**"Library Management System"**

**📖 Assignment Description**

**🔍 Assignment Details and Tasks**

**1. Java Fundamentals**

* Define classes:
  + Book with fields: bookId, title, author, isIssued
  + Member with fields: memberId, name, email
  + LendingRecord with fields: recordId, book, member, issueDate, dueDate, returnDate
* Use enums for statuses if needed (e.g., BookStatus).
* Implement methods for:
  + Adding/removing books and members.
  + Issuing books to members (creating a LendingRecord).
  + Returning books and updating records.

**2. Exception Handling and Generics**

* Create a generic class Repository<T> that handles collections of Book, Member, and LendingRecord.
* Define custom exceptions such as:
  + BookNotAvailableException when trying to issue an already issued book.
  + MemberNotFoundException when searching for a member.
  + OverdueBookException if trying to issue books to members with overdue books.
* Use these exceptions to make your code robust and user-friendly.

**3. Java Collections**

* Use HashMap<String, Book> and HashMap<String, Member> to store books and members by their IDs.
* Use ArrayList<LendingRecord> to keep track of all lending activities.
* Implement searching capabilities:
  + Find books by author or title.
  + Find members by name or email.
* Implement sorting of books by title or author using Comparable or Comparator.

**4. Java Multithreading**

* Create a background thread called OverdueMonitor that runs periodically (e.g., every 1 minute).
* It scans LendingRecords for overdue books (dueDate < current date and returnDate is null).
* Logs overdue books and member info to the console.
* Optionally, sends notifications or flags members with overdue books.

**5. Java Advanced Concepts**

* Implement Comparable<Book> to allow sorting books alphabetically by title.
* Use Java Streams and lambda expressions to:
  + Filter available books.
  + List members who currently have borrowed books.
  + Generate reports of all overdue books.
* Use inner classes if helpful (e.g., utility classes inside Repository).

**📂 Deliverables**

* Full source code organized into packages (model, repository, exception, service).
* A README file including:
  + Setup instructions.
  + Features implemented.
  + Description of how each Java concept was used.
* Sample console output demonstrating the system features.
* Optional: Unit tests for core classes.

**💡 Example Scenario**

1. Librarian adds books and registers members.
2. Member borrows a book; the system creates a lending record.
3. If the member has overdue books, issuing new books is denied with an exception.
4. Background thread detects overdue books and prints notifications.
5. Librarian views sorted book list or searches books by author.
6. Member returns book, system updates records accordingly.