**Day 13**
**Dharunya S**

**Create a backup commit of the existing book and sales details project.**
**- Move the queries of the book store from the controller into the Service using Dependency injection**

**BookController.cs**

```
using Day12.Models;
using Day12.Services;
using Microsoft.AspNetCore.Mvc;

namespace Day12.Controllers
{
   [ApiController]
   [Route("api/[controller]")]
   public class BookController : ControllerBase
   {
       IBookService _bookService;

      public BookController(IBookService bookService)
      {
         _bookService = bookService;
      }

      [HttpGet("GetAllBooks")]
      public IActionResult GetAllBooks()
      {
         return Ok(_bookService.GetAllBooks());
      }

      [HttpPost("AddAuthor")]
      public IActionResult AddAuthor([FromBody] Author author)
      {
         _bookService.AddAuthor(author);
         return Ok("Author added successfully");
      }

      [HttpPost("AddNewBook")]
      public IActionResult AddNewBook([FromBody] BookAndAuthorEntry
bookAndAuthorEntry)
      {
```

```
    _bookService.AddNewBook(bookAndAuthorEntry);
    return Ok("Book added successfully");
}

[HttpGet("FetchAllNewBooks")]
public IActionResult FetchAllNewBooks()
{
    return Ok(_bookService.FetchAllNewBooks());
}

[HttpGet("FetchBookByAuthor")]
public IActionResult FetchBookByAuthor([FromQuery] string authorName)
{
    return Ok(_bookService.FetchBookByAuthor(authorName));
}

[HttpGet("GetTotalsalesbyAuthor")]
public IActionResult GetTotalsalesbyAuthor([FromQuery] string authorName)
{
    return Ok(_bookService.GetTotalsalesbyAuthor(authorName));
}

[HttpPost("InsertSales")]
public IActionResult AddSalesInfo([FromBody] SalesEntry entry)
{
    _bookService.AddSalesInfo(entry);
    return Ok("Sales info added successfully");
}

[HttpGet("GetSalesHistory")]
public IActionResult GetSalesHistory([FromQuery] string autname)
{
    return Ok(_bookService.GetSalesHistory(autname));
}

[HttpGet("GetCountSalesHistoryByYear")]
public IActionResult GetCountSalesHistoryByYear([FromQuery] int year)
{
    return Ok(_bookService.GetCountSalesHistoryByYear(year));
}

[HttpGet("GetDetailsbyname")]
public IActionResult GetDetailsbyname([FromQuery] string autname)
{
```

```
        return Ok(_bookService.GetDetailsbyname(autname));
    }

    [HttpPost("InsertBook")]
    public IActionResult AddBook([FromBody] Book book)
    {
        _bookService.AddBook(book);
        return Ok("Book added to the store successfully");
    }

    [HttpGet("GetBook")]
    public IActionResult GetBook()
    {
        return Ok(_bookService.GetBook());
    }

    [HttpDelete("DeleteBook/{id}")]
    public IActionResult DeleteBook(int id)
    {
        _bookService.DeleteBook(id);
        return Ok("Deleted successfully");
    }

    [HttpGet("GetBookOrderedbyprice")]
    public IActionResult GetBookOrderedbyprice([FromQuery] bool status)
    {
        return Ok(_bookService.GetBookOrderedbyprice(status));
    }

    [HttpGet("GetTop5Book")]
    public IActionResult GetTop5Book()
    {
        return Ok(_bookService.GetTop5Book());
    }

    [HttpGet("GetCostDetails")]
    public IActionResult GetCostDetails()
    {
        return Ok(_bookService.GetCostDetails());
    }
  }
}
```

**IBookservice**

```
using Day12.Models;

namespace Day12.Services
{
   public interface IBookService
   {
       public IEnumerable<Book> GetAllBooks();
      public void AddAuthor(Author author);
      public void AddNewBook(BookAndAuthorEntry bookAndAuthorEntry);
      public IEnumerable<NewBook> FetchAllNewBooks();
      public IEnumerable<object> FetchBookByAuthor(string authorName);
      public IEnumerable<object> GetTotalsalesbyAuthor(string authorName);
      public void AddSalesInfo(SalesEntry entry);
      public IEnumerable<object> GetSalesHistory(string autname);
      public int GetCountSalesHistoryByYear(int year);
      public IEnumerable<object> GetDetailsbyname(string autname);
      public void AddBook(Book book);
      public IEnumerable<Book> GetBook();
      public void DeleteBook(int Id);
      public IEnumerable<Book> GetBookOrderedbyprice(bool status);
      public IEnumerable<Book> GetTop5Book();
      public object GetCostDetails();
   }
}
```

BookService.cs

```
using Day12.context;
using Day12.Models;
using Microsoft.EntityFrameworkCore;

namespace Day12.Services
{
   public class BookService : IBookService
   {
      private readonly MyAppDbContext appDbContext;

      public BookService(MyAppDbContext ctx)
      {
         appDbContext = ctx;
      }

      public IEnumerable<NewBook> FetchAllNewBooks()
      {
         return appDbContext.NewBooks.Include(x => x.Author).ToList();
```

```
}

public IEnumerable<object> FetchBookByAuthor(string authorName)
{
   return appDbContext.NewBooks.Include(x => x.Author)
      .Where(x => x.Author.AuthorName == authorName)
      .Select(y => new
      {
         Title = y.Title,
         Rate = y.Price,
         Authorname = y.Author.AuthorName
      })
      .ToList();
}

public IEnumerable<object> GetTotalsalesbyAuthor(string authorName)
{
   return appDbContext.NewBooks.Include(x => x.Author)
      .Where(x => x.Author.AuthorName == authorName)
      .Select(y => new
      {
         Title = y.Title,
         sales = y.sales,
         profit = (y.sales * y.Price),
         Authorname = y.Author.AuthorName
      })
      .ToList();
}

public IEnumerable<object> GetSalesHistory(string autname)
{
   return appDbContext.TotSales
      .Where(x => x.Book.Author.AuthorName == autname)
      .Select(s => new
      {
         BookName = s.Book.Title,
         AuthorName = s.Book.Author.AuthorName,
         No_of_copies = s.No_of_copies,
         Year = s.year,
         Price = s.Book.Price,
         TotalSales = s.No_of_copies * s.Book.Price
      })
      .ToList();
}
```

```
public int GetCountSalesHistoryByYear(int year)
{
   return appDbContext.TotSales
      .Where(x => x.year == year)
      .Count();
}

public IEnumerable<object> GetDetailsbyname(string autname)
{
   var sales = from s in appDbContext.TotSales
               join b in appDbContext.NewBooks on s.BookId equals b.NewBookId
               join a in appDbContext.Authors on b.AuthorId equals a.AuthorId
               where a.AuthorName == autname
               select new
               {
                  BookName = b.Title,
                  AuthorName = a.AuthorName,
                  No_of_copies = s.No_of_copies,
                  Year = s.year,
                  Price = b.Price,
                  TotalSales = s.No_of_copies * b.Price
               };
   return sales.ToList();
}

public IEnumerable<Book> GetBook()
{
   return appDbContext.Books.ToList();
}

public IEnumerable<Book> GetBookOrderedbyprice(bool status)
{
   return status
      ? appDbContext.Books.OrderBy(x => x.Price).ToList()
      : appDbContext.Books.OrderByDescending(x => x.Price).ToList();
}

public IEnumerable<Book> GetTop5Book()
{
   return appDbContext.Books.OrderByDescending(x => x.Rating).Take(5).ToList();
}

public object GetCostDetails()
```

```
{
    var books = appDbContext.Books.ToList();
    var total_cost = books.Sum(x => x.Price);
    var average_cost = books.Average(x => x.Price);

    return new { TotalCost = total_cost, AverageCost = average_cost };
}

public void AddAuthor(Author author)
{
    appDbContext.Authors.Add(author);
    appDbContext.SaveChanges();
}

public void AddNewBook(BookAndAuthorEntry bookAndAuthorEntry)
{
    var author = appDbContext.Authors
        .FirstOrDefault(x => x.AuthorName == bookAndAuthorEntry.AuthorName);

    if (author == null)
        throw new Exception("Author not found. Please add the author first.");

    NewBook newBook = new NewBook
    {
        Title = bookAndAuthorEntry.BookTitle,
        Price = bookAndAuthorEntry.Price,
        sales = bookAndAuthorEntry.sales,
        AuthorId = author.AuthorId
    };
    appDbContext.NewBooks.Add(newBook);
    appDbContext.SaveChanges();
}

public void AddSalesInfo(SalesEntry entry)
{
    var book_to_be_inserted = appDbContext.NewBooks
        .FirstOrDefault(x => x.Title == entry.Bookname);

    if (book_to_be_inserted == null)
        throw new Exception("Book not found. Please add the book first.");

    Sales sales = new Sales
    {
        BookId = book_to_be_inserted.NewBookId,
```

```
            No_of_copies = entry.No_of_copies,
            year = entry.year
        };
        appDbContext.TotSales.Add(sales);
        appDbContext.SaveChanges();
    }

    public IEnumerable<Book> GetAllBooks()
    {
        return appDbContext.Books.ToList();
    }

    public void AddBook(Book book)
    {
        appDbContext.Books.Add(book);
        appDbContext.SaveChanges();
    }

    public void DeleteBook(int Id)
    {
        var book = appDbContext.Books.Find(Id);
        if (book != null)
        {
            appDbContext.Books.Remove(book);
            appDbContext.SaveChanges();
        }
    }
  }
}
```

```
curl -X 'GET' \
  'http://localhost:5100/api/Book/FetchAllNewBooks' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5100/api/Book/FetchAllNewBooks
```

Server response

| Code | Details |
| --- | --- |
| 200 | Response body |


```
[
  {
    "newBookId": 1,
    "title": "Harry potter",
    "price": 0,
    "sales": 0,
    "authorId": 1,
    "author": {
      "authorId": 1,
      "authorName": "Jk rowling"
    }
  },
  {
    "newBookId": 2,
    "title": "clean code",
    "price": 0,
    "sales": 0,
    "authorId": 2,
    "author": {
      "authorId": 2,
      "authorName": "Robert"
    }
  },
  {
    "newBookId": 3,
    "title": "The Pragmatic Programmer",
    "price": 200,
    "sales": 0,
```


Download