Lab 2: Setting Up the Database Context for a Retail Store

a)ProductModel.cs:

namespace RetailStoreApp.Models;

public class ProductModel

{

    public int ProductModelId { get; set; }

    public string Name { get; set; } = string.Empty;

    public decimal Price { get; set; }

    public int Stock { get; set; }

}

b)CustomerModel.cs

namespace RetailStoreApp.Models;

public class CustomerModel

{

    public int CustomerModelId { get; set; }

    public string FullName { get; set; } = string.Empty;

    public string Email { get; set; } = string.Empty;

}

c)RetailDBContext.cs

using Microsoft.EntityFrameworkCore;

using RetailStoreApp.Models;

namespace RetailStoreApp.Data

{

    public class RetailDbContext : DbContext

    {

        public RetailDbContext(DbContextOptions<RetailDbContext> options)

            : base(options)

        {

        }

        public DbSet<ProductModel> Products => Set<ProductModel>();

        public DbSet<CustomerModel> Customers => Set<CustomerModel>();

        protected override void OnModelCreating(ModelBuilder modelBuilder)

        {

            modelBuilder.Entity<ProductModel>()

                .Property(p => p.Price)

                .HasPrecision(18, 2);

        }

    }

}

d)Program.cs

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.EntityFrameworkCore;

using RetailStoreApp.Data;

using RetailStoreApp.Models;

var host = CreateHostBuilder(args).Build();

// Create scope to resolve services

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

var services = scope.ServiceProvider;

try

{

    var dbContext = services.GetRequiredService<RetailDbContext>();

    // Ensure database is created

    dbContext.Database.EnsureCreated();

    Console.WriteLine("✅ Database has been created successfully.");

    // Seed data if the database is empty

    if (!dbContext.Products.Any())

    {

        dbContext.Products.Add(new ProductModel

        {

            Name = "Laptop",

            Price = 1200.00m,

            Stock = 10

        });

        dbContext.SaveChanges();

    }

    // Output all products

    var products = dbContext.Products.ToList();

    Console.WriteLine("📦 Products in database:");

    foreach (var product in products)

    {

        Console.WriteLine($"- {product.Name} (${product.Price}) - {product.Stock} in stock");

    }

}

catch (Exception ex)

{

    Console.WriteLine("❌ Failed to create or access the database:");

    Console.WriteLine(ex.Message);

}

static IHostBuilder CreateHostBuilder(string[] args) =>

    Host.CreateDefaultBuilder(args)

        .ConfigureAppConfiguration((context, config) =>

        {

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

        })

        .ConfigureServices((context, services) =>

        {

            var connectionString = context.Configuration.GetConnectionString("RetailDbConnection");

            // ✅ Using SQLite instead of SQL Server

            services.AddDbContext<RetailDbContext>(options =>

                options.UseSqlite(connectionString));

        });

e)appsettings.json

{

  "ConnectionStrings": {

    "RetailDbConnection": "Data Source=retailstore.db"

  }

}

Output:

