**Aim:** Design Web Application to produce and consume a web Service, WCF Service

### Code:

1. Write a program to implement to create a simple web service that converts the temperature from Fahrenheit to Celsius and vice a versa.

Temperature.aspx

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="Temperature.aspx.cs" Inherits="Temperature\_practical\_1.Temperature" %>

<!DOCTYflE html>

<html xmlns="<http://www.w3.org/1999/xhtml>">

<head runat="server">

<title></title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

=<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>

<br />

<asp:RadioButtonList ID="RadioButtonList1" runat="server" AutoflostBack="True" OnSelectedIndexChanged="RadioButtonList1\_SelectedIndexChanged">

<asp:ListItem>celsius</asp:ListItem>

<asp:ListItem>Fahrenhiet</asp:ListItem>

</asp:RadioButtonList>

<br />

</div>

</form>

</body>

</html>

Temperature.aspx.cs

using System;

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

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls;

namespace Temperature\_practical\_1

{

public partial class Temperature : System.Web.UI.flage

{

protected void flage\_Load(object sender, EventArgs e)

{

}

protected void RadioButtonList1\_SelectedIndexChanged(object sender, EventArgs e)

{

mywebservice ws=new mywebservice(); switch(RadioButtonList1.SelectedIndex)

{

case 0: TextBox2.Text = ws.ftoc(Convert.ToDouble(TextBox1.Text)).ToString();

break;

case 1:TextBox2.Text = ws.ctof(Convert.ToDouble(TextBox1.Text)).ToString();

break;

}

}

}

}

Mywebservices.asmx.cs using System;

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

using System.Web;

using System.Web.Services;

namespace Temperature\_practical\_1

{

/// <summary>

/// Summary description for mywebservice

/// </summary>

[WebService( amespace = "<http://tempuri.org/>")] [WebServiceBinding(ConformsTo = Wsiflrofiles.Basicflrofile1\_1)] [System.ComponentModel.ToolboxItem(false)]

// To allow this Web Service to be called from script, using ASfl. ET AJAX, uncomment the following line.

// [System.Web.Script.Services.ScriptService]

public class mywebservice : System.Web.Services.WebService

{

[WebMethod]

public string HelloWorld()

{

return "Hello World";

}

[WebMethod]

public double ctof(double c)

{

return ((c \* 1.8) + 32);

}

[WebMethod]

public double ftoc(double f)

{

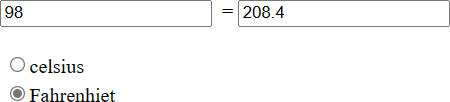
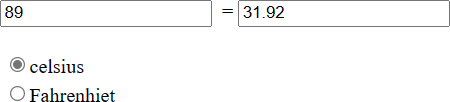
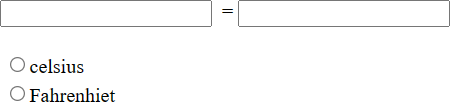
return ((f - 32) \* 0.56);

}

}

}

### Output:



1. Write a program to implement to create a simple web service accepts a number and return the factorial of the number.

### Code:

factorial.cs:

using System;

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

using System.Web;

using System.Web.Services; using System.Web.UI;

using System.Web.UI.WebControls;

namespace prac8b

{

public partial class factorial : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

protected void Button1\_Click(object sender, EventArgs e)

{

myWebService ws = new myWebService(); double input = 0;

if (double.TryParse(TextBox1.Text, out input))

{

double result = ws.fac(input); Label1.Text = result.ToString();

}

else

{

Label1.Text = "Invalid input";

}

}

}

}

myWebService.asmx.cs:

using System;

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

using System.Web;

using System.Web.Services;

namespace prac8b

{

/// <summary>

/// Summary description for myWebService

/// </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 myWebService : System.Web.Services.WebService

{

[WebMethod]

public string HelloWorld()

{

return "Hello World";

}

public double fac(double c)

{

double factorial = 1;

for (int i = 1; i <= c; i++)

{

factorial \*= i;

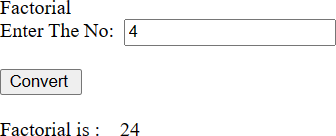
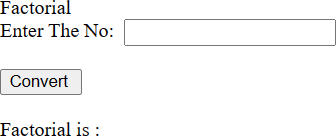
return factorial;

}

}

}

### Output:



1. Write a program to implement to create a WCF accepts a number and return the factors of that number

Code:

**ProduceApp:**

Iservice1.cs

using System;

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

using System.Runtime.Serialization; using System.ServiceModel;

using System.ServiceModel.Web; using System.Text;

namespace produceapp

{

// OTE: You can use the "Rename" command on the "Refactor" menu to change the interface name "IService1" in both code and config file together.

[ServiceContract]

public interface IService1

{

[OperationContract]

int[] FindFactors(int number);

}

}

Service1.svc.cs

using System;

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

using System.Runtime.Serialization; using System.ServiceModel;

using System.ServiceModel.Web; using System.Text;

namespace produceapp

{

public class Service1 : IService1

{

public int[] FindFactors(int number)

{

List<int> factors = new List<int>(); for (int i = 1; i <= number; i++)

{

if (number % i == 0)

{

factors.Add(i);

}

}

return factors.ToArray();

}

}

}

**ConsumeApp:**

Factor.aspx

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="factors.aspx.cs" Inherits="consumeapp.factors" %>

<!DOCTYflE html>

<html xmlns="<http://www.w3.org/1999/xhtml>">

<head runat="server">

<title>Factorial Calculator</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<h1>Factorial Calculator</h1>

<asp:Label ID="InputLabel" runat="server" Text="Enter a number:"></asp:Label>

<asp:TextBox ID=" umberTextBox" runat="server"></asp:TextBox>

<asp:Button ID="CalculateButton" runat="server" Text="Find Factors" OnClick="CalculateButton\_Click" />

<br />

<asp:GridView ID="FactorsGridView" runat="server" AutoGenerateColumns="False">

<Columns>

<asp:BoundField DataField="Factors" HeaderText="Factors" />

</Columns>

</asp:GridView>

</div>

</form>

</body>

</html>

Factor.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 consumeapp

{

public partial class factors : System.Web.UI.flage

{

protected void flage\_Load(object sender, EventArgs e)

{

}

protected void CalculateButton\_Click(object sender, EventArgs e)

{

"Factor"

int number = Convert.ToInt32( umberTextBox.Text); myfactors.Service1Client s=new myfactors.Service1Client(); int[] factors = s.FindFactors(number);

DataTable dt = new DataTable();

dt.Columns.Add("Factors", typeof(int)); // Add a column with the name

foreach (int factor in factors)

{

dt.Rows.Add(factor);

}

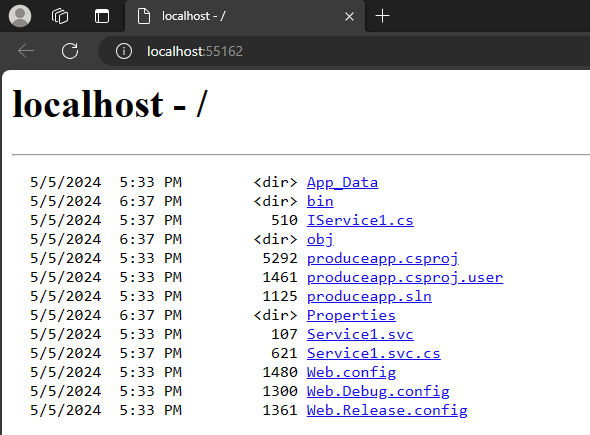
FactorsGridView.DataSource = dt; FactorsGridView.DataBind();

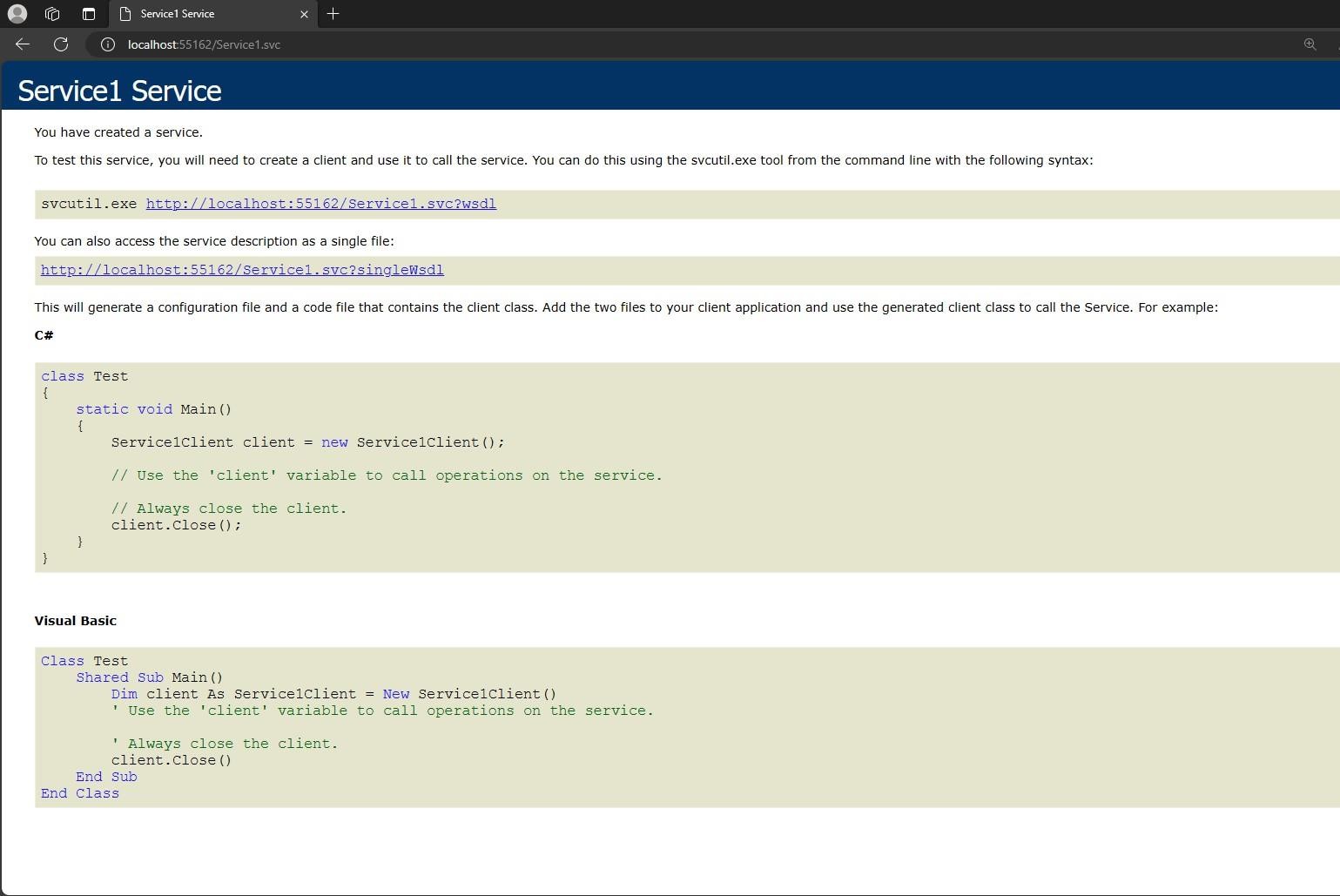
}

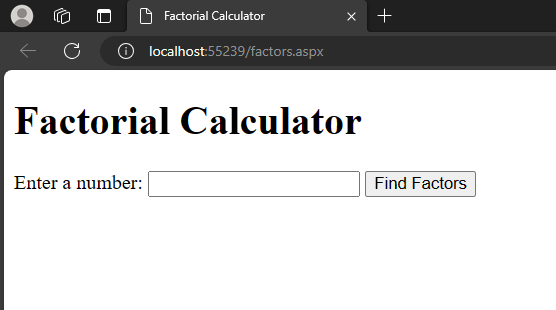
}

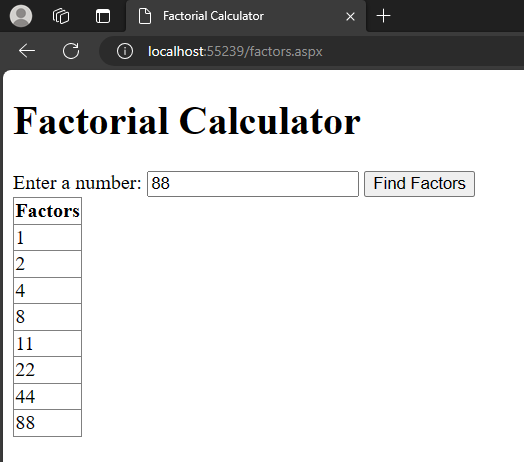
}

Output:









1. **Write a program to implement to create a WCF accepts a String and return reverse (Self-study)**

**CODE:**

**IService1.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization; using System.ServiceModel;

using System.ServiceModel.Web; using System.Text;

namespace produceapp

{

// OTE: You can use the "Rename" command on the "Refactor" menu to change the interface name "IService1" in both code and config file together.

[ServiceContract]

public interface IService1

{

[OperationContract]

string ReverseString(string str);

}

}

**IService1.svc**

using System;

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

using System.Runtime.Serialization; using System.ServiceModel;

using System.ServiceModel.Web; using System.Text;

namespace produceapp

{

public class Service1 : IService1

{

public string ReverseString(string input)

{

char[] charArray = input.ToCharArray(); Array.Reverse(charArray);

return new string(charArray);

}

}

}

**WebForm1.aspx**

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="consumerapp.WebForm1" %>

<!DOCTYflE html>

<html xmlns="<http://www.w3.org/1999/xhtml>">

<head runat="server">

<title></title>

</head>

<body>

<form id="form1" runat="server">

<div>

<h1>Reverse String</h1>

<asp:TextBox ID="txtInput" runat="server"></asp:TextBox>

<asp:Button ID="btnReverse" runat="server" Text="Reverse" OnClick="btnReverse\_Click" />

<br />

<asp:Label ID="lblReversed" runat="server" Text=""></asp:Label>

</div>

</form>

</body>

</html>

**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 consumerapp

{

public partial class WebForm1 : System.Web.UI.flage

{

protected void flage\_Load(object sender, EventArgs e)

{

}

protected void btnReverse\_Click(object sender, EventArgs e)

{

// Get the input string from the TextBox string input = txtInput.Text;

reverseString.Service1Client s = new reverseString.Service1Client();

// Reverse the input string

string reversed = s.ReverseString(input);

// Display the reversed string in the Label

lblReversed.Text = reversed;

}

**)**

**)**

**OUHflUH:**

