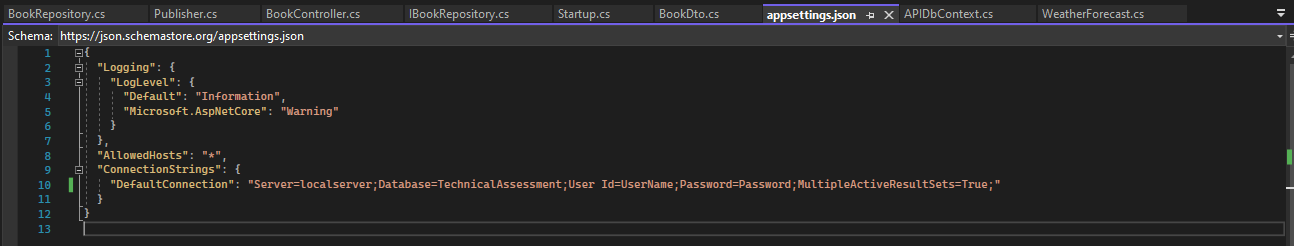
**Technical Assessment**

GitHub link:

Solution Explanation

Swagger URL: <https://localhost:7252/swagger/index.html>

* Replace the Servername, Database, User ID and Password and make sure the changes in the appsettings.json file 
* In startup.cs file done the configuration services and used dependency injection to inject the database connectivity.
* APIDBContext.cs inherit the DbContext class and creating the Dbcontextoptions and initialize the Dbset for Book,Author and publisher class and configured in startup.cs file based on dependency injection.
* Implemented the logger with the use of **serilog** and dependency injection promotes loose coupling between components in application. In Program.cs file it sets up Serilog, creates a host using CreateHostBuilder, and runs the host.
* Log of the each method execution will capture in the logger(date).txt file in the same solutionexplorer path.
* Please make the sure the sql connection and run the Migration
  + Tools -> NuGet Package Manager – Package Manager Console execute the below comments
    - **Add-Migration "Initial Create"** (can rename the initial create)
    - **Update-Database**
  + Please make the Database has been created with the tables Books,Authors,Publishers and the Migration folder created in solution explorer.

Models:

* Author.cs – AuthorId [key], LastName, FirstName
* Publisher.cs – PublisherId[Key], Name
* Book.cs - Id, Title, Price, PublisherId (foreign key), AuthorId (foreign key)

Repository:

* BookRepository involves the All implementation logics like fetching and storing the data in db and the method are implements from the IRepository interface
* Initialized the private readonly property for APIDbcontext and ILogger make use for the db interactions and capturing the log in text file.

Controller:

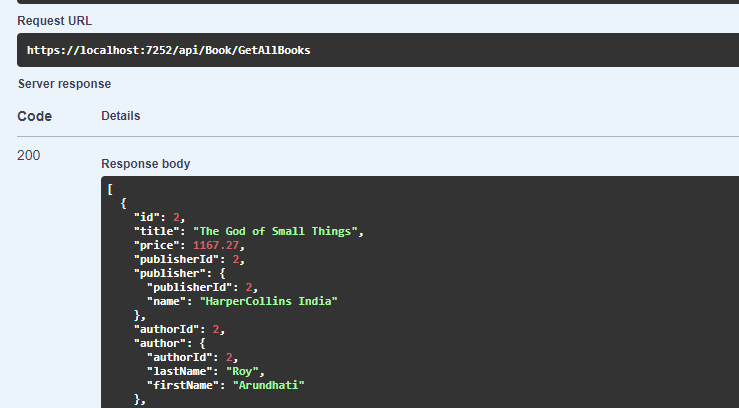
* BookController used apicontroller attribute and Route attribute defined the routing information
* IBookRepository, ILogger<BookController> inject dependencies into the BookController

API Methods:

**GetAllBooks : [HTTPGET]**

It will retrieve the All Book information in the Booktable. searchParam,sortBy,sortOrder are the optional parameter. Searchparam used to search the book title, publisher name, author firstname and lastname.

sortBy is which field need to be sort and SortOrder would be descending or ascending.

Sample Output: 

**CreateBook: [HTTPPOST]**

This method insert a single set of data in Book, Author and Publisher table.

If the Author/Publisher details already exist it will retrieve the id and inset in Book table otherwise create a new data in table then retrieve the id and insert in Book table.

**Sample Input:**

{

"title": "The God of Small Things",

"price": 1167.27,

"publisher": {

"name": "HarperCollins India"

},

"author": {

"lastName": "Roy",

"firstName": "Arundhati"

}

}

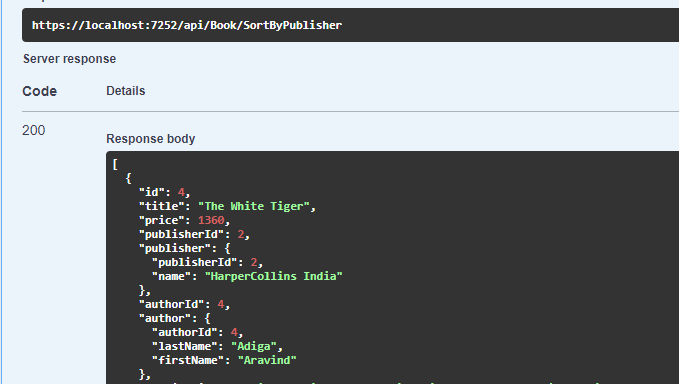
Sample Output: 

**Problem**

**1.Create a REST API using .Net Core MVC and write a method to return a sorted list of these by Publisher, Author (last, first), then title.**

Use SortByPublisher method (HTTPGET) to sort the data by publisher, Author lastname,firstname and title

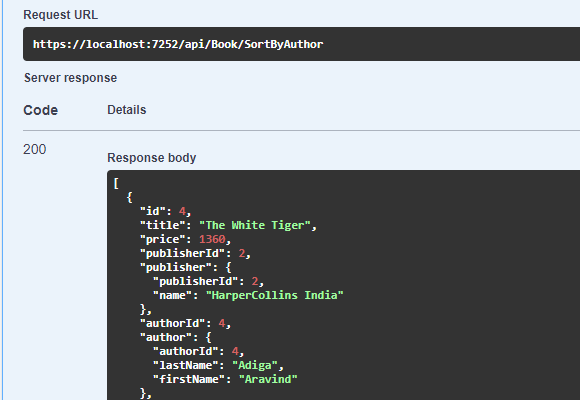
Request URL: <https://localhost:7252/api/Book/SortByPublisher>

Sample Output: 

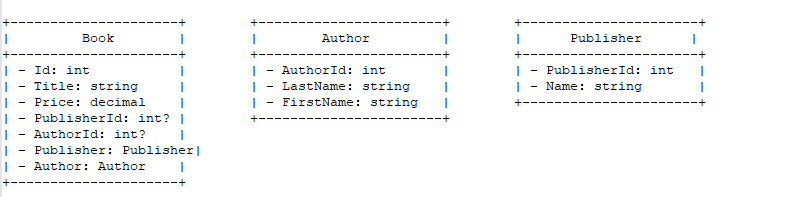
**2. Write another API method to return a sorted list by Author (last, first) then title.**

Use SortByAuthor method (HTTPGET) to sort the data by Author lastname,firstname and title

Request URL: <https://localhost:7252/api/Book/SortByAuthor>

Sample Output: 

**3. If you had to create one or more tables to store the Book data in a SQL Server/Sql Lite database, outline the table design along with fields and their datatypes**.



Normalization is a process in database design that organizes tables and relationships to minimize redundancy and dependency. In this case, the classes Book, Author, and Publisher represent entities in a database.

First Normal Form (1NF): All attributes in each table are atomic (indivisible). The provided classes meet this criterion.

Second Normal Form (2NF): No partial dependencies on the primary key. It appears that the classes are already in 2NF, as there are no partial dependencies on the primary keys.

Third Normal Form (3NF): No transitive dependencies. The classes seem to be in 3NF as well, with attributes depending only on the primary key.

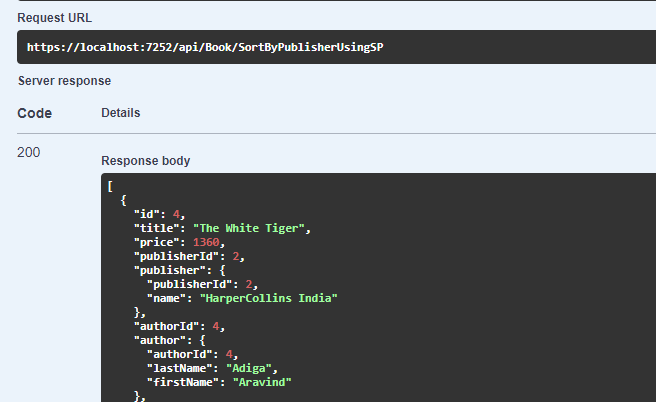
The relationships between Book, Author, and Publisher are established through foreign keys (AuthorId and PublisherId). This structure helps in maintaining data integrity and avoids redundancy.

**4. Write stored procedures for steps 1 and 2, and use them in separate API methods to return the same results.**

Kindly execute the **GetBooksSortedByPublisherAuthorTitle.sql** and **GetBooksSortedByAuthorTitle.sql** file stored procedure the in database.

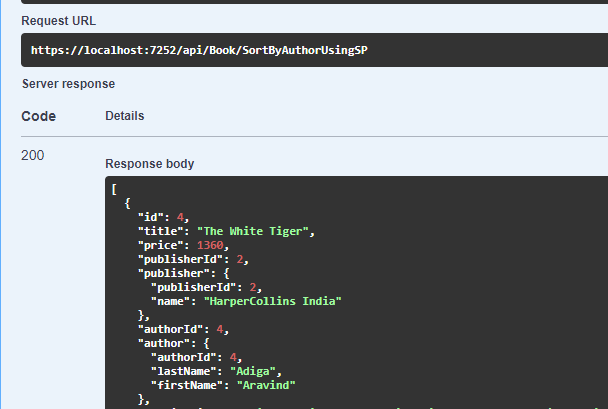
Use SortByPublisherUsingSP [HTTPGET] – to sort the data by publisher, Author lastname,firstname and title

Request URL: <https://localhost:7252/api/Book/SortByPublisherUsingSP>

Sample Output: 

Use SortByAuthorUsingSP [HTTPGET] – to sort the data by Author lastname,firstname and title

Request URL: <https://localhost:7252/api/Book/SortByAuthorUsingSP>

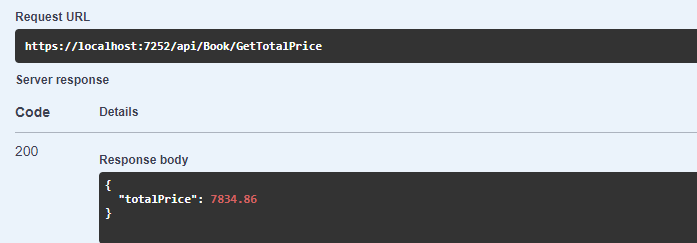
Sample Output: 

**5. Write an API method to return the total price of all books in the database.**

Use GetTotalPrice [HTTPGET] – to return the total price in db.

Request URL: <https://localhost:7252/api/Book/GetTotalPrice>

Sample Output:



**6. If you have a large list of these in memory and want to save the entire list to the database, with only one call to the DB server.**

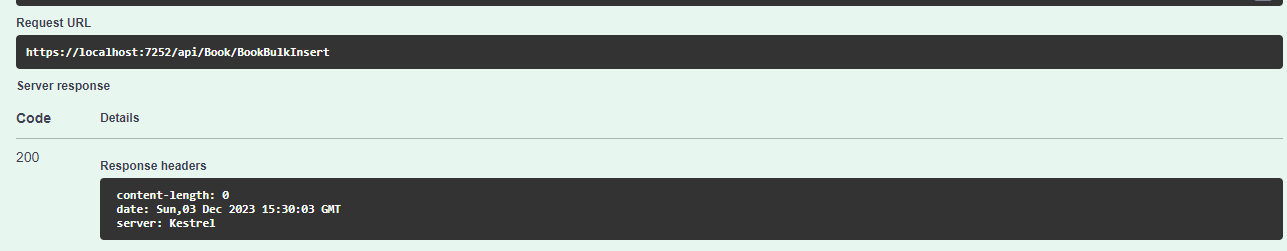
Use BookBulkInsert [HTTPPOST] – to insert the bulk data in db. In this If the Author/Publisher details already exist it will retrieve the id and inset in Book table otherwise create a new data in table then retrieve the id and insert in Book table and AddRange used to insert the data in db.

Request URL: <https://localhost:7252/api/Book/BookBulkInsert>

Sample Input:

Pls refer the BookBulkInsert.txt file for the sample Payload

Sample Output:



**7. Add a property to the Book class that outputs the MLA (Modern Language Association) style citation as a string (https://images.app.goo.gl/YkFgbSGiPmie9GgWA). Please add whatever additional properties the Book class needs to generate the citation.**

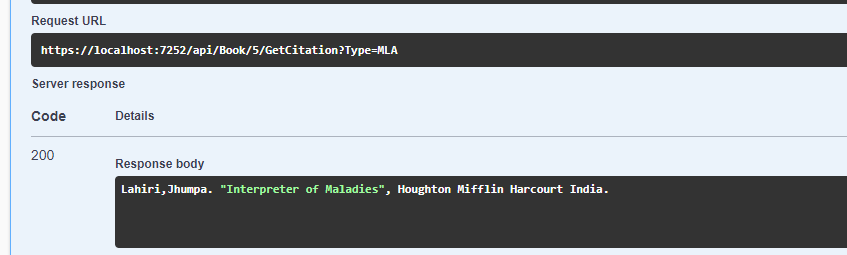
MLA Citation:

Request URL : <https://localhost:7252/api/Book/5/GetCitation?Type=MLA>

Sample Input:

Id = 5 // book id

Type = MLA // for MLA citation

Sample Output: 

**8. Add another property to generate a Chicago style citation (Chicago Manual of Style)** (<https://images.app.goo.gl/w3SRpg2ZFsXewdAj7>).

Chicago Citation:

Request URL : <https://localhost:7252/api/Book/5/GetCitation?Type=Chicago>

Sample Input:

Id = 5 // book id

Type = Chicago// for Chicago citation

Sample Output: 