Document Manager API

A comprehensive Spring Boot REST API for managing scanned documents with user authentication, file uploads, and tagging system.

🚀 Features

- **User Management**: Registration, login, profile management
- **JWT Authentication**: Secure token-based authentication
- **Document Management**: Upload, download, search, and organize documents
- **Tag System**: Categorize documents with tags
- **File Storage**: Secure file upload and storage
- **Role-Based Security**: User and admin roles
- **API Documentation**: OpenAPI/Swagger integration
- **Comprehensive Testing**: Unit and integration tests

🛠 Tech Stack

- **Java 21**
- **Spring Boot 3.5.5**
- **Spring Security** (JWT Authentication)
- **Spring Data JPA** (Hibernate)
- **PostgreSQL** (Production) / **H2** (Development)
- **Lombok** (Code generation)
- **Maven** (Build tool)
- **JUnit 5** (Testing)

📋 Prerequisites

- Java 21 or higher
- Maven 3.6+
- PostgreSQL (for production) or H2 (for development)

🏃 Quick Start

1. Clone the Repository

```bash
git clone <repository-url>
cd docmanager
```

2. Run with Development Profile (H2 Database)

```bash
mvn spring-boot:run
```

3. Access the Application

- **API Base URL**: http://localhost:8080/api
- **H2 Console**: http://localhost:8080/h2-console
- **Swagger UI**: http://localhost:8080/swagger-ui/index.html

4. Default Test Users

The application automatically creates test users on startup:

Admin User:

- Username: `admin`
- Password: `admin123`
- Roles: ADMIN, USER

Regular User:

- Username: `testuser`
- Password: `password123`
- Roles: USER

📚 API Endpoints

Authentication Endpoints

```http
```

```
POST /api/auth/register      # User registration
POST /api/auth/login         # User login
POST /api/auth/refresh       # Refresh JWT token
PUT  /api/auth/change-password # Change password
GET  /api/auth/me            # Get current user info
```

User Management Endpoints

```http
GET    /api/users           # Get all users (Admin only)
GET    /api/users/{id}      # Get user by ID
PUT    /api/users/{id}      # Update user
DELETE /api/users/{id}      # Delete user (Admin only)
GET    /api/users/check-username/{username} # Check username availability
GET    /api/users/check-email/{email}      # Check email availability
```

Document Management Endpoints

```http
GET    /api/documents               # Get user's documents
POST   /api/documents               # Create document
GET    /api/documents/{id}          # Get document by ID
PUT    /api/documents/{id}          # Update document
DELETE /api/documents/{id}          # Delete document
POST   /api/documents/upload        # Upload file
GET    /api/documents/{id}/download     # Download file
GET    /api/documents/search?query=...  # Search documents
GET    /api/documents/stats         # Get document statistics
```

Tag Management Endpoints

```http
```

```
GET    /api/tags          # Get all tags
GET    /api/tags/my       # Get current user's tags
POST   /api/tags          # Create tag
PUT    /api/tags/{id}     # Update tag (Admin only)
DELETE /api/tags/{id}     # Delete tag (Admin only)
GET    /api/tags/search?query=... # Search tags
DELETE /api/tags/unused   # Delete unused tags (Admin only)
GET    /api/tags/stats    # Get tag statistics (Admin only)
```

🔧 Configuration

Database Configuration

H2 (Development)

yaml

```yaml
spring:
  datasource:
    url: jdbc:h2:mem:docmanager
    driver-class-name: org.h2.Driver
    username: sa
    password:
```

PostgreSQL (Production)

yaml

```yaml
spring:
  datasource:
    url: jdbc:postgresql://localhost:5432/docmanager
    driver-class-name: org.postgresql.Driver
    username: your_username
    password: your_password
```

JWT Configuration

yaml

```yaml
app:
 jwt:
  secret: your-secret-key-base64-encoded
  expiration-ms: 86400000  # 24 hours
  refresh-expiration-ms: 604800000  # 7 days
```

File Upload Configuration

```yaml
file:
 upload-dir: ./uploads

spring:
 servlet:
  multipart:
   max-file-size: 10MB
   max-request-size: 10MB
```

🧪 Testing

Run All Tests

```bash
mvn test
```

Run Specific Test Class

```bash
mvn test -Dtest=UserServiceTest
```

Run Integration Tests

```bash
mvn test -Dtest=*IntegrationTest
```

🔐 Authentication

The API uses JWT (JSON Web Token) for authentication. Include the token in the Authorization header:

```
Authorization: Bearer <your-jwt-token>
```

Example Login Flow

1. **Register or Login**:

```bash
curl -X POST http://localhost:8080/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{"usernameOrEmail":"testuser","password":"password123"}'
```

2. **Use the returned JWT token** in subsequent requests:

```bash
curl -X GET http://localhost:8080/api/documents \
  -H "Authorization: Bearer <jwt-token>"
```

📁 Project Structure

```
src/
├── main/java/com/app/docmanager/
│   ├── config/        # Configuration classes
│   ├── controller/    # REST controllers
│   ├── dto/           # Data Transfer Objects
│   ├── entity/        # JPA entities
│   ├── exception/     # Custom exceptions
│   ├── mapper/        # Entity-DTO mappers
│   ├── repository/    # JPA repositories
│   ├── security/      # Security configuration
│   └── service/       # Business logic services
└── test/              # Unit and integration tests
```

🚀 Deployment

Building for Production

```bash
mvn clean package -DskipTests
```

Running with Production Profile

```bash
java -jar target/docmanager-0.0.1-SNAPSHOT.jar --spring.profiles.active=prod
```

📝 Example API Usage

Create a Document

```bash
curl -X POST http://localhost:8080/api/documents \
 -H "Content-Type: application/json" \
 -H "Authorization: Bearer <jwt-token>" \
 -d '{
  "title": "My Important Document",
  "category": "Work",
  "tags": ["important", "work", "2024"]
 }'
```

Upload a File

```bash
curl -X POST http://localhost:8080/api/documents/upload \
 -H "Authorization: Bearer <jwt-token>" \
 -F "file=@document.pdf" \
 -F "title=Annual Report" \
 -F "category=Reports"
```

Search Documents

```bash
curl -X GET "http://localhost:8080/api/documents/search?query=annual" \
 -H "Authorization: Bearer <jwt-token>"
```

🔍 Monitoring and Debugging

H2 Console Access (Development)

- URL: http://localhost:8080/h2-console

- JDBC URL: `jdbc:h2:mem:docmanager`
- Username: `sa`
- Password: (leave empty)

API Documentation

Access the Swagger UI at: http://localhost:8080/swagger-ui/index.html

🤝 Contributing

1. Fork the repository
2. Create a feature branch: `git checkout -b feature/new-feature`
3. Commit your changes: `git commit -am 'Add new feature'`
4. Push to the branch: `git push origin feature/new-feature`
5. Submit a pull request

📄 License

This project is licensed under the MIT License - see the LICENSE file for details.

🆘 Troubleshooting

Common Issues

1. **Port 8080 already in use**
   - Change the port in `application.yml`: `server.port: 8081`
2. **JWT Token expired**
   - Login again to get a new token
3. **File upload fails**
   - Check file size limits in configuration
   - Ensure upload directory exists and is writable
4. **Database connection issues**
   - Verify database configuration in `application.yml`
   - Ensure PostgreSQL is running (for production profile)

Getting Help

- Check the logs in the console for detailed error messages
- Use the H2 console to inspect database state in development

- Review the test files for usage examples