**Library Management System (LMS) Backend**

**Project Report**

***1. Introduction***

The Library Management System (LMS) backend is a robust, secure, and scalable RESTful application designed to automate and manage the core operations of a modern library. Built with Java, Spring Boot, and MySQL, the system supports both administrative and user-facing functionalities, ensuring efficient management of library resources, users, and transactions.

***2. System Architecture***

* Framework: Java with Spring Boot
* Database: MySQL (accessed via Spring Data JPA)
* Security: Spring Security with role-based access control (Librarian/Admin and Customer/Patron)
* API Style: RESTful endpoints for all operations
* Layered Design:
* Domain Layer: Entity classes (Book, Customer, Librarian, BorrowTransaction, Reservation, etc.)
* Repository Layer: JPA repositories for data access
* Service Layer: Business logic for all major operations
* Controller Layer: REST controllers for API endpoints
* Config Layer: Security and CORS configuration

***3. Core Features***

**3.1. Catalog and Inventory Management**

* Book Management: Add, update, delete, and view books with detailed metadata (title, author, subject, etc.).
* Inventory Tracking: Each book tracks total copies, available copies, location (e.g., shelf), and status (AVAILABLE, ON\_LOAN, RESERVED, DAMAGED).
* Admin Control: Only librarians/admins can modify the catalog.

**3.2. User Management**

* Registration & Login: Users can create accounts and log in securely.
* Role-Based Access: Two main roles—CUSTOMER (patron) and LIBRARIAN (admin).
* Profile Management: Stores user details, borrowing history, and reservations.

**3.3. Circulation and Transactions**

* Borrowing: Customers can borrow available books. The system logs each transaction, sets due dates, and updates inventory.
* Returning: When a book is returned, the system records the return date and calculates overdue fines if applicable.
* BorrowTransaction Entity: Tracks every borrow/return, due/return dates, and fines.
* Borrowing History: Each customer’s borrowing history is maintained and accessible.

**3.4. Reservation System**

* Book Reservations: Customers can reserve books that are currently unavailable.
* Reservation Entity: Tracks reservation date, notification status, and fulfillment.
* Notification Support: System can be extended to notify users when reserved books become available.

**3.5. Reporting and Analytics**

* Usage Reports: Statistics on total borrows, borrowing trends, and popular books.
* Inventory Reports: Statistics on total books, available/missing/damaged items.
* Financial Reports: Tracks and reports on fines collected for overdue items.

**3.6. Security and Error Handling**

* Spring Security: All endpoints are protected by role-based access control.
* Password Encryption: User passwords are securely hashed.
* Global Exception Handling: User-friendly error messages for all API errors.

***4. Recent Enhancements***

* Inventory and Book Model Extensions: Added fields to Book: totalCopies, availableCopies, location, status for better inventory control.
* Borrowing and Returning System: Introduced BorrowTransaction entity and service for logging all borrow/return actions and calculating overdue fines.
* Reservation System: Introduced Reservation entity and service for tracking book reservations and notification status.
* User Borrowing History and Reservations: Customer entity now maintains borrowing history and reservations.
* Reporting and Analytics: Added ReportController for usage, inventory, and financial reports.
* Error Handling: Added a global exception handler for robust, user-friendly API responses.

***5. System Strengths***

* Modular and Extensible: Easy to add new features or integrate with external systems (e.g., notification services, payment gateways).
* Secure: Strong authentication and authorization, with encrypted passwords and role-based access.
* Data Integrity: JPA relationships and normalization ensure consistent, reliable data.
* Scalable: Designed to handle growth in users, books, and transactions.
* Maintainable: Clear separation of concerns and best practices in code organization.

***6. Potential Future Enhancements***

* Notification Integration: Email/SMS notifications for reservations and overdue reminders.
* Backup/Restore: Automated or manual database backup and restore features.
* Advanced Analytics: More detailed reporting (e.g., genre trends, peak times).
* Payment Integration: Online payment for fines and fees.
* UI/UX: Integration with a frontend for a complete user experience.

***7. Conclusion***

The LMS backend is a feature-rich, secure, and maintainable system that covers all essential library operations, including catalog management, user management, circulation, reservations, reporting, and security. It is ready for production use and future enhancements, providing a solid foundation for a modern library’s digital infrastructure.