**1. Create a new ASP.NET Core Web API Project**

dotnet new webapi -n dotnetapp

cd dotnetapp

dotnet restore

**2. Create the Order Model**

**File:** Models/Order.cs

namespace dotnetapp.Models

{

public class Order

{

public int OrderId { get; set; }

public string CustomerName { get; set; }

public DateTime OrderDate { get; set; }

public decimal TotalAmount { get; set; }

public string Status { get; set; }

}

}

**3. Create the OrderService**

**File:** Services/OrderService.cs

using dotnetapp.Models;

using System.Collections.Generic;

using System.Linq;

namespace dotnetapp.Services

{

public class OrderService

{

private readonly List<Order> \_orders;

public OrderService()

{

\_orders = new List<Order>

{

new Order { OrderId = 1, CustomerName = "John Doe", OrderDate = new DateTime(2023, 1, 1), TotalAmount = 100.50m, Status = "Shipped" },

new Order { OrderId = 2, CustomerName = "Jane Smith", OrderDate = new DateTime(2023, 2, 15), TotalAmount = 250.75m, Status = "Processing" },

new Order { OrderId = 3, CustomerName = "Alice Johnson", OrderDate = new DateTime(2023, 3, 20), TotalAmount = 150.00m, Status = "Delivered" }

};

}

public List<Order> GetAllOrders() => \_orders;

public Order GetOrderById(int orderId) => \_orders.FirstOrDefault(o => o.OrderId == orderId);

public Order AddOrder(Order newOrder)

{

newOrder.OrderId = \_orders.Max(o => o.OrderId) + 1;

\_orders.Add(newOrder);

return newOrder;

}

}

}

**4. Register OrderService in Dependency Injection**

**File:** Program.cs

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddSingleton<dotnetapp.Services.OrderService>(); // Register OrderService

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**5. Create OrderController**

**File:** Controllers/OrderController.cs

using dotnetapp.Models;

using dotnetapp.Services;

using Microsoft.AspNetCore.Mvc;

namespace dotnetapp.Controllers

{

[ApiController]

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

public class OrderController : ControllerBase

{

private readonly OrderService \_orderService;

public OrderController(OrderService orderService)

{

\_orderService = orderService;

}

// GET: api/Order

[HttpGet]

public IActionResult GetAllOrders()

{

var orders = \_orderService.GetAllOrders();

return Ok(orders); // 200 OK

}

// GET: api/Order/{id}

[HttpGet("{id}")]

public IActionResult GetOrderById(int id)

{

var order = \_orderService.GetOrderById(id);

if (order == null)

return NotFound($"Order with ID {id} not found."); // 404 NotFound

return Ok(order); // 200 OK

}

// POST: api/Order

[HttpPost]

public IActionResult CreateOrder([FromBody] Order newOrder)

{

if (newOrder == null || string.IsNullOrEmpty(newOrder.CustomerName) || newOrder.TotalAmount <= 0)

return BadRequest("Invalid order data."); // 400 BadRequest

var createdOrder = \_orderService.AddOrder(newOrder);

return CreatedAtAction(nameof(GetOrderById), new { id = createdOrder.OrderId }, createdOrder); // 201 Created

}

}

}

**6. Test API with Swagger**

After running the application:

dotnet run

* Open browser: https://localhost:8080/swagger/index.html
* Test the endpoints:
  + GET /api/Order → Returns all orders (200 OK)
  + GET /api/Order/{id} → Returns specific order or 404 NotFound
  + POST /api/Order → Create new order (201 Created or 400 BadRequest)

**7. Optional Enhancements**

* Implement **Update (PUT)** and **Delete (DELETE)** methods.
* Add **validation attributes** for model properties.
* Add **logging** for each request.