Book Store API Documentation

Welcome to the Book Store API! This API allows you to perform CRUD (Create, Read, Update, Delete) operations on a collection of books. It supports features such as pagination, searching, and retrieving statistics about the book collection.

**Table of contents**

1. Base URL
2. Endpoints
   1. Create a Book
   2. Get All Books
   3. Get a Book by ID
   4. Update a Book
   5. Delete a Book
   6. Search for Books
   7. Get Books Statistics
3. Pagination
4. CORS Policy

**Note:** The project uses connection to the remote mongodb server so database dump is not needed to run the project locally. Following the instructions in the README should run the project.

**Base URL**

The base URL for all API requests is: [https://localhost:5555/api/books](https://your-api-domain.com/api/books)

**Endpoints**

1. **Create a Book:**

**Endpoint**: POST /

Create a new book in the database.

**Request:**

{

"title": "The Great Gatsby",

"author": "F. Scott Fitzgerald",

"publicationYear": 1925

}

**Response**:

{

"title": "The Great Gatsby",

"author": "F. Scott Fitzgerald",

"publicationYear": 1925,

"createdAt": "2022-03-01T12:00:00.000Z",

"updatedAt": "2022-03-01T12:00:00.000Z"

}

**Error Codes:**

* 400 Bad Request: Required fields (title, author, publicationYear) are missing in the request body.
* 500 Internal Server Error: An unexpected server error occurred.

1. **Get All Books**

**Endpoint:** GET /

Retrieve a paginated list of all books.

**Query Parameters:**

page (optional): Page number (default is 1).

limit (optional): Number of books per page (default is 10).

**Response:**

{

"totalBooks": 30,

"totalPages": 3,

"count": 10,

"data": [

{

"title": "The Great Gatsby",

"author": "F. Scott Fitzgerald",

"publicationYear": 1925,

"createdAt": "2022-03-01T12:00:00.000Z",

"updatedAt": "2022-03-01T12:00:00.000Z"

},

// ... (more books)

],

"next": {

"page": 2,

"limit": 10

},

"previous": null

}

**Error Codes**:

* 500 Internal Server Error: An unexpected server error occurred.

1. **Get a Book by ID**

**Endpoint:** GET /:id

Retrieve details of a specific book by its ID.

**Response:**

{

"title": "The Great Gatsby",

"author": "F. Scott Fitzgerald",

"publicationYear": 1925,

"createdAt": "2022-03-01T12:00:00.000Z",

"updatedAt": "2022-03-01T14:30:00.000Z"

}

**Error Codes:**

* 404 Not Found: Book with the specified ID is not found.
* 500 Internal Server Error: An unexpected server error occurred.

1. **Update a Book**

**Endpoint:** PUT /:id

Update details of a specific book by its ID.

**Request:**

{

"title": "New Title",

"author": "New Author",

"publicationYear": 2022

}

**Response:**

{

"message": "Book updated successfully",

"data": {

"title": "New Title",

"author": "New Author",

"publicationYear": 2022,

"createdAt": "2022-03-01T12:00:00.000Z",

"updatedAt": "2022-03-01T15:00:00.000Z"

}

}

**Error Codes:**

* 400 Bad Request: Required fields (title, author, publicationYear) are missing in the request body.
* 404 Not Found: Book with the specified ID is not found.
* 500 Internal Server Error: An unexpected server error occurred.

1. **Delete a Book**

**Endpoint: DELETE /:id**

Delete a specific book by its ID.

**Response:**

{

"message": "Book deleted successfully"

}

**Error Codes:**

* 404 Not Found: Book with the specified ID is not found.
* 500 Internal Server Error: An unexpected server error occurred.

1. **Search for Books**

**Endpoint:** GET /search?q={query}

Search for books by title or author.

**Response:**

{

"count": 2,

"data": [

{

"title": "The Great Gatsby",

"author": "F. Scott Fitzgerald",

"publicationYear": 1925,

"createdAt": "2022-03-01T12:00:00.000Z",

"updatedAt": "2022-03-01T12:00:00.000Z"

},

{

"title": "Another Book",

"author": "Author Name",

"publicationYear": 2020,

"createdAt": "2022-03-01T14:00:00.000Z",

"updatedAt": "2022-03-01T14:00:00.000Z"

}

]

}

**Error Codes:**

* 400 Bad Request: Search query is missing.
* 500 Internal Server Error: An unexpected server error occurred.

1. **Get Books Statistics**

**Endpoint:** GET /stats

Retrieve statistics about the collection of books.

**Response:**

{

"totalBooks": [{ "count": 30 }],

"earliestAndLatestYear": [{ "earliestPublicationYear": 1925, "latestPublicationYear": 2022 }],

"booksByAuthor": [

{ "\_id": "Author 1", "count": 15 },

{ "\_id": "Author 2", "count": 10 },

// ... (more authors)

],

"booksByYear": [

{ "\_id": 2010, "count": 5 },

{ "\_id": 2015, "count": 10 },

// ... (more years)

],

"topAuthors": [

{ "\_id": "Top Author 1", "count": 10 },

{ "\_id": "Top Author 2", "count": 8 },

// ... (more top authors)

]

}

**Error Codes:**

* 500 Internal Server Error: An unexpected server error occurred.

**Pagination**

For endpoints that support pagination (GET /), you can use the page and limit query parameters to navigate through the collection of books.

Example: GET /?page=2&limit=15

**Additional features in the project:**

**CORS Policy**

In the project, a CORS (Cross-Origin Resource Sharing) policy has been implemented to allow or restrict access to resources from different origins. This enhances security by controlling which domains can access the API, helping to prevent unauthorized cross-origin requests.

**Employed $facet for low latency response**

In the implementation of the "Get Books Statistics" API, the MongoDB aggregation framework's $facet stage has been utilized. This optimization strategy enhances performance by allowing parallel execution of multiple aggregations, thereby reducing the number of database queries and improving overall response time. The $facet stage efficiently computes various book statistics concurrently, contributing to a more responsive and optimized API.