Advanced Web Technologies
Module 5 & 6: Web Services and WCF & ASP.NET MVC

Assignment 3

Web Services and WCF

1. Write a program to create a web service that performs arithmetic operations, and also create a client to call that web service.

Aim: To develop a web service for performing arithmetic operations and create a client to utilize the service

Objectives:

1. Implement a web service using ASP.NET WCF to perform arithmetic operations.
2. Create a client application to consume the web service.
3. Demonstrate the communication between the client and the web service for executing arithmetic operations.

Theory:

Web Services: Web services are software systems designed to support interoperable machine-to-machine interaction over a network.

WCF (Windows Communication Foundation): A framework for building service-oriented applications, enabling secure and reliable communication between distributed systems.

ASP.NET MVC (Model-View-Controller): A web application framework that implements the model-view-controller pattern, providing a structured way to build dynamic web applications.

Code:

> WebService1.asmx.cs

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

namespace WebApplication1
{
    /// <summary>
```

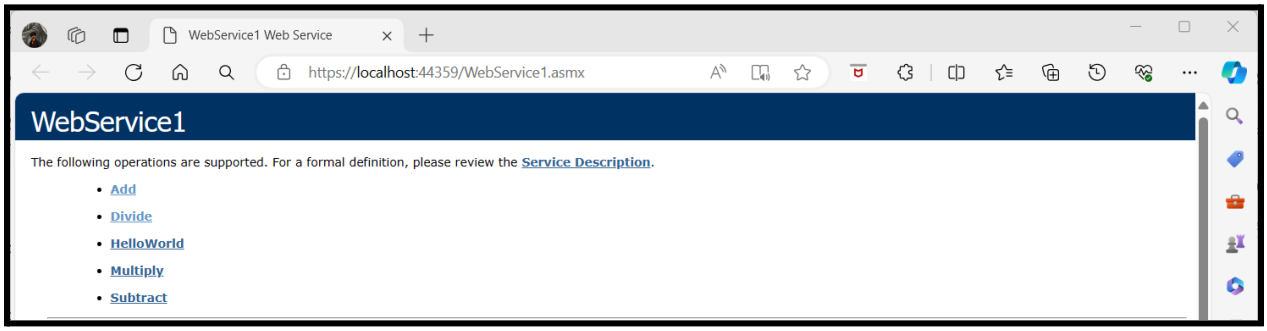
```
    /// Summary description for WebService1
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
     // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the
following line.
    // [System.Web.Script.Services.ScriptService]
    public class WebService1 : System.Web.Services.WebService
    {

        [WebMethod]
        public string HelloWorld()
        {
            return "Hello World";
        }
        [WebMethod]
        public int Add(int a, int b)
        {
            return a + b;
        }

        [WebMethod]
        public int Subtract(int a, int b)
        {
            return a - b;
        }

        [WebMethod]
        public int Multiply(int a, int b)
        {
            return a * b;
        }

        [WebMethod]
        public int Divide(int a, int b)
        {
            return a / b;
        }
    }
}
```

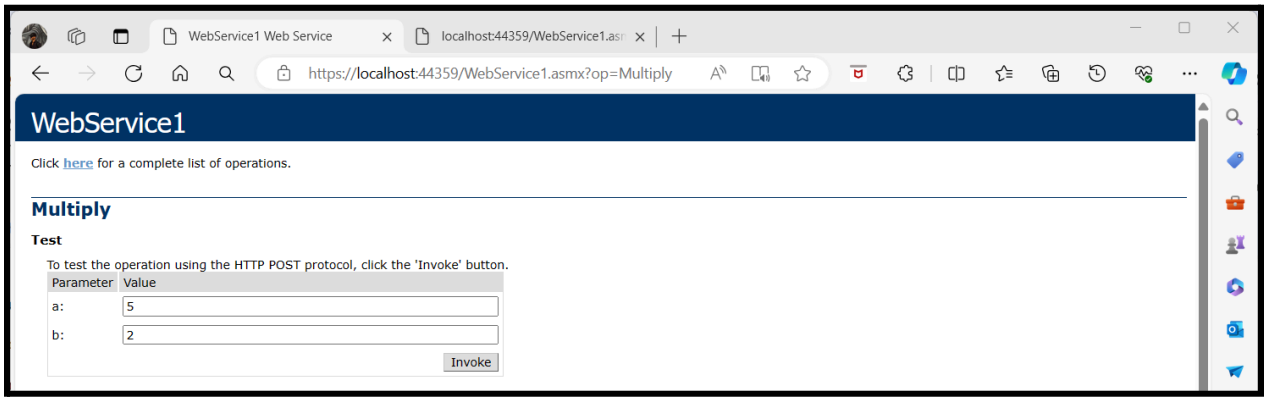
> WebForm1.aspx.cs

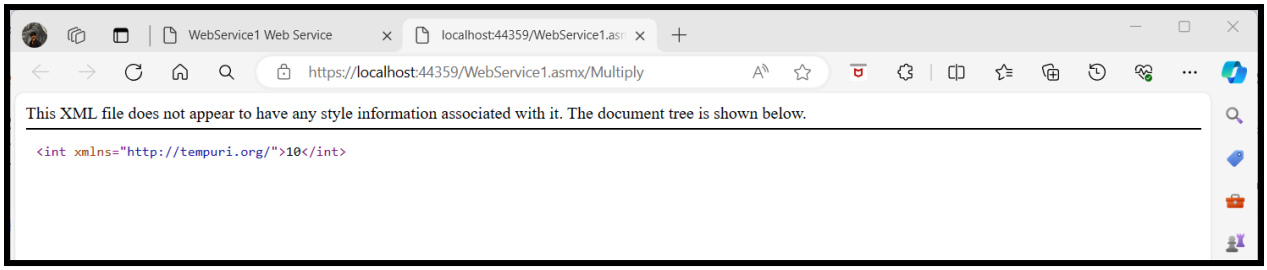
```
using System;
using System.Collections.Generic;
using System.Linq;
```

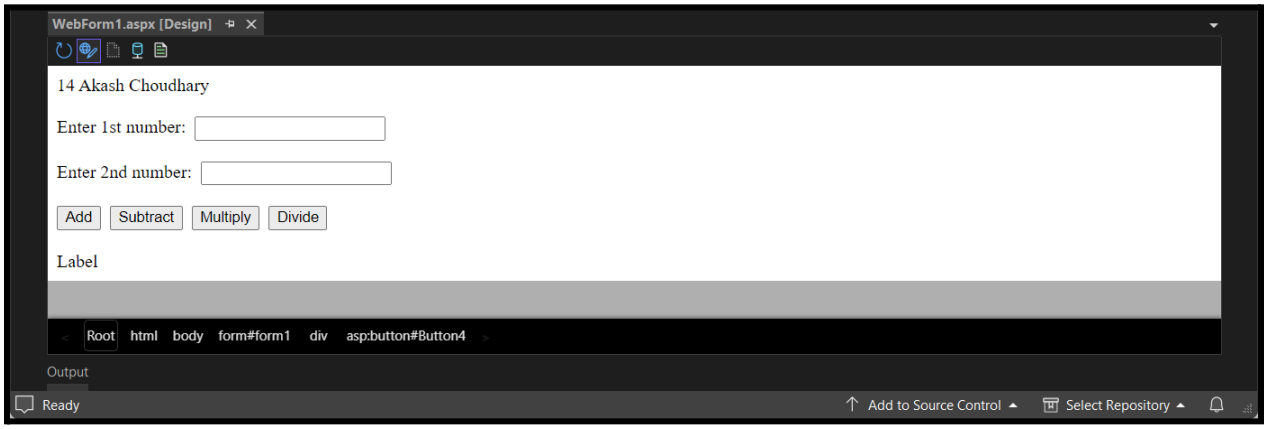
```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace WebApplication2
{
   public partial class WebForm1 : System.Web.UI.Page
   {
      localhost.WebService1 service1 = new localhost.WebService1();

      protected void Page_Load(object sender, EventArgs e)
      {

      }

      protected void Button1_Click(object sender, EventArgs e)
      {
         int result = service1.Add(int.Parse(TextBox1.Text), int.Parse(TextBox2.Text));
         Label4.Text = result.ToString();
      }

      protected void Button2_Click(object sender, EventArgs e)
      {
         int result = service1.Subtract(int.Parse(TextBox1.Text), int.Parse(TextBox2.Text));
         Label4.Text = result.ToString();
      }

      protected void Button3_Click(object sender, EventArgs e)
      {
         int result = service1.Multiply(int.Parse(TextBox1.Text), int.Parse(TextBox2.Text));
         Label4.Text = result.ToString();
      }

      protected void Button4_Click(object sender, EventArgs e)
      {
         int result = service1.Divide(int.Parse(TextBox1.Text), int.Parse(TextBox2.Text));
         Label4.Text = result.ToString();
      }
   }
}
```

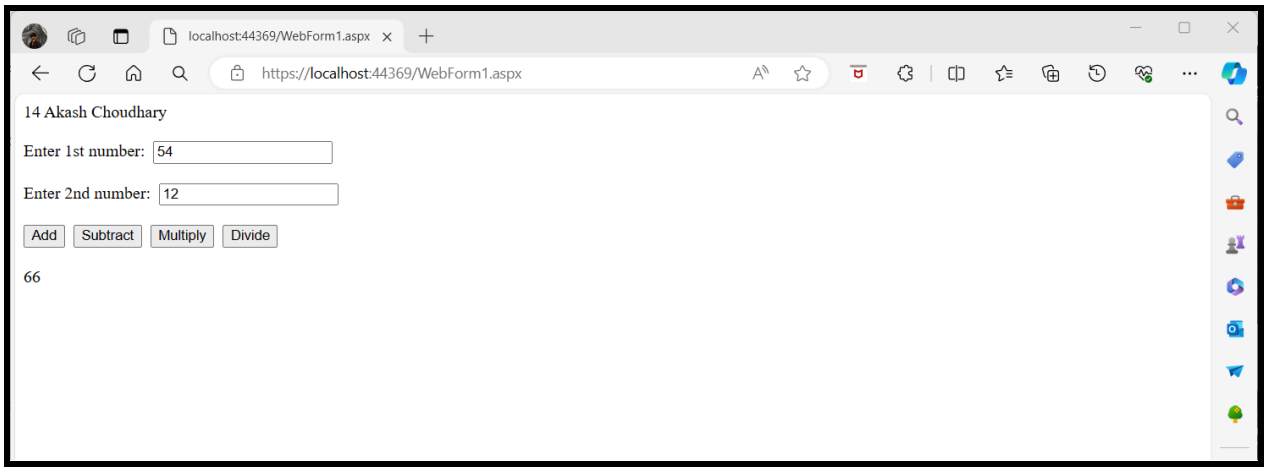
Output:

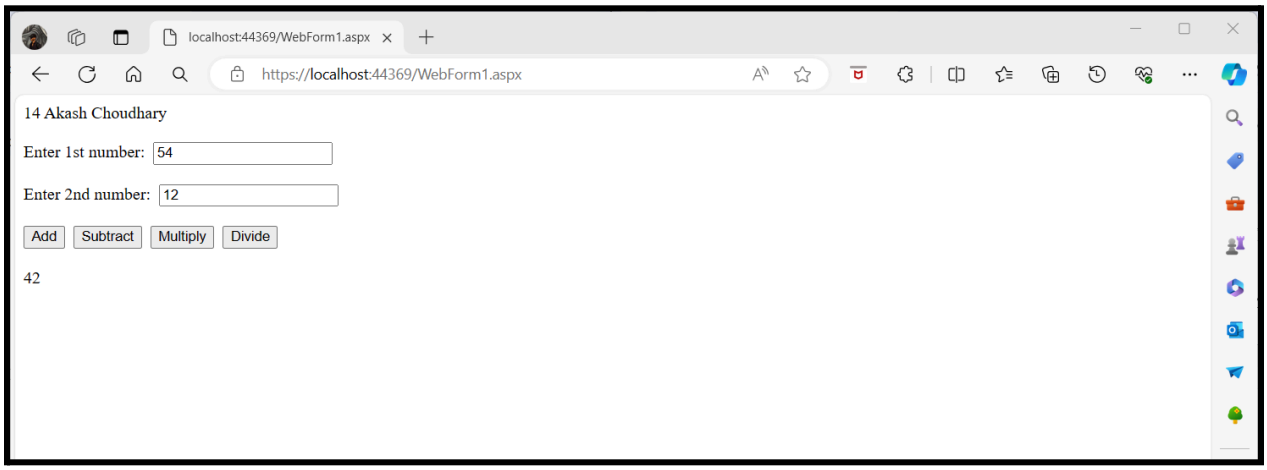

WebService1 Web Service
https://localhost:44359/WebService1.asmx
WebService1
The following operations are supported. For a formal definition, please review the Service Description.
Add
Divide
HelloWorld
Multiply
Subtract

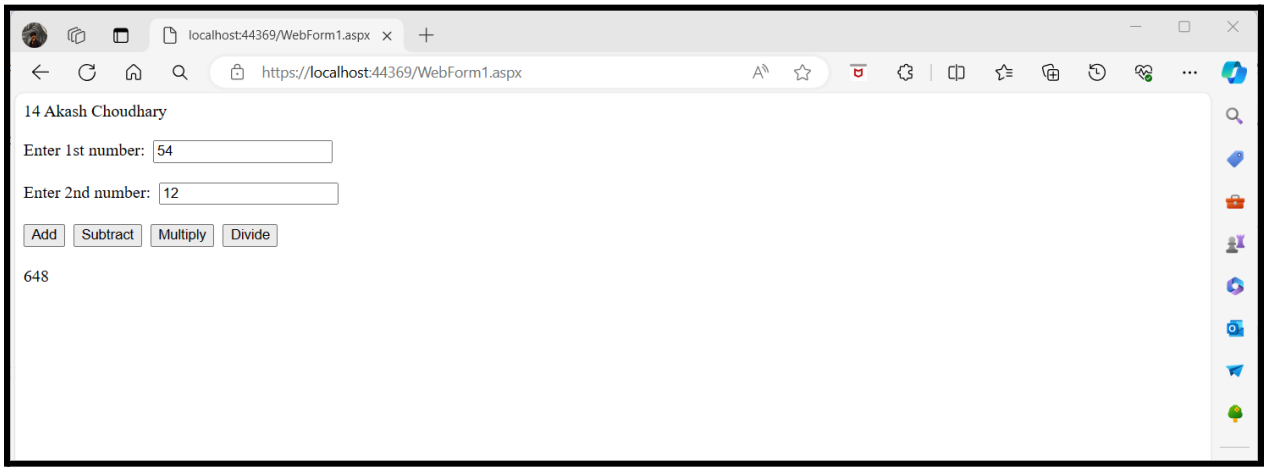

WebService1 Web Service
localhost:44359/WebService1.asr
https://localhost:44359/WebService1.asmx?op=Multiply
WebService1
Click here for a complete list of operations.
Multiply
Test
To test the operation using the HTTP POST protocol, click the 'Invoke' button.
Parameter Value
a: 5
b: 2
Invoke

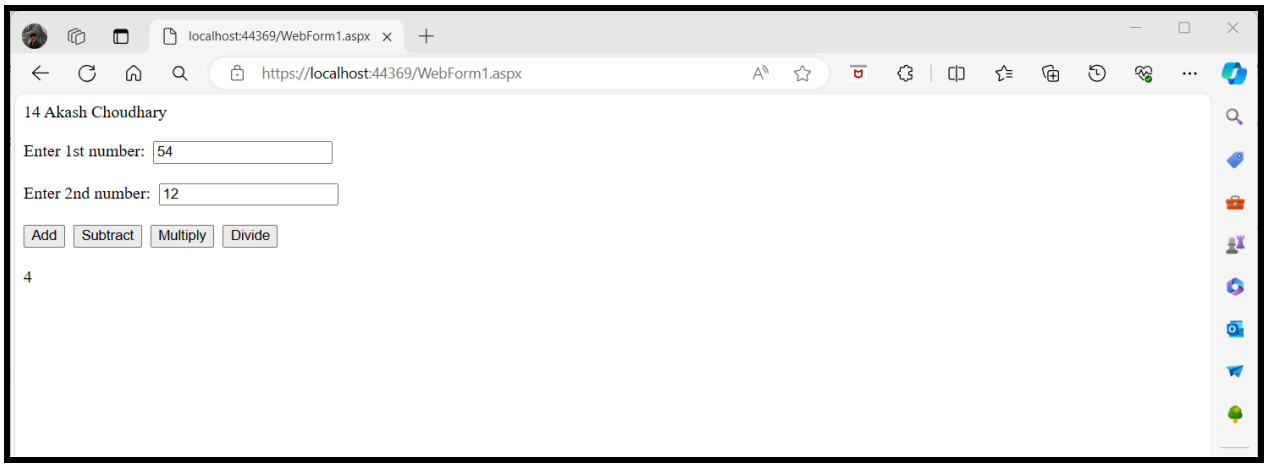

WebService1 Web Service
localhost:44359/WebService1.asr
https://localhost:44359/WebService1.asmx/Multiply
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<int xmlns="http://tempuri.org/">10</int>


WebForm1.aspx [Design]
14 Akash Choudhary
Enter 1st number:
Enter 2nd number:
Add
Subtract
Multiply
Divide
Label
Root html body form#form1 div asp:button#Button4
Output
Ready
Add to Source Control
Select Repository


localhost:44369/WebForm1.aspx
https://localhost:44369/WebForm1.aspx
14 Akash Choudhary
Enter 1st number: 54
Enter 2nd number: 12
Add
Subtract
Multiply
Divide
66


localhost:44369/WebForm1.aspx
https://localhost:44369/WebForm1.aspx
14 Akash Choudhary
Enter 1st number: 54
Enter 2nd number: 12
Add
Subtract
Multiply
Divide
42


localhost:44369/WebForm1.aspx
https://localhost:44369/WebForm1.aspx
14 Akash Choudhary
Enter 1st number: 54
Enter 2nd number: 12
Add
Subtract
Multiply
Divide
648



Conclusion: Creating a web service for arithmetic operations provides a scalable and accessible solution for performing calculations remotely, demonstrating the principles of client-server communication.

2. Create a web service demonstrating insert, select, and search functionality.

Aim: To develop a web service showcasing CRUD (Create, Read, Update, Delete) operations including insertion, selection, and searching functionality

Objectives:

1. Implement a web service using ASP.NET and C#.
2. Develop methods for inserting, selecting, and searching data.
3. Test the functionality of the web service endpoints.

Theory:

Web services are software systems designed to support interoperable machine-to-machine interaction over a network. They provide a standardized way for different applications to communicate with each other. ASP.NET is a popular framework for building web applications and services using the .NET framework. CRUD operations are fundamental to database management, allowing users to Create, Read, Update, and Delete data.

Code:

> SQL Query

```
CREATE TABLE [dbo].[EmpData]
(
        [Id] INT NOT NULL PRIMARY KEY IDENTITY,
   [Name] VARCHAR(50) NOT NULL,
   [Salary] INT NOT NULL
)
```

> WebService1.asmx.cs

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

namespace WebApplication3
{
   /// <summary>
   /// Summary description for WebService1
   /// </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 WebService1 : System.Web.Services.WebService
{
                                SqlConnection conn = new SqlConnection(@"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=EmployeeDB;Integrated Security=True;");

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

    public void Connect()
    {
        conn.Open();
    }

    [WebMethod]
    public int InsertData(string name, int salary)
    {
        Connect();
            SqlCommand cmd = new SqlCommand("INSERT INTO EmpData (Name, Salary) VALUES ('" + name + "', " + salary + ")", conn);
        return cmd.ExecuteNonQuery();

    }

    [WebMethod]
    public DataSet SelectRecord()
    {
        Connect();
            SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT * FROM EmpData", conn);
        DataSet dataSet = new DataSet();
        dataAdapter.Fill(dataSet);
        return dataSet;
    }

    [WebMethod]
    public DataSet SearchRecord(string searchData)
    {
        Connect();
```

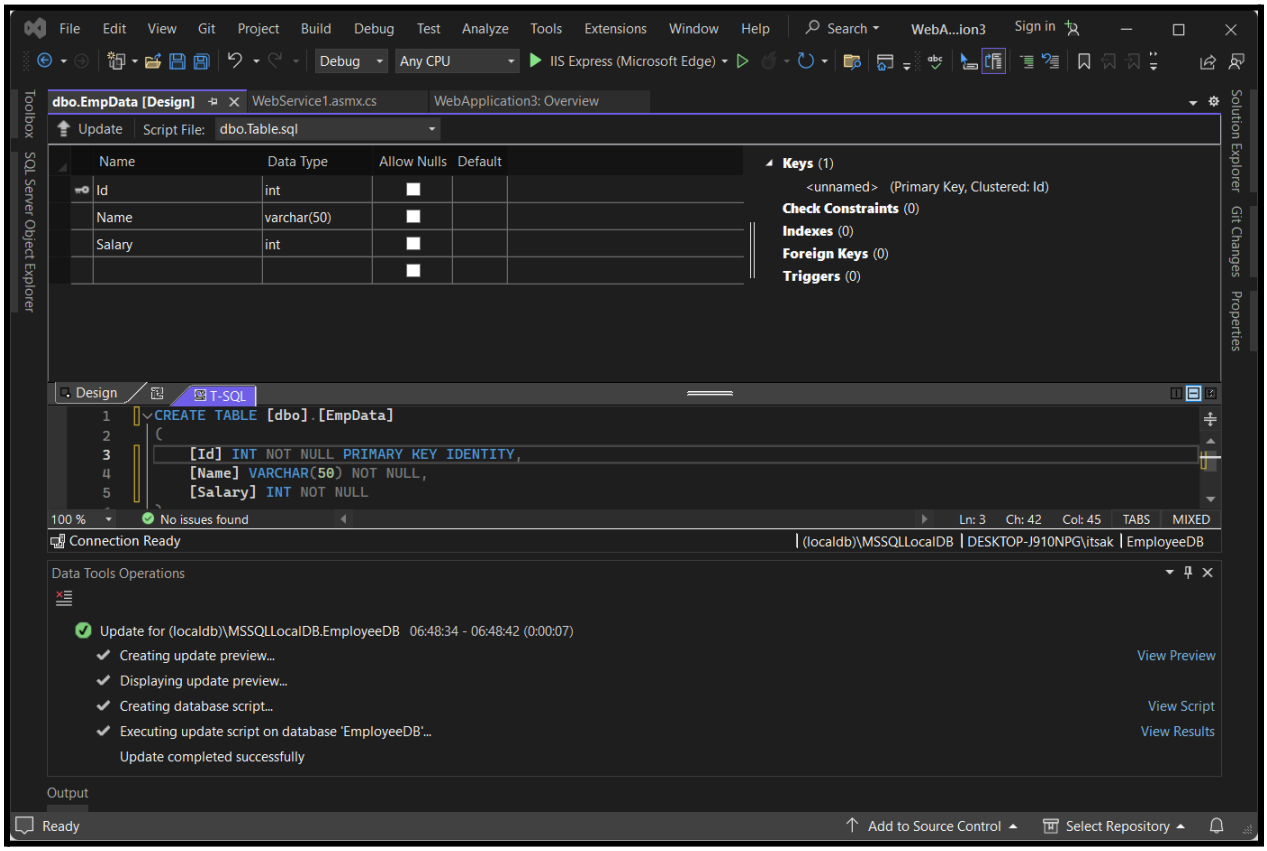
```
            SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT * FROM EmpData
WHERE Name LIKE '%" + searchData + "%'", conn);
        DataSet ds = new DataSet();
        dataAdapter.Fill(ds);
        return ds;
    }
  }
}
```

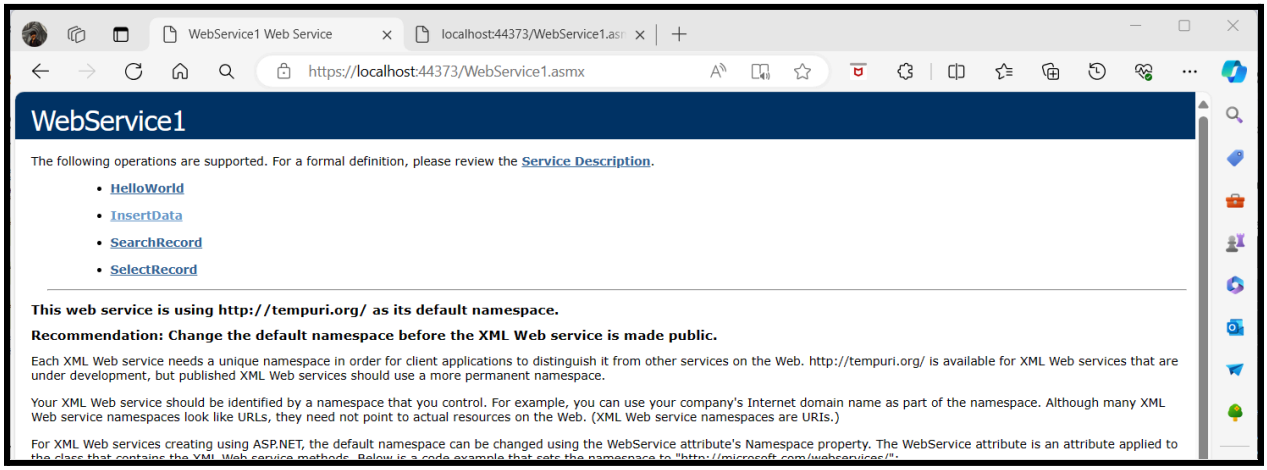
> WebForm1.aspx.cs

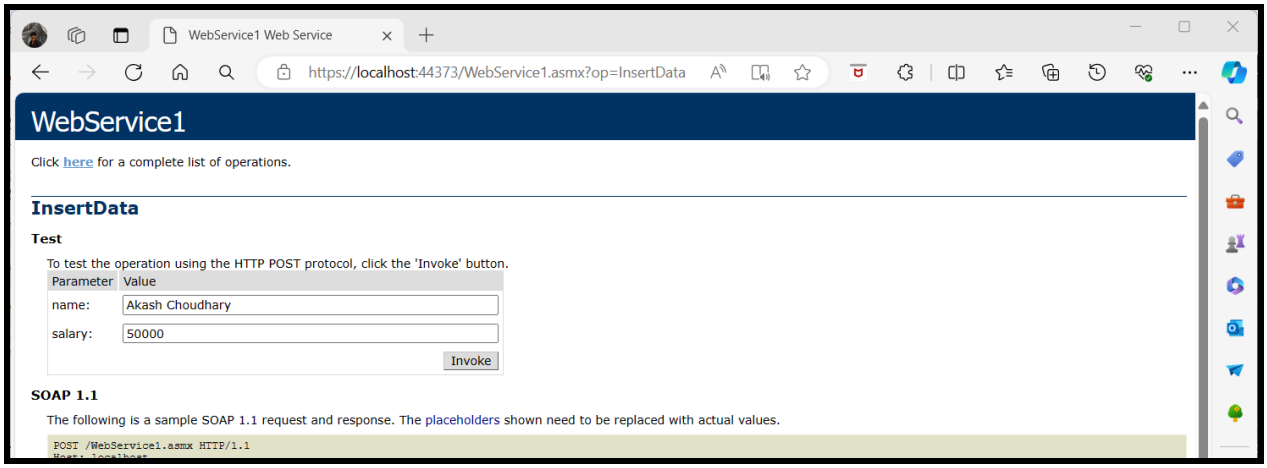
```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace WebApplication4
{
  public partial class WebForm1 : System.Web.UI.Page
  {
    localhost.WebService1 webService1 = new localhost.WebService1();
    string name;
    int salary;

    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
      name = TextBox1.Text;
      salary = int.Parse(TextBox2.Text);
      int result = webService1.InsertData(name, salary);
      DataSet dataSet = new DataSet();
      dataSet = webService1.SelectRecord();
      GridView1.DataSource = dataSet;
      GridView1.DataBind();
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
      name = TextBox1.Text;
      DataSet dataSet = new DataSet();
```

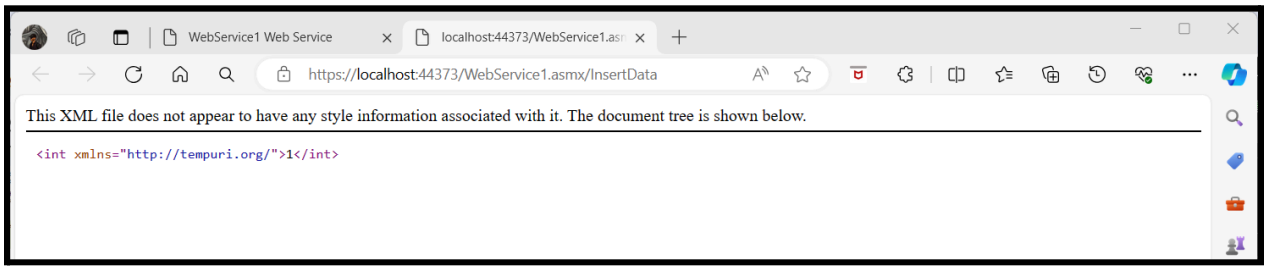
```
            dataSet = webService1.SearchRecord(name);
            GridView1.DataSource = dataSet;
            GridView1.DataBind();
        }
    }
}
```

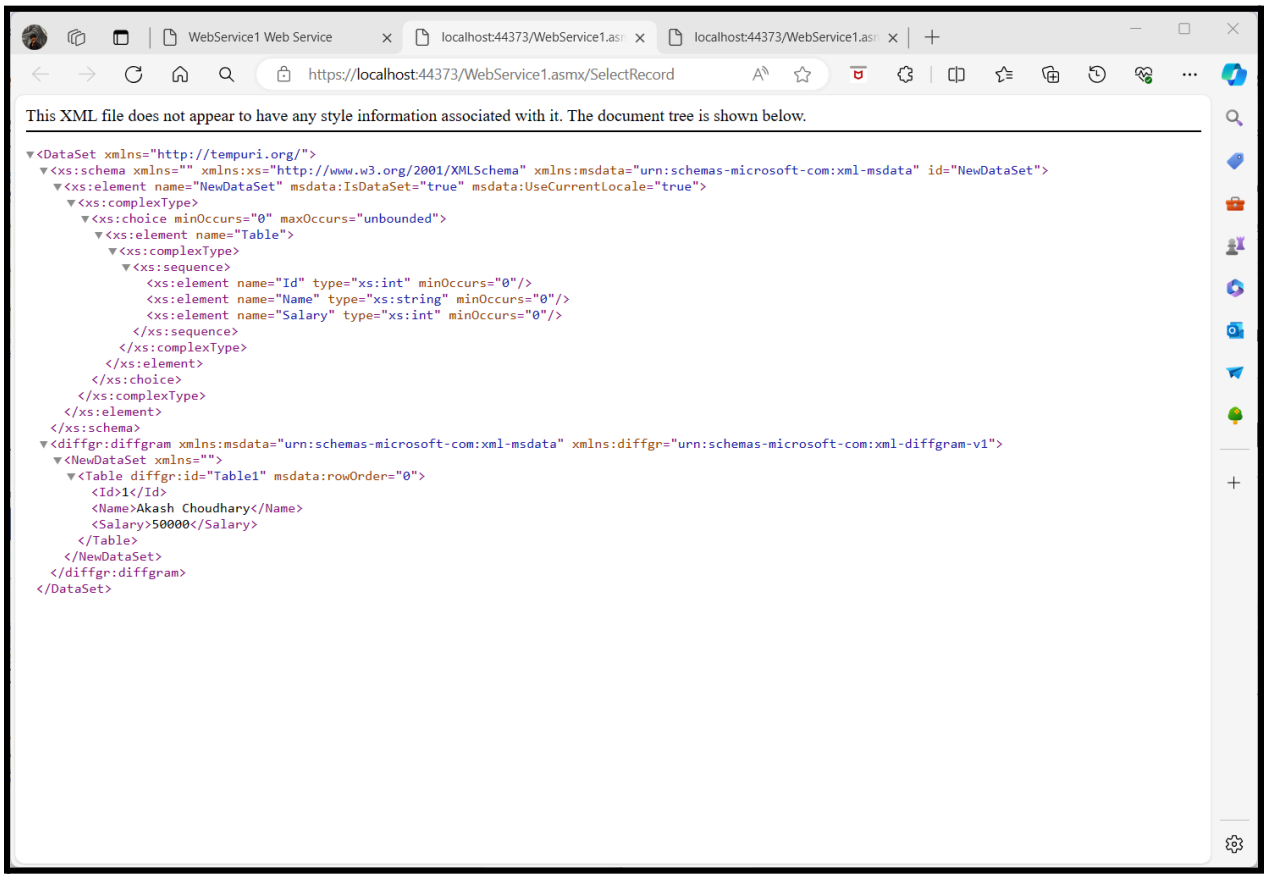
Output:

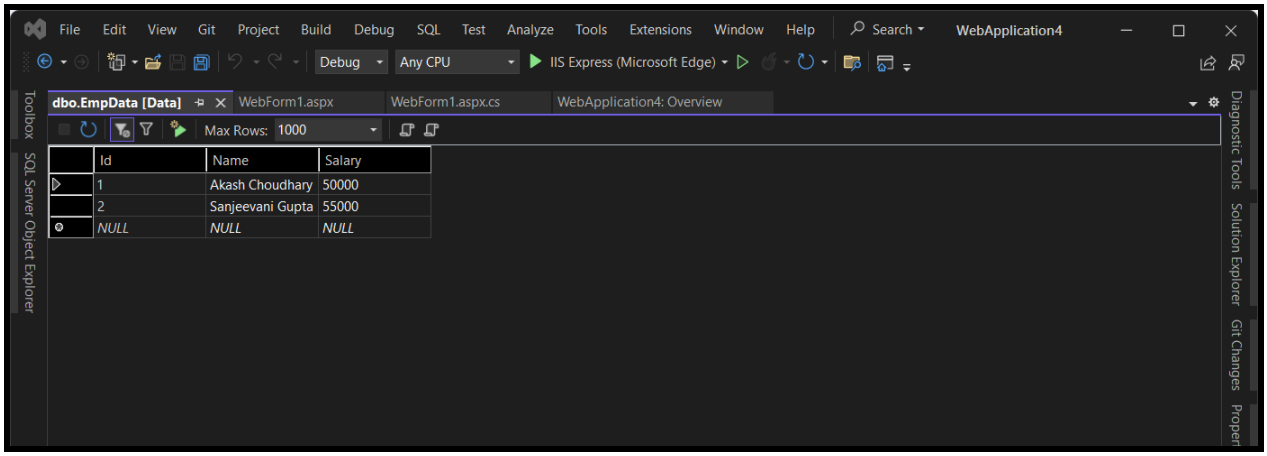


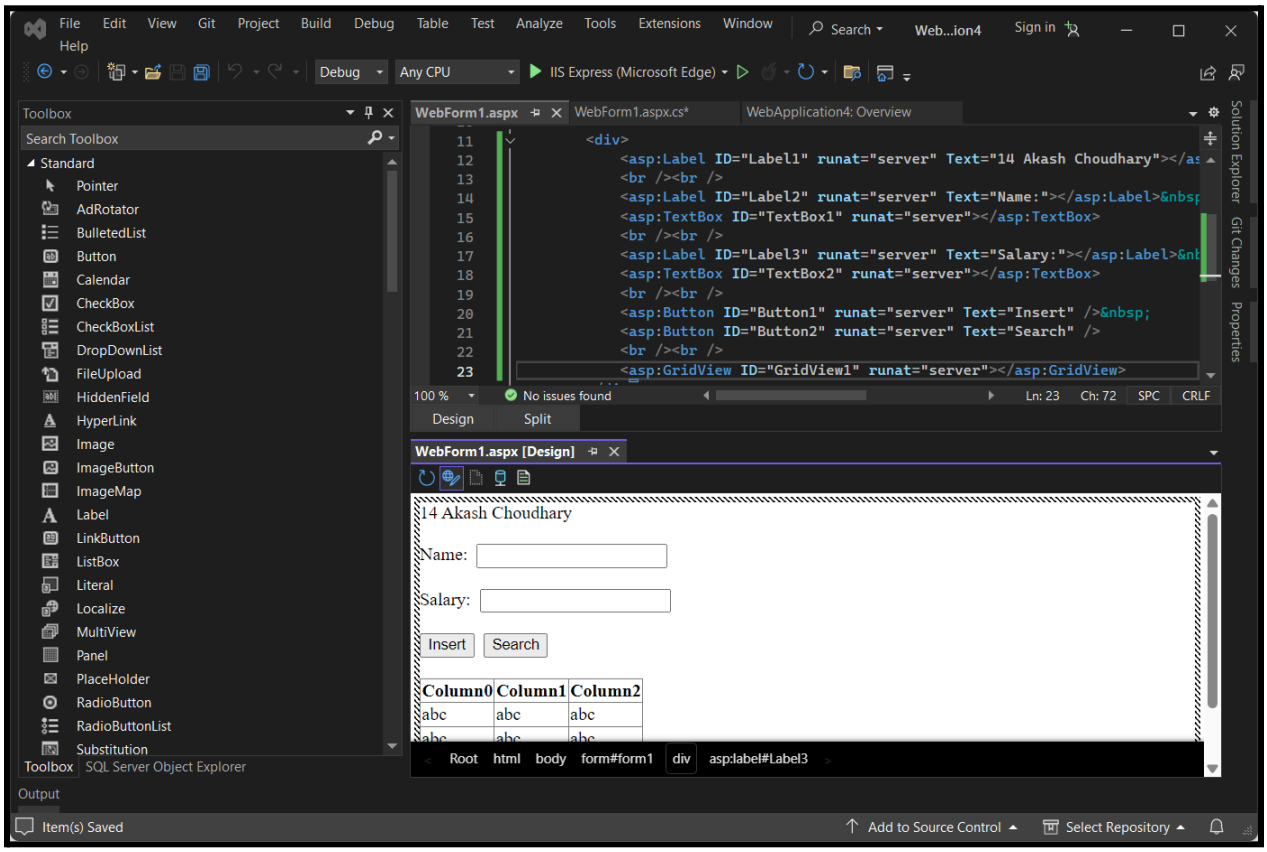



WebService1 Web Service
https://localhost:44373/WebService1.asmx?op=InsertData
WebService1
Click here for a complete list of operations.
InsertData
Test
To test the operation using the HTTP POST protocol, click the 'Invoke' button.
Parameter Value
name: Akash Choudhary
salary: 50000
Invoke
SOAP 1.1
The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.
POST /WebService1.asmx HTTP/1.1


WebService1 Web Service
localhost:44373/WebService1.asr
https://localhost:44373/WebService1.asmx/InsertData
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/">1</int>


WebService1 Web Service
localhost:44373/WebService1.asr
localhost:44373/WebService1.asr
https://localhost:44373/WebService1.asmx/SelectRecord
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<DataSet xmlns="http://tempuri.org/">
<xs:schema xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" id="NewDataSet">
<xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:UseCurrentLocale="true">
<xs:complexType>
<xs:choice minOccurs="0" maxOccurs="unbounded">
<xs:element name="Table">
<xs:complexType>
<xs:sequence>
<xs:element name="Id" type="xs:int" minOccurs="0"/>
<xs:element name="Name" type="xs:string" minOccurs="0"/>
<xs:element name="Salary" type="xs:int" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:schema>
<diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
<NewDataSet xmlns="">
<Table diffgr:id="Table1" msdata:rowOrder="0">
<Id>1</Id>
<Name>Akash Choudhary</Name>
<Salary>50000</Salary>
</Table>
</NewDataSet>
</diffgr:diffgram>
</DataSet>


File Edit View Git Project Build Debug SQL Test Analyze Tools Extensions Window Help
Search
WebApplication4
Debug
Any CPU
IIS Express (Microsoft Edge)
dbo.EmpData [Data]
WebForm1.aspx
WebForm1.aspx.cs
WebApplication4: Overview
Max Rows: 1000
Id Name Salary
1 Akash Choudhary 50000
2 Sanjeevani Gupta 55000
NULL NULL NULL
Toolbox
SQL Server Object Explorer
Diagnostic Tools
Solution Explorer
Git Changes


File Edit View Git Project Build Debug Table Test Analyze Tools Extensions Window Help
Search
Web...ion4
Sign in
Debug
Any CPU
IIS Express (Microsoft Edge)
Toolbox
Search Toolbox
Standard
Pointer
AdRotator
BulletedList
Button
Calendar
CheckBox
CheckBoxList
DropDownList
FileUpload
HiddenField
HyperLink
Image
ImageButton
ImageMap
Label
LinkButton
ListBox
Literal
Localize
MultiView
Panel
PlaceHolder
RadioButton
RadioButtonList
Substitution
WebForm1.aspx
WebForm1.aspx.cs*
WebApplication4: Overview

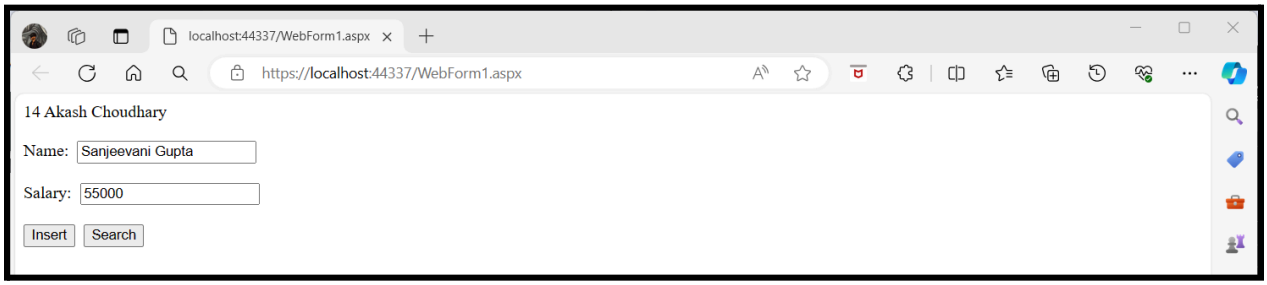
<asp:Label ID="Label1" runat="server" Text="14 Akash Choudhary"></as

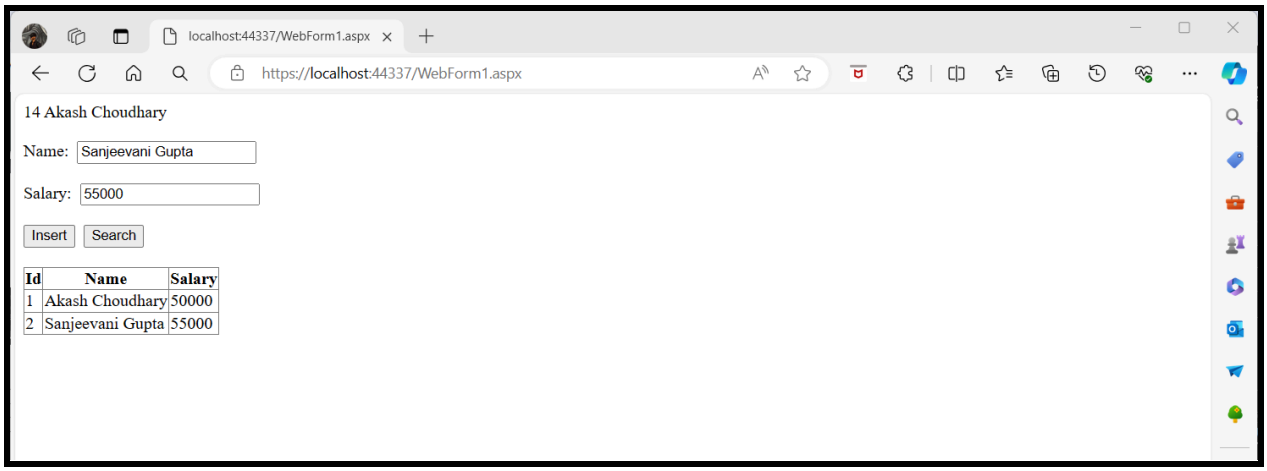
<asp:Label ID="Label2" runat="server" Text="Name:"></asp:Label>
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

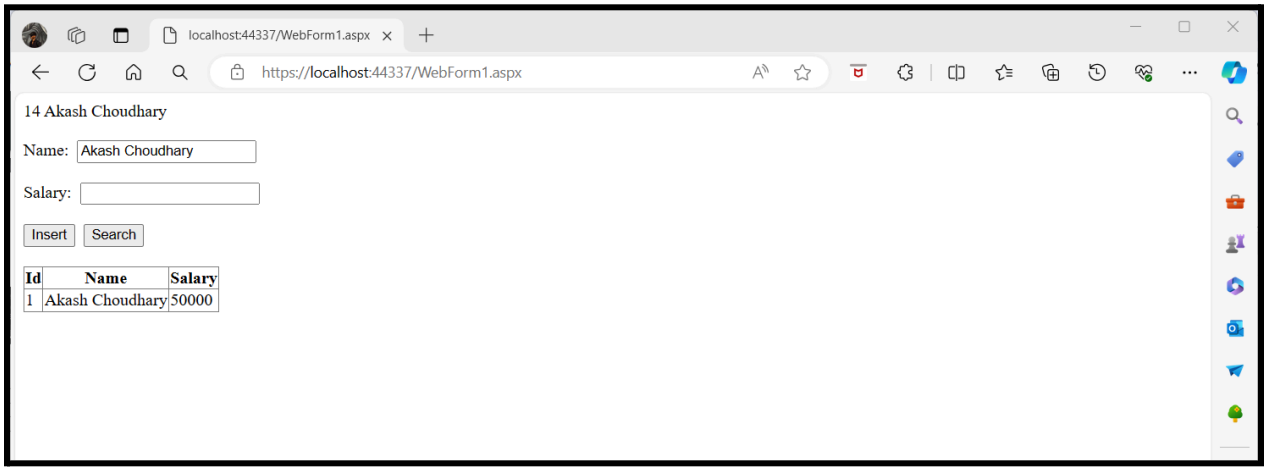
<asp:Label ID="Label3" runat="server" Text="Salary:"></asp:Label>&nb
<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>

<asp:Button ID="Button1" runat="server" Text="Insert" />
<asp:Button ID="Button2" runat="server" Text="Search" />

<asp:GridView ID="GridView1" runat="server"></asp:GridView>
100 %
No issues found
Ln: 23 Ch: 72 SPC CRLF
Design
Split
WebForm1.aspx [Design]
14 Akash Choudhary
Name:
Salary:
Insert
Search
Column0 Column1 Column2
abc abc abc
Root html body form#form1 div asp:label#Label3
Toolbox
SQL Server Object Explorer
Output
Item(s) Saved
Add to Source Control
Select Repository


localhost:44337/WebForm1.aspx
https://localhost:44337/WebForm1.aspx
14 Akash Choudhary
Name: Sanjeevani Gupta
Salary: 55000
Insert
Search





Conclusion: The practical demonstrates the versatility and utility of web services in facilitating data management operations over the web.

3. Write a program to create a WCF Service that adds two numbers, and also create a client to call that service.

Aim: To create a WCF Service for adding two numbers and develop a client to consume the service

Objectives:

1. Implement a WCF Service contract to define the operation for adding two numbers.
2. Host the WCF Service in a suitable environment.
3. Develop a client application to consume the WCF Service.
4. Test the functionality of the service by invoking it from the client.

Theory:

Windows Communication Foundation (WCF) is a framework for building service-oriented applications. It allows developers to create secure, reliable, and interoperable services. WCF supports various communication protocols and messaging patterns. ASP.NET MVC is a web application framework that follows the Model-View-Controller architectural pattern. It provides a robust framework for building dynamic web applications.

Code:

> IService.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the interface name "IService" in both code and config file together.
[ServiceContract]
public interface IService
{
        // TODO: Add your service operations here
    [OperationContract]
    int Add(int a, int b);

    [OperationContract]
    int Subtract(int a, int b);
```

```
    [OperationContract]
    int Multiply(int a, int b);

    [OperationContract]
    int Divide(int a, int b);
}
```

> Service.cs

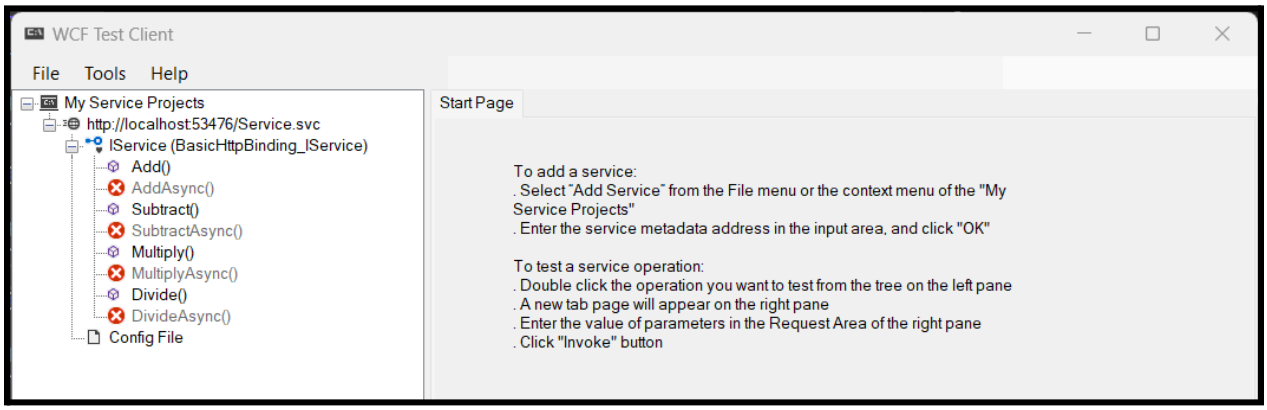
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the class name
"Service" in code, svc and config file together.
public class Service : IService
{
    public int Add(int a, int b)
    {
        return (a + b);
    }

    public int Subtract(int a, int b)
    {
        return (a - b);
    }

    public int Multiply(int a, int b)
    {
        return (a * b);
    }

    public int Divide(int a, int b)
    {
        return (a / b);
    }
}
```

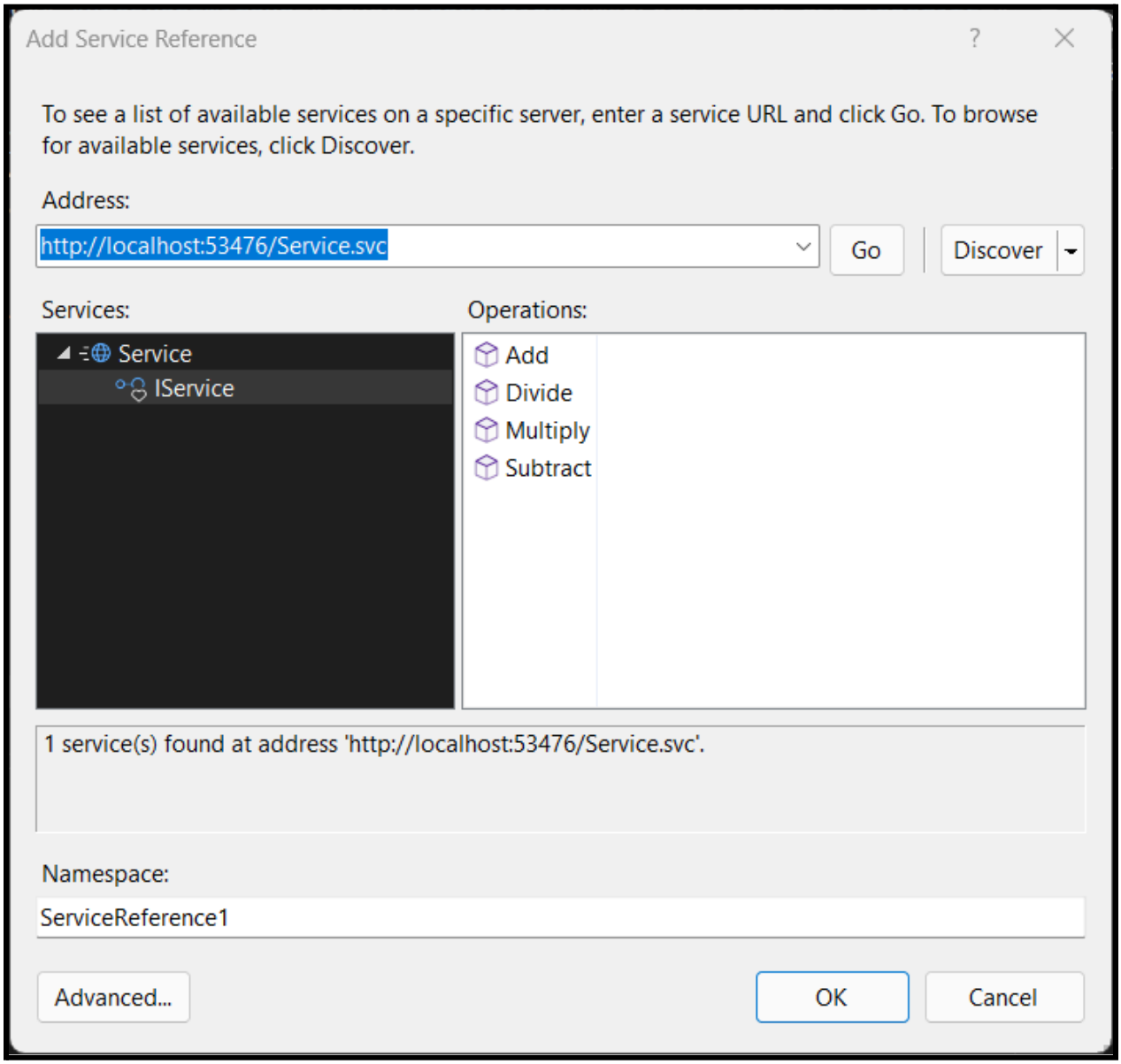
> WebForm1.aspx.cs

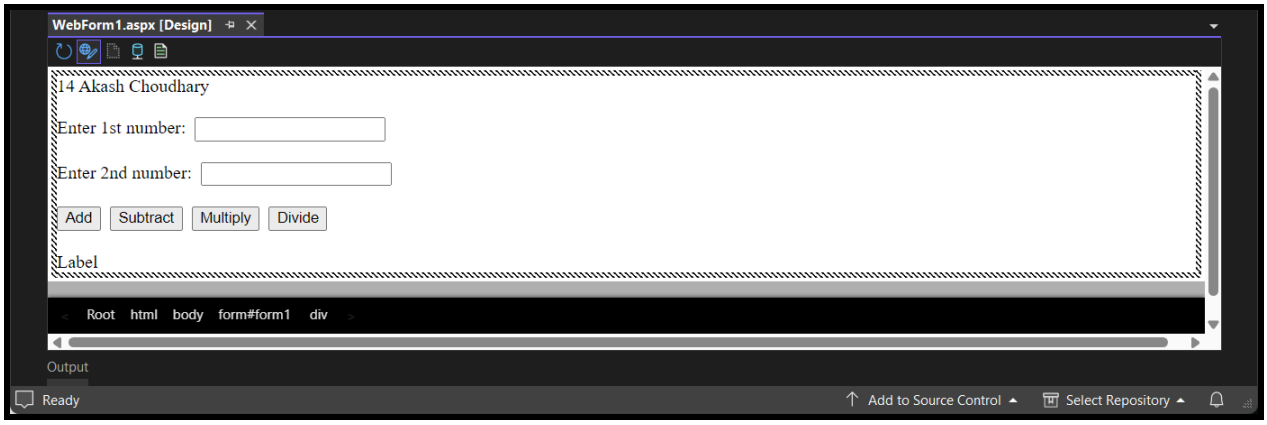
```
using System;
using System.Collections.Generic;
using System.Linq;
```

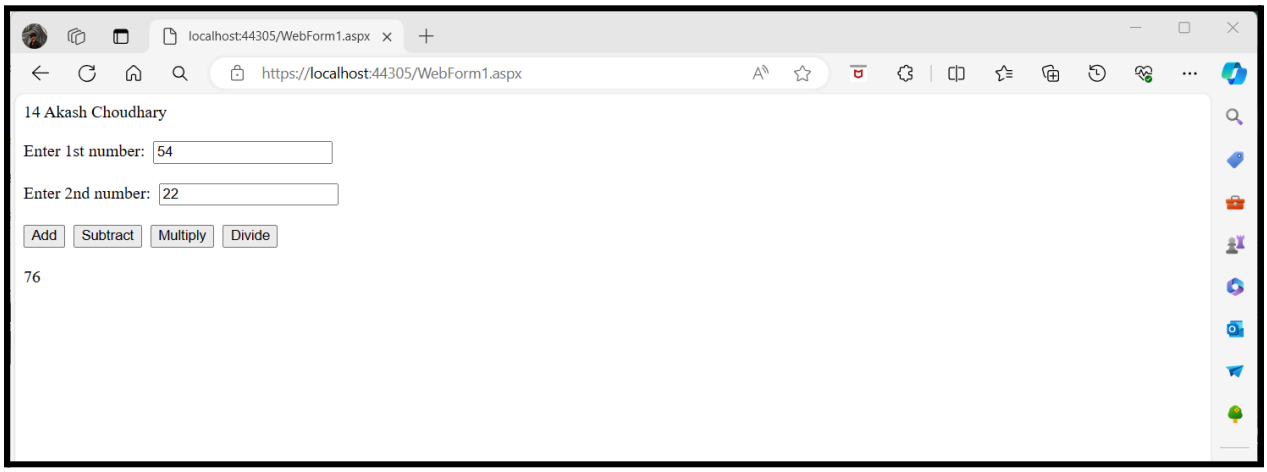
```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace WebApplication5
{
   public partial class WebForm1 : System.Web.UI.Page
   {
      ServiceReference1.ServiceClient serviceClient = new ServiceReference1.ServiceClient();

      protected void Page_Load(object sender, EventArgs e)
      {

      }

      protected void Button1_Click(object sender, EventArgs e)
      {
         int result = serviceClient.Add(int.Parse(TextBox1.Text), int.Parse(TextBox2.Text));
         Label4.Text = result.ToString();
      }

      protected void Button2_Click(object sender, EventArgs e)
      {
         int result = serviceClient.Subtract(int.Parse(TextBox1.Text), int.Parse(TextBox2.Text));
         Label4.Text = result.ToString();
      }

      protected void Button3_Click(object sender, EventArgs e)
      {
         int result = serviceClient.Multiply(int.Parse(TextBox1.Text), int.Parse(TextBox2.Text));
         Label4.Text = result.ToString();
      }

      protected void Button4_Click(object sender, EventArgs e)
      {
         int result = serviceClient.Divide(int.Parse(TextBox1.Text), int.Parse(TextBox2.Text));
         Label4.Text = result.ToString();
      }
   }
}
```

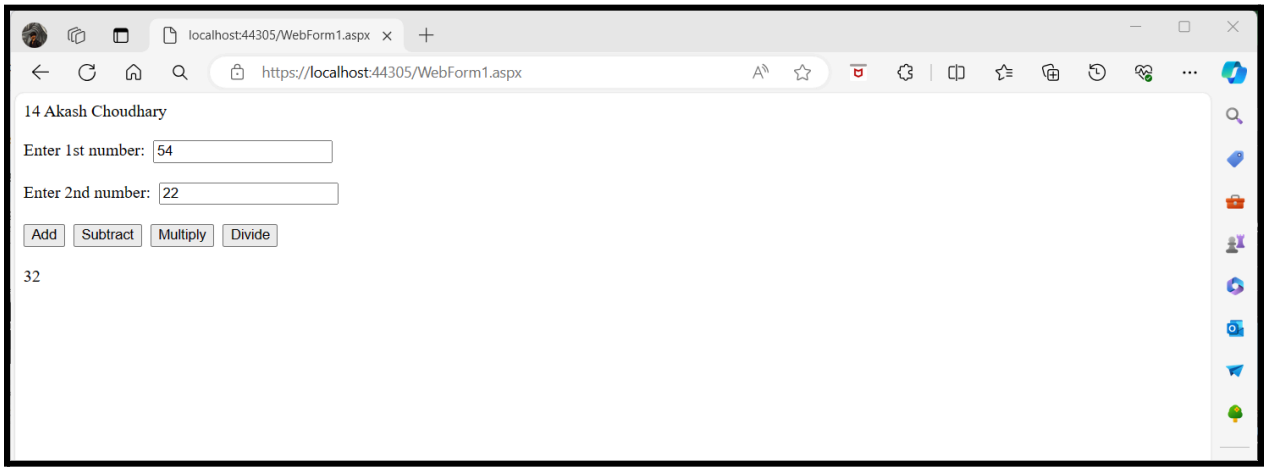
Output:

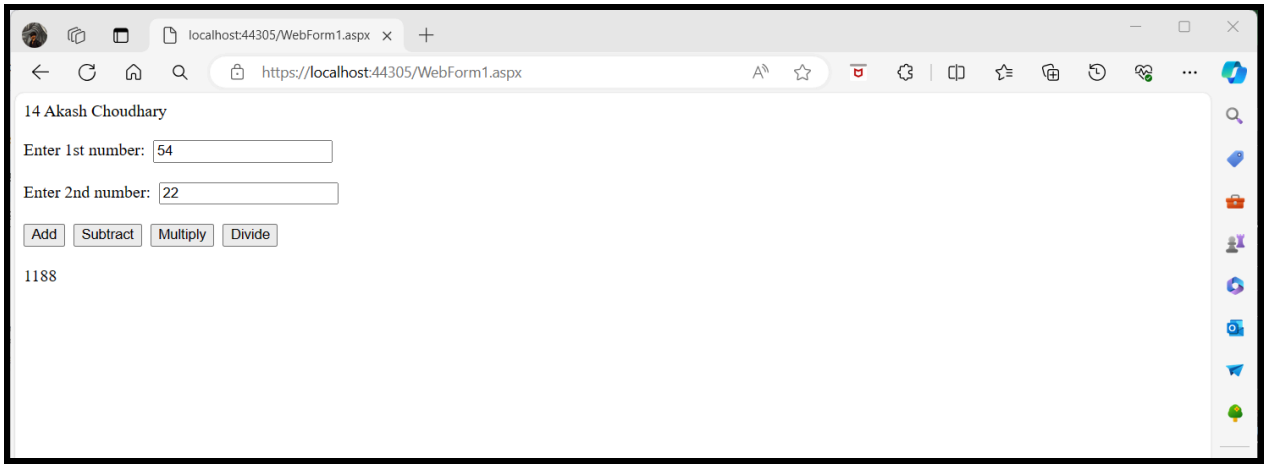

WCF Test Client
File
Tools
Help
My Service Projects
http://localhost:53476/Service.svc
IService (BasicHttpBinding_IService)
Add()
AddAsync()
Subtract()
SubtractAsync()
Multiply()
MultiplyAsync()
Divide()
DivideAsync()
Config File
Start Page
To add a service:
. Select "Add Service" from the File menu or the context menu of the "My Service Projects"
. Enter the service metadata address in the input area, and click "OK"
To test a service operation:
. Double click the operation you want to test from the tree on the left pane
. A new tab page will appear on the right pane
. Enter the value of parameters in the Request Area of the right pane
. Click "Invoke" button

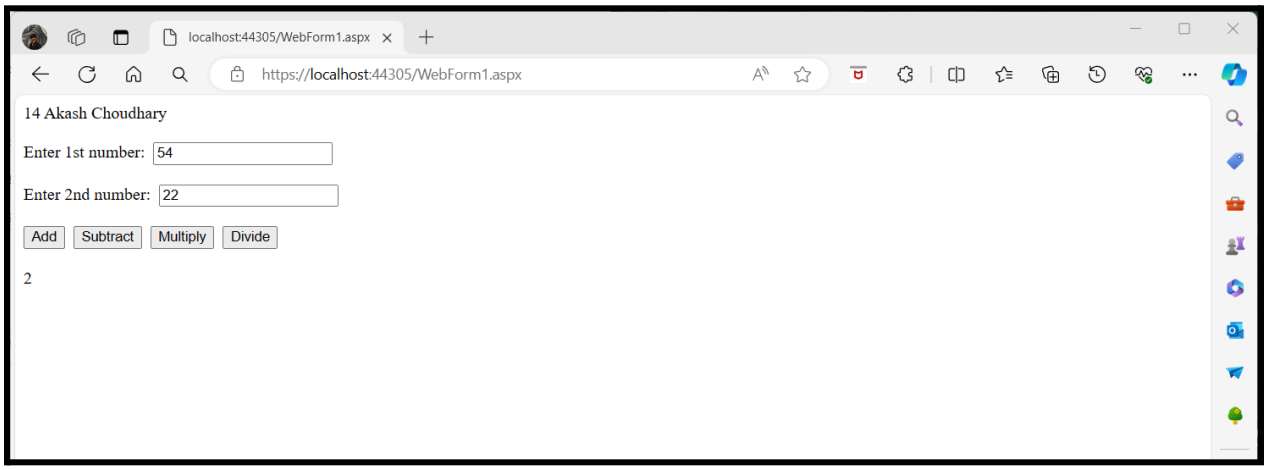

Add Service Reference
To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.
Address:
http://localhost:53476/Service.svc
Go
Discover
Services:
Service
IService
Operations:
Add
Divide
Multiply
Subtract
1 service(s) found at address 'http://localhost:53476/Service.svc'.
Namespace:
ServiceReference1
Advanced...
OK
Cancel


WebForm1.aspx [Design]
14 Akash Choudhary
Enter 1st number:
Enter 2nd number:
Add
Subtract
Multiply
Divide
Label
Root
html
body
form#form1
div
Output
Ready
Add to Source Control
Select Repository


localhost:44305/WebForm1.aspx
https://localhost:44305/WebForm1.aspx
14 Akash Choudhary
Enter 1st number: 54
Enter 2nd number: 22
Add
Subtract
Multiply
Divide
76


localhost:44305/WebForm1.aspx
https://localhost:44305/WebForm1.aspx
14 Akash Choudhary
Enter 1st number: 54
Enter 2nd number: 22
Add
Subtract
Multiply
Divide
32





Conclusion: By implementing a WCF Service and developing a client to consume it, we demonstrate the interoperability and communication capabilities provided by WCF, enhancing the functionality of web applications.

4. Create a WCF service to display student information, and also create a client that can access the same information.

Aim: To develop a WCF service for displaying student information and create a client application to access this information

Objectives:

1. Design and implement a WCF service contract for student information.
2. Develop methods within the WCF service to retrieve student data.
3. Create a client application capable of consuming the WCF service.
4. Implement functionality in the client application to display student information obtained from the service.

Theory:

WCF (Windows Communication Foundation) is a framework for building service-oriented applications. It enables you to create services that expose functionality over various transport protocols like HTTP, TCP, and named pipes. In this scenario, we'll use WCF to create a service that provides access to student information. The client application will then consume this service, allowing users to retrieve and view student data.

Code:

> SQL Query

```
CREATE TABLE [dbo].[StudentData]
(
        [Id] INT NOT NULL PRIMARY KEY IDENTITY,
   [Name] VARCHAR(50) NOT NULL,
   [Percentage] INT NOT NULL
)
```

> IService.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;
```

```
// NOTE: You can use the "Rename" command on the "Refactor" menu to change the interface name "IService" in both code and config file together.
[ServiceContract]
public interface IService
{
    // TODO: Add your service operations here
    [OperationContract]
    void Connect();

    [OperationContract]
    int InsertData(string name, int percentage);

    [OperationContract]
    DataSet SearchRecord(string searchData);

    [OperationContract]
    DataSet SelectRecord();
}
```

> Service.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the class name "Service" in code, svc and config file together.
public class Service : IService
{
     SqlConnection conn = new SqlConnection(@"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=StudentDB;Integrated Security=True;");

    public void Connect()
    {
        conn.Open();
    }

    public int InsertData(string name, int percentage)
    {
        Connect();
```

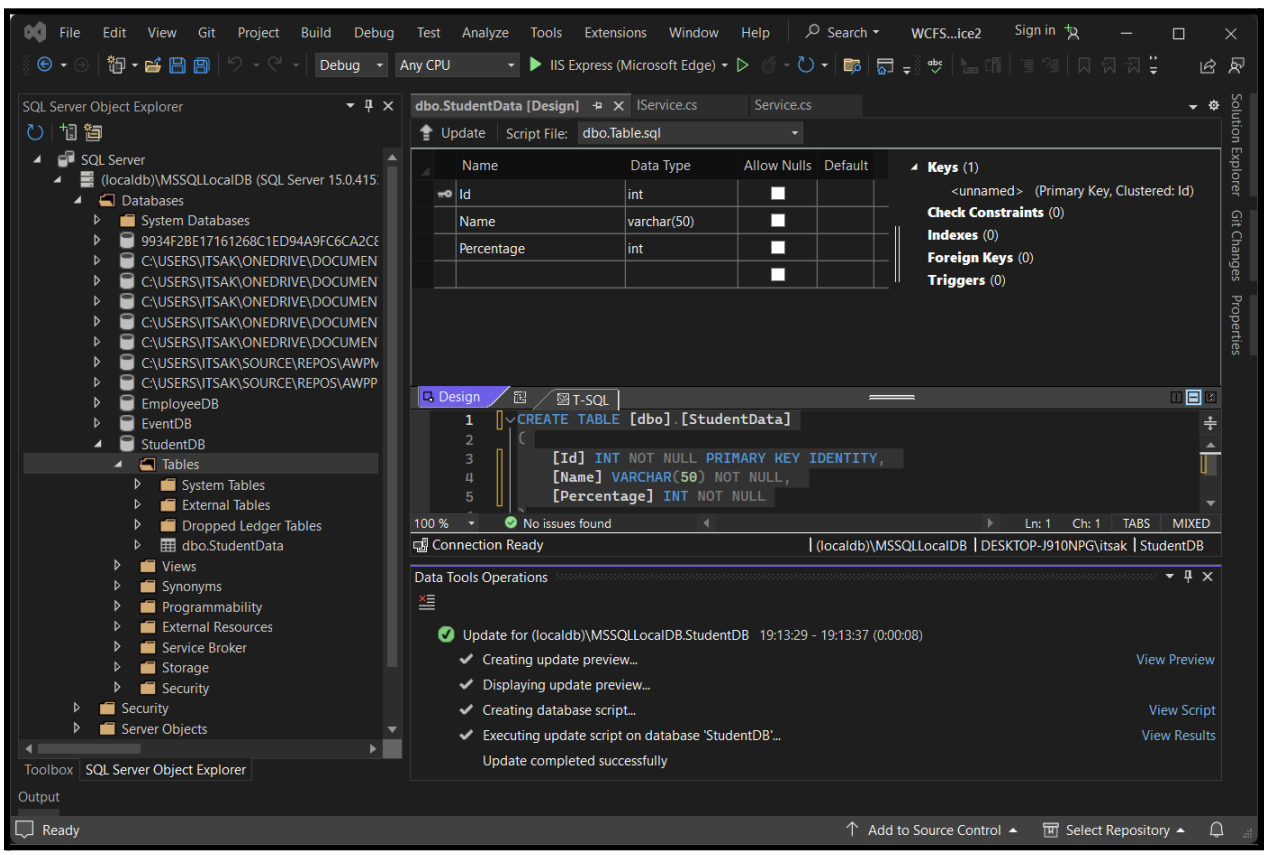
```
        SqlCommand cmd = new SqlCommand("INSERT INTO StudentData (Name, Percentage) VALUES ('" + name + "', " + percentage + ")", conn);
        return cmd.ExecuteNonQuery();
    }

    public DataSet SearchRecord(string searchData)
    {
        Connect();
        SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM StudentData WHERE Name LIKE '%" + searchData + "%'", conn);
        DataSet dataSet = new DataSet();
        adapter.Fill(dataSet);
        return dataSet;
    }

    public DataSet SelectRecord()
    {
        Connect();
        SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM StudentData", conn);
        DataSet dataSet = new DataSet();
        adapter.Fill(dataSet);
        return dataSet;
    }
}
```

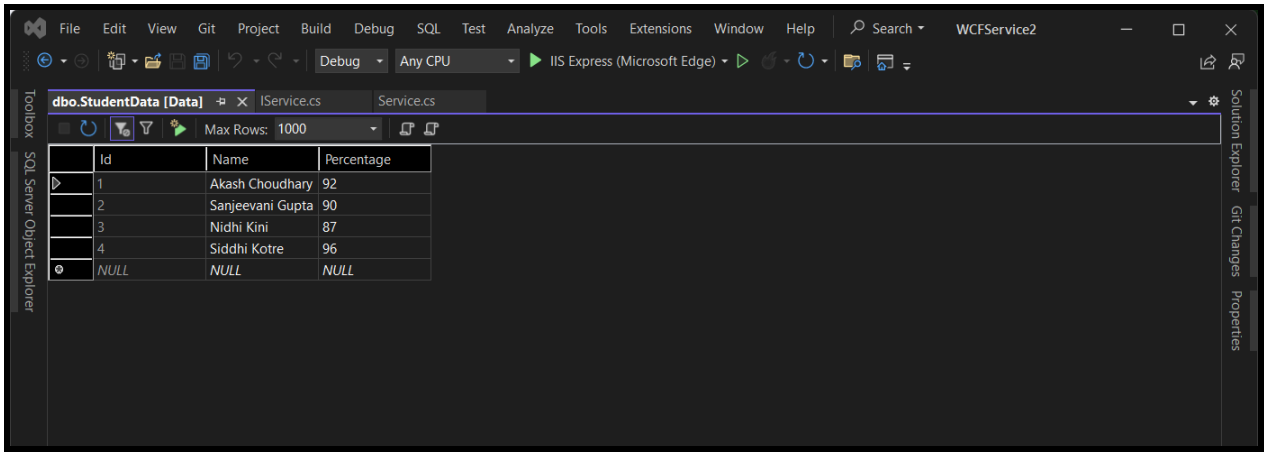
> WebForm1.aspx.cs

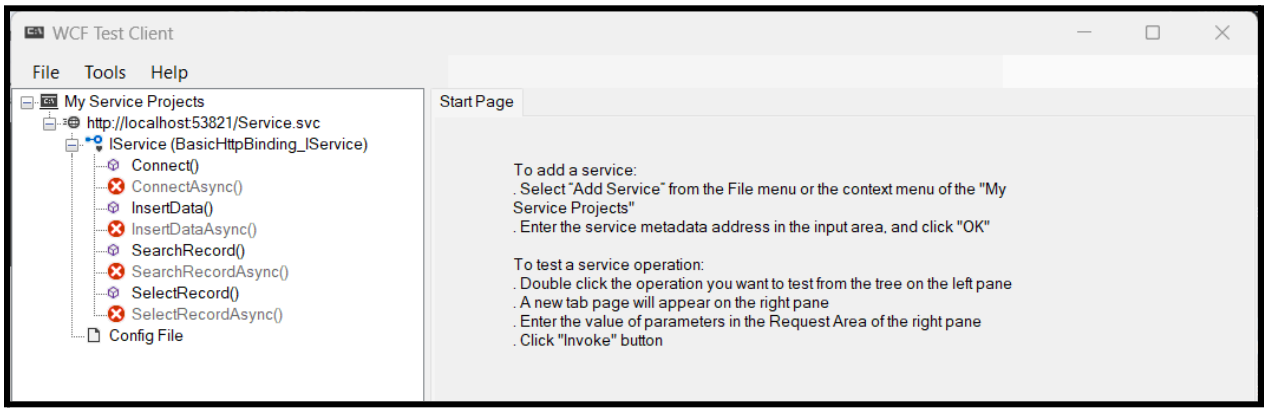
```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace WebApplication6
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        ServiceReference1.ServiceClient service = new ServiceReference1.ServiceClient();

        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
```

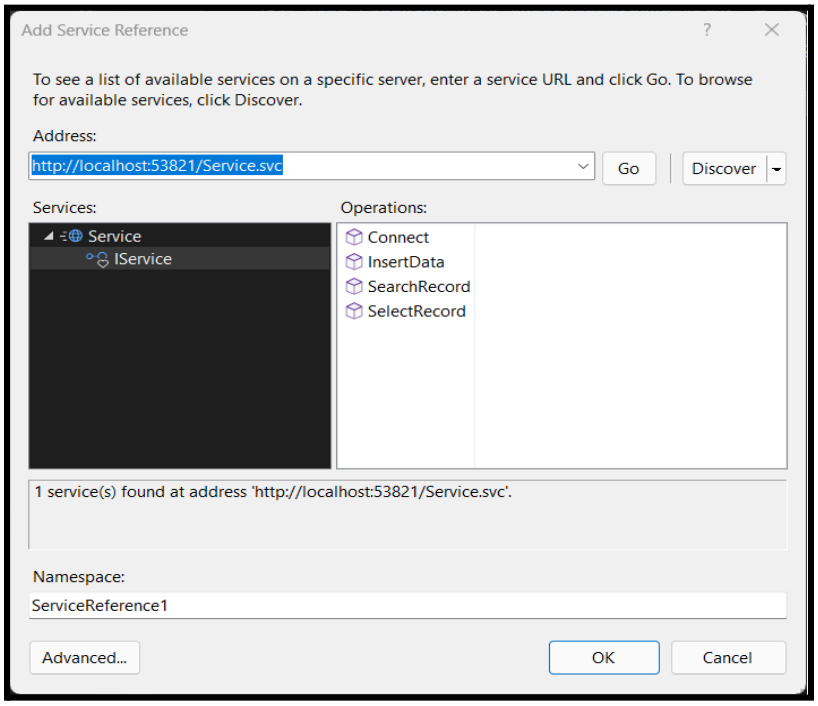
```
        {
            int result = service.InsertData(TextBox1.Text, int.Parse(TextBox2.Text));
            DataSet dataSet = new DataSet();
            dataSet = service.SelectRecord();
            GridView1.DataSource = dataSet;
            GridView1.DataBind();
        }

        protected void Button2_Click(object sender, EventArgs e)
        {
            DataSet dataSet = new DataSet();
            dataSet = service.SearchRecord(TextBox1.Text);
            GridView1.DataSource = dataSet;
            GridView1.DataBind();
        }
    }
}
```

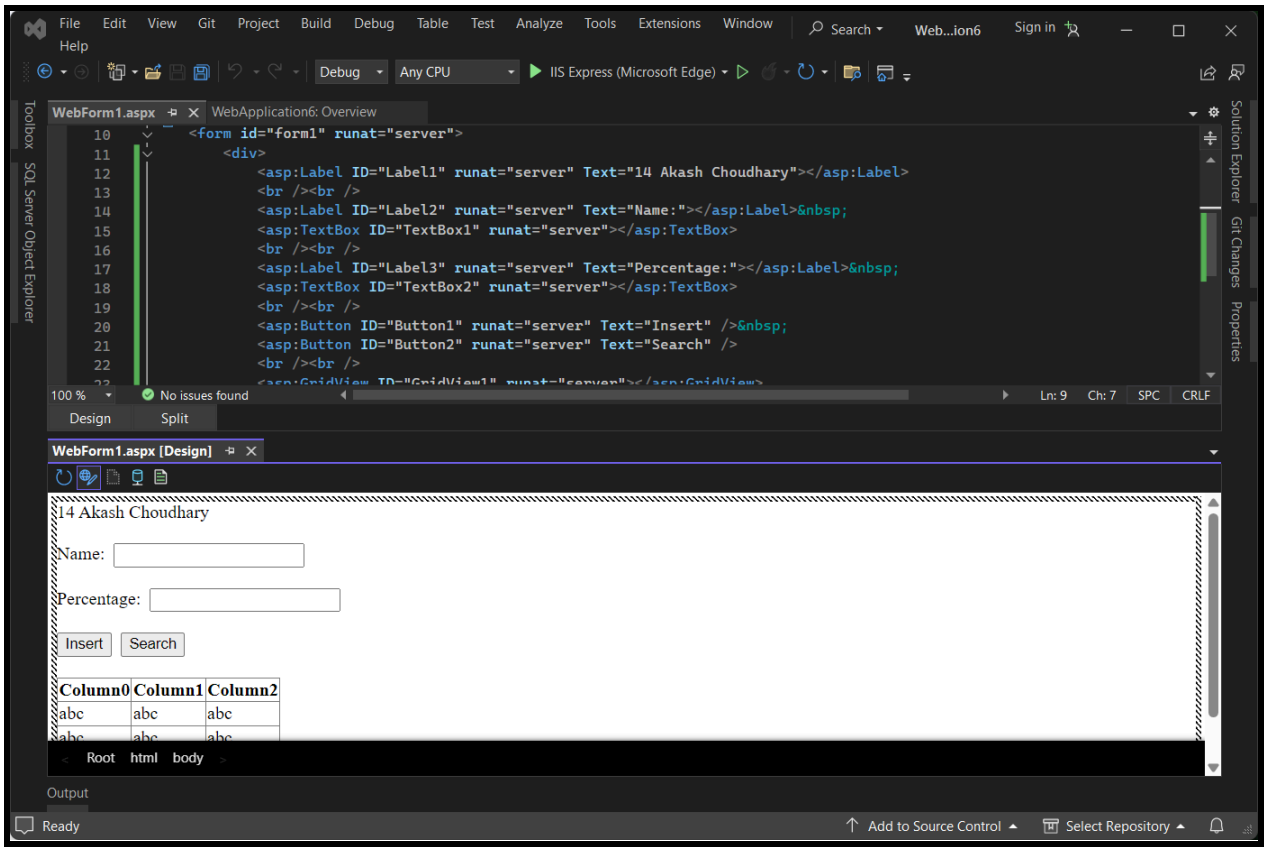
Output:




File Edit View Git Project Build Debug SQL Test Analyze Tools Extensions Window Help
Search
WCFService2
Debug
Any CPU
IIS Express (Microsoft Edge)
dbo.StudentData [Data]
IService.cs
Service.cs
Max Rows: 1000
Id Name Percentage
1 Akash Choudhary 92
2 Sanjeevani Gupta 90
3 Nidhi Kini 87
4 Siddhi Kotre 96
NULL NULL NULL
Toolbox
SQL Server Object Explorer
Solution Explorer
Git Changes
Properties


WCF Test Client
File Tools Help
My Service Projects
http://localhost:53821/Service.svc
IService (BasicHttpBinding_IService)
Connect()
ConnectAsync()
InsertData()
InsertDataAsync()
SearchRecord()
SearchRecordAsync()
SelectRecord()
SelectRecordAsync()
Config File
Start Page
To add a service:
. Select "Add Service" from the File menu or the context menu of the "My Service Projects"
. Enter the service metadata address in the input area, and click "OK"
To test a service operation:
. Double click the operation you want to test from the tree on the left pane
. A new tab page will appear on the right pane
. Enter the value of parameters in the Request Area of the right pane
. Click "Invoke" button


Add Service Reference
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.
Address:
http://localhost:53821/Service.svc
Go
Discover
Services:
Service
IService
Operations:
Connect
InsertData
SearchRecord
SelectRecord
1 service(s) found at address 'http://localhost:53821/Service.svc'.
Namespace:
ServiceReference1
Advanced...
OK
Cancel


WebForm1.aspx
WebApplication6: Overview
<form id="form1" runat="server">

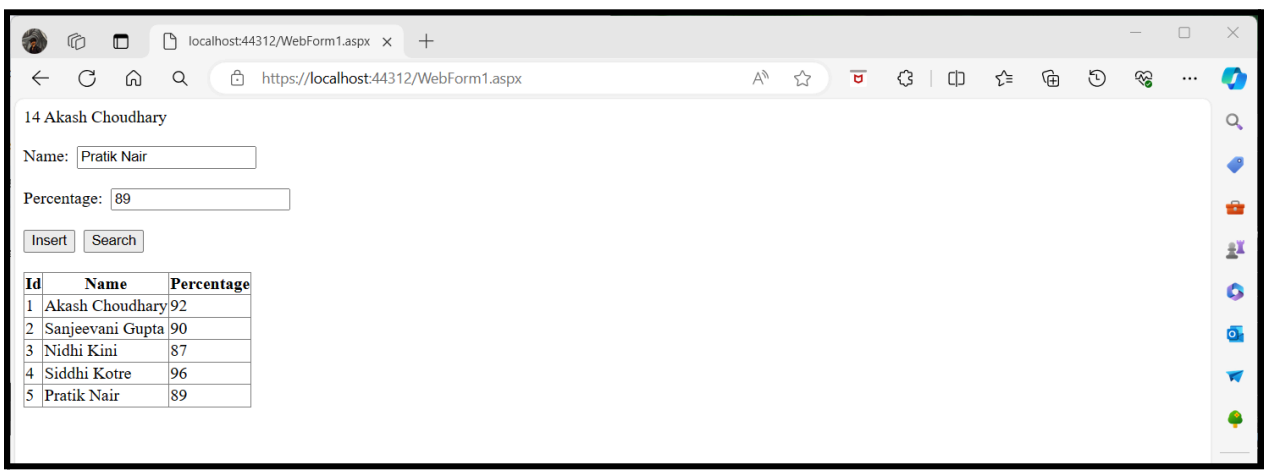
<asp:Label ID="Label1" runat="server" Text="14 Akash Choudhary"></asp:Label>

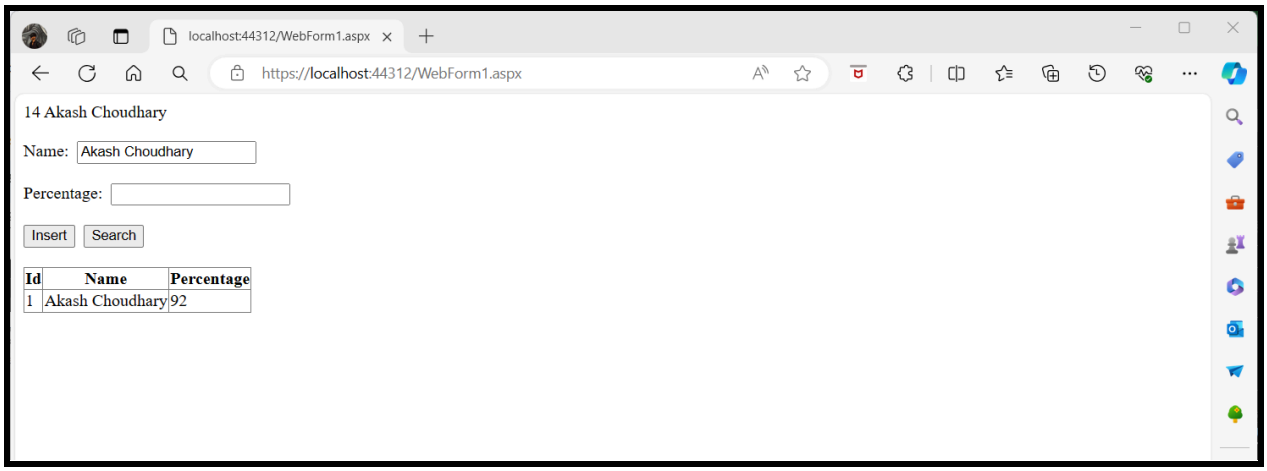
<asp:Label ID="Label2" runat="server" Text="Name:"></asp:Label>
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

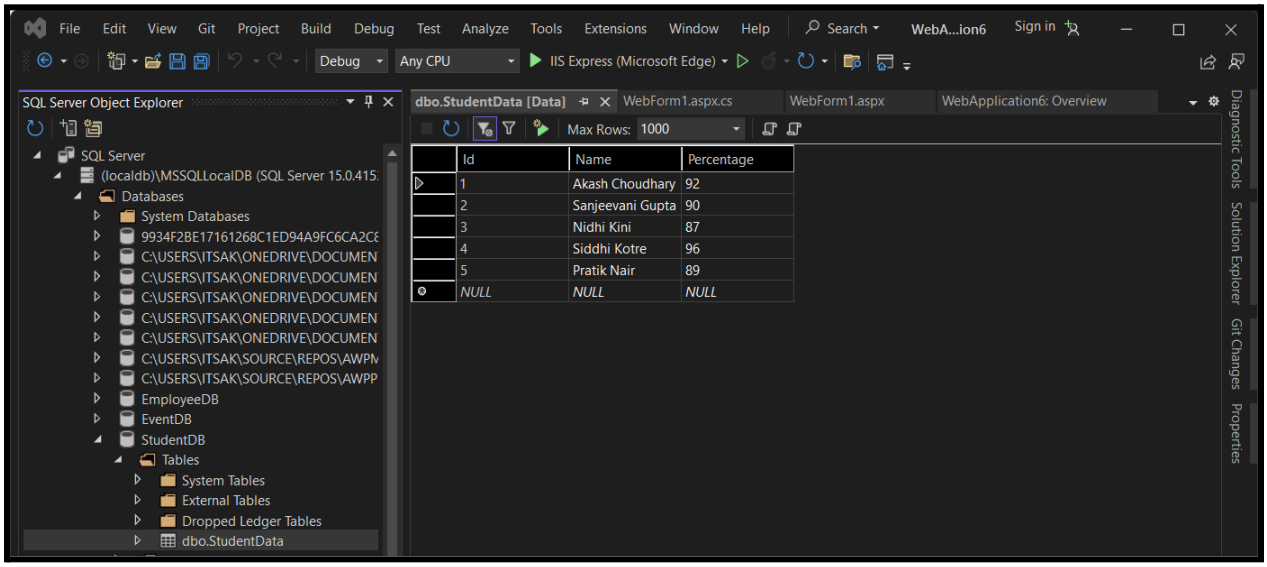
<asp:Label ID="Label3" runat="server" Text="Percentage:"></asp:Label>
<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>

<asp:Button ID="Button1" runat="server" Text="Insert" />
<asp:Button ID="Button2" runat="server" Text="Search" />

No issues found
Design
Split
WebForm1.aspx [Design]
14 Akash Choudhary
Name:
Percentage:
Insert
Search
Column0 Column1 Column2
abc abc abc
Root html body
Output
Ready
Add to Source Control
Select Repository


localhost:44312/WebForm1.aspx
https://localhost:44312/WebForm1.aspx
14 Akash Choudhary
Name: Pratik Nair
Percentage: 89
Insert
Search
Id Name Percentage
1 Akash Choudhary 92
2 Sanjeevani Gupta 90
3 Nidhi Kini 87
4 Siddhi Kotre 96
5 Pratik Nair 89





Conclusion: By implementing a WCF service and a corresponding client application, we've demonstrated the capability to distribute and consume student information efficiently over a network.

ASP.NET MVC

5. Develop an MVC application that displays data from the model.

Aim: To develop an MVC application displaying data from the model

Objectives:

1. Implement MVC architecture.
2. Create a model representing the data.
3. Develop views to display the data.
4. Configure controllers to interact with the model and views.

Theory:

ASP.NET MVC is a web application framework developed by Microsoft, which implements the Model-View-Controller pattern. Models represent the data of the application, views display the data to the user, and controllers handle user interaction, updating the model and selecting views to display.

Code:

> Student.cs

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

namespace WebApplication7.Models
{
   public class Student
   {
      public int Id { get; set; }
      public string Name { get; set; }
   }
}
```

> StudentController.cs

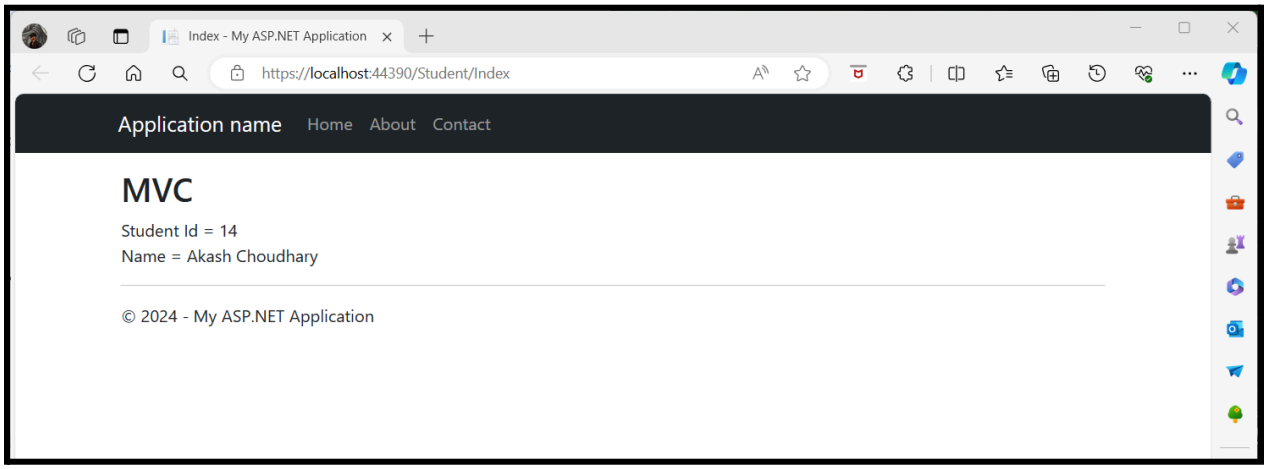
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using WebApplication7.Models;
```

```
namespace WebApplication7.Controllers
{
   public class StudentController : Controller
   {
      Student student = new Student();

      // GET: Student
      public ActionResult Index()
      {
         student.Id = 14;
         student.Name = "Akash Choudhary";
         return View(student);
      }
   }
}
```

> Index.cshtml

```
@{
   ViewBag.Title = "Index";
}

<h2>MVC</h2>
Student Id = @Model.Id
<br />
Name = @Model.Name
```

Output:



Conclusion: Developing an MVC application facilitates separation of concerns and allows for a more organized and maintainable codebase.

6. Write an application that accepts data from one page and displays it on another page using MVC (Passing data between views).

Aim: To create an MVC application that demonstrates passing data between views

Objectives:

1. Understand the MVC (Model-View-Controller) architecture.
2. Implement controllers to handle user requests and actions.
3. Pass data from one view to another using models, ViewBag, ViewData, or TempData.
4. Render the data on the target view.

Theory:

MVC (Model-View-Controller) is a software architectural pattern that divides an application into three interconnected components: the model, the view, and the controller.

- Model: Represents the data and business logic.
- View: Renders the user interface based on the model data.
- Controller: Handles user input, manipulates the model, and selects views to render to the user.

To pass data between views in MVC, several techniques can be used:

- Using Models: Passing data through strongly-typed models.
- ViewBag: A dynamic wrapper around ViewData that allows passing data between controllers and views.
- ViewData: A dictionary object that helps to pass data from a controller to a view.
- TempData: Similar to ViewData but persists only for the duration of an HTTP request.

Code:

> Employee.cs

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

namespace WebApplication8.Models
{
   public class Employee
   {
      public int Id { get; set; }
```

```
        public string Name { get; set; }
        public string Title { get; set; }
    }
}
```

> EmployeeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using WebApplication8.Models;

namespace WebApplication8.Controllers
{
    public class EmployeeController : Controller
    {
        // GET: Employee
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult DisplayEmployee()
        {
            Employee employee = new Employee();
            employee.Id = int.Parse(Request.Form["id"].ToString());
            employee.Name = Request.Form["name"];
            employee.Title = Request.Form["title"];
            return View(employee);
        }
    }
}
```

> Index.cshtml

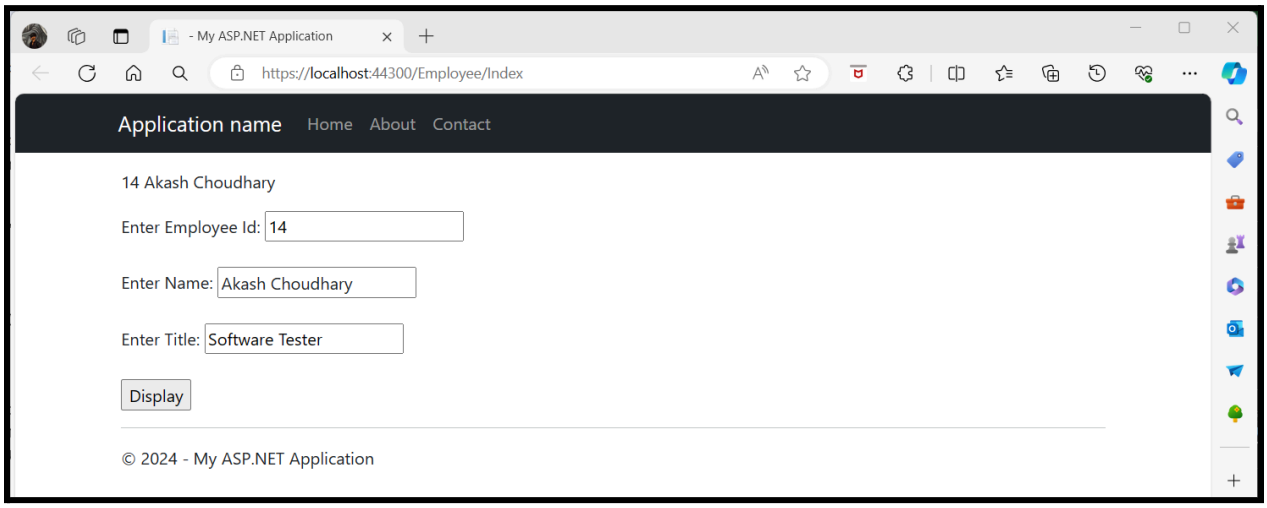
```
@model WebApplication8.Models.Employee

@{
    ViewBag.Title = "";
}
<p>14 Akash Choudhary</p>
@using (Html.BeginForm("DisplayEmployee", "Employee", FormMethod.Post))
{
<div>
    Enter Employee Id: @Html.TextBox("id")<br /><br />
    Enter Name: @Html.TextBox("name")<br /><br />
```

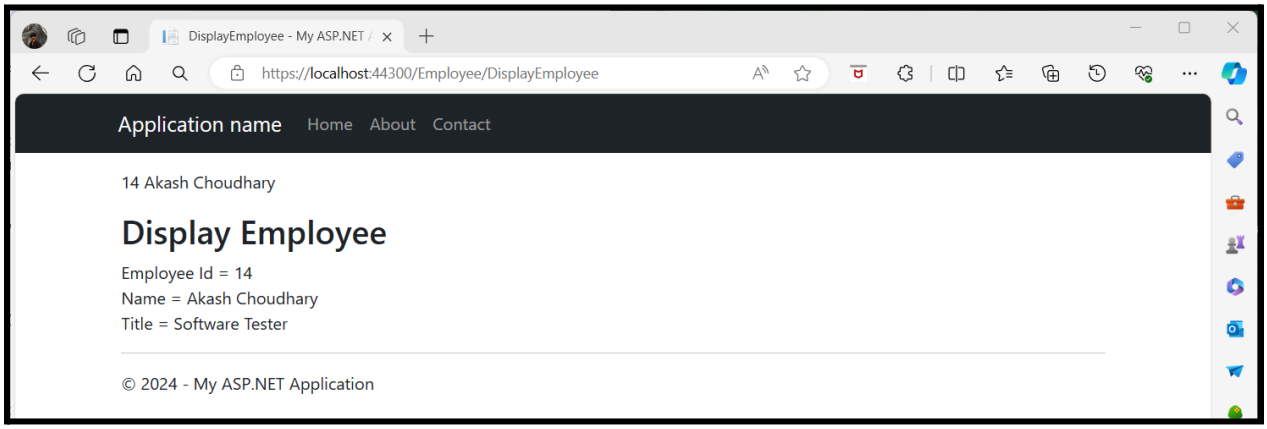
```
    Enter Title: @Html.TextBox("title")<br /><br />
    <input type="submit" value="Display" />
</div>
}
```

> DisplayEmployee.cshtml

```
@model WebApplication8.Models.Employee

@{
    ViewBag.Title = "DisplayEmployee";
}
<p>14 Akash Choudhary</p>
<h2>Display Employee</h2>
<div>
    Employee Id = @Model.Id<br />
    Name = @Model.Name<br />
    Title = @Model.Title
</div>
```

Output:





Conclusion: Through this practical exercise, we demonstrate the seamless flow of data between views in an MVC application, showcasing effective communication between the various components of the MVC architecture.

7. Write a program that demonstrates CRUD operations using MVC.

Aim: To develop a program demonstrating CRUD operations using MVC architecture

Objectives:

1. Implement Create, Read, Update, and Delete operations within an MVC application.
2. Utilize MVC's model-binding features to interact with data.
3. Validate user inputs and handle errors effectively.
4. Display data using appropriate views and templates.

Theory:

MVC (Model-View-Controller) is an architectural pattern widely used in web application development.

- Model: Represents the application's data and business logic.
- View: Presents the data to the user and handles user interactions.
- Controller: Acts as an intermediary, handling user input, processing requests, and updating the model and view accordingly.

CRUD (Create, Read, Update, Delete) operations are fundamental to database interactions in web applications.

- Create: Adding new records to the database.
- Read: Retrieving data from the database.
- Update: Modifying existing records in the database.
- Delete: Removing records from the database.

ASP.NET MVC provides a structured approach to building web applications, separating concerns and facilitating maintainability and testability.

Code:

> SQL Query

```
CREATE TABLE [dbo].[ProductData]
(
        [Id] INT NOT NULL PRIMARY KEY IDENTITY,
   [Name] VARCHAR(50) NOT NULL,
   [Price] INT NOT NULL
)
```

> Product.cs

```
using System;
```

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

namespace WebApplication9.Models
{
   public class Product
   {
      public int Id { get; set; }
      public string Name { get; set; }
      public int Price { get; set; }
   }
}
```

> ProductController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using WebApplication9.Models;
using System.Data.SqlClient;
using System.Data;

namespace WebApplication9.Controllers
{
   public class ProductController : Controller
   {
      Product product = new Product();
                                    SqlConnection conn = new SqlConnection(@"Data
Source=(localdb)\MSSQLLocalDB;Initial Catalog=ProductDB;Integrated Security=True;");

      // GET: Product
      public ActionResult Index()
      {
         SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM ProductData", conn);
         DataSet ds = new DataSet();
         da.Fill(ds);
         List<Product> prod = new List<Product>();
         for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
         {
            Product p1 = new Product();
            p1.Id = int.Parse(ds.Tables[0].Rows[i][0].ToString());
            p1.Name = ds.Tables[0].Rows[i][1].ToString();
            p1.Price = int.Parse(ds.Tables[0].Rows[i][2].ToString());
            prod.Add(p1);
```

```
        }
        return View(prod);
    }

    // GET: Product/Details/5
    public ActionResult Details(int id)
    {
        SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM ProductData WHERE Id
= " + id, conn);
        DataSet ds = new DataSet();
        da.Fill(ds);
        Product p1 = new Product();
        p1.Id = int.Parse(ds.Tables[0].Rows[0][0].ToString());
        p1.Name = ds.Tables[0].Rows[0][1].ToString();
        p1.Price = int.Parse(ds.Tables[0].Rows[0][2].ToString());
        return View(p1);
    }

    // GET: Product/Create
    public ActionResult Create()
    {
        return View();
    }

    // POST: Product/Create
    [HttpPost]
    public ActionResult Create(Product product)
    {
        try
        {
            // TODO: Add insert logic here
            conn.Open();
                SqlCommand cmd = new SqlCommand("INSERT INTO ProductData VALUES ('" +
product.Name + "', " + product.Price + ")", conn);
            cmd.ExecuteNonQuery();
            conn.Close();
            return RedirectToAction("Index");
        }
        catch
        {
            return View();
        }
    }

    // GET: Product/Edit/5
    public ActionResult Edit(int id)
```

```
        {
            SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM ProductData WHERE Id
= " + id, conn);
            DataSet ds = new DataSet();
            da.Fill(ds);
            Product p1 = new Product();
            p1.Id = int.Parse(ds.Tables[0].Rows[0][0].ToString());
            p1.Name = ds.Tables[0].Rows[0][1].ToString();
            p1.Price = int.Parse(ds.Tables[0].Rows[0][2].ToString());
            return View(p1);
        }

        // POST: Product/Edit/5
        [HttpPost]
        public ActionResult Edit(int id, Product product)
        {
            try
            {
                // TODO: Add update logic here
                conn.Open();
                    SqlCommand cmd = new SqlCommand("UPDATE ProductData SET Name = '" +
product.Name + "', Price = " + product.Price + " WHERE Id = " + id, conn);
                cmd.ExecuteNonQuery();
                conn.Close();
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        // GET: Product/Delete/5
        public ActionResult Delete(int id)
        {
            SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM ProductData WHERE Id
= " + id, conn);
            DataSet ds = new DataSet();
            da.Fill(ds);
            Product p1 = new Product();
            p1.Id = int.Parse(ds.Tables[0].Rows[0][0].ToString());
            p1.Name = ds.Tables[0].Rows[0][1].ToString();
            p1.Price = int.Parse(ds.Tables[0].Rows[0][2].ToString());
            return View(p1);
        }
```

```
        // POST: Product/Delete/5
        [HttpPost]
        public ActionResult Delete(int id, FormCollection collection)
        {
            try
            {
                // TODO: Add delete logic here
                conn.Open();
                 SqlCommand cmd = new SqlCommand("DELETE FROM ProductData WHERE Id =
" + id, conn);
                cmd.ExecuteNonQuery();
                conn.Close();
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }
    }
}
```

> Index.cshtml

```
@model IEnumerable<WebApplication9.Models.Product>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Name)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Price)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
```

```
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Name)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Price)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.Id }) |
            @Html.ActionLink("Details", "Details", new { id=item.Id }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.Id })
        </td>
    </tr>
}

</table>
```

> Create.cshtml

```
@model WebApplication9.Models.Product

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Product</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label
col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class =
"form-control" } })
                @Html.ValidationMessageFor(model => model.Name, "", new { @class =
"text-danger" })
            </div>
        </div>
```

```
        <div class="form-group">
            @Html.LabelFor(model => model.Price, htmlAttributes: new { @class = "control-label
col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Price, new { htmlAttributes = new { @class =
"form-control" } })
                    @Html.ValidationMessageFor(model => model.Price, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

> Edit.cshtml

```
@model WebApplication9.Models.Product

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Product</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.Id)
```

```
        <div class="form-group">
            @Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label
col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class =
"form-control" } })
                @Html.ValidationMessageFor(model => model.Name, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Price, htmlAttributes: new { @class = "control-label
col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Price, new { htmlAttributes = new { @class =
"form-control" } })
                @Html.ValidationMessageFor(model => model.Price, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

> Delete.cshtml

```
@model WebApplication9.Models.Product

@{
    ViewBag.Title = "Delete";
}
```

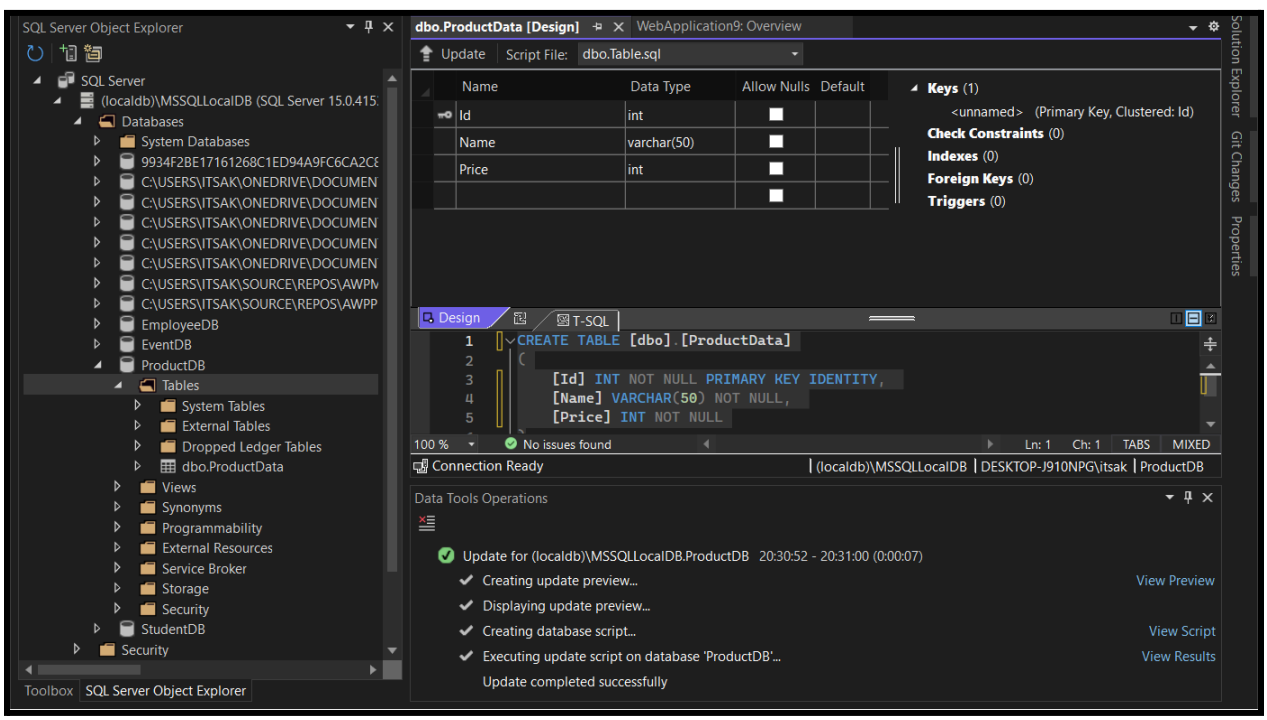
```
<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
   <h4>Product</h4>
   <hr />
   <dl class="dl-horizontal">
      <dt>
         @Html.DisplayNameFor(model => model.Name)
      </dt>

      <dd>
         @Html.DisplayFor(model => model.Name)
      </dd>

      <dt>
         @Html.DisplayNameFor(model => model.Price)
      </dt>

      <dd>
         @Html.DisplayFor(model => model.Price)
      </dd>

   </dl>

   @using (Html.BeginForm()) {
      @Html.AntiForgeryToken()

      <div class="form-actions no-color">
         <input type="submit" value="Delete" class="btn btn-default" /> |
         @Html.ActionLink("Back to List", "Index")
      </div>
   }
</div>
```

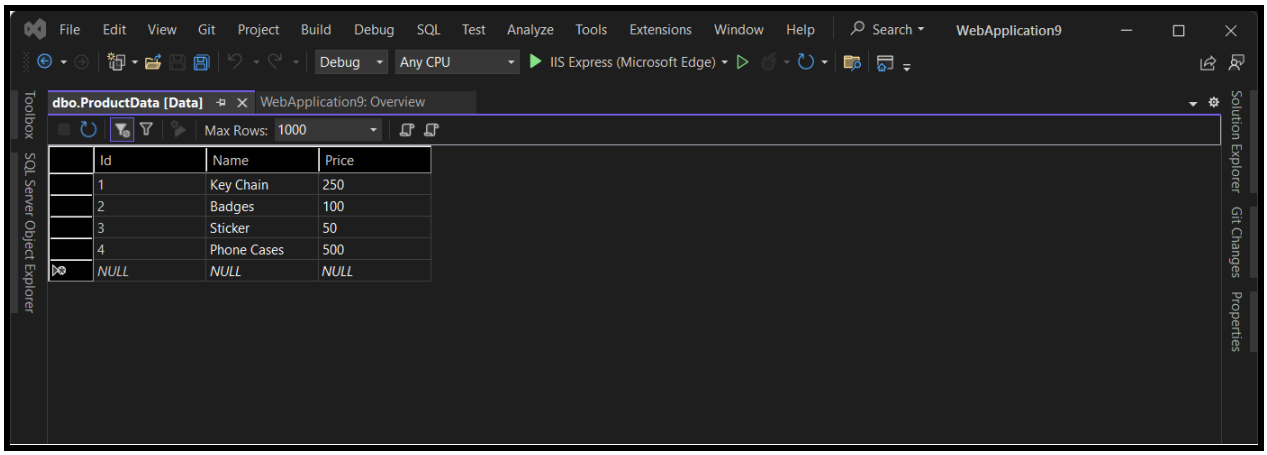
> Details.cshtml

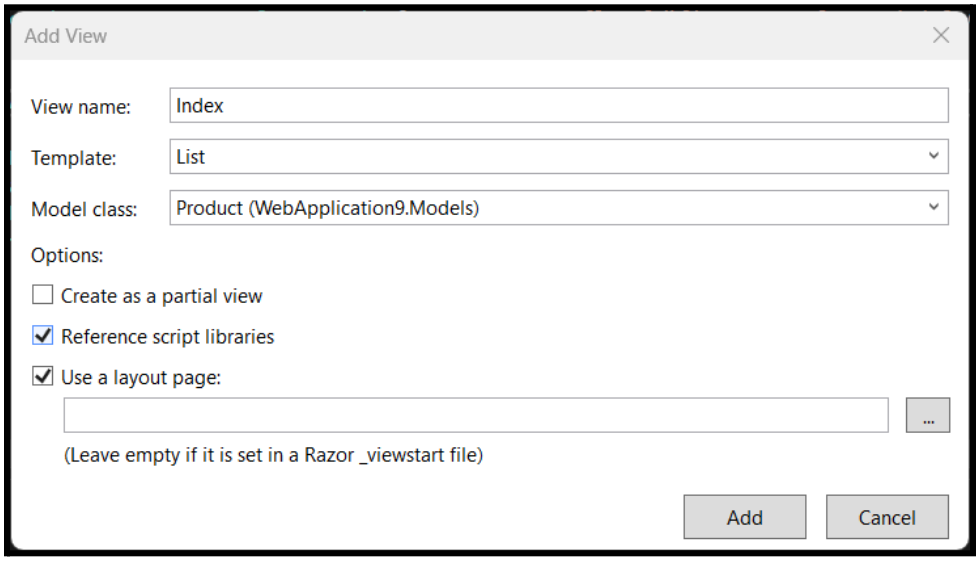
```
@model WebApplication9.Models.Product

@{
   ViewBag.Title = "Details";
}

<h2>Details</h2>

<div>
   <h4>Product</h4>
```

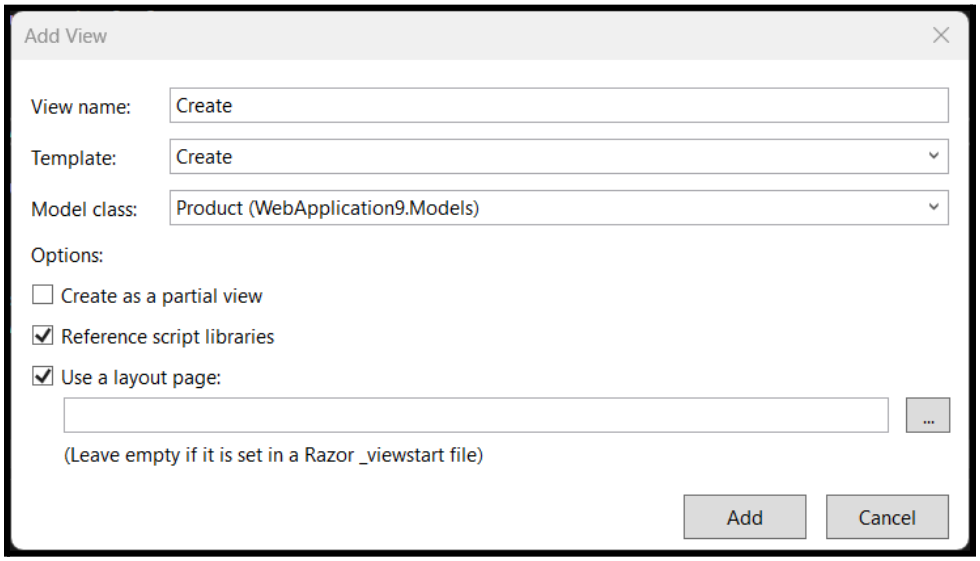
```
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Name)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Name)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Price)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Price)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.Id }) |
    @Html.ActionLink("Back to List", "Index")
</p>
```

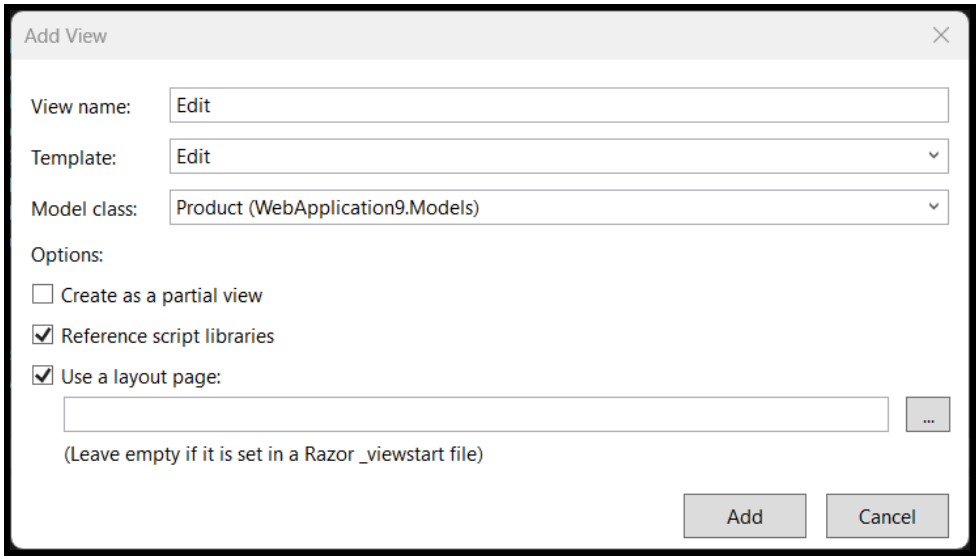
Output:

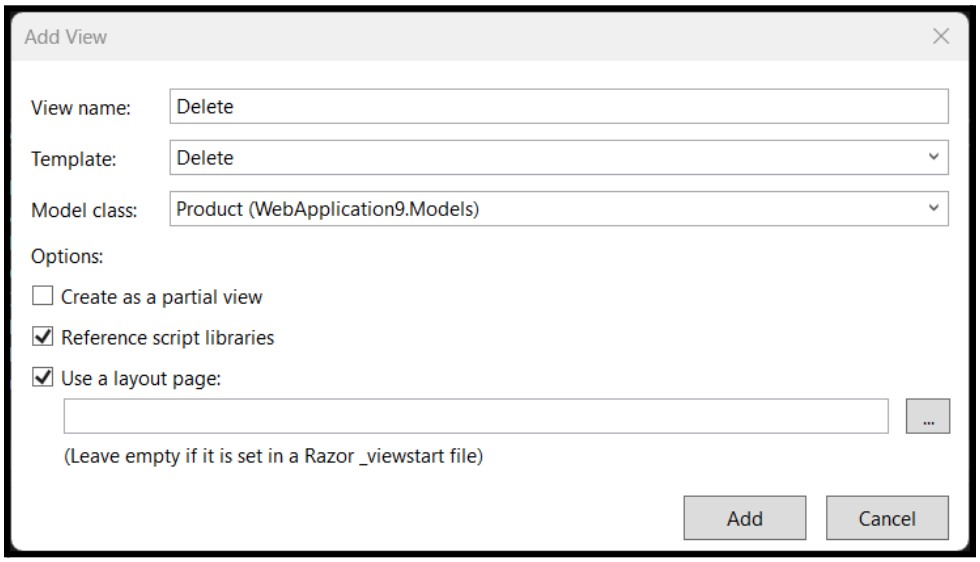


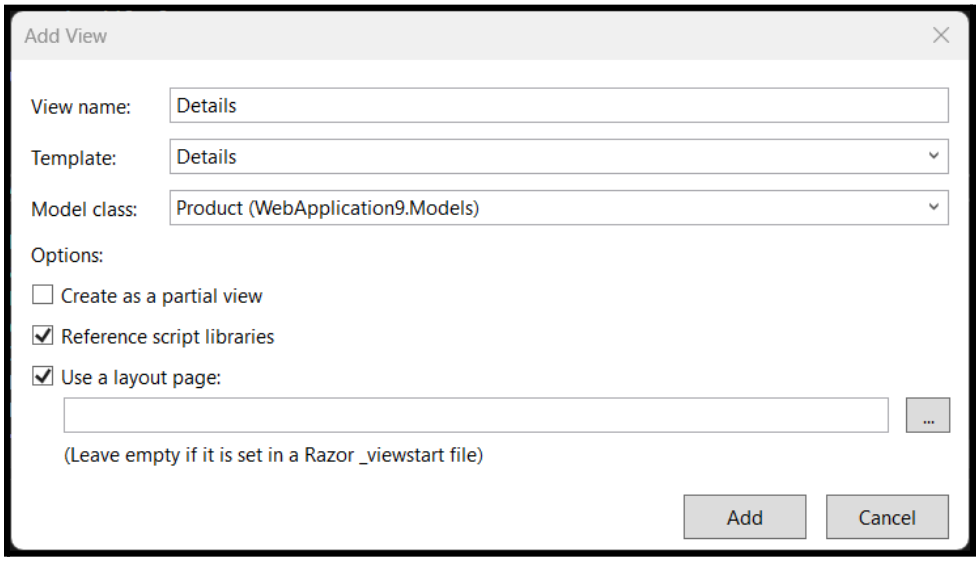

File Edit View Git Project Build Debug SQL Test Analyze Tools Extensions Window Help
Search
WebApplication9
Debug
Any CPU
IIS Express (Microsoft Edge)
Toolbox
SQL Server Object Explorer
dbo.ProductData [Data]
WebApplication9: Overview
Max Rows: 1000
Id Name Price
1 Key Chain 250
2 Badges 100
3 Sticker 50
4 Phone Cases 500
NULL NULL NULL
Solution Explorer
Git Changes
Properties

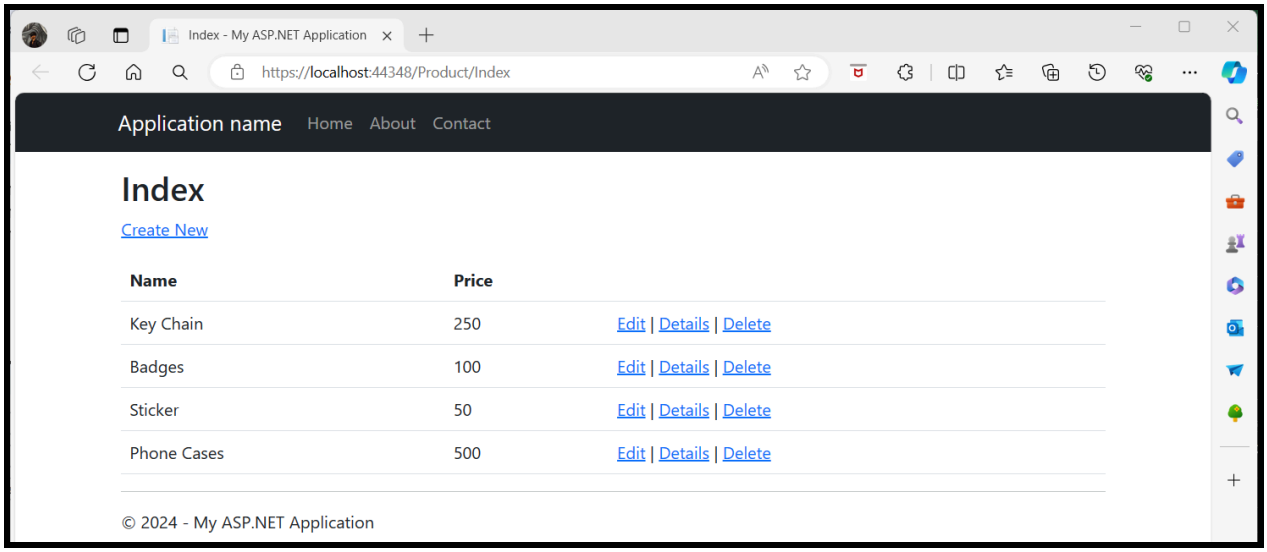

Add View
View name: Index
Template: List
Model class: Product (WebApplication9.Models)
Options:
Create as a partial view
Reference script libraries
Use a layout page:
(Leave empty if it is set in a Razor _viewstart file)
Add
Cancel

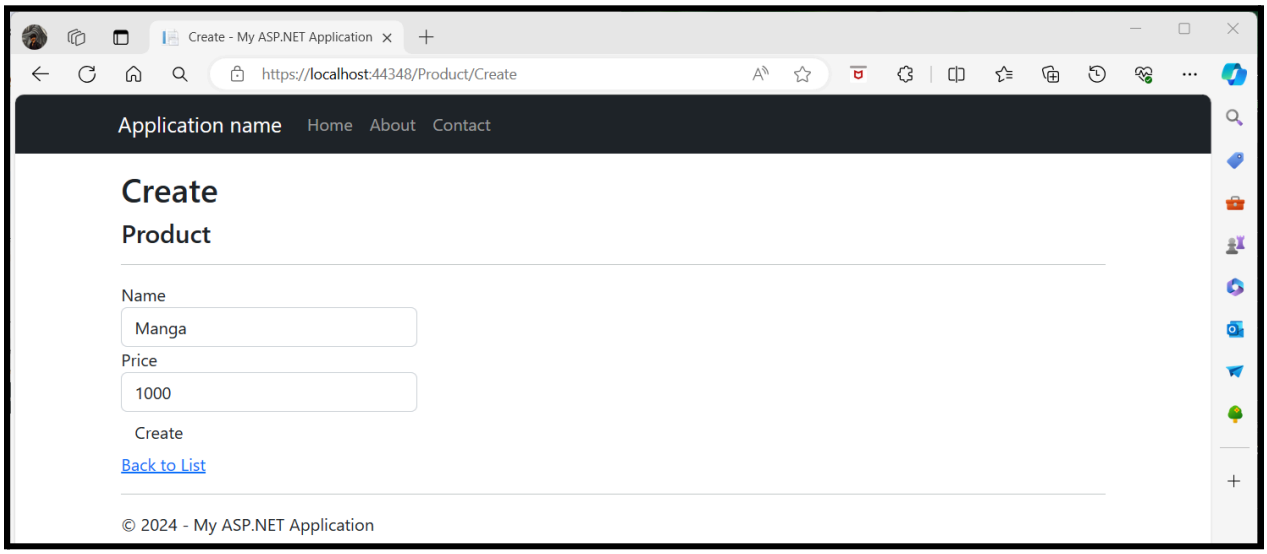

Add View
View name: Create
Template: Create
Model class: Product (WebApplication9.Models)
Options:
Create as a partial view
Reference script libraries
Use a layout page:
(Leave empty if it is set in a Razor _viewstart file)
Add
Cancel

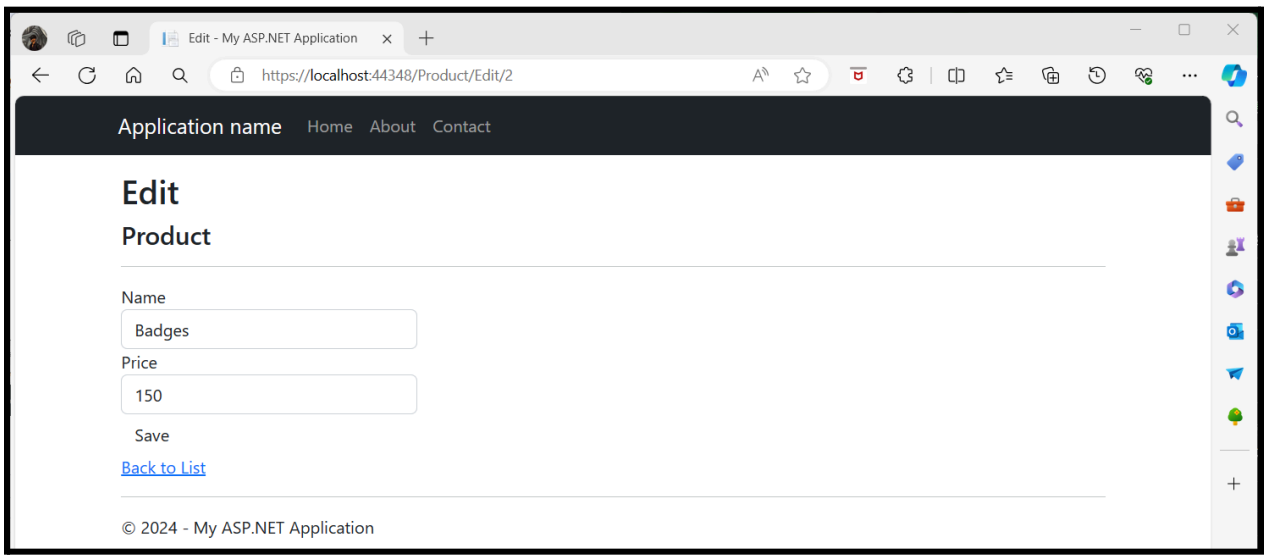

Add View
View name:
Edit
Template:
Edit
Model class:
Product (WebApplication9.Models)
Options:
Create as a partial view
Reference script libraries
Use a layout page:
...
(Leave empty if it is set in a Razor _viewstart file)
Add
Cancel

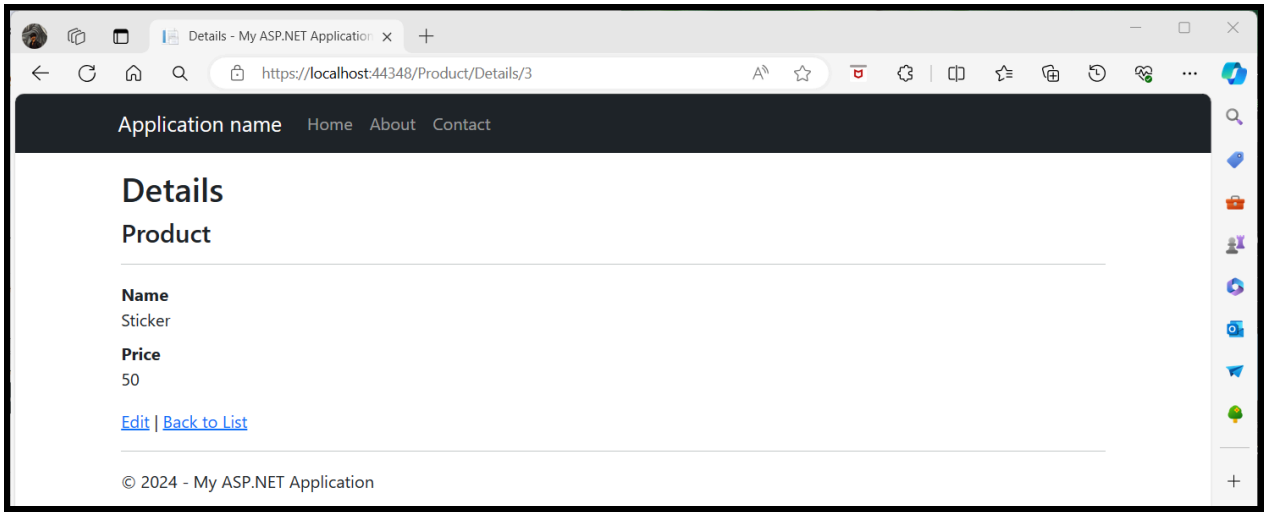

Add View
View name:
Delete
Template:
Delete
Model class:
Product (WebApplication9.Models)
Options:
Create as a partial view
Reference script libraries
Use a layout page:
...
(Leave empty if it is set in a Razor _viewstart file)
Add
Cancel

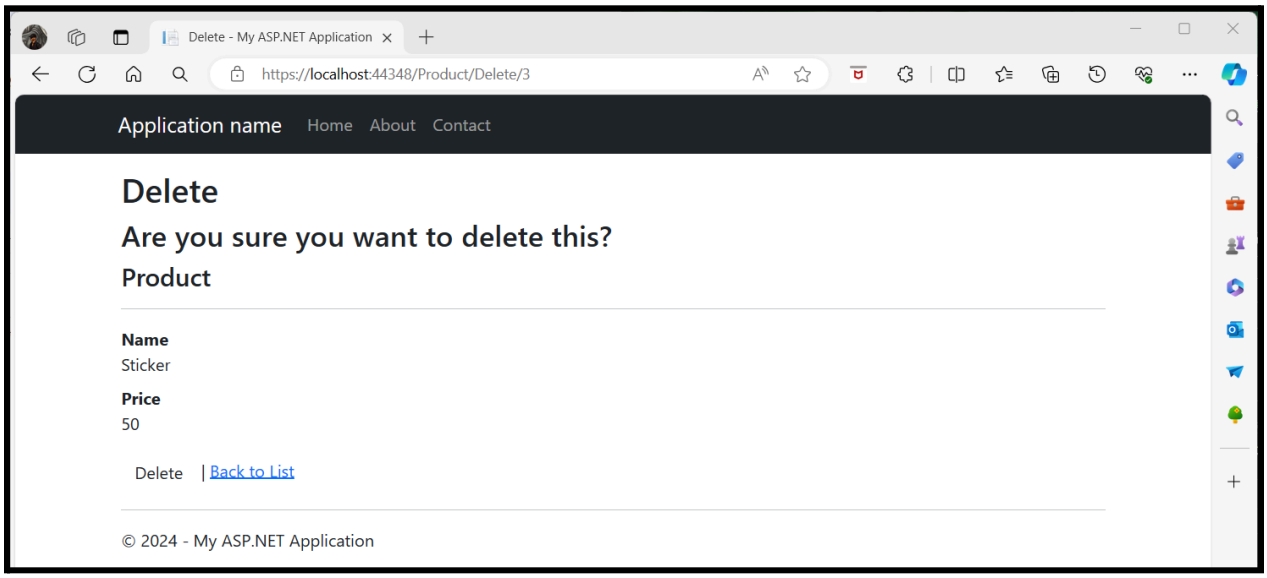

Add View
View name:
Details
Template:
Details
Model class:
Product (WebApplication9.Models)
Options:
Create as a partial view
Reference script libraries
Use a layout page:
...
(Leave empty if it is set in a Razor _viewstart file)
Add
Cancel

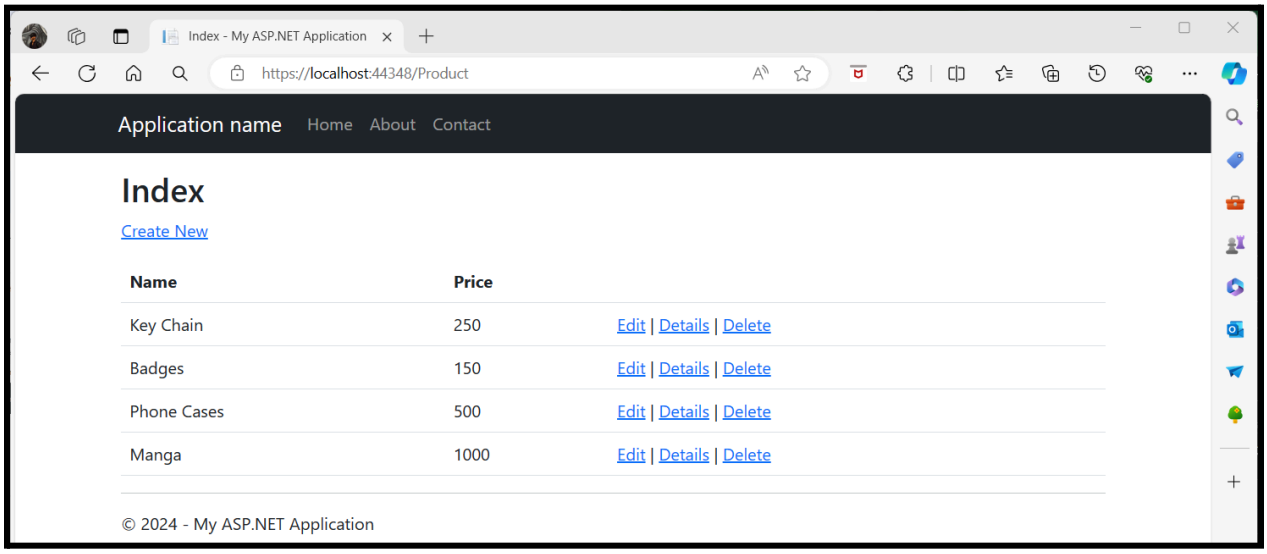

Index - My ASP.NET Application
https://localhost:44348/Product/Index
Application name Home About Contact
Index
Create New
Name Price
Key Chain 250 Edit | Details | Delete
Badges 100 Edit | Details | Delete
Sticker 50 Edit | Details | Delete
Phone Cases 500 Edit | Details | Delete
© 2024 - My ASP.NET Application

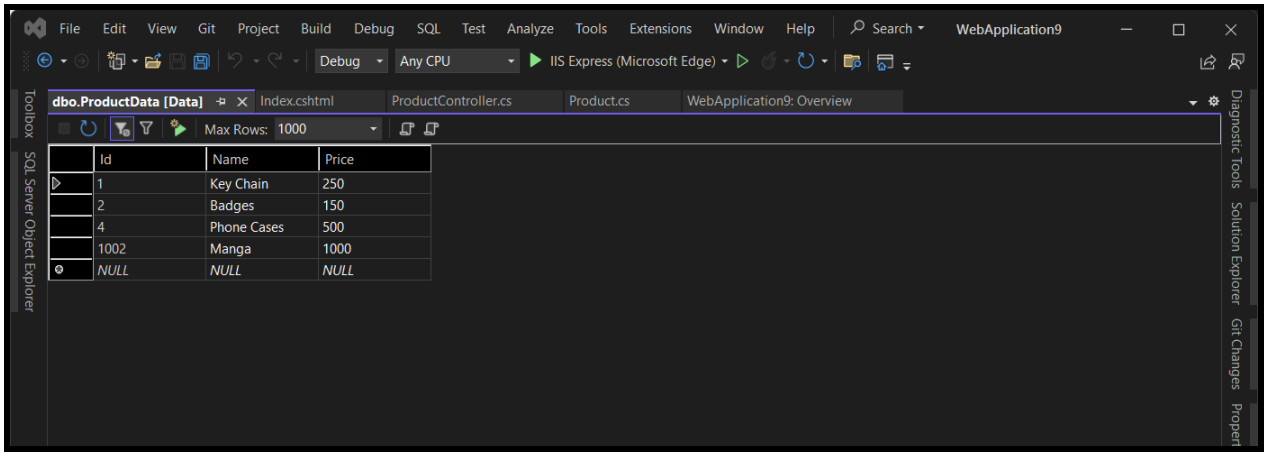

Create - My ASP.NET Application
https://localhost:44348/Product/Create
Application name Home About Contact
Create
Product
Name
Manga
Price
1000
Create
Back to List
© 2024 - My ASP.NET Application


Edit - My ASP.NET Application
https://localhost:44348/Product/Edit/2
Application name Home About Contact
Edit
Product
Name
Badges
Price
150
Save
Back to List
© 2024 - My ASP.NET Application


Details - My ASP.NET Application
https://localhost:44348/Product/Details/3
Application name Home About Contact
Details
Product
Name
Sticker
Price
50
Edit | Back to List
© 2024 - My ASP.NET Application


Delete - My ASP.NET Application
https://localhost:44348/Product/Delete/3
Application name Home About Contact
Delete
Are you sure you want to delete this?
Product
Name
Sticker
Price
50
Delete | Back to List
© 2024 - My ASP.NET Application


Index - My ASP.NET Application
https://localhost:44348/Product
Application name Home About Contact
Index
Create New
Name Price
Key Chain 250 Edit | Details | Delete
Badges 150 Edit | Details | Delete
Phone Cases 500 Edit | Details | Delete
Manga 1000 Edit | Details | Delete
© 2024 - My ASP.NET Application



Conclusion: By implementing CRUD operations in an MVC application, developers can understand how to efficiently manage data interactions within a web environment, adhering to best practices in software design and development.