Console-Based Library Management System (LMS) - Multi-Storage Data Persistence Integration

Assignment Overview:

This assignment is a continuation of your previous work on the console-based Library Management System (LMS). You will extend the functionality to include data persistence through two different storage mechanisms: File-based storage and SQL Database. In addition to adding multi-storage support, you will also implement fixes from the first assignment, improve data management, and add more functionalities related to search, sorting, and user fines.

Assignment Requirements:

1. Multi-Storage Data Persistence Integration:

Objective: The system must support data persistence through two different storage options:
1. File-based storage (e.g., storing data in `.txt` or `.csv` files).
2. SQL Database (e.g., MySQL or PostgreSQL).
User Selection of Storage Method: When the application starts, users should be presented with a menu to select one of the storage methods. The selected method should then be used for all subsequent operations (e.g., adding books, retrieving data).
Consistency Across Storage Methods: Ensure that changes made to the data are correctly reflected based on the chosen storage method during runtime.

2. New Functionalities:

• Search Functionality: Implement the ability to search for books and users by various attributes such as `Book ID`, `Title`, `Author`, `ISBN`, `User ID`, or `User Name`.
• Sorting Functionality: Implement sorting functionality for listing books and users based on different criteria (e.g., alphabetical order, loan status, return date).
• Fine Calculation for Late Returns: Implement a feature to calculate fines for late returns based on a flat daily fee. Users should be notified if they have any unpaid fines.
• Loan Extension Requests: Allow users to request loan extensions for certain books (e.g., Textbooks), while ensuring only one extension is allowed.

3. Fix Issues from Assignment 1:

• Input Validation: Ensure that no two books or users have the same `Book ID` or `User ID`. Validate all inputs to prevent incorrect data entry (e.g., invalid data types, missing fields).
• Data Structure Improvements: Properly structure and implement additional attributes related to damage and late return penalties for books, ensuring they follow the correct data structure. Correct usage of static and final variables where applicable.

• Deletion Logic: Prevent deletion of users with active loans. Ensure that deleting a book is only possible if the book is not currently loaned.

4. Data Management Requirements:

• File-based Storage: Implement efficient techniques to store and retrieve data using a structured file format (e.g., `.txt`, `.csv`).
• SQL Database: Establish a connection to an SQL database (e.g., MySQL, PostgreSQL) and perform the same CRUD operations (Create, Read, Update, Delete) as required in File-based storage.

5. CRUD Operations:

• Books: Implement functionality to add, remove, update, and display book details. Ensure that all operations update the data across the selected storage method.
• Users: Implement functionality to add, remove, update, and display user details across both storage options (File and SQL).
• Transactions: Implement functionality to log and display loan transactions, including rental cost, damage details, and penalties. These transaction records should be maintained across all storage methods.

6. Error Handling:

Implement proper error handling for database connectivity issues or invalid operations (e.g., trying to delete a book that is currently loaned). Ensure smooth handling of CRUD operations across both storage options, including error handling during read/write operations.

Deliverables:

1. Java Code: A fully implemented Java program that supports data persistence across File-based and SQL databases.
2. Database Schema: Provide the schema for the SQL database, including tables, fields, and relationships for books, users, and transactions.
3. Documentation: Submit a brief report describing the storage method selection process and how the program supports each of the two storage mechanisms.