Library REST API Design

Functional Requirements

1. **CRUD Operations**: The API must support Create, Read, Update, and Delete operations for books, authors, and categories.
2. **Filtering**: The API must allow filtering of books, authors, and categories based on various attributes.
3. **Pagination**: The API must support pagination for listing books, authors, and categories.
4. **Validation**: All inputs must be validated to ensure data integrity (e.g., valid UUIDs, dates, and required fields).

Non-Functional Requirements

1. **Security**: The API must implement authentication and authorization mechanisms (e.g., OAuth, JWT) to protect data.
2. **Performance**: The API should be optimized for performance, ensuring quick response times, especially for search and listing operations.
3. **Scalability**: The API should be designed to scale horizontally to handle increased load.
4. **Documentation**: The API must be well-documented using tools like Swagger or OpenAPI, providing clear information about endpoints, request/response formats, and error codes.
5. **Error Handling**: The API must provide meaningful error messages and HTTP status codes for client and server errors.
6. **Consistency**: The API must follow RESTful principles and provide consistent resource URIs and HTTP methods.

Entities

1. **Book**
    - **Attributes**:
        - id: Unique identifier for the book (UUID)
        - title: Title of the book
        - author_id: Reference to the Author entity (UUID)
        - category_id: Reference to the Category entity (UUID)
        - published_date: Date when the book was published

        - isbn: International Standard Book Number
        - summary: Short summary of the book
2. **Author**
    - **Attributes**:
        - id: Unique identifier for the author (UUID)
        - name: Name of the author
        - biography: Short biography of the author
        - birth_date: Date of birth of the author
        - death_date: Date of death of the author (nullable)
3. **Category**
    - **Attributes**:
        - id: Unique identifier for the category (UUID)
        - name: Name of the category
        - description: Description of the category

Operations

Book Operations

- **Create a new book**
    - **Endpoint**: POST /books
    - **Request body**

```
{
   "title": "string",
   "author_id": "UUID",
   "category_id": "UUID",
   "published_date": "date",
   "isbn": "string",
   "summary": "string"
}
```

Responses:

- 201 Created: Book successfully created.
- 400 Bad Request: Invalid input data.
- 401 Unauthorized: Authentication required.
- 403 Forbidden: Insufficient permissions.
- 500 Internal Server Error: Server encountered an unexpected condition.

Retrieve a book by ID

- **Endpoint**: GET /books/{id}
- **Responses**:

```
{
   "id": "UUID",
   "title": "string",
   "author_id": "UUID",
   "category_id": "UUID",
   "published_date": "date",
   "isbn": "string",
   "summary": "string"
}
```

- **Responses**:
  - 200 OK: Book data retrieved successfully.
  - 404 Not Found: Book not found.
  - 401 Unauthorized: Authentication required.
  - 403 Forbidden: Insufficient permissions.
  - 500 Internal Server Error: Server encountered an unexpected condition.

Update a book by ID

- **Endpoint**: PUT /books/{id}
- **Request Body**:

json

Copy code

```
{
   "title": "string",
   "author_id": "UUID",
   "category_id": "UUID",
   "published_date": "date",
   "isbn": "string",
   "summary": "string"
}
```

- **Responses**:
  - 200 OK: Book updated successfully.
  - 400 Bad Request: Invalid input data.

    - 404 Not Found: Book not found.
    - 401 Unauthorized: Authentication required.
    - 403 Forbidden: Insufficient permissions.
    - 500 Internal Server Error: Server encountered an unexpected condition.

- Delete **a book by ID**
  - **Endpoint**: DELETE /books/{id}
  - **Responses**:
    - 204 No Content: Book deleted successfully.
    - 404 Not Found: Book not found.
    - 401 Unauthorized: Authentication required.
    - 403 Forbidden: Insufficient permissions.
    - 500 Internal Server Error: Server encountered an unexpected condition.
- **List all books**
  - **Endpoint**: GET /books
  - **Query Parameters**:
    - author_id (optional): Filter by author
    - category_id (optional): Filter by category
    - published_date (optional): Filter by published date (range)
    - title (optional): Filter by title (partial match)
    - page (optional): Page number for pagination
    - page_size (optional): Number of items per page for pagination
  - **Responses**

```
{
   "total": "int",
   "page": "int",
   "page_size": "int",
   "books": [
      {
         "id": "UUID",
```

```
            "title": "string",
            "author_id": "UUID",
            "category_id": "UUID",
            "published_date": "date",
            "isbn": "string",
            "summary": "string"
        },
        ...
    ]
}
```


Responses:

- 200 OK: List of books retrieved successfully.
- 400 Bad Request: Invalid query parameters.
- 401 Unauthorized: Authentication required.
- 403 Forbidden: Insufficient permissions.
- 500 Internal Server Error: Server encountered an unexpected condition.

Author Operations

- Create **a new author**

- **Endpoint**: POST /authors
- **Request Body**:


```
{
    "name": "string",
    "biography": "string",
    "birth_date": "date",
    "death_date": "date"
}
```


- **Responses**:

    - 201 Created: Author successfully created.
    - 400 Bad Request: Invalid input data.
    - 401 Unauthorized: Authentication required.
    - 403 Forbidden: Insufficient permissions.
    - 500 Internal Server Error: Server encountered an unexpected condition.

- **Retrieve an author by ID**

- **Endpoint**: GET /authors/{id}
- **Responses**:


```
{
  "id": "UUID",
  "name": "string",
  "biography": "string",
  "birth_date": "date",
  "death_date": "date"
}
```


- **Responses**:
    - 200 OK: Author data retrieved successfully.
    - 404 Not Found: Author not found.
    - 401 Unauthorized: Authentication required.
    - 403 Forbidden: Insufficient permissions.
    - 500 Internal Server Error: Server encountered an unexpected condition.

- **Update an author by ID**

- **Endpoint**: PUT /authors/{id}
- **Request Body**:


```
{
  "name": "string",
  "biography": "string",
  "birth_date": "date",
  "death_date": "date"
```

}

- **Responses**:
    - 200 OK: Author updated successfully.
    - 400 Bad Request: Invalid input data.
    - 404 Not Found: Author not found.
    - 401 Unauthorized: Authentication required.
    - 403 Forbidden: Insufficient permissions.
    - 500 Internal Server Error: Server encountered an unexpected condition.

- **Delete an author by ID**

- **Endpoint**: DELETE /authors/{id}
- **Responses**:
    - 204 No Content: Author deleted successfully.
    - 404 Not Found: Author not found.
    - 401 Unauthorized: Authentication required.
    - 403 Forbidden: Insufficient permissions.
    - 500 Internal Server Error: Server encountered an unexpected condition.

 List all authors

- **Endpoint**: GET /authors
- **Query Parameters**:
    - name (optional): Filter by name (partial match)
    - birth_date (optional): Filter by birth date (range)
    - page (optional): Page number for pagination
    - page_size (optional): Number of items per page for pagination
- **Responses**:

{

"total": "int",

"page": "int",

"page_size": "int",

"authors": [

```
    {
      "id": "UUID",
      "name": "string",
      "biography": "string",
      "birth_date": "date",
      "death_date": "date"
    },
    ...
  ]
}
```

- **Responses**:
  - 200 OK: List of authors retrieved successfully.
  - 400 Bad Request: Invalid query parameters.
  - 401 Unauthorized: Authentication required.
  - 403 Forbidden: Insufficient permissions.
  - 500 Internal Server Error: Server encountered an unexpected condition.

Category Operations

- **Create a new category**

- **Endpoint**: POST /categories
- **Request Body**:

```
{
  "name": "string",
  "description": "string"
}
```

- **Responses**:
  - 201 Created: Category successfully created.
  - 400 Bad Request: Invalid input data.
  - 401 Unauthorized: Authentication required.
  - 403 Forbidden: Insufficient permissions.

    - 500 Internal Server Error: Server encountered an unexpected condition.

- **Retrieve a category by ID**

- **Endpoint**: GET /categories/{id}
- **Responses**:

```
{
  "id": "UUID",
  "name": "string",
  "description": "string"
}
```

- **Responses**:
    - 200 OK: Category data retrieved successfully.
    - 404 Not Found: Category not found.
    - 401 Unauthorized: Authentication required.
    - 403 Forbidden: Insufficient permissions.
    - 500 Internal Server Error: Server encountered an unexpected condition.

- **Update a category by ID**

- **Endpoint**: PUT /categories/{id}
- **Request Body**:

```
{
  "name": "string",
  "description": "string"
}
```

- **Responses**:
    - 200 OK: Category updated successfully.
    - 400 Bad Request: Invalid input data.
    - 404 Not Found: Category not found.
    - 401 Unauthorized: Authentication required.
    - 403 Forbidden: Insufficient permissions.
    - 500 Internal Server Error: Server encountered an unexpected condition.

- **Delete a category by ID**
    - **Endpoint**: DELETE /categories/{id}
    - **Responses**:
        - 204 No Content: Category deleted successfully.
        - 404 Not Found: Category not found.
        - 401 Unauthorized: Authentication required.
        - 403 Forbidden: Insufficient permissions.
        - 500 Internal Server Error: Server encountered an unexpected condition.
- **List all categories**
    - **Endpoint**: GET /categories
    - **Query Parameters**:
        - name (optional): Filter by name (partial match)
        - page (optional): Page number for pagination
        - page_size (optional): Number of items per page for pagination
    - **Responses**:

```
{
  "total": "int",
  "page": "int",
  "page_size": "int",
  "categories": [
    {
      "id": "UUID",
      "name": "string",
      "description": "string"
    },
    ...
  ]
}
```

    - **Responses**:

- 200 OK: List of categories retrieved successfully.
- 400 Bad Request: Invalid query parameters.
- 401 Unauthorized: Authentication required.
- 403 Forbidden: Insufficient permissions.
- 500 Internal Server Error: Server encountered an unexpected condition.

Richardson Maturity Model

This API is designed to achieve Level 3 of the Richardson Maturity Model:

1. **Level 0**: The API uses standard HTTP methods (GET, POST, PUT, DELETE).
2. **Level 1**: The API is resource-based (e.g., /books, /authors, /categories).
3. **Level 2**: The API uses proper HTTP verbs and status codes for operations.
4. **Level 3**: The API uses hypermedia (HATEOAS) to provide information on available actions.

Authentication and Authorization

- **Method**: JWT (JSON Web Tokens)
  - Clients must provide a valid JWT token in the Authorization header using the Bearer schema: Authorization: Bearer <token>.
  - Endpoints will validate the JWT token to ensure the user is authenticated.
  - Authorization checks will be performed to ensure the user has the necessary permissions for the requested operation.

Error Handling

- **4xx Client Errors**:
  - 400 Bad Request: The request is malformed or contains invalid data.
  - 401 Unauthorized: The request requires user authentication.
  - 403 Forbidden: The server understands the request but refuses to authorize it.
  - 404 Not Found: The requested resource could not be found.
  - 409 Conflict: The request could not be completed due to a conflict with the current state of the target resource.

- **5xx Server Errors**:
    - 500 Internal Server Error: The server encountered an unexpected condition that prevented it from fulfilling the request.
    - 502 Bad Gateway: The server received an invalid response from the upstream server.
    - 503 Service Unavailable: The server is currently unable to handle the request due to temporary overload or maintenance.

Pagination

- All list endpoints support pagination through the page and page_size query parameters.
    - page: The page number to retrieve (default is 1).
    - page_size: The number of items per page (default is 10, maximum is 100).

Caching

- **GET requests** should be cached to improve performance:
    - **Cache-Control** header: public, max-age=3600 (1 hour)
    - Clients can use ETag headers for cache validation and conditional requests.
    - Cache invalidation occurs when the underlying data changes (e.g., book details are updated or a new book is added).