
Assignment Set 1: E-Commerce Domain

Assignment 1: Product Catalog & Cart Management System

Complexity: Medium → Advanced
Focus Areas:

- Blazor Components
- Component Parameters
- Event Handling
- Domain Models
- Parent–Child Communication

Business Scenario

You are building a simplified **E-commerce Product Catalog** where users can browse products and add them to a cart.

Requirements

Models

Create the following models:

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
    public bool InStock { get; set; }
}


public class CartItem
{
```

```
    public Product Product { get; set; }
    public int Quantity { get; set; }
}
```

Components to Build

1. **ProductList.razor**
    - Accepts a list of Product via [Parameter]
    - Renders products using a child component
2. **ProductItem.razor**
    - Displays product details
    - Includes an **Add to Cart** button
    - Raises an event to the parent using EventCallback<Product>
3. **CartSummary.razor**
    - Displays total items and total price
    - Reacts to cart updates

Event Handling Requirements

- Use @onclick to add items to cart
- Disable **Add to Cart** button when InStock == false
- Use EventCallback<T> instead of direct method calls

Advanced Tasks

- Add quantity increment/decrement buttons
- Prevent adding duplicate products (update quantity instead)
- Highlight out-of-stock products using conditional rendering

---

Assignment Set 2: Banking Domain

Assignment 2: Bank Account Dashboard

Complexity: Advanced
Focus Areas:

- Component Parameters
- Event Delegates
- State Synchronization
- Business Rules in UI

Business Scenario

Create a **Bank Account Dashboard** that allows users to perform transactions and view balance updates in real time.

Models

```
public class BankAccount
{
    public string AccountNumber { get; set; }
    public string HolderName { get; set; }
    public decimal Balance { get; set; }
}


public class Transaction
{
    public decimal Amount { get; set; }
    public string Type { get; set; } // Deposit / Withdraw
    public DateTime Date { get; set; }
}
```

Components

1. **AccountDetails.razor**
    - Receives BankAccount via [Parameter]
    - Displays account info and current balance
2. **TransactionForm.razor**
    - Accepts transaction amount
    - Emits events for Deposit and Withdraw
3. **TransactionHistory.razor**
    - Displays list of transactions
    - Updates dynamically

Event Handling Requirements

- Use separate event handlers for **Deposit** and **Withdraw**
- Block withdrawals if balance is insufficient
- Use EventCallback<Transaction>

Advanced Tasks

- Add validation messages using conditional rendering
- Maintain transaction history in parent component
- Disable Withdraw button when balance is zero

Assignment Set 3: Healthcare Domain

Assignment 3: Patient Appointment Management System

Complexity: Medium → Advanced
Focus Areas:

- Parameter Binding
- Custom Event Handling
- Conditional UI Rendering
- Domain-Driven Models

Business Scenario

Develop a **Patient Appointment Booking System** for a healthcare portal.

Models

```
public class Patient
{
   public int Id { get; set; }
   public string Name { get; set; }
   public int Age { get; set; }
}

public class Appointment
{
   public DateTime Date { get; set; }
   public string DoctorName { get; set; }
   public string Status { get; set; } // Booked, Cancelled
}
```

Components

1. **PatientList.razor**
    - Displays list of patients

    - Allows selecting a patient
2. **AppointmentForm.razor**
    - Receives selected patient via [Parameter]
    - Books appointments using event callbacks
3. **AppointmentStatus.razor**
    - Shows appointment details
    - Allows cancellation

Event Handling Requirements

- Use @onclick to book and cancel appointments
- Use EventCallback<Appointment>
- Prevent booking appointments for past dates

Advanced Tasks

- Disable booking button until patient is selected
- Change appointment status dynamically
- Display warning messages using conditional rendering
