Name: Muhammad Saad Khan

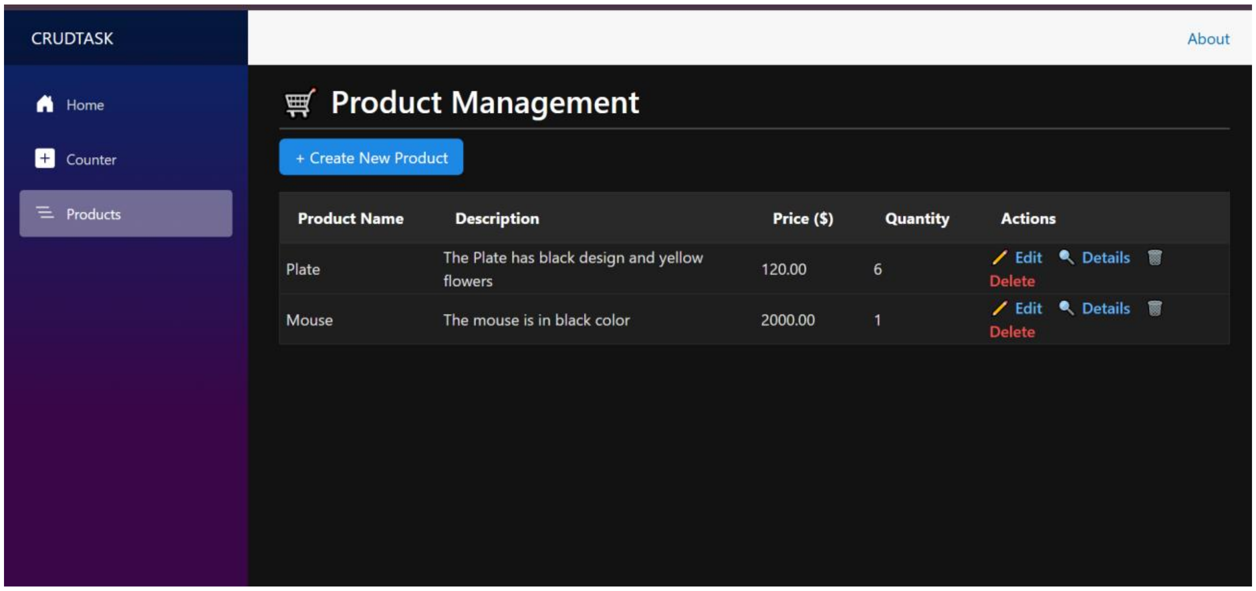
ROLL NO: 232423

BSCS-eve-4-B

Visual Programming Lab 11 and 12

OUTPUT:

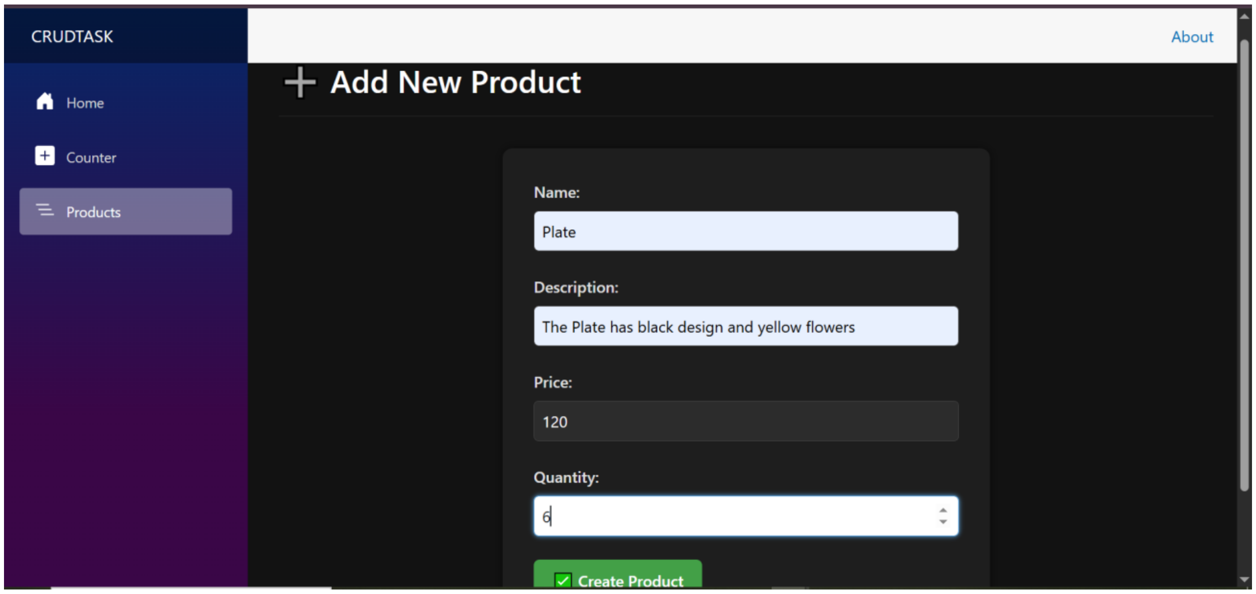
HOME:



Product Management

Product Name	Description	Price ($)	Quantity	Actions
Plate	The Plate has black design and yellow flowers	120.00	6	Edit Details Delete
Mouse	The mouse is in black color	2000.00	1	Edit Details Delete

CREATE:



Add New Product

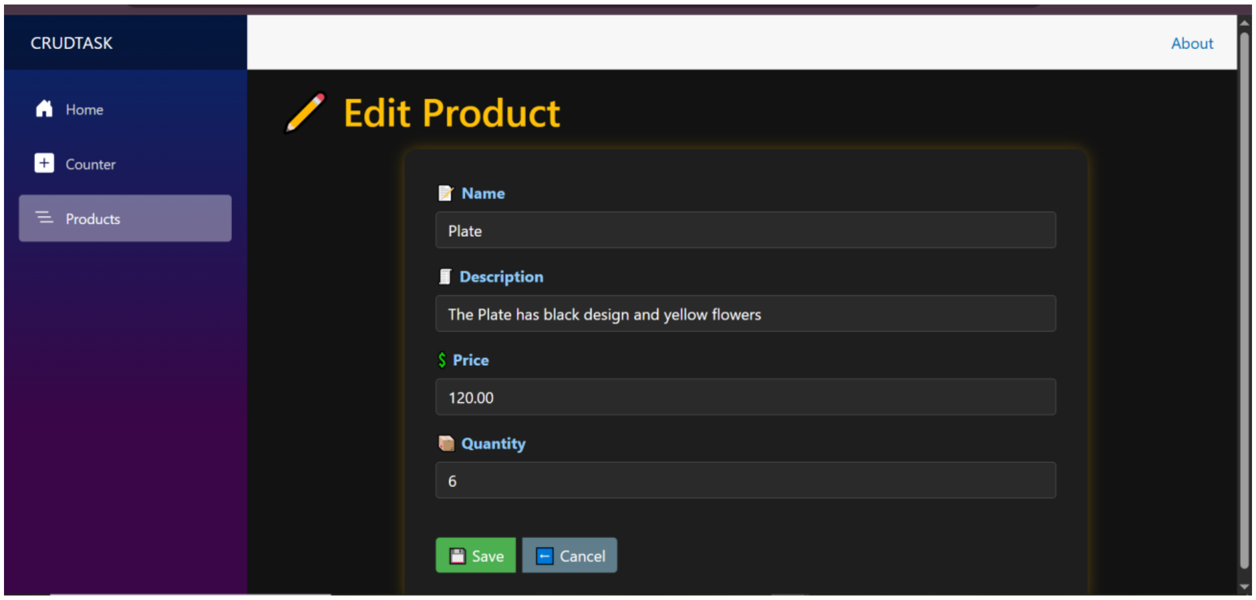
Name:
Plate

Description:
The Plate has black design and yellow flowers

Price:
120

Quantity:
6

EDIT:



Edit Product

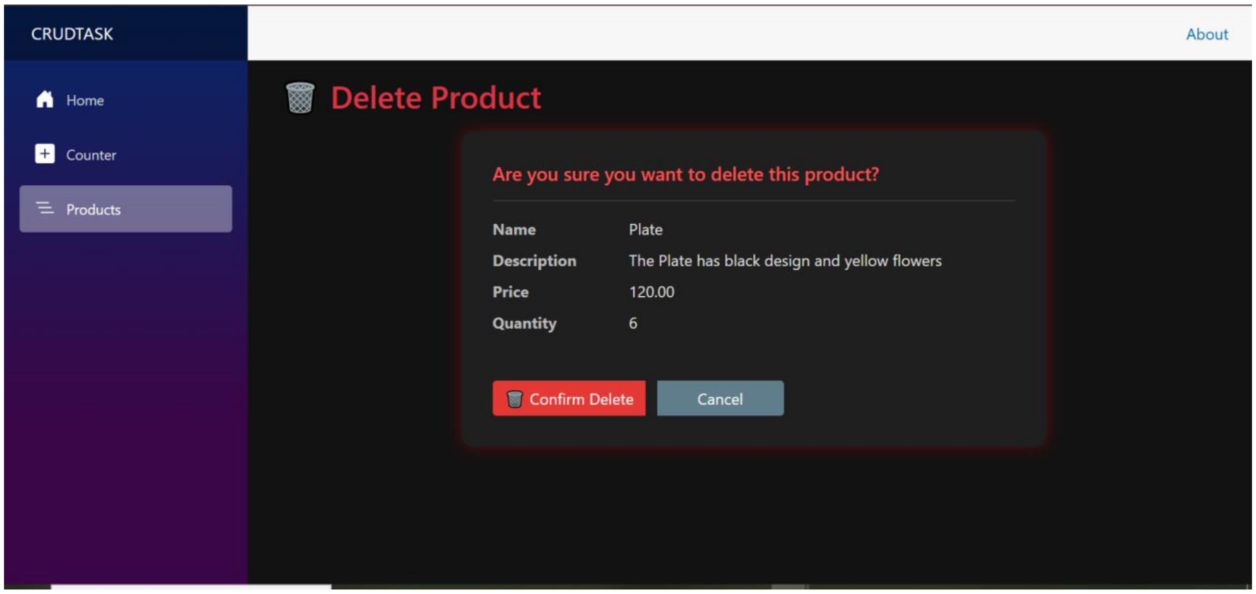
Name
Plate

Description
The Plate has black design and yellow flowers

Price
120.00

Quantity
6

DELETE:

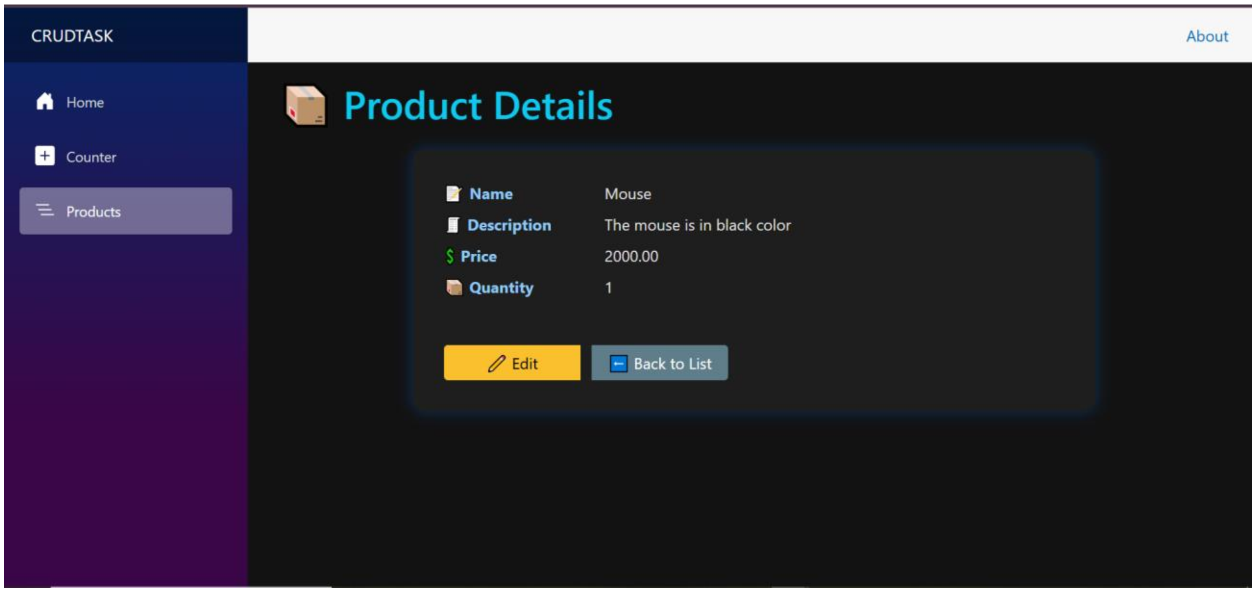


Delete Product

Are you sure you want to delete this product?

Name	Plate
Description	The Plate has black design and yellow flowers
Price	120.00
Quantity	6

DETAILS:



Product Details

Name	Mouse
Description	The mouse is in black color
Price	2000.00
Quantity	1

CODE:

INDEX.RAZOR

```
@page "/products"
@using Microsoft.EntityFrameworkCore
@using Microsoft.AspNetCore.Components.QuickGrid
@using CRUDTASK.Models
@using CRUDTASK.Data
@implements IAsyncDisposable
@inject IDbContextFactory<CRUDTASKContext> DbFactory

<PageTitle>Products</PageTitle>

<h1 class="page-title">🛒 Product Management</h1>

<p class="create-link">
    <a class="btn-create" href="products/create">+ Create New Product</a>
</p>

<QuickGrid Class="dark-table" Items="context.Products">
    <PropertyColumn Property="products => products.Name" Title="Product Name" />
    <PropertyColumn Property="products => products.Description" Title="Description"
/>
    <PropertyColumn Property="products => products.Price" Title="Price ($)" />
    <PropertyColumn Property="products => products.Quantity" Title="Quantity" />

    <TemplateColumn Title="Actions" Context="products">
        <a class="action-link" href="@($"products/edit?id={products.Id}")">✏️
Edit</a>
        <a class="action-link" href="@($"products/details?id={products.Id}")">🔍
Details</a>
        <a class="action-link delete"
href="@($"products/delete?id={products.Id}")">🗑 Delete</a>
    </TemplateColumn>
</QuickGrid>

<style>
    body {
        background-color: #121212;
        color: #e0e0e0;
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    }

    .page-title {
        color: #ffffff;
        font-size: 2rem;
        margin-bottom: 1rem;
        border-bottom: 2px solid #444;
        padding-bottom: 0.5rem;
    }

    .create-link {
        margin-bottom: 1.5rem;
    }

    .btn-create {
        background-color: #1e88e5;
        color: #fff;
```

```
            padding: 0.5rem 1rem;
            text-decoration: none;
            border-radius: 6px;
            transition: background-color 0.3s ease;
        }

            .btn-create:hover {
                background-color: #1565c0;
            }

        .dark-table {
            width: 100%;
            border-collapse: collapse;
            background-color: #1e1e1e;
            border: 1px solid #333;
        }

            .dark-table th, .dark-table td {
                padding: 12px;
                text-align: left;
                border-bottom: 1px solid #333;
            }

            .dark-table th {
                background-color: #292929;
                color: #ffffff;
            }

            .dark-table tr:hover {
                background-color: #2e2e2e;
            }

        .action-link {
            margin-right: 10px;
            color: #64b5f6;
            text-decoration: none;
            font-weight: 500;
        }

            .action-link:hover {
                color: #90caf9;
                text-decoration: underline;
            }

            .action-link.delete {
                color: #ef5350;
            }

                .action-link.delete:hover {
                    color: #ff867c;
                }
    </style>

    @code {
        private CRUDTASKContext context = default!;

        protected override void OnInitialized()
        {
```

```
        context = DbFactory.CreateDbContext();
    }

    public async ValueTask DisposeAsync() => await context.DisposeAsync();
}
```

CREATE.Razor

```
@page "/products/create"
@using Microsoft.EntityFrameworkCore
@using CRUDTASK.Models
@inject IDbContextFactory<CRUDTASK.Data.CRUDTASKContext> DbFactory
@inject NavigationManager NavigationManager

<PageTitle>Create Product</PageTitle>

<h1 class="title">➕ Add New Product</h1>
<hr class="divider" />

<div class="form-container">
    <EditForm Model="Products" OnValidSubmit="AddProducts">
        <DataAnnotationsValidator />
        <ValidationSummary class="text-danger" role="alert" />

        <div class="form-group">
            <label for="name">Name:</label>
            <InputText id="name" @bind-Value="Products.Name" class="form-control
dark-input" />
            <ValidationMessage For="() => Products.Name" class="text-danger small-
msg" />
        </div>

        <div class="form-group">
            <label for="description">Description:</label>
            <InputText id="description" @bind-Value="Products.Description"
class="form-control dark-input" />
            <ValidationMessage For="() => Products.Description" class="text-danger
small-msg" />
        </div>

        <div class="form-group">
            <label for="price">Price:</label>
            <InputNumber id="price" @bind-Value="Products.Price" class="form-control
dark-input" />
            <ValidationMessage For="() => Products.Price" class="text-danger small-
msg" />
        </div>

        <div class="form-group">
            <label for="quantity">Quantity:</label>
            <InputNumber id="quantity" @bind-Value="Products.Quantity" class="form-
control dark-input" />
            <ValidationMessage For="() => Products.Quantity" class="text-danger
small-msg" />
        </div>

        <button type="submit" class="btn btn-success">✅ Create Product</button>
    </EditForm>
</div>
```

```
<div class="back-link">
    <a href="/products">← Back to Product List</a>
</div>

<style>
    body {
        background-color: #121212;
        color: #e0e0e0;
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    }

    .title {
        font-size: 2rem;
        margin-bottom: 1rem;
        color: #ffffff;
    }

    .divider {
        border: 0;
        border-top: 2px solid #333;
        margin-bottom: 2rem;
    }

    .form-container {
        max-width: 500px;
        margin: auto;
        background-color: #1e1e1e;
        padding: 2rem;
        border-radius: 10px;
        box-shadow: 0 0 10px rgba(0,0,0,0.5);
    }

    .form-group {
        margin-bottom: 1.5rem;
    }

        .form-group label {
            display: block;
            font-weight: 600;
            margin-bottom: 0.5rem;
        }

    .dark-input {
        background-color: #2c2c2c;
        color: #ffffff;
        border: 1px solid #444;
        padding: 0.5rem;
        border-radius: 5px;
    }

        .dark-input:focus {
            outline: none;
            border-color: #64b5f6;
            box-shadow: 0 0 5px #64b5f6;
        }

    .small-msg {
        font-size: 0.85rem;
    }

    .btn-success {
        background-color: #43a047;
        border: none;
        padding: 0.6rem 1.2rem;
        font-weight: 600;
        border-radius: 6px;
        transition: background-color 0.3s;
    }

        .btn-success:hover {
            background-color: #2e7d32;
        }

    .back-link {
        text-align: center;
        margin-top: 2rem;
    }

        .back-link a {
            color: #90caf9;
            text-decoration: none;
        }

            .back-link a:hover {
                text-decoration: underline;
            }
</style>

@code {
    [SupplyParameterFromForm]
    private Products Products { get; set; } = new();

    private async Task AddProducts()
    {
        using var context = DbFactory.CreateDbContext();
        context.Products.Add(Products);
        await context.SaveChangesAsync();
        NavigationManager.NavigateTo("/products");
    }
}
```

EDIT.razor

```
@page "/products/edit"
@using Microsoft.EntityFrameworkCore
@using CRUDTASK.Models
@inject IDbContextFactory<CRUDTASK.Data.CRUDTASKContext> DbFactory
@inject NavigationManager NavigationManager

<PageTitle>Edit Product</PageTitle>

<h1 class="text-warning mb-3">✏ Edit Product</h1>

@if (Products is null)
{
    <p class="loading"><em>Loading product data...</em></p>
}
else
```

```
{
    <div class="edit-card">
        <EditForm method="post" Model="Products" OnValidSubmit="UpdateProducts"
FormName="edit" Enhance>
            <DataAnnotationsValidator />
            <ValidationSummary class="text-danger" />

            <input type="hidden" name="Products.Id" value="@Products.Id" />

            <div class="mb-3">
                <label for="name" class="form-label">📝 Name</label>
                <InputText id="name" @bind-Value="Products.Name" class="form-control
dark-input" />
                <ValidationMessage For="() => Products.Name" class="text-danger" />
            </div>

            <div class="mb-3">
                <label for="description" class="form-label">📜 Description</label>
                <InputText id="description" @bind-Value="Products.Description"
class="form-control dark-input" />
                <ValidationMessage For="() => Products.Description" class="text-
danger" />
            </div>

            <div class="mb-3">
                <label for="price" class="form-label">💲 Price</label>
                <InputNumber id="price" @bind-Value="Products.Price" class="form-
control dark-input" />
                <ValidationMessage For="() => Products.Price" class="text-danger" />
            </div>

            <div class="mb-3">
                <label for="quantity" class="form-label">📦 Quantity</label>
                <InputNumber id="quantity" @bind-Value="Products.Quantity"
class="form-control dark-input" />
                <ValidationMessage For="() => Products.Quantity" class="text-danger"
/>
            </div>

            <div class="btn-group mt-4">
                <button type="submit" class="btn btn-success me-2">💾 Save</button>
                <a href="/products" class="btn btn-secondary">⬅ Cancel</a>
            </div>
        </EditForm>
    </div>
}

<style>
    body {
        background-color: #121212;
        color: #e0e0e0;
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    }

    .loading {
        text-align: center;
        font-size: 1.2rem;
        margin-top: 2rem;
    }

    .edit-card {
        background-color: #1e1e1e;
        padding: 2rem;
        border-radius: 12px;
        box-shadow: 0 0 15px rgba(255, 193, 7, 0.3);
        max-width: 700px;
        margin: auto;
    }

    .form-label {
        font-weight: bold;
        color: #90caf9;
    }

    .dark-input {
        background-color: #2a2a2a;
        color: #fff;
        border: 1px solid #555;
    }

        .dark-input:focus {
            background-color: #333;
            border-color: #90caf9;
            box-shadow: 0 0 5px rgba(144, 202, 249, 0.5);
        }

    .btn-success {
        background-color: #4caf50;
        border: none;
    }

        .btn-success:hover {
            background-color: #388e3c;
        }

    .btn-secondary {
        background-color: #607d8b;
        border: none;
    }

        .btn-secondary:hover {
            background-color: #455a64;
        }
</style>

@code {
    [SupplyParameterFromQuery]
    private int Id { get; set; }

    [SupplyParameterFromForm]
    private Products? Products { get; set; }

    protected override async Task OnInitializedAsync()
    {
        using var context = DbFactory.CreateDbContext();
        Products ??= await context.Products.FirstOrDefaultAsync(m => m.Id == Id);

        if (Products is null)
        {
            NavigationManager.NavigateTo("notfound");
        }
    }

    private async Task UpdateProducts()
    {
        using var context = DbFactory.CreateDbContext();
        context.Attach(Products!).State = EntityState.Modified;

        try
        {
            await context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!ProductsExists(Products!.Id))
            {
                NavigationManager.NavigateTo("notfound");
            }
            else
            {
                throw;
            }
        }

        NavigationManager.NavigateTo("/products");
    }

    private bool ProductsExists(int id)
    {
        using var context = DbFactory.CreateDbContext();
        return context.Products.Any(e => e.Id == id);
    }
}
```

DETAILS.razor

```
@page "/products/details"
@using Microsoft.EntityFrameworkCore
@using CRUDTASK.Models
@inject IDbContextFactory<CRUDTASK.Data.CRUDTASKContext> DbFactory
@inject NavigationManager NavigationManager

<PageTitle>Product Details</PageTitle>

<h1 class="text-info mb-4">📦 Product Details</h1>

@if (products is null)
{
    <p class="loading"><em>Loading product details...</em></p>
}
```

```
else
{
    <div class="details-card">
        <dl class="row">
            <dt class="col-sm-3">📝 Name</dt>
            <dd class="col-sm-9">@products.Name</dd>

            <dt class="col-sm-3">📜 Description</dt>
            <dd class="col-sm-9">@products.Description</dd>

            <dt class="col-sm-3">💲 Price</dt>
            <dd class="col-sm-9">@products.Price</dd>

            <dt class="col-sm-3">📦 Quantity</dt>
            <dd class="col-sm-9">@products.Quantity</dd>
        </dl>

        <div class="mt-4 btn-group">
            <a href="@($"/products/edit?id={products.Id}")" class="btn btn-
warning">🖊 Edit</a>
            <a href="/products" class="btn btn-secondary">← Back to List</a>
        </div>
    </div>
}

<style>
    body {
        background-color: #121212;
        color: #e0e0e0;
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    }

    .loading {
        text-align: center;
        font-size: 1.2rem;
        margin-top: 2rem;
    }

    .details-card {
        background-color: #1e1e1e;
        padding: 2rem;
        border-radius: 12px;
        box-shadow: 0 0 15px rgba(0, 123, 255, 0.3);
        max-width: 700px;
        margin: auto;
    }

    dl dt {
        color: #90caf9;
        font-weight: bold;
    }

    .btn-group .btn {
        margin-right: 0.8rem;
        min-width: 140px;
    }
    .btn-warning {
        background-color: #fbc02d;
        border: none;
    }

        .btn-warning:hover {
            background-color: #f9a825;
        }

    .btn-secondary {
        background-color: #607d8b;
        border: none;
    }

        .btn-secondary:hover {
            background-color: #455a64;
        }
</style>

@code {
    private Products? products;

    [SupplyParameterFromQuery]
    private int Id { get; set; }

    protected override async Task OnInitializedAsync()
    {
        using var context = DbFactory.CreateDbContext();
        products = await context.Products.FirstOrDefaultAsync(m => m.Id == Id);

        if (products is null)
        {
            NavigationManager.NavigateTo("notfound");
        }
    }
}
```

DELETE.razor

```
@page "/products/delete"
@using Microsoft.EntityFrameworkCore
@using CRUDTASK.Models
@inject IDbContextFactory<CRUDTASK.Data.CRUDTASKContext> DbFactory
@inject NavigationManager NavigationManager

<PageTitle>Delete Product</PageTitle>

<h1 class="title text-danger">🗑️ Delete Product</h1>

@if (products is null)
{
    <p class="loading"><em>Loading product details...</em></p>
}
else
{
    <div class="delete-card">
        <h4 class="confirm-text">Are you sure you want to delete this product?</h4>
        <hr />
        <dl class="row">
            <dt class="col-sm-3">Name</dt>
            <dd class="col-sm-9">@products.Name</dd>

            <dt class="col-sm-3">Description</dt>
            <dd class="col-sm-9">@products.Description</dd>

            <dt class="col-sm-3">Price</dt>
            <dd class="col-sm-9">@products.Price</dd>

            <dt class="col-sm-3">Quantity</dt>
            <dd class="col-sm-9">@products.Quantity</dd>
        </dl>

        <EditForm Model="products" OnValidSubmit="DeleteProducts">
            <div class="btn-group mt-4">
                <button type="submit" class="btn btn-danger">🗑️ Confirm
Delete</button>
                <a href="/products" class="btn btn-secondary">Cancel</a>
            </div>
        </EditForm>
    </div>
}

<style>
    body {
        background-color: #121212;
        color: #e0e0e0;
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    }

    .title {
        font-size: 2rem;
        margin-bottom: 1rem;
    }

    .loading {
        text-align: center;
        font-size: 1.2rem;
        margin-top: 2rem;
    }

    .delete-card {
        max-width: 600px;
        margin: auto;
        background-color: #1f1f1f;
        padding: 2rem;
        border-radius: 12px;
        box-shadow: 0 0 15px rgba(255, 0, 0, 0.3);
    }

    .confirm-text {
        font-size: 1.2rem;
        margin-bottom: 1rem;
        color: #ff5252;
    }

    dl dt {
        color: #bbb;
    }

    .btn-group .btn {
        margin-right: 0.8rem;
        min-width: 130px;
    }

    .btn-danger {
        background-color: #e53935;
        border: none;
    }

        .btn-danger:hover {
            background-color: #c62828;
        }

    .btn-secondary {
        background-color: #607d8b;
        border: none;
    }

        .btn-secondary:hover {
            background-color: #455a64;
        }
</style>

@code {
    private Products? products;

    [SupplyParameterFromQuery]
    private int Id { get; set; }

    protected override async Task OnInitializedAsync()
    {
        using var context = DbFactory.CreateDbContext();
        products = await context.Products.FirstOrDefaultAsync(m => m.Id == Id);

        if (products is null)
        {
            NavigationManager.NavigateTo("notfound");
        }
    }

    private async Task DeleteProducts()
    {
        using var context = DbFactory.CreateDbContext();
        context.Products.Remove(products!);
        await context.SaveChangesAsync();
        NavigationManager.NavigateTo("/products");
    }
}
```

Products.cs

```
namespace CRUDTASK.Models
{
    public class Products
    {
        public int Id { get; set; }
        public string Name { get; set; } = string.Empty;
        public string Description { get; set; } = string.Empty;
        public decimal Price { get; set; }
        public int Quantity { get; set; }
    }
}
```