**Online Bookstore API Documentation**

**Overview**

This documentation outlines the development of an online bookstore solution using C, ASP.NET, and RESTful APIs. The solution allows users to browse books, add them to a cart, place orders, and check order status. It consists of two main parts: Books API and Orders API.

**Books API Description**

The Books API provides endpoints to fetch book information, including a list of books and details of a specific book. Also, I added endpoints to Post, Put and Delete books.

**Endpoints**

1. GET /api/books

- Description: Fetches a list of all available books.

- Example Usage: ‘/api/books’

2. GET /api/books/{id}

- Description: Fetches details of a specific book by its ID.

- Example Usage: ‘/api/books/1’ (to fetch details of book with ID 1)

3. POST /api/books

- Description: Adds details of a specific book while its ID is auto generated.

- Example Usage: ‘/api/books/’ then add the values in JSON format

{

"title": "string",

"author": "string",

"quantityInStock": 0,

"price": 0

}

- Response: Returns status code 200, book details, including the ID.

**Orders API Description**

The Orders API provides endpoints for placing orders and checking order status. It also includes logic for managing book stock.

**Endpoints**

1. GET /api/order

- Description: Fetches a list of all available orders in the list.

- Example Usage: ‘/api/order’

2. POST /api/order

- Description: Places an order for one or more books specified by their IDs.

- Example Usage: ‘/api/orders’

- Request Payload Example:

‘‘‘json

{

"bookIds": [1, 2, 3]

}

‘‘‘

- Response: Returns status code 200, order details, including the order ID, book details, total price, and order date.

3. GET /api/order/{id}

- Description: Fetches the status of a specific order by its ID.

- Example Usage: ‘/api/order/1’ (to fetch details of order with ID 1)

- Response: Returns order details or a 404 Not Found response if the order does not exist.

**Error Handling**

- Proper error handling was not implemented in both APIs, including status codes and informative error messages.

- Common errors include 400 Bad Request for invalid requests, 404 Not Found for resources that do not exist, and 500 Internal Server Error for unexpected server issues.

**Data Models**

**Book**

- ‘Id’ (int): Unique identifier for the book.

- ‘Title’ (string): Title of the book.

- ‘Author’ (string): Author of the book.

- ‘QuantityInStock’ (int): Quantity of available books in stock.

- ‘Price’ (decimal): Price of the book.

**Order**

- ‘Id’ (int): Unique identifier for the order.

- ‘Books’ (List<Book>): List of books in the order.

- ‘Quantity’ (int): The amount of books to buy.

- ‘TotalPrice’ (decimal): Total price of the order.

- ‘OrderDate’ (DateTime): Date and time when the order was placed.

**Security and Authentication** (Not Included)

- Due to my current skill I couldn’t implement authentication and authorization mechanisms to secure the API.

- Use tokens or other authentication methods to protect sensitive operations.

**Database Integration** (Not Included)

- With time I would be able to integrate the APIs with a database to store and manage book and order data.

- The use of Entity Framework Core or other ORM tools for database interactions.

**Conclusion**

This documentation provides an overview of the Online Bookstore solution, including the Books API and Orders API, data models, error handling, and important considerations for a production-ready application. Further development and enhancements can be made to meet specific requirements and business needs.