Lab 3: Using EF Core CLI to Create and Apply Migrations

a)SuperStoreContext.cs

using Microsoft.EntityFrameworkCore;

using SuperStoreApp.Entities;

namespace SuperStoreApp.DataLayer

{

    public class SuperStoreContext : DbContext

    {

        public SuperStoreContext(DbContextOptions<SuperStoreContext> options)

            : base(options)

        {

        }

        public DbSet<Item> Items { get; set; }

        public DbSet<Client> Clients { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)

        {

            base.OnModelCreating(modelBuilder);

            // Additional model config here if needed

        }

    }

}

b)Client.cs

namespace SuperStoreApp.Entities

{

    public class Client

    {

        public int Id { get; set; }

        public string FullName { get; set; }

    }

}

c)Item.cs

namespace SuperStoreApp.Entities

{

    public class Item

    {

        public int Id { get; set; }

        public string Name { get; set; }

        public decimal Price { get; set; }

    }

}

d)appsettings.json

{

  "ConnectionStrings": {

    "SuperStoreDb": "Data Source=superstore.db"

  }

}

e)program.cs

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Configuration;

using Microsoft.EntityFrameworkCore;

using SuperStoreApp.DataLayer;

using SuperStoreApp.Entities;

using System;

using System.Threading.Tasks;

await RunAsync();

async Task RunAsync()

{

    var host = Host.CreateDefaultBuilder(args)

        .ConfigureAppConfiguration((context, builder) =>

        {

            builder.AddJsonFile("appsettings.json", optional: false, reloadOnChange: true);

        })

        .ConfigureServices((context, services) =>

        {

            var connString = context.Configuration.GetConnectionString("SuperStoreDb");

            services.AddDbContext<SuperStoreContext>(options =>

                options.UseSqlite(connString));

        })

        .Build();

    using var scope = host.Services.CreateScope();

    var dbContext = scope.ServiceProvider.GetRequiredService<SuperStoreContext>();

    try

    {

        // Apply any pending migrations and create database if needed

        Console.WriteLine("Applying migrations (if any)...");

        await dbContext.Database.MigrateAsync();

        // Output some info

        var itemCount = await dbContext.Items.CountAsync();

        var clientCount = await dbContext.Clients.CountAsync();

        Console.WriteLine("✅ Database is ready!");

        Console.WriteLine($"Items count: {itemCount}");

        Console.WriteLine($"Clients count: {clientCount}");

    }

    catch (Exception ex)

    {

        Console.WriteLine("❌ Failed to initialize database.");

        Console.WriteLine($"Error: {ex.Message}");

    }

    Console.WriteLine("Press any key to exit...");

    Console.ReadKey();

}

Output:

