**Step 1: Create the ASP.NET Core Web API Project**

dotnet new webapi -n ProductManagementAPI

cd ProductManagementAPI

dotnet restore

**Step 2: Create the Product Model**

Create a new file Models/Product.cs:

namespace ProductManagementAPI.Models

{

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

public string Description { get; set; }

}

}

**Step 3: Create the ProductService**

Create Services/ProductService.cs to handle CRUD operations using an in-memory list:

using ProductManagementAPI.Models;

using System.Collections.Generic;

using System.Linq;

namespace ProductManagementAPI.Services

{

public class ProductService

{

private readonly List<Product> \_products;

public ProductService()

{

\_products = new List<Product>

{

new Product { Id = 1, Name = "Product 1", Price = 10.99m, Description = "Description 1" },

new Product { Id = 2, Name = "Product 2", Price = 20.49m, Description = "Description 2" },

new Product { Id = 3, Name = "Product 3", Price = 15.99m, Description = "Description 3" }

};

}

public List<Product> GetAllProducts()

{

return \_products;

}

public Product GetProductById(int id)

{

return \_products.FirstOrDefault(p => p.Id == id);

}

public void AddProduct(Product newProduct)

{

\_products.Add(newProduct);

}

public void UpdateProduct(int id, Product updatedProduct)

{

var product = \_products.FirstOrDefault(p => p.Id == id);

if (product != null)

{

product.Name = updatedProduct.Name;

product.Price = updatedProduct.Price;

product.Description = updatedProduct.Description;

}

}

public void DeleteProduct(int id)

{

var product = \_products.FirstOrDefault(p => p.Id == id);

if (product != null)

{

\_products.Remove(product);

}

}

}

}

**Step 4: Create the ProductController**

Create Controllers/ProductController.cs:

using Microsoft.AspNetCore.Mvc;

using ProductManagementAPI.Models;

using ProductManagementAPI.Services;

using System.Collections.Generic;

namespace ProductManagementAPI.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class ProductController : ControllerBase

{

private readonly ProductService \_productService;

public ProductController()

{

\_productService = new ProductService();

}

// GET: /api/product

[HttpGet]

public IActionResult GetAllProducts()

{

var products = \_productService.GetAllProducts();

if (products.Count == 0)

return NoContent(); // 204 No Content

return Ok(products); // 200 OK

}

// GET: /api/product/{id}

[HttpGet("{id}")]

public IActionResult GetProductById(int id)

{

var product = \_productService.GetProductById(id);

if (product == null)

return NotFound(); // 404 Not Found

return Ok(product); // 200 OK

}

// POST: /api/product

[HttpPost]

public IActionResult CreateProduct([FromBody] Product newProduct)

{

if (newProduct == null || string.IsNullOrEmpty(newProduct.Name))

return BadRequest("Invalid product data"); // 400 Bad Request

\_productService.AddProduct(newProduct);

return CreatedAtAction(nameof(GetProductById), new { id = newProduct.Id }, newProduct); // 201 Created

}

// PUT: /api/product/{id}

[HttpPut("{id}")]

public IActionResult UpdateProduct(int id, [FromBody] Product updatedProduct)

{

var product = \_productService.GetProductById(id);

if (product == null)

return NotFound(); // 404 Not Found

\_productService.UpdateProduct(id, updatedProduct);

return Ok(updatedProduct); // 200 OK

}

// DELETE: /api/product/{id}

[HttpDelete("{id}")]

public IActionResult DeleteProduct(int id)

{

var product = \_productService.GetProductById(id);

if (product == null)

return NotFound(); // 404 Not Found

\_productService.DeleteProduct(id);

return NoContent(); // 204 No Content

}

}

}

**Step 5: Configure Swagger**

In Program.cs:

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "Product Management API v1");

});

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**Step 6: Run the API**

dotnet build

dotnet run

Swagger will be available at:

https://localhost:8080/swagger/index.html

You can test all endpoints there:

* **GET /api/product**
* **GET /api/product/{id}**
* **POST /api/product**
* **PUT /api/product/{id}**
* **DELETE /api/product/{id}**