DEPI

E-Commerce

Full Stack .Net Web Developers

Dr.Fathy Ahmed

30 April, 2025

Team Members

FrontEnd:

Alaa Hassan

Youssef Ahmed

Manar Awad

BackEnd:

Ahmed Amin

Mohamed Khalid

Introduction

This project is an Online Book Shopping built with ASP.NET Core MVC . It allows users to browse, add books to a cart, and place orders securely. Admins can manage books, genres, stocks, and view reports. The system ensures a smooth and secure shopping experience for all users.

Idea

This project is an online bookstore web application.Users can browse books, add them to a shopping cart, and place orders.Admins can manage books, genres, stock, and reports easily.The platform aims to deliver a fast and secure online shopping experience.

Objectives

- Easy-to-use interface.
- Fast, secure checkout.
- Manage books and stock easily.
- Support multiple categories and genres.

System Analysis

- We started by understanding the business needs and defining the main entities like Books, Users, Orders, and Genres.

- Through Entity Framework, the database was generated automatically based on my models and configurations.

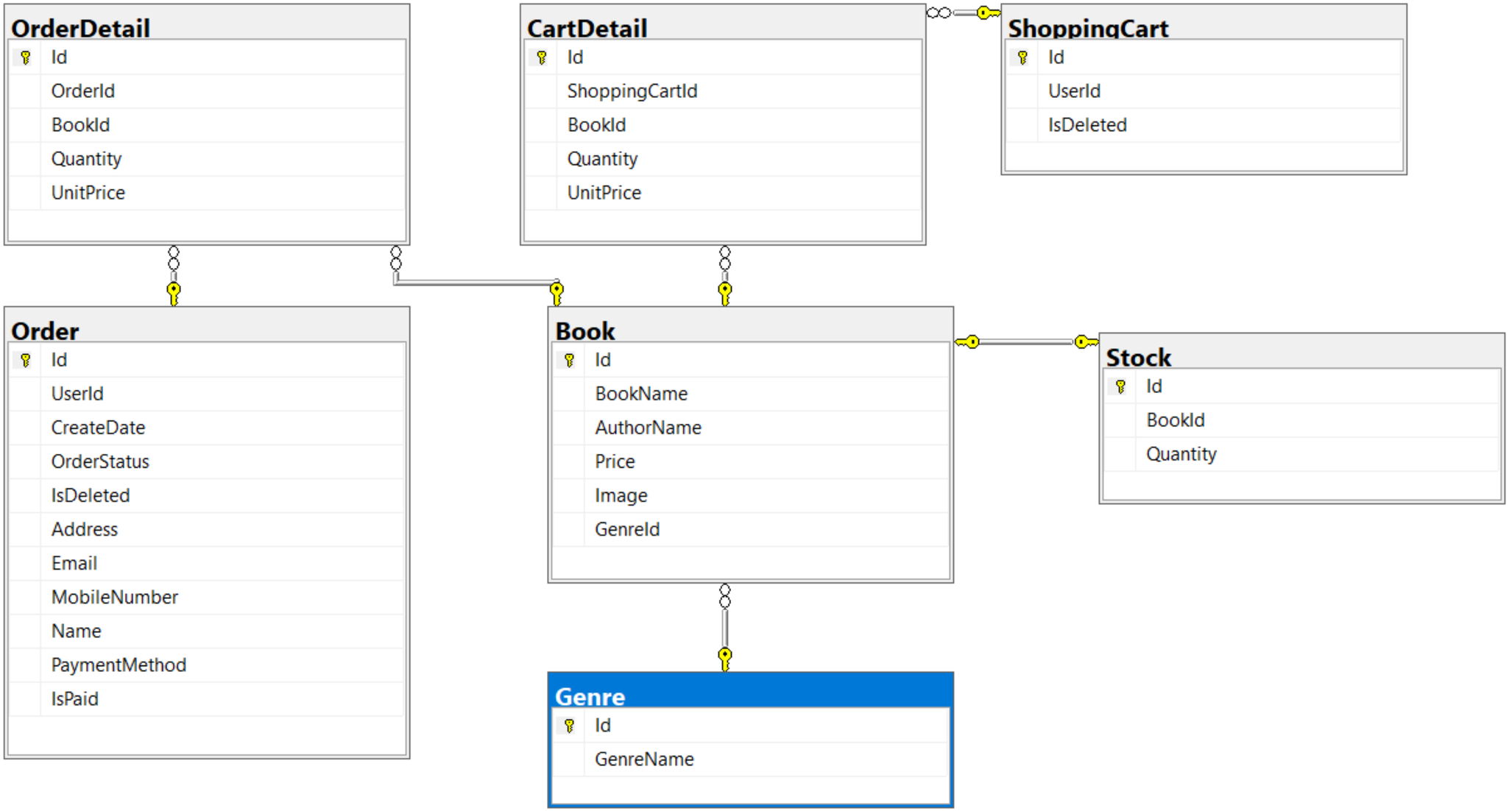
- I designed the database structure using the Code First approach by creating models and setting their relationships.

- Finally, I analyzed user roles and system operations to ensure a complete and organized structure before development.

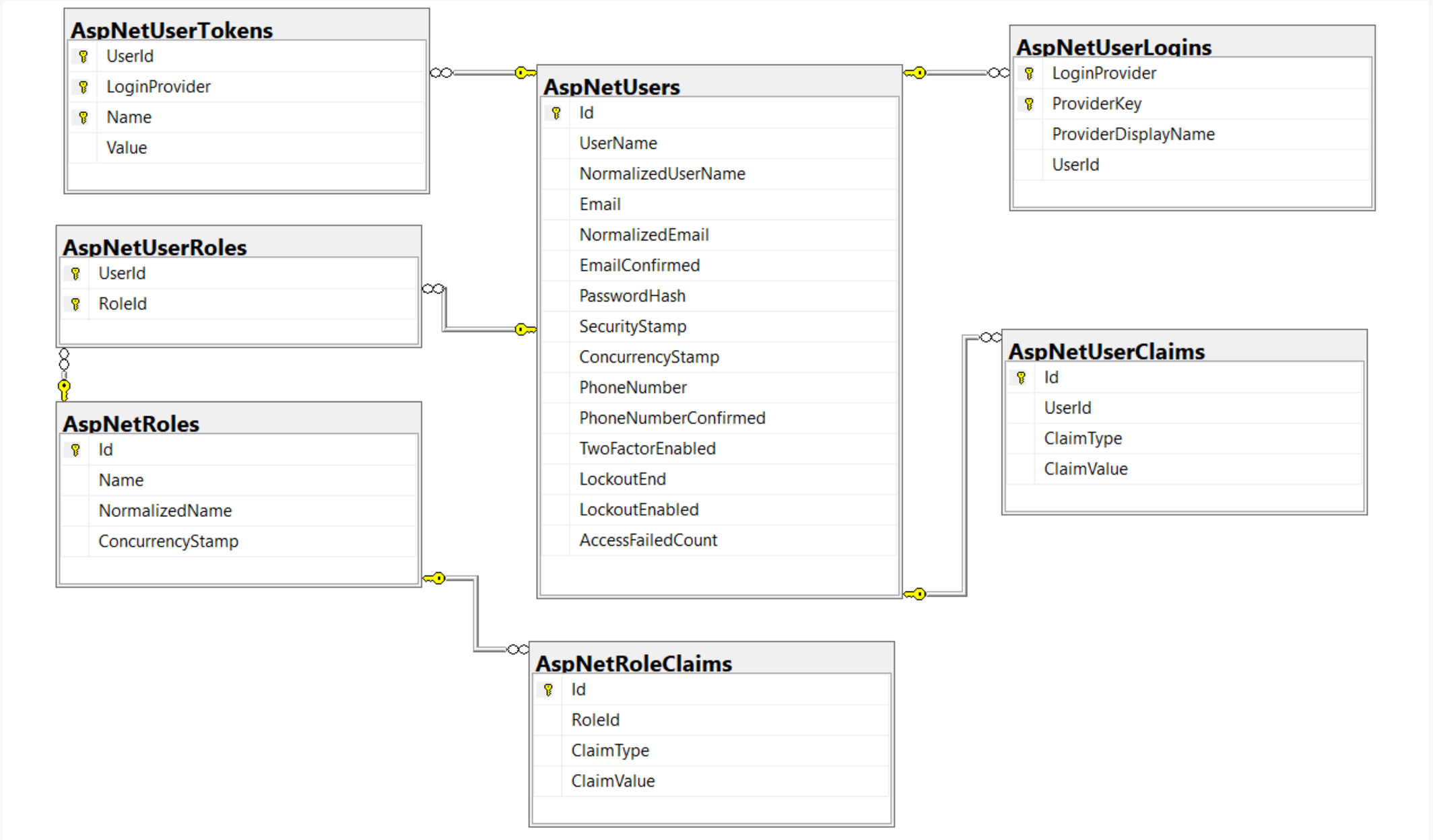
System Requirements

- Login/Registration
- Display-Add-Update(Book)
- Homepage
- Cart
- Checkout
- Admin Dashboard
- Top Selling Books
- Display-Update (Stock)
- Display-Add-Update(Genre)
- Orders(Details-Update-Status)

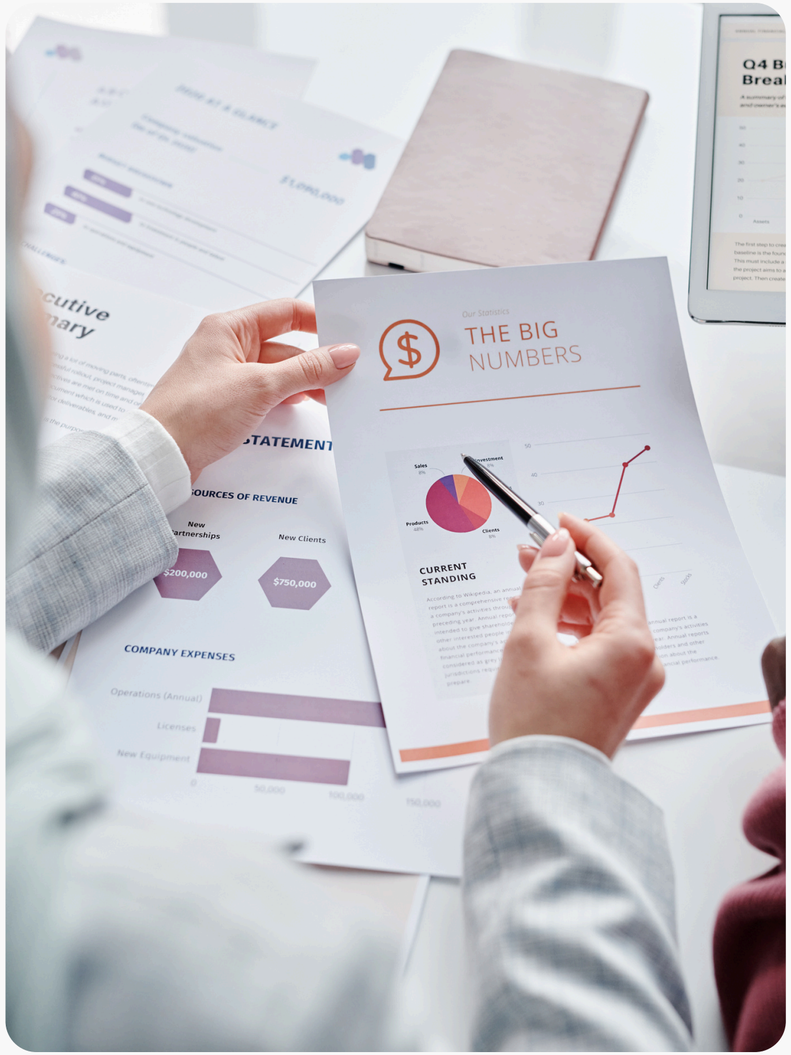
Data Base


OrderDetail
Id
OrderId
BookId
Quantity
UnitPrice
CartDetail
Id
ShoppingCartId
BookId
Quantity
UnitPrice
ShoppingCart
Id
UserId
IsDeleted
Order
Id
UserId
CreateDate
OrderStatus
IsDeleted
Address
Email
MobileNumber
Name
PaymentMethod
IsPaid
Book
Id
BookName
AuthorName
Price
Image
GenreId
Stock
Id
BookId
Quantity
Genre
Id
GenreName

Data Base


AspNetUserTokens
UserId
LoginProvider
Name
Value
AspNetUsers
Id
UserName
NormalizedUserName
Email
NormalizedEmail
EmailConfirmed
PasswordHash
SecurityStamp
ConcurrencyStamp
PhoneNumber
PhoneNumberConfirmed
TwoFactorEnabled
LockoutEnd
LockoutEnabled
AccessFailedCount
AspNetUserLogins
LoginProvider
ProviderKey
ProviderDisplayName
UserId
AspNetUserRoles
UserId
RoleId
AspNetRoles
Id
Name
NormalizedName
ConcurrencyStamp
AspNetUserClaims
Id
UserId
ClaimType
ClaimValue
AspNetRoleClaims
Id
RoleId
ClaimType
ClaimValue

Implementation



-Architecture Used:

The project uses the MVC (Model-View-Controller) architecture, which organizes the application into three main components:

- Model: Manages data and business logic.
- View: Handles the display of information to the user.
- Controller: Handles user input and interactions, updates the model and view accordingly.

-Benefits of MVC:

- Separation of concerns: makes the code easier to manage and scale.
- Improves testability and maintainability.
- Enables parallel development: designers can work on views while developers work on controllers.
- Supports clean and organized project structure.

Implementation

- In this project, we used the Code First approach to easily design the database directly from the C# classes, giving us more control over the models and saving time during development
- We chose ASP.NET Core MVC (.NET 9) because it offers high performance, security, and flexibility for building modern web applications. ASP.NET was selected for its strong support, built-in authentication features, and its ability to create scalable and maintainable projects.
- Finally, we used C# as the main programming language because it is powerful, object-oriented, and perfectly integrates with the .NET ecosystem, making development faster and more efficient.

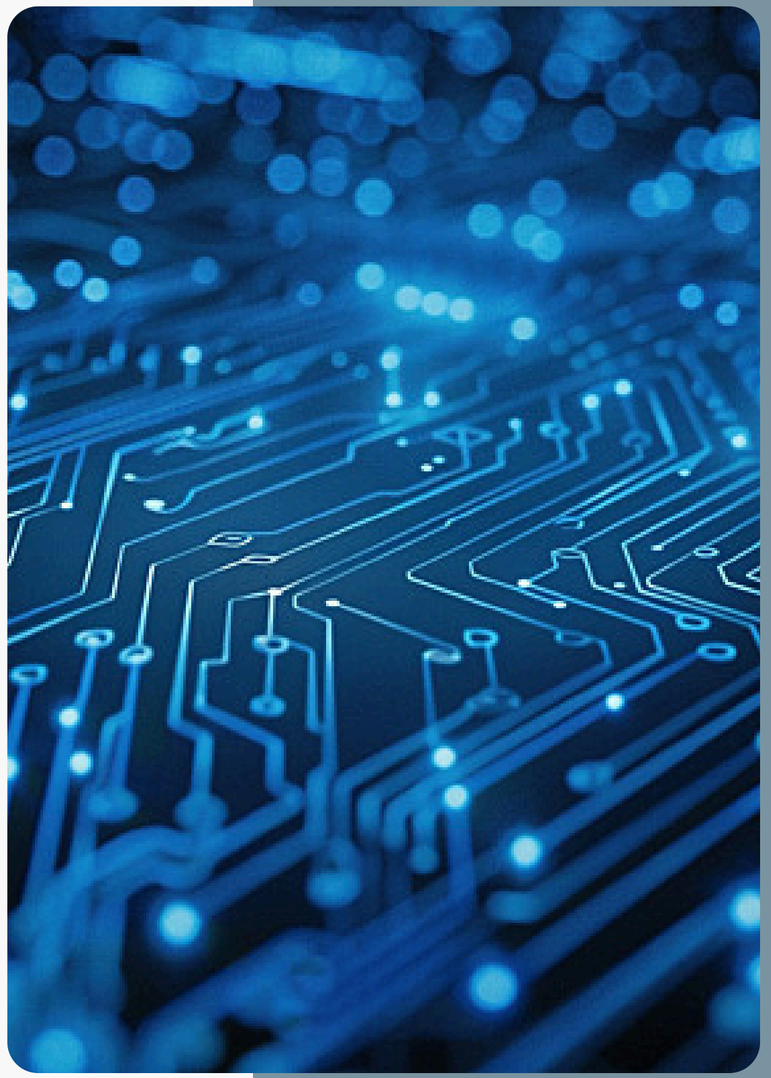
Algorithms

- Sorting & Filtering (LINQ Algorithms):
  - OrderBy, Where used for product listing, and search filters.
  - Efficient in-memory or deferred execution using LINQ-to-Entities.

- Hashing (Security):
  - ASP.NET Identity uses hashing algorithm for securely storing user passwords.

- Model Validation Algorithms:
  - Data Annotations trigger validation logic via reflection and attribute-based checks.

- Session & State Management:
  - ASP.NET uses built-in stateful session tracking algorithms to manage user carts and data across requests.



Methods

- Model-View-Controller (MVC)
  - Separates data (Model), UI (View), and control logic (Controller).
- Entity Framework (EF)
  - Handles database CRUD operations with LINQ queries.
- Session Management
  - Cart data is stored per user session.

Techniques

- Form Validation
  - Data Annotations used to enforce constraints on models.
- Security Measures
  - ASP.NET Identity for login, roles, and claims-based authorization.
- Shopping Cart Logic
  - Total = Σ (Item Price × Quantity).
  - Cart actions: Add, Update, Remove.

Test Cases

Test Case 1

Test Case 1 Test Case Identifier	TC-1
Use Case tested	Create Account
Pass/Fail criteria	The test passes if the user enter valid id and email) it has not been used by another account before)
Input data	Id , Name , Email , Password
Test Procedure	**Expected Result**
Step.1 Insert invalid id with valid mail	System will show a message “The id used before ”
Step.2 Insert invalid id with invalid mail.	System will show a message “The id and mail used before ”
Step.3 Insert valid id with invalid mail	System will show a message “The mail used before ”
Step.4 Insert valid id with valid mail.	User account is created successfully.

Test Case 2

Test Case 2 Test Case Identifier	TC-2
Use Case tested	User Login
Pass/Fail criteria	Valid credentials grant access; invalid ones show errors.
Input data	Email , Password
Test Procedure	**Expected Result**
Step.1 Valid email + wrong password	Error: "Invalid password"
Step.2 Invalid email + any password	Error: "Email not found"
Step.3 Valid credentials	Redirect to homepage

Test Case 3

Test Case 3 Add Items to Cart	TC-3
Use Case tested	Add Items to Cart
Pass/Fail criteria	Items added if stock available
Input data	Book ID, Quantity
Test Procedure	**Expected Result**
Step.1 Quantity > stock	"Only [X] items available" warning
Step.2 Valid quantity	Item added to cart
Step.3 Same book added again	Quantity increases (if stock allows)

Test Case 4

Test Case 4 Checkout Process	TC-4
Use Case tested	Complete Purchase
Pass/Fail criteria	Order placed with valid payment
Input data	Payment Info, Shipping Address
Test Procedure	**Expected Result**
Step.1 Empty cart checkout	"Your cart is empty" error
Step.2 Invalid payment	"Payment failed" error
Step.3 Valid checkout	Order confirmation page

Innovation & Added Value

This system brings innovation by providing an easy and efficient way for users to browse, order, and manage books online. It adds value through features like a user-friendly shopping cart, secure order handling, and dynamic stock management. Using modern technologies like ASP.NET Core MVC ensures high performance, security, and a smooth user experience. The system also supports admin control, reporting, and real-time updates, making it a complete solution for online book shopping.

Demo

Thank you