 **Book Library System**

**1. Overview**

The **Library Management System** is a Java-based application designed to facilitate the management of a library's key entities: users, books, authors, borrowing records, librarians, and library branches. It supports operations such as adding, editing, deleting, and querying these entities. The system ensures data consistency through well-defined relationships and business logic.

**2. Project Structure and Packages**

The project follows a layered architecture and is organized into the following main packages:

**2.1 Models (Domain Entities)**

* Contains all domain entities that represent the core objects in the system.
* Examples of classes:
  + User
  + Book
  + Author
  + BorrowRecord
  + LibraryBranch
  + Librarian
* Each class contains fields mapping to database columns, constructors, getters/setters, and possibly basic validation or utility methods.

**2.2 Repositories**

* Contains interfaces that extend JpaRepository or similar, used for data access and CRUD operations.
* Typical repositories:
  + UserRepository
  + BookRepository
  + AuthorRepository
  + BorrowRecordRepository
  + LibraryBranchRepository
  + LibrarianRepository
* Responsible for querying the database, custom queries, and paging/sorting support.

**2.3 Servicies**

* Contains service interfaces defining business logic operations.
* Examples:
  + UserService
  + BookService
  + AuthorService
  + BorrowRecordService
  + LibraryBranchService
  + LibrarianService
* Methods here implement validation, transactional logic, and calls to repositories.
* **Service.impl**
* Contains implementations of the service interfaces.
* Implements the business rules, manages transactions, and delegates data operations to repositories.

**2.4 DTO’S**

* Contains Data Transfer Objects.
* Used to transfer data between layers, especially between client and server in web applications.
* Helps to decouple internal models from external representation.
* Typical DTOs:
  + RegisterDTO
  + AddBookDTO
  + AddAuthorDTO
  + BorrowRecordDTO
  + LibraryBranchDTO
  + AddLibrarianDTO

**2.5 Exception**

* Contains custom exception classes.
* Used for handling specific error cases, such as EntityNotFoundException, InvalidOperationException, etc.
* Enhances error reporting and handling across the app.

**3. Summary of Core Classes and Their Responsibilities**

| **Package** | **Class Example** | **Responsibility** |
| --- | --- | --- |
| model | User | Represents a library user with personal data |
| model | Book | Represents a book with metadata and availability |
| model | Author | Represents an author of books |
| model | BorrowRecord | Records borrowing transactions |
| model | LibraryBranch | Represents a library branch location |
| model | Librarian | Represents a library staff member |
| repository | UserRepository | Data access for User entities |
| service | UserService | Business logic for user operations |
| service.impl | UserServiceImpl | Concrete implementation of UserService |
| dto | UserDto | Data transfer representation of User |
| exception | EntityNotFoundException | Handles errors when entities are missing |

**4. Entity Relationships Recap**

* **User – BorrowRecord:** One-to-many, a user can have many borrow records.
* **Book – BorrowRecord:** One-to-many, a book can have many borrowings.
* **Author – Book:** One-to-many, an author can write many books.
* **LibraryBranch – Book:** One-to-many, books are associated with branches.
* **LibraryBranch – Librarian:** One-to-many, librarians assigned per branch.
* **BorrowRecord – LibraryBranch:** Many-to-one, borrow record belongs to a branch.

**5. Application Functionality**

* **User Management:** Create, read, update, and delete user profiles.
* **Book and Author Management:** Manage books and authors, track book availability.
* **Borrowing Records:** Log borrow and return operations, track due dates and statuses.
* **Library Branch Operations:** Maintain branches with their books and staff.
* **Librarian Management:** Assign librarians to branches and manage their details.