create a library management with book recommendation system .System should have admin panel and student panel. Admin should have privilege to add students

**Here’s a comprehensive design for a Library Management System with a Book Recommendation System that includes both an Admin Panel and a Student Panel.**

**System Overview**

**This system integrates traditional library management features (e.g., book borrowing, user management) with a personalized book recommendation engine. It has two main user roles:**

* **Admin: Manages books, students, and library operations.**
* **Student: Searches for books, borrows/returns books, and receives personalized recommendations.**

**Key Features**

**Admin Panel**

1. **Student Management**
   * **Add, edit, or delete student records.**
   * **View student borrowing history and overdue fines.**
2. **Book Management**
   * **Add, update, or remove books (ISBN, title, author, genre, etc.).**
   * **Track book availability and borrowing statistics.**
3. **Reports and Analytics**
   * **View popular books and genres.**
   * **Monitor overdue books and fines.**
   * **Analyze library usage trends.**

**Student Panel**

1. **Book Search**
   * **Search by title, author, genre, or ISBN.**
   * **Filter books by availability or rating.**
2. **Borrow/Return Books**
   * **Borrow available books and track due dates.**
   * **Return borrowed books with fine calculations for overdue returns.**
3. **Personalized Recommendations**
   * **Content-based recommendations using book metadata (e.g., genres, authors).**
   * **Collaborative filtering based on similar users’ preferences.**
4. **Borrowing History**
   * **View past borrowed books and ratings.**

**Database Schema**

**1. Users Table**

**sql**

**CREATE TABLE Users (**

**user\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**name VARCHAR(100),**

**email VARCHAR(100) UNIQUE,**

**password\_hash VARCHAR(255),**

**role ENUM('admin', 'student'),**

**created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP**

**);**

**2. Books Table**

**sql**

**CREATE TABLE Books (**

**book\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**title VARCHAR(255),**

**author VARCHAR(255),**

**genre VARCHAR(100),**

**isbn VARCHAR(13) UNIQUE,**

**num\_copies INT DEFAULT 1,**

**avg\_rating FLOAT DEFAULT 0,**

**created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP**

**);**

**3. BorrowedBooks Table**

**sql**

**CREATE TABLE BorrowedBooks (**

**borrow\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**user\_id INT,**

**book\_id INT,**

**borrow\_date DATE,**

**due\_date DATE,**

**return\_date DATE NULL,**

**fine FLOAT DEFAULT 0,**

**FOREIGN KEY (user\_id) REFERENCES Users(user\_id),**

**FOREIGN KEY (book\_id) REFERENCES Books(book\_id)**

**);**

**4. Ratings Table**

**sql**

**CREATE TABLE Ratings (**

**rating\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**user\_id INT,**

**book\_id INT,**

**rating INT CHECK (rating BETWEEN 1 AND 5),**

**review TEXT NULL,**

**FOREIGN KEY (user\_id) REFERENCES Users(user\_id),**

**FOREIGN KEY (book\_id) REFERENCES Books(book\_id)**

**);**

**Recommendation System Integration**

**The recommendation system can be integrated into the library management system using the dataset provided in the attached file or other datasets like Goodreads.**

**Implementation Steps:**

1. **Use the dataset to train a recommendation model (e.g., collaborative filtering using user-book ratings).**
2. **Store the model predictions in a database table for quick access:**

**sql**

**CREATE TABLE Recommendations (**

**user\_id INT,**

**book\_id INT,**

**predicted\_rating FLOAT,**

**FOREIGN KEY (user\_id) REFERENCES Users(user\_id),**

**FOREIGN KEY (book\_id) REFERENCES Books(book\_id)**

**);**

1. **Generate recommendations dynamically based on:**
   * **User preferences (e.g., genres of previously borrowed books).**
   * **Similar users' ratings (collaborative filtering).**

**Backend API Endpoints**

**Admin APIs**

* **Add Student: POST /api/admin/students**
* **Add Book: POST /api/admin/books**
* **View Reports: GET /api/admin/reports**

**Student APIs**

* **Search Books: GET /api/books?query=...**
* **Borrow Book: POST /api/students/borrow**
* **Return Book: POST /api/students/return**
* **Get Recommendations: GET /api/students/recommendations**

**Frontend Design**

**Admin Panel**

* **Dashboard with analytics (e.g., popular books, overdue fines).**
* **Forms for adding/editing students and books.**
* **Tables for viewing all students/books with search functionality.**

**Student Panel**

* **Search bar for finding books.**
* **Borrowed books section with due dates.**
* **Recommendations displayed as a carousel or grid.**