Evidence Exam: ASP.NET Core MVC with view component Master Details CRUD Operation in a Signal Page

Project Name: Product Master Details CRUD

Explanation:

1. **ASP.NET Core Web App (Model View Controller)**
2. **View Component**
3. **Master Details**
4. **CRUD Operation**
5. **Signal Page**

Step 1: Creating the ASP.NET Core MVC Project

- A new **ASP.NET Core Web Application (MVC)** is created.

Step 2: Installing Required NuGet Packages

1. `Microsoft.EntityFrameworkCore.SqlServer`
2. `Microsoft.EntityFrameworkCore.Tools`
3. `EntityFrameWork.Core`
4. `Microsoft.EntityFrameworkCore.Design`
5. `Microsoft.VisualStudio.Web.CodeGeneration.Design`
6. `jQuery`

Step 3: Database Configuration (appsettings.json)

- **Connection Strings** are added to connect to **SQL Server** or **LocalDB**.

```
//Localdb
"AllowedHosts": "*",
  "ConnectionStrings": {
    "appCon": "Data Source=(localdb)\\MSSQLLocalDB;Initial
Catalog=PrductMasterDetails_CRUD_DB;Integrated Security=True;Connect Timeout=30;Encrypt=False;Trust
Server Certificate=False;Application Intent=ReadWrite;Multi Subnet Failover=False"
  }
}

//Or sql
"AllowedHosts": "*", "ConnectionStrings": { "EmployeeDB": "Server=
DESKTOPES8G333;Database=DatabseName; Trusted_Connection=True;
MultipleActiveResultSets=True;" }


For Password:
```

"ConnectionStrings": { "DefaultConnection": "Server=EnterServerName; Database=DatabaseName; User Id=sa; Password=EnterPassword; Trusted_Connection=True; MultipleActiveResultSets=true" },

Step 4: Creating Models

- **`Product.cs`**:

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace ProductMatarDetails_CRUD.Models
{
    public class Product
    {
        [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int PId { get; set; }
        public string PName { get; set; }
        public int Price { get; set; }
        public string? PImage { get; set; }
        public int Quantity { get; set; }
        public DateTime PDate { get; set; }
        public bool IsAvailable { get; set; }
        public virtual IList<Details>? Details { get; set; }
    }
}
```

`Category.cs`:

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace ProductMatarDetails_CRUD.Models
{
    public class Category
    {
        [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int CId { get; set; }
        public string CName { get; set; }
        public virtual IList<Details>? Details { get; set; }
    }
}
```

`Details.cs`:

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace ProductMatarDetails_CRUD.Models
{
    public class Details
    {
```

```
        [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int DId { get; set; }
        [ForeignKey("Product")]
        public int PId { get; set; }
        [ForeignKey("Category")]
        public int CId { get; set; }

        public virtual Product? Product { get; set; }
        public virtual Category? Category { get; set; }

    }
}
```

Step 5: VModel under the Models Folder

VModel.cs

```
namespace ProductMatarDetails_CRUD.Models.VModel
{
    public class VModel
    {
        public VModel()
        {
            this.Details = new List<Details>();
        }
        public int PId { get; set; }
        public string PName { get; set; }
        public int Price { get; set; }
        public string? PImage { get; set; }
        public IFormFile? ImageFile { get; set; }
        public int Quantity { get; set; }
        public DateTime PDate { get; set; }
        public bool IsAvailable { get; set; }
        public virtual List<Details>? Details { get; set; }
    }
}
```

Step 6: Data Folder under the Project

- **ProductDbContext.cs**

```
using Microsoft.EntityFrameworkCore;
using ProductMatarDetails_CRUD.Models;

namespace ProductMatarDetails_CRUD.Data
{
    public class ProductDbContext:DbContext
    {
        public ProductDbContext(DbContextOptions<ProductDbContext>options) :
base(options)
```

```
        {

        }

        public DbSet<Product> Products { get; set; }
        public DbSet<Category> Categories { get; set; }
        public DbSet<Details> Details { get; set; }

    }
}
```

Step 7: Registering DbContext in `Program.cs -5`

```
var CString = builder.Configuration.GetConnectionString("appCon");
builder.Services.AddDbContext<ProductDbContext>(opt =>
{
    opt.UseSqlServer(CString);
});
```

Change the ControllerRoute,
Home to Product

Step 8: Database Migration

- Using **Package Manager Console (PMC)**:

```
add-migration "ScriptA"
update-database

After that Input some category in the Category Table
```

Step 9: ProductController

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using ProductMatarDetails_CRUD.Data;
using ProductMatarDetails_CRUD.Models.VModel;
using ProductMatarDetails_CRUD.Models;

namespace ProductMatarDetails_CRUD.Controllers
{
    public class ProductController : Controller
    {
        private ProductDbContext _context;
        private IWebHostEnvironment _env;
        public ProductController(ProductDbContext context, IWebHostEnvironment env)
        {
            _context = context;
            _env = env;
        }
```

```
        public async Task<ActionResult> Index()
        {
            var products = await _context.Products.ToListAsync();
            return View(products);
        }

        public IActionResult Details(int id)
        {
            var product = _context.Products
                .Include(p => p.Details)
                .ThenInclude(d => d.Category)
                .FirstOrDefault(p => p.PId == id);

            if (product == null)
            {
                return NotFound();
            }
            return PartialView("Details", product);
            //return View(product);
        }


        public IActionResult Create()
        {
            // Load categories for dropdown
            ViewBag.Categories = new SelectList(_context.Categories, "CId",
"CName");
            return PartialView("Create");
            //return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create(VModel vModel, int[] CId)
        {
            if (ModelState.IsValid)
            {
                Product product = new Product
                {
                    PName = vModel.PName,
                    PDate = vModel.PDate,
                    IsAvailable = vModel.IsAvailable,
                    Price = vModel.Price,
                    Quantity = vModel.Quantity,
                };

                if (vModel.ImageFile != null)
                {
                    var file = DateTime.Now.Ticks.ToString() +
Path.GetExtension(vModel.ImageFile.FileName);
                    var fileName = _env.WebRootPath + "/Images/" + file;
                    using (var stream = System.IO.File.Create(fileName))
                    {
                        await vModel.ImageFile.CopyToAsync(stream);
                    }
                    product.PImage = "/Images/" + file;
                }
```

```
                _context.Add(product);
                await _context.SaveChangesAsync();

                if (CId != null && CId.Length > 0)
                {
                    foreach (var cId in CId.Where(id => id > 0))
                    {
                        Details details = new Details
                        {
                            PId = product.PId,
                            CId = cId
                        };
                        _context.Add(details);
                    }
                    await _context.SaveChangesAsync();
                }

                return RedirectToAction(nameof(Index));
            }

            // If we got this far, something failed, reload categories
            ViewBag.Categories = new SelectList(_context.Categories, "CId",
"CName");

            return View(vModel);
        }


        public IActionResult Edit(int id)
        {
            var product = _context.Products
                .Include(p => p.Details)
                .FirstOrDefault(p => p.PId == id);

            if (product == null)
            {
                return NotFound();
            }

            VModel vModel = new VModel
            {
                PId = product.PId,
                PName = product.PName,
                PDate = product.PDate,
                IsAvailable = product.IsAvailable,
                Price = product.Price,
                Quantity = product.Quantity,
                PImage = product.PImage
            };

            // Load categories for dropdown
            ViewBag.Categories = new SelectList(_context.Categories, "CId",
"CName");
            ViewBag.SelectedCategories = product.Details.Select(d =>
d.CId).ToList();

            return PartialView("Edit", vModel);
```

```
            //return View(vModel);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id, VModel vModel, int[] CId)
        {
            if (id != vModel.PId)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    var product = await _context.Products.FindAsync(id);
                    if (product == null)
                    {
                        return NotFound();
                    }

                    product.PName = vModel.PName;
                    product.PDate = vModel.PDate;
                    product.IsAvailable = vModel.IsAvailable;
                    product.Price = vModel.Price;
                    product.Quantity = vModel.Quantity;

                    if (vModel.ImageFile != null)
                    {
                        var file = DateTime.Now.Ticks.ToString() +
Path.GetExtension(vModel.ImageFile.FileName);
                        var fileName = _env.WebRootPath + "/Images/" + file;
                        using (var stream = System.IO.File.Create(fileName))
                        {
                            await vModel.ImageFile.CopyToAsync(stream);
                        }

                        // Delete old image if exists
                        if (!string.IsNullOrEmpty(product.PImage))
                        {
                            var oldImagePath = _env.WebRootPath + product.PImage;
                            if (System.IO.File.Exists(oldImagePath))
                            {
                                System.IO.File.Delete(oldImagePath);
                            }
                        }

                        product.PImage = "/Images/" + file;
                    }

                    // Update categories
                    var existingDetails = _context.Details.Where(d => d.PId ==
id).ToList();
                    _context.Details.RemoveRange(existingDetails);

                    if (CId != null && CId.Length > 0)
                    {
```

```
                        foreach (var cId in CId.Where(cid => cid > 0))
                        {
                            Details details = new Details
                            {
                                PId = product.PId,
                                CId = cId
                            };
                            _context.Add(details);
                        }
                    }

                    _context.Update(product);
                    await _context.SaveChangesAsync();

                    return RedirectToAction(nameof(Index));
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!ProductExists(vModel.PId))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
            }

            // If we got this far, something failed, reload categories
            ViewBag.Categories = new SelectList(_context.Categories, "CId",
"CName");
            ViewBag.SelectedCategories = _context.Details.Where(d => d.PId ==
id).Select(d => d.CId).ToList();
            return View(vModel);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Delete(int id)
        {
            var product = await _context.Products
                .Include(p => p.Details)
                .FirstOrDefaultAsync(p => p.PId == id);

            if (product == null)
            {
                return NotFound();
            }

            // Delete image if exists
            if (!string.IsNullOrEmpty(product.PImage))
            {
                var imagePath = _env.WebRootPath + product.PImage;
                if (System.IO.File.Exists(imagePath))
                {
                    System.IO.File.Delete(imagePath);
                }
```

```
            }

            _context.Products.Remove(product);
            await _context.SaveChangesAsync();

            return RedirectToAction(nameof(Index));
        }

        public IActionResult AddCategory(int index)
        {
            ViewBag.Categories = new SelectList(_context.Categories, "CId",
"CName");
            return ViewComponent("CategoryManagement", new { index = index });
        }

        private bool ProductExists(int id)
        {
            return _context.Products.Any(e => e.PId == id);
        }
    }

}
```

Step 10: Index View (`Index.cshtml`)

```
@model IEnumerable<ProductMatarDetails_CRUD.Models.Product>

<h1 class="text-center">Product List</h1>

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

@if (Model.Any())
{
    <table class="table">
        <thead>
            <tr>
                <th>Product Name</th>
                <th>Price</th>
                <th>Image</th>
                <th>Quantity</th>
                <th>Date</th>
                <th>Available</th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var item in Model)
            {
                <tr>
```

```
                    <td>@item.PName</td>
                    <td>@item.Price</td>
                    <td>
                        @if (!string.IsNullOrEmpty(item.PImage))
                        {
                            <img src="@item.PImage" class="rounded-circle"
alt="@item.PName" style="max-width: 60px; max-height: 60px;" />
                        }
                    </td>
                    <td>@item.Quantity</td>
                    <td>@item.PDate.ToShortDateString()</td>
                    <td>@(item.IsAvailable ? "Yes" : "No")</td>
                    <td>
                        <button class="btn btn-warning detailsBtn" type="button"
data-id="@item.PId">
                            Details
                        </button>
                        <button class="btn btn-warning editBtn" type="button" data-
id="@item.PId">
                            Edit
                        </button>
                        <form asp-action="Delete" asp-route-id="@item.PId"
method="post" style="display:inline">
                            <button type="submit" class="btn btn-danger btn-sm"
onclick="return confirm('Are you sure you want to delete this item?');">
                                Delete
                            </button>
                        </form>
                    </td>
                </tr>
            }
        </tbody>
    </table>
}
else
{
    <div class="alert alert-info">No products available.</div>
}

<section class="mcon" style="display:none;">
    <div class="mbod p-5">
        <div class="macon"></div>
        <div class="modal-footer">
            <button class="btn btn-danger clbtn">Close</button>
        </div>
    </div>
</section>
```

Step 11: Details Views

Details.cshtml

```
@model ProductMatarDetails_CRUD.Models.Product

<div class="container mt-4">
    <h1 class="text-center">Product Details</h1>
    <div class="row">
```

```
        <div class="col-md-6">
            <div class="card">
                <div class="card-header">
                    <h4>@Model.PName</h4>
                </div>
                <div class="card-body">
                    @if (!string.IsNullOrEmpty(Model.PImage))
                    {
                        <img src="@Model.PImage" class="img-fluid mb-3"
alt="@Model.PName" />
                    }

                    <table class="table">
                        <tr>
                            <th>Price</th>
                            <td>@Model.Price</td>
                        </tr>
                        <tr>
                            <th>Quantity</th>
                            <td>@Model.Quantity</td>
                        </tr>
                        <tr>
                            <th>Date</th>
                            <td>@Model.PDate.ToShortDateString()</td>
                        </tr>
                        <tr>
                            <th>Available</th>
                            <td>@(Model.IsAvailable ? "Yes" : "No")</td>
                        </tr>
                    </table>
                </div>
            </div>
        </div>
        <div class="col-md-6">
            <div class="card">
                <div class="card-header">
                    <h4>Categories</h4>
                </div>
                <div class="card-body">
                    @await Component.InvokeAsync("CategoryManagement", new {
productId = Model.PId, mode = "display" })
                </div>
            </div>
        </div>
    </div>
    <div class="mt-3">
        <a asp-action="Edit" asp-route-id="@Model.PId" class="btn btn-
primary">Edit</a>
        <a asp-action="Index" class="btn btn-secondary">Back to List</a>
    </div>
</div>
```

Step 12: Create Views

Create.cshtml

```
@model ProductMatarDetails_CRUD.Models.VModel.VModel
<h4 class="text-center">Create Product</h4>
<hr />
<div class="row">
    <div class="col-md-12">
        <form asp-action="Create" method="post" enctype="multipart/form-data">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="PName" class="control-label"></label>
                <input asp-for="PName" class="form-control" />
                <span asp-validation-for="PName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Price" class="control-label"></label>
                <input asp-for="Price" class="form-control" />
                <span asp-validation-for="Price" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="ImageFile" class="control-label"></label>
                <input asp-for="ImageFile" class="form-control" type="file" />
                <span asp-validation-for="ImageFile" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Quantity" class="control-label"></label>
                <input asp-for="Quantity" class="form-control" />
                <span asp-validation-for="Quantity" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="PDate" class="control-label"></label>
                <input asp-for="PDate" class="form-control" type="date" />
                <span asp-validation-for="PDate" class="text-danger"></span>
            </div>
            <div class="form-group form-check">
                <label class="form-check-label">
                    <input class="form-check-input" asp-for="IsAvailable" />
                    @Html.DisplayNameFor(model => model.IsAvailable)
                </label>
            </div>

            <div class="my-3">
                <h5>Product Categories</h5>
                <hr />
                <div>
                    <button type="button" class="btn btn-info addMore">
                        Add Category
                    </button>
                </div>
                <div class="categories-container mt-2">
                    @await Component.InvokeAsync("CategoryManagement", new { mode =
"create", index = 0 })
                </div>
            </div>

            <div class="form-group mt-3">
                <input type="submit" value="Create" class="btn btn-primary" />
                <a asp-action="Index" class="btn btn-secondary">Back to List</a>
            </div>
        </form>
```

```
    </div>
</div>

<script>
    $(document).ready(() => {
        var index = 1;
        $(".addMore").click(() => {
            $.ajax({
                url: "/Product/AddCategory",
                type: "get",
                data: { index: index },
                success: (d) => {
                    $(".categories-container").append(d);
                    index++;
                },
            });
        });
    });
    $(document).on("click", ".remove", (e) => {
        $(e.currentTarget).closest(".rc").remove();
    });

</script>
```

Step 13: Create Views

Edit.cshtml

```
@model ProductMatarDetails_CRUD.Models.VModel.VModel
<h4 class="text-center">Edit Product</h4>
<hr />
<div class="row">
    <div class="col-md-12">
        <form asp-action="Edit" method="post" enctype="multipart/form-data">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="PId" />
            <input type="hidden" asp-for="PImage" />

            <div class="form-group">
                <label asp-for="PName" class="control-label"></label>
                <input asp-for="PName" class="form-control" />
                <span asp-validation-for="PName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Price" class="control-label"></label>
                <input asp-for="Price" class="form-control" />
                <span asp-validation-for="Price" class="text-danger"></span>
            </div>

            <div class="form-group">
                <label asp-for="ImageFile" class="control-label"></label>
                <input asp-for="ImageFile" class="form-control" type="file" />
                <span asp-validation-for="ImageFile" class="text-danger"></span>
            </div>
```

```
            @if (!string.IsNullOrEmpty(Model.PImage))
            {
                <div class="form-group">
                    <label>Current Image</label>
                    <div>
                        <img src="@Model.PImage" class="rounded-circle" style="max-
width: 200px; max-height: 200px;" />
                    </div>
                </div>
            }

            <div class="form-group">
                <label asp-for="Quantity" class="control-label"></label>
                <input asp-for="Quantity" class="form-control" />
                <span asp-validation-for="Quantity" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="PDate" class="control-label"></label>
                <input asp-for="PDate" class="form-control" type="date" />
                <span asp-validation-for="PDate" class="text-danger"></span>
            </div>
            <div class="form-group form-check">
                <label class="form-check-label">
                    <input class="form-check-input" asp-for="IsAvailable" />
                    @Html.DisplayNameFor(model => model.IsAvailable)
                </label>
            </div>

            <div class="my-3">
                <h5>Product Categories</h5>
                <hr />
                <div>
                    <button type="button" class="btn btn-info addMore">
                        Add Category
                    </button>
                </div>
                <div class="categories-container mt-2">
                    @{
                        var selectedCategories = ViewBag.SelectedCategories as
List<int> ?? new List<int>();
                        for (int i = 0; i < selectedCategories.Count; i++)
                        {
                            @await Component.InvokeAsync("CategoryManagement", new {
productId = Model.PId, mode = "edit", index = i })
                        }

                        if (selectedCategories.Count == 0)
                        {
                            @await Component.InvokeAsync("CategoryManagement", new {
mode = "create", index = 0 })
                        }
                    }
                </div>
            </div>

            <div class="form-group mt-3">
                <input type="submit" value="Save" class="btn btn-primary" />
```

```
                <a asp-action="Index" class="btn btn-secondary">Back to List</a>
            </div>
        </form>
    </div>
</div>

<script>
       $(document).ready(() => {
        var index = @(ViewBag.SelectedCategories != null ?
((List<int>)ViewBag.SelectedCategories).Count : 1);

        // Add more categories
        $(".addMore").click(() => {
            $.ajax({
                url: "/Product/AddCategory",
                type: "get",
                data: { index: index },
                success: (d) => {
                    $(".categories-container").append(d);
                    index++;
                },
            });
        });

        // Initialize selected category values (if in edit mode)
        if ($(".categories-container select").length > 0) {
            // This will be handled by the inline script in each CategoryManagement
component
            // No additional code needed here
        }
    });

    // Remove category item
    $(document).on("click", ".remove", (e) => {
        $(e.currentTarget).closest(".rc").remove();
    });

</script>
```

Step 14: ViewComponent folder under the Project

CategoryManagementViewComponent.cs

```
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.AspNetCore.Mvc;
using ProductMatarDetails_CRUD.Data;
using ProductMatarDetails_CRUD.Models;
using Microsoft.EntityFrameworkCore;

namespace ProductMatarDetails_CRUD.ViewComponents
{
    public class CategoryManagementViewComponent : ViewComponent
    {
```

```
        private readonly ProductDbContext _context;

        public CategoryManagementViewComponent(ProductDbContext context)
        {
            _context = context;
        }

        public IViewComponentResult Invoke(int? productId = null, string mode =
"create", int index = 0)
        {
            // Get all categories as SelectList
            var categories = new SelectList(_context.Categories, "CId", "CName");
            ViewBag.Categories = categories;
            ViewBag.Index = index;
            ViewBag.Mode = mode;

            // If in display mode and productId is provided, get selected categories
            if (mode == "display" && productId.HasValue)
            {
                var productCategories = _context.Details
                    .Include(d => d.Category)
                    .Where(d => d.PId == productId.Value)
                    .Select(d => d.Category)
                    .ToList();
                return View(productCategories);
            }
            // If in edit mode and productId is provided, get selected category ids
            else if (mode == "edit" && productId.HasValue)
            {
                var selectedCategoryIds = _context.Details
                    .Where(d => d.PId == productId.Value)
                    .Select(d => d.CId)
                    .ToList();
                ViewBag.SelectedCategories = selectedCategoryIds;
                return View(new List<Category>());
            }

            return View(new List<Category>());
        }
    }
}
```

Step 15: Components Folder Under the Shared Folder

CategoryManagement Folder under the Components Folder

Default.cshtml

```
@model IEnumerable<ProductMatarDetails_CRUD.Models.Category>
@{
    var index = ViewBag.Index;
    var mode = ViewBag.Mode;
```

```
    var categories = ViewBag.Categories as SelectList;
    var selectedCategories = ViewBag.SelectedCategories as List<int> ?? new
List<int>();
}
@if (mode == "display")
{
    <div class="categories-list">
        @if (Model.Any())
        {
            <table class="table text-center">
                <thead>
                    <tr>
                        <th>Category Id</th>
                        <th>Category Name</th>
                    </tr>
                </thead>
                <tbody>
                    @foreach (var category in Model)
                    {
                        <tr>
                            <td>@category.CId</td>
                            <td>@category.CName</td>
                        </tr>
                    }
                </tbody>
            </table>
        }
        else
        {
            <span>No Categories Available</span>
        }
    </div>
}
else
{
    <div class="rc my-2 d-flex">
        @if (mode == "edit" && index < selectedCategories.Count)
        {
            <select name="CId" class="form-control" asp-items="categories">
                <option value="">Select Category</option>
            </select>
        }
        else
        {
            <select name="CId" class="form-control" asp-items="categories">
                <option value="">Select Category</option>
            </select>
        }
        <button type="button" class="btn btn-danger remove ms-2">Remove</button>
    </div>

    @if (mode == "edit" && index < selectedCategories.Count)
    {
        <script>
            $(document).ready(function() {
                // Use jQuery to set the selected value for this specific dropdown
                $(".categories-container
.rc").eq(@index).find("select[name='CId']").val('@selectedCategories[index]');
```

```
            });
        </script>
    }
}
```

Step 16: `_Layout.cshtml`

In the Head Section:

```
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/jquery/dist/jquery.js"></script>


    <style>
        .mcon {
            width: 100vw;
            height: 100vh;
            background-color: rgba(0,0,0,.2);
            position: fixed;
            top: 0;
            left: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            z-index: 1000;
        }

        .mbod {
            width: 50%;
            max-height: 80vh;
            background-color: #fff;
            overflow: scroll;
            scrollbar-width: none;
        }
    </style>
```

Header Change to Product

End of the Body Section (After Footer)

```
    <script>
        $(document).ready(() => {
            // Create Button Click
            $(".crBtn").click((e) => {
                e.preventDefault();
                $.ajax({
                    url: "/Product/Create",
                    type: "get",
                    success: (d) => {
                        $(".macon").html(d);
```

```
                $(".mcon").show();
            }
        });
    });

    // Edit Button Click (use class selector)
    $(document).on("click", ".editBtn", (e) => {
        e.preventDefault();
        var id = $(e.currentTarget).data("id");
        console.log(id);
        $.ajax({
            url: "/Product/Edit/" + id,
            type: "get",
            success: (d) => {
                $(".macon").html(d);
                $(".mcon").show();
            }
        });
    });

    // Remove Button Click (Dynamic)
    $(document).on("click", ".remove", (e) => {
        e.preventDefault();
        $(e.currentTarget).closest(".rc").remove();
    });
    // Edit Button Click (use class selector)
    $(document).on("click", ".detailsBtn", (e) => {
        e.preventDefault();
        var id = $(e.currentTarget).data("id");
        console.log(id);
        $.ajax({
            url: "Product/Details/" + id,
            type: "get",
            success: (d) => {
                $(".macon").html(d);
                $(".mcon").show();
            }
        });
    });
    // Close Button Click
    $(".clbtn").click(() => {
        $(".mcon").hide();
    });
});
</script>
```