DESIGNING A REST API

Topic Selection and Justification

The library system is chosen for its educational value and relevance to a wide array of REST API design principles, including entity relationships, complex filtering, and the handling of various data types. This system not only involves basic CRUD operations but also advanced features like search, filtering, and relational data management, providing a rich learning experience in API design.

Entity Description

Books

- ID: Unique identifier for each book (Integer or UUID).
- Title: The title of the book (String).
- Author(s): References to Author entities (Array of IDs).
- ISBN: International Standard Book Number (String).
- Publication Year: Year of publication (Integer).
- Category: Reference to a Category entity (ID).
- Status: Current status of the book (Enum: Available, Checked Out, Reserved).

Authors

- ID: Unique identifier for each author (Integer or UUID).
- Name: Full name of the author (String).
- Biography: A short biography of the author (String, optional).
- Books Authored: List of books authored (Array of Book IDs).

Categories

- ID: Unique identifier for each category (Integer or UUID).
- Name: Name of the category (String).
- Description: A brief description of what types of books fall under this category (String, optional).

API Operations

CRUD Operations

- Create: POST request to `/entities` (e.g., `/books`) with entity details in the request body.
- Read: GET request to `/entities/{id}` for fetching a single entity or `/entities` for a list, with optional query parameters for filtering.
- Update: PUT or PATCH request to `/entities/{id}` with updated entity details in the request body.
- Delete: DELETE request to `/entities/{id}` to remove the entity.

Filtering, Sorting, and Pagination

- Filtering: Query parameters on list endpoints to filter results (e.g., `/books?category=Science`).
- Sorting: Query parameters for sorting (e.g., `/books?sort=publicationYear`).
- Pagination: Implement pagination through query parameters `page` and `limit` to manage large datasets (e.g., `/books?page=2&limit=10`).

REST API Design

Endpoint Naming and Methods

- Books: `GET /books`, `POST /books`, `GET /books/{id}`, `PUT /books/{id}`, `DELETE /books/{id}`
- Authors: Similar structure to books, with `/authors` as the base path.
- Categories: Similar structure to books, with `/categories` as the base path.

Request and Response Models

- Define clear JSON schemas for each operation, specifying required and optional fields, data types, and validation rules.

Status Codes and Error Handling

- Use standard HTTP status codes effectively to indicate success, failure, and errors in a self-descriptive manner.
- Provide error payloads with a consistent format, including an error code, message, and possibly hints to resolve the issue.

Authentication and Authorization

- Authentication: Implement OAuth 2.0 for secure API access, with different scopes for read-only and write operations.
- Authorization: Ensure that users can only access and manipulate resources they are entitled to, using roles and permissions.

Caching Strategy

- Use HTTP cache headers such as `ETag`, `If-None-Match`, and `Cache-Control` to optimize response times and reduce server load.