**WebLibrary Project Documentation**

## Overview

**WebLibrary** is a RESTful backend service built using Spring Boot, with JWT-based authentication and role-based authorization. It leverages MongoDB for data persistence and includes integration tests and test scripts powered by Node.js, npm, Jest, and Supertest.

## Technologies & Characteristics

### Backend:

* + **Java & Spring Boot**
  + RESTful API endpoints.
  + **Spring Security with JSON Web Tokens (JWT) for stateless authentication.**
  + Supports role-based access control with roles such as ROLE\_USER and ROLE\_ADMIN.
  + **MongoDB**
  + Stores book and user data.
  + Spring Data MongoDB is used for data access.
  + Unique index on username to prevent duplicate registrations.
  + Secure endpoints:
  + **Authenticated Users** (either role) can read and create books.
  + **Admins only** can update or delete books.

### Testing & Scripting:

* + **Maven**
  + Build and dependency management.
  + **Node.js, npm, Jest, and Supertest**
  + Integration tests and scripts for seeding sample data.
  + Additional npm scripts to clean up test data and users.

### Configuration:

* + **JWT Configuration**
  + JWT secret and expiration are configured via application.properties (or overridden via environment variables).
  + **Method-Level Security**
  + Uses @PreAuthorize for admin-only endpoints.
  + **Servlet API & Jakarta Namespace**

## Setup & Running the Project

### Prerequisites:

* + **Java (JDK 17 or later)**
  + **Maven**
  + **MongoDB**
  + Ensure MongoDB is installed and running (e.g., by executing mongod).
  + **Node.js & npm (v18 or later)**

### Step 1: Clone the Project

git clone https://github.com/Yakub-M/web-library

cd web-library

Description of project files and structure:

|  |
| --- |
| web-library/  ├── src/main/java/com/yakub/weblibrary  │ ├── config *(Contains configuration classes)*  │ ├── controller *(Contains REST API controllers that handle incoming HTTP requests)*  │ ├── model *(Contains domain model classes representing your data entities)*  │ ├── repository *(Contains Spring Data repository interfaces for database operations)*  │ ├── security *(Contains security-related classes for JWT authentication and authorization)*  │ └── WebLibraryApplication.java *(The main entry point of the Spring Boot application)*  ├── src/test/java/com/yakub/weblibrary *(Contains JUnit tests for the Spring Boot application)*  └── jest-tests/  ├── test *(Contains tests written in JavaScript using Jest and Supertest to test API endpoints)*  ├── scripts *(Contains Node.js scripts for managing sample data in the MongoDB database)*  └── package.json (*The Node.js package configuration file with dependencies and npm scripts for testing and running scripts)* |

### Step 2: Build and Run the Spring Boot Application

### ****1. Build the Project:**** Navigate to the project root and run:

mvn clean install

**2.** **Run the Application:** Start the application with:

mvn spring-boot:run

**3.** **Access at URL:** The server will be running at:

http://localhost:8080

**Step 3: Using the API Endpoints**

* **Register a New User:**

POST /api/auth/register

*Payload Example:*

{

“username”: “admin”,

“password”: “password”,

“roles: [“ROLE\_ADMIN”]

}

* **Login:**

POST /api/auth/login

*Payload Example:*

{

“username”: “admin”,

“password”: “password”

}

*Response:* A JWT token, e.g.,

{

“token”: “eyJhbGciOiJIUzI1NiIsInR5cCI6...”

}

* **Get All Books:**

GET /api/books/

*Requires Authorization Header:*

Authorization: Bearer <JWT\_TOKEN>

* **Create a Book**

POST /api/books/

*Requires a valid JWT token (ROLE\_USER or ROLE\_ADMIN).*

*Payload Example:*

{

“titile”: “Test Book”,

“author”: “John Doe”,

“publicationDate”: “2025-02-11”,

“genre”: “Science Fiction”,

“isbn”: “1234567890”

}

* **Update/Delete a Book:**

PUT /api/books/{id} and DELETE /api/books/{id}

*These endpoints require a JWT token for a user with the ROLE\_ADMIN role*

* **Cleanup Test Users:**

DELETE /api/auth/cleanup/{username}

*This endpoint is protected by @PreAuthorize(“hasRole(‘ADMIN’)”).*

**API Endpoints**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Endpoint** | **HTTP Method** | **CRUD Operation** | ****Required Role**** | ****Description**** |
| /api/auth/register | POST | Create | Public | Registers a new user (defaults to ROLE\_USER if no role provided). |
| /api/auth/login | POST | Read (Auth) | Public | Authenticates a user and returns a JWT token. |
| /api/auth/cleanup/{username} | DELETE | Delete | ROLE\_ADMIN | Deletes the specified user (for test cleanup purposes). |
| /api/books/ | GET | Read | ROLE\_USER or ROLE\_ADMIN | Retrieves all books. |
| /api/books/ | POST | Create | ROLE\_USER or ROLE\_ADMIN | Creates a new book. |
| /api/books/{id} | GET | Read | ROLE\_USER or ROLE\_ADMIN | Retrieves a single book by its ID. |
| /api/books/{id} | PUT | Update | ROLE\_ADMIN | Updates an existing book (admin-only). |
| /api/books/{id} | DELETE | Delete | ROLE\_ADMIN | Deletes a book (admin-only). |
| /api/books/title/{title} | GET | Read | ROLE\_USER or ROLE\_ADMIN | Searches for books by title. |
| /api/books/author/{author} | GET | Read | ROLE\_USER or ROLE\_ADMIN | Searches for books by author. |
| /api/books/genre/{genre} | GET | Read | ROLE\_USER or ROLE\_ADMIN | Searches for books by genre. |
| /api/books/isbn/{isbn} | GET | Read | ROLE\_USER or ROLE\_ADMIN | Searches for books by ISBN. |

**Step 4: Testing with Postman or cURL**

* **Using Postman:**Create requests as described above (set the proper HTTP method, headers, and JSON payload).
* **Using cURL:**  
  *Example login request:*

curl -X POST http://localhost:8080/api/auth/login \

-H "Content-Type: application/json" \

-d '{"username": "admin", "password": "admin\_password"}'

**Step 5: Running Integration Tests & Scripts**

**1. In a separate terminal, navigate to the Jest Tests Directory:**

cd jest-tests

**2. Install Node.js Dependencies:**

npm install

**3. Run Integration Tests:**

npm run test

**4. Seed Sample Books:**

npm run seed-books

**5. Clear Sample Books:**

npm run clear-books

**6. Clear Test Users:**

npm run cleanup