PART -1

Setup Asp.net core MVC

1. **Create a new project:**
    - Select ASP.NET Core Web App (Model-View-Controller).
    - Choose .NET 10 (ASP.NET Core 10) as the framework.

2. Configure the Model

- Create a folder called Models.

Product.cs:

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

3. Setup the Database Context

Install Entity Framework Core packages:

1. Microsoft.EntityFrameworkCore.SqlServer
2. Microsoft.EntityFrameworkCore.Tools

Create a class ApplicationDbContext.cs **in the Data folder:**

```
using Microsoft.EntityFrameworkCore;
using MvcCrudDemo.Models;
public class ApplicationDbContext : DbContext
{
   public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
      : base(options) { }
   public DbSet<Product> Products { get; set; }
}
```

IN: Program.cs:

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

Add a connection string in **appsettings.json:**

"ConnectionStrings": {

"DefaultConnection": "Server=YOUR_SERVER;Database=MvcCrudDemo;Trusted_Connection=True;"

}

Apply Migrations

Code

Add-Migration InitialCreate

Update-Database

PART -2:

ProductsController (Scaffolded)

```
public class ProductsController : Controller
{
  private readonly ApplicationDbContext _context;

  public ProductsController(ApplicationDbContext context)
  {
    _context = context;
  }
```

```
// GET: Products
public async Task<IActionResult> Index()
{
    return View(await _context.Products.ToListAsync());
}

// GET: Products/Details/5
public async Task<IActionResult> Details(int? id)
{
    if (id == null) return NotFound();

    var product = await _context.Products.FirstOrDefaultAsync(m => m.Id == id);
    if (product == null) return NotFound();

    return View(product);
}

// GET: Products/Create
public IActionResult Create()
{
    return View();
}

// POST: Products/Create
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create([Bind("Id,Name,Price")] Product product)
{
    if (ModelState.IsValid)
    {
        _context.Add(product);
```

```
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        return View(product);
    }

    // GET: Products/Edit/5
    public async Task<IActionResult> Edit(int? id)
    {
        if (id == null) return NotFound();

        var product = await _context.Products.FindAsync(id);
        if (product == null) return NotFound();

        return View(product);
    }

    // POST: Products/Edit/5
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Edit(int id, [Bind("Id,Name,Price")] Product product)
    {
        if (id != product.Id) return NotFound();

        if (ModelState.IsValid)
        {
            try
            {
                _context.Update(product);
                await _context.SaveChangesAsync();
            }
```

```
        catch (DbUpdateConcurrencyException)
        {
            if (!_context.Products.Any(e => e.Id == product.Id))
                return NotFound();
            else
                throw;
        }
        return RedirectToAction(nameof(Index));
    }
    return View(product);
}

// GET: Products/Delete/5
public async Task<IActionResult> Delete(int? id)
{
    if (id == null) return NotFound();

    var product = await _context.Products.FirstOrDefaultAsync(m => m.Id == id);
    if (product == null) return NotFound();

    return View(product);
}

// POST: Products/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var product = await _context.Products.FindAsync(id);
    _context.Products.Remove(product);
    await _context.SaveChangesAsync();
```

```
    return RedirectToAction(nameof(Index));
  }
}
```

Index.cshtml

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


<h2>Products</h2>
<p>
  <a asp-action="Create">Create New</a>
</p>
<table class="table">
  <thead>
    <tr>
      <th>@Html.DisplayNameFor(model => model.Name)</th>
      <th>@Html.DisplayNameFor(model => model.Price)</th>
      <th></th>
    </tr>
  </thead>
  <tbody>
@foreach (var item in Model) {
    <tr>
      <td>@item.Name</td>
      <td>@item.Price</td>
      <td>
        <a asp-action="Edit" asp-route-id="@item.Id">Edit</a> |
        <a asp-action="Details" asp-route-id="@item.Id">Details</a> |
        <a asp-action="Delete" asp-route-id="@item.Id">Delete</a>
```

```
        </td>
      </tr>
}
    </tbody>
</table>
```

Create.cshtml

```
@model MvcCrudDemo.Models.Product


<h2>Create Product</h2>
<form asp-action="Create">
    <div class="form-group">
        <label asp-for="Name"></label>
        <input asp-for="Name" class="form-control" />
    </div>
    <div class="form-group">
        <label asp-for="Price"></label>
        <input asp-for="Price" class="form-control" />
    </div>
    <button type="submit" class="btn btn-primary">Save</button>
</form>
```