**Project Documentation**

**By Team Rohok**

**Contents:**

1. **Overview of the Project**
2. **Key Features**
3. **OOP Concepts Used**
4. **Each class Description**
5. **Project Phases**
6. **Instructions for Running**
7. **Improvements For this Project.**

**Overview of the Project**

The Library Management System is a software application designed to streamline library operations such as managing books, members, and transactions. It provides users with the ability to generate reports for books, members, and transactions using a graphical user interface. This project showcases the application of Object-Oriented Programming concepts, ensuring maintainable, scalable, and efficient code design.

**Key Features**

1. Book Management: Add, update, and display book records with details like title, author, genre, and availability.
2. Member Management: Maintain member details such as ID, name, and contact information.
3. Transaction Management: Handle book issuance, returns, and fines.
4. Report Generation: Generate tabular reports for books, members, and transactions.
5. User Interface: Intuitive GUI for interacting with the system.

**OOP Concepts Used**

**1. Class and Object**

* Classes are used to represent entities like **Books**, **Members**, **Transactions**, and **Reports**.
* Objects are created to instantiate these classes and store data (Book, Member).

**2. Encapsulation**

* Private fields in classes ensure data is hidden from outside interference.
* Getter and Setter methods provide controlled access to data (getTitle() for Books).

**3. Inheritance**

* Classes like AdminPanelGUI and ReportsGUI inherit from the base Swing components such as JFrame and JPanel.

**4. Polymorphism**

* Overriding is used in Swing components to provide custom event-handling behavior for buttons and tabs.

**5. Abstraction**

* The project abstracts details of book, member, and transaction management through well-defined classes and methods.

**6. Association**

* Relationships between classes, such as:
  + Transaction associates Members and Books.
  + ReportsGUI depends on Library to fetch data for generating reports.

**7. Modularity**

* Separate classes handle different functionalities:
  + Library: Core library operations.
  + ReportsGUI: GUI for displaying reports.
  + Transaction: Handles book issuance and return details.

**Each class Description**

**The files of this classes are written in the notepad of file type java.**

Books:

* Represents a book in the library.
* Attributes include title, author, genre, published year, availability, and a unique book number.
* Methods:
  + Static method to generate book numbers.
  + Constructors for different initialization scenarios.
  + Getters and setters for book properties.
  + Methods to display book details, borrow, and return a book.

BaseMember:

* Represents a basic library member.
* Attributes: name and contact information.
* Methods:
  + Getters for name and contact information.
  + Method to update contact information.

Members (inherits from BaseMember):

* Represents a library member with borrowing capabilities.
* Attributes include a unique member ID, a list of borrowed books, and a maximum borrow limit.
* Methods:
  + Borrow and return books with validations.
  + Getters for borrowed books and eligibility to borrow.

BaseTransaction:

* Represents the basic structure of a book borrowing transaction.
* Attributes: member, book, issue date, due date, return date, and fine.
* Method to calculate overdue fines.

Transaction (inherits from BaseTransaction):

* Represents a detailed transaction with a unique transaction ID.
* Additional methods for fine calculation and printing transaction details.

Library:

* Represents the library system.
* Attributes include the library name, address, contact info, books, members, librarians, transactions, and fine calculator.
* Methods for managing books, members, and librarians; issuing and returning books; generating reports.

LibraryOperations (interface):

* Defines operations for library management, such as adding/removing books and members, issuing/returning books, and generating reports.

Librarian (inherits from BaseMember):

* Represents a librarian in the system.
* Attributes: ID, role, and years of experience.
* Methods for librarian-specific tasks.

FineCalculator (abstract class):

* Defines a structure for calculating overdue fines.
* Subclasses implement specific fine calculation logic.

StandardFineCalculator (inherits from FineCalculator):

* Implements a standard fine calculation method based on overdue days.

Report:

* Generates reports for books, members, and transactions in the library.
* Attributes include a list of books, members, and transactions.

Search:

* Provides search functionalities for books and members by title or name.

AdminPanel:

* Provides an administrative interface for managing the library, updating settings, and generating reports.

Settings:

* Represents library settings, including the name, address, contact info, and fine rate.
* Methods for viewing and updating settings.

UserInterface:

* Provides a user interface for searching books, borrowing, and returning books.

LibraryManagementTest:

* A test class for the library system. Includes initial data setup and login system for admin and user roles.

MainMenu:

* Main menu for navigating between user interface, admin panel, and other features.

Statistics:

* Generates library usage statistics, such as total books and members.

UserInterfaceGUI:

* GUI for the user interface, allowing users to search, borrow, and return books.

AdminLoginGUI:

* GUI for admin login.

AdminPanelGUI:

* GUI for the admin panel to manage books, members, and generate reports.

ReportsGUI:

* GUI for generating detailed reports about books, members, and transactions.

SettingsGUI:

* GUI for updating and viewing library settings.

CreateLoginGUI:

* GUI for creating new admin or user logins.

Main:

* Entry point for the application.
* Initializes the library, settings, and search functionalities, and launches the main menu.

**Project Phases**

**Phase – 1:**

We have done 1st phase of project by running the project without GUI to check all classes working properly and interlinked, through terminal command by running the LibraryManagementTest class.

**Phase – 2:**

We have done 2nd phase of project by running the project with GUI to check all classes working properly and interlinked, through terminal command by running the Main class.

**Code Overview**

The project employs a layered architecture:

* **Frontend**: GUI components using Java Swing.
* **Backend**: Data operations through Library and associated classes.

**Instructions for Running**

**Step-1:** Ensure your PC have java version of java 22.0.2

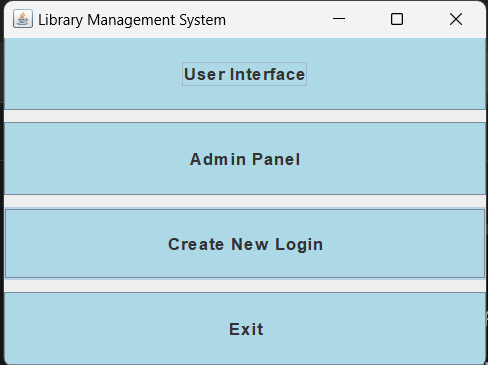
**Step-2:** Save the code file in one directory

**Step-3:** open the terminal command from the directory where codes are saved

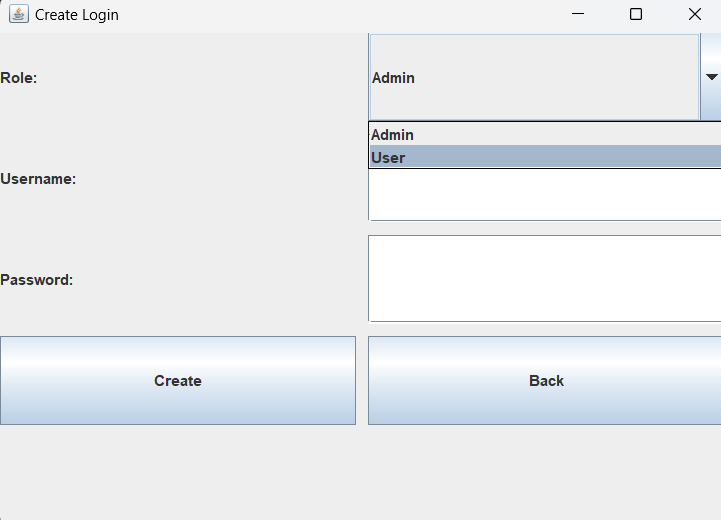
**Step-4:** run the command javac \*.java

**Step-5**: run the command java Main

You open to our **MainMenuGUI**  
It contains User Interface, Admin Panel, Create New Login and Exit



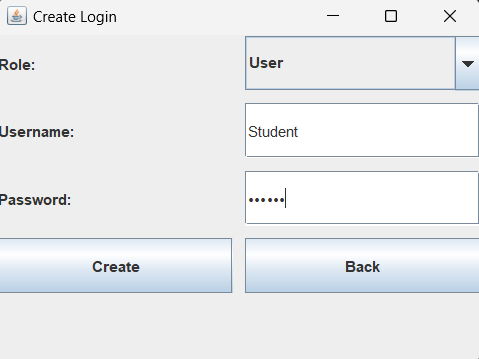
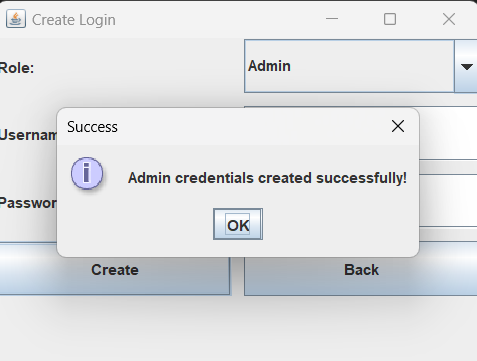
**Open Create New Login**



Once we open the Create New Login It shows Role,Username,Password   
Select what role we are going to create ?   
You can choose between ADMIN and USER   
After selecting Create a Username (of Admin or User) with a Password

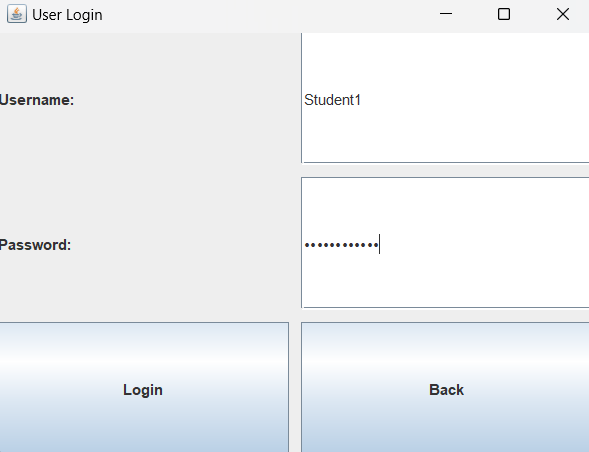
For Example:

**For Creating a new User**                                               **For Creating a new Admin**

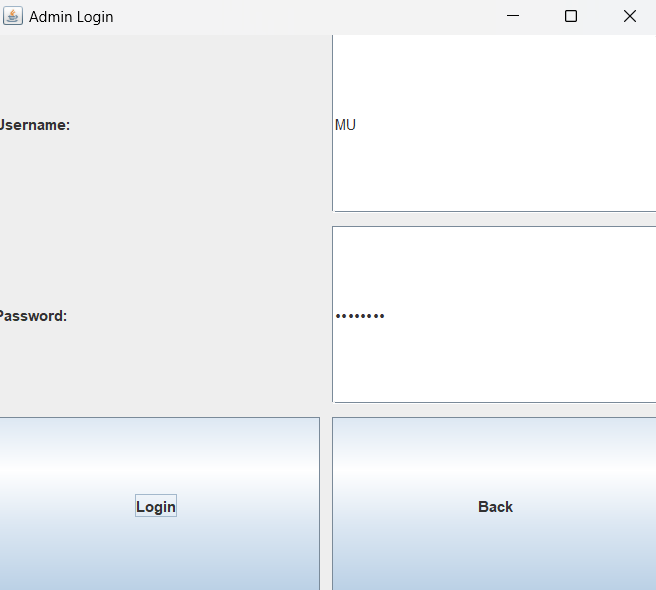
            

After creating the User and Admin, login with the password to User login or Admin Login

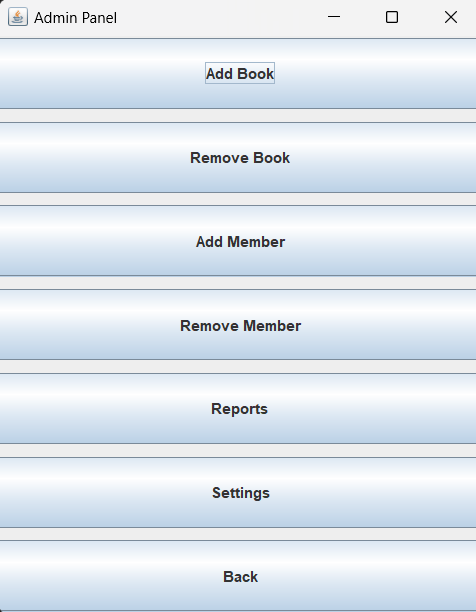
**User Login**



**Admin Login**

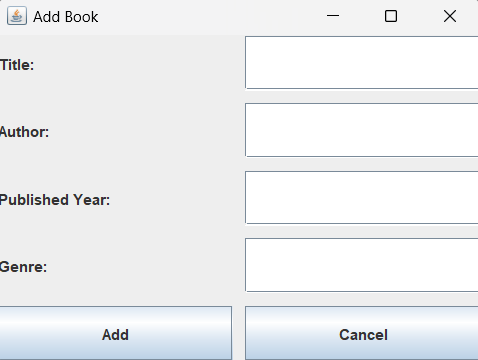
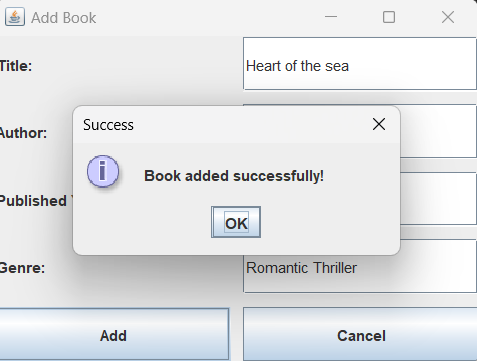


If you login to Admin page we get **Admin Panel**

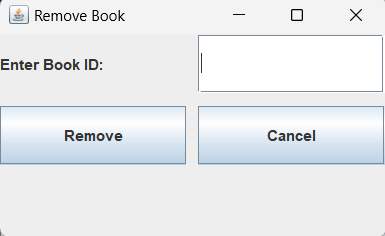


After logging in to Admin we get Add Book, Remove Book, Add Member, Remove Member, Reports, Settings, back (to go back of Admin panel)

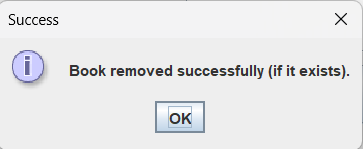
**Add Book**

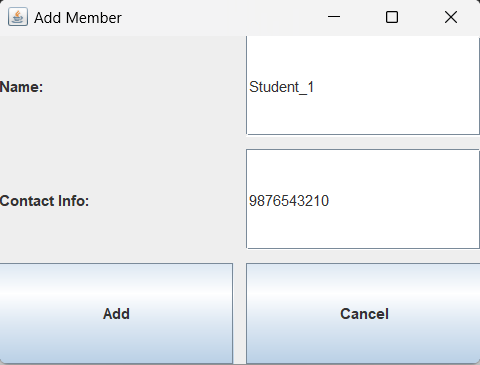
**Remove Book**

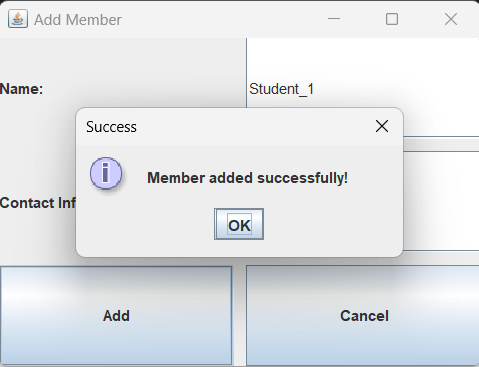


Enter the Book ID and Press Remove

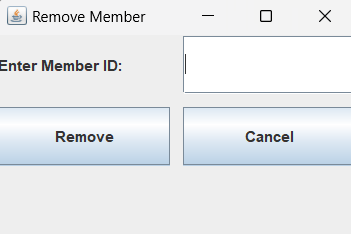


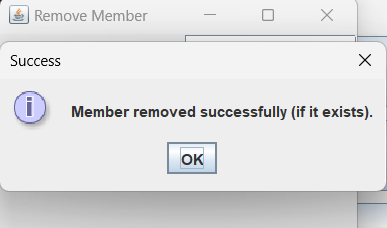
**Add Member**



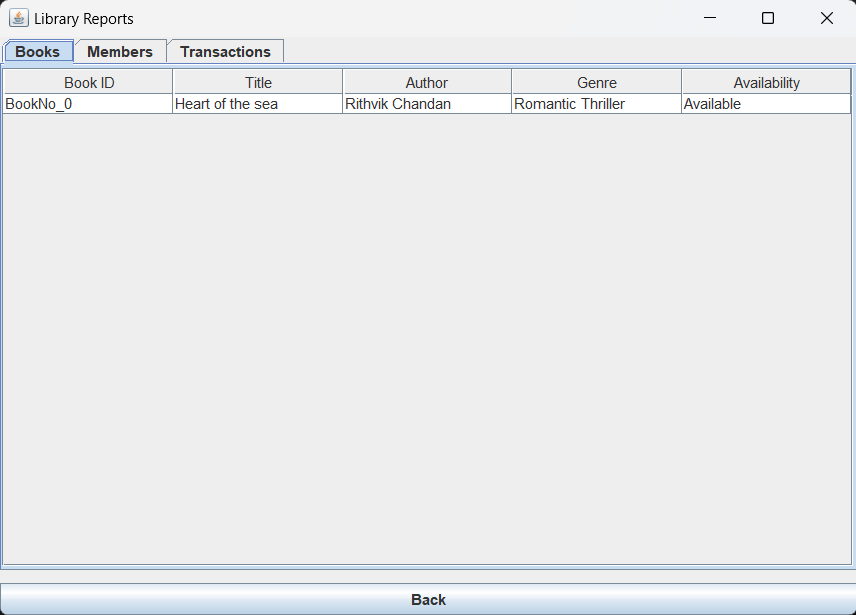


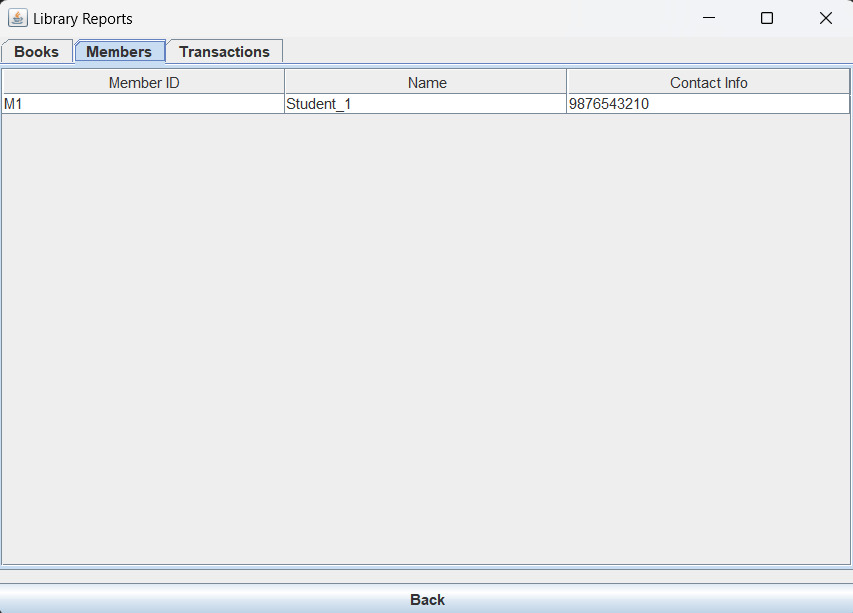
**Remove Member**

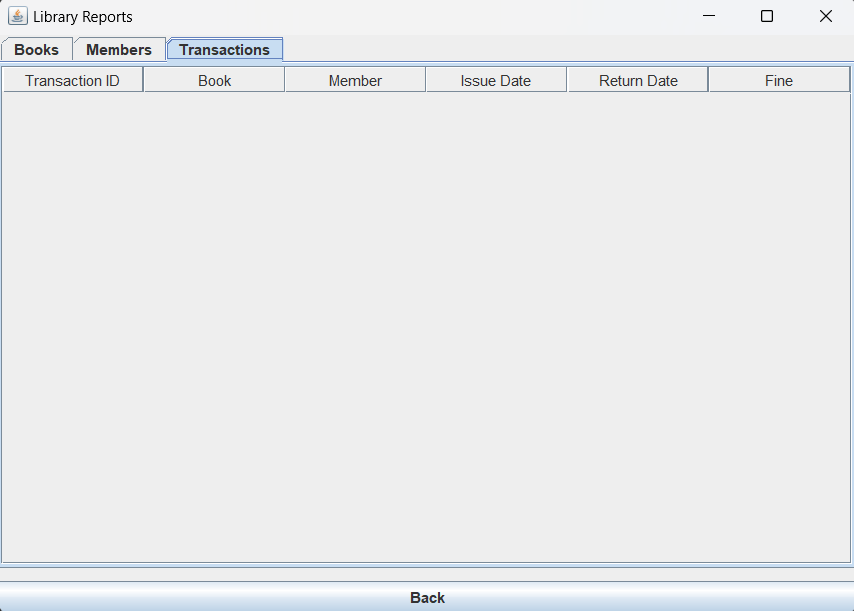




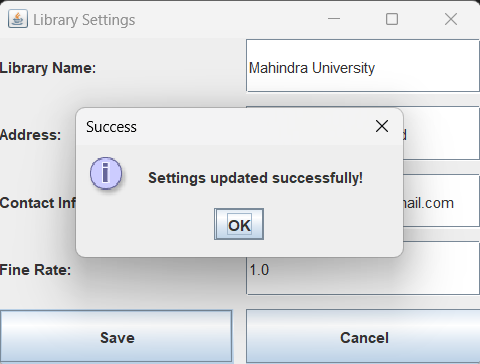
**Reports**



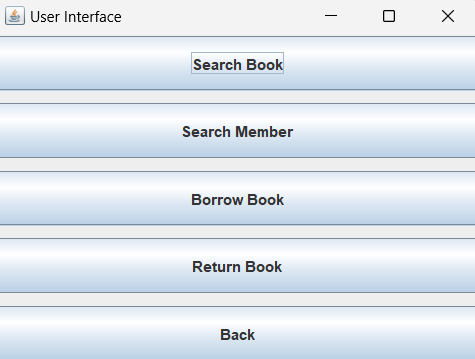




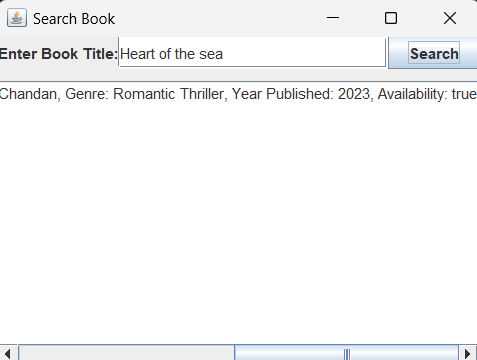
**Settings**



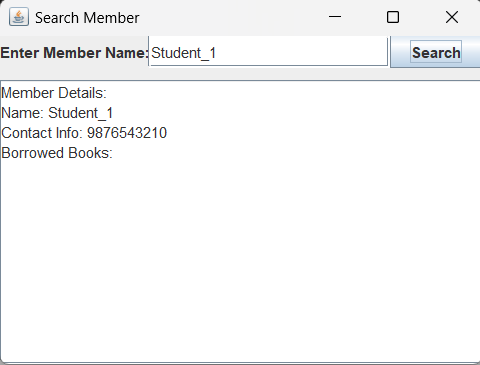
If you **login to User** page



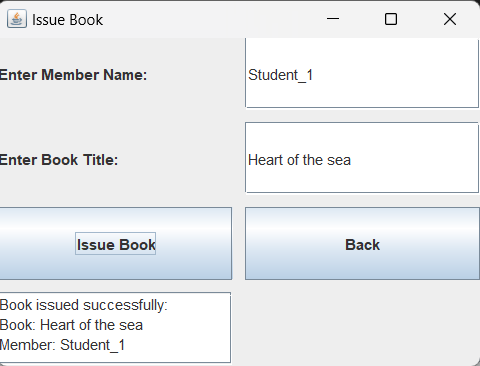
**Search Book**



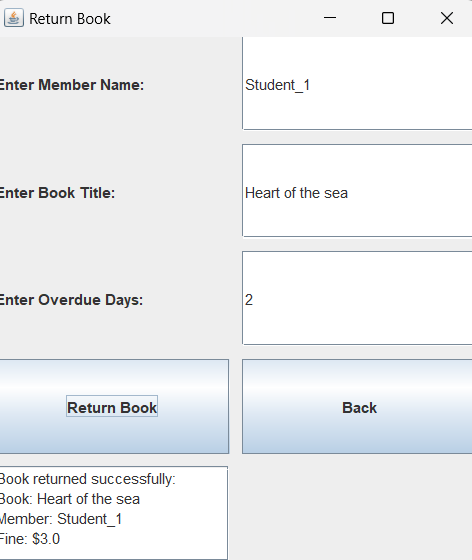
**Search Member**



**Borrow Book**



**Return Book**



**Improvements For this Project**

1. Integration of database for persistent data storage.
2. Advanced search and filter options in the report GUI.
3. Enhanced report export functionality (PDF/Excel).
4. We can use the DIP (Digital Image Processing) for creating a Book Catalog and Book Transaction of books by using Book Cover Detection.

Book Cover Detection- When a new book cover is detected, the system could identify it and automatically pull metadata from a connected database, adding the book to the catalog without manual entry.

Book Transactions- Check-In/Check-Out by Scanning: For transactions, users can scan a book cover at a self-service kiosk or with a mobile app. The system identifies the book and processes it as checked in or checked out.

Automated Logs: Each transaction updates the book’s status in the library’s database, keeping track of borrowed, returned, and reserved books.

User-Specific History: This feature allows users to view their borrowing history, due dates, and penalties, with updates based on scanned transactions.

**Authors  
CH. VAMSHI KRISHNA BABU-SE22UARI038**

**CH. VAMSI KRISHNA-SE22UARI039**

**G. RITHVIK-SE22UARI144**

**K. BHUVAN-SE22UARI212**