**1. Create the Project**

dotnet new webapi -n StoreManagementAPI

cd StoreManagementAPI

dotnet restore

**2. Install EF Core & Tools**

dotnet add package Microsoft.EntityFrameworkCore.SqlServer --version 6.0.0

dotnet add package Microsoft.EntityFrameworkCore.Design --version 6.0.0

# Setup EF tool locally

dotnet new tool-manifest

dotnet tool install --local dotnet-ef --version 6.0.6

dotnet dotnet-ef

**3. Define Models**

**InventoryItem.cs**:

namespace StoreManagementAPI.Models

{

public class InventoryItem

{

public int ItemId { get; set; }

public string ItemName { get; set; }

public int Quantity { get; set; }

public decimal Price { get; set; }

public string Category { get; set; }

}

}

**4. Define DbContext**

**ApplicationDbContext.cs**:

using Microsoft.EntityFrameworkCore;

using StoreManagementAPI.Models;

namespace StoreManagementAPI.Data

{

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{ }

public DbSet<InventoryItem> InventoryItems { get; set; }

}

}

**Add Connection String in appsettings.json**:

"ConnectionStrings": {

"DefaultConnection": "User ID=sa;password=examlyMssql@123;server=localhost;Database=StoreDB;trusted\_connection=false;Persist Security Info=False;Encrypt=False"

}

**Configure DbContext in Program.cs**:

using StoreManagementAPI.Data;

using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

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

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

builder.Services.AddControllers();

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. Implement Inventory Service**

**InventoryService.cs**:

using StoreManagementAPI.Data;

using StoreManagementAPI.Models;

using Microsoft.EntityFrameworkCore;

namespace StoreManagementAPI.Services

{

public class InventoryService

{

private readonly ApplicationDbContext \_context;

public InventoryService(ApplicationDbContext context)

{

\_context = context;

}

public async Task<List<InventoryItem>> GetAllItemsAsync()

{

return await \_context.InventoryItems.ToListAsync();

}

public async Task<InventoryItem?> GetItemByIdAsync(int itemId)

{

return await \_context.InventoryItems.FindAsync(itemId);

}

public async Task<InventoryItem> AddItemAsync(InventoryItem newItem)

{

\_context.InventoryItems.Add(newItem);

await \_context.SaveChangesAsync();

return newItem;

}

}

}

**6. Implement Inventory Controller**

**InventoryController.cs**:

using Microsoft.AspNetCore.Mvc;

using StoreManagementAPI.Models;

using StoreManagementAPI.Services;

namespace StoreManagementAPI.Controllers

{

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

[ApiController]

public class InventoryController : ControllerBase

{

private readonly InventoryService \_service;

public InventoryController(InventoryService service)

{

\_service = service;

}

// GET: api/Inventory

[HttpGet]

public async Task<IActionResult> GetAllInventories()

{

var items = await \_service.GetAllItemsAsync();

if (items == null || items.Count == 0)

return NoContent(); // 204 No Content

return Ok(items); // 200 OK

}

// GET: api/Inventory/{id}

[HttpGet("{id}")]

public async Task<IActionResult> GetInventoryById(int id)

{

var item = await \_service.GetItemByIdAsync(id);

if (item == null)

return NotFound(); // 404 Not Found

return Ok(item); // 200 OK

}

// POST: api/Inventory

[HttpPost]

public async Task<IActionResult> CreateInventoryItem([FromBody] InventoryItem newItem)

{

if (newItem == null)

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

var createdItem = await \_service.AddItemAsync(newItem);

return CreatedAtAction(nameof(GetInventoryById), new { id = createdItem.ItemId }, createdItem); // 201 Created

}

}

}

**7. Apply EF Core Migrations**

dotnet dotnet-ef migrations add InitialSetup

dotnet dotnet-ef database update

**8. Test with Swagger**

* Run the project:

dotnet run

* Open Swagger: https://localhost:8080/swagger/index.html
* Test all endpoints (GET, GET/{id}, POST) with different scenarios to check status codes: 200, 201, 204, 400, 404.