QUESTION 1

using System;

using System.Collections.Generic;

class Program

{

    static void Main(string[] args)

    {

        // Create a sample product

        Product product = new Product

        {

            Id = 1,

            Name = "Laptop",

            Price = 75000

        };

        Console.WriteLine($"Product: {product.Name}, Price: ₹{product.Price}");

    }

}

public class Product

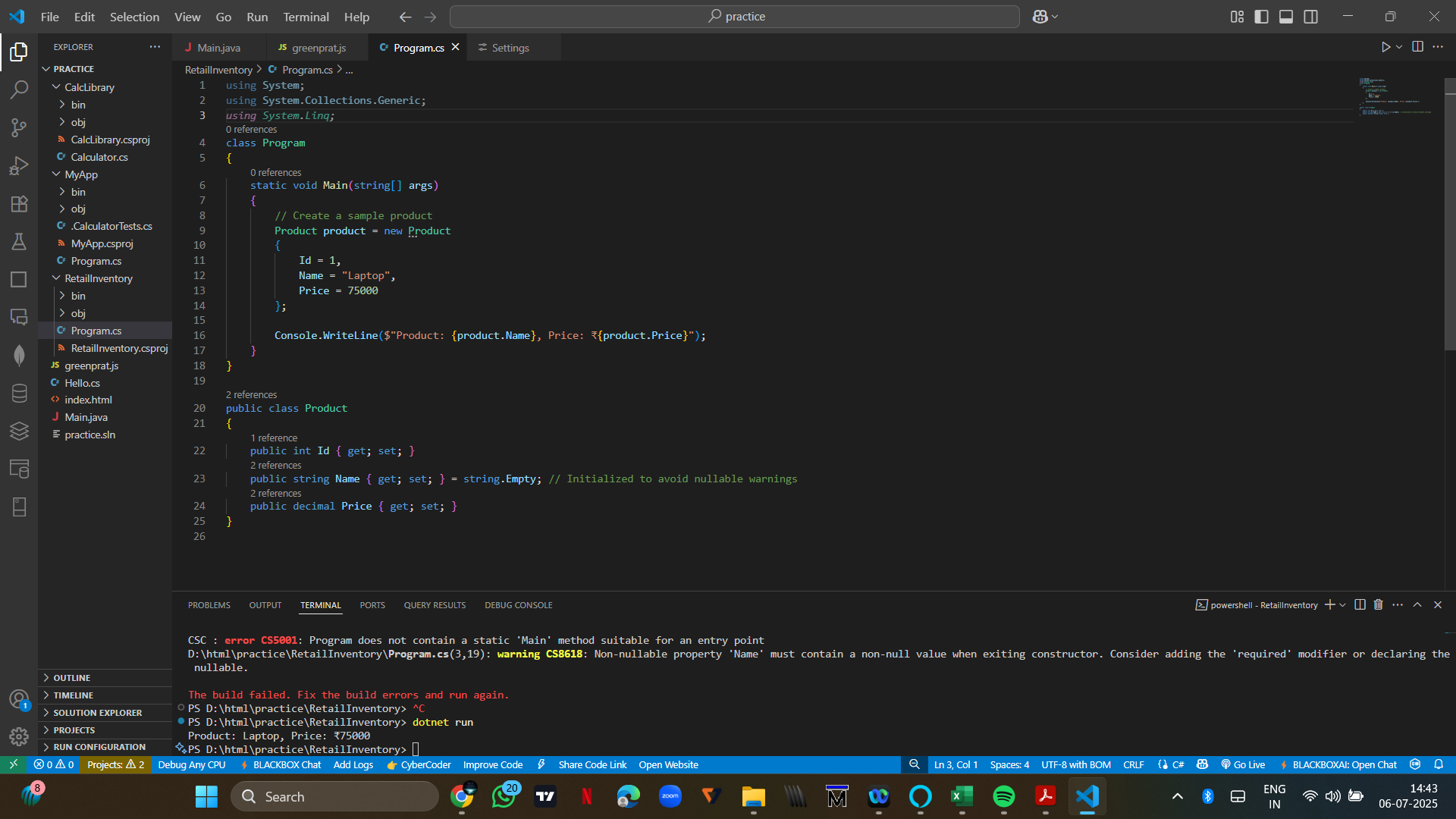
{

    public int Id { get; set; }

    public string Name { get; set; } = string.Empty; // Initialized to avoid nullable warnings

    public decimal Price { get; set; }

}



QUESTION 2

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

class Program

{

    static async Task Main(string[] args)

    {

        Console.WriteLine("EF Core vs EF Framework Demo");

        using var context = new AppDbContext();

        // Ensure database and table are created

        await context.Database.EnsureCreatedAsync();

        // Insert data only if database is empty

        if (!await context.Products.AnyAsync())

        {

            var electronics = new Category { Name = "Electronics" };

            var groceries = new Category { Name = "Groceries" };

            var laptop = new Product { Name = "Laptop", Price = 75000, Category = electronics };

            var rice = new Product { Name = "Rice Bag", Price = 1200, Category = groceries };

            await context.Categories.AddRangeAsync(electronics, groceries);

            await context.Products.AddRangeAsync(laptop, rice);

            await context.SaveChangesAsync();

            Console.WriteLine("Inserted sample categories and products.");

        }

        // Fetch and display products

        var products = await context.Products.Include(p => p.Category).ToListAsync();

        foreach (var p in products)

        {

            Console.WriteLine($"Product: {p.Name}, Price: ₹{p.Price}, Category: {p.Category?.Name}");

        }

    }

}

// Product Model

public class Product

{

    public int Id { get; set; }

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

    public decimal Price { get; set; }

    public int CategoryId { get; set; }

    public Category? Category { get; set; }

}

// Category Model

public class Category

{

    public int Id { get; set; }

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

    public List<Product> Products { get; set; } = new();

}

// DbContext

public class AppDbContext : DbContext

{

    public DbSet<Product> Products { get; set; }

    public DbSet<Category> Categories { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

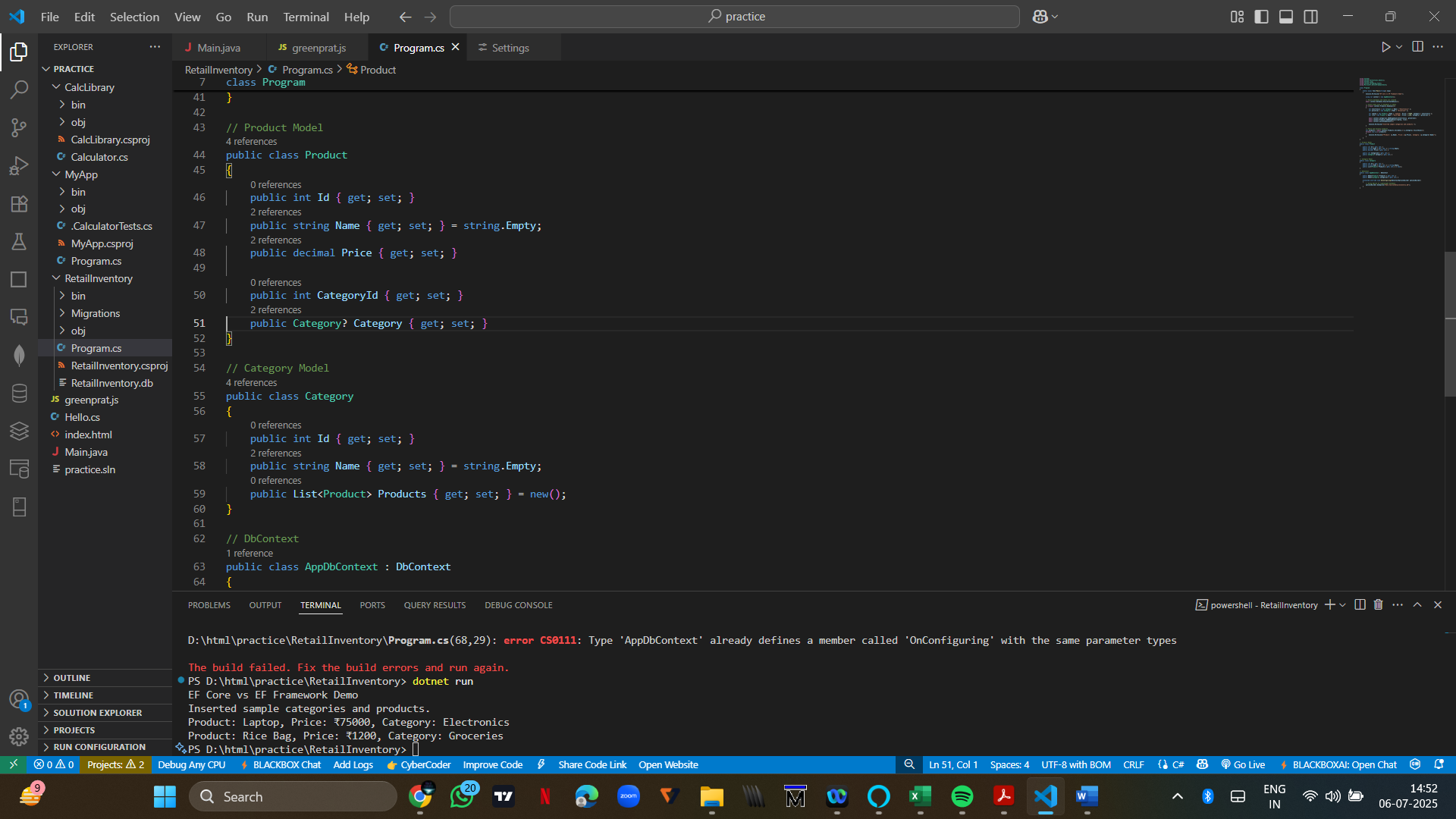
    {

        // Using SQLite as lightweight database

        optionsBuilder.UseSqlite("Data Source=RetailInventory.db");

    }

}



Question 3

dotnet tool install --global dotnet-ef

dotnet ef migrations add InitialCreate

dotnet ef database update

RetailInventory/

1. Migrations/

* 20250706123456\_InitialCreate.cs
* RetailInventoryModelSnapshot.cs

1. RetailInventory.db
2. Program.cs
3. RetailInventory.csproj

Question 4

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

class Program

{

static async Task Main(string[] args)

{

using var context = new AppDbContext();

// Ensure the database is created (SQLite)

await context.Database.EnsureCreatedAsync();

// ✅ Insert only if the database is empty

if (!await context.Products.AnyAsync())

{

var electronics = new Category { Name = "Electronics" };

var groceries = new Category { Name = "Groceries" };

await context.Categories.AddRangeAsync(electronics, groceries);

var product1 = new Product { Name = "Laptop", Price = 75000, Category = electronics };

var product2 = new Product { Name = "Rice Bag", Price = 1200, Category = groceries };

await context.Products.AddRangeAsync(product1, product2);

await context.SaveChangesAsync();

Console.WriteLine("✅ Initial data inserted.");

}

else

{

Console.WriteLine("ℹ️ Database already contains data.");

}

// Show inserted data

var products = await context.Products.Include(p => p.Category).ToListAsync();

foreach (var p in products)

{

Console.WriteLine($"📦 {p.Name} - ₹{p.Price} [{p.Category?.Name}]");

}

}

}

// Model: Product

public class Product

{

public int Id { get; set; }

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

public decimal Price { get; set; }

public int CategoryId { get; set; }

public Category? Category { get; set; }

}

// Model: Category

public class Category

{

public int Id { get; set; }

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

public List<Product> Products { get; set; } = new();

}

// DbContext

public class AppDbContext : DbContext

{

public DbSet<Product> Products { get; set; }

public DbSet<Category> Categories { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

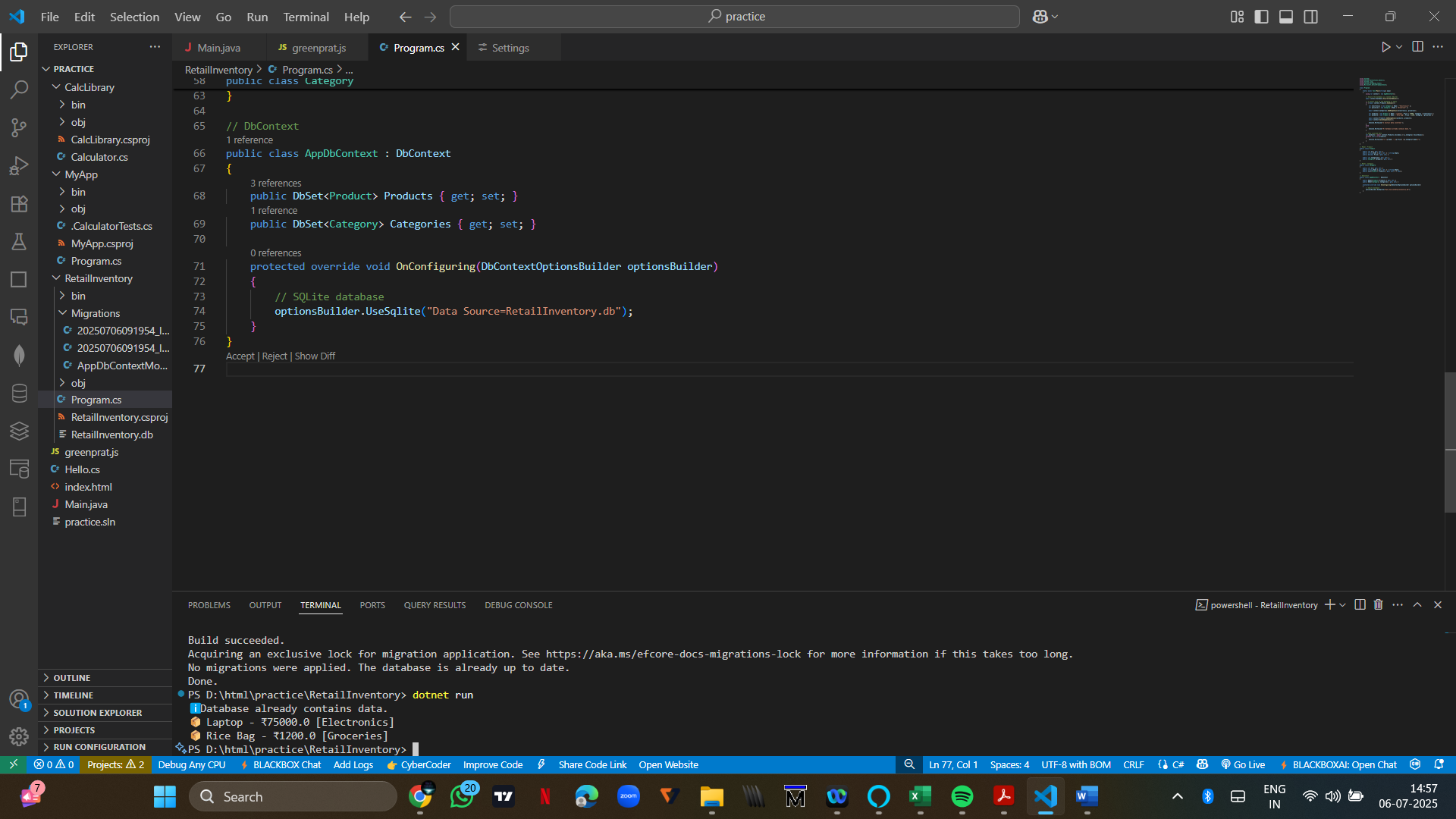
{

// SQLite database

optionsBuilder.UseSqlite("Data Source=RetailInventory.db");

}

}



Question 5

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

class Program

{

static async Task Main(string[] args)

{

using var context = new AppDbContext();

await context.Database.EnsureCreatedAsync();

Console.WriteLine("Retrieving product data from database...\n");

// 1. Retrieve all products

var allProducts = await context.Products.Include(p => p.Category).ToListAsync();

Console.WriteLine(" All Products:");

foreach (var p in allProducts)

{

Console.WriteLine($"- {p.Name} - ₹{p.Price} [{p.Category?.Name}]");

}

// 2. Find by ID

var productById = await context.Products.FindAsync(1);

Console.WriteLine($"\n Product with ID 1: {productById?.Name ?? "Not found"}");

// 3. First product with Price > 50,000

var expensiveProduct = await context.Products

.Include(p => p.Category)

.FirstOrDefaultAsync(p => p.Price > 50000);

Console.WriteLine($"\n Expensive Product (> ₹50,000): {expensiveProduct?.Name ?? "None found"}");

}

}

// Model: Product

public class Product

{

public int Id { get; set; }

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

public decimal Price { get; set; }

public int CategoryId { get; set; }

public Category? Category { get; set; }

}

// Model: Category

public class Category

{

public int Id { get; set; }

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

public List<Product> Products { get; set; } = new();

}

// DbContext

public class AppDbContext : DbContext

{

public DbSet<Product> Products { get; set; }

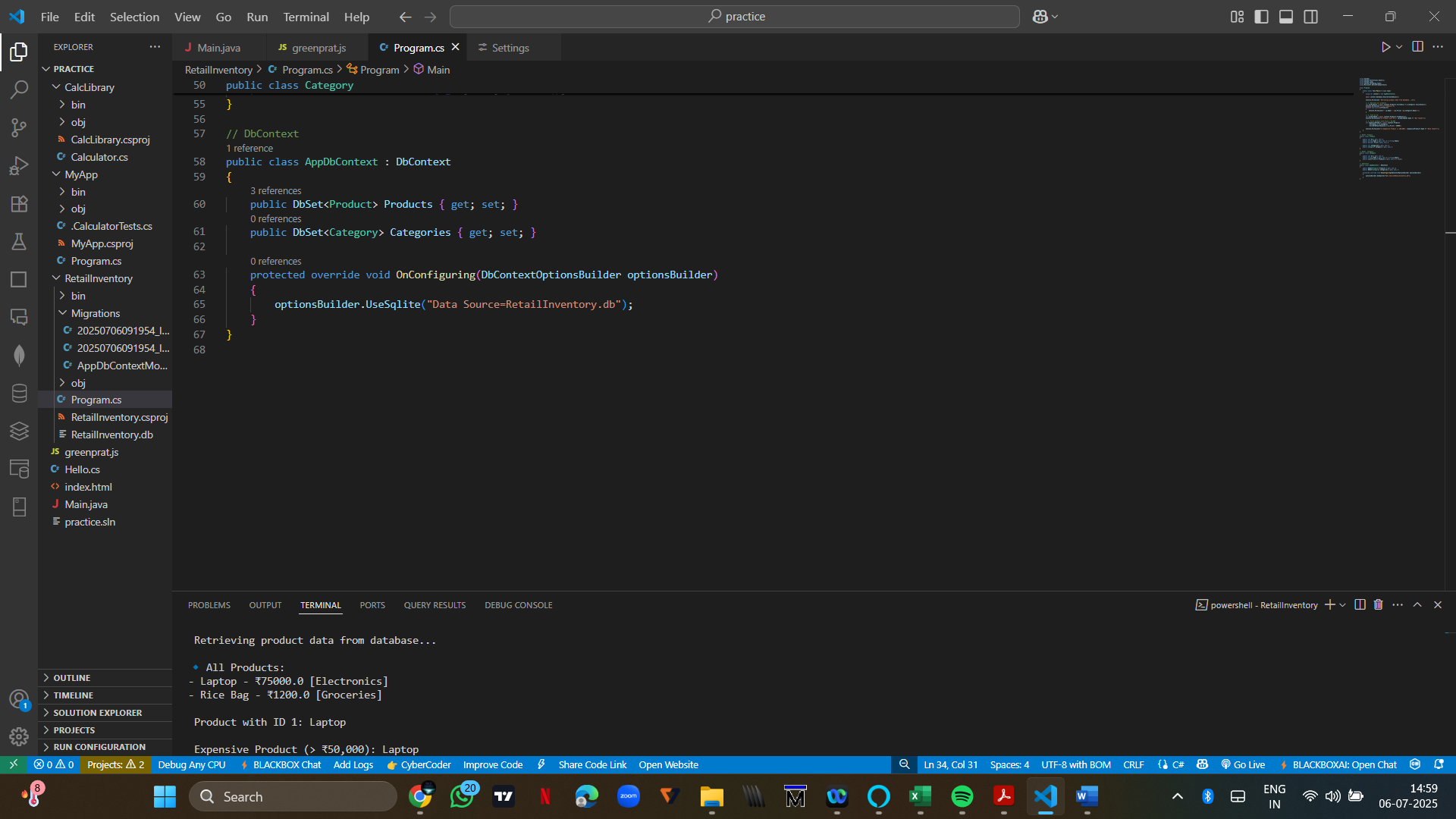
public DbSet<Category> Categories { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlite("Data Source=RetailInventory.db");

}

}  


Question 6

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

class Program

{

    static async Task Main(string[] args)

    {

        using var context = new AppDbContext();

        await context.Database.EnsureCreatedAsync();

        Console.WriteLine("🛠 Updating and Deleting Products...\n");

        // 1. UPDATE a product ("Laptop" to ₹70,000)

        var productToUpdate = await context.Products.FirstOrDefaultAsync(p => p.Name == "Laptop");

        if (productToUpdate != null)

        {

            productToUpdate.Price = 70000;

            await context.SaveChangesAsync();

            Console.WriteLine($" Updated price of {productToUpdate.Name} to ₹{productToUpdate.Price}");

        }

        else

        {

            Console.WriteLine(" Product 'Laptop' not found for update.");

        }

        // 2. DELETE a product ("Rice Bag")

        var productToDelete = await context.Products.FirstOrDefaultAsync(p => p.Name == "Rice Bag");

        if (productToDelete != null)

        {

            context.Products.Remove(productToDelete);

            await context.SaveChangesAsync();

            Console.WriteLine($"🗑 Deleted product: {productToDelete.Name}");

        }

        else

        {

            Console.WriteLine("Product 'Rice Bag' not found for deletion.");

        }

        // 3. Display remaining products

        var remaining = await context.Products.Include(p => p.Category).ToListAsync();

        Console.WriteLine("\n Remaining Products:");

        foreach (var p in remaining)

        {

            Console.WriteLine($"- {p.Name} - ₹{p.Price} [{p.Category?.Name}]");

        }

    }

}

// Model: Product

public class Product

{

    public int Id { get; set; }

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

    public decimal Price { get; set; }

    public int CategoryId { get; set; }

    public Category? Category { get; set; }

}

// Model: Category

public class Category

{

    public int Id { get; set; }

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

    public List<Product> Products { get; set; } = new();

}

// DbContext

public class AppDbContext : DbContext

{

    public DbSet<Product> Products { get; set; }

    public DbSet<Category> Categories { get; set; }

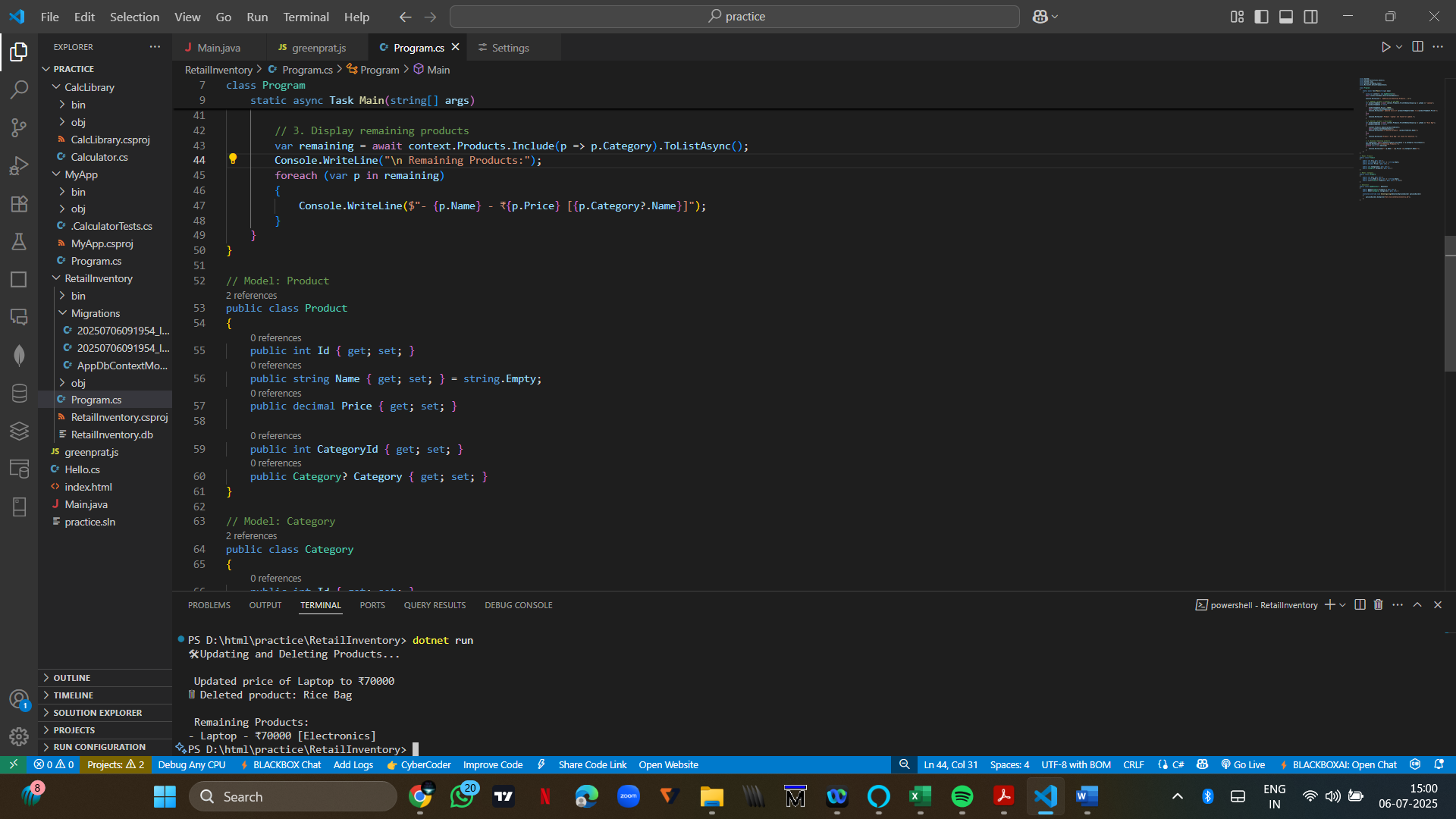
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

    {

        optionsBuilder.UseSqlite("Data Source=RetailInventory.db");

    }

}



Question 7

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

class Program

{

    static async Task Main(string[] args)

    {

        using var context = new AppDbContext();

        await context.Database.EnsureCreatedAsync();

        Console.WriteLine("LINQ Queries on Product Data\n");

        // 1. Filter products with Price > 1000 and sort by descending price

        var filtered = context.Products

        .Where(p => p.Price > 1000)

        .AsEnumerable()

        .OrderByDescending(p => p.Price)

        .ToList();

        Console.WriteLine("Filtered & Sorted Products (Price > 1000):");

        foreach (var p in filtered)

        {

            Console.WriteLine($"- {p.Name}: ₹{p.Price} [{p.Category?.Name}]");

        }

        // 2. Project products into DTOs (Name + Price only)

        var productDTOs = await context.Products

            .Select(p => new ProductDTO

            {

                Name = p.Name,

                Price = p.Price

            })

            .ToListAsync();

        Console.WriteLine("\n Product DTOs (Name & Price):");

        foreach (var dto in productDTOs)

        {

            Console.WriteLine($"- {dto.Name}: ₹{dto.Price}");

        }

    }

}

// DTO class to project query results

public class ProductDTO

{

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

    public decimal Price { get; set; }

}

// Product model

public class Product

{

    public int Id { get; set; }

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

    public decimal Price { get; set; }

    public int CategoryId { get; set; }

    public Category? Category { get; set; }

}

// Category model

public class Category

{

    public int Id { get; set; }

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

    public List<Product> Products { get; set; } = new();

}

// DbContext

public class AppDbContext : DbContext

{

    public DbSet<Product> Products { get; set; }

    public DbSet<Category> Categories { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

    {

        optionsBuilder.UseSqlite("Data Source=RetailInventory.db");

    }

}

