporates Object-Oriented Programming (OOP) concepts, data structures, and algorithms to create a comprehensive and user-friendly solution.

* History of Library Management Systems:

Library Management Systems have evolved over the years, transitioning from manual card catalogs to digital databases. The shift to automation has significantly improved the efficiency of library operations, enabling easy cataloging, tracking, and management of library resources. The advent of computer technology has facilitated the development of sophisticated Library Management Systems, streamlining tasks for librarians and enhancing the user experience. The concept of Library Management Systems dates back to the early days of computerization when libraries began to adopt automated methods to manage their resources efficiently. Over time, these systems evolved to incorporate advanced features, such as user authentication, transaction tracking, and seamless user interfaces.

* Objective:

The primary objective of the Library Management System is to provide a user-friendly platform for efficiently managing library resources, user interactions, and transactions. The system aims to automate tasks such as book and user management, borrowing and returning books, maintaining transaction history, and implementing search and sorting functionalities.

* Scope:

The Library Management System encompasses a comprehensive scope, focusing on efficient management of both books and users. It ensures seamless storage and retrieval of information, facilitating the smooth handling of borrowing and returning transactions while keeping availability statuses up-to-date. The system also maintains a detailed transaction history, recording pertinent information about books borrowed and returned by users. To enhance user experience, robust search functionalities for books and users are implemented, complemented by effective sorting operations for organized accessibility of information within the library system.

* Functionality & Implementation:

1. **Main\_Window.java (Splash Screen)**

**Functionality:**

Displays a splash screen upon application launch.

Provides an initial visual interface before loading the main application.

**Implementation:**

Utilizes Java Swing for GUI components.

May include a timer for a predetermined display time.

Invokes subsequent login functionality after the splash screen.

1. **Login.java (User Authentication)**

**Functionality:**

Manages user authentication to access the library system.

Validates user credentials against stored information.

**Implementation:**

Incorporates Swing components for a user-friendly login interface.

Utilizes JavaConnect.java for database connection.

Implements action listeners to handle login attempts.

Navigates to the Loading.java upon successful authentication.

1. **Loading.java (Loading Screen)**

**Functionality:**

Displays a loading screen after successful user authentication.

Prepares the application for user interaction.

**Implementation:**

Utilizes Swing components for visual representation.

May include background tasks or thread management for loading operations.

Transitions to the Home.java after the loading process is complete.

1. **Home.java (Menu and Dashboard)**

**Functionality:**

Serves as the main dashboard for users post-authentication.

Provides access to various functionalities and features.

**Implementation:**

Uses Swing components to create an intuitive dashboard.

Includes buttons or menu items for different operations (e.g., AddBook, Issue, Return, Statistics, Registered Students, etc.).

Implements action listeners to handle user interactions and navigation.

1. **AddBook.java (Add Books)**

**Functionality:**

Allows users to input and add new books to the library system.

**Implementation:**

Incorporates Swing components for book information input.

Utilizes JavaConnect.java for database interaction.

Implements event handling for the addition of new books.

1. **JavaConnect.java (Database Connection)**

**Functionality:**

Manages the connection between the Java application and MySQL database.

**Implementation:**

Utilizes JDBC to establish and manage connections.

May include methods for executing queries, updates, and handling database transactions.

Ensures secure and efficient communication between the application and the database.

1. **AllBooks.java (View All Books)**

**Functionality:**

Displays a comprehensive list of all books available in the library.

**Implementation:**

Utilizes Swing components for an organized and user-friendly display.

Retrieves book information from the database using JavaConnect.java.

Allows users to scroll through the list of books.

1. **Issue.java (Issue Books)**

**Functionality:**

Manages the process of issuing books to registered students.

**Implementation:**

Incorporates Swing components for student and book selection.

Updates the database to reflect the issued books and maintains transaction records.

May include validation to ensure proper book availability.

1. **Return.java (Return Books)**

**Functionality:**

Handles the return of borrowed books, updating availability and transaction records.

**Implementation:**

Uses Swing components for book return interface.

Interacts with JavaConnect.java to update the database.

Calculates any applicable fines or penalties.

1. **Statistics.java (Transaction History)**

**Functionality:**

Presents statistical information about library transactions, such as issue and return history.

**Implementation:**

Utilizes Swing components to display statistical data.

Retrieves transaction information from the database using JavaConnect.java.

May include graphical representations for enhanced data visualization.

1. **RegisteredStudents.java (View Registered Students)**

**Functionality:**

Displays a list of all registered students in the library system.

**Implementation:**

Utilizes Swing components to organize and present student information.

Retrieves student data from the database using JavaConnect.java.

Allows users to browse through the list of registered students.

1. **Student.java (Add Student)**

**Functionality:**

Enables the addition of new students to the library system.

**Implementation:**

Incorporates Swing components for student information input.

Interacts with JavaConnect.java for database updates.

Implements event handling for student registration.

1. **About.java**

**Functionality:**

Provides information about the Library Management System application.

**Implementation:**

Utilizes Swing components to display application details.

May include version information, development credits, and other relevant details.

1. **Register.java (User Registration)**

**Functionality:**

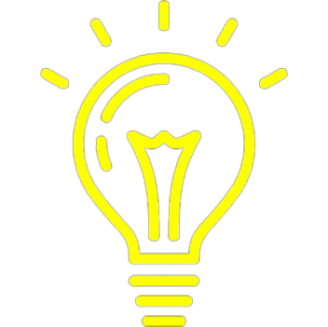
Manages the registration of new users, such as librarians or administrators.

**Implementation:**

Incorporates Swing components for user registration input.

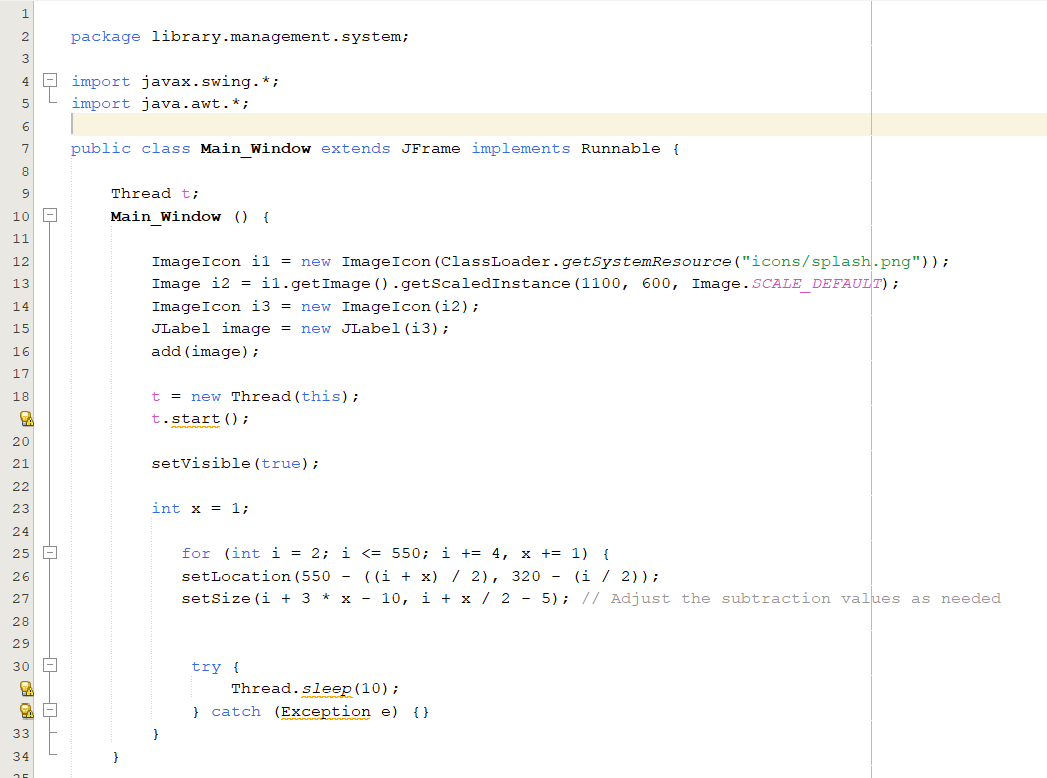
Utilizes JavaConnect.java for database interaction to store user information securely.

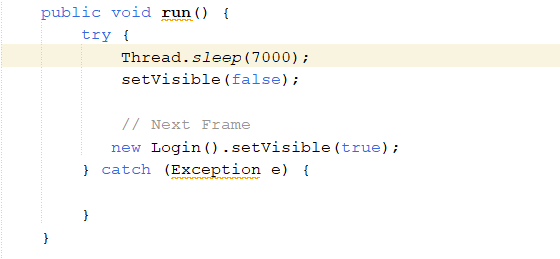
Implements event handling for user registration processes.

****

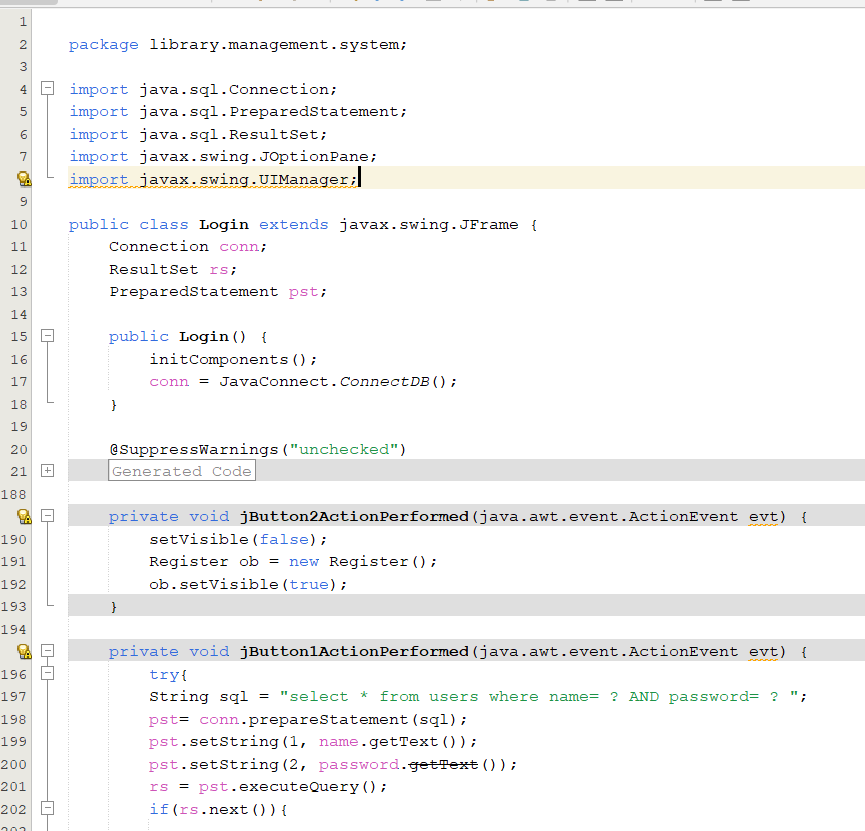
Java Classes

* ***Main\_Window.java***

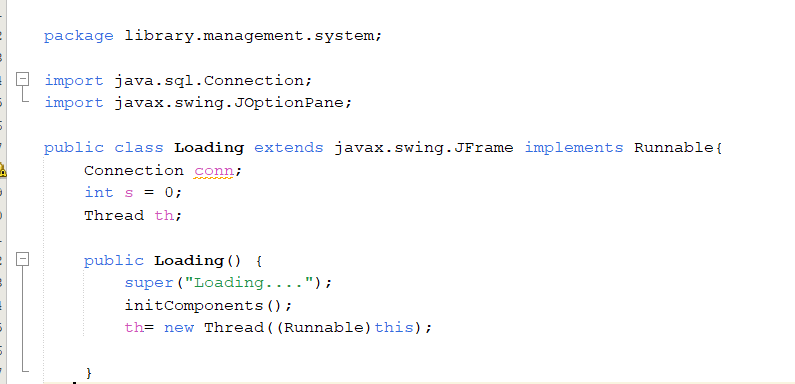




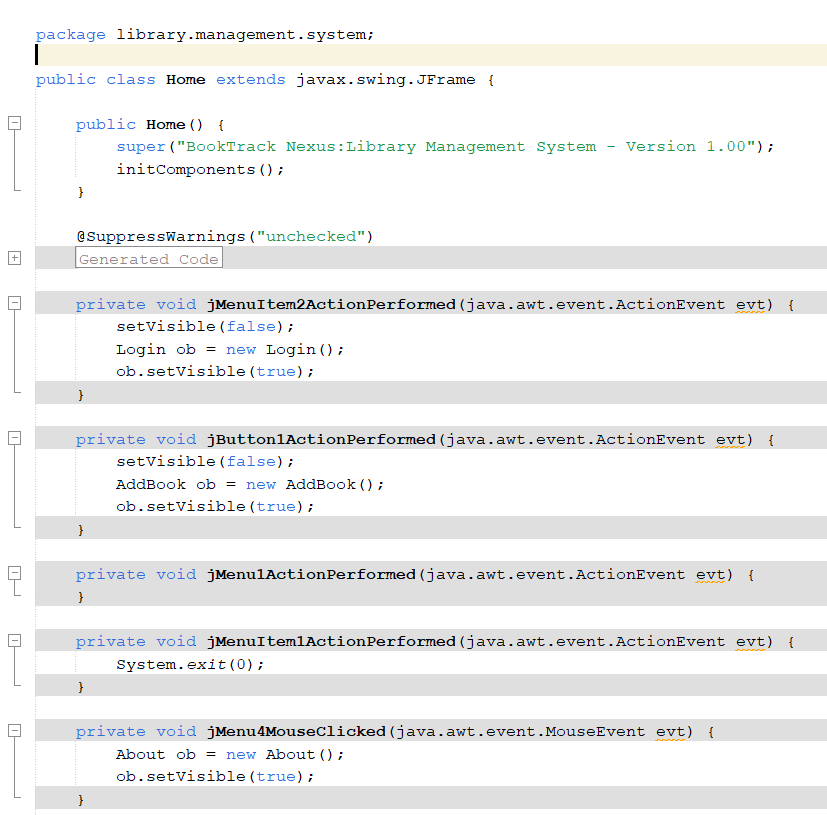
* ***Login.java***



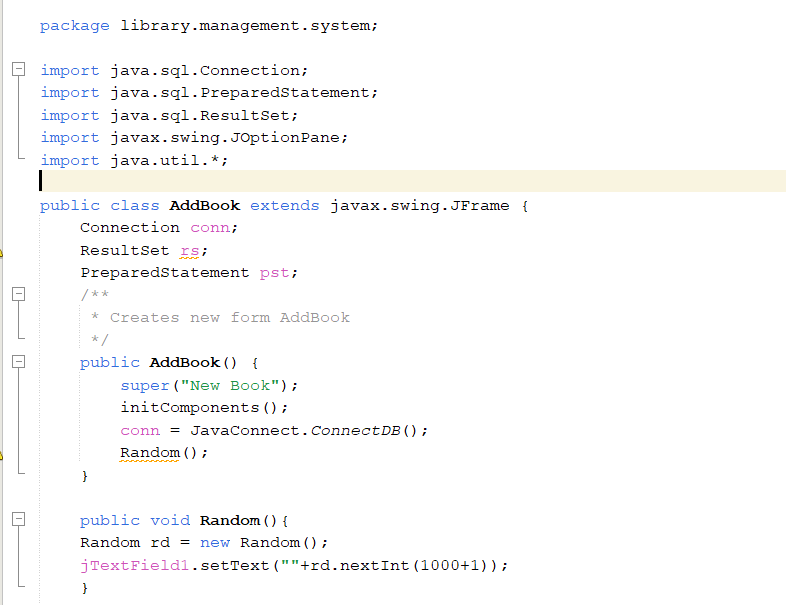
* ***Loading.java***



* ***Home.java***



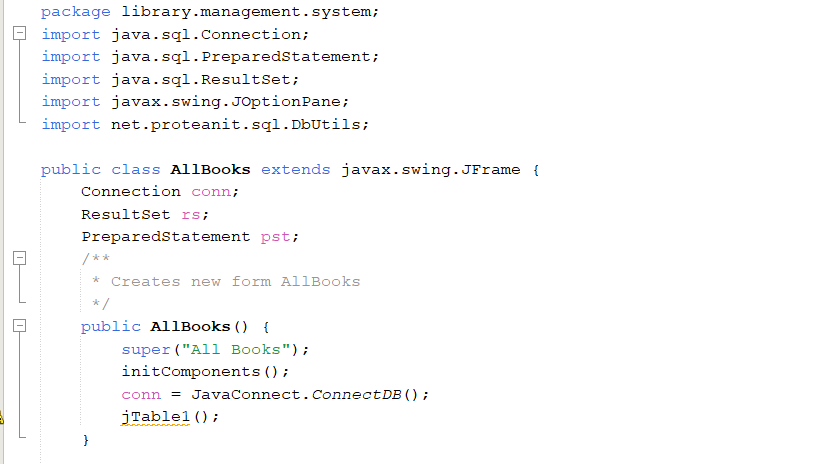
* ***AddBook.java***

****

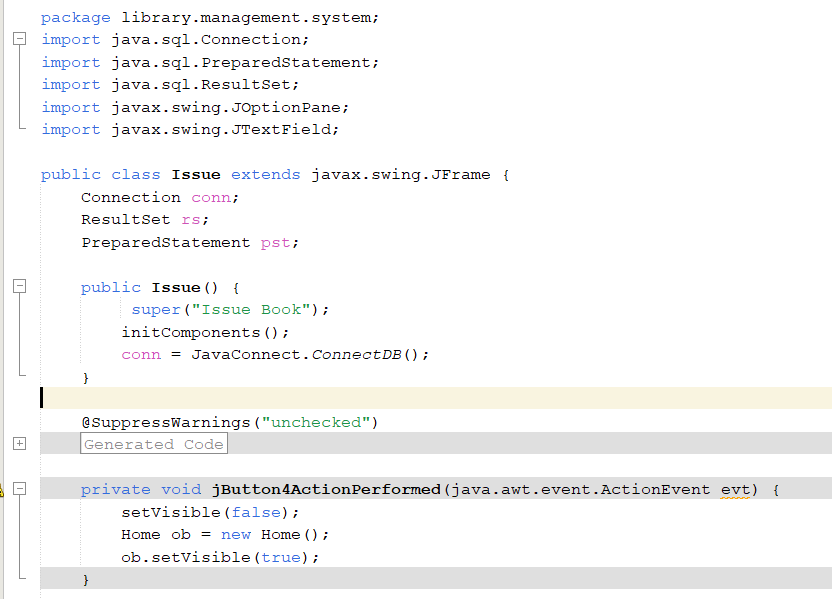
* ***JavaConnect.java***

****

* ***AllBooks.java***



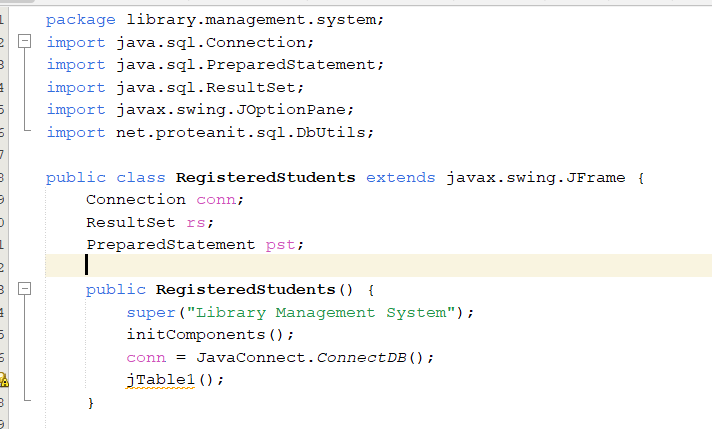
* ***Issue.java***

****

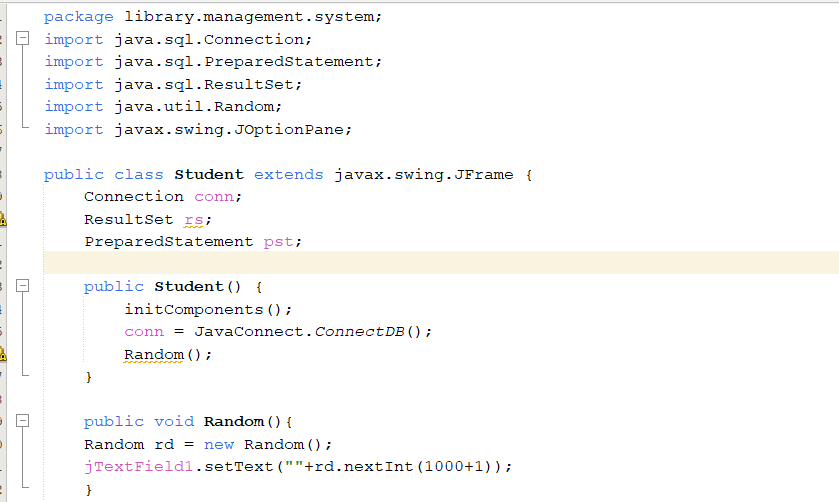
* ***Register.java***



* ***RegisteredStudents.java***



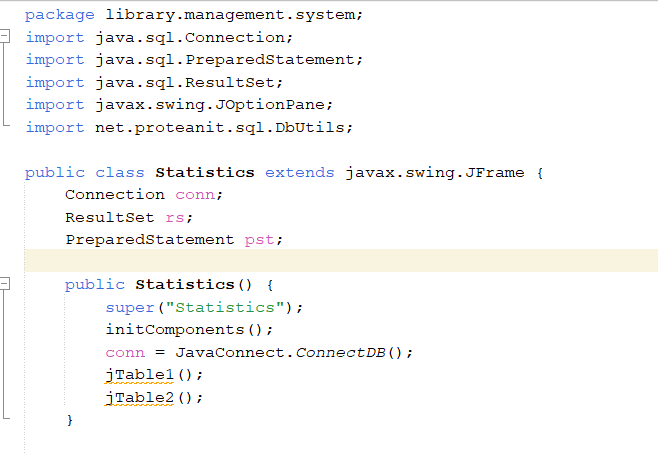
* ***Student.java***



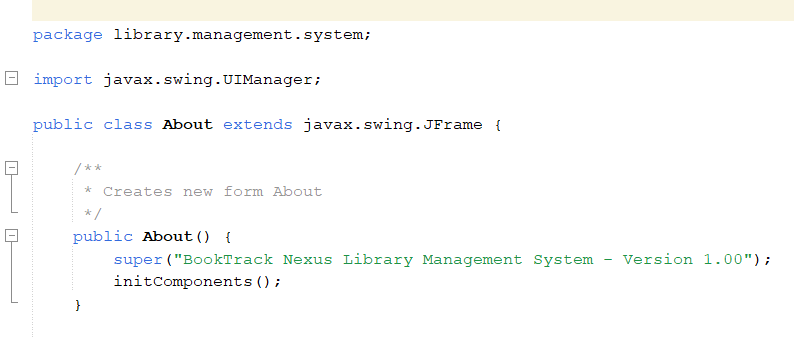
* ***Return.java***



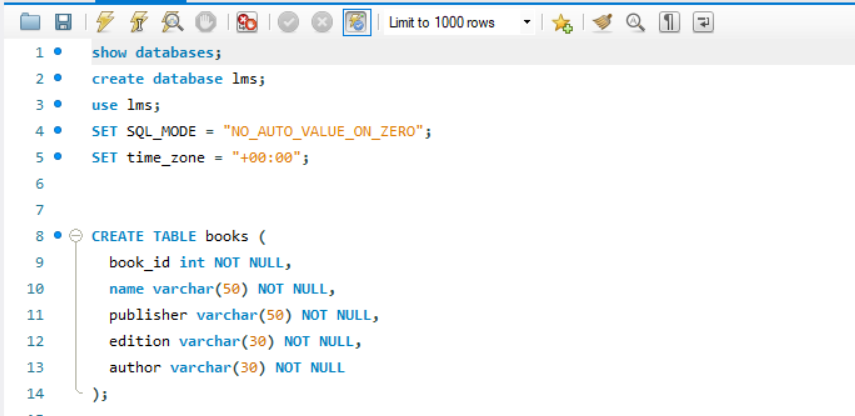
* ***Statistics.java***

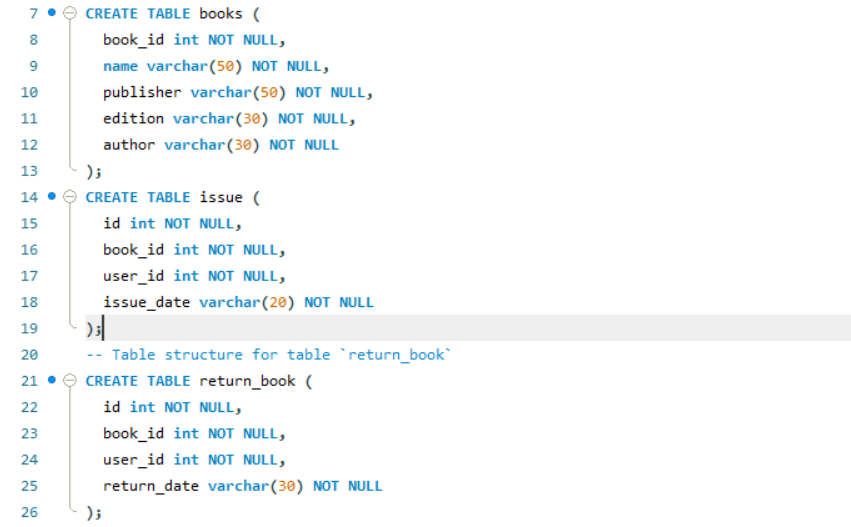
****

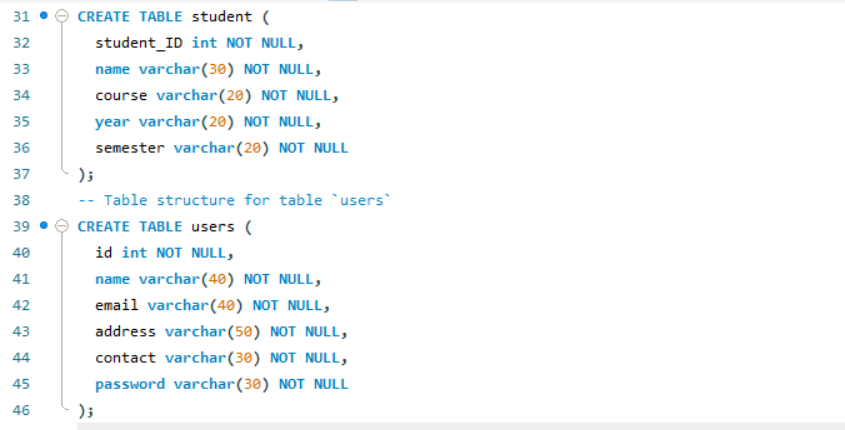
* ***About.java***

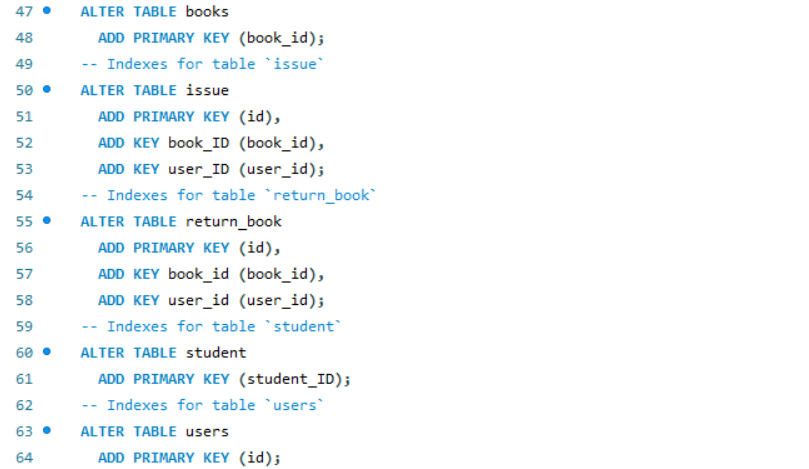


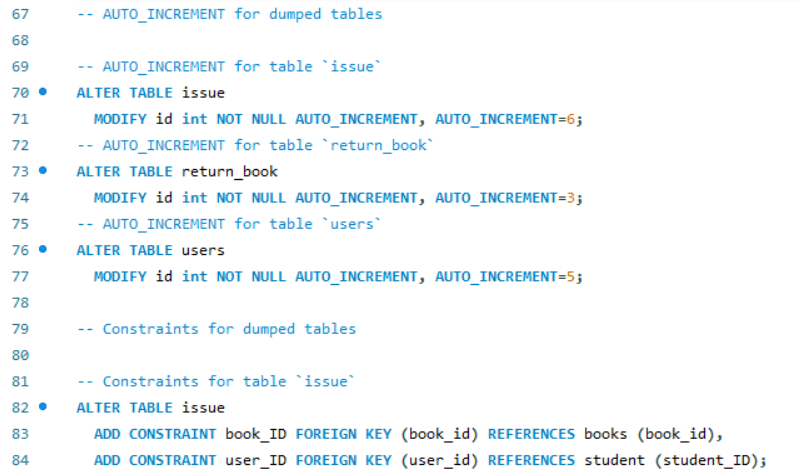
MySQL Database Queries













Explanation:

**1.Database Initialization:**

The initial command "SHOW DATABASES;" lists existing databases.

"CREATE DATABASE LMS;" creates a new database named "lms."

"USE LMS;" selects the "lms" database for subsequent operations.

**2. SQL Configuration:**

"SET SQL\_MODE = 'NO\_AUTO\_VALUE\_ON\_ZERO';" configures SQL mode to disallow automatic increment for zero values.

"SET time\_zone = '+00:00';" sets the time zone for the database to UTC.

**3. Books Table:**

"CREATE TABLE books" defines the structure for the 'books' table with fields: book\_id, name, publisher, edition, and author.

"ALTER TABLE books ADD PRIMARY KEY (book\_id);" establishes 'book\_id' as the primary key.

**4. Issue Table:**

"CREATE TABLE issue" defines the 'issue' table with fields: id, book\_id, user\_id, and issue\_date.

"ALTER TABLE issue" adds primary and foreign key constraints linking 'book\_id' to 'books' and 'user\_id' to 'student.'

**5. Return Book Table:**

"CREATE TABLE return\_book" defines the 'return\_book' table with fields: id, book\_id, user\_id, and return\_date.

"ALTER TABLE return\_book" adds primary and foreign key constraints linking 'book\_id' to 'books' and 'user\_id' to 'student.'

**6. Student Table:**

"CREATE TABLE student" defines the 'student' table with fields: student\_ID, name, course, year, and semester.

"ALTER TABLE student ADD PRIMARY KEY (student\_ID);" establishes 'student\_ID' as the primary key.

**7. Users Table:**

"CREATE TABLE users" defines the 'users' table with fields: id, name, email, address, contact, and password.

"ALTER TABLE users ADD PRIMARY KEY (id);" sets 'id' as the primary key.

**8. Auto-increment and Constraints:**

"ALTER TABLE issue MODIFY id" and "ALTER TABLE return\_book MODIFY id" configure auto-increment for 'id' in the 'issue' and 'return\_book' tables, respectively.

"ALTER TABLE users MODIFY id" sets auto-increment for 'id' in the 'users' table.

Foreign key constraints ensure data integrity, linking 'book\_id' and 'user\_id' in 'issue' and 'return\_book' tables to 'books' and 'student' tables, respectively.

* Detailed Implementation:
* ***Technologies Used***

**Java Swing (GUI Development)**

Java Swing is employed to create a graphical user interface (GUI) for the Library Management System. It enables the development of interactive and visually appealing interfaces, enhancing the user experience.

**MySQL Workbench (Database Management)**

MySQL Workbench serves as the database management system for the application. It facilitates the creation and management of tables storing crucial information about books and users. Key concepts include relational tables, establishing relationships, and optimizing database performance through efficient indexing.

* ***Database Concepts***

The application utilizes MySQL Workbench for efficient database management. Key database concepts include:

**Relational Tables**

Structured tables are designed to store information about books and users. This relational approach ensures data integrity and facilitates efficient data retrieval.

**Relationships Between Tables**

Establishing relationships between tables, such as a one-to-many relationship between books and transactions, ensures the coherence of the database and supports complex queries.

**Efficient Indexing for Faster Retrieval**

Strategic indexing is implemented to enhance data retrieval speed. Indexing allows the system to locate and retrieve specific records more swiftly, contributing to overall system performance.

* ***Data Structures and Algorithms***

**Arrays and Lists**

Arrays and lists are fundamental data structures used for storing and managing information about books and users. They enable efficient storage and retrieval operations, ensuring optimal performance.

**Sorting Algorithms**

Sorting algorithms are implemented to arrange book records in ascending order by title or author. This enhances the readability of the data and facilitates quicker search operations.

* ***Additional Data Structures***

**Hash Tables:** Hash tables can be utilized for rapid search operations, especially when looking up information based on unique identifiers like book ISBNs. This ensures constant-time complexity for retrieval, enhancing search efficiency.

**Trees (e.g., Binary Search Trees):** Binary Search Trees (BST) can be employed for maintaining a sorted order of records. For instance, a binary search tree based on book titles allows for efficient insertion, deletion, and search operations while preserving a hierarchical structure.

**Queues and Stacks:** Queues and stacks can be useful in managing transactions. For instance, a queue can represent a first-in, first-out (FIFO) system for book borrowings, ensuring fairness in serving requests. Stacks may be used for tracking the history of returned books.

**Lists:** Linked lists can be utilized to store and manage data dynamically. This flexibility is particularly useful for scenarios where the number of books or users may change frequently. A linked list of transactions, for example, provides an adaptable structure for handling borrowing and return history.

* ***DSA Concepts***

**Object-Oriented Programming (OOP)**

The system leverages Object-Oriented Programming principles for code organization and modularity. Classes are utilized for encapsulation and abstraction, promoting a modular and maintainable codebase.

**Efficient Search Operations**

Arrays and Lists for Search

Data structures like arrays and lists are employed for efficient search operations. The system allows users to search for books based on title, author, or ISBN, leveraging these data structures for quick and accurate results.

**Algorithmic Search Implementations**

Algorithmic search techniques are implemented to enhance search functionality. These may include binary search for sorted data structures, ensuring optimal search efficiency.

* Input and Output Specifications
* **Input**

1. User input via the GUI for operations like adding books, issuing books, etc.
2. Data input from the database for displaying information about books, users, and transactions.

* **Output**

1. Displaying information on the GUI based on user queries and system operations.
2. Transaction records stored in the database for future reference.

* Usage of OOP Concepts in Library Management System

**1. Classes and Objects:**

Different classes, such as Book, User, Issue, Return, and JavaConnect, represent entities within the library management system.

Objects of these classes are instantiated to represent individual books, users, transactions, and database connections.

**2. Inheritance:**

Inheritance establishes relationships between classes in the library management system.

A base class like LibraryItem can be created, extended by subclasses like Book, Issue, and Return to inherit common attributes and behaviors related to library items.

**3. Encapsulation:**

Encapsulation is applied to classes to encapsulate data and functionality within the library management system.

For example, the Book class encapsulates book-specific details like name, author, and publisher, providing methods to access and modify this information securely.

**4. Method Overriding:**

Method overriding is employed when there is a need to provide custom implementations of inherited methods in the library management system.

For instance, the displayDetails() method in the Book class can override the same method in the LibraryItem class to display book-specific information.

**5. Constructor:**

Constructors are utilized in various classes of the library management system to initialize object properties when they are created.

For example, a constructor in the Issue class can be used to set initial values such as issue ID, book ID, user ID, and issue date.

**6. Polymorphism:**

Polymorphism is applied in the library management system to treat objects of different classes as objects of a common superclass or interface.

For flexibility and code reusability, a method that processes book transactions can accept both Book and Issue objects, treating them as instances of a common LibraryItem superclass.

***By leveraging these OOP principles, the library management system project ensures a well-structured and maintainable codebase, facilitating efficient management of library resources and user interactions***

* Future Enhancements for Library Management System

**1. User Roles and Permissions:**

Implement role-based access control for librarians, administrators, and users.

**2. Advanced Search and Filtering:**

Improve search functionality with advanced filters like genre and publication date.

**3. Integration with External APIs:**

Integrate external book databases or library catalog APIs for real-time updates.

**4. Mobile Application Development:**

Develop a mobile app for on-the-go access to the library system.

**5. RFID Technology Integration:**

Automate book transactions with RFID technology.

**6. User Notifications and Alerts:**

Implement notifications for due dates, overdue books, and announcements.

**7. Data Analytics and Reporting:**

Introduce analytics for user behavior and library usage trends.

**8. Integration with Learning Management Systems:**

Sync with educational institutions' learning platforms for seamless access.

**9. Fine Management System:**

Automate fine calculations and provide online payment options.

* Output and Execution Flow in Library Management System

**1. Splash Screen (Main\_Window.java):** Upon launching the Library Management System, users are greeted with a visually appealing splash screen, providing a brief and engaging introduction to the application. The splash screen serves as an initial point of interaction and sets the tone for the user experience.

**2. Login Window (Login.java and Register.java):** Following the splash screen, users are directed to the login window, where they can authenticate themselves to access the library system. This window also provides the option for new users to register. The registration process ensures that users can sign up securely, expanding the user base and enhancing system accessibility.

**3. Loading Screen (Loading.java):** Upon successful login or registration, users are directed to a loading screen. This screen indicates that the system is initializing and preparing for user interaction. It ensures a smooth transition between authentication and the main application, providing a seamless user experience.

**4. Home Screen (Home.java):** After the loading process, users are presented with the main home screen, featuring a dashboard that serves as a central hub for various library management operations. The home screen is designed with user-friendly navigation, allowing users to access functionalities like adding students, adding books, viewing transaction history, searching, sorting, issuing and returning books, signing out, accessing information about the application and the developer, and exiting the system.

**5. Dashboard Operations:**

**Add Student and Add Book:** Users can input information to add new students and books to the system, contributing to an updated and comprehensive database.

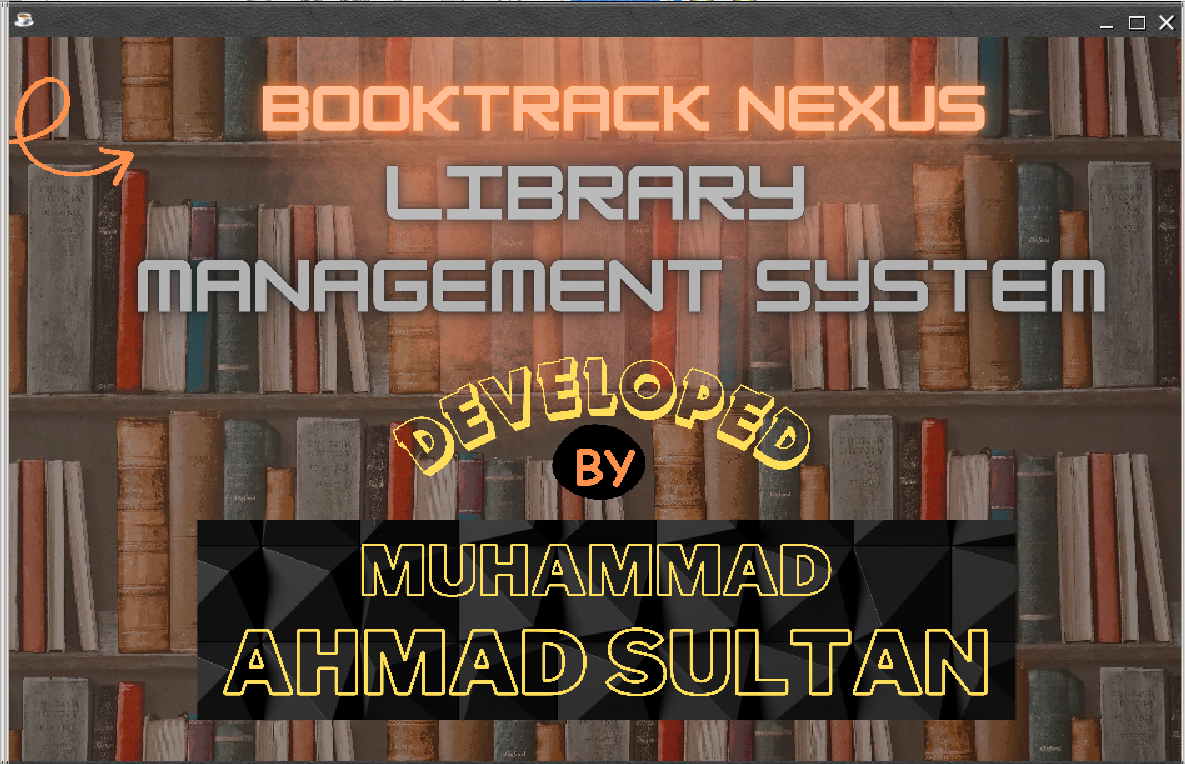
**View Transaction History:** The transaction history feature enables users to view a detailed log of all library transactions, providing transparency and accountability.

**Search and Sort:** The search and sort functionalities allow users to quickly locate specific books or students based on various criteria, enhancing the efficiency of information retrieval.

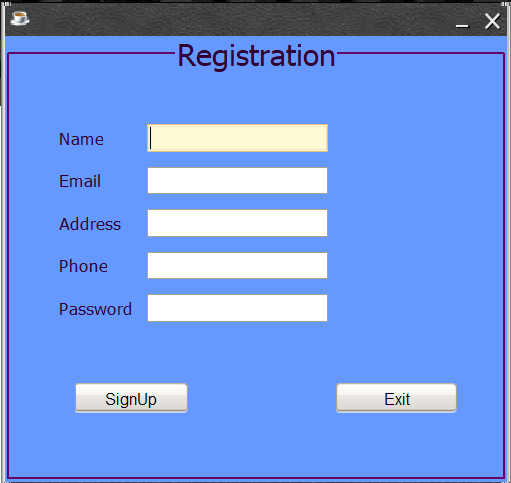
**Issue and Return Books:** Users can seamlessly manage the borrowing and returning processes, ensuring accurate tracking of book availability and transaction records.

**About and Exit:** The dashboard provides information about the application and the developer, creating a user-friendly environment. Users can also choose to exit the system when done with their library tasks.

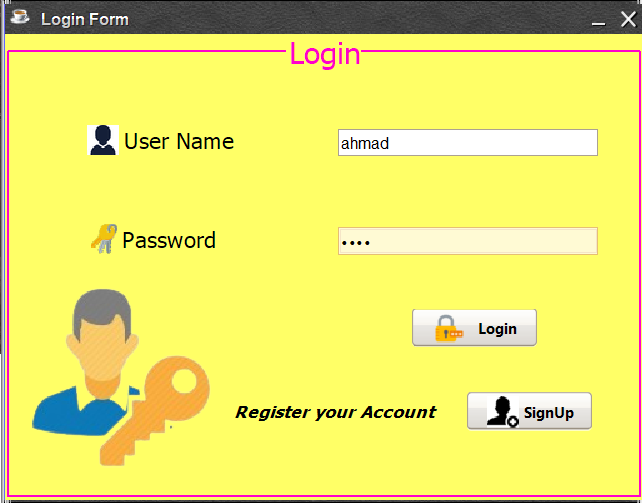
* Output Display:
* ***Main\_Window.java***



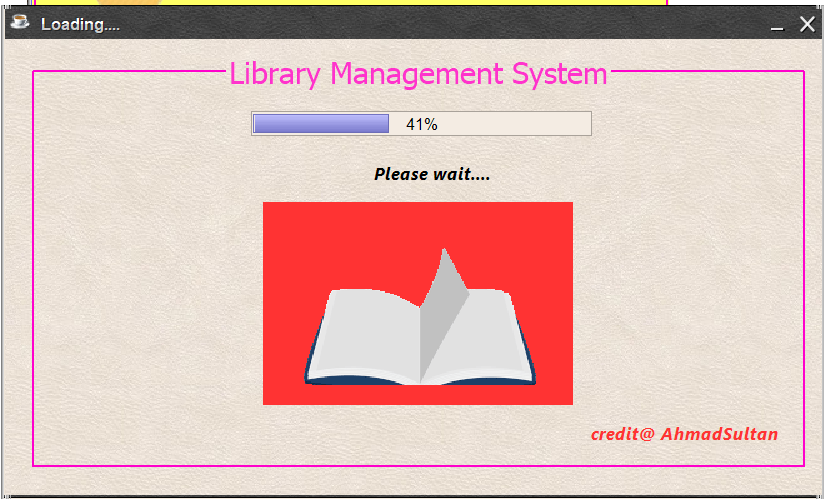
* ***Register.java***



* ***Login.java***



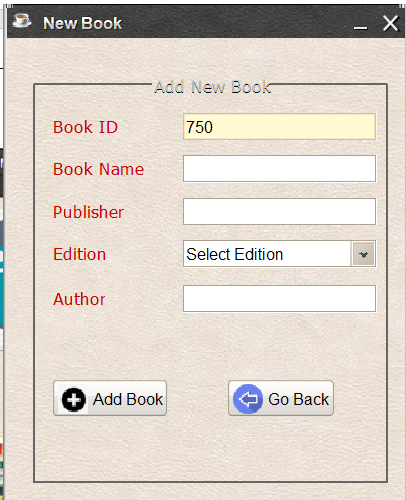
* ***Loading.java***



* ***Home.java***

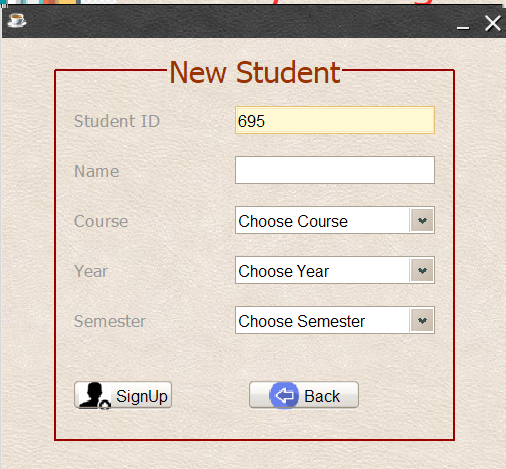


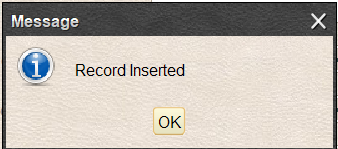
* ***AddBook.java***



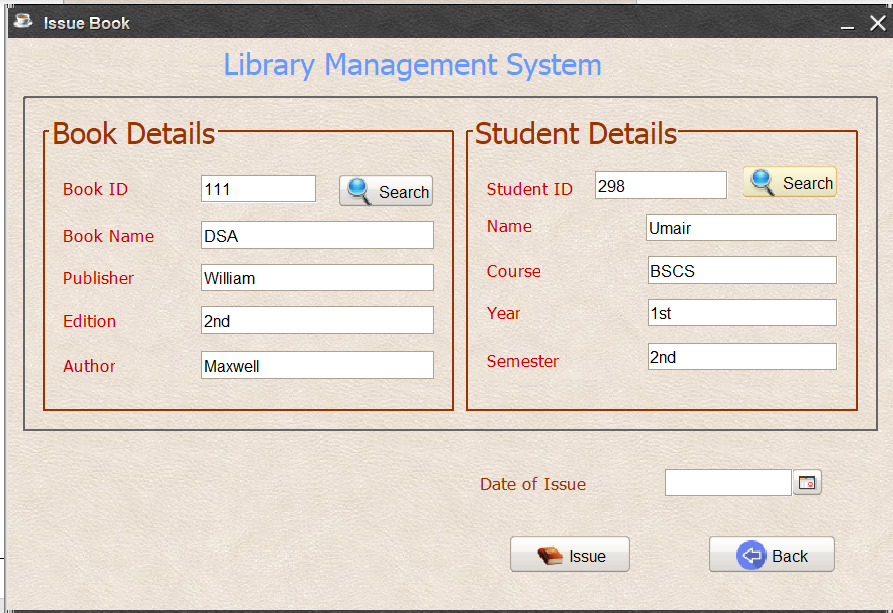


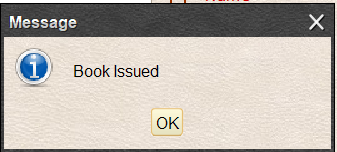
* ***Student.java***



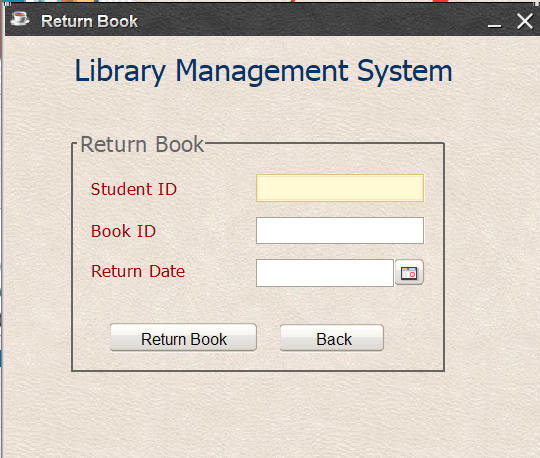


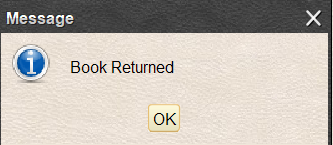
* ***Issue.java***



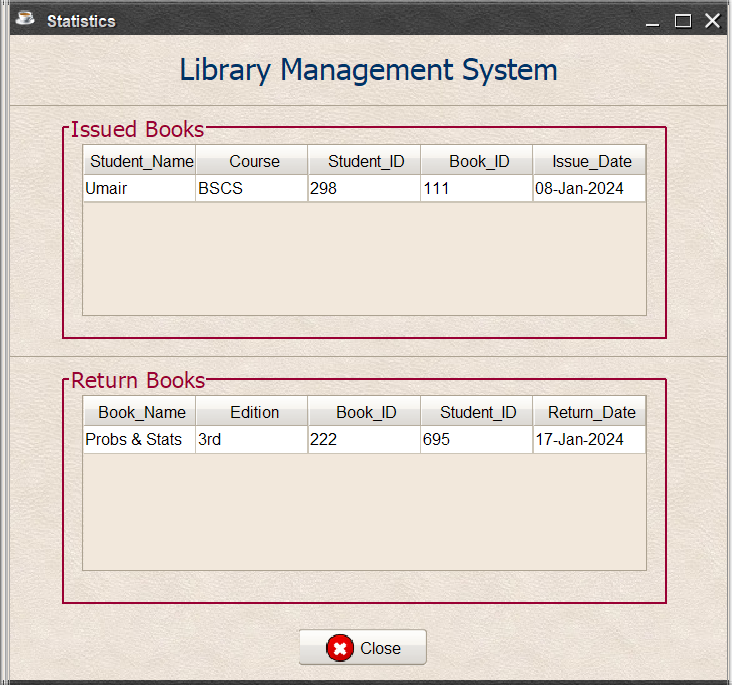


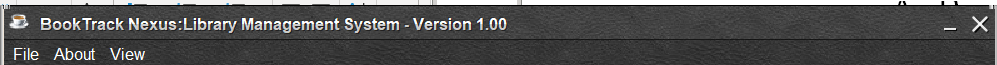
* ***Return.java***

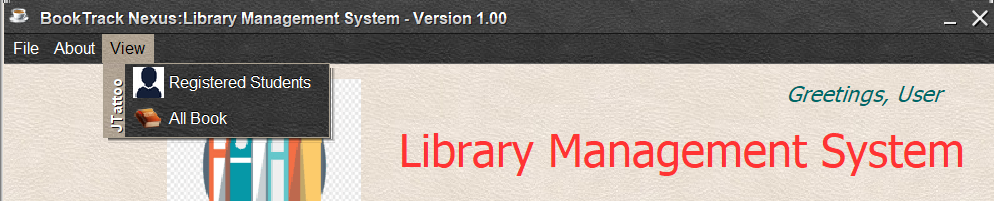


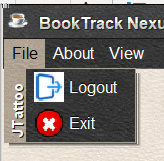


* ***Statistics.java***

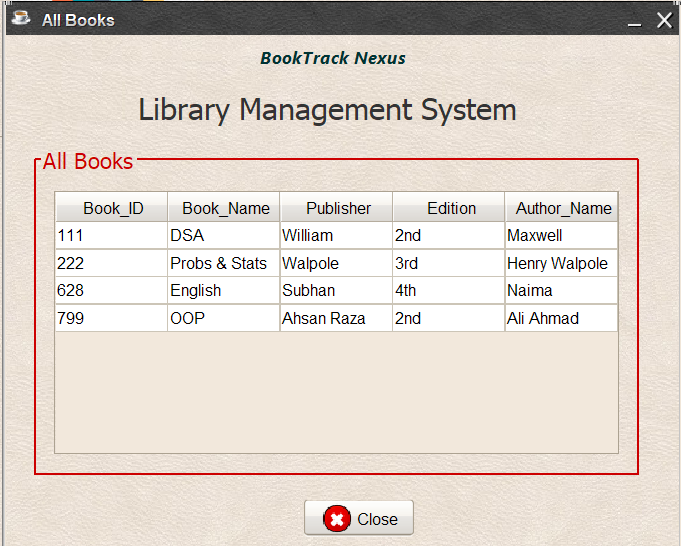




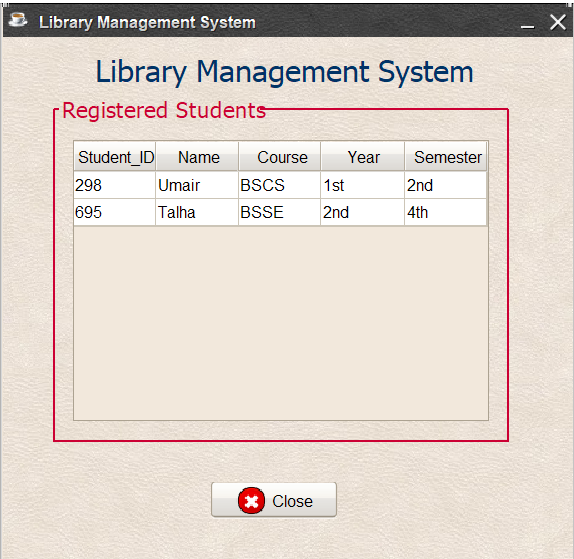




* ***AllBooks.java***



* ***RegisteredStudents.java***



* ***About.java***



* Potential Questions:
* How does your Library Management System leverage data structures to optimize information retrieval and enhance overall system performance?
* In what ways does the implementation of Java Swing contribute to creating a user-friendly and intuitive interface for your library application, ensuring a seamless user experience?
* Can you elaborate on the security measures implemented in your project to safeguard user authentication and protect sensitive information within the Library Management System's database?
* Conclusion:

In summary, the Library Management System project is a sophisticated and user-centric solution crafted to streamline the management of library resources, user interactions, and transactions. Developed using Java with the Swing framework and MySQL for database management, the project embodies key Object-Oriented Programming (OOP) principles, resulting in a well-structured and easily maintainable codebase. The project's execution flow unfolds seamlessly, commencing with an engaging splash screen, progressing through user authentication and loading processes, and culminating in a feature-rich home screen dashboard. This dashboard seamlessly integrates functionalities like adding students and books, viewing transaction history, searching, sorting, issuing and returning books, while also offering insights about the application and the developer. Crucially, data structures play a pivotal role in optimizing system efficiency, facilitating rapid and effective retrieval, storage, and manipulation of information regarding books, users, transactions, and student details. The incorporation of these data structures, coupled with MySQL database management and Java Swing GUI, underscores the project's commitment to delivering a robust, functional, and user-friendly Library Management System.

* Reference Citations:

<https://www.youtube.com/>

<https://www.w3schools.com/>

<https://www.codecademy.com/>

<https://www.learnjavaonline.org/>

<https://www.geeksforgeeks.org/learn-data-structures-and-algorithms-dsa-tutorial/>

https://www.homeandlearn.co.uk/java/java\_and\_databases.html

***THE END***