Library Management System

**Scope:**

This document provides the API specification for the Library Management System (LMS). The system allows users based on roles to manage, borrow and return books, and view transaction history.

The LMS has been implemented as a Maven Spring Boot Project with PostgreSQL & JWT. The entire system has been containerized using Docker.

**Features:**

1. **Database Management:**

**Since the dataset has a specific structure and there are co-relations between the various data, the LMS has been implemented with a Relational Database using PostgreSQL to support ACID transactions. There are 3 tables created for Books Management (Table – Books), User Management (Table – User) and Transaction Management (History – Table). Based on the functionality required, vertical scaling has been chosen as the appropriate scaling option.**

**Below are the DDL scripts used:**

1. **Creating the Schema**

**create schema librarymanagementschema authorization postgres;**

1. **Creating Books table for Books Management**

**CREATE TABLE librarymanagementschema.books (**

**bookid SERIAL PRIMARY KEY,**

**title VARCHAR(255),**

**author VARCHAR(255),**

**category VARCHAR(255),**

**availability VARCHAR(255),  
books\_count int**

**);**

1. **Creating User table for User Management**

**CREATE TABLE librarymanagementschema.users (**

**id SERIAL PRIMARY KEY,**

**name VARCHAR(255),**

**email VARCHAR(255),**

**password VARCHAR(255),**

**roles VARCHAR(255)**

**);**

1. **Creating History table for Transaction Management**

**CREATE TABLE librarymanagementschema.history (**

**auditid SERIAL PRIMARY KEY,**

**userid BIGINT,**

**bookid BIGINT,**

**borrowed\_date VARCHAR(255),**

**returned\_date VARCHAR(255),**

**status VARCHAR(255)**

**);**

1. Authentication & Authorization:

The API uses JWT for authentication and Role based Authorization. Users must authenticate to receive a token, which must be included in the Authorization header of subsequent requests. User must be either ADMIN(Librarian)/MEMBER(User). Only admin has privileges to all APIs. Members have access to only search/borrow/return book APIs.

1. APIs
2. Create User API

|  |  |
| --- | --- |
| API - DESCRIPTION | Create USER  This API is used for creating new user with specific roles. The password will be encrypted and saved in DB. This needs to be isolated for ADMIN & MEMBER account creations by the superuser. |
| URL | http://localhost:8081/users/create |
| Http Method | POST |
| Header & Roles access | No Auth header required & No role is required |
| Request JSON | {  "id": null,  "name": "test",  "email": "test@gmail.com",  "password": "test",  "roles":"MEMBER"  } |
| Response JSON | {  "**id**"**:** 8,  "**name**"**:** "test",  "**email**"**:** "test@gmail.com",  "**password**"**:** "$2a$10$cEb159L5SBiRFvh1V/2TneweZD..ltTGdTRoiG8isfeSr0koxJesy",  "**roles**"**:** "MEMBER"  } |

|  |  |
| --- | --- |
| API - DESCRIPTION | Authenticate USER  This API is used to authenticate & authorize users. This will generate a JWT token which will be valid for 30 mins. The users should pass this token in the Header to access services. |
| URL | http://localhost:8081/users/authenticate |
| Http Method | POST |
| Header & Roles access | No Auth header required & No role is required |
| Request JSON | {  "username": "subhasree",  "password":"subha123"  } |
| Response JSON | **e**yJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzdWJoYXNyZWUiLCJpYXQiOjE3NDExOTUyMDQsImV4cCI6MTc0MTE5NzAwNH0.SZgo4j0RdFjLeYPpOzTkn3snUDZkb1fzUwpcKSZ5r1k |

1. Authenticate User API

1. Update User API

|  |  |
| --- | --- |
| API - DESCRIPTION | Update USER details (role/name)  This API helps to update user information such as name and roles by the ADMIN. |
| URL | http://localhost:8081/users/:id |
| Http Method | PUT |
| Header & Roles access | Authorization = Bearer {valid token} only ADMIN can use this API |
| Request JSON | {  "name": "test",  "roles":"ADMIN"  } |
| Response JSON | {  "**id**"**:** 8,  "**name**"**:** "test",  "**email**"**:** "test@gmail.com",  "**password**"**:** "$2a$10$2U7XpxnYnyJ2XcBSbwh/G..S0g7FChHPgRinxpSFLLYIogPRBuQpa",  "**roles**"**:** "ADMIN"  }  Error Responses:  {  "**timestamp**"**:** 1741198797135,  "**status**"**:** 400,  "**error**"**:** "Bad Request",  "**message**"**:** "User not found20",  "**path**"**:** "/users/20"  }  {  "**path**"**:** "/users/1",  "**error**"**:** "Unauthorized",  "**message**"**:** "JWT Token has expired",  "**timestamp**"**:** 1741195199691,  "**status**"**:** 401  }  {  "**path**"**:** "/users/1",  "**error**"**:** "Unauthorized",  "**message**"**:** "Malformed JWT JSON: �쉅����!L��؉",  "**timestamp**"**:** 1741195271898,  "**status**"**:** 401  } |

1. Delete User API

|  |  |
| --- | --- |
| API - DESCRIPTION | delete USER  This API is used to delete existing users by the ADMIN. |
| URL | http://localhost:8081/users/:id |
| Http Method | DELETE |
| Header & Roles access | Authorization = Bearer {valid token} only ADMIN can use this API |
| Request JSON | No Body required |
| Response JSON | 200 status.  Error Response: {  "**timestamp**"**:** 1741198797135,  "**status**"**:** 400,  "**error**"**:** "Bad Request",  "**message**"**:** "User not found20",  "**path**"**:** "/users/20"  } |

1. Book Management:
2. ADD Books

|  |  |
| --- | --- |
| API - DESCRIPTION | ADD Book Using this API Admin can add a new book/existing book. If the same book exists, only available book count will be updated. |
| URL | http://localhost:8081/books/add |
| Http Method | POST |
| Header & Roles access | Authorization = Bearer {valid token} only ADMIN can use this API |
| Request JSON | {  "bookid": null,  "title": "test",  "author":"test",  "category": "Scify",  "availability": "Available"  } |
| Response JSON | Success Response:  {  "**bookid**"**:** 12,  "**title**"**:** "test",  "**author**"**:** "test",  "**category**"**:** "Scify",  "**availability**"**:** "Available",  "**booksCount**"**:** 1  }  Error response:  {  "**timestamp**"**:** 1741195786992,  "**status**"**:** 500,  "**error**"**:** "Internal Server Error",  "**message**"**:** "JSON parse error: Unrecognized field \"bookId\" (class com.example.librarymanagement.app.entity.Books), not marked as ignorable",  "**path**"**:** "/books/add"  } |

1. Update Book details

|  |  |
| --- | --- |
| API - DESCRIPTION | Update Book details ADMINs can update any BOOK fields by passing book id as a path variable |
| URL | http://localhost:8081/books/:bookid |
| Http Method | PUT |
| Header & Roles access | Authorization = Bearer {valid token} only ADMIN can use this API |
| Request JSON | {  "title": "hamlet",  "author": "Shakespeare",  "category": "Fiction",  "availability": "Available"  }  Or only particular fields {  "booksCount": 4  } |
| Response JSON | {  "**bookid**"**:** 9,  "**title**"**:** "dune",  "**author**"**:** "frank",  "**category**"**:** "Scify",  "**availability**"**:** "Available",  "**booksCount**"**:** 4  }  Error Response:  {  "**timestamp**"**:** 1741195616163,  "**status**"**:** 400,  "**error**"**:** "Bad Request",  "**message**"**:** "Book not found with id: 10",  "**path**"**:** "/books/10"  } |

1. Delete Books

|  |  |
| --- | --- |
| API - DESCRIPTION | Delete a Book  This API is used by ADMINs to delete Books completely. Book Count adjustment is handled via Update API & not by this API. |
| URL | http://localhost:8081/books/:bookid |
| Http Method | DELETE |
| Header & Roles access | Authorization = Bearer {valid token} only ADMIN have access to this API |
| Request JSON | NA |
| Response JSON | 200 response  Error Response:  {  "**timestamp**"**:** 1741195737551,  "**status**"**:** 400,  "**error**"**:** "Bad Request",  "**message**"**:** "No Books found with this id: 2",  "**path**"**:** "/books/2"  } |

1. Search Books

|  |  |
| --- | --- |
| API - DESCRIPTION | Search a Book  USERS & ADMINS can search for books with single or multiple search criteria |
| URL | <http://localhost:8081/books?title=hamlet&availability=Available> &category=Fiction&author=homer |
| Http Method | GET |
| Header & Roles access | Authorization = Bearer {valid token} Both ADMIN /Member can access this API |
| Request JSON | NA |
| Response JSON | [  {  "**bookid**"**:** 9,  "**title**"**:** "dune",  "**author**"**:** "frank",  "**category**"**:** "Scify",  "**availability**"**:** "Available",  "**booksCount**"**:** 3  },  {  "**bookid**"**:** 8,  "**title**"**:** "Hamilton",  "**author**"**:** "Ron",  "**category**"**:** "History",  "**availability**"**:** "Available",  "**booksCount**"**:** 1  },..  ]  Error Response:  [] |

1. **Transaction management**
2. **Borrow Books**

|  |  |
| --- | --- |
| API - DESCRIPTON | Borrow a Book  Allows both USERs and ADMINs to borrow books. If more than 1 count of the same book is available, the book count will be updated and remaining will be available upon successful transaction completion. Transaction History for each borrower will be updated with the date and status. |
| URL | http://localhost:8081/borrow |
| Http Method | POST |
| Header & Roles access | Authorization = Bearer {valid token} Both ADMIN & MEMBER have access to this API |
| Request JSON | {  "books": {  "bookid": 9  },  "userInfo": {  "id": 7  }  } |
| Response JSON | {  "**auditid**"**:** 19,  "**borrowed\_date**"**:** "2025-03-05",  "**status**"**:** "Borrowed",  "userInfo":{  "**id**"**:** 7  },  "books":{  "**bookid**"**:** 9  }  }  }  Error Responses: {  "**timestamp**"**:** 1741195250913,  "**status**"**:** 400,  "**error**"**:** "Bad Request",  "**message**"**:** "Book is already borrowed: 9",  "**path**"**:** "/borrow"  } |

1. Return Books

|  |  |
| --- | --- |
| API - DESCRIPTION | Return a Book  Allows both USERs and ADMINs to return borrowed books. If more than 1 count of the same book is present, the book count will be updated upon successful transaction completion. Transaction History for each returner will be updated with the date and status. |
| URL | http://localhost:8081/return |
| Http Method | POST |
| Header & Roles access | Authorization = Bearer {valid token} Both ADMIN & MEMBER have access to this API |
| Request JSON | {  "books": {  "bookid": 9  },  "userInfo": {  "id": 7  }  } |
| Response JSON | {  "**auditid**"**:** 18,  "**borrowed\_date**"**:** "2025-03-05",  "**returned\_date**"**:** "2025-03-05",  "**status**"**:** "Returned",  "userInfo":{  "**id**"**:** 7,  "**name**"**:** "abisheik",  "**email**"**:** "abisheik@gmail.com",  "**roles**"**:** "MEMBER"  },  "books":{  "**bookid**"**:** 9,  "**title**"**:** "dune",  "**author**"**:** "frank",  "**category**"**:** "Scify",  "**availability**"**:** "Available",  "**booksCount**"**:** 4  }  } |

1. Get History of All Transactions

|  |  |
| --- | --- |
| API - DESCRIPTION | Get History  This API is used by ADMINs to obtain the complete transaction history. |
| URL | http://localhost:8081/history |
| Http Method | GET |
| Header & Roles access | Authorization = Bearer {valid token}  Only ADMIN has access to this API |
| Request JSON |  |
| Response JSON | [  {  "**auditid**"**:** 6,  "**borrowed\_date**"**:** "2025-03-05",  "**status**"**:** "Borrowed",  "userInfo":{"id": 7, "name": "abisheik", "email": "abisheik@gmail.com", "roles": "MEMBER"…},  "books":{"bookid": 6, "title": "hamlet", "author": "Shakespeare", "category": "Fiction",…}  },  {  "**auditid**"**:** 7,  "**borrowed\_date**"**:** "2025-03-05",  "**status**"**:** "Borrowed",  "userInfo":{"id": 6, "name": "abi", "email": "abi@gmail.com", "roles": "ADMIN"…},  "books":{"bookid": 6, "title": "hamlet", "author": "Shakespeare", "category": "Fiction",…}  },..  ] |

1. Get History by UserID or BookID or a combination of both

|  |  |
| --- | --- |
| API - DESCRIPTION | Get History by UserID or BookID or a combination of both  This API is ADMINs to list the transactions for specific users, books or books and users using their respective IDs. |
| URL | http://localhost:8081/getHistoryByUserOrBook?userid=5 |
| Http Method | GET |
| Header & Roles access | Authorization = Bearer {valid token}  Only ADMIN has access to this API |
| Request JSON | NA |
| Response JSON | [  {  "auditid": 9,  "borrowed\_date": "2025-03-05",  "returned\_date": "2025-03-05",  "status": "Returned",  "userInfo": {  "id": 5,  "name": "subhasree",  "email": "subha@gmail.com",  "roles": "ADMIN,MEMBER"  },  "books": {  "bookid": 7,  "title": "odyssey",  "author": "homer",  "category": "Fiction",  "availability": "Available",  "booksCount": 2  }  }  ] |