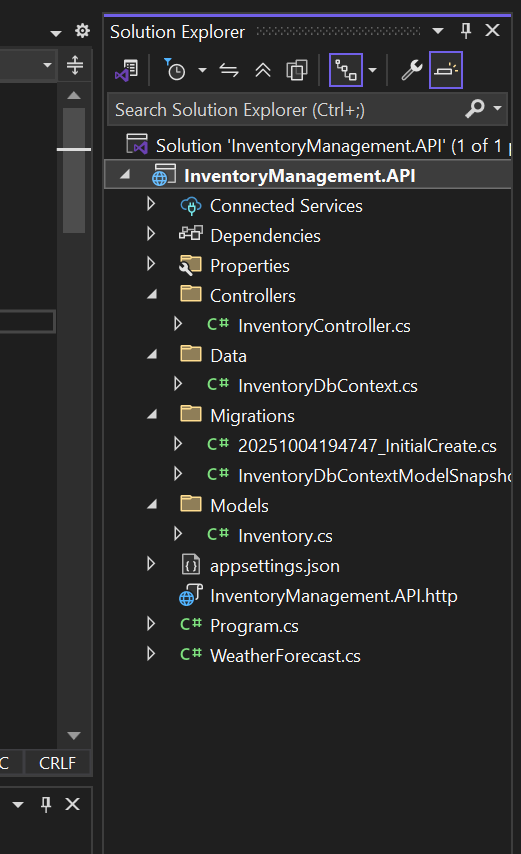
**Backend code for inventory management front end using Angular:**

Folder structure:



**Inventory.cs**

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace InventoryManagement.API.Models

{

public class Inventory

{

[Key]

public int InventoryID { get; set; }

[Required]

[StringLength(100)]

public string ItemName { get; set; }

[Required]

public int StockQty { get; set; }

[Required]

public int ReorderQty { get; set; }

[Required]

public int PriorityStatus { get; set; } // 0 = Low, 1 = High

public DateTime CreatedDate { get; set; } = DateTime.Now;

public DateTime? LastUpdated { get; set; }

// Navigation property

public virtual ICollection<Stock> Stocks { get; set; } = new List<Stock>();

}

public class User

{

[Key]

public int UserID { get; set; }

[Required]

[StringLength(100)]

public string UserName { get; set; }

[Required]

[EmailAddress]

[StringLength(150)]

public string Email { get; set; }

[Required]

public string UserType { get; set; } // Admin, User, Supplier

public DateTime CreatedDate { get; set; } = DateTime.Now;

public bool IsActive { get; set; } = true;

}

public class Stock

{

[Key]

public int StockID { get; set; }

[Required]

public int InventoryID { get; set; }

[ForeignKey("InventoryID")]

public virtual Inventory Inventory { get; set; }

[Required]

public int Quantity { get; set; }

[Required]

public string TransactionType { get; set; } // IN, OUT, ADJUSTMENT

public string Remarks { get; set; }

public int UserID { get; set; } // Who performed the transaction

public DateTime TransactionDate { get; set; } = DateTime.Now;

}

}

**InventoryDbContext.cs**

using Microsoft.EntityFrameworkCore;

using InventoryManagement.API.Models;

namespace InventoryManagement.API.Data

{

public class InventoryDbContext : DbContext

{

public InventoryDbContext(DbContextOptions<InventoryDbContext> options)

: base(options) { }

public DbSet<Inventory> Inventories { get; set; }

public DbSet<User> Users { get; set; }

public DbSet<Stock> Stocks { get; set; }

}

}

**InventoryController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using InventoryManagement.API.Data;

using InventoryManagement.API.Models;

namespace InventoryManagement.API.Controllers

{

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

[ApiController]

public class InventoryController : ControllerBase

{

private readonly InventoryDbContext \_context;

public InventoryController(InventoryDbContext context)

{

\_context = context;

}

// GET: api/Inventory

[HttpGet]

public async Task<ActionResult<IEnumerable<Inventory>>> GetInventories()

{

return await \_context.Inventories.ToListAsync();

}

// GET: api/Inventory/5

[HttpGet("{id}")]

public async Task<ActionResult<Inventory>> GetInventory(int id)

{

var inventory = await \_context.Inventories.FindAsync(id);

if (inventory == null) return NotFound();

return inventory;

}

// POST: api/Inventory

[HttpPost]

public async Task<ActionResult<Inventory>> PostInventory(Inventory inventory)

{

\_context.Inventories.Add(inventory);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetInventory", new { id = inventory.InventoryID }, inventory);

}

// PUT: api/Inventory/5

[HttpPut("{id}")]

public async Task<IActionResult> PutInventory(int id, Inventory inventory)

{

if (id != inventory.InventoryID) return BadRequest();

inventory.LastUpdated = DateTime.Now;

\_context.Entry(inventory).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!InventoryExists(id)) return NotFound();

throw;

}

return NoContent();

}

// DELETE: api/Inventory/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteInventory(int id)

{

var inventory = await \_context.Inventories.FindAsync(id);

if (inventory == null) return NotFound();

\_context.Inventories.Remove(inventory);

await \_context.SaveChangesAsync();

return NoContent();

}

// POST: api/Inventory/UpdateStock

[HttpPost("UpdateStock")]

public async Task<ActionResult> UpdateStock([FromBody] StockUpdateRequest request)

{

var inventory = await \_context.Inventories.FindAsync(request.InventoryID);

if (inventory == null) return NotFound();

// Update inventory quantity

if (request.TransactionType == "IN")

inventory.StockQty += request.Quantity;

else if (request.TransactionType == "OUT")

inventory.StockQty -= request.Quantity;

else

inventory.StockQty = request.Quantity; // ADJUSTMENT

inventory.LastUpdated = DateTime.Now;

// Create stock transaction record

var stockTransaction = new Stock

{

InventoryID = request.InventoryID,

Quantity = request.Quantity,

TransactionType = request.TransactionType,

Remarks = request.Remarks,

UserID = request.UserID,

TransactionDate = DateTime.Now

};

\_context.Stocks.Add(stockTransaction);

await \_context.SaveChangesAsync();

return Ok(new { message = "Stock updated successfully", newQuantity = inventory.StockQty });

}

private bool InventoryExists(int id)

{

return \_context.Inventories.Any(e => e.InventoryID == id);

}

}

public class StockUpdateRequest

{

public int InventoryID { get; set; }

public int Quantity { get; set; }

public string TransactionType { get; set; } // IN, OUT, ADJUSTMENT

public string Remarks { get; set; }

public int UserID { get; set; }

}

}

**Program.cs**

using Microsoft.EntityFrameworkCore;

using InventoryManagement.API.Data;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

// EF Core

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

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

// CORS

builder.Services.AddCors(options =>

{

options.AddPolicy("AllowAngularApp", policy =>

{

policy.WithOrigins("http://localhost:4200")

.AllowAnyHeader()

.AllowAnyMethod()

.AllowCredentials();

});

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseCors("AllowAngularApp");

app.UseAuthorization();

app.MapControllers();

app.Run();