EASTERN INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTING AND INFORMATION TECHNOLOGY

Practice Assignment – Quarter 1, 2024-2025
Course Name: .Net Programming
Course Code: CSE 443
Student's Full Name:
Student ID:

Practice Assignment 4

LIBRARY MANAGEMENT SYSTEM

Following Practice Assignment 3, further develop the new layout and add model in the MVC framework

---

Exercise 1: Create a admin layout for library management system.

In Practice Assignment 3, there is a menu in area 1 in "Figure1". Add one item to that menu named 'Admin.' When clicking on the admin menu, it should navigate to the admin layout, which is basically similar to the following image:

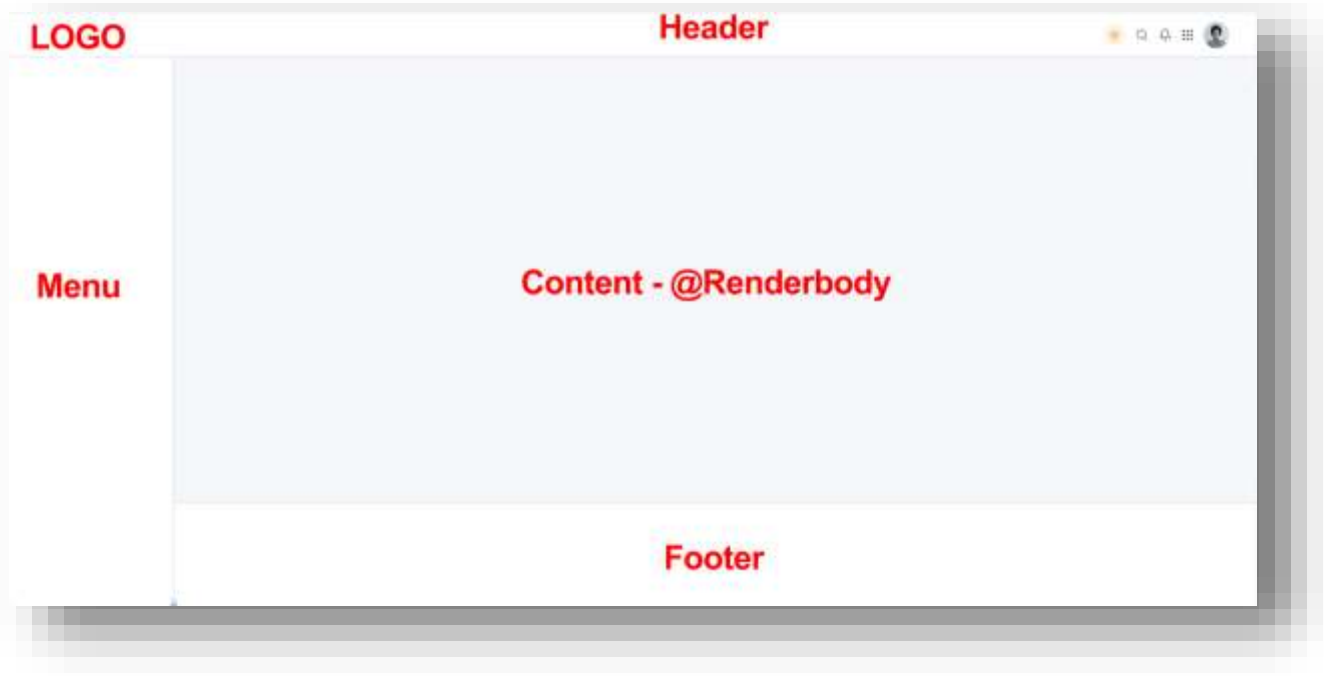

Figure 1: Admin layout

Requirements:

- Initialize sample data to display the menu.
- When selecting any item in the menu, it should navigate to the corresponding screen and render the content.
- Prepare sample data to be displayed on the interface.

Exercise 2:

Design a database using SQL Server and migrate it to the model in the MVC project with the tables listed below.

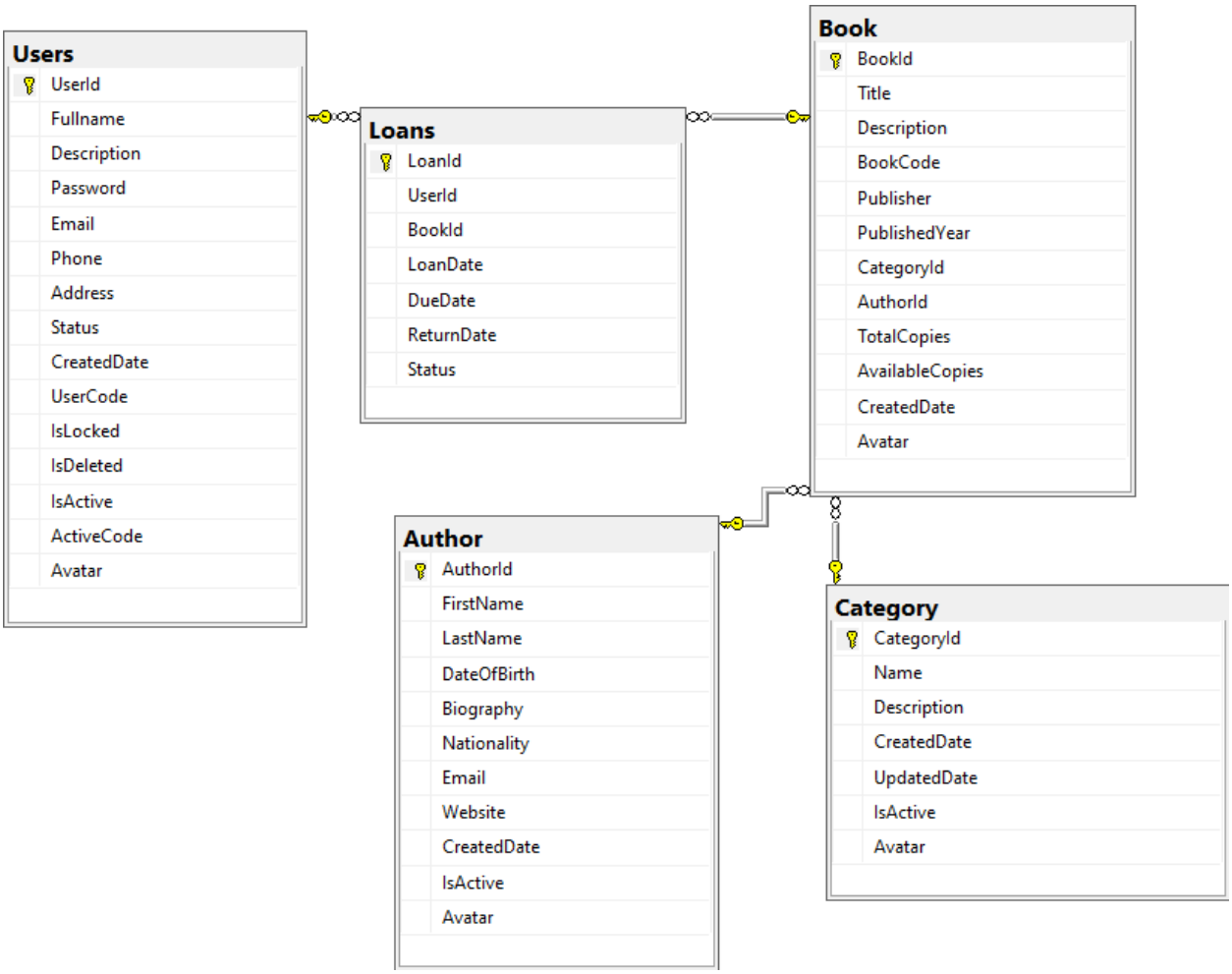

Figure 2: Diagram of the library management database

Users:

User			
#	**Col_Name**	**Data Type**	**Descriptions**
1	UserId	int	Unique identifier for each user, auto-incremented.
2	Fullname	nvarchar(200)	Full name of the user.
3	Description	nvarchar(MAX)	Additional info about the user, like interests or bio.
4	Password	nvarchar(MAX)	Hashed password for secure authentication.

5	Email	nvarchar(100)	User's email address, used for communication and verification.
6	Phone	nvarchar(20)	User's contact phone number.
7	Address	nvarchar(MAX)	User's physical address for mailing or location purposes.
8	Status	int	Represents the user's status.
9	CreatedDate	datetime	Date and time when the user account was created.
10	UserCode	nvarchar(MAX)	Unique code for internal identification of the user.
11	IsLocked	bit	Indicates if the account is locked (1) or active (0).
12	IsDeleted	bit	Marks the account as deleted (1) or active (0) for soft deletion.
13	IsActive	bit	Shows if the account is active (1) or inactive (0).
14	ActiveCode	nvarchar(MAX)	Code for account activation, sent via email during registration.
15	Avatar	nvarchar(MAX)	Avatar’s User - Local location in the server to get the picture.

Loans:

Loans			
#	**Col_Name**	**Data Type**	**Descriptions**
1	LoanId	int	(Primary Key, Auto-Increment)
2	UserId	int	References the user who borrowed the book.
3	BookId	int	References the borrowed book.
4	LoanDate	datetime	Date when the book was borrowed.
5	DueDate	datetime	Date when the book is due.
6	ReturnDate	datetime	Date when the book was returned.
7	Status	int	Status of the loan (e.g., "Active", "Returned", "Overdue"). 0: Active , 1: Returned , 2: Overdue

Books:

Books			
#	**Col_Name**	**Data Type**	**Descriptions**
1	BookId	int	(Primary Key, Auto-Increment)

2	Title	nvarchar(200)	Title of the book.
3	Description	nvarchar(MAX)	Description of the book.
4	BookCode	nvarchar(MAX)	Standard Book Number.
5	Publisher	nvarchar(MAX)	
6	PublishedYear	datetime	Year the book was published.
7	CategoryId	int	References the category.
8	AuthorId	int	References the author.
9	TotalCopies	int	Total number of physical copies of the book in the library
10	AvailableCopies	int	Total number of physical copies of the book currently in the library, excluding copies on loan
11	CreatedDate	datetime	Date when the book record was created.
12	Avatar	nvarchar(MAX)	Cover image of the book - Local location in the server to get the picture.
13	Pdf	nvarchar(MAX)	Store the version pdf for reading online

Authors:

Authors			
#	**Col_Name**	**Data Type**	**Descriptions**
1	AuthorId	int	(Primary Key, Auto-Increment). Unique identifier for each author.
2	FirstName	nvarchar(100)	Author's first name.
3	LastName	nvarchar(100)	Author's last name.
4	DateOfBirth	datetime	Author's date of birth.
5	Biography	nvarchar(MAX)	A short biography of the author.
6	Nationality	nvarchar(100)	Nationality of the author.
7	Email	nvarchar(100)	Author's Email.
8	Website	nvarchar(100)	Author's Website.
9	CreatedDate	datetime	Date when the author record was created.

10	IsActive	bit	The IsActive column helps indicate whether a record is currently active and usable within the application.
11	Avatar	nvarchar(MAX)	Authors' Avatar - Local location in the server to get the picture.

Categorys:

Categorys			
#	**Col_Name**	**Data Type**	**Descriptions**
1	CategoryId	int	Unique identifier for each category (Primary Key).
2	Name	nvarchar(MAX)	Name of the category; must be unique for identification.
3	Description	nvarchar(MAX)	Additional information about the category.
4	CreatedDate	datetime	Date and time when the category was created.
5	UpdatedDate	datetime	Date and time when the category was last updated.
6	IsActive	bit	Indicates if the category is active (1) or inactive (0).
7	Avatar	nvarchar(MAX)	Categorys's image - Local location in the server to get the picture.

Note:

Use Data Annotations for Validation:

- Implement data annotations such as `[Required]`, `[StringLength]`, `[EmailAddress]`, etc., to enforce validation rules on model properties.
- Ensure that validation attributes reflect business rules and requirements.

Implement Navigation Properties:

- Define relationships between models using navigation properties (e.g., `one-to-many`, `many-to-many`).
- Use collections to represent related entities (e.g., `ICollection<Product>` in a `Category` model).

Create a DbContext Class:

- Define a `DbContext` class that represents the session with the database.
- Include `DbSet<T>` properties for each model to facilitate data access.

Exercise 3:

Add a new table **"Carousel"** and make a model in project for displaying images or content on the homepage of your library management system with the following requirements:

- **Purpose**:
  - Display images or promotional content on the homepage.
- **Fields**:
  - **CarouselId**: `int` (Primary Key)
  - **ImageUrl**: `nvarchar(MAX)` (Required) - URL of the image.
  - **Title**: `nvarchar(200)` (Required) - Title for the item.
  - **Description**: `nvarchar(MAX)` (Optional) - Additional details.
  - **LinkUrl**: `nvarchar(MAX)` (Optional) - URL linked to the item.
  - **Order**: `int` (Required) - Display order of items.
  - **IsActive**: `bit` - Indicates if the item is active.
  - **CreatedDate**: `datetime` - Timestamp of creation.
  - **UpdatedDate**: `datetime` - Timestamp of last update.

In Practice Assignment 3, the Carousel currently loads static data. With this table, implement the business logic and load data dynamically from the database.

Guide:

To serve image files from the server in your .NET Core 8.0 MVC project, you can configure your application to serve static files from a specific directory on the server. Here's how to set it up:

Step 1: Store Images in a Server Directory

1. Decide on a directory where you'll store the images for the carousel. For example, let's use a folder named carousel_images inside the wwwroot folder of your project (e.g., wwwroot/carousel_images).
2. Place your image files in this directory.

** You can set anywhere in the server if you want to store the image !*

Step 2: Configure Static Files in Startup or Program.cs

In **ASP.NET Core 8.0**, static files are served by default from the wwwroot folder. You just need to make sure the UseStaticFiles middleware is added to your pipeline, which is typically already included.

In your Program.cs, you should have:

E: > Desktop > C# test.cs

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles(); // Ensures wwwroot is accessible for static files.

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

Step 3: Update the Model to Use Server Path for Images

In the Carousel model, set the ImageUrl to store relative paths, such as /carousel_images/image1.jpg.

```
[Required]
public string ImageUrl { get; set; } // Store as "/carousel_images/image1.jpg"
```

Step 4: Display the Image in the Carousel View

1. In the _Carousel.cshtml partial view, ensure the src attribute of the <img> tag is set to the relative path stored in ImageUrl.

```
<div id="carouselExample" class="carousel slide" data-bs-ride="carousel">
    <div class="carousel-inner">
        @foreach (var item in Model)
        {
            <div class="carousel-item @(item == Model.First() ? "active" : "")">
                <a href="@item.LinkUrl">
                    <img src="@item.ImageUrl" class="d-block w-100" alt="@item.Title">
                </a>
                <div class="carousel-caption d-none d-md-block">
                    <h5>@item.Title</h5>
                    <p>@item.Description</p>
                </div>
            </div>
        }
    </div>
    <button class="carousel-control-prev" type="button" data-bs-target="#carouselExample" data-bs-slide="prev">
        <span class="carousel-control-prev-icon" aria-hidden="true"></span>
        <span class="visually-hidden">Previous</span>
    </button>
    <button class="carousel-control-next" type="button" data-bs-target="#carouselExample" data-bs-slide="next">
        <span class="carousel-control-next-icon" aria-hidden="true"></span>
        <span class="visually-hidden">Next</span>
    </button>
</div>
```

Step 5: Access Images from the Server

When you run your application, ASP.NET Core will serve images directly from the wwwroot directory. So if your ImageUrl is /carousel_images/image1.jpg, it will be resolved to https://yourdomain.com/carousel_images/image1.jpg.