**Library Management System**

**Scope Freeze v1.0**

**Authentication & User Management**

**User Registration**

* Users register with username, email, password
* All new accounts created with USER role only
* Username and email must be unique
* Minimum password length: 8 characters

**Bootstrap Admin**

* System creates default admin account: username="admin", password="admin123"
* Marked as protected (cannot be deleted or demoted)
* Ensures system always has at least one admin

**User Self-Management**

* Users can edit own profile (username, email, password)
* Users can view own account information
* Users can view their current rentals

**Admin User Management**

* View all user accounts and roles
* Edit any user account information
* Promote USER to ADMIN
* Demote ADMIN to USER (except protected accounts)
* Delete user accounts (except protected accounts)

**Book Management**

**Book Data Structure**

* Each book has: title, author, genre, status
* Status: AVAILABLE or RENTED
* Single copy per book (one title/author combination)

**User Book Operations**

* Browse all available books
* Search books by title or author
* Rent available books (changes status to RENTED)
* Return rented books (changes status to AVAILABLE)
* View list of currently rented books

**Admin Book Operations**

* All user book operations PLUS:
* Add new books to catalog
* Edit existing book information
* Delete books from catalog
* Cannot delete books that are currently RENTED

**Import/Export Functionality**

**Export (Admin Only)**

* Export complete book catalog to JSON file
* Includes all books with current status
* Downloaded file format: library\_export\_YYYY-MM-DD.json

**Import (Admin Only)**

* Import books from JSON file
* Append-only operation (adds to existing catalog)
* Skip books with identical title AND author
* Display summary: books added vs books skipped
* All imported books default to AVAILABLE status

**Data Storage**

**JSON File Structure**

**users.json**

[

{

"id": "550e8400-e29b-41d4-a716-446655440000",

"username": "admin",

"passwordHash": "hashedPassword",

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

"role": "ADMIN",

"protected": true,

"mustChangePassword": true

}

]

**books.json**

[

{

"id": "550e8400-e29b-41d4-a716-446655440001",

"title": "Clean Code",

"author": "Robert Martin",

"genre": "Programming",

"status": "AVAILABLE"

}

]

**rentals.json**

[

{

"id": "550e8400-e29b-41d4-a716-446655440002",

"userId": "550e8400-e29b-41d4-a716-446655440000",

"bookId": "550e8400-e29b-41d4-a716-446655440001",

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

"status": "ACTIVE",

"returnDate": null

}

]

**Business Rules**

**Data Standards**

* **ID Format:** UUID strings for all entities (prevents import collisions)
* **Text Processing:** Case-insensitive uniqueness checks, trim whitespace
* **Timestamps:** UTC format (ISO 8601)
* **Username Reservation:** "admin" username is reserved for system admin

**Authentication Rules**

* **Password Policy:** Minimum 8 characters, must contain letter and number
* **Default Admin:** Must change password on first login
* **Session Management:** Login/logout with session-based authentication

**Rental Rules**

* Only AVAILABLE books can be rented
* Maximum active rentals per user: 5
* Only the renter (or admin) can return books
* Cannot rent already RENTED books (409 Conflict)
* Cannot return non-rented books (409 Conflict)

**Deletion Rules**

* Cannot delete RENTED books (must return first)
* Cannot delete users with ACTIVE rentals (must return books first)
* Cannot delete protected admin accounts
* System must maintain at least one admin account

**Duplicate Prevention**

* Case-insensitive unique usernames and emails
* No duplicate book combinations (same title AND author, case-insensitive)
* Import skips duplicates and reports summary

**Access Control**

**USER Role Can Access:**

* Book browsing and searching
* Rent/return operations
* View/edit own profile
* View own rentals

**ADMIN Role Can Access:**

* All USER capabilities PLUS:
* Book management (add/edit/delete)
* User management (view/edit/delete/promote)
* Import/export operations

**Technical Requirements**

**Framework & Architecture**

* Spring Boot application
* Repository pattern with dependency injection
* JSON file-based persistence
* Constructor injection for all dependencies

**Required Spring Components**

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

**Testing Requirements**

* JUnit 5 for unit testing
* Mockito for service mocking
* Test coverage for business logic
* Mock repository implementations for testing

**Documentation**

* Swagger UI automatically generated at /swagger-ui.html for API testing
* OpenAPI specification for professional API documentation

**API Endpoints (Basic Structure)**

**Authentication**

* POST /auth/register - User registration
* POST /auth/login - User login
* POST /auth/logout - User logout

**User Management**

* GET /users/me - View own profile
* PUT /users/me - Edit own profile
* GET /users - Admin: list all users
* PUT /users/{id} - Admin: edit user
* DELETE /users/{id} - Admin: delete user
* POST /users/{id}/promote - Admin: promote to admin

**Book Operations**

* GET /books?q=&author=&genre=&year=&page=0&size=20&sort=title&dir=asc - Search/list books with pagination
* POST /books - Admin: add book
* PUT /books/{id} - Admin: edit book
* DELETE /books/{id} - Admin: delete book
* POST /books/{id}/rent - Rent book
* POST /books/{id}/return - Return book

**Import/Export**

* GET /admin/export - Export books catalog only (JSON array)
* POST /admin/import - Import books (append-only, skip duplicates)

**HTTP Status Codes**

**Standard Responses**

* **200 OK** - Successful operation
* **400 Bad Request** - Validation errors (invalid email, blank fields)
* **401 Unauthorized** - Not logged in
* **403 Forbidden** - Insufficient permissions (role violation)
* **404 Not Found** - Resource not found (book/user)
* **409 Conflict** - Business rule violation (duplicate data, rental conflicts)
* **422 Unprocessable Entity** - Invalid import file format

**API Response Formats**

**Paginated Book Search Response**

{

"content": [

{

"id": "550e8400-e29b-41d4-a716-446655440001",

"title": "Clean Code",

"author": "Robert Martin",

"genre": "Programming",

"status": "AVAILABLE"

}

],

"page": 0,

"size": 20,

"total": 150,

"totalPages": 8

}

**Import Summary Response**

{

"added": 15,

"skipped": 3,

"errors": [

"Row 5: Missing author field",

"Row 12: Invalid year format"

]

}

**Technical Implementation**

**File I/O Requirements**

* **Atomic Writes:** Write to temp file, then rename to prevent corruption
* **Single Writer Lock:** Synchronize JSON file saves to prevent conflicts
* **Data Directory:** Configurable path (default: data/)
* **File Size Limits:** Maximum 10MB for import files

**Search Implementation**

* **Default Parameters:** page=0, size=20, sort=title, dir=asc
* **Search Logic:** Case-insensitive contains matching for title and author
* **Response Format:** {content: [...], page: 0, size: 20, total: 150}

**Out of Scope for v1.0**

* Database integration (JPA/Hibernate)
* Advanced authentication (JWT, OAuth)
* Email notifications
* Due dates and late fees
* Multiple copies per book
* Advanced UI beyond basic REST endpoints
* Real-time notifications
* Advanced search filters

**Success Criteria**

* All authentication workflows functional
* Role-based access control working
* Complete CRUD operations for books and users
* Import/export functionality operational
* Comprehensive unit test coverage
* Clean dependency injection architecture
* Professional code organization and documentation