Module 8

What to learn

Async and await
Extension Method

Practice Exercise

Practice 1
Async and await
Practice 2
Do one example of async await
Practice 3
Do one example of Extention Method
Practice 4

Async and Await

Asynchronous File Download
Write a method that uses async and await to download a file from a given URL and save it locally. Ensure that the method can handle the download asynchronously and provide feedback on the download progress.

Asynchronous Database Query
Implement an async method to query customer details from a database. The method should simulate a database delay using Task.Delay and return the customer data asynchronously.

Parallel API Calls
Write an async method that makes parallel API calls to fetch product details from three different online stores. Use await for each request and ensure that all data is returned in a consolidated format once all calls are completed.

Asynchronous Order Processing
Simulate an e-commerce system where order processing involves multiple asynchronous operations, such as checking stock, calculating taxes, and applying discounts. Write an async method that processes the order and returns a confirmation asynchronously.

Async File Processing
Write an async method to process a large number of files asynchronously. The method should read the files, process their content, and save the results back to a

storage system. Ensure that the method completes each file's processing asynchronously.

Practice 5

Extension Method

Custom String Validation
Implement an extension method IsValidEmail() for the string type that validates if an email address is in a valid format (basic validation). Use this extension to check whether a list of email addresses is valid.

Collection Average Calculation
Create an extension method AveragePrice() for a List<Product> class, where each Product contains a Price property. Use this method to calculate the average price of all products in a list.

String Reversal
Implement an extension method ReverseString() for the string class that returns the reversed version of a given string. Use this method to reverse a list of strings and output the results.

Custom Sorting
Create an extension method SortByName() for a List<Employee> class that sorts employees by their Name property. Implement a small program that uses this extension method to sort and display employees.

Date Formatting
Implement an extension method ToFormattedDate() for DateTime that formats the date into a custom format (e.g., "yyyy-MM-dd"). Write a program to display a list of dates in this formatted version.

Assignment Exercise

Assignment 1
Objective:
This assignment combines the concepts of **Async and Await** and **Extension Methods** to create a real-world business application that processes orders, performs asynchronous operations, and uses extension methods to extend the functionality of existing types.

Scenario: E-Commerce Order Processing System

You are building an **E-Commerce System** that processes customer orders. The system needs to handle the following tasks:

Order Validation: Check if the product is in stock.

- **Tax Calculation**: Asynchronously calculate taxes based on the product price.
- **Discount Application**: Asynchronously apply discounts for premium customers.
- **Shipping**: Use async operations to simulate different shipping methods.
- **Order Summary**: Extend existing types to format and output order details.

You will implement the following features:

Requirements

Data Models

Create the following classes to represent the system:

Customer

- CustomerId (int)
- Name (string)
- IsPremium (bool)

Product

- ProductId (int)
- Name (string)
- Price (decimal)
- Stock (int)

Order

- OrderId (int)
- CustomerId (int)
- ProductId (int)
- Quantity (int)
- OrderDate (DateTime)

ShippingInfo

- ShippingId (int)
- ShippingMethod (string)
- ShippingCost (decimal)

Tasks

1. Asynchronous Order Processing

Write an async method ProcessOrderAsync(Order order) that:

- Validates if the product is in stock.
- Simulates tax calculation with a delay.
- Applies a discount for premium customers (if applicable).

Determines the shipping cost based on the shipping method chosen.

2. Extension Methods

Product Extension:
Create an extension method IsInStock(this Product product) for the Product class that checks if the quantity of a product is greater than zero.

Order Extension:
Create an extension method CalculateTotal(this Order order, Product product) that calculates the total price for the order (price * quantity) and applies any applicable discount for premium customers.

Shipping Extension:
Create an extension method ApplyShippingCost(this ShippingInfo shippingInfo) that calculates the shipping cost based on the shipping method.

Date Extension:
Create an extension method ToFormattedDate(this DateTime date) for the DateTime class to return a custom formatted date string (e.g., "yyyy-MM-dd").

3. Main Program Logic

Implement a main program that:

Creates a list of customers, products, and orders.
Calls the ProcessOrderAsync method for each order and outputs the order summary after processing.
Use extension methods for calculating totals, checking stock, and applying shipping costs.
Displays the order details with formatted dates.

Online Reference

No online Reference

.NET Core Web API

WEB API (old)

Authentication And Authorization (WEBAPI)(old)

FullStackDevelopment_WIth_Dotnet_AND_Angular