LIBRARY MANAGEMENT SYSTEM

INCREMENTAL SDLC MODEL

CSC702 SOFTWARE ENGINEERING PROJECT

SUBMITTED TO: DR. OISHILA BANDOPADHYAY

Roll Number	Name	Work Done
CSE/21030/690	Chandni Jha	Library Management System Development, DFD level 1 and 2, ER Diagram, database schema, Formatting
CSE/21064/724	Purnamrita Bhattacharjee	SRS, SDD documentation, Use Case Diagram, DFD level 0 Diagram
CSE/21082/742	Shubham Kumar Jha	Authentication System Development, Testing, Database Integration

Index

1. Introduction

1.1 Problem statement

Conventional libraries have inefficient cataloguing, resource tracking bottlenecks, and a lack of analytics tools hinder librarians from optimizing collections and improving user experiences. To close the gap, libraries require a modern library management system with an intuitive interface, effective cataloguing, and analytics capabilities

1.2 Proposed Solution

To solve the traditional issue we are building a Web based project of library management system using python and SQL database in which we will be providing user-friendly interface for easy navigation, efficient book adding and listing functionality, seamless book issuance and return policy , flagging student for not following library rules and secure login and access control depending on user role.

1.3 Objective of the Project:

Library Management System project is to design and implement a role-based system that facilitates the efficient management of books and borrowing activities within a library. The system aims to provide:

1. **Role-Based Access**:
   - **Admin** users manage library operations, including adding books, tracking borrowed books, and monitoring students who have overdue books.
   - **Student** users can borrow and return books within predefined constraints (a maximum of three books at a time, only one copy per book).
2. **Book Borrowing Rules**:
   - Each book has a limited number of copies (three).
   - Students can borrow up to three books simultaneously.
   - Students can only borrow one copy of any particular book.
3. **Penalty for Overdue Books**:
   - Students who fail to return books within the set time frame are flagged and restricted from borrowing additional books until they resolve the issue by returning the overdue book and paying any associated fines.

By providing a clear division of roles and rules, the system ensures smooth library operations, enforces borrowing limitations, and encourages timely book returns.

1.4 Scope of the Project:

1.4.1. Functional Scope:

The **functional scope** outlines the specific features and functionalities that the Library Management System will provide to both admin and student users.

1. User Authentication & Role-Based Access:

- **Registration**: Users (both admins and students) can register with unique credentials.
- **Login**: Users can log in with their credentials, and access will be based on their roles (admin or student).
- **Role-based access control**:
    - Admins have full management privileges.
    - Students have limited access to borrowing, returning, and viewing books.

2. Admin Functionalities:

- **Add Book**: Admins can add new books to the library database, specifying the title, author, and availability (each book starts with 3 copies).
- **View Borrowed Books**: Admins can view the current borrowing status, including which books are borrowed and by whom.
- **Show All Books**: Admins can view a list of all books in the library, with details on available copies.
- **Show Flagged Students**: Admins can view a list of students who have been flagged for overdue books and manage their status.

3. Student Functionalities:

- **Borrow Book**: Students can borrow books with the following rules:
    - A student can borrow up to three books at a time.
    - A student can borrow only one copy of a particular book.
    - Borrowing a book updates the number of available copies.
- **Return Book**: Students can return borrowed books. If a student is flagged for overdue books, the flag is removed once the book is returned and any penalties are addressed.
- **Show All Books**: Students can view a list of available books in the library and check the number of available copies for each.

4. Book Borrowing Rules and Flagging System:

- Each book has 3 copies available for borrowing.
- Students are flagged if they do not return books on time. A flagged student cannot borrow more books until the overdue books are returned and fines (if applicable) are paid.

1.4.2. Non-Functional Scope:

The **non-functional scope** includes aspects that ensure the system's performance, usability, and reliability.

1. Performance:

- The system should handle multiple concurrent users (both admins and students) without performance degradation.
- Queries related to book availability, borrowing status, and flagged students should be executed efficiently to maintain a smooth user experience.

2. Scalability:

- The system should be able to scale with an increasing number of books and users (both admins and students).
- It should support future enhancements, such as the addition of new roles (e.g., librarians) or functionalities (e.g., book reservation, notifications).

3. Security:

- User credentials should be securely stored (e.g., encrypted passwords).
- Role-based access control should be enforced strictly to prevent unauthorized access to admin functions by students.
- Sensitive operations, such as borrowing and returning books, should be logged for audit purposes.

4. Usability:

- The system interface should be user-friendly, allowing admins and students to easily navigate through their respective functionalities.
- Error messages and validations should be clear and help users understand what actions are required.

5. Reliability:

- The system should provide consistent availability and avoid downtime, especially during periods of high usage (e.g., when many students are borrowing or returning books).
- Data integrity must be maintained, ensuring that book counts, borrowing records, and student flags are accurate.

6. Maintainability:

- The system should be easy to maintain, with clean code and a modular structure that allows future updates and fixes without significant effort.
- Database backup and recovery mechanisms should be in place to prevent data loss.

7. Compliance:

- The system should comply with privacy regulations regarding the storage and handling of user data, such as names, credentials, and borrowing history.

2. SDLC Model Overview

The **Incremental SDLC model** is used for this Library Management System project. Here's how it applies:

2. 1. Incremental Model Overview

- This model involves breaking down the project into smaller, manageable modules or "increments." Each increment adds specific features or functionality, and each one is tested and validated before moving on to the next.

- In each increment, the code is designed, implemented, and tested, allowing for faster feedback and adjustments based on any user input or issues identified in previous increments.

2.2. Application to the Library Management System Project

Increment 1: User Registration and Authentication Objective: Establish a role-based system where users (students and admins) can register and authenticate. Tasks: Set up the SQL database for user authentication. Implement role-based registration (admin vs. student). Implement secure login and registration functionality. Testing: Ensure that registration, login, and role-based access (admin vs. student) work correctly before moving to the next increment.

Increment 2: Book Management (Admin) Objective: Allow the admin to manage the library inventory (add books, view available books). Tasks: Create a SQL database for storing book information. Implement a feature where admins can add new books and view all available books in the library. Ensure each book has 3 copies available by default in the system. Testing: Test the admin functionality for book addition and retrieval of book data.

Increment 3: Borrowing and Returning Books (Student) Objective: Add functionality for students to borrow and return books. Tasks: Implement borrowing functionality for students, ensuring they can borrow up to 3 books in total. Ensure that each student can borrow only one copy of each book. Track the number of available copies in the library when a book is borrowed or returned. Testing: Test the borrowing and returning features, making sure limits (3 books max per student, 1 copy per book) are enforced. Check if available book counts are updated correctly.

Increment 4: Late Return and Flagging System Objective: Implement a system for tracking late returns and flagging students who fail to return books on time. Tasks: Extend the student module to keep track of borrowing dates and due dates. Automatically flag students who fail to return a book by the due date. Prevent flagged students from borrowing more books until they return the overdue book or clear their flag. Testing: Ensure that overdue books are tracked properly, flagged students are restricted from borrowing, and the flag is removed after book return.

Increment 5: View Borrowed Books and Flagged Students (Admin) Objective: Allow the admin to view which books are currently borrowed and see a list of flagged students. Tasks: Implement a feature for the admin to view all borrowed books and the details of students who borrowed them. Add functionality to show flagged students (those with overdue books). Testing: Verify that the admin can see the correct details of borrowed books and flagged students.

Increment 6: Final Testing and Optimization Objective: Conduct full end-to-end testing and ensure the system meets all functional and non-functional requirements. Tasks: Conduct thorough testing of all functionalities (admin, student roles, book management, borrowing/returning, flagging). Optimize performance (e.g., database queries, user interface) and ensure the system works smoothly under different use cases. Testing: Test the system as a whole to ensure that all increments work together seamlessly. Perform user acceptance testing (UAT).

2. 3. Benefits of Using the Incremental Model:

- Gradual progress: Each phase adds a key part of the system, making it easier to track progress and make adjustments if needed.
- Early feedback: With each completed increment, you can gather feedback from stakeholders (e.g., users, testers) and make improvements early in the development process.
- Flexibility: The model allows for adding new requirements (e.g., new features for the admin or students) at any increment without disrupting the core functionalities.
- Risk reduction: Each module (e.g., user registration, book management, borrowing/returning) is built, tested, and validated independently, reducing the risk of major issues later.
- Early usability: Basic features (like user registration and book management) can be deployed early, allowing partial system usage while other features (like flagging and penalties) are still under development.

3. SRS

Functional Requirements

Admin Features

Add Books

- Admins can add new books by specifying:
    - Book ID
    - Title
    - Author
    - Genre
    - Number of copies available=3

Once added, the book's availability is displayed to students.

View Borrowed Books

Admins can view:

- All borrowed books
- Student details (name, borrow date, return date , return status)

Show All Books

The admin can view the complete list of books, along with the number of available copies.

View Flagged Students

Admins can see the list of flagged students who are restricted from borrowing due to overdue books.

Student Features

Borrow Books

- Students can borrow up to three different books at a time.
- Students can borrow only one copy of a specific book.
- A student can borrow a book by selecting the book ID and entering their student ID.

Conditions:

- Borrowing is allowed only if copies are available.
- Students flagged for late returns cannot borrow books.

Return Books

Students can return books they have borrowed.

- The system maintains the borrow and return dates.
- If a student does not return a book within one day, they are flagged and restricted from further borrowing until cleared by the admin.
- Upon return, the database updates the book's return status and adjusts the available copy count.

Show All Books

Students can view the complete list of books in the library, including genres and the number of available copies.

Common Features

User Type Selection

Users select their role (Admin or Student) upon login to access their specific features.

Data Persistence

The system ensures data integrity and continuity for:

- Book inventory
- Borrowed/returned book records
- Flagged students

Flagging System

- Automatic flagging: Students who do not return books within one day are automatically flagged.
- Restriction: Flagged students cannot borrow any more books until the admin clears their flag status.

Non-functional Requirements

Performance

- The system should be able to handle up to 100 concurrent users (students and admins) without lag or downtime.

Security

- Passwords are encrypted for both admin and student users.
- Role-based access controls are enforced to protect admin features.

Usability

- The system should have a simple, intuitive interface for both students and admins to navigate with minimal training.

Database Management

The system uses a relational database to store:

- User data (name, password, role)
- Book inventory (book ID, title, genre, available copies, total copies, author)
- Borrowed Books (book id, student id, borrow date, return date, return status)
- Flagged students (Student ID)

4. SDD

System Architecture

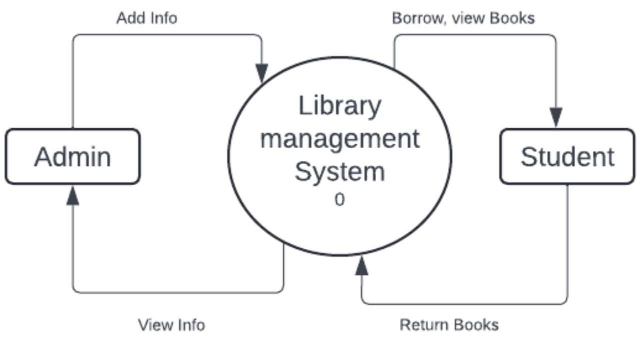
Architecture Overview

The system follows a modular 3-tier architecture, consisting of:

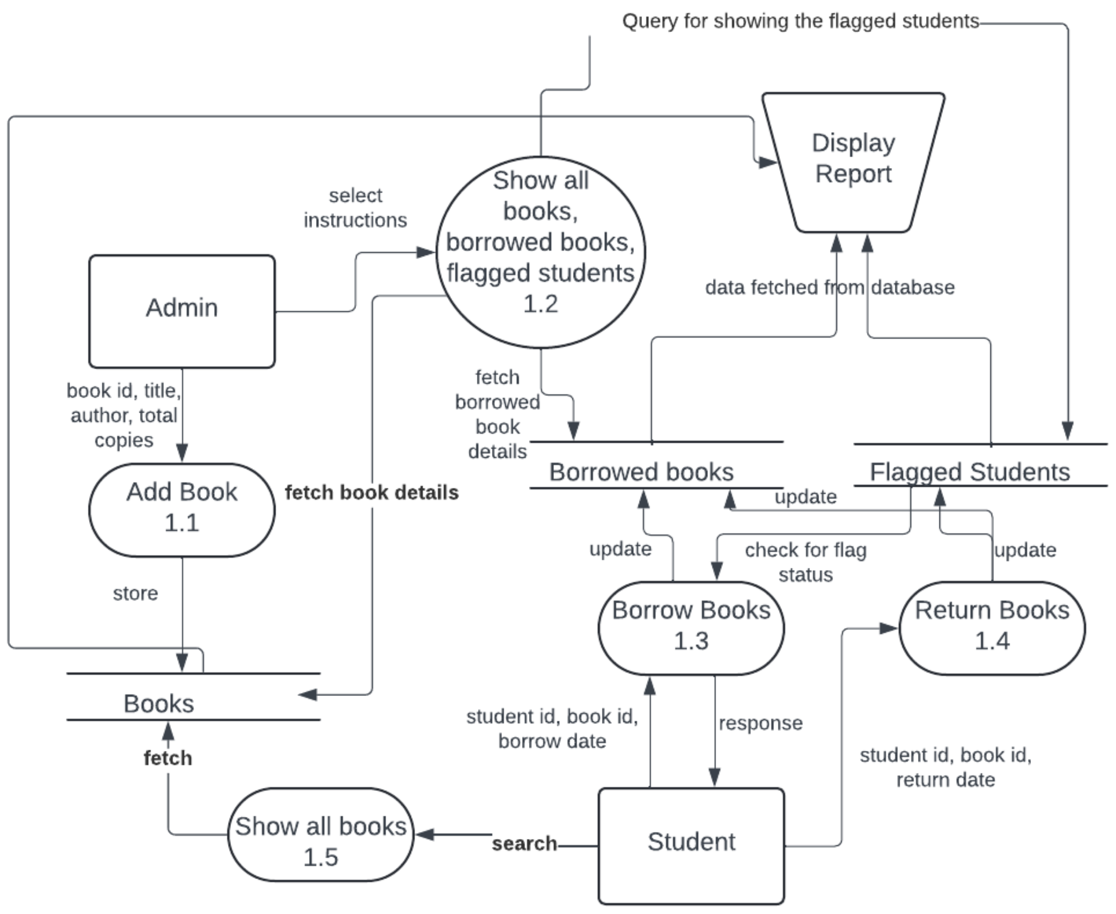
- Presentation Layer (UI): Built using Streamlit, it provides separate interfaces for students and admins.
- Business Logic Layer: Contains the core functionalities such as borrowing/returning books, flagging students, and managing book inventory.
- Data Layer: Interacts with a SQLite database to handle persistent storage of books, borrowed books, flagged students, and users.

Data Flow Diagram

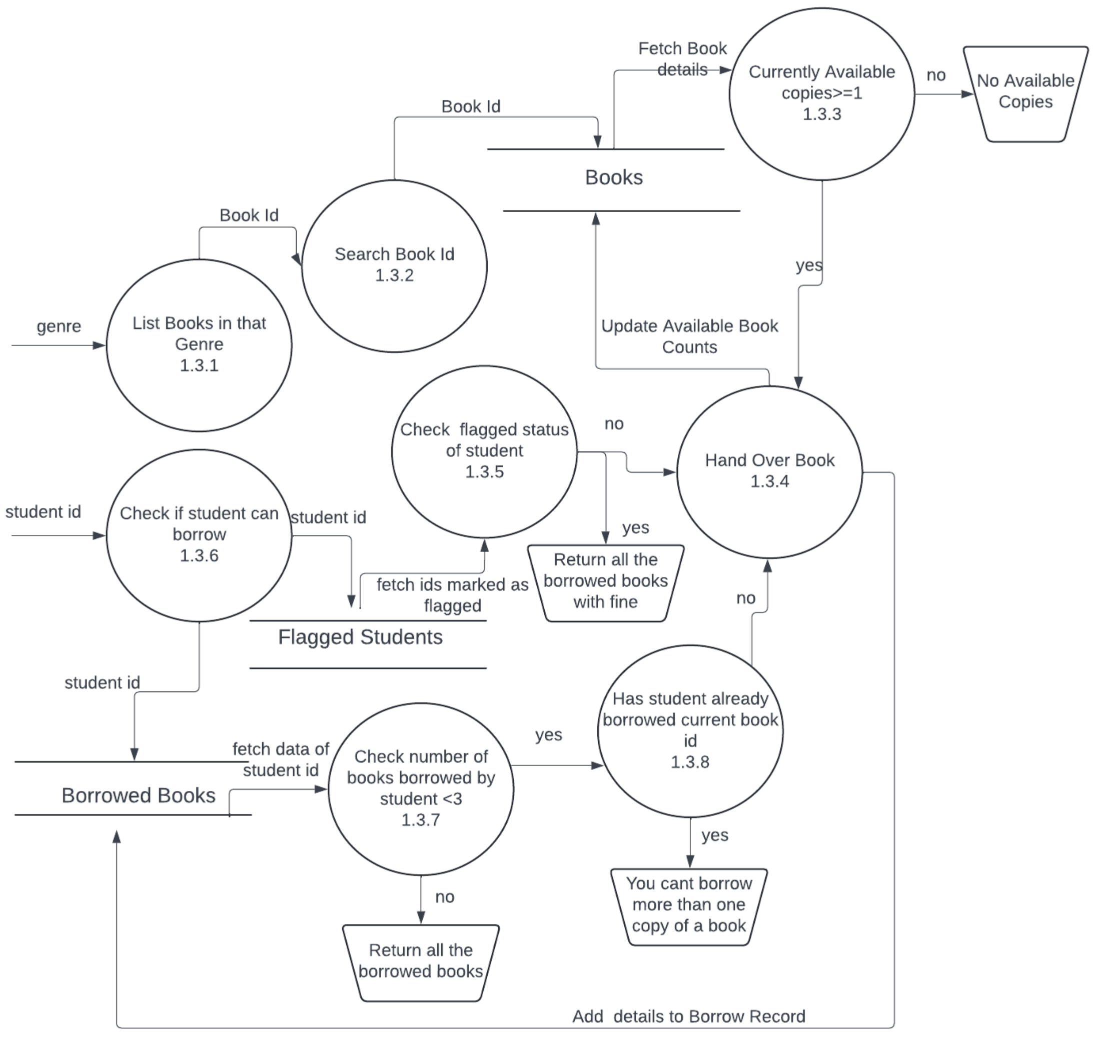
DFD 0 Level:



DFD 1 Level:


Query for showing the flagged students
Display Report
select instructions
Show all books, borrowed books, flagged students 1.2
Admin
data fetched from database
fetch borrowed book details
book id, title, author, total copies
Borrowed books
Flagged Students
Add Book 1.1
fetch book details
update
update
check for flag status
update
store
Borrow Books 1.3
Return Books 1.4
Books
student id, book id, borrow date
response
fetch
student id, book id, return date
Show all books 1.5
search
Student

DFD 2 Level:


Fetch Book details
Currently Available copies>=1 1.3.3
no
No Available Copies
Book Id
Books
Book Id
Search Book Id 1.3.2
yes
genre
List Books in that Genre 1.3.1
Update Available Book Counts
Check flagged status of student 1.3.5
no
Hand Over Book 1.3.4
student id
Check if student can borrow 1.3.6
student id
yes
Return all the borrowed books with fine
fetch ids marked as flagged
no
Flagged Students
student id
Has student already borrowed current book id 1.3.8
fetch data of student id
yes
Borrowed Books
Check number of books borrowed by student <3 1.3.7
yes
no
You cant borrow more than one copy of a book
Return all the borrowed books
Add details to Borrow Record

Module Design

The LMS is divided into three major modules: Admin, Student, and Authentication, with each handling a specific set of functionalities.

Admin Module

The Admin Module enables administrators to manage the library, including adding new books, viewing borrowed books, and handling flagged students.

Components:

- BookManager: Manages adding, displaying, and updating book inventories.
- BorrowManager: Manages viewing borrowed books and monitoring flagged students.

Key Methods:

- add_book(book_id, title, author, genre, available_copies, total_copies): Adds a new book to the system.
- get_all_books(): Retrieves the list of all books in the library.
- get_borrowed_books(): Retrieves all borrowed books with associated student details.
- get_flagged_students(): Retrieves the list of students flagged for overdue book returns.

Database Tables:

- books: Stores book details (ID, title, author, genre, available copies, total copies).
- borrowed_books: Tracks borrowed books with the student's ID and borrow/return dates.
- flagged_students: Stores a list of flagged students due to late returns.

Student Module

The Student Module allows students to interact with the library by borrowing and returning books, as well as viewing the available inventory.

Components:

- BorrowManager: Manages borrowing books.
- ReturnManager: Manages returning books and checks for overdue returns.

Key Methods:

- borrow_book(student_id, book_id): Allows a student to borrow a book, ensuring they have not exceeded the borrowing limit and that the book is available.
- return_book(student_id, book_id): Returns a borrowed book, updating the database and flagging the student if the return is overdue.
- get_available_books_by_genre(genre): Retrieves available books filtered by genre.
- get_borrowed_books_by_student(student_id): Retrieves books borrowed by a specific student.

Authentication Module

The Authentication Module ensures that users can log in and access their respective interfaces. Admins and students have different privileges.

Components:

- AuthManager: Manages user login, registration, and session tracking.

Key Methods:

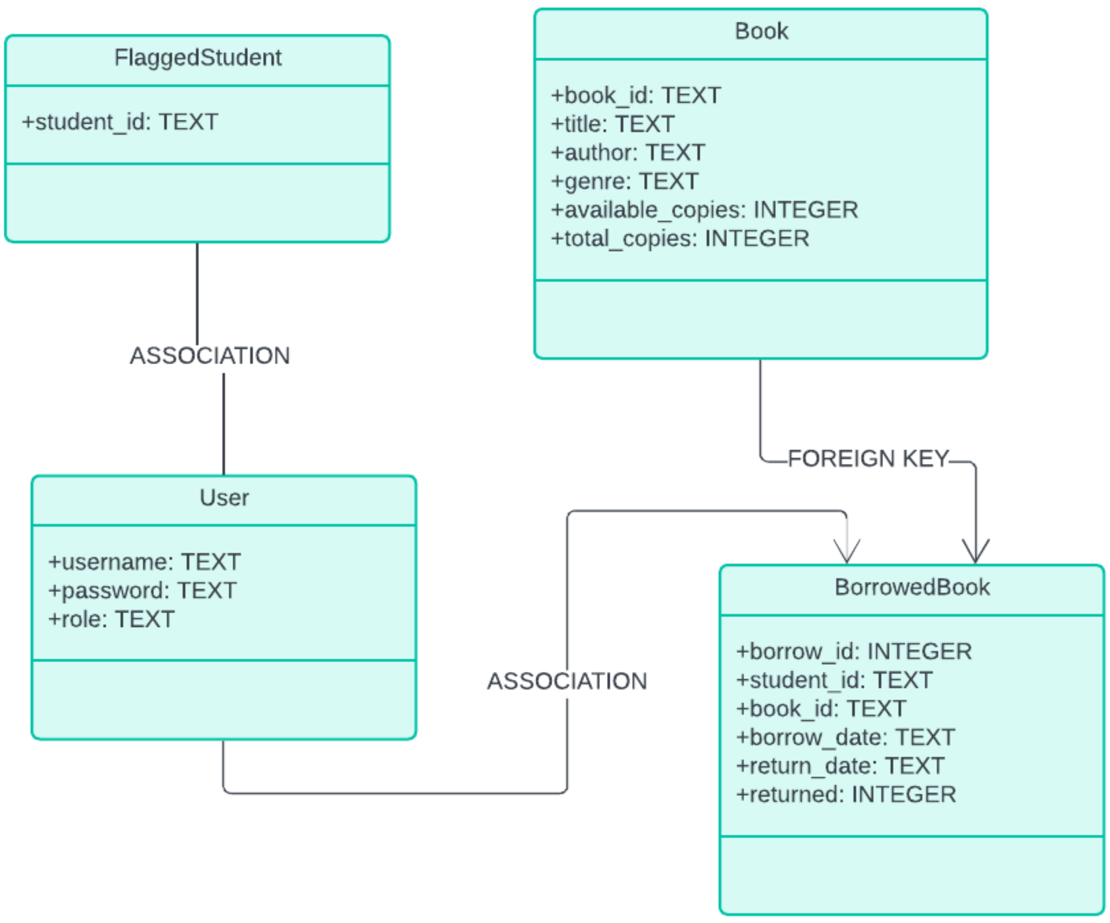
- login (username, password): Authenticates the user and loads the correct interface.
- save_user(username, password, role): Registers new users in the system.
- user_exists(username): Checks if a user is already registered.

Data Design

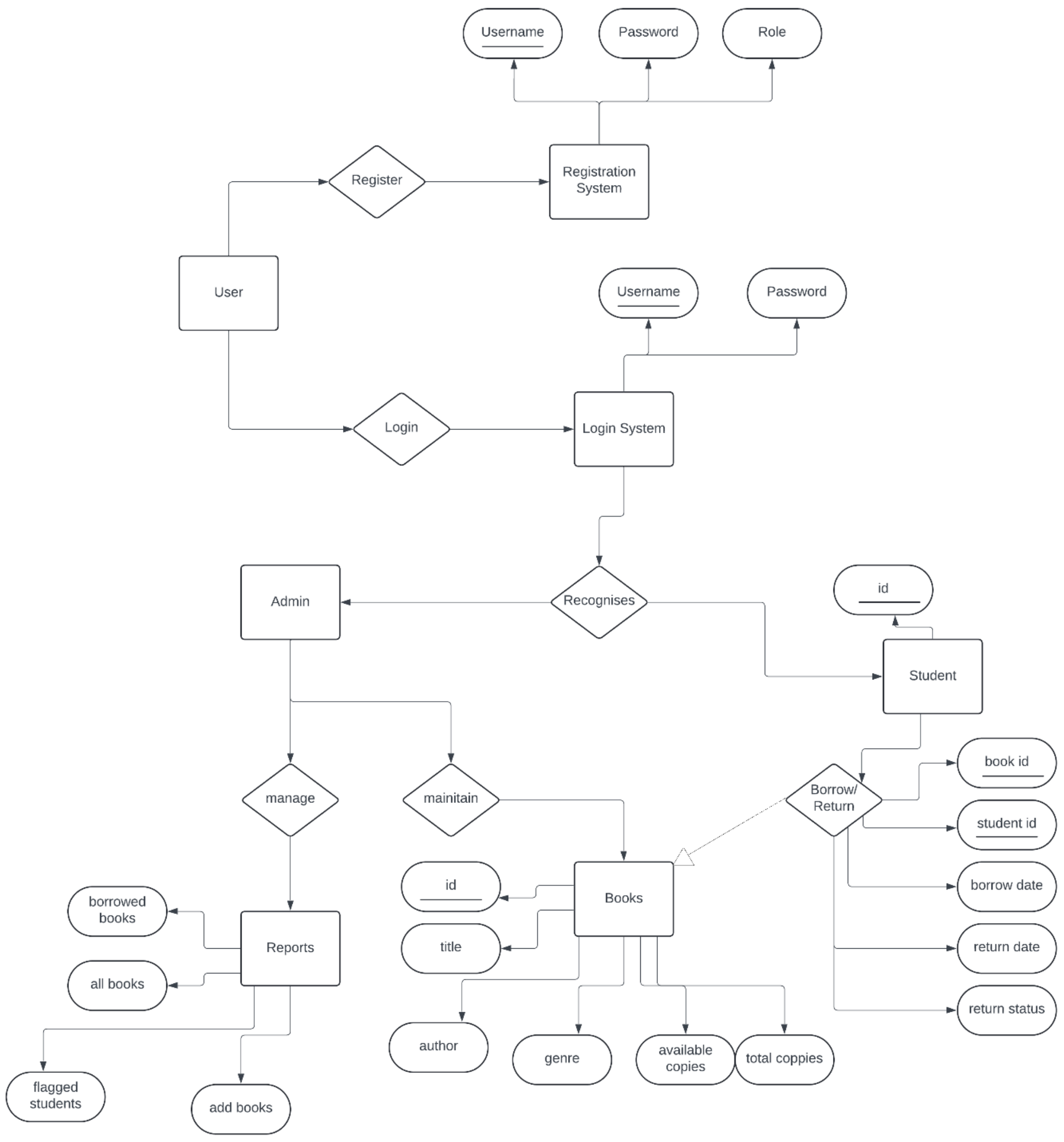
Database Schema

The LMS uses a SQLite database to store data for books, borrowing records, flagged students, and users.

Database Tables:



ER Model:



Component Interaction

Borrowing a Book

- A student selects a genre and chooses a book to borrow.
- The system verifies that the student hasn't exceeded the borrowing limit (3 books) and that copies of the book are available.
- The selected book is then marked as borrowed, with the available copies reduced in the books table and the borrowing record logged in the borrowed_books table.

Returning a Book

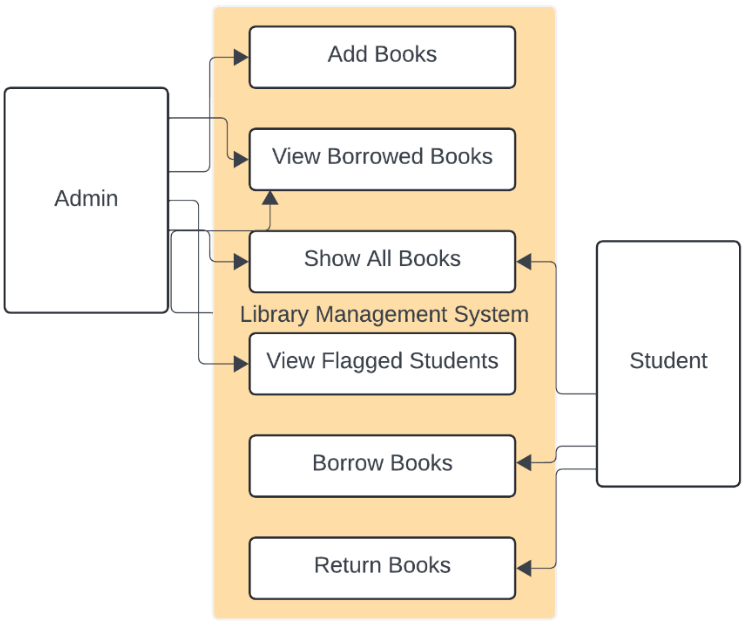
- A student selects a borrowed book to return.

- The system checks if the return date is overdue (more than one day).
- If overdue, the student is flagged in the flagged_students table.
- The borrowed_books table is updated to reflect the return, and the available copies for the book are increased in the books table.

Admin Actions

- Admins can add new books to the system, manage the book inventory, view all borrowed books, and handle flagged students.
- When a new book is added, it is stored in the books table.
- Admins can also view and clear the flagged status of students who have returned overdue books.

Use Case Diagram



User Interface Design

Key Screens:

- Login/Register Screen: Users can log in or register for access.
- Admin Dashboard: Admins can add books, view all books, view borrowed books, and manage flagged students.
- Student Dashboard: Students can borrow, return, and view available books.

Key UI Components:

- Login Screen: Username, password input, and role selection (admin or student).
- Admin Panel: Allows admins to add books, view borrowed books, manage flagged students, and display all books.
- Student Panel: Allows students to view books, borrow, and return books.

Security Design

Authentication

- The system uses role-based access control to manage admin and student privileges.
- Passwords are securely stored in the user's table.
- Each user's role determines their access to specific functionalities (admin vs. student).

Data Validation

- All inputs (e.g., student IDs, book IDs) are validated to ensure data integrity and prevent unauthorized access.

Assumptions and Dependencies

- Streamlit is used for the front-end interface, relying on Python for the business logic.
- SQLite is used for database management, requiring no external server setup.
- Pandas is used for data display and manipulation within the application.

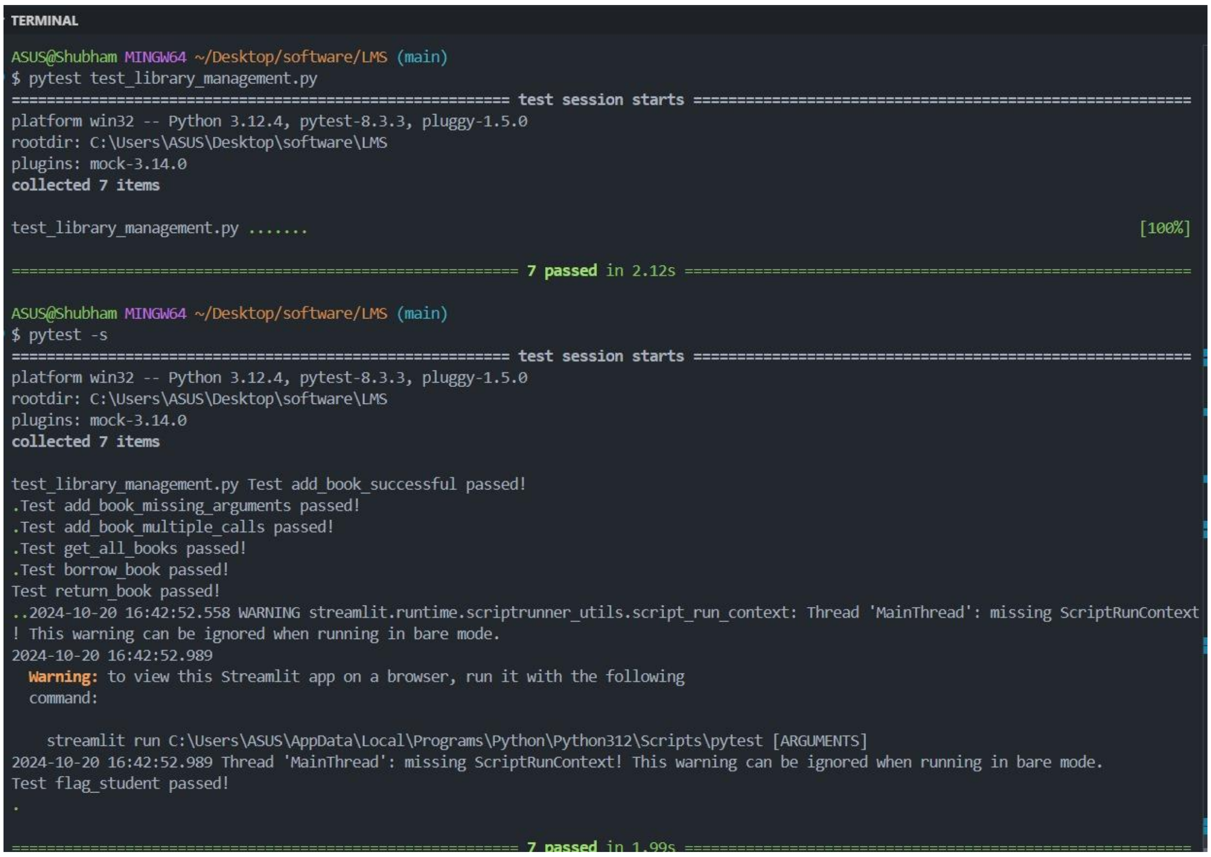
5. CODE

Code Repository

6. TESTING

Test Code

Output:



```
TERMINAL

ASUS@Shubham MINGW64 ~/Desktop/software/LMS (main)
$ pytest test_library_management.py
========================================================= test session starts =========================================================
platform win32 -- Python 3.12.4, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\ASUS\Desktop\software\LMS
plugins: mock-3.14.0
collected 7 items

test_library_management.py .......                                                                                               [100%]

========================================================== 7 passed in 2.12s ==========================================================

ASUS@Shubham MINGW64 ~/Desktop/software/LMS (main)
$ pytest -s
========================================================= test session starts =========================================================
platform win32 -- Python 3.12.4, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\ASUS\Desktop\software\LMS
plugins: mock-3.14.0
collected 7 items

test_library_management.py Test add_book_successful passed!
.Test add_book_missing_arguments passed!
.Test add_book_multiple_calls passed!
.Test get_all_books passed!
.Test borrow_book passed!
Test return_book passed!
..2024-10-20 16:42:52.558 WARNING streamlit.runtime.scriptrunner_utils.script_run_context: Thread 'MainThread': missing ScriptRunContext
! This warning can be ignored when running in bare mode.
2024-10-20 16:42:52.989
  Warning: to view this Streamlit app on a browser, run it with the following
  command:

    streamlit run C:\Users\ASUS\AppData\Local\Programs\Python\Python312\Scripts\pytest [ARGUMENTS]
2024-10-20 16:42:52.989 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
Test flag_student passed!
.

========================================================== 7 passed in 1.99s ==========================================================
```