BookStore MVC — Deep Study & Teaching Guide *(Generated on 2025-09-02 02:47)*

Table of Contents

1) Project Concept & Architecture This is a full-stack **ASP.NET Core 8 MVC** bookstore with **EF Core** and Razor Views. It supports CUSTOMER and ADMIN roles. Core modules: **User Management**, **Product/Category**, **Cart**, **Orders**, **Payment (mock)**. Front-end uses **Bootstrap 5** (layout/components) and **jQuery** (light UI scripts).

2) How MVC Works Here (Request Flow) URL → Router → Controller Action → Service → DbContext (EF Core) → back to Controller → Razor View → HTML to browser. - Keep controllers thin: call services for business rules and data access. - Keep models clean: attributes for validation and relationships. - Views should not talk to DbContext.

3) Folder-by-Folder Explanation - **Controllers/**: request handlers for each feature. - **Services/**: business / data logic; injected via DI. - **Models/**: EF Core entities and possibly DbContext. - **Views/**: Razor pages grouped by feature; `_Layout.cshtml` (site shell) and possibly `_AdminLayout.cshtml`. - **wwwroot/**: static assets (css, js, images). - **appsettings*.json**: configuration (ConnectionStrings, Logging). - **Program.cs**: app composition (DI, EF provider, auth, middleware, routing). - **Migrations/**: EF Core migrations (if Code First).

4) Program.cs — EF Provider, Auth, Routing - **EF Provider detected**: Sqlite - **Cookie authentication configured**: True - LoginPath:

/Account/Login - AccessDeniedPath: /Account/AccessDenied - **Policies: AdminOnly - **Routes**: - default: {controller=Home}/{action=Index}/{id?}**

5) Models — Line-by-Line Highlights **ApplicationDbContext** (Models/ApplicationDbContext.cs) - Users : DbSet (no attributes) - Categories : DbSet (no attributes) - Books : DbSet (no attributes) - CartItems : DbSet (no attributes) - Orders : DbSet (no attributes) - OrderItems : DbSet (no attributes) - Payments : DbSet (no attributes) **Book** (Models/Book.cs) - Id : int (Key) - Title : string (MaxLength) - Description : string (MaxLength) - Price : decimal (no attributes) - CategoryId : int (Required) - Category : Category? (no attributes) - StockQuantity : int (Required) - ImageUrl : string (no attributes) - CartItems : ICollection (no attributes) - OrderItems : ICollection (no attributes) **CartItem** (Models/CartItem.cs) - Id : int (Key) - UserId : int (Required) - User : User? (no attributes) - BookId : int (Required) - Book : Book? (no attributes) - Quantity : int (Required) **Category** (Models/Category.cs) - Id : int (Key) - Name : string (MaxLength) - Books : ICollection (no attributes) **DbInitializer** (Models/DbInitializer.cs)

Order (Models/Order.cs)
- Id : int (Key)
- UserId : int (Required)
- User : User? (no attributes)
- TotalAmount : decimal (Required)
- OrderDate : DateTime (Required)
- Status : OrderStatus (Required)
- ShippingName : string? (no attributes)
- ShippingAddress : string? (no attributes)
- ShippingCity : string? (no attributes)
- ShippingState : string? (no attributes)
- ShippingZip : string? (no attributes)
- Phone : string? (no attributes)
- OrderItems : ICollection<OrderItem> (no attributes)
- Payment : Payment? (no attributes)

OrderItem (Models/OrderItem.cs)
- Id : int (Key)
- OrderId : int (Required)
- Order : Order? (no attributes)
- BookId : int (Required)
- Book : Book? (no attributes)
- Quantity : int (Required)
- UnitPrice : decimal (Required)

Payment (Models/Payment.cs)
- Id : int (Key)
- OrderId : int (Required)
- Order : Order? (no attributes)
- Amount : decimal (Required)
- PaymentStatus : PaymentStatus (Required)
- PaymentMethod : string (Required)
- PaymentDate : DateTime (Required)

User (Models/User.cs)
- Id : int (Key)
- Username : string (MaxLength)
- PasswordHash : string (MaxLength)
- Role : UserRole (Required)
- Email : string (EmailAddress)
- CartItems : ICollection<CartItem> (no attributes)
- Orders : ICollection<Order> (no attributes)

6) Services — Responsibilities & Method Signatures **Services/BookService.cs — class - AddAsync - BookService - DeleteAsync - GetBooksAsync - GetByIdAsync - GetCategoriesAsync - UpdateAsync **Services/CartService.cs** — class - AddToCartAsync - CartService - ClearAsync - GetCartItemsAsync - RemoveAsync - UpdateQuantityAsync **Services/IBookService.cs** — interface - (no top-level methods parsed) **Services/ICartService.cs** — interface - (no top-level methods parsed) **Services/IOrderService.cs** — interface - (no top-level methods parsed) **Services/IPaymentService.cs** — interface - (no top-level methods parsed) **Services/OrderService.cs** — class - CreateOrderAsync - GetOrderAsync - GetOrdersAsync - OrderService - UpdateStatusAsync **Services/PaymentService.cs** — class - PaymentService - ProcessPaymentAsync**

7) Controllers — Routes, Actions, Authorization **AccountController (Controllers/AccountController.cs) Authorize (class): — Base route: conventional Actions: - Register [HttpGet] - Register [HttpGet, HttpPost] - ForgotPassword [HttpGet] - ForgotPassword [HttpGet, HttpPost] - Login [HttpGet] - Login [HttpGet, HttpPost] - Logout [HttpGet] - AccessDenied [HttpGet, HttpGet] **AdminController** (Controllers/AdminController.cs) Authorize (class): Roles = "ADMIN" Base route: conventional Actions: - Dashboard [HttpGet] - Users [HttpGet] **BooksController** (Controllers/BooksController.cs) Authorize (class): Roles = "ADMIN", Roles = "ADMIN", Roles = "ADMIN", Roles = "ADMIN", Roles = "ADMIN", Roles = "ADMIN" Base route: conventional Actions: - Index [HttpGet] - Details [HttpGet] - Create [HttpGet] - Create [HttpPost] - Edit [HttpGet] - Edit [HttpPost] - Delete [HttpGet] - DeleteConfirmed [HttpGet] **CartController** (Controllers/CartController.cs) Authorize (class): Base route: conventional Actions: - Index [HttpGet] - Add [HttpGet] - Update [HttpGet] - Remove [HttpGet] - Checkout [HttpGet] - Checkout [HttpPost] - Payment [HttpGet] - Payment [HttpPost] - Success [HttpGet] **HomeController** (Controllers/HomeController.cs) Authorize (class): — Base route: conventional Actions: - Index [HttpGet] - Privacy [HttpGet] - Error [HttpGet]**

OrdersController (Controllers/OrdersController.cs) Authorize (class): , Roles = "ADMIN" Base route: conventional Actions: - Index [HttpGet] - Details [HttpGet] - UpdateStatus [HttpPost]

8) Views — Bootstrap/jQuery Usage Per Page **Views/Account/AccessDenied.cshtml** Bootstrap: Buttons, Cards jQuery: — **Views/Account/ForgotPassword.cshtml** Bootstrap: Alerts, Buttons, Cards, Forms, Grid, Rows/Cols jQuery: — **Views/Account/Login.cshtml** Bootstrap: Alerts, Buttons, Cards, Forms, Grid, Rows/Cols jQuery: $(function () { $('#togglePassword').on('click', function () { var input = $('#password'); $(this).find('i').toggleClass('bi-eye bi-eye-slash'); **Views/Account/Register.cshtml** Bootstrap: Alerts, Buttons, Cards, Forms, Grid, Rows/Cols jQuery: $(function () { $('#togglePassword').on('click', function () { var input = $('#password'); $(this).find('i').toggleClass('bi-eye bi-eye-slash'); $('#toggleConfirmPassword').on('click', function () { var input = $('#confirmPassword'); $(this).find('i').toggleClass('bi-eye bi-eye-slash'); **Views/Admin/Dashboard.cshtml** Bootstrap: Cards, Rows/Cols jQuery: — **Views/Admin/Users.cshtml** Bootstrap: Cards, Table jQuery: — **Views/Books/Create.cshtml** Bootstrap: Buttons, Forms, Rows/Cols jQuery: — **Views/Books/Delete.cshtml** Bootstrap: Buttons jQuery: — **Views/Books/Details.cshtml** Bootstrap: Badges, Buttons, Cards, Rows/Cols jQuery: — **Views/Books/Edit.cshtml** Bootstrap: Buttons, Forms, Rows/Cols jQuery: — **Views/Books/Index.cshtml** Bootstrap: Badges, Buttons, Cards, Rows/Cols jQuery: — **Views/Cart/Checkout.cshtml** Bootstrap: Buttons, Cards, Forms, Grid, Rows/Cols jQuery: $(function () { $('#checkoutForm').on('submit', function () { $(this).find('input[required]').each(function () { if (!$(this).val().trim()) { $(this).addClass('is-invalid'); $(this).removeClass('is-invalid'); **Views/Cart/Index.cshtml** Bootstrap: Alerts, Buttons, Cards, Forms, Rows/Cols, Table jQuery: — **Views/Cart/Payment.cshtml** Bootstrap: Alerts, Buttons, Cards, Forms, Grid, Rows/Cols jQuery: $(function () { $('#cardUPI').addClass('active'); $('#cardCARD').removeClass('active'); $('#upiBlock').show(); $('#cardBlock').hide(); $('#cardCARD').addClass('active'); $('#cardUPI').removeClass('active'); $('#cardBlock').show(); **Views/Cart/Success.cshtml** Bootstrap: Buttons, Cards, Rows/Cols jQuery: — **Views/Home/Error.cshtml** Bootstrap: — jQuery: — **Views/Home/Index.cshtml** Bootstrap: Badges, Buttons, Cards, Rows/Cols jQuery: — **Views/Home/Privacy.cshtml** Bootstrap: — jQuery: — **Views/Orders/Details.cshtml** Bootstrap: Badges, Buttons, Cards, Rows/Cols jQuery: — **Views/Orders/Index.cshtml** Bootstrap: Alerts, Badges, Buttons, Cards, Rows/Cols, Table jQuery: —

****Views/Shared/_AdminLayout.cshtml** Bootstrap: Dropdowns/Modals, Grid, Navbar jQuery: — **Views/Shared/_Layout.cshtml** Bootstrap: Dropdowns/Modals, Grid, Navbar jQuery: — **Views/Shared/_ValidationScriptsPartial.cshtml** Bootstrap: — jQuery: — **Views/_ViewImports.cshtml** Bootstrap: — jQuery: — **Views/_ViewStart.cshtml** Bootstrap: — jQuery: —**

9) Module Walkthroughs ### 9.1 User Management - **Model: \`User\` (Username, Email, PasswordHash, Role, nav: CartItems, Orders). - **Service**: Register (hash password, save), ValidateCredentials (verify hash), SignIn/SignOut (cookie). - **Controller**: \`AccountController\` with \`Login\`, \`Register\`, \`Logout\`, \`AccessDenied\`. - **Views**: \`Views/Account/Login.cshtml\`, \`Register.cshtml\`, \`AccessDenied.cshtml\`. - **Authorization**: \`[Authorize(Roles="ADMIN")]\` on admin controllers/actions.**

9.2 Products (Books & Categories) - **Models**: \`Book\`, \`Category\` (1→many). - **CRUD (admin)** and **listing** (public) via \`BooksController\` + \`BookService\`. - Views use Bootstrap cards and grids.

9.3 Cart - **Model**: \`CartItem\` (UserId, BookId, Quantity). - **Actions**: Add, Update, Remove; Summary shows subtotal.

9.4 Orders - **Models**: \`Order\` + \`OrderItem\` (1→many). - **Customer**: order history & details. **Admin**: update status.

9.5 Payment (Mock) - **Model**: \`Payment\` (OrderId, Amount, Method, Status, Date). - **Service**: \`RecordPaymentAsync\`. - Pages for Checkout and Payment include light client-side checks.

10) SQL Server Setup & Migrations 1. **.csproj: Use stable EF 8 + SqlServer provider. 2. **appsettings.json**: \`"DefaultConnection"\` to your SQL Server. 3. **Program.cs**: \`UseSqlServer(...)\` with that connection string. 4. **Migrations** (if Code First): - \`dotnet ef migrations add Init\` - \`dotnet ef database update\` - Or call \`context.Database.Migrate()\` on startup.**

11) End-to-End Workflows (Trace These) - **Login: GET form → POST credentials → verify hash → create claims → \`SignInAsync\` (cookie) → redirect by role. - **Add to Cart**: POST BookId → upsert \`CartItem\` for user → cart page. - **Checkout→Payment→Order**: compute totals server-side → mock payment recorded → order created and status set. - **Admin**: protected by role; manage books/categories/orders.**

12) Security, Validation & Best Practices - Store **PasswordHash only (PBKDF2/BCrypt/Argon2). Salt + per-user iterations. - Use **[ValidateAntiForgeryToken]** on POST forms. - Server-side validation attributes (`[Required]`, `[MaxLength]`) + client hints via Bootstrap. - Prefer `Include(...)` when you need related data; paginate large lists. - For production, consider **ASP.NET Identity** to replace custom auth for password resets, lockouts, etc.**

13) Study Plan & Presenter Notes - Start with **Program.cs (wiring), then pick **one workflow** to demo. - For each controller, show its **service** and the **view** it returns. - Keep a whiteboard diagram: Controller ■ Service ■ DbContext; Views separate. - Demo a role-restricted page and show AccessDenied behavior.**

14) Interview Questions (with hints) - MVC boundaries? (Controller thin; service rules; models as entities/DTOs) - EF relationships example from code? - Cookie auth vs JWT for MVC apps? - Where to add policies and how to consume `[Authorize(Policy="...")]`? - Steps to move from SQLite → SQL Server?