API DOCUMENTATION –

LIBRARY MANAGEMENT SYSTEM

The Library Management System (LMS) is a web-based application that streamlines cataloguing, lending/return workflows, reservations, fines, and user management. It exposes secured REST APIs with JWT-based authentication and role-based authorization (READER, LIBRARIAN, ADMIN).

Base URL

All endpoints are available under the base URL:

http://localhost:8080/api

AUTHENTICATION

1) User Registration

POST /auth/register

Description: Create a new user account (Reader/Librarian/Admin).
Access: Public (no token required).

Request Body (JSON):

```
{
"name": "Nishya",
"email": "Nishya25@gmail.com",
"password": "Nishya@25",
"role": "READER"   // or "LIBRARIAN" or "ADMIN"
}
```

Success Response (200 OK):

```
{
 "success": true,
 "data": {
  "token": "..."
 }
}
```

Error Responses:

- 400 Bad Request — Invalid input / email already used
- 422 Unprocessable Entity — Validation failed

2) Login

POST /auth/login

Description: Authenticate an existing user and receive a JWT.
Access: Public

Request Body (JSON):

```
{
  "email": "Nishya25@gmail.com",
  "password": "Nishya@25"
}
```

Success Response (200 OK):

```
{
  "success": true,
  "data": {
   "token": "..."
  }}
```

Error Responses:

- 401 Unauthorized — Invalid credentials

USERS

1) Get My Profile

GET /users/me

Description: Fetch the authenticated user's profile.
Access: Authenticated users (any role)

Response (200 OK):

```
{
  "success": true,
   "data": {
```

```
  "id": 12,
  "name": "Nishya",
  "email": "Nishya25@gmail.com",
  "role": "READER",
  "status": "ACTIVE",
  "createdAt": "2025-03-26T12:00:00Z"
}}
```


2) List All Users

GET /users
Description: Fetch all users.
Access: ADMIN
Response:


```
{
  "success": true,
  "data": [
    { "id": 1, "name": "John Doe", "email": "john@example.com", "role": "READER" }
  ]
}
```


3) Update User Status

PUT /users/{userId}/status
Description: Activate or deactivate a user account.
Access: ADMIN
Request Body:


```
{ "status": "ACTIVE" }
```


Response:


```
{ "success": true, "data": { "id": 1, "status": "ACTIVE" } }
```


4) Change User Role

PATCH /users/{userId}/role
Description: Change the role of a user.
Access: ADMIN
Request Body:


```
{ "role": "LIBRARIAN" }
```

Response:

{ "success": true, "data": { "id": 1, "role": "LIBRARIAN" } }

5) Get User History

GET /users/{userId}/history
Description: Fetch loan and fine history for a user.
Access: ADMIN or LIBRARIAN
Response:

{ "success": true, "data": { "loans": [...], "fines": [...] } }

BOOKS & CATALOG

1) List Books

GET /books

Description: List all books, with basic filters.
Access: Authenticated users (any role)

Response (200 OK):


```
{
  "success": true,
  "data": [
    {
      "id": 1,
      "title": "Clean Code",
      "authors": "Robert C. Martin",
      "isbn": "9780132350884",
      "category": "Programming",
      "language": "ENGLISH",
      "publishedYear": 2008
    }]}
```

2) Create Book

POST /books

Description: Add a new book to the catalog.
Access: LIBRARIAN or ADMIN

Request Body:

```
{
  "title": "Clean Code",
  "authors": "Robert C. Martin",
  "isbn": "9780132350884",
  "category": "Programming",
  "language": "ENGLISH",
  "publishedYear": 2008
}
```

Response (201 Created):

```
{
  "success": true,
  "data": {
    "id": 10,
    "title": "Clean Code",
    "authors": "Robert C. Martin",
    "isbn": "9780132350884",
    "category": "Programming",
    "language": "ENGLISH",
    "publishedYear": 2008
  }}
```

3) Get Book by ID

GET/books/{bookId}
Access: Authenticated

Response (200 OK):

```
{
  "success": true,
  "data": {
    "id": 10,
    "title": "Clean Code",
    "authors": "Robert C. Martin",
    "isbn": "9780132350884",
    "category": "Programming",
    "language": "ENGLISH",
    "publishedYear": 2008
  }}
```


4) Update Book

PUT/books/{bookId}
Access: LIBRARIAN or ADMIN

Request Body :


```
{
  "category": "Software Engineering",
  "language": "ENGLISH"
}
```


Response (200 OK):


```
{ "success": true, "data": { "id": 10, "category": "Software Engineering", "language": "ENGLISH" } }
```


5) Delete Book

DELETE/books/{bookId}
Access: ADMIN

Response (204 No Content)

6) Advanced Book Search

GET /books/search
Description: Search books with filters like category, language, rating.
Access: Authenticated
Response:

```
{ "success": true, "data": [...] }
```

7) Popular Books

GET /books/popular
Description: Fetch most borrowed books.
Access: Authenticated
Response:

```
{ "success": true, "data": [...] }
```

8) Recommended Books

GET /books/recommended
Description: Get personalized recommendations for a user.
Access: Authenticated
Response:

```
{ "success": true, "data": [...] }
```

9) Update Book Status

PATCH /books/{bookId}/status
Description: Mark a book as archived or unavailable.
Access: LIBRARIAN or ADMIN
Request Body:

```
{ "status": "ARCHIVED" }
```

Response:

```
{ "success": true, "data": { "id": 10, "status": "ARCHIVED" } }
```

COPIES (Physical Copies of a Book)

1) Add a Copy to a Book

POST/copies/book/{bookId}
Access: LIBRARIAN or ADMIN

Request Body:

```
{
  "barcode": "CC-001",
```

```
  "shelfLocation": "A1",
  "status": "AVAILABLE"   // AVAILABLE | ON_LOAN | RESERVED | LOST
}
```

Response (201 Created):

```
{
  "success": true,
  "data": {
    "id": 101,
    "bookId": 10,
    "barcode": "CC-001",
    "shelfLocation": "A1",
    "status": "AVAILABLE"
  }}
```

2) List Available Copies for a Book

GET/copies/available/{bookId}
Access: Authenticated

Response (200 OK):

```
{
  "success": true,
  "data": [
    { "id": 101, "barcode": "CC-001", "shelfLocation": "A1", "status": "AVAILABLE" }
  ]
}
```

3) Get Copy Details

GET /copies/{copyId}
Description: Fetch details of a specific copy.
Access: Authenticated
Response:

```
{ "success": true, "data": { "id": 101, "barcode": "CC-001", "status": "AVAILABLE" } }
```

4) Update Copy Status

PATCH /copies/{copyId}/status
Description: Update copy status (LOST, DAMAGED).

Access: LIBRARIAN or ADMIN
Request Body:

```
{ "status": "LOST" }
```

Response:

```
{ "success": true, "data": { "id": 101, "status": "LOST" } }
```

LOANS (Borrow & Return)

1) Issue (Borrow) a Copy

POST/loans/issue/{copyId}?days=14
Description: Borrow a specific copy for days (default 14).
Access: Authenticated user (READER)

Response (201 Created):


```
{
  "success": true,
  "data": {
    "id": 500,
    "userId": 12,
    "copyId": 101,
    "issuedAt": "2025-03-26T12:00:00Z",
    "dueAt": "2025-04-09T12:00:00Z",
    "status": "ISSUED"
  }
}
```


Errors:

- 400 Bad Request — Copy not available
- 404 Not Found — Copy does not exist

2) Return a Loan

POST /loans/return/{loanId}
Access: Authenticated (the borrower or Librarian/Admin, depending on policy)

Response (200 OK):


```
{
  "success": true,
  "data": {
    "id": 500,
    "returnedAt": "2025-04-08T10:00:00Z",
    "status": "RETURNED"
  }
}
```


3) View My Loans

GET /loans/me
Description: View all active loans for the authenticated user.
Access: Authenticated
Response:


```
{ "success": true, "data": [...] }
```


4) Extend Loan

PATCH /loans/{loanId}/extend
Description: Extend loan due date (policy-based).
Access: READER
Request Body:


```
{ "extraDays": 7 }
```


Response:


```
{ "success": true, "data": { "id": 500, "dueAt": "2025-04-16T12:00:00Z" } }
```


5) List Overdue Loans

GET /loans/overdue
Description: List all overdue loans.
Access: LIBRARIAN or ADMIN
Response:


```
{ "success": true, "data": [...] }
```

RESERVATIONS

1) Place a Reservation

POST/reservations
Access: READER

Request Body:

```
{ "bookId": 10 }
```

Response (201 Created):

```
{
  "success": true,
  "data": {
    "id": 900,
    "userId": 12,
    "bookId": 10,
    "status": "PENDING",     // PENDING | APPROVED | CANCELLED | FULFILLED
    "createdAt": "2025-03-26T12:00:00Z",
    "expiresAt": "2025-03-29T12:00:00Z"
  }
}
```

2) Approve/Cancel Reservation

PATCH/reservations/{reservationId}
Access: LIBRARIAN or ADMIN

Request Body:

```
{ "status": "APPROVED" }   // or "CANCELLED"
```

Response (200 OK):

```
{ "success": true, "data": { "id": 900, "status": "APPROVED" } }
```

3) View My Reservations

GET /reservations/me
Description: View user's reservations.

Access: READER
Response:

```
{ "success": true, "data": [...] }
```

4) Cancel Reservation

DELETE /reservations/{reservationId}
Description: Cancel reservation by user.
Access: READER
Response:

```
{ "success": true, "message": "Reservation cancelled successfully" }
```

FINES

1) View My Fines

GET/fines/me
Access: READER

Response (200 OK):

```
{
  "success": true,
  "data": [ {
     "id": 300,
     "loanId": 500,
     "amount": 60.0,
     "calculatedAt": "2025-04-10T08:30:00Z",
     "paid": false
   }]}
```

2) Pay a Fine

POST/fines/{fineId}/pay
Access: READER (fine owner) or LIBRARIAN/ADMIN

Response (200 OK):

{ "success": true, "data": { "id": 300, "paid": true, "paidAt": "2025-04-10T09:00:00Z" } }

3) List Unpaid Fines

GET /fines/unpaid
Description: List all unpaid fines.
Access: LIBRARIAN or ADMIN
Response:

{ "success": true, "data": [...] }

4) Waive Fine

POST /fines/{fineId}/waive
Description: Waive a fine (Admin only).
Access: ADMIN
Response:

{ "success": true, "data": { "id": 300, "waived": true } }

CATEGORIES

1) List Categories

GET /categories
Access: Authenticated
Description: List all categories.

Response:


```
{
  "success": true,
  "data": [
    { "id": 1, "name": "Programming" },
    { "id": 2, "name": "Science" }
  ]
}
```

2) Create Category

POST /categories
Access: LIBRARIAN or ADMIN
Description: Add a new category.

Request Body:

```
{
  "name": "Data Science"
}
```

Response:

```
{
  "success": true,
  "data": {
    "id": 3,
    "name": "Data Science"
  }
}
```

3) Update Category

PUT /categories/{id}
Access: LIBRARIAN or ADMIN
Description: Update category details.

Request Body:

```
{
  "name": "Machine Learning"
}
```

Response:

```
{
  "success": true,
  "data": {
```

```
    "id": 3,
    "name": "Machine Learning"
  }
}
```


4) Delete Category

DELETE /categories/{id}
Access: ADMIN
Description: Remove a category.

Response:


```
{
  "success": true,
  "message": "Category deleted successfully"
}
```

LOAN POLICIES

1) View Loan Policy

GET /policies/loan
Access: ADMIN
Description: Retrieve current loan policy settings.

Response:


```
{
  "success": true,
  "data": {
    "maxBooks": 5,
    "loanDays": 14,
    "finePerDay": 10.0
  }
}
```

2) Update Loan Policy

PUT /policies/loan
Access: ADMIN
Description: Update loan policy settings.

Request Body:

```
{
  "maxBooks": 7,
  "loanDays": 21,
  "finePerDay": 15.0
}
```

Response:

```
{
  "success": true,
  "data": {
    "maxBooks": 7,
    "loanDays": 21,
    "finePerDay": 15.0
  }
}
```

REVIEWS

1) Add Review

POST /books/{bookId}/reviews
Access: READER
Description: Add a review for a specific book.

Request Body:


```
{
  "rating": 5,
  "comment": "Excellent book on clean coding!"
}
```


Response:


```
{
  "success": true,
  "data": {
    "id": 101,
    "bookId": 10,
    "userId": 12,
    "rating": 5,
    "comment": "Excellent book on clean coding!",
    "createdAt": "2025-03-26T12:00:00Z"
  }
}
```


2) Get Reviews for a Book

GET /books/{bookId}/reviews
Access: Authenticated
Description: Fetch all reviews for a given book.

Response:


```
{
  "success": true,
  "data": [
    { "id": 101, "rating": 5, "comment": "Excellent book!", "userId": 12 }
  ]
}
```

REPORTS

1) Reports Dashboard

GET /reports/dashboard
Description: Summary of loans, fines, popular books.
Access: ADMIN
Response:


```
{
  "success": true,
  "data": {
    "totalLoans": 120,
    "overdueLoans": 15,
    "finesCollected": 500.0
  }
}
```