**Library Management System - Business Requirements Document**

**Project Overview**

A Spring Boot web application for managing library operations including book catalog management, user authentication, and rental tracking. The system demonstrates enterprise-level dependency injection patterns and role-based access control.

**Library Model**

This system is designed for specialized library collections where each title represents a unique or limited resource (such as rare books, reference materials, or special collections). The single-copy-per-title constraint allows for focused resource management and demonstrates clear availability tracking without the complexity of inventory quantity management.

**Technical Stack**

* **Framework:** Spring Boot
* **Data Storage:** JSON files (no database)
* **Authentication:** Custom login system with role-based access
* **Testing:** JUnit 5 + Mockito
* **Architecture:** Repository pattern with dependency injection

**User Management**

**User Registration**

* New users can only register with "USER" role
* Registration requires: username, password, email
* No role selection during registration (security best practice)

**User Authentication**

* Standard username/password login
* Session-based authentication
* Role-based access control after login

**Bootstrap Admin Account**

* **Default Admin:** username = "admin", password = "admin123"
* Marked as protected: true in users.json
* Cannot be deleted or demoted
* Ensures system always has at least one admin

**Role-Based Access Control**

**Regular User Capabilities**

* Browse and search books by title or author
* Rent available books (single copy per book)
* View currently rented books
* Return books they have rented
* Edit own account information (username, email, password)
* View own account profile

**Admin Capabilities**

* All user capabilities PLUS:
* Add new books to catalog
* Edit existing book information
* Delete books from catalog
* Import additional books (append to existing catalog)
* Export complete catalog to JSON file
* View all users and their roles
* Edit any user account information (username, email, reset password)
* Promote users from USER to ADMIN
* Demote admins to USER (except protected accounts)
* Delete user accounts (except protected accounts)

**Data Storage Structure**

**File Organization**

Three JSON files for data persistence:

**1. users.json**

[

{

"id": 1,

"username": "admin",

"password": "hashedPassword",

"email": "admin@library.com",

"role": "ADMIN",

"protected": true

},

{

"id": 2,

"username": "john",

"password": "hashedPassword",

"email": "john@email.com",

"role": "USER",

"protected": false

}

]

**2. books.json**

[

{

"id": 1,

"title": "Clean Code",

"author": "Robert Martin",

"genre": "Programming",

"status": "AVAILABLE"

},

{

"id": 2,

"title": "1984",

"author": "George Orwell",

"genre": "Fiction",

"status": "RENTED"

}

]

**3. rentals.json**

[

{

"id": 1,

"userId": 2,

"bookId": 2,

"rentDate": "2025-09-16T10:30:00",

"status": "ACTIVE"

}

]

**Book Management**

**Book Status**

* **AVAILABLE:** Book can be rented
* **RENTED:** Book is currently checked out

**Single Copy Policy**

* One copy per book title/author combination
* Simplifies rental logic and data management
* Focus remains on Spring concepts rather than inventory complexity

**Rental System**

* User can rent available books
* Book status changes to RENTED
* Rental record created with user and date information
* User can return books, reverting status to AVAILABLE

**Import/Export Functionality**

**Export Books**

* Admin can export complete catalog to JSON file
* Includes all book information and current status
* Downloads as timestamped file (e.g., "library\_export\_2025-09-16.json")

**Import Books**

* Admin can import additional books from JSON file
* **Append-only operation** (does not replace existing catalog)
* Duplicate detection: books with identical title AND author are skipped
* Import summary displayed (books added vs skipped)

**Duplicate Handling**

* Check for exact match: same title AND same author
* If duplicate found: skip and log message
* Different editions or variations: treated as separate books

**Technical Architecture**

**Repository Interfaces**

public interface UserRepository { ... }

public interface BookRepository { ... }

public interface RentalRepository { ... }

**Service Layer**

@Service

public class BookService {

private final BookRepository bookRepository;

private final RentalRepository rentalRepository;

// Constructor injection demonstrates DI concepts

}

**Spring Boot Components**

* @SpringBootApplication main class
* @RestController for API endpoints
* @Service for business logic
* @Repository for data access
* @Component for utility classes

**Core Requirements Mapping**

**Java Fundamentals**

* Control statements for search/filter logic
* ArrayLists for book collections
* Exception handling for file I/O operations
* Loops for data processing

**Object-Oriented Design**

* User, Book, Rental entity classes
* Service classes with proper encapsulation
* Interface-based repository pattern

**Spring Framework Integration**

* Component scanning and dependency injection
* Role-based access control
* RESTful endpoint design
* Configuration management

**Unit Testing**

* JUnit tests for service classes
* Mockito for repository mocking
* Test coverage for business logic
* Authentication and authorization testing

**Development Phases**

**Phase 1: Core Structure**

1. Entity classes (User, Book, Rental)
2. Repository interfaces and JSON implementations
3. Basic service classes with dependency injection

**Phase 2: Authentication**

1. User registration and login endpoints
2. Session management
3. Role-based access control

**Phase 3: Book Management**

1. CRUD operations for books
2. Search functionality
3. Rental/return system

**Phase 4: Admin Features**

1. User management (promote/demote/delete)
2. Import/export functionality
3. Administrative reporting

**Phase 5: Testing & Documentation**

1. Comprehensive unit tests
2. Integration testing
3. API documentation
4. User guide and setup instructions

**Success Criteria**

* Demonstrates all required Java and Spring concepts
* Clean, maintainable code architecture
* Functional authentication and authorization
* Complete import/export capability
* Comprehensive test coverage
* Professional presentation quality