Lab Manual for Cloud Computing

Lab No. 8

Consuming Web Services Using ASP .Net

LAB 08: CONSUMING WEB SERVICES USING ASP .NET

1. INTRODUCTION:

In ASP.NET, consuming web services serves as a pivotal mechanism for seamlessly integrating diverse external functionalities into web applications. Developers can employ a variety of tools such as HttpClient and WebClient to effectively manage HTTP requests and responses, thereby ensuring a high degree of flexibility and control over communication with remote services.

Furthermore, ASP.NET simplifies the process by offering built-in support for generating proxies tailored to specific web services. These proxies abstract away the complexities of remote communication, enabling developers to interact with external services as if they were local components within their application.

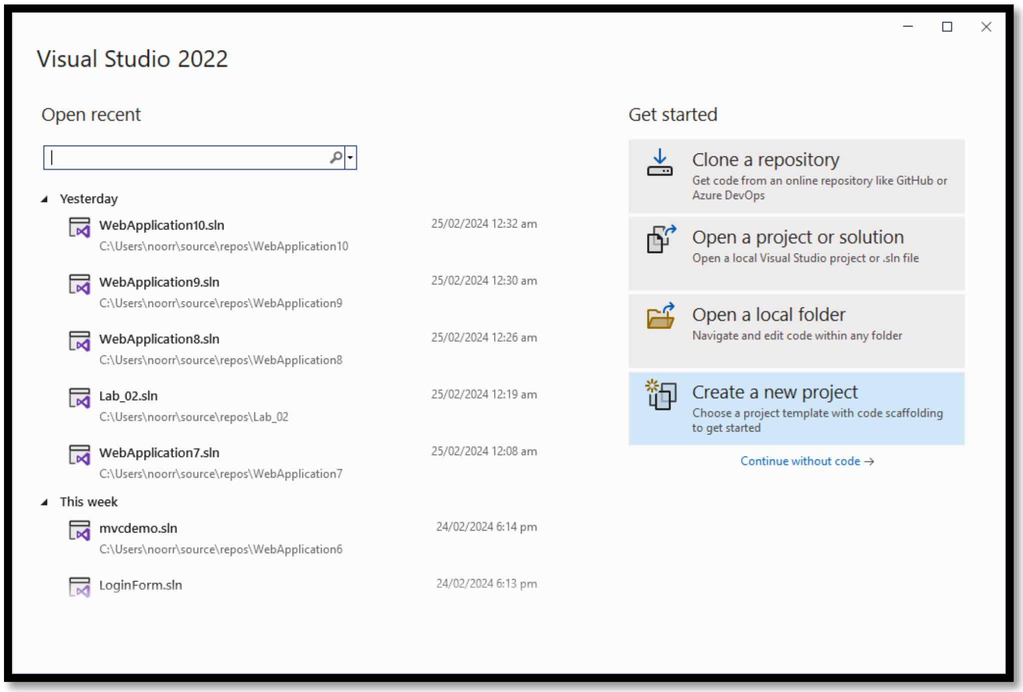
Whether interacting with RESTful APIs, which employ standard HTTP methods for resource manipulation, or SOAP-based web services, which adhere to a more structured messaging protocol, ASP.NET provides a versatile toolkit to efficiently consume and integrate web services. By leveraging these techniques, developers can extend the functionality of their applications, enhance user experiences, and seamlessly access external resources with ease.

Example of Creating Web Service and consuming it in ASP .Net

Firstly, Follow the same example of creating simple web service of calculator as did in previous lecture.

Step 1: Create a new project in Visual Studio 2022

- Open Visual Studio 2022 and Click on "Create a new project" in the start window.



Visual Studio 2022

Open recent

Yesterday

WebApplication10.sln 25/02/2024 12:32 am
C:\Users\noorr\source\repos\WebApplication10

WebApplication9.sln 25/02/2024 12:30 am
C:\Users\noorr\source\repos\WebApplication9

WebApplication8.sln 25/02/2024 12:26 am
C:\Users\noorr\source\repos\WebApplication8

Lab_02.sln 25/02/2024 12:19 am
C:\Users\noorr\source\repos\Lab_02

WebApplication7.sln 25/02/2024 12:08 am
C:\Users\noorr\source\repos\WebApplication7

This week

mvcdemo.sln 24/02/2024 6:14 pm
C:\Users\noorr\source\repos\WebApplication6

LoginForm.sln 24/02/2024 6:13 pm

Get started

Clone a repository
Get code from an online repository like GitHub or Azure DevOps

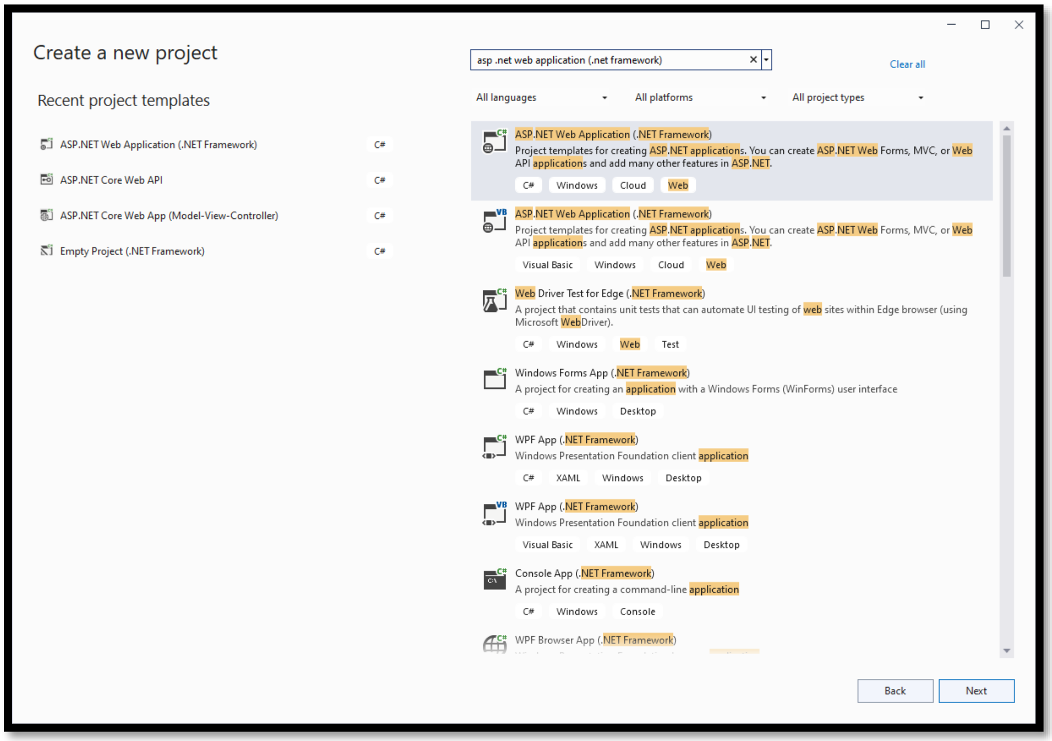
Open a project or solution
Open a local Visual Studio project or .sln file

Open a local folder
Navigate and edit code within any folder

Create a new project
Choose a project template with code scaffolding to get started

Continue without code →

- In the "Create a new project" window, search for **ASP.NET Web Application (.net framework)** in the search bar, and then click NEXT.



Create a new project

asp .net web application (.net framework)

Recent project templates

- ASP.NET Web Application (.NET Framework) C#
- ASP.NET Core Web API C#
- ASP.NET Core Web App (Model-View-Controller) C#
- Empty Project (.NET Framework) C#

All languages | All platforms | All project types

ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
C# Windows Cloud Web

ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
Visual Basic Windows Cloud Web

Web Driver Test for Edge (.NET Framework)
A project that contains unit tests that can automate UI testing of web sites within Edge browser (using Microsoft WebDriver).
C# Windows Web Test

Windows Forms App (.NET Framework)
A project for creating an application with a Windows Forms (WinForms) user interface
C# Windows Desktop

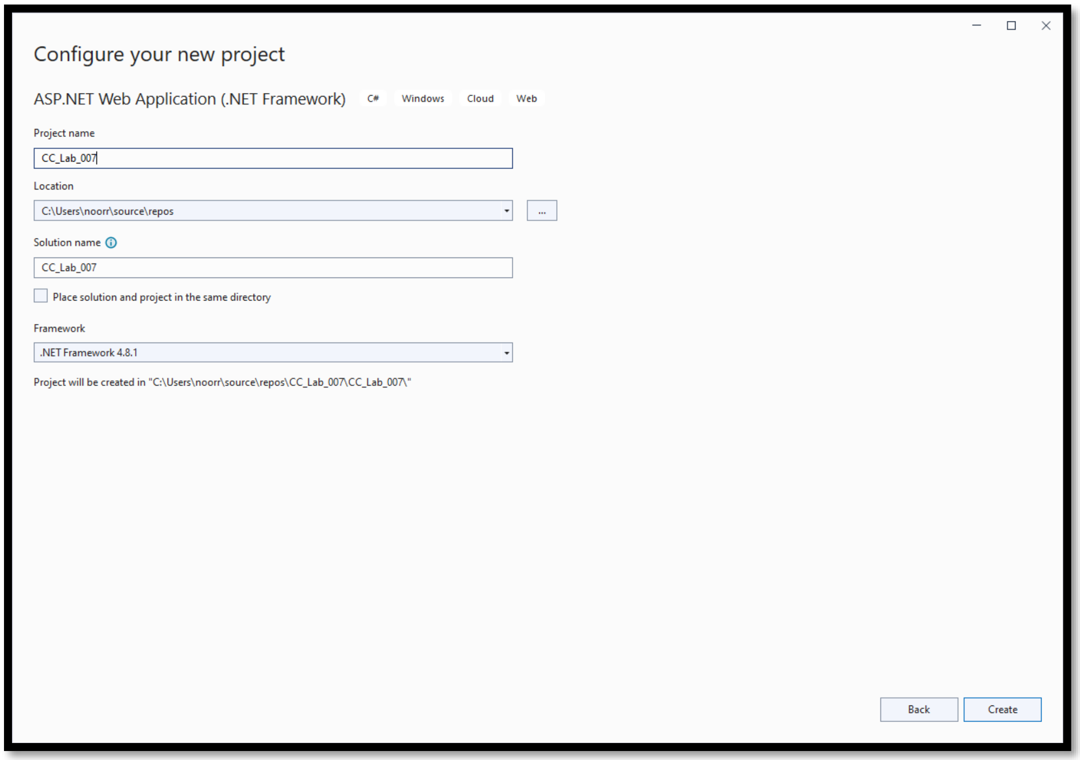
WPF App (.NET Framework)
Windows Presentation Foundation client application
C# XAML Windows Desktop

WPF App (.NET Framework)
Windows Presentation Foundation client application
Visual Basic XAML Windows Desktop

Console App (.NET Framework)
A project for creating a command-line application
C# Windows Console

WPF Browser App (.NET Framework)

- Provide a name and location for your project and then Click NEXT.



Configure your new project

ASP.NET Web Application (.NET Framework) C# Windows Cloud Web

Project name: CC_Lab_007

Location: C:\Users\noorr\source\repos

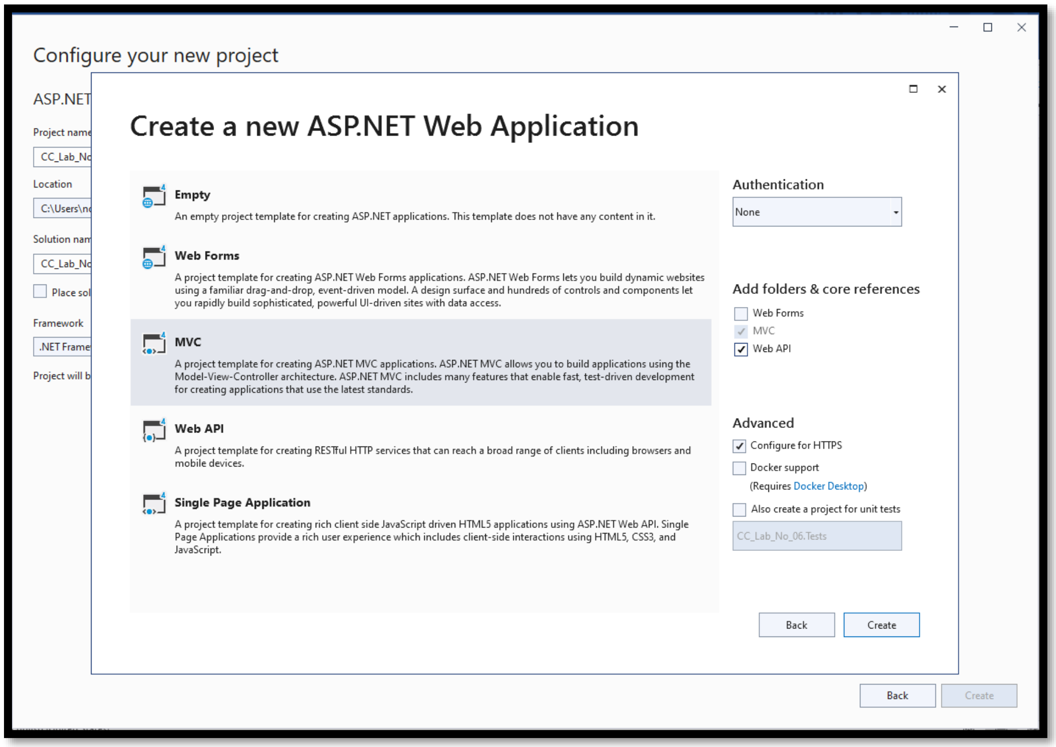
Solution name: CC_Lab_007

☐ Place solution and project in the same directory

Framework: .NET Framework 4.8.1

Project will be created in "C:\Users\noorr\source\repos\CC_Lab_007\CC_Lab_007\"

- Select the **Empty template** and check **web API** option from the list of “add folders and core references”, and Click CREATE.



Configure your new project

Create a new ASP.NET Web Application

Empty
An empty project template for creating ASP.NET applications. This template does not have any content in it.

Web Forms
A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

MVC
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

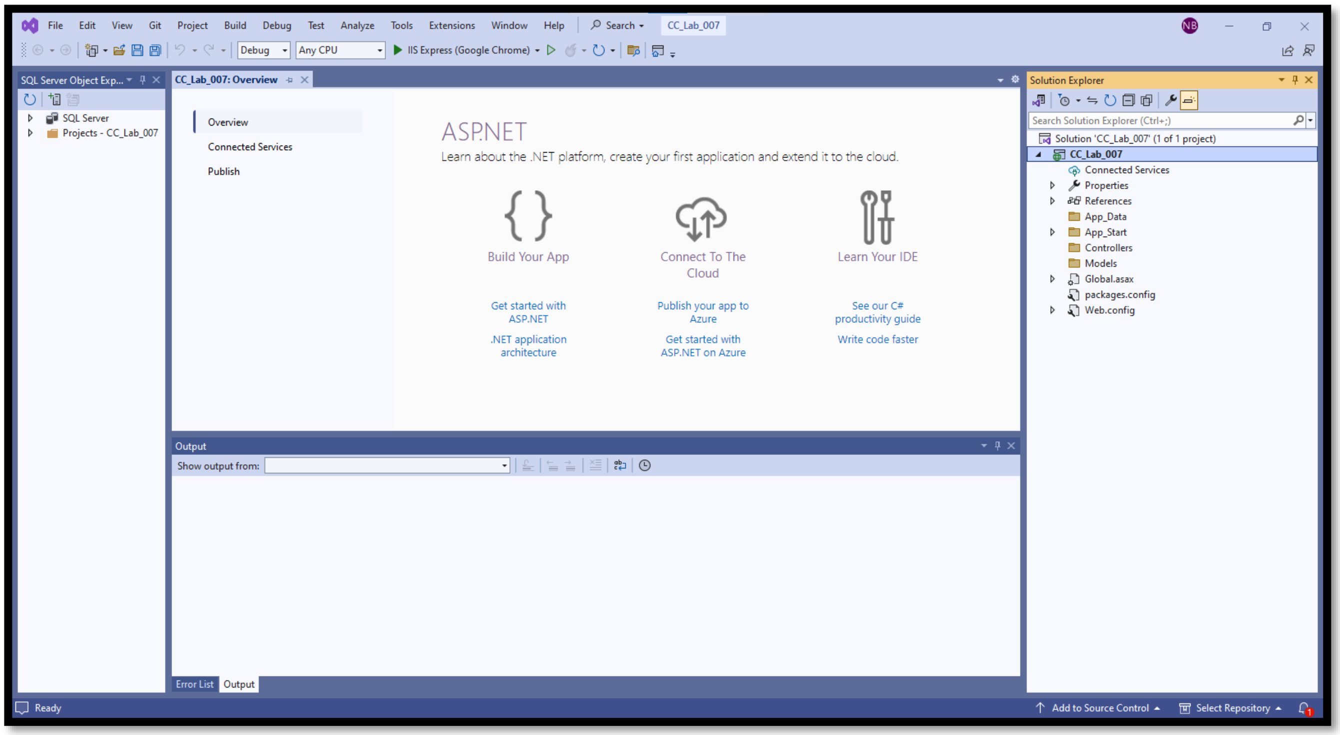
Web API
A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

Single Page Application
A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5, CSS3, and JavaScript.

Authentication: None

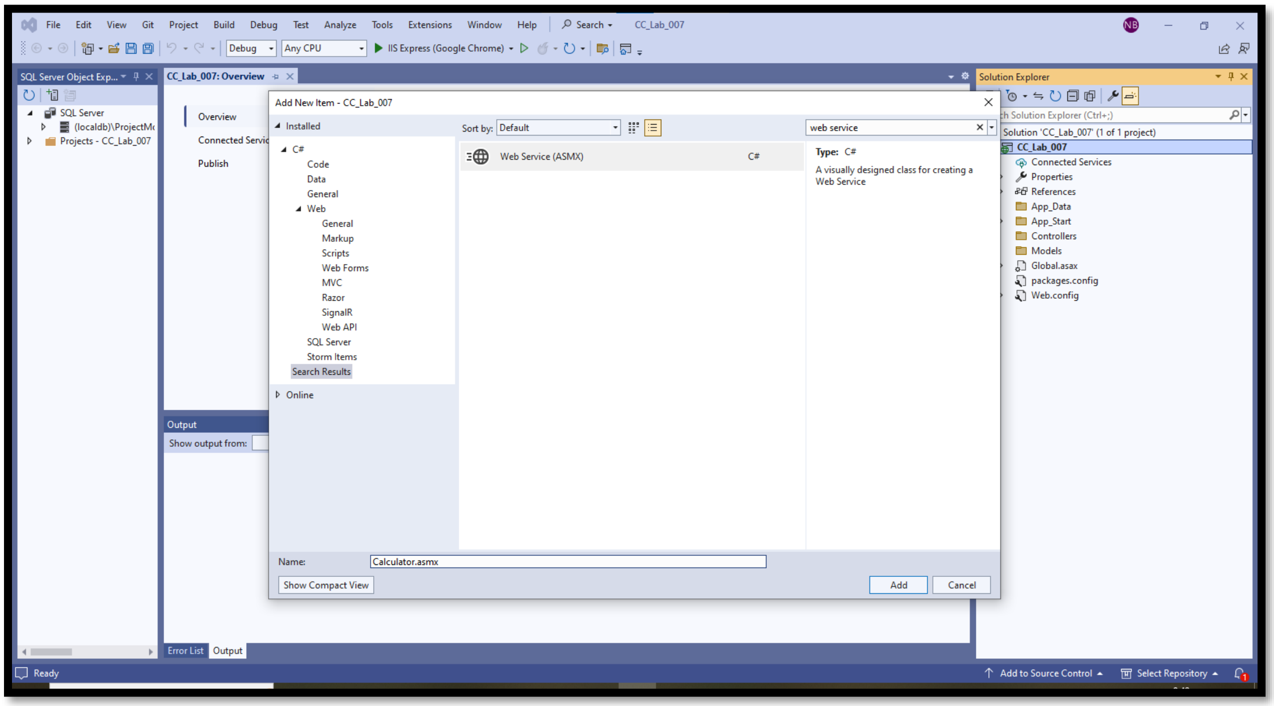
Add folders & core references
- [ ] Web Forms
- [✓] MVC
- [✓] Web API

Advanced
- [✓] Configure for HTTPS
- [ ] Docker support (Requires Docker Desktop)
- [ ] Also create a project for unit tests: CC_Lab_No_06.Tests



Step 2: Adding a Web Service

- Right-click on your project in Solution Explorer, click on add > new item. In the new opened window, search "web service", name it calculator and then add it.


Add New Item - CC_Lab_007

Sort by: Default

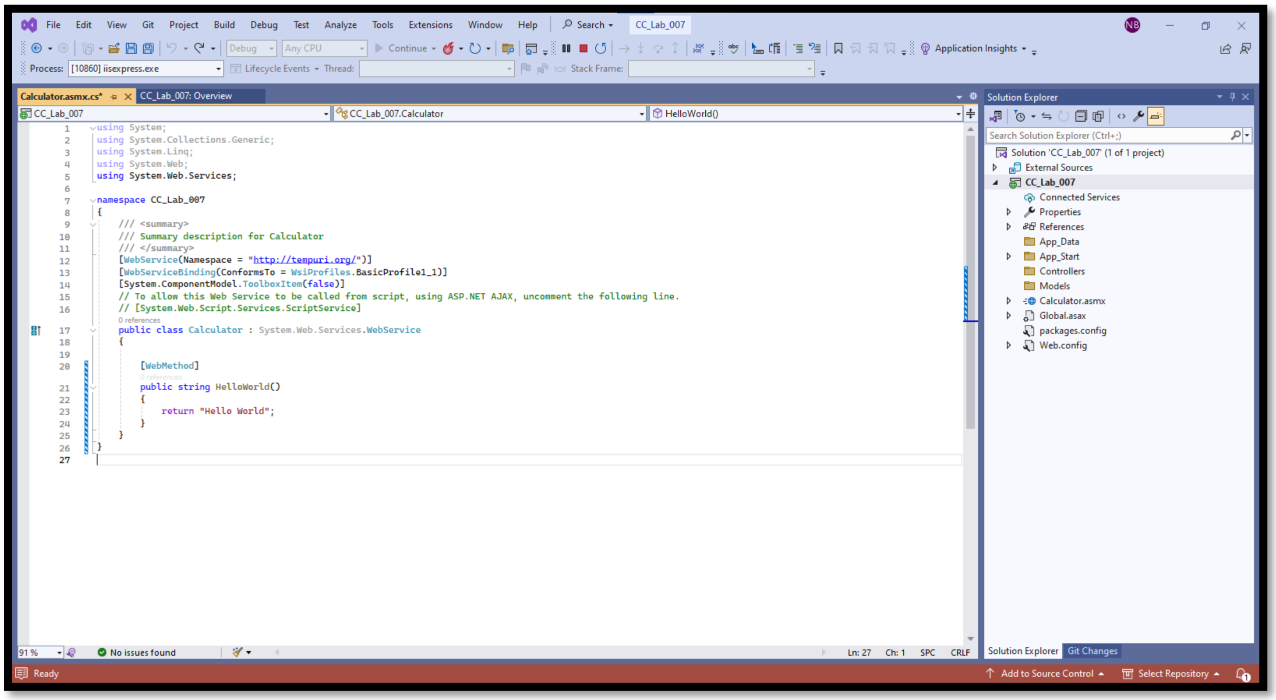
web service

Web Service (ASMX) C#

Type: C#

A visually designed class for creating a Web Service

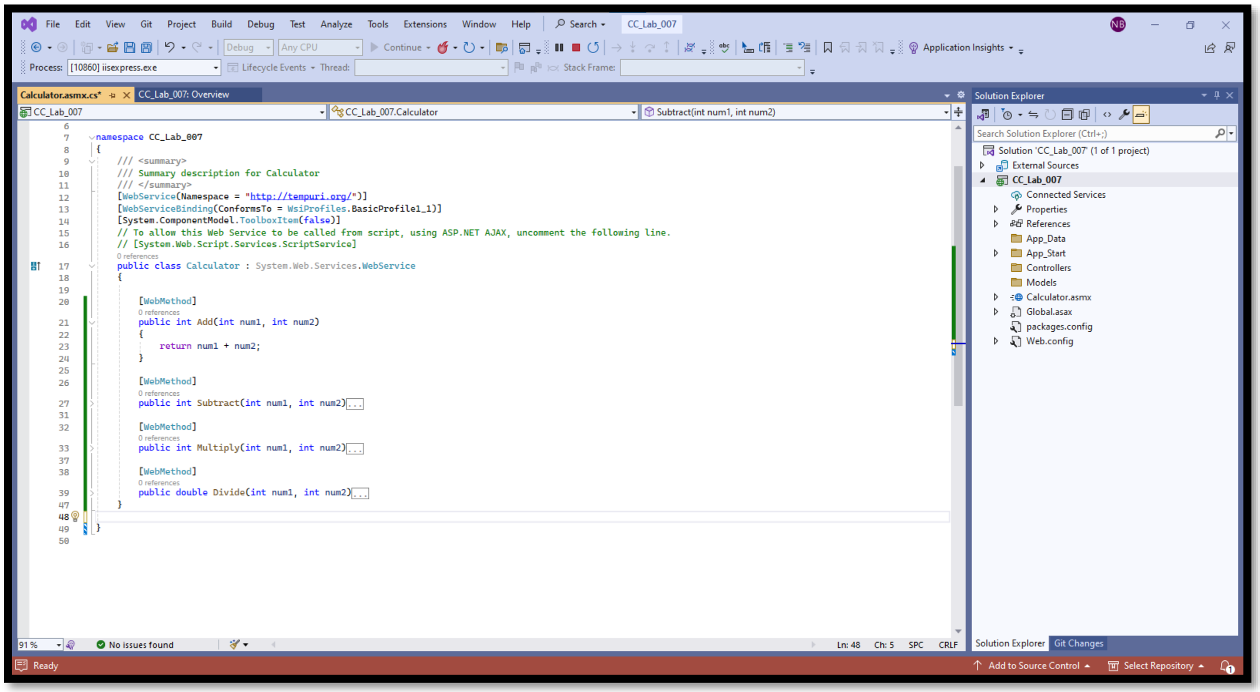
Name: Calculator.asmx


Calculator.asmx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;

namespace CC_Lab_007
{
    /// <summary>
    /// Summary description for Calculator
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.
    // [System.Web.Script.Services.ScriptService]
    public class Calculator : System.Web.Services.WebService
    {

        [WebMethod]
        public string HelloWorld()
        {
            return "Hello World";
        }
    }
}
```

Here (in the above figure), you will note that there is predefined method "**HelloWorld**" which returns the string "**Hello World**". You can use your own method and can perform various operations

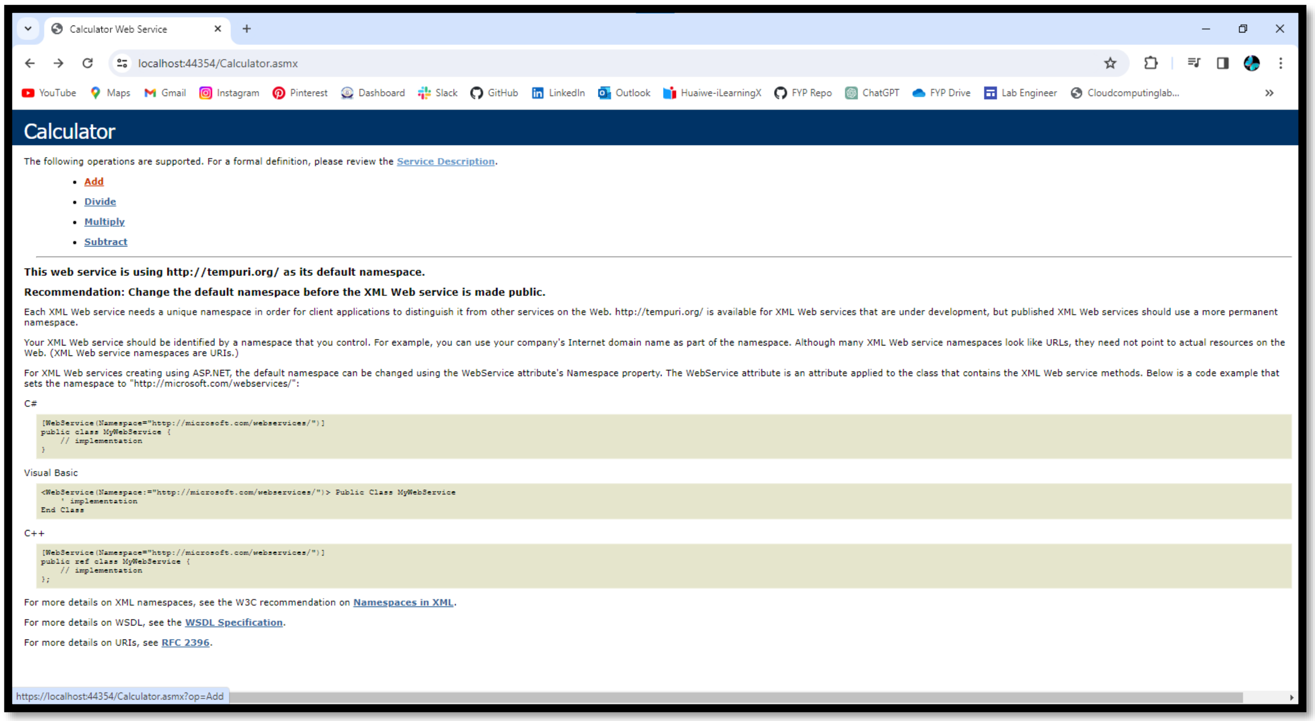
- As we've to make the simple calculator service, so remove the method present in it, and add the methods or essential for simple calculator.



```
namespace CC_Lab_007
{
    /// <summary>
    /// Summary description for Calculator
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.
    // [System.Web.Script.Services.ScriptService]
    public class Calculator : System.Web.Services.WebService
    {

        [WebMethod]
        public int Add(int num1, int num2)
        {
            return num1 + num2;
        }

        [WebMethod]
        public int Subtract(int num1, int num2)[...]

        [WebMethod]
        public int Multiply(int num1, int num2)[...]

        [WebMethod]
        public double Divide(int num1, int num2)[...]
    }

}
```

Step 3: Build and run the application

- Build Web Service and Run the Web Service for testing by pressing F5 function key.



localhost:44354/Calculator.asmx

Calculator

The following operations are supported. For a formal definition, please review the Service Description.

- Add
- Divide
- Multiply
- Subtract

This web service is using http://tempuri.org/ as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)

For XML Web services creating using ASP.NET, the default namespace can be changed using the WebService attribute's Namespace property. The WebService attribute is an attribute applied to the class that contains the XML Web service methods. Below is a code example that sets the namespace to "http://microsoft.com/webservices/":

C#

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public class MyWebService {
    // implementation
}
```

Visual Basic

```
<WebService(Namespace:="http://microsoft.com/webservices/")> Public Class MyWebService
    ' implementation
End Class
```

C++

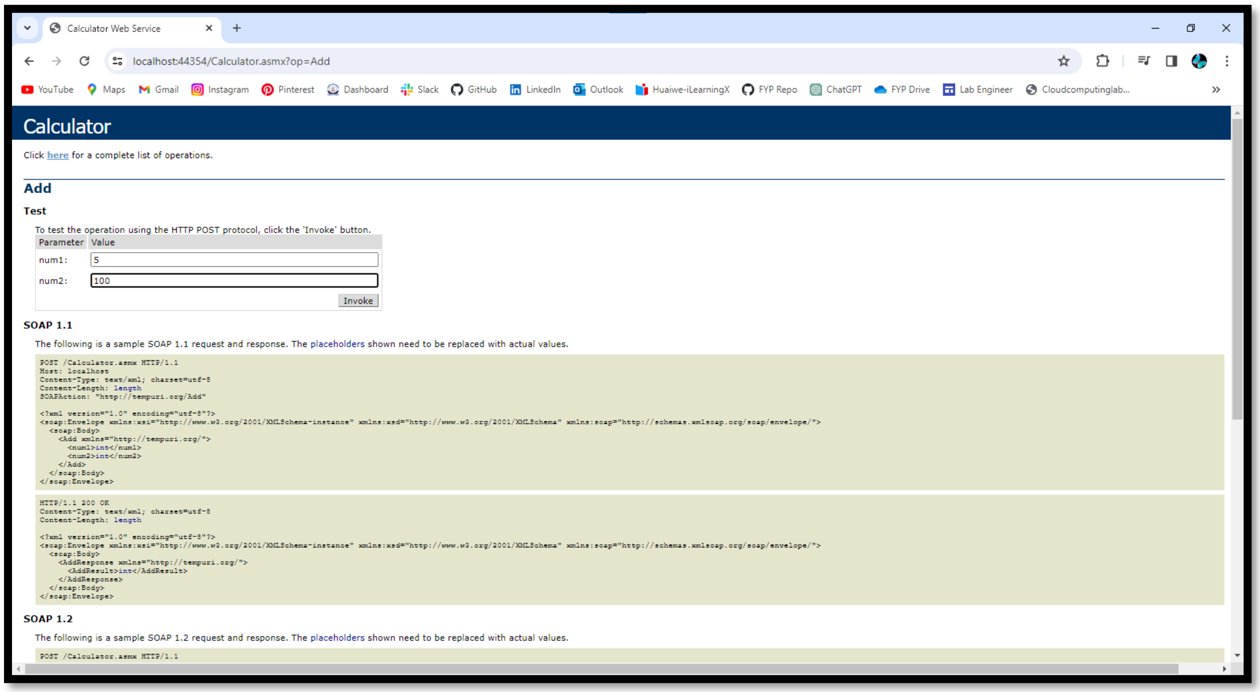
```
[WebService(Namespace="http://microsoft.com/webservices/")]
public ref class MyWebService {
    // implementation
};
```

For more details on XML namespaces, see the W3C recommendation on Namespaces in XML.

For more details on WSDL, see the WSDL Specification.

For more details on URIs, see RFC 2396.

https://localhost:44354/Calculator.asmx?op=Add



Calculator

Click here for a complete list of operations.

Add

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
num1:	5
num2:	100

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

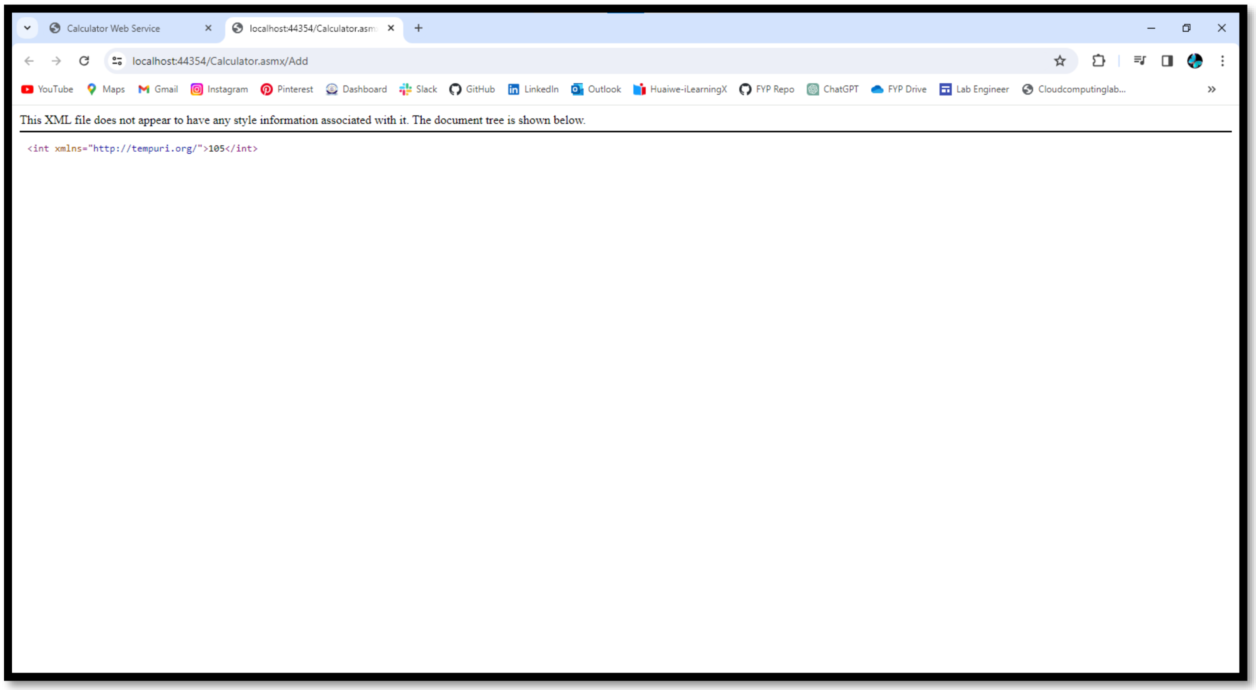
```
POST /Calculator.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/Add"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Add xmlns="http://tempuri.org/">
      <num1>int</num1>
      <num2>int</num2>
    </Add>
  </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <AddResponse xmlns="http://tempuri.org/">
      <AddResult>int</AddResult>
    </AddResponse>
  </soap:Body>
</soap:Envelope>
```

SOAP 1.2

The following is a sample SOAP 1.2 request and response. The placeholders shown need to be replaced with actual values.

```
POST /Calculator.asmx HTTP/1.1
```



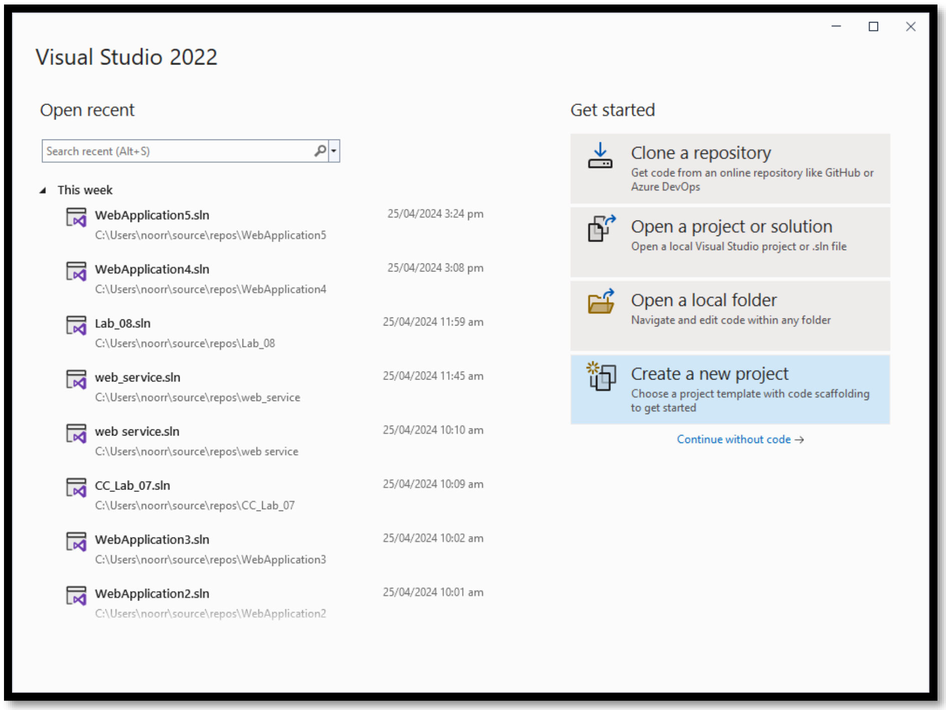
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<int xmlns="http://tempuri.org/">105</int>
```

Now, create the new project to consume the web service created.

Step 1: Create a new project in Visual Studio 2022

- Open Visual Studio 2022 and Click on "Create a new project" in the start window.



Visual Studio 2022

Open recent

Search recent (Alt+S)

This week

WebApplication5.sln C:\Users\noorr\source\repos\WebApplication5	25/04/2024 3:24 pm
WebApplication4.sln C:\Users\noorr\source\repos\WebApplication4	25/04/2024 3:08 pm
Lab_08.sln C:\Users\noorr\source\repos\Lab_08	25/04/2024 11:59 am
web_service.sln C:\Users\noorr\source\repos\web_service	25/04/2024 11:45 am
web service.sln C:\Users\noorr\source\repos\web service	25/04/2024 10:10 am
CC_Lab_07.sln C:\Users\noorr\source\repos\CC_Lab_07	25/04/2024 10:09 am
WebApplication3.sln C:\Users\noorr\source\repos\WebApplication3	25/04/2024 10:02 am
WebApplication2.sln C:\Users\noorr\source\repos\WebApplication2	25/04/2024 10:01 am

Get started

Clone a repository
Get code from an online repository like GitHub or Azure DevOps

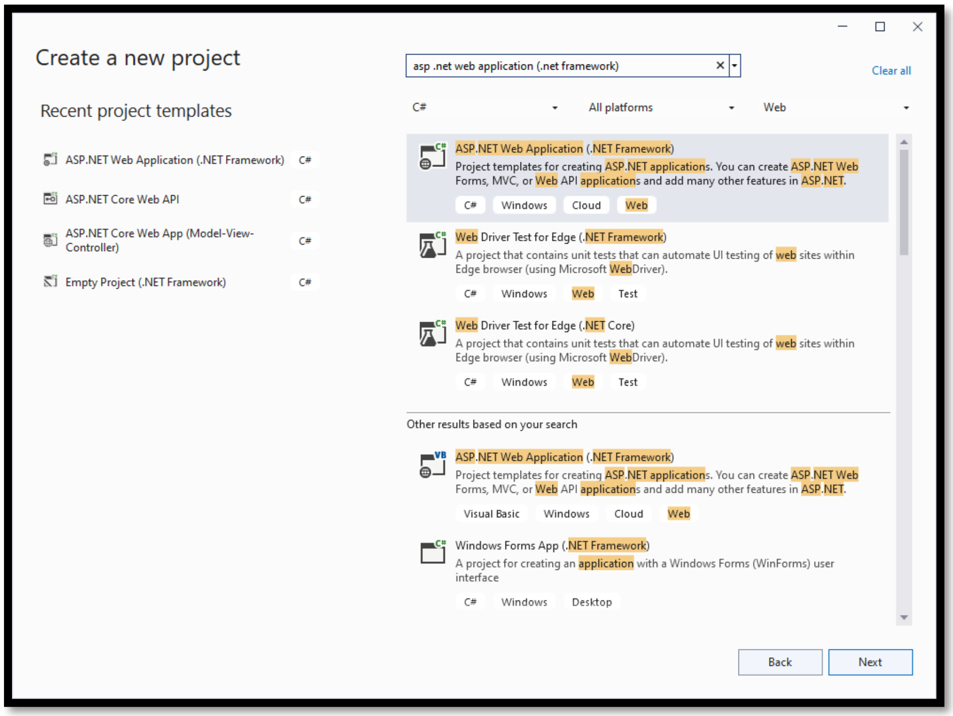
Open a project or solution
Open a local Visual Studio project or .sln file

Open a local folder
Navigate and edit code within any folder

Create a new project
Choose a project template with code scaffolding to get started

Continue without code →

- In the "Create a new project" window, search for **ASP.NET Web Application (.net framework)** in the search bar, and then click NEXT.



Create a new project

asp .net web application (.net framework)

Clear all

C# | All platforms | Web

Recent project templates

ASP.NET Web Application (.NET Framework)	C#
ASP.NET Core Web API	C#
ASP.NET Core Web App (Model-View-Controller)	C#
Empty Project (.NET Framework)	C#

ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
C# Windows Cloud Web

Web Driver Test for Edge (.NET Framework)
A project that contains unit tests that can automate UI testing of web sites within Edge browser (using Microsoft WebDriver).
C# Windows Web Test

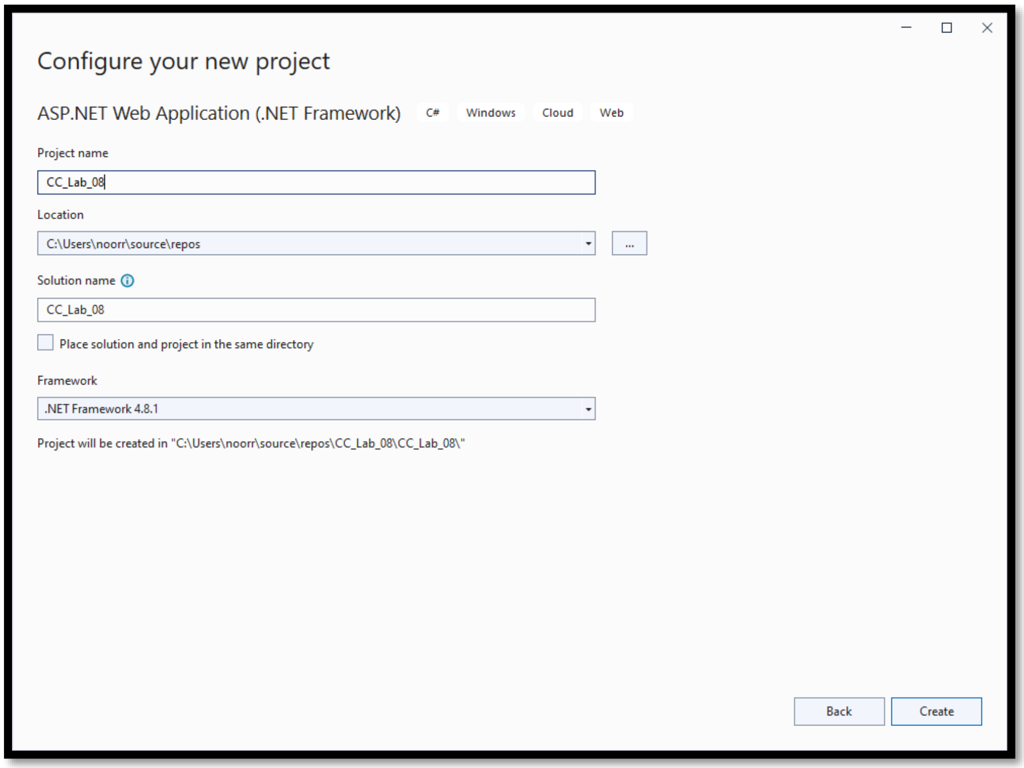
Web Driver Test for Edge (.NET Core)
A project that contains unit tests that can automate UI testing of web sites within Edge browser (using Microsoft WebDriver).
C# Windows Web Test

Other results based on your search

ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
Visual Basic Windows Cloud Web

Windows Forms App (.NET Framework)
A project for creating an application with a Windows Forms (WinForms) user interface
C# Windows Desktop

- Provide a name and location for your project and then Click NEXT.



Configure your new project

ASP.NET Web Application (.NET Framework) C# Windows Cloud Web

Project name: CC_Lab_08

Location: C:\Users\noorr\source\repos

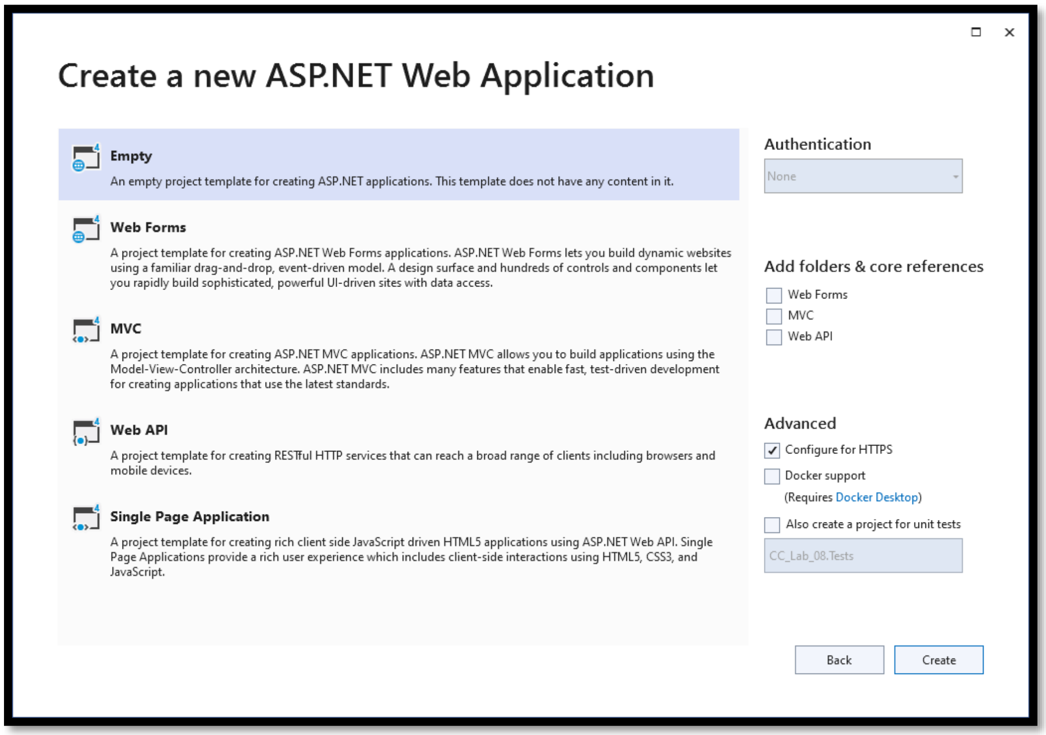
Solution name: CC_Lab_08

☐ Place solution and project in the same directory

Framework: .NET Framework 4.8.1

Project will be created in "C:\Users\noorr\source\repos\CC_Lab_08\CC_Lab_08\"

- Select the **Empty template** and Click CREATE.



Create a new ASP.NET Web Application

Empty
An empty project template for creating ASP.NET applications. This template does not have any content in it.

Web Forms
A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

MVC
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Web API
A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

Single Page Application
A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5, CSS3, and JavaScript.

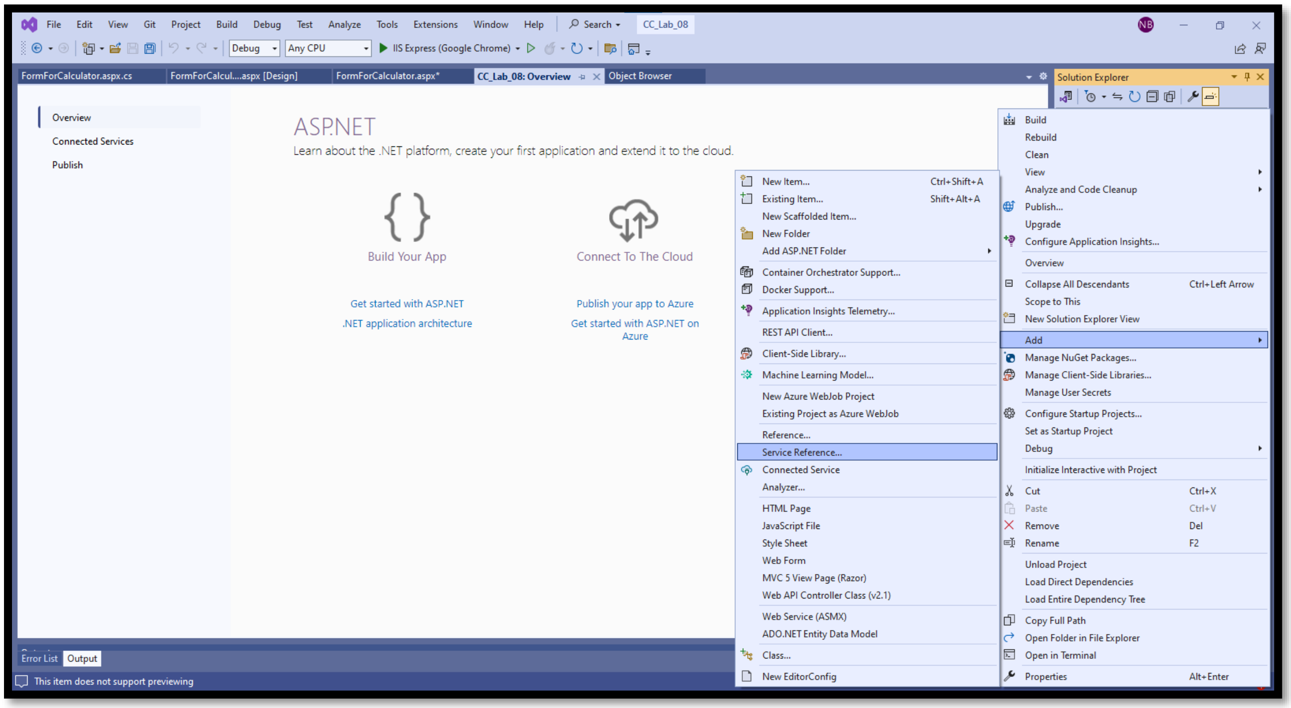
Authentication: None

Add folders & core references
☐ Web Forms
☐ MVC
☐ Web API

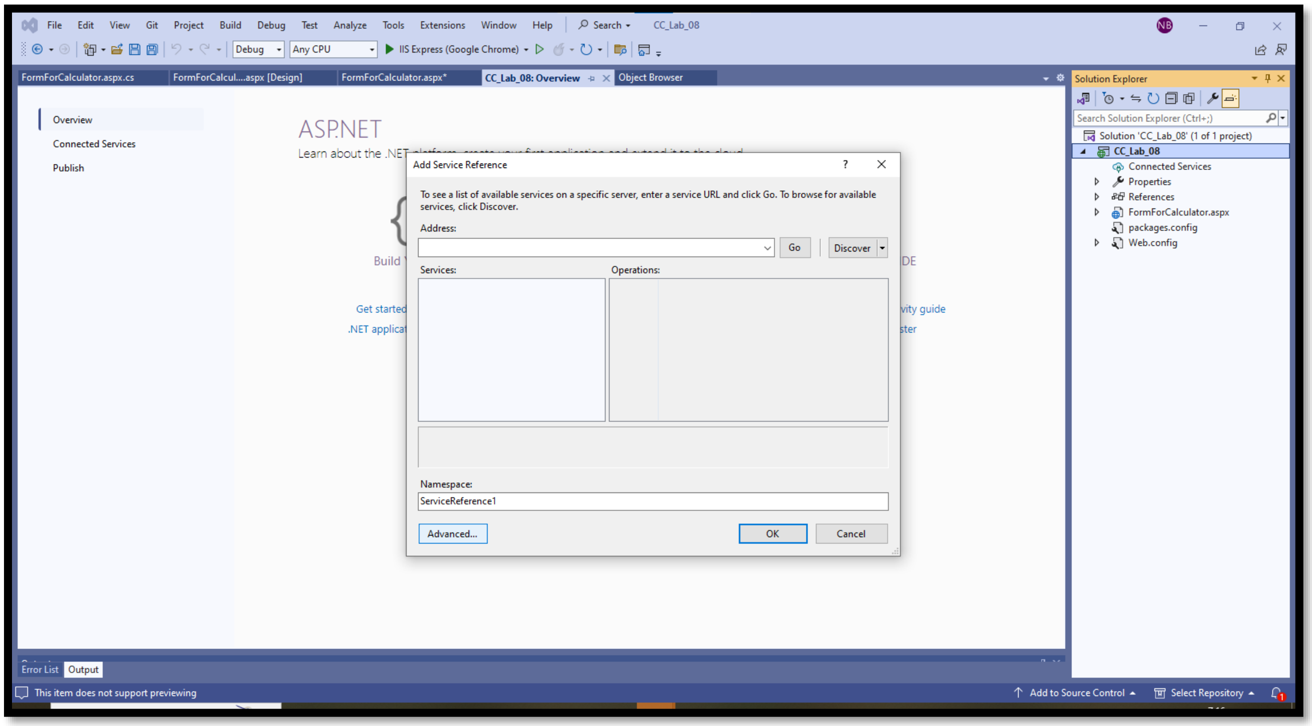
Advanced
☑ Configure for HTTPS
☐ Docker support
(Requires Docker Desktop)
☐ Also create a project for unit tests
CC_Lab_08.Tests

Step 2: Add a web service reference in Visual Studio 2022

- Right-click on your project in Solution Explorer, click on "Add Service Reference" as in thefollowing:


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search CC_Lab_08
Debug Any CPU IIS Express (Google Chrome)
FormForCalculator.aspx.cs FormForCalcul....aspx [Design] FormForCalculator.aspx* CC_Lab_08: Overview Object Browser
Overview
Connected Services
Publish
ASP.NET
Learn about the .NET platform, create your first application and extend it to the cloud.
Build Your App
Connect To The Cloud
Get started with ASP.NET
.NET application architecture
Publish your app to Azure
Get started with ASP.NET on Azure
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Scaffolded Item...
New Folder
Add ASP.NET Folder
Container Orchestrator Support...
Docker Support...
Application Insights Telemetry...
REST API Client...
Client-Side Library...
Machine Learning Model...
New Azure WebJob Project
Existing Project as Azure WebJob
Reference...
Service Reference...
Connected Service
Analyzer...
HTML Page
JavaScript File
Style Sheet
Web Form
MVC 5 View Page (Razor)
Web API Controller Class (v2.1)
Web Service (ASMX)
ADO.NET Entity Data Model
Class...
New EditorConfig
Solution Explorer
Build
Rebuild
Clean
View
Analyze and Code Cleanup
Publish...
Upgrade
Configure Application Insights...
Overview
Collapse All Descendants Ctrl+Left Arrow
Scope to This
New Solution Explorer View
Add
Manage NuGet Packages...
Manage Client-Side Libraries...
Manage User Secrets
Configure Startup Projects...
Set as Startup Project
Debug
Initialize Interactive with Project
Cut Ctrl+X
Paste Ctrl+V
Remove Del
Rename F2
Unload Project
Load Direct Dependencies
Load Entire Dependency Tree
Copy Full Path
Open Folder in File Explorer
Open in Terminal
Properties Alt+Enter
Error List Output
This item does not support previewing

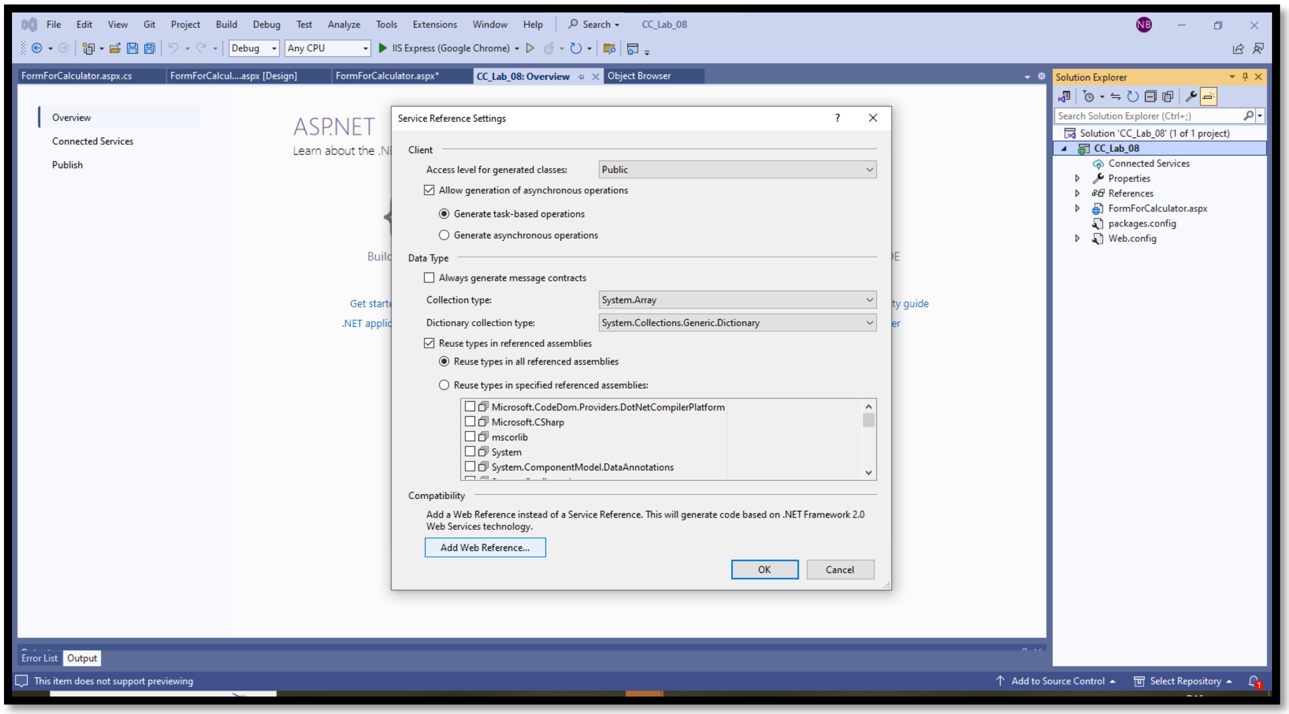
- Then after clicking on the above option, following window will appear, then click on the "Advanced" tab.


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search CC_Lab_08
Debug Any CPU IIS Express (Google Chrome)
FormForCalculator.aspx.cs FormForCalcul....aspx [Design] FormForCalculator.aspx* CC_Lab_08: Overview Object Browser
Overview
Connected Services
Publish
ASP.NET
Add Service Reference
Address:
Go
Discover
Services:
Operations:
Namespace:
Advanced...
OK
Cancel
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'CC_Lab_08' (1 of 1 project)
CC_Lab_08
Connected Services
Properties
References
FormForCalculator.aspx
packages.config
Web.config
Error List Output
This item does not support previewing
Add to Source Control
Select Repository

To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Namespace: ServiceReference1

- Now after clicking on the Advanced tab, it will show the following window then click on the "Add Web Reference" option :



Service Reference Settings

Client

Access level for generated classes: Public

[✓] Allow generation of asynchronous operations

(●) Generate task-based operations

( ) Generate asynchronous operations

Data Type

[ ] Always generate message contracts

Collection type: System.Array

Dictionary collection type: System.Collections.Generic.Dictionary

[✓] Reuse types in referenced assemblies

(●) Reuse types in all referenced assemblies

( ) Reuse types in specified referenced assemblies:

[ ] Microsoft.CodeDom.Providers.DotNetCompilerPlatform
[ ] Microsoft.CSharp
[ ] mscorlib
[ ] System
[ ] System.ComponentModel.DataAnnotations

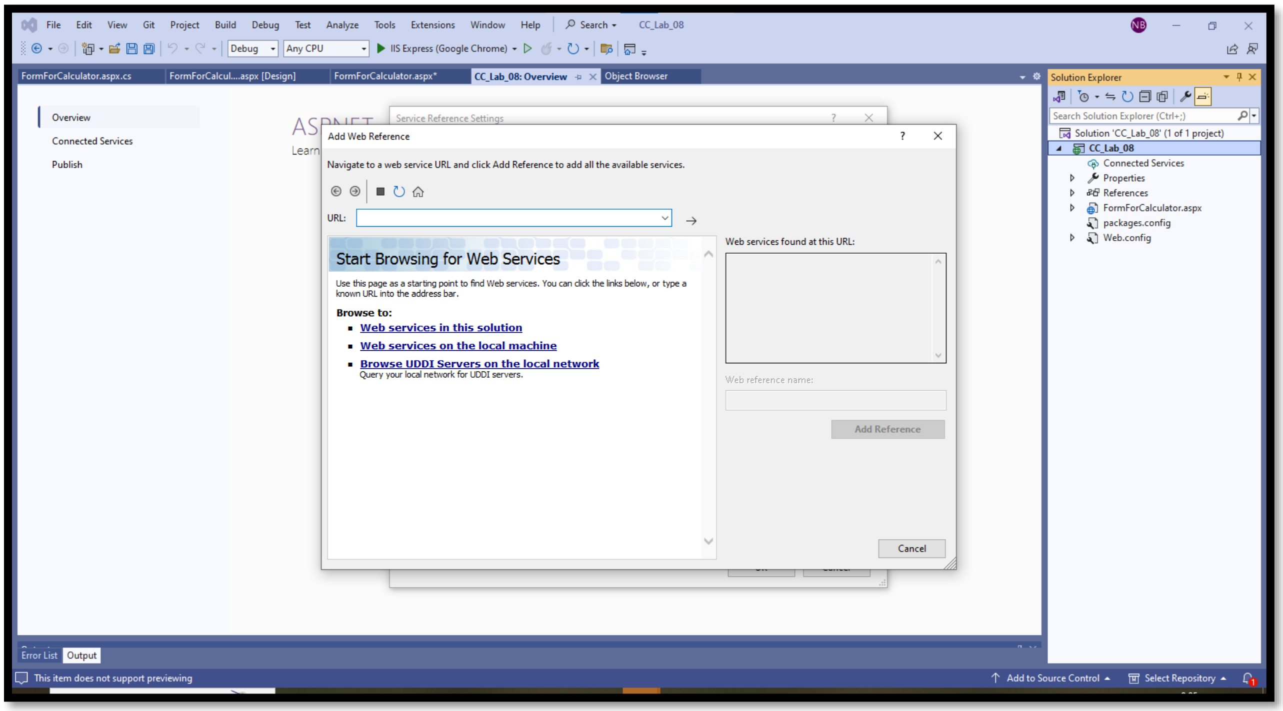
Compatibility

Add a Web Reference instead of a Service Reference. This will generate code based on .NET Framework 2.0 Web Services technology.

Add Web Reference...

OK Cancel

- After clicking on the Add Web Reference tab, it will show the following window. Now this is a very important step, when adding the web reference to the ASP.Net web Application. Since you see "URL" option in the following window, on that window we need to paste or type the Web Service URL address.



Add Web Reference

Navigate to a web service URL and click Add Reference to add all the available services.

URL:

Start Browsing for Web Services

Use this page as a starting point to find Web services. You can click the links below, or type a known URL into the address bar.

Browse to:

- Web services in this solution
- Web services on the local machine
- Browse UDDI Servers on the local network
  Query your local network for UDDI servers.

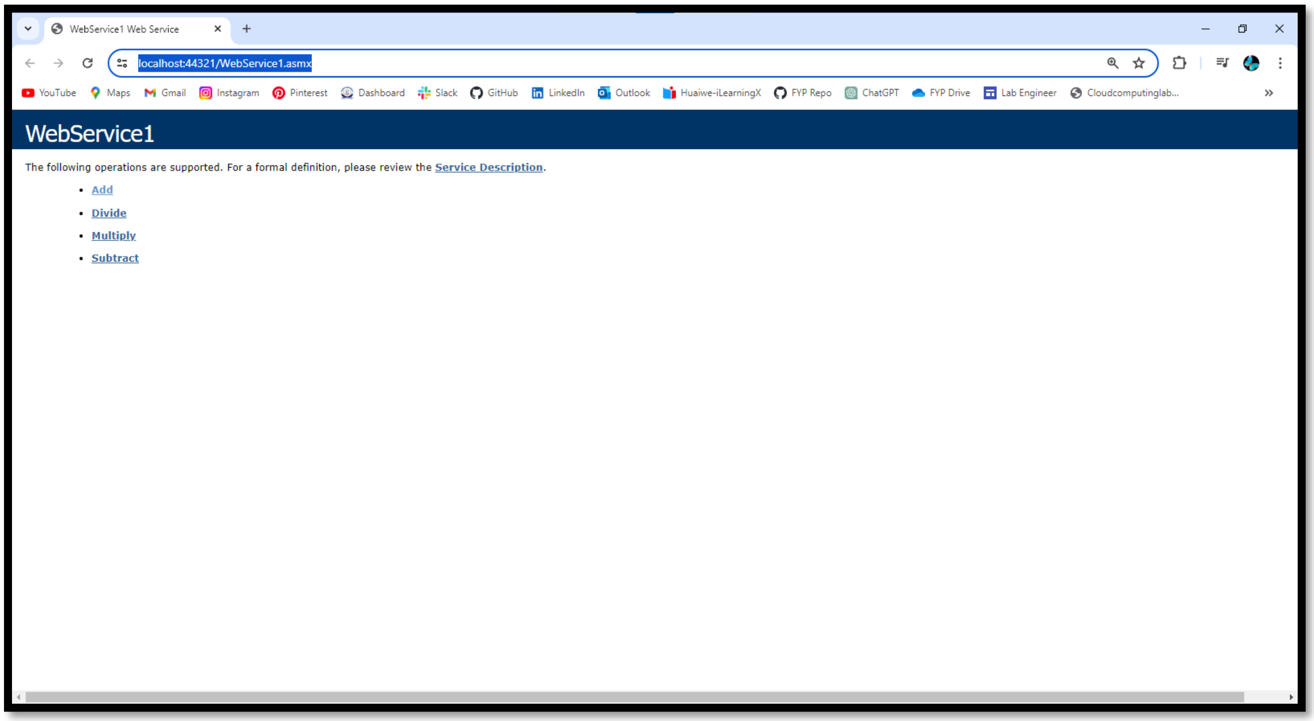
Web services found at this URL:

Web reference name:

Add Reference

Cancel

- **For adding the URL,** run the Web Service we created and copy the URL as shown below:



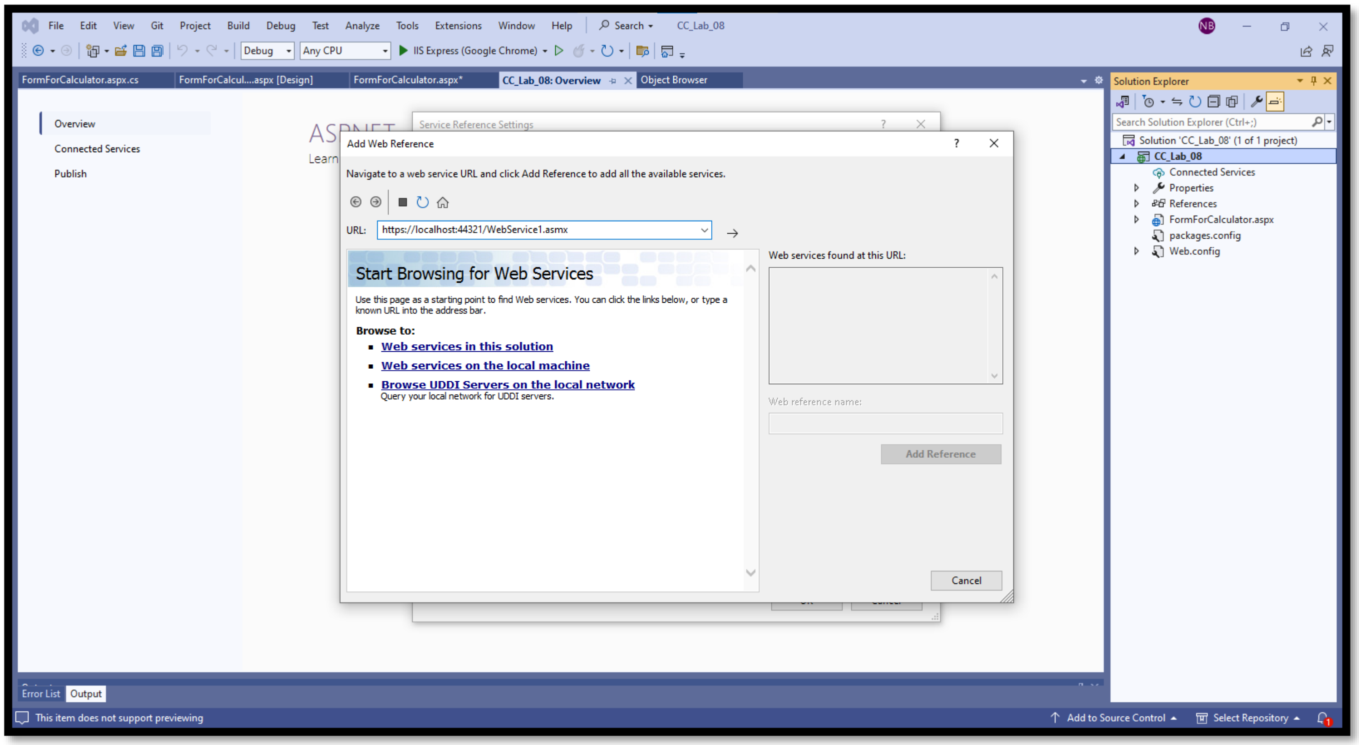
localhost:44321/WebService1.asmx

WebService1

The following operations are supported. For a formal definition, please review the Service Description.

- Add
- Divide
- Multiply
- Subtract

- As you clearly see there, in the preceding window, it displays the methods named "Add", "Divide", "Multiplier" and "Subtract", as we created in our Web Service, now just you need to copy the preceding URL and paste it into the previous Step's window URL option, then the Step 4 window will look as in the following:



Add Web Reference

Navigate to a web service URL and click Add Reference to add all the available services.

URL: https://localhost:44321/WebService1.asmx

Start Browsing for Web Services

Use this page as a starting point to find Web services. You can click the links below, or type a known URL into the address bar.

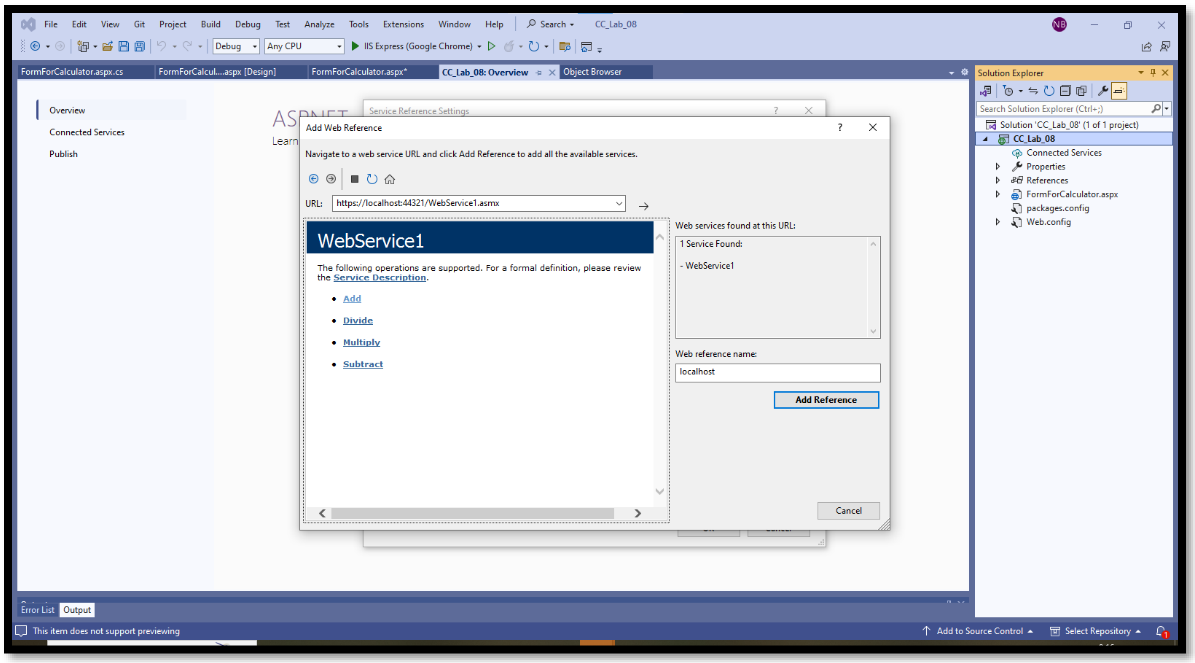
Browse to:
- Web services in this solution
- Web services on the local machine
- Browse UDDI Servers on the local network
  Query your local network for UDDI servers.

Web services found at this URL:

Web reference name:

- After pasting the URL in the preceding window box, click on right headed arrow button, it will discover the Web Services available related to that URL address and you see that in that related URL one Web Service is found message is displayed along with the Web Service name, "Web Service 1" in the preceding right hand side window.

  The Name of the Web Service is "WebService 1" because I have given the class name as Web Service, that's why the name is Web Service 1, in your case it might be different or the class name is anything so you can use any name for Web Service so don't be confused about it. In the right hand corner of the window you have seen the option for the Web reference name; theweb reference name is anything you wish and this name will be added in your ASP.Net Web Application as allies name for Web Service. In my article I have given the web reference name as"local host".



Add Web Reference

Navigate to a web service URL and click Add Reference to add all the available services.

URL: https://localhost:44321/WebService1.asmx

WebService1

The following operations are supported. For a formal definition, please review the Service Description.

- Add
- Divide
- Multiply
- Subtract

Web services found at this URL:

1 Service Found:

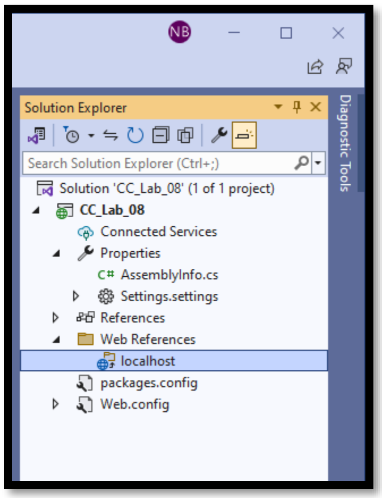
- WebService1

Web reference name: localhost

Add Reference

Cancel

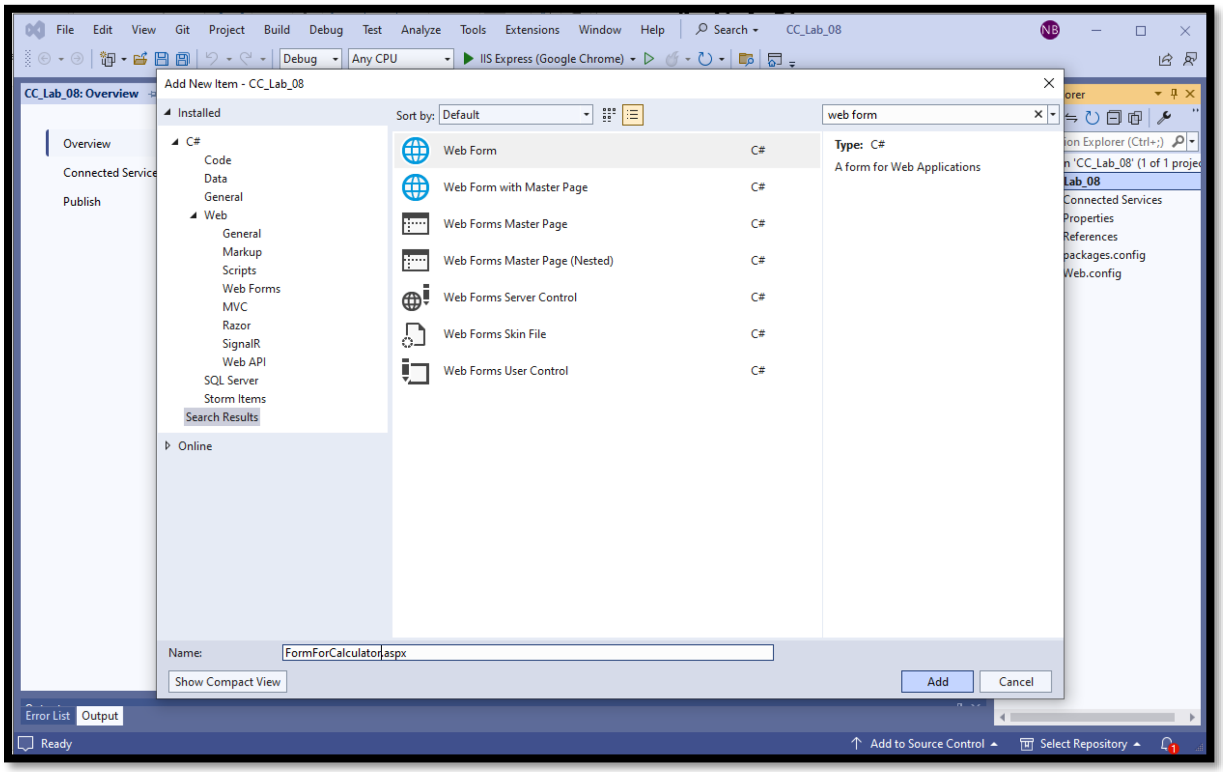
- Then after adding the Web Service reference in the ASP.Net web application the Solution Explorer will look as in the following:



Solution 'CC_Lab_08' (1 of 1 project)
CC_Lab_08
Connected Services
Properties
AssemblyInfo.cs
Settings.settings
References
Web References
localhost
packages.config
Web.config

Step 3: Add a simple web form in Visual Studio 2022

- Right-click on your project in Solution Explorer, click on add > new item. In the new opened window, search “web form”, name it “FormForCalculator” and then add it.



Add New Item - CC_Lab_08

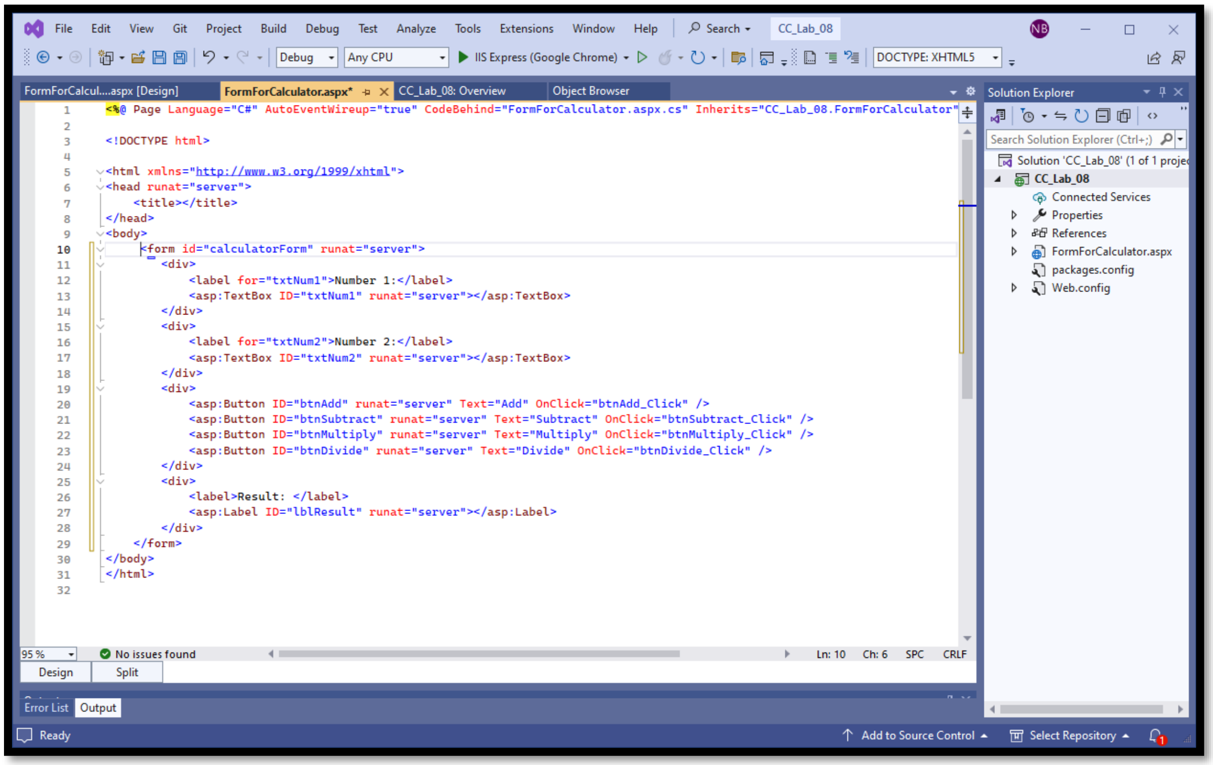
Sort by: Default — search: web form

Web Form	C#
Web Form with Master Page	C#
Web Forms Master Page	C#
Web Forms Master Page (Nested)	C#
Web Forms Server Control	C#
Web Forms Skin File	C#
Web Forms User Control	C#

Type: C#
A form for Web Applications

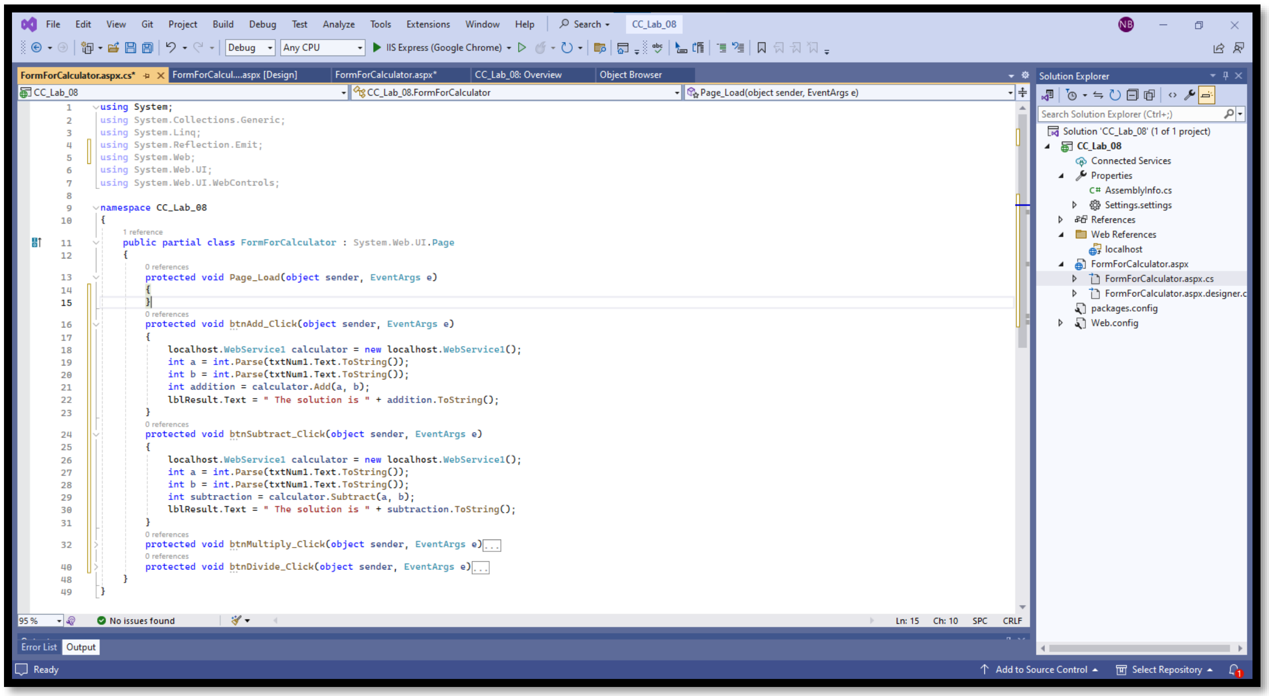
Name: FormForCalculator.aspx

- Then add the required textboxes, buttons and labels onto the created web form.



```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="FormForCalculator.aspx.cs" Inherits="CC_Lab_08.FormForCalculator"

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="calculatorForm" runat="server">
        <div>
            <label for="txtNum1">Number 1:</label>
            <asp:TextBox ID="txtNum1" runat="server"></asp:TextBox>
        </div>
        <div>
            <label for="txtNum2">Number 2:</label>
            <asp:TextBox ID="txtNum2" runat="server"></asp:TextBox>
        </div>
        <div>
            <asp:Button ID="btnAdd" runat="server" Text="Add" OnClick="btnAdd_Click" />
            <asp:Button ID="btnSubtract" runat="server" Text="Subtract" OnClick="btnSubtract_Click" />
            <asp:Button ID="btnMultiply" runat="server" Text="Multiply" OnClick="btnMultiply_Click" />
            <asp:Button ID="btnDivide" runat="server" Text="Divide" OnClick="btnDivide_Click" />
        </div>
        <div>
            <label>Result: </label>
            <asp:Label ID="lblResult" runat="server"></asp:Label>
        </div>
    </form>
</body>
</html>
```

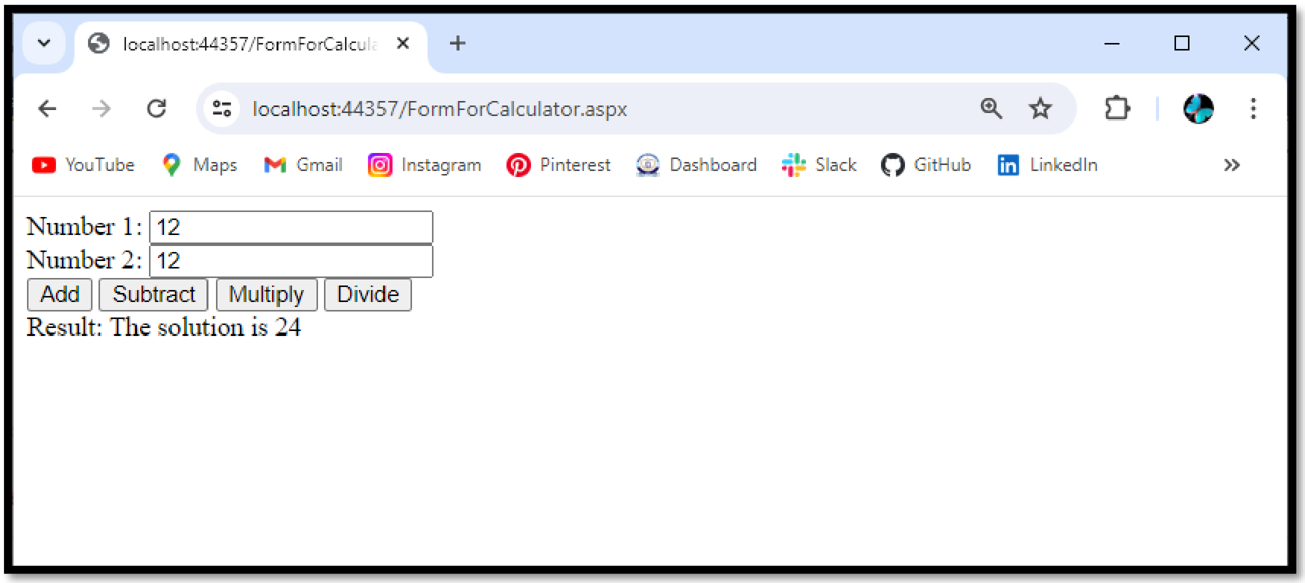
- Now implement the functionality behind the buttons in the functions named in "On Click" via consuming web service. For this open .aspx.cs file of the web form.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection.Emit;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace CC_Lab_08
{
    public partial class FormForCalculator : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void btnAdd_Click(object sender, EventArgs e)
        {
            localhost.WebService1 calculator = new localhost.WebService1();
            int a = int.Parse(txtNum1.Text.ToString());
            int b = int.Parse(txtNum1.Text.ToString());
            int addition = calculator.Add(a, b);
            lblResult.Text = " The solution is " + addition.ToString();
        }
        protected void btnSubtract_Click(object sender, EventArgs e)
        {
            localhost.WebService1 calculator = new localhost.WebService1();
            int a = int.Parse(txtNum1.Text.ToString());
            int b = int.Parse(txtNum1.Text.ToString());
            int subtraction = calculator.Subtract(a, b);
            lblResult.Text = " The solution is " + subtraction.ToString();
        }
        protected void btnMultiply_Click(object sender, EventArgs e)...
        protected void btnDivide_Click(object sender, EventArgs e)...
    }
}
```

Step 3: Run the application

- Run the Application by pressing F5 function key.



localhost:44357/FormForCalculator.aspx

Number 1: 12
Number 2: 12
Add Subtract Multiply Divide
Result: The solution is 24

2. Time Boxing

Activity Name	Activity Time	Total Time
Login Systems + Setting up Visual studio Environment	3 mints + 5 mints	8 mints
Walk through Theory & Tasks	60 mints	60 mints
Implement Tasks	80 mints	80 mints
Evaluation Time	30 mints	30 mints
	Total Duration	178 mints

3. Objectives

After completing this lab the student should be able to:

a. *Learn to integrate external functionalities into ASP.NET applications by consuming web services.*

b. *Understand the vital role of web service consumption in enabling seamless communication between different software applications.*

c. *Acquire the skills to create service references and implement service clients for consuming web services in ASP.NET applications.*

4. Lab Tasks/Practical Work

1. Develop a web service in ASP.NET to retrieve current weather data based on location. Then build an ASP.NET web application to consume the weather service and display current weather information for a given location.
2. Design and implement a web service that solves quadratic equations using different method (atleast 3), then build an ASP.NET web application to consume this service.