Topic: Library Catalogue System

1. Entities

1.1. Book

- id (UUID): Unique identifier for each book.
- title (String): Title of the book.
- authorId (UUID): ID of the author of the book.
- categoryId (UUID): ID of the book's category.
- publicationDate (Date): Date of publication.
- isbn (String): International Standard Book Number.
- status (Enum): Availability status (AVAILABLE, BORROWED, RESERVED).

1.2. Author

- id (UUID): Unique identifier for the author.
- name (String): Author's full name.
- biography (String): A short biography.

1.3. Category

- id (UUID): Unique identifier for the category.
- name (String): Name of the category.
- description (String): Description of the category.

1.4. User

- id (UUID): Unique identifier for the user.
- name (String): User's full name.
- email (String): Email address.
- borrowedBooks (List<UUID>): List of borrowed book IDs.
- role (Enum): User role (ADMIN, MEMBER).

2. Operations

2.1. Book Operations

- Add a new book.
- Update book details.
- Delete a book.
- Get book details by ID.
- List all books (with filters and pagination).
- Search for books by title, author, or category.

- Borrow/Reserve/Return a book.

2.2. Author Operations

- Add a new author.
- Update author details.
- Delete an author.
- Get author details by ID.
- List all authors.

2.3. Category Operations

- Add a new category.
- Update category details.
- Delete a category.
- Get category details by ID.
- List all categories.

2.4. User Operations

- Register a new user.
- Update user details.
- Get user details by ID.
- List all users.
- View borrowed books.

3. REST API Design

3.1. Endpoints

Books

- GET /api/books
  - List books with filters (e.g., title, category, author).
  - Support pagination (page, size) and sorting (sort).
  - Cache responses with a TTL of 5 minutes.
- POST /api/books
  - Create a new book.
  - Requires ADMIN role.
- GET /api/books/{id}
  - Retrieve a book by ID.
- PUT /api/books/{id}
  - Update book details.
  - Requires ADMIN role.
- DELETE /api/books/{id}

    - Delete a book.
    - Requires ADMIN role.
- PATCH /api/books/{id}/status
    - Update book status (BORROWED, RESERVED, AVAILABLE).

Authors

- GET /api/authors
    - List all authors.
    - Pagination and sorting supported.
- POST /api/authors
    - Add a new author.
    - Requires ADMIN role.
- GET /api/authors/{id}
    - Get author details by ID.
- PUT /api/authors/{id}
    - Update author details.
    - Requires ADMIN role.
- DELETE /api/authors/{id}
    - Delete an author.
    - Requires ADMIN role.

Categories

- GET /api/categories
    - List all categories.
- POST /api/categories
    - Add a new category.
    - Requires ADMIN role.
- GET /api/categories/{id}
    - Get category details by ID.
- PUT /api/categories/{id}
    - Update category details.
    - Requires ADMIN role.
- DELETE /api/categories/{id}
    - Delete a category.
    - Requires ADMIN role.

Users

- POST /api/users/register
    - Register a new user.
- GET /api/users/{id}
    - Get user details by ID.

- GET /api/users/{id}/borrowed-books
    - Get the list of books borrowed by the user.

4. Functional and Non-Functional Requirements

Functional

- Users can search, borrow, reserve, and return books.
- Admins can manage books, authors, and categories.
- All endpoints provide meaningful status codes.

Non-Functional

- API conforms to the Richardson Maturity Model Level 3.
- Use JWT for authentication and role-based access control.
- Pagination implemented for lists.
- Cache frequently accessed endpoints (/api/books, /api/authors).