Lab Exercise 02: Building a Book Management API

Objective:

This lab exercise is designed to help students practice using HttpPost, HttpPut, HttpPatch, and HttpDelete operations in a Web API. Students will build a simple API to manage a collection of books, authors, and categories. The API will allow users to perform CRUD (Create, Read, Update, Delete) operations on these entities.

Scenario:

You are developing an online book store management system. The system allows administrators to manage the store's inventory, including books, authors, and categories. Your task is to implement the API endpoints required for managing these entities.

Models:

1. **Book**
    - BookId: int (Primary Key)
    - Title: string
    - AuthorId: int (Foreign Key)
    - CategoryId: int (Foreign Key)
    - PublishedYear: int
    - Price: decimal
2. **Author**
    - AuthorId: int (Primary Key)
    - Name: string
    - Bio: string
3. **Category**
    - CategoryId: int (Primary Key)
    - CategoryName: string
    - Description: string

Tasks:

1. **Creating a New Book (HttpPost)**
    - Implement an API endpoint to add a new book to the system.
    - URL: /api/books
    - Method: POST

- Request Body:

```
{
  "title": "Book Title",
  "authorId": 1,
  "categoryId": 2,
  "publishedYear": 2020,
  "price": 19.99
}
```

- Response: Returns the created book with its BookId.

2. **Updating a Book (HttpPut)**
    - Implement an API endpoint to update an existing book's details entirely.
    - URL: /api/books/{id}
    - Method: PUT
    - Request Body:

```
{
  "title": "Updated Book Title",
  "authorId": 1,
  "categoryId": 2,
  "publishedYear": 2021,
  "price": 24.99
}
```

    - Response: Returns the updated book.
3. **Partially Updating a Book's Price (HttpPatch)**
    - Implement an API endpoint to partially update a book's price.
    - URL: /api/books/{id}
    - Method: PATCH
    - Request Body:

```
{
  "price": 22.99
}
```

    - Response: Returns the updated book with the new price.

4. **Deleting a Book (HttpDelete)**
    - Implement an API endpoint to delete a book from the system.
    - URL: /api/books/{id}
    - Method: DELETE
    - Response: Returns a success message or the deleted book's details.
5. **Creating a New Author (HttpPost)**
    - Implement an API endpoint to add a new author.
    - URL: /api/authors
    - Method: POST
    - Request Body:

```
{
  "name": "Author Name",
  "bio": "Short bio of the author."
}
```

    - Response: Returns the created author with their AuthorId.
6. **Updating a Category (HttpPut)**
    - Implement an API endpoint to update an entire category's details.
    - URL: /api/categories/{id}
    - Method: PUT
    - Request Body:

```
{
  "categoryName": "Updated Category Name",
  "description": "Updated description."
}
```

    - Response: Returns the updated category.
7. **Partially Updating an Author's Bio (HttpPatch)**
    - Implement an API endpoint to partially update an author’s bio.
    - URL: /api/authors/{id}
    - Method: PATCH
    - Request Body:

```
{
```

"bio": "Updated bio for the author."

}

- Response: Returns the updated author with the new bio.

8. **Deleting a Category (HttpDelete)**
    - Implement an API endpoint to delete a category from the system.
    - URL: /api/categories/{id}
    - Method: DELETE
    - Response: Returns a success message or the deleted category's details.

Requirements:

- Use appropriate HTTP status codes for each operation.
- Validate input data and handle errors gracefully.
- Ensure that HttpPut replaces the entire entity, while HttpPatch only updates the specified fields.
- Return meaningful responses for each operation.

Bonus:

- Implement pagination and filtering for the list of books.
- Implement search functionality for books by title or author.

Submission:

Submit the completed API code along with a Postman collection demonstrating the usage of each endpoint.

This exercise will help you gain practical experience with CRUD operations using different HTTP methods in a real-world context.