Library Management System Brief Documentation

Models:

User:

- id
- username (String): The username of the user.
- password (String): The user's password (hashed and salted).
- role (Enum): The user's role (Librarian or Patron).

Book:

- id
- title (String): The title of the book.
- author (String): The author of the book.
- ISBN (String): The International Standard Book Number.
- genre (String): The genre of the book.
- is_available (String): Indicates whether the book is available for borrowing.

Patron:

- id
- name (String): The name of the patron.
- contact_info (String): Contact information of the patron.
- membership_status (String): Patron's membership status (e.g., Active or Inactive).

BorrowingTransaction:

- id
- book_id (Book): Foreign key referencing the Book table.
- patron_id (Patron): Foreign key referencing the Patron table.
- borrow_date (Date): The date the book was borrowed.
- due_date (Date): The due date for returning the book.
- return_date (Date): The date the book was returned.
- is_returned (Boolean): Indicates whether the book has been returned.

Services:

Services are responsible for encapsulating the business logic and application-specific functionality of the programme. They act as an intermediary layer between the controllers (which handle HTTP requests) and the repositories (which interact with the database).

Service example:

- ”BookService “-> responsible for handling operations related to books, such as adding, updating, and retrieving book details

Repositories:

Repositories are responsible for data access and database interaction. They provide an abstraction layer over the database, allowing the application to interact with models without dealing directly with database-specific details

Repositories example:

- “UserRepository” -> Provides methods for accessing user data stored in the database, such as finding users by username, role, or ID

Controllers:

Controllers are responsible for handling incoming HTTP requests and managing the flow of data. They bridge the gap between the client and the backend services.

Controller example:

- “UserController “ -> Handles user-related HTTP requests, such as creating user accounts

Scheme structure: