LAB 1
ASP.NET CORE MVC (P1)

❖ CONTENT

- Create an ASP.NET Core MVC project
- Setup database connection & implement Authentication (login/logout, register)
- Create tables with ORM technique
- Implement CRUD features with Scaffolding technique
- Customize web layout (template) & identity pages
- Set default role for user registration
- Seed (populate) initial data to database
- Setup Authorization (role-based access) for different features

❖ INSTRUCTION

1. Open Visual Studio and create new ASP.NET Core MVC web app

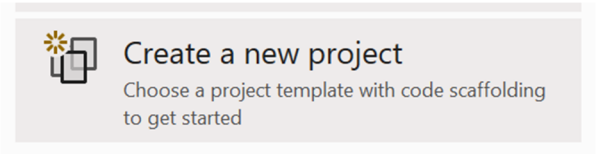


Figure 1 - Create new project in Visual Studio

2. Config ASP.NET Core project

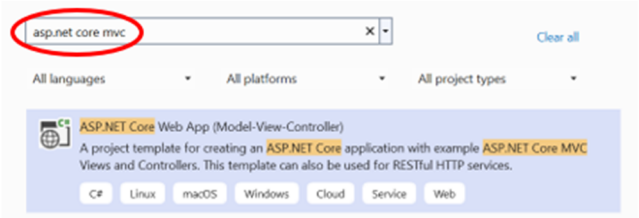


Figure 2 - Search & select ASP.NET Core Web App (MVC)

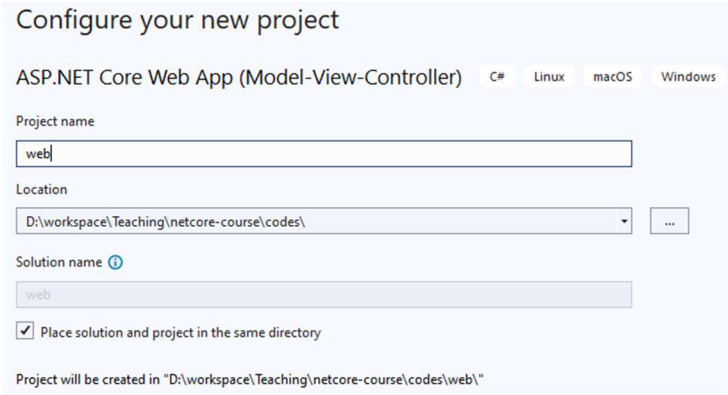


Figure 3 – Set project location, solution & project name

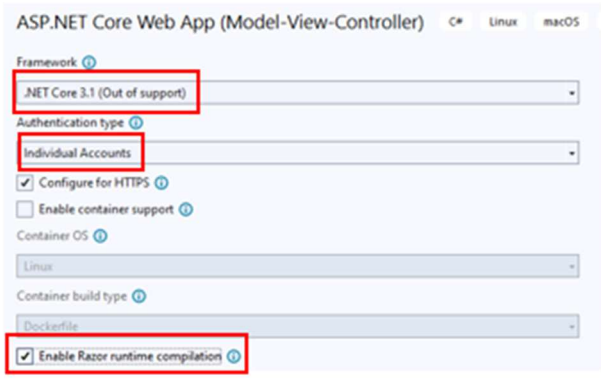


Figure 4 – Select framework version & authentication type

3. Add model to automatically create table

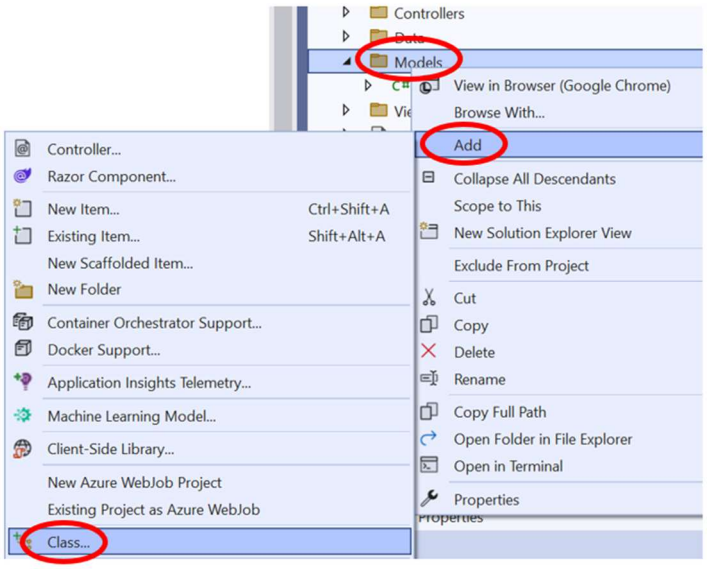


Figure 5 - Add new Model (class)

```
public class Book
{
    public int BookId { get; set; }
    public string BookTitle { get; set; }
    public double BookPrice { get; set; }
    public string BookCover { get; set; }
}
```

*Figure 6 – **Models/Book.cs***

4. Generate Controller with Views for CRUD features with Scaffolding technique

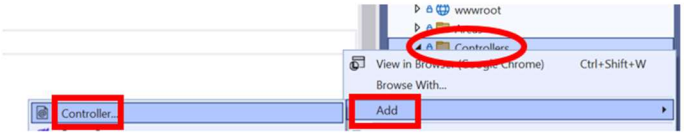


Figure 7 - Add new controller (class)

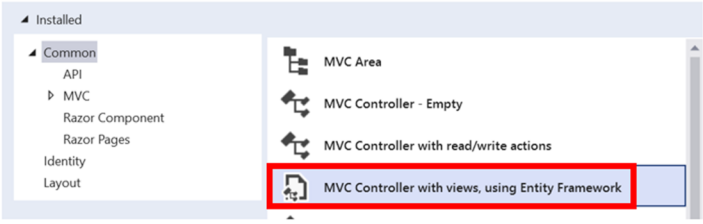


Figure 8 - Select MVC Controller with views

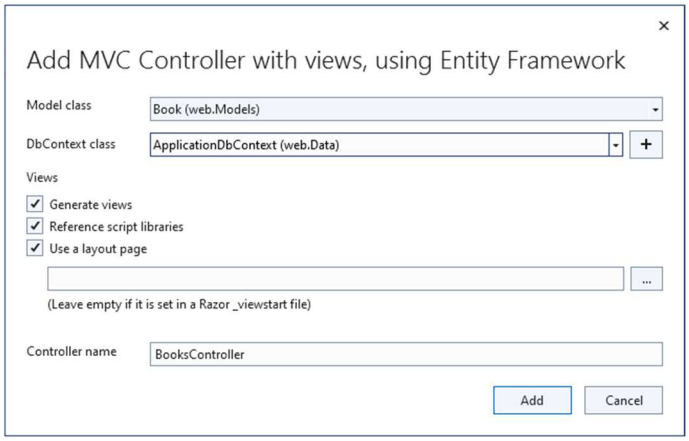


Figure 9 - Select Model class & DbContext class

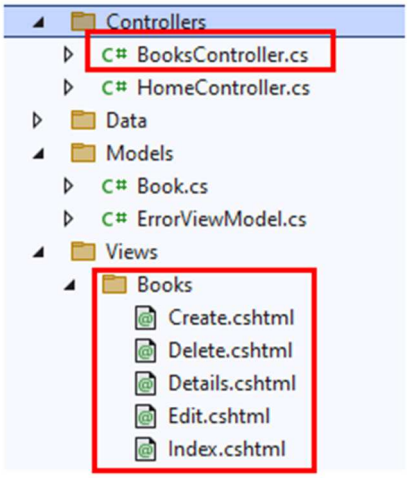


Figure 10 - Controller & Views have been generated for selected Model

5. Override Identity pages to customize view of Authentication (Login, Register,…)

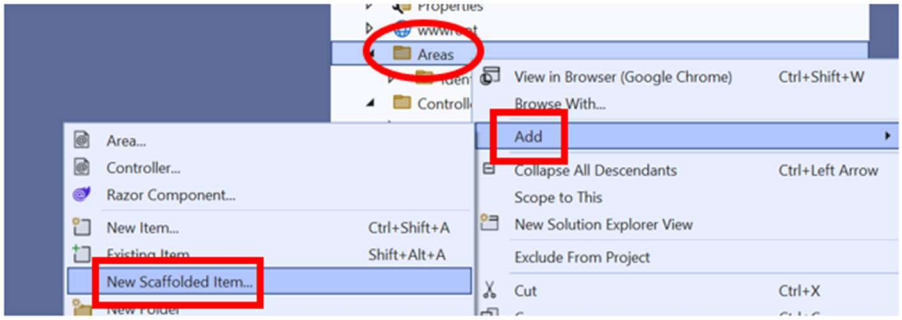


Figure 11 - Add new Scaffolded Item

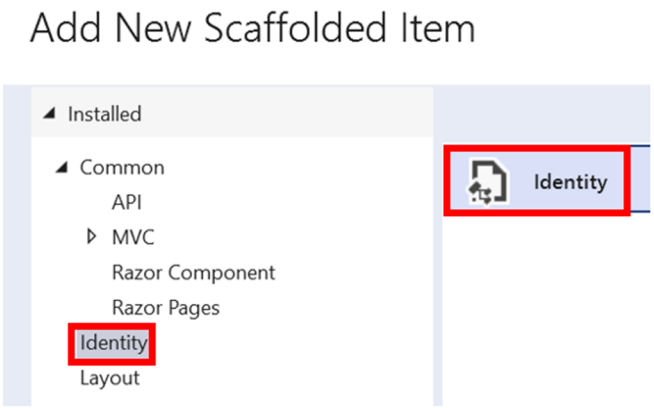


*Figure 12 - Select option **Identity***

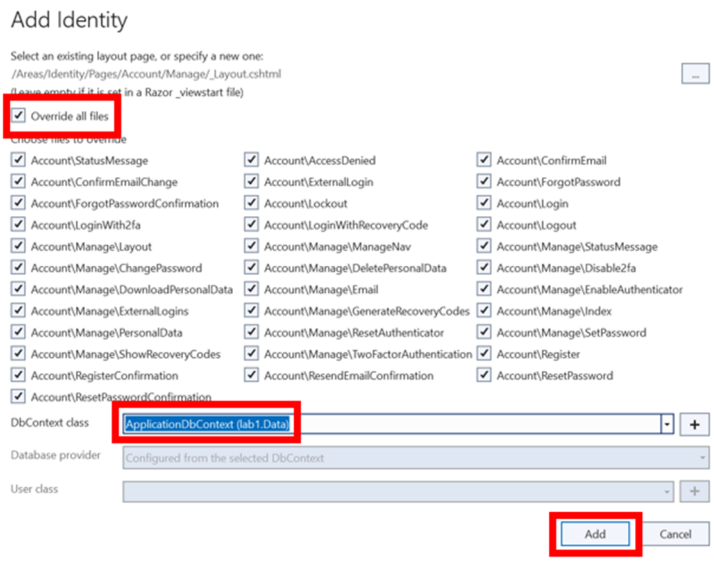


Figure 13 - Override all files & select DBContext class

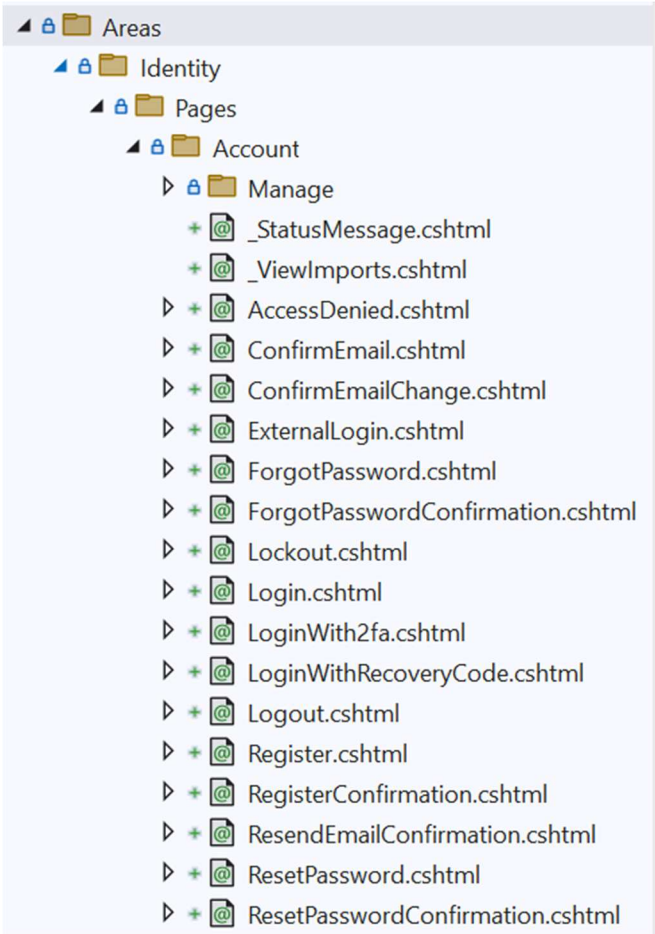


Figure 14 - All Identity pages were displayed to be overridden

6. Customize web layout & other web pages

```
<ul class="navbar-nav flex-grow-1">
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
    </li>
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="" asp-controller="Books" asp-action="Index">Books</a>
    </li>
</ul>
```

*Figure 15 - **Views/Shared/_Layout.cshtml***

```
<div class="text-center">
    <img src="https://monamedia.co/wp-content/uploads/2023/06/dac-diem-noi-bat-cua-asp-net.jpg" />
</div>
```

*Figure 16 - **Views/Home/Index.cshtml***

```
<p>
    <a class="btn btn-primary" asp-action="Create">Create New</a>
</p>
```

*Figure 17 - **Views/Books/Index.cshtml (1)***

```
<td>
    <a class="btn btn-warning" asp-action="Edit" asp-route-id="@item.BookId">Edit</a>
    <a class="btn btn-info" asp-action="Details" asp-route-id="@item.BookId">Details</a>
    <a class="btn btn-danger" asp-action="Delete" asp-route-id="@item.BookId">Delete</a>
</td>
```

*Figure 18 - **Views/Books/Index.cshtml (2)***

```
<td>
    <img src="@Html.DisplayFor(modelItem => item.BookCover)" width="100" height="120" />
</td>
```

*Figure 19 - **Views/Laptops/Index.cshtml (3)***

7. Seed initial data to database

```
protected override void OnModelCreating(ModelBuilder builder)
{
    base.OnModelCreating(builder);

    //Seed data for User & Role
    SeedUserRole(builder);

    //Seed data for table Book
    SeedBook(builder);
}
```

*Figure 20 - **Data/ApplicationDbContext.cs (1)***

```
private void SeedUserRole(ModelBuilder builder)
{
    //A) Setup IdentityUser
    //1. Create accounts
    var adminAccount = new IdentityUser
    {
        Id = "user1",
        UserName = "admin@gmail.com",
        Email = "admin@gmail.com",
        NormalizedUserName = "ADMIN@GMAIL.COM",
        NormalizedEmail = "ADMIN@GMAIL.COM",
        EmailConfirmed = true
    };

    var readerAccount = new IdentityUser
    {
        Id = "user2",
        UserName = "reader@gmail.com",
        Email = "reader@gmail.com",
        NormalizedUserName = "READER@GMAIL.COM",
        NormalizedEmail = "READER@GMAIL.COM",
        EmailConfirmed = true
    };

    //2. Declare hasher for password encryption
    var hasher = new PasswordHasher<IdentityUser>();

    //3. Setup password for accounts
    adminAccount.PasswordHash = hasher.HashPassword(adminAccount, "123456");
    readerAccount.PasswordHash = hasher.HashPassword(readerAccount, "123456");

    //4. Add accounts to database
    builder.Entity<IdentityUser>().HasData(adminAccount, readerAccount);
```

*Figure 21 - **Data/ApplicationDbContext.cs (2)***

```
        //B) Setup IdentityRole
        builder.Entity<IdentityRole>().HasData(
            new IdentityRole
            {
                Id = "role1",
                Name = "Admin",
                NormalizedName = "ADMIN"
            },
             new IdentityRole
             {
                 Id = "role2",
                 Name = "Reader",
                 NormalizedName = "READER"
             });

        //C) Setup IdentityUserRole
        builder.Entity<IdentityUserRole<string>>().HasData(
            new IdentityUserRole<string>
            {
                UserId = "user1",
                RoleId = "role1"
            },
            new IdentityUserRole<string>
            {
                UserId = "user2",
                RoleId = "role2"
            }
        );
    }
```

*Figure 22 - **Data/ApplicationDbContext.cs (3)***

```
private void SeedBook(ModelBuilder builder)
{
    builder.Entity<Book>().HasData(
        new Book
        {
            BookId = 1,
            BookTitle = "Clean Code",
            BookPrice = 12.34,
            BookCover = "https://images-na.ssl-images-amazon.com/images/I/51E2055ZGUL.jpg"
        },
        new Book
        {
            BookId = 2,
            BookTitle = "The Mindfulness Journal",
            BookPrice = 9.99,
            BookCover = "https://m.media-amazon.com/images/I/91nJdrcNBtL._AC_UF1000,1000_QL80_.jpg"
        },
        new Book
        {
            BookId = 3,
            BookTitle = "Harry Porter",
            BookPrice = 6.78,
            BookCover = "https://nhasachphuongnam.com/images/detailed/160/81YOuOGFCJL.jpg"
        }
    );
}
```

*Figure 23 - **Data/ApplicationDbContext.cs (4)***

8. Config system to use roles for web authorization

```
services.AddDefaultIdentity<IdentityUser>(options => options.SignIn
    .AddRoles<IdentityRole>()
    .AddEntityFrameworkStores<ApplicationDbContext>();
services.AddControllersWithViews();
services.AddRazorPages();
```

*Figure 24 - **Startup.cs***

9. Setup role-based access for different features
   - Restrict user access in backend (controller)

```
//Setup role for a controller
[Authorize(Roles = "Admin")]
public class BooksController : Controller
```

*Figure 25 - **Controllers/BooksController.cs (1)***

```
//Setup mutiple roles for an action
[Authorize(Roles = "Admin, Reader")]
public async Task<IActionResult> Index()
```

*Figure 26 - **Controllers/BooksController.cs (2)***

```
//Setup single role for an action
[Authorize(Roles = "Admin")]
public IActionResult Create()
```

*Figure 27 - **Controllers/BookssController.cs (3)***

```
//Setup access for all roles (logged in users)
[Authorize]
public async Task<IActionResult> Index()
```

*Figure 28 - **Controllers/BooksController.cs (4)***

- Restrict user access in frontend (view)

```
<p>
    @if (User.Identity.IsAuthenticated && User.IsInRole("Admin"))
    {
        <a asp-action="Create">Create New</a>
    }
</p>
```

*Figure 29 - **Views/Laptops/Index.cshtml***

10.Set default role for user registration *(optional)*

```
var callbackUrl = Url.Page(
    "/Account/ConfirmEmail",
    pageHandler: null,
    values: new { area = "Identity", userId = user.Id, code = code, returnUrl = returnUrl },
    protocol: Request.Scheme);

//Set default role "Reader" for new user registration
var role = "Reader";
await _userManager.AddToRoleAsync(user, role);

await _emailSender.SendEmailAsync(Input.Email, "Confirm your email",
    $"Please confirm your account by <a href='{HtmlEncoder.Default.Encode(callbackUrl)}'>clicking here</a>.");
```

*Figure 30 - **Areas/Identity/Pages/Account/Register.cshtml.cs***

11.Migrate data to database

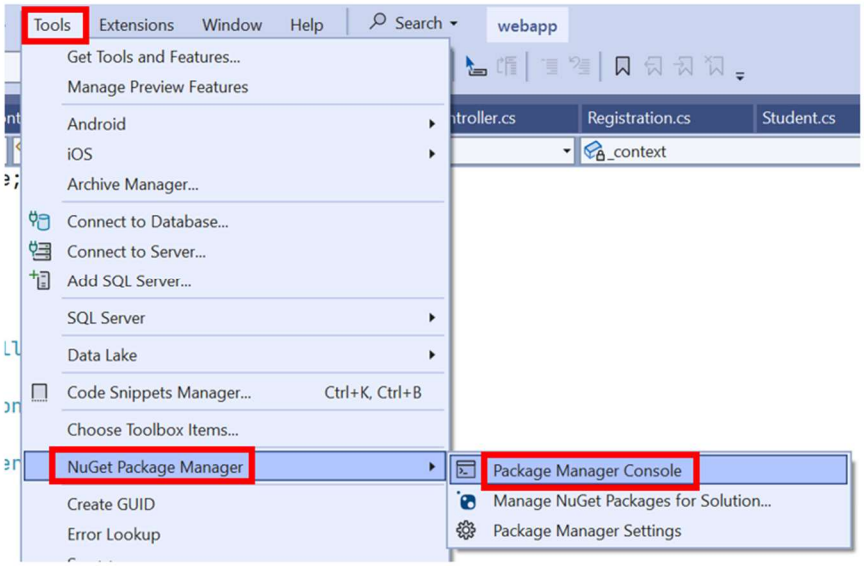


*Figure 31 - Open **Package Manager Console (PMC)***

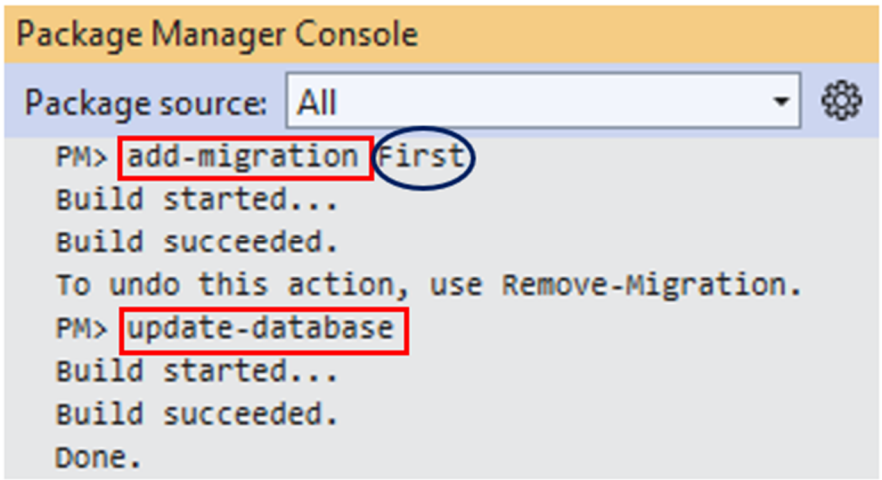


Figure 32 - Add migration & update database using PMC

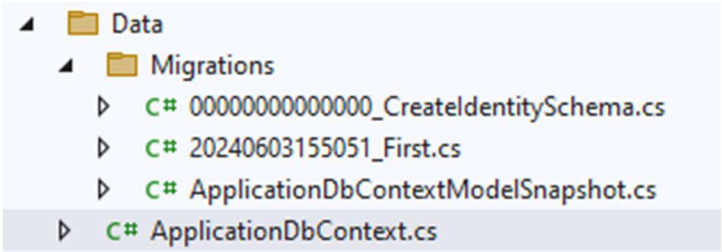


*Figure 33 - Migration file is created automatically in **Data/Migrations***

12.Sample final result

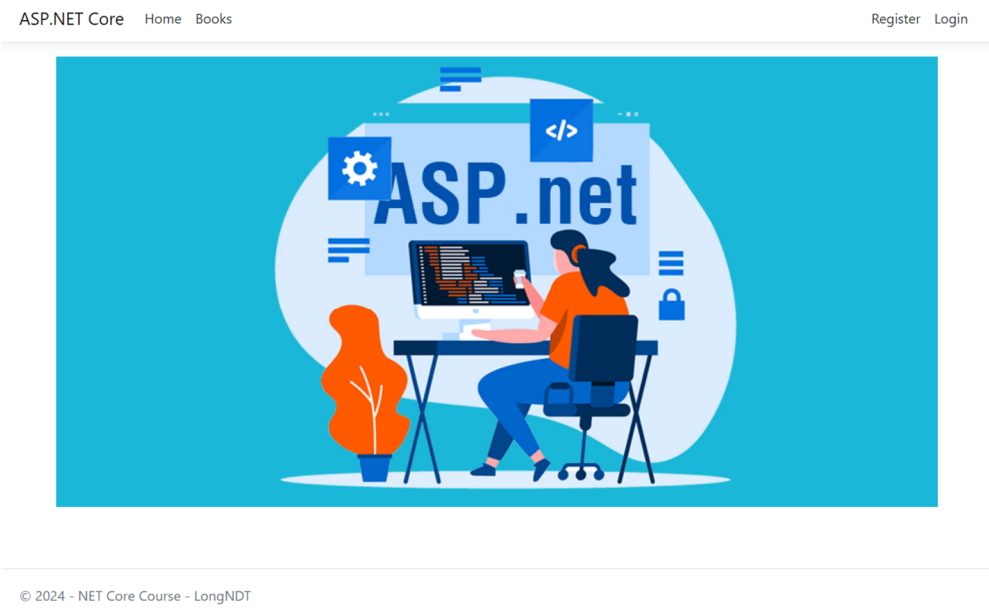


Figure 34 - Homepage

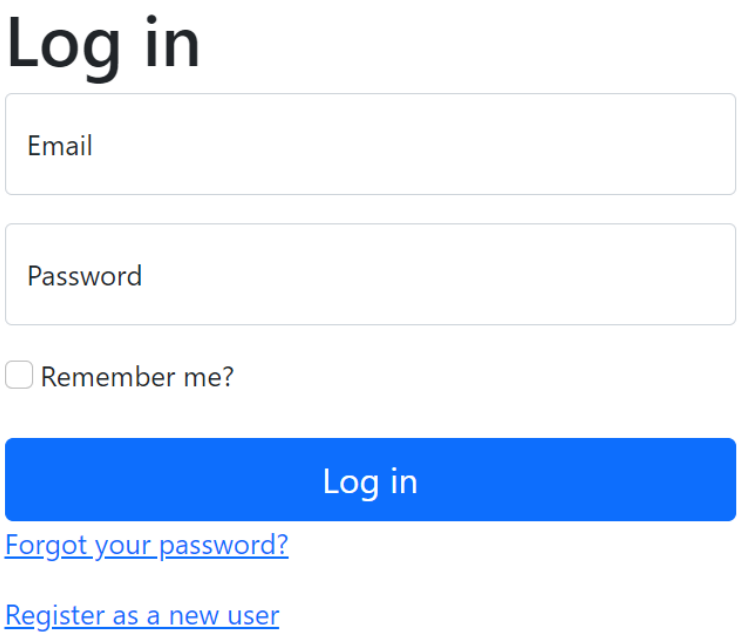


Figure 35 - Login page

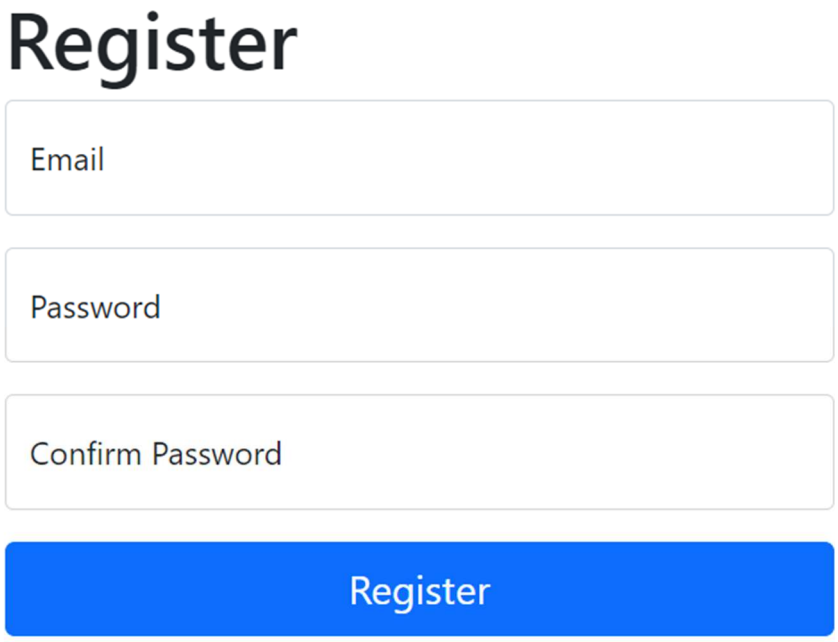


Figure 36 - Register page

Index

Create New

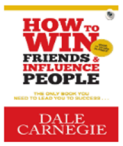
BookTitle	BookPrice	BookCover	Menu
Clean Code	$12.34		Edit Details Delete
How to win friends & influence people	$9.99		Edit Details Delete
Harry Porter	$6.78		Edit Details Delete

Figure 37 - View book list

Create

Book

BookTitle

BookPrice

BookCover

Create

Back to List

Figure 38 - Create new book

Edit

Book

BookTitle

Clean Code

BookPrice

12.34

BookCover

https://images-na.ssl-images-amazon.com/im

Save

Back to List

Figure 39 - Edit existing book

Delete

Are you sure you want to delete this?

Book

BookTitle	How to win friends & influence people
BookPrice	9.99
BookCover	https://rukminim2.flixcart.com/image/850/1000/kkr72q80/book/k/4/l/how-to-win-friends-influence-people-international-bestseller-original-imagyf2wgzsbgvba.jpeg?q=90&crop=false

Delete | Back to List

Figure 40 - Delete existing book

Access denied

You do not have access to this resource.

Figure 41 - Access denied page