Library Management System - Modeling Documentation

1. Project Overview

The Library Management System (LMS) is designed to automate and manage daily library activities.

It supports book inventory management, user handling (admin, librarian, students), book issuing/returning,

fine calculation, and report generation to streamline library workflows.

2. Functional Requirements

- Add, update, delete books

- Register new users (students, librarians)

- Login system with roles

- Issue and return books

- Calculate and display fines

- View and search book catalog

- Generate usage reports

3. Non-Functional Requirements

- Easy-to-use interface

- Role-based access control

- Fast search capability

- Secure login and session handling

- Backup and recovery system

4. Use Case Diagram

Actors:

- Admin

- Librarian

- Student

Main Use Cases:

- Login

- Manage Books

- Issue/Return Books

- Register Users

- Search Catalog

- View Fines and History

(Mermaid code can be used to visualize the diagram in a renderer.)

5. Class Diagram

User <|-- Librarian

User <|-- Student

Classes:

User (userID, name, role, email)

Book (bookID, title, author, available)

Transaction (transactionID, bookID, studentID, issueDate, returnDate)

Fine (fineID, transactionID, amount, isPaid)

6. Sequence Diagram: Book Issuing

Student -> System : Login

Student -> System : Search Book

System -> DB : Fetch Book Info

Student -> System : Request Issue

System -> DB : Create Transaction

System -> Student : Confirm Issue & Due Date

7. Activity Diagram: Return Book & Fine

Start -> Login -> Return Book Request -> Is Book Late?

Yes -> Calculate Fine -> Update Inventory -> Update Transaction Record -> End

No -> Update Inventory -> Update Transaction Record -> End

8. ER Diagram

Entities:

- Users (user_id, name, role, email)

- Books (book_id, title, author, status)

- Transactions (transaction_id, book_id, user_id, issue_date, return_date)

- Fines (fine_id, transaction_id, amount, status)

Relationships:

- A user can have many transactions

- A transaction belongs to one book and one user

- A transaction may have one fine

9. System Architecture

Frontend: HTML, CSS, JavaScript, Bootstrap

Backend: PHP or Python (Flask/Django)

Database: MySQL or SQLite

Optional Desktop App: Tkinter

Design Pattern: MVC