

Practical Number 1: Design a Web Application for an Organization with Registration forms and advanced controls.

Registration Form

First Name:

Last Name:

Email:

Date of Birth:

Gender: Select Gender ▼

☐ HR

☐ IT

☐ Finance

☐ I accept the terms and conditions

Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="Experiment_Number_1.Default" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Registration Form</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <h2>Registration Form</h2>
            <table>
                <tr>
                    <td><label for="txtFirstName">First Name:</label></td>
                    <td><asp:TextBox ID="txtFirstName" runat="server"
CssClass="form-control"></asp:TextBox></td>
                </tr>
                <tr>
                    <td><label for="txtLastName">Last Name:</label></td>
                    <td><asp:TextBox ID="txtLastName" runat="server"
CssClass="form-control"></asp:TextBox></td>
                </tr>
                <tr>
                    <td><label for="txtEmail">Email:</label></td>
                    <td><asp:TextBox ID="txtEmail" runat="server" CssClass="form-
control"></asp:TextBox></td>
                </tr>
            </table>
            <p><input type="radio"/> HR<br>
<input type="radio"/> IT<br>
<input type="radio"/> Finance</p>
            <p><input type="checkbox"/> I accept the terms and conditions</p>
            <p><input type="button" value="Register"/></p>
        </div>
    </form>
</body>
</html>
```

```

        </tr>
        <tr>
            <td><label for="txtDOB">Date of Birth:</label></td>
            <td><asp:TextBox ID="txtDOB" runat="server" TextMode="Date"
CssClass="form-control"></asp:TextBox></td>
        </tr>
        <tr>
            <td><label for="ddlGender">Gender:</label></td>
            <td>
                <asp:DropDownList ID="ddlGender" runat="server"
CssClass="form-control">
                    <asp:ListItem Text="Select Gender"
Value=""></asp:ListItem>
                    <asp:ListItem Text="Male"
Value="Male"></asp:ListItem>
                    <asp:ListItem Text="Female"
Value="Female"></asp:ListItem>
                    <asp:ListItem Text="Other"
Value="Other"></asp:ListItem>
                </asp:DropDownList>
            </td>
        </tr>
        <tr>
            <td><label>Department:</label></td>
            <td>
                <asp:RadioButtonList ID="rblDepartment" runat="server">
                    <asp:ListItem Text="HR" Value="HR"></asp:ListItem>
                    <asp:ListItem Text="IT" Value="IT"></asp:ListItem>
                    <asp:ListItem Text="Finance"
Value="Finance"></asp:ListItem>
                </asp:RadioButtonList>
            </td>
        </tr>
        <tr>
            <td>
                <asp:CheckBox ID="chkTerms" runat="server" Text="I accept
the terms and conditions" />
            </td>
        </tr>
        <tr>
            <td colspan="2">
                <asp:Button ID="btnSubmit" runat="server" Text="Register"
OnClick="btnSubmit_Click" />
            </td>
        </tr>
    </table>
</div>
</form>
</body>
</html>

```

Default.aspx.cs

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

namespace Experiment_Number_1
{
    public partial class Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void btnSubmit_Click(object sender, EventArgs e)
        {
            if (chkTerms.Checked)
            {
                string firstName = txtFirstName.Text;
                string lastName = txtLastName.Text;
                string email = txtEmail.Text;
                string dob = txtDOB.Text;
                string gender = ddlGender.SelectedValue;
                string department = rblDepartment.SelectedValue;

                // Display confirmation message
                Response.Write($"<h3>Registration Successful!</h3>");
                Response.Write($"<p>Name: {firstName} {lastName}</p>");
                Response.Write($"<p>Email: {email}</p>");
                Response.Write($"<p>Date of Birth: {dob}</p>");
                Response.Write($"<p>Gender: {gender}</p>");
                Response.Write($"<p>Department: {department}</p>");
            }
            else
            {
                Response.Write("<h3 style='color:red'>Please accept the terms and conditions.</h3>");
            }
        }
    }
}
```

Registration Form

localhost:44391/Default.aspx

New TabOBE | MasterSoftAdobe Acrobat

Registration Form

First Name:	<input type="text" value="Ganesh"/>
Last Name:	<input type="text" value="Bhagwat"/>
Email:	<input type="text" value="Gbhagwat@gmail.com"/>
Date of Birth:	<input type="text" value="22 - 09 - 1987"/>
Gender:	<input type="text" value="Male"/>
	<input checked="" type="radio"/> HR
	<input type="radio"/> IT
	<input type="radio"/> Finance
Department:	
<input checked="" type="checkbox"/> I accept the terms and conditions	
<input type="button" value="Register"/>	

Registration Form

localhost:44391/Default.aspx

New Tab OBE | MasterSoft Adobe Acrobat

Registration Successful!

Name: Ganesh Bhagwat

Email: Gbhagwat@gmail.com

Date of Birth: 1987-09-22

Gender: Male

Department: HR

Registration Form

First Name:	Ganesh
Last Name:	Bhagwat
Email:	Gbhagwat@gmail.com
Date of Birth:	22 - 09 - 1987
Gender:	Male
Department:	<input checked="" type="radio"/> HR <input type="radio"/> IT <input type="radio"/> Finance
<input checked="" type="checkbox"/> I accept the terms and conditions	
<input type="button" value="Register"/>	

Practical Number 2

Aim: Create a website using the master page concept.

Overview of the Master Page Concept

The **Master Page** allows you to define a consistent layout and design for multiple pages in a web application. Content pages can use the master page to inherit this layout while providing their unique content.

1. Steps to Create the Website

Step 1: Create an ASP.NET Web Forms Project

1. Open Visual Studio.
 2. Create a new project: **ASP.NET Web Forms Application**.
 3. Name the project WebsiteWithMasterPage.
-

Step 2: Add a Master Page

1. Right-click on the project in **Solution Explorer**.
2. Select **Add > New Item > Web Forms Master Page**.
3. Name it SiteMaster.master.

SiteMaster.master

This is the layout for the website.

```
<% @ Master Language="C#" AutoEventWireup="true"
CodeFile="SiteMaster.master.cs" Inherits="SiteMaster" %>
```

```
<!DOCTYPE html>
<html>
<head>
  <title>My Website</title>
  <link rel="stylesheet" href="styles.css" />
</head>
<body>
  <div class="header">
    <h1>Welcome to My Website</h1>
    <nav>
      <a href="Home.aspx">Home</a>
```

```

        <a href="About.aspx">About</a>
        <a href="Contact.aspx">Contact</a>
    </nav>
</div>

<div class="content">
    <asp:ContentPlaceHolder ID="MainContent"
runat="server"></asp:ContentPlaceHolder>
</div>

<div class="footer">
    <p>&copy; 2025 My Website. All Rights Reserved.</p>
</div>
</body>
</html>

```

Step 3: Add Content Pages

1. Right-click on the project in **Solution Explorer**.
2. Select **Add > New Item > Web Form with Master Page**.
3. Name the page Home.aspx and link it to SiteMaster.master.
4. Repeat to create About.aspx and Contact.aspx.

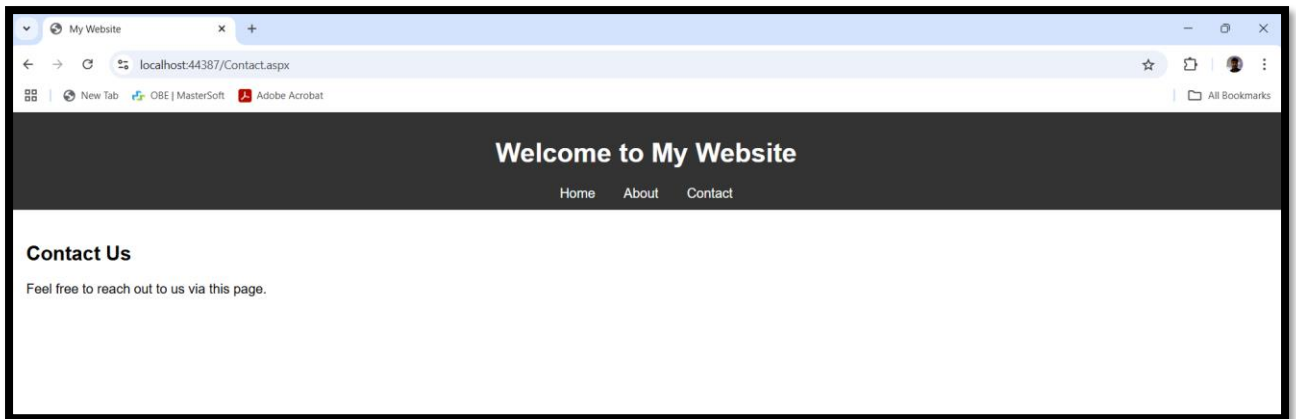
Home.aspx

```

<% @ Page Title="Home" Language="C#"
MasterPageFile="~/SiteMaster.master" AutoEventWireup="true"
CodeFile="Home.aspx.cs" Inherits="Home" %>

<asp:Content ID="Content1" ContentPlaceHolderID="MainContent"
runat="server">
    <h2>Welcome to the Home Page</h2>
    <p>This is the main content of the Home page.</p>
</asp:Content>

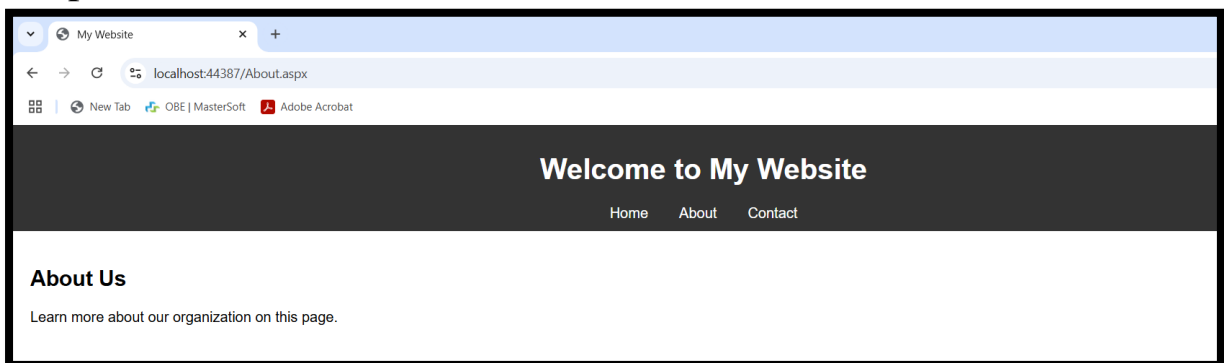
```



About.aspx

```
<% @ Page Title="About" Language="C#"
MasterPageFile="~/SiteMaster.master" AutoEventWireup="true"
CodeFile="About.aspx.cs" Inherits="About" %>
```

```
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent"
runat="server">
    <h2>About Us</h2>
    <p>Learn more about our organization on this page.</p>
</asp:Content>
```



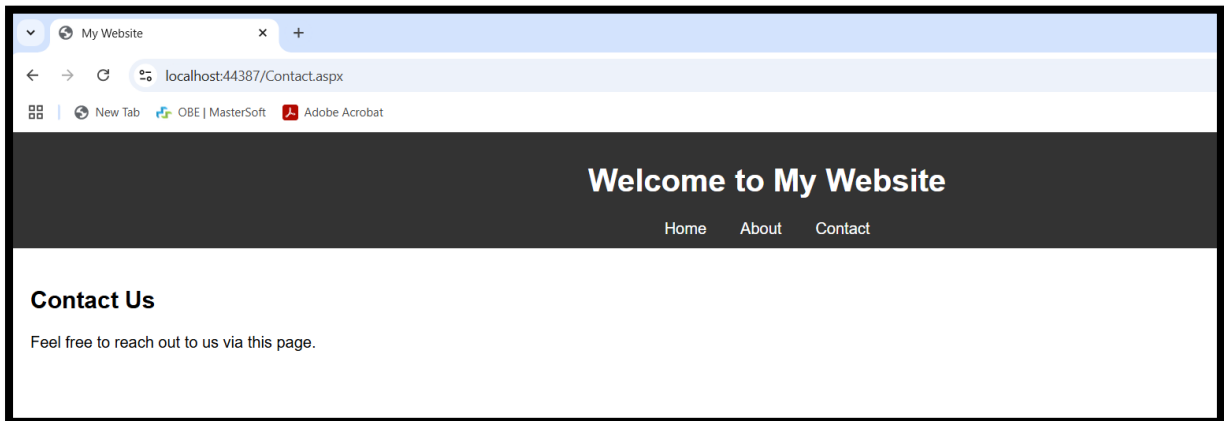
Contact.aspx

```
<% @ Page Title="Contact" L
```

```
anguage="C#" MasterPageFile="~/SiteMaster.master"
AutoEventWireup="true" CodeFile="Contact.aspx.cs"
Inherits="Contact" %>
```



```
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent"
runat="server">
    <h2>Contact Us</h2>
    <p>Feel free to reach out to us via this page.</p>
</asp:Content>
```



Step 4: Add CSS for Styling

1. Right-click the project in **Solution Explorer**.
2. Add a new folder named Content and a new file styles.css inside it.

styles.css

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}
```

```
.header {
    background-color: #333;
    color: white;
    padding: 10px;
    text-align: center;
```

```
}

.header nav a {
    color: white;
    text-decoration: none;
    margin: 0 15px;
}

.header nav a:hover {
    text-decoration: underline;
}

.content {
    padding: 20px;
}

.footer {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 10px;
    position: fixed;
    width: 100%;
    bottom: 0;
}
```

Step 5: Run the Application

- Press **F5** to run the application.
 - Navigate between the Home.aspx, About.aspx, and Contact.aspx pages. Each page will have a consistent layout inherited from SiteMaster.master.
-

Key Advantages of Using Master Pages

1. **Consistency:** Provides a unified layout for all pages.

2. **Maintainability:** Changes to the master page automatically reflect on all linked pages.
3. **Code Reusability:** Common elements like navigation bars and footers are defined only once.

Practical Number 3: Design a Web Application using advanced controls.

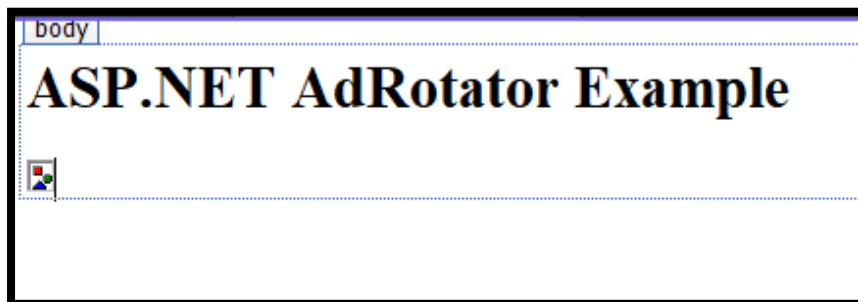
1. AdRotator Control in ASP.NET

The **AdRotator** control displays rotating advertisements based on an XML file.

Steps to Run:

1. Open Visual Studio and create an **ASP.NET Web Application**.
2. Add a new **Web Form** (AdRotator.aspx).
3. Create an XML file (Ads.xml) in the project.
4. Place some ad images in the Images folder.

AdRotator.aspx

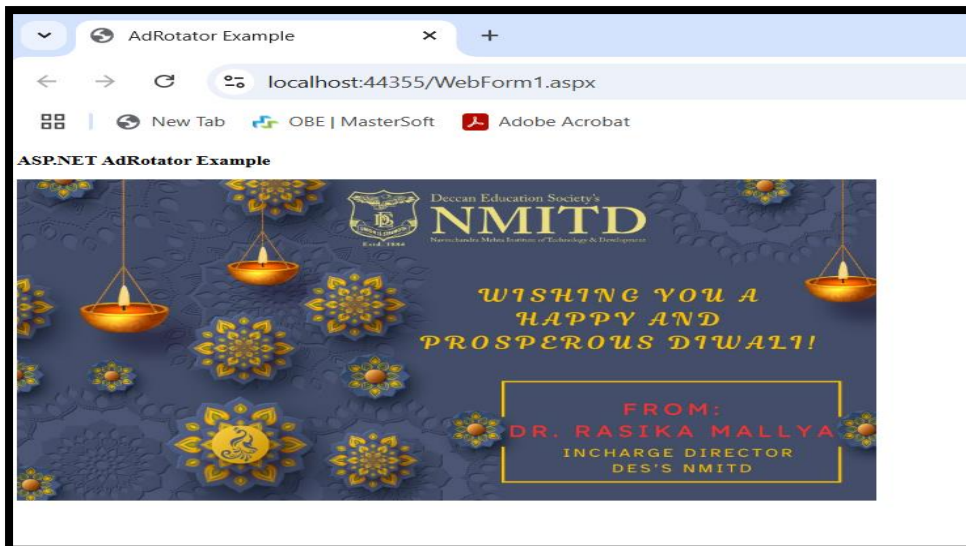


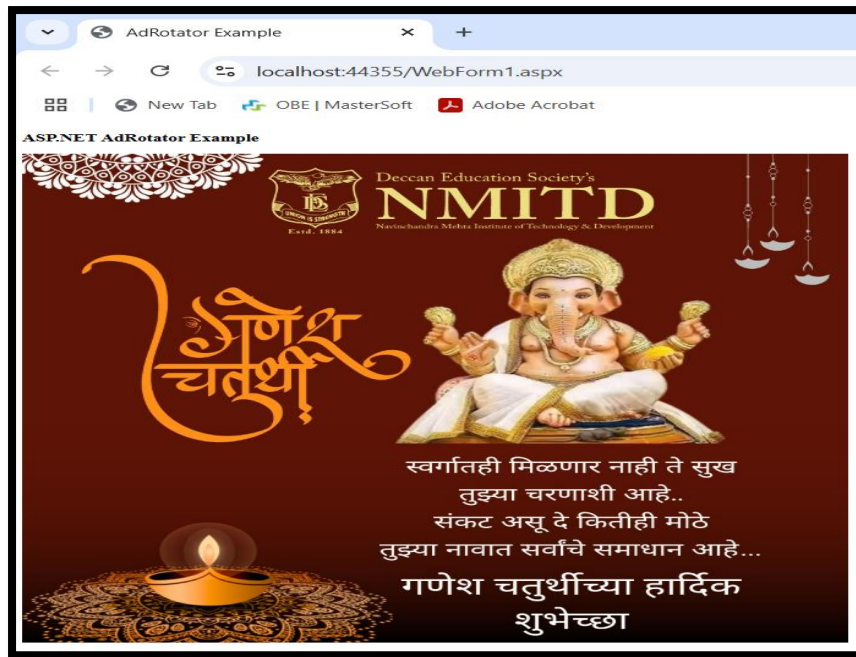
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="Practical_Number_3.WebForm1" %>

<!DOCTYPE html>
<html>
<head>
    <title>AdRotator Example</title>
</head>
<body>
    <h2>ASP.NET AdRotator Example</h2>
    <asp:AdRotator ID="AdRotator1" runat="server" AdvertisementFile="~/Ads.xml"
/>
</body>
</html>
```

Ads.xml (Advertisement File)

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements>
  <Ad>
    <ImageUrl>Images/ad1.jpg</ImageUrl>
    <NavigateUrl>https://www.example2.com</NavigateUrl>
    <AlternateText>First Ad</AlternateText>
    <Impressions>50</Impressions>
  </Ad>
  <Ad>
    <ImageUrl>Images/ad2.jpg</ImageUrl>
    <NavigateUrl>https://www.example1.com</NavigateUrl>
    <AlternateText>Second Ad</AlternateText>
    <Impressions>30</Impressions>
  </Ad>
</Advertisements>
```





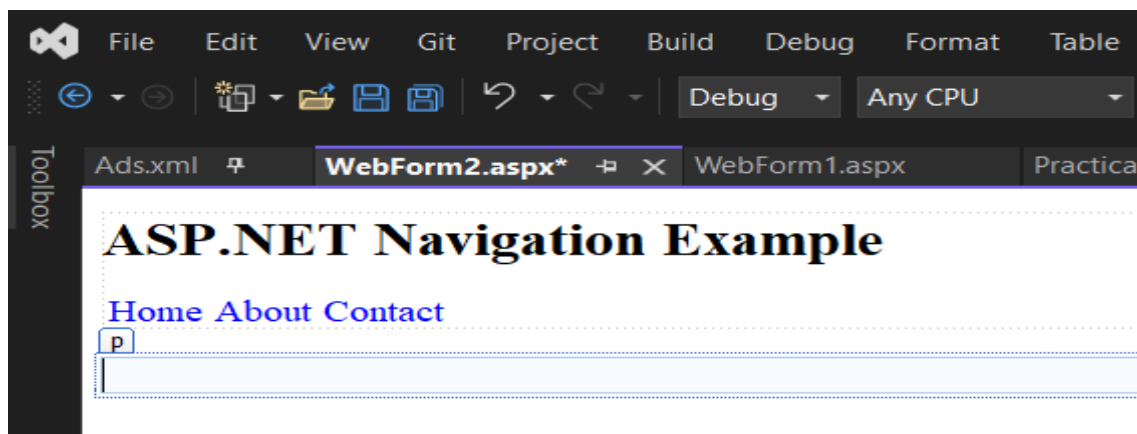
2. Navigation Control (Menu Navigation)

This example demonstrates how to use a **Menu control** for site navigation.

Steps to Run:

1. Create a new Web Form (Navigation.aspx).
2. Use the Menu control inside an asp:SiteMapDataSource.

Navigation.aspx



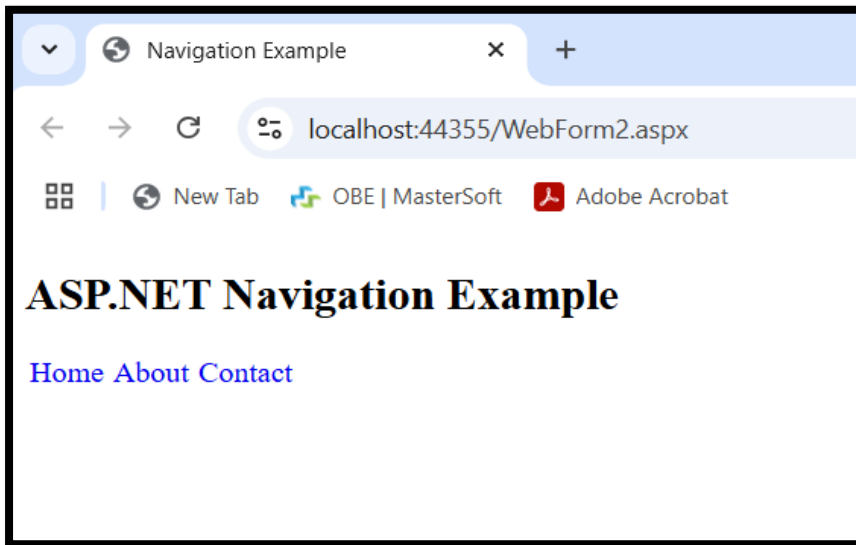
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm2.aspx.cs"
Inherits="Practical_Number_3.WebForm2" %>
```

```
<!DOCTYPE html>
<html>
```

```

<head runat="server">
<title>Navigation Example</title>
</head>
<body>
  <form id="form1" runat="server">
    <h2>ASP.NET Navigation Example</h2>
    <asp:Menu ID="Menu1" runat="server" Orientation="Horizontal">
      <Items>
        <asp:MenuItem Text="Home" NavigateUrl="Home.aspx"/>
        <asp:MenuItem Text="About" NavigateUrl="About.aspx"/>
        <asp:MenuItem Text="Contact" NavigateUrl="Contact.aspx"/>
      </Items>
    </asp:Menu>
  </form>
</body>
</html>

```



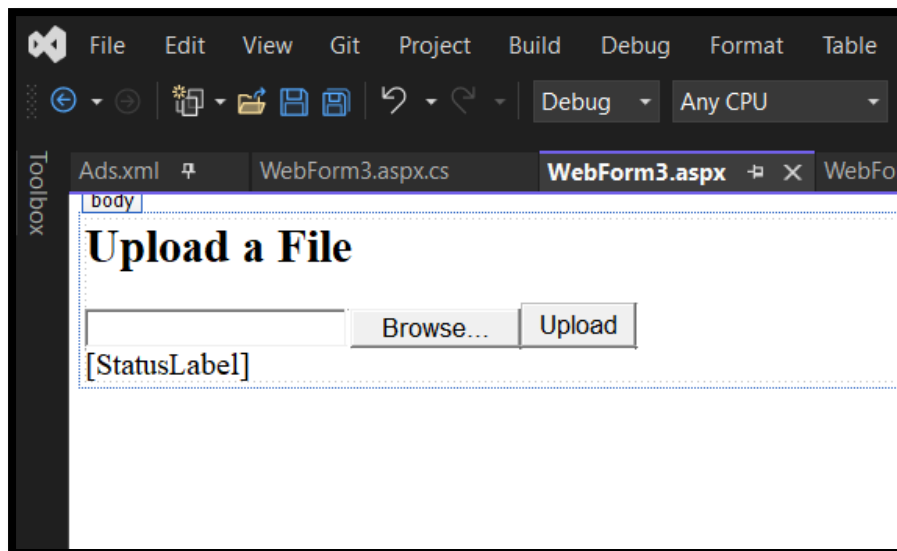
3. File Upload in ASP.NET

This example allows users to upload a file to the server.

Steps to Run:

1. Create a Web Form (FileUpload.aspx).
2. Implement the file upload functionality in C#.

FileUpload.aspx



```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm3.aspx.cs"
Inherits="Practical_Number_3.WebForm3" %>
```

```
<!DOCTYPE html>
<html>
<head runat="server">
    <title>File Upload Example</title>
</head>
<body>
    <form id="form1" runat="server">
        <h2>Upload a File</h2>
        <asp:FileUpload ID="FileUploadControl" runat="server" />
        <asp:Button ID="UploadButton" runat="server" Text="Upload"
OnClick="UploadButton_Click" />
        <br />
        <asp:Label ID="StatusLabel" runat="server" Text=""></asp:Label>
    </form>
</body>
</html>
```

FileUpload.aspx.cs

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

namespace Practical_Number_3
{
    public partial class WebForm3 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

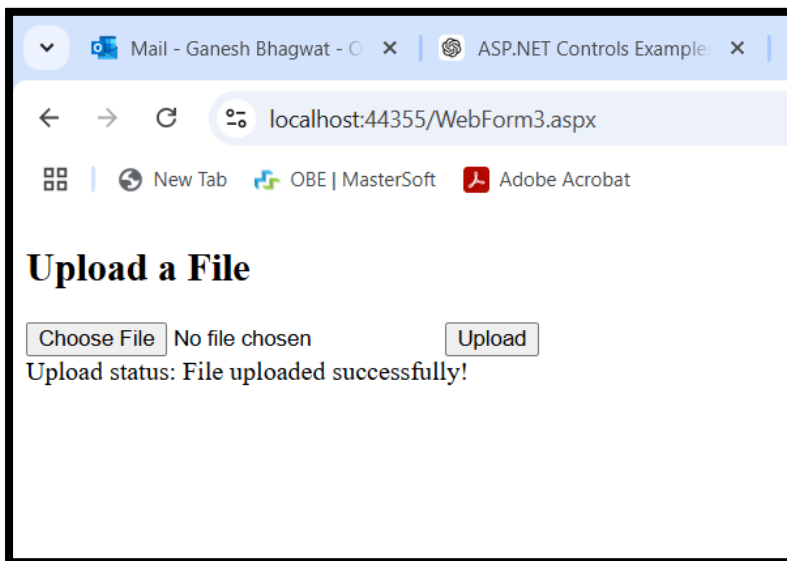
        }

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

```

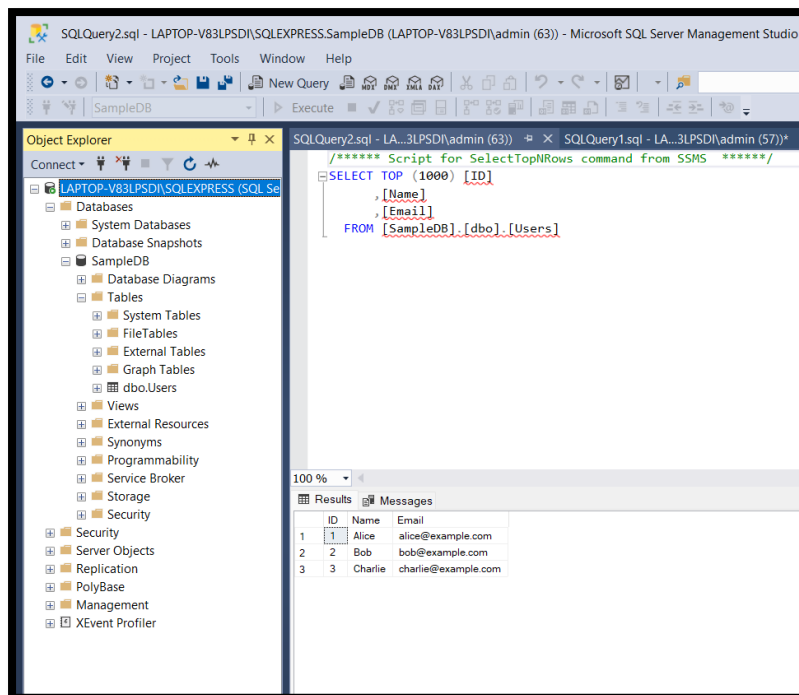
    {
        if (FileUploadControl.HasFile)
        {
            try
            {
                string filename =
Path.GetFileName(FileUploadControl.FileName);
                FileUploadControl.SaveAs(Server.MapPath("~/Uploads/") +
filename);
                StatusLabel.Text = "Upload status: File uploaded
successfully!";
            }
            catch (Exception ex)
            {
                StatusLabel.Text = "Upload status: Error - " + ex.Message;
            }
        }
        else
        {
            StatusLabel.Text = "Upload status: No file selected.";
        }
    }
}
}

```



Practical Number 4: Webpage Demonstrating Connection-Oriented Architecture (ASP.NET Web Forms with SQL Server Database)

A **connection-oriented architecture** involves establishing a persistent connection between the client and server. This is typically demonstrated using **ADO.NET with SQL Server**, where a connection is opened, data is fetched, and then the connection is closed.



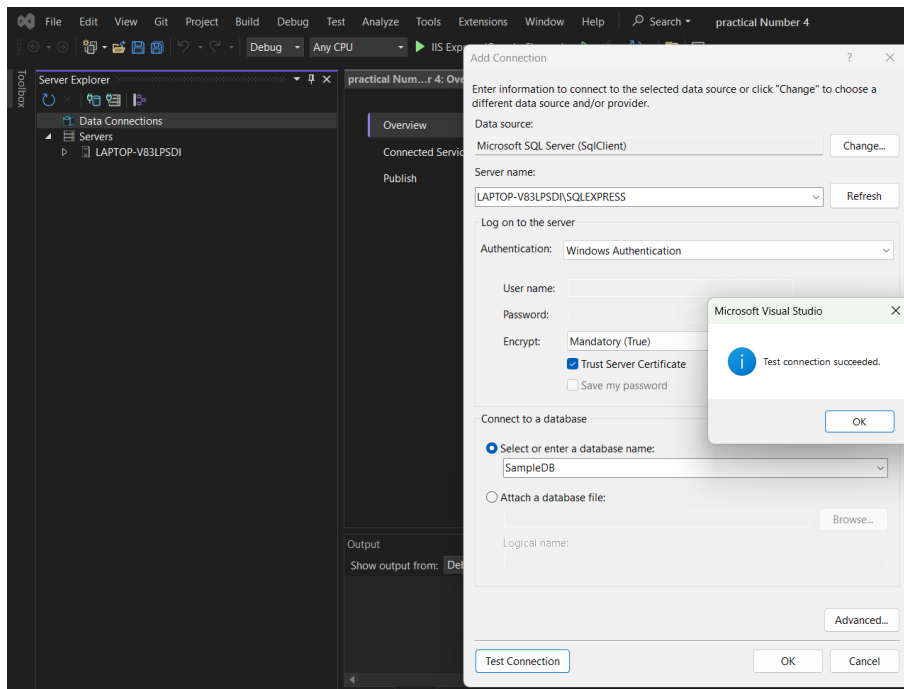
✦ Step 1: Create SQL Server Table

Run the following SQL script in **SQL Server Management Studio (SSMS)**:

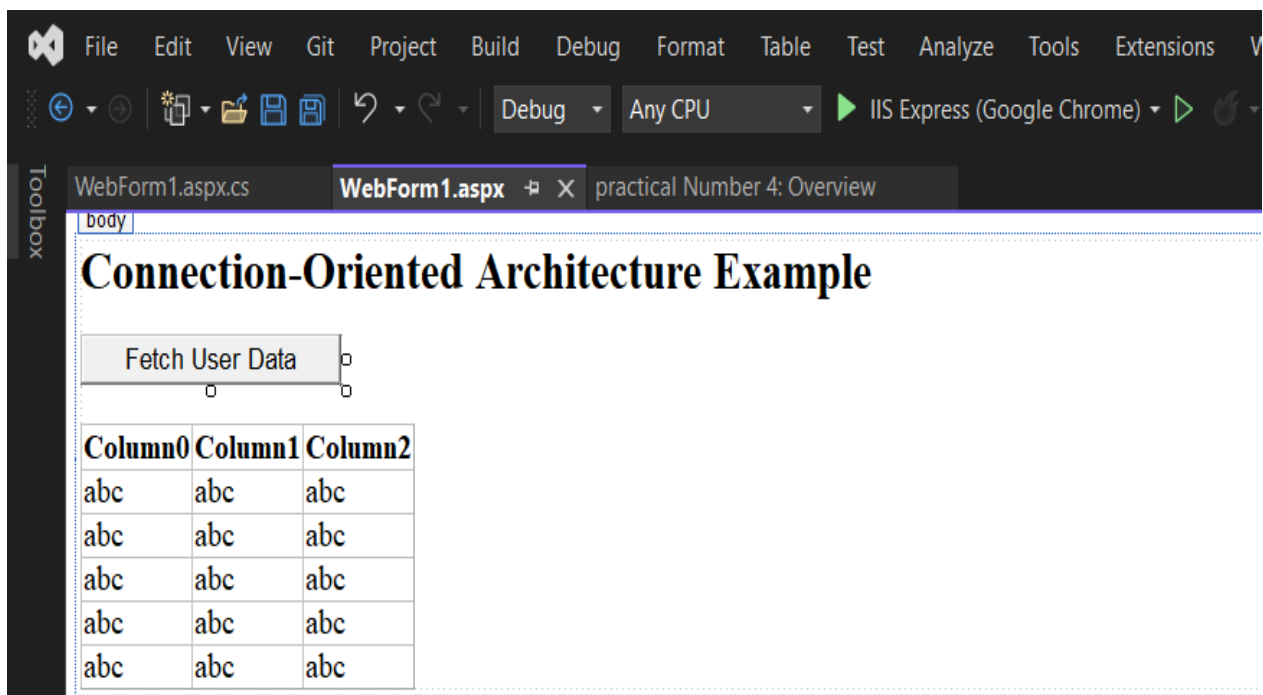
```
CREATE DATABASE SampleDB;  
USE SampleDB;
```

```
CREATE TABLE Users (  
    ID INT IDENTITY(1,1) PRIMARY KEY,  
    Name NVARCHAR(50),  
    Email NVARCHAR(100)  
);
```

```
INSERT INTO Users (Name, Email) VALUES  
( 'Alice', 'alice@example.com'),  
( 'Bob', 'bob@example.com'),  
( 'Charlie', 'charlie@example.com');
```



✦ Step 2: Design web form (Frontend UI)



```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="practical_Number_4.WebForm1" %>
```

```
<!DOCTYPE html>
<html>
<head runat="server">
    <title>Connection-Oriented Architecture</title>
</head>
<body>
    <form id="form1" runat="server">
```

```

<h2>Connection-Oriented Architecture Example</h2>

<asp:Button ID="FetchDataButton" runat="server" Text="Fetch User Data"
OnClick="FetchDataButton_Click" />
<br /><br />

<asp:GridView ID="UsersGridView" runat="server"
AutoGenerateColumns="true" BorderWidth="1" />

</form>
</body>
</html>

```

✦ Step 3: Code-Behind

```

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

namespace practical_Number_4
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void FetchDataButton_Click(object sender, EventArgs e)
        {
            // Define the connection string (Update with your server details)
            string connectionString = "Data Source=LAPTOP-
V83LPSDI\\SQLEXPRESS;Initial Catalog=SampleDB;Integrated Security=True";

            // Create a connection object
            using (SqlConnection conn = new SqlConnection(connectionString))
            {
                try
                {
                    conn.Open(); // Open the connection

                    // SQL query to fetch data
                    string query = "SELECT * FROM Users";
                    SqlDataAdapter da = new SqlDataAdapter(query, conn);
                    DataTable dt = new DataTable();
                    da.Fill(dt);

                    // Bind data to GridView
                    UsersGridView.DataSource = dt;
                    UsersGridView.DataBind();
                }
                catch (Exception ex)
                {
                    Response.Write("<script>alert('Error: " + ex.Message +
"');</script>");
                }
            }
        }
    }
}

```

```
        }  
    } // Connection closes au  
}  
}
```



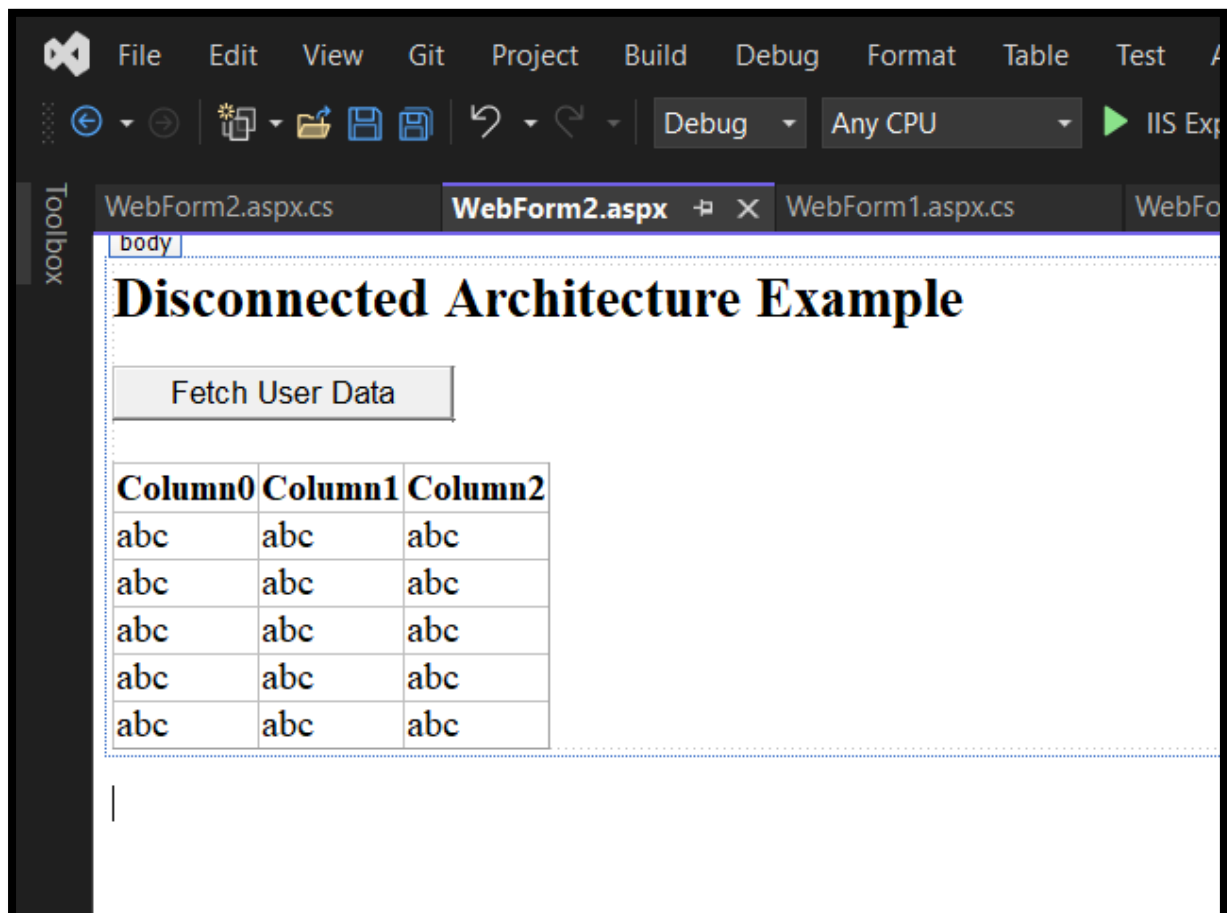
Practical Number 5: Webpage Demonstrating Disconnected Architecture (ASP.NET Web Forms with SQL Server Database)

✦ Step 1: Create SQL Server Table

Run the following SQL script in **SQL Server Management Studio (SSMS)**:

```
CREATE DATABASE SampleDB;  
USE SampleDB;  
  
CREATE TABLE Users (  
    ID INT IDENTITY(1,1) PRIMARY KEY,  
    Name NVARCHAR(50),  
    Email NVARCHAR(100)  
);  
  
INSERT INTO Users (Name, Email) VALUES  
('Alice', 'alice@example.com'),  
('Bob', 'bob@example.com'),  
('Charlie', 'charlie@example.com');
```

✦ Step 2: Design a web page(Frontend UI)



```

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm2.aspx.cs"
Inherits="practical_Number_4.WebForm2" %>

<!DOCTYPE html>
<html>
<head runat="server">
    <title>Disconnected Architecture Example</title>
</head>
<body>
    <form id="form1" runat="server">
        <h2>Disconnected Architecture Example</h2>

        <asp:Button ID="FetchDataButton" runat="server" Text="Fetch User Data"
OnClick="FetchDataButton_Click" />
        <br /><br />

        <asp:GridView ID="UsersGridView" runat="server"
AutoGenerateColumns="true" BorderWidth="1" />

    </form>
</body>
</html>

```

✦ Step 3: Code-Behind

```

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

namespace practical_Number_4
{
    public partial class WebForm2 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void FetchDataButton_Click(object sender, EventArgs e)
        {
            // Define the connection string (Update with your server details)
            string connectionString = "Data Source=LAPTOP-
V83LP5DI\\SQLEXPRESS;Initial Catalog=SampleDB;Integrated Security=True";

            // Create objects for disconnected architecture
            SqlDataAdapter da;
            DataSet ds = new DataSet();

            try
            {
                using (SqlConnection conn = new SqlConnection(connectionString))
                {
                    // SQL query to fetch data
                    string query = "SELECT * FROM Users";
                    da = new SqlDataAdapter(query, conn);
                }
            }
        }
    }
}

```

```

        // Fill dataset with data from the database
        da.Fill(ds, "Users");
    } // Connection is closed after this block

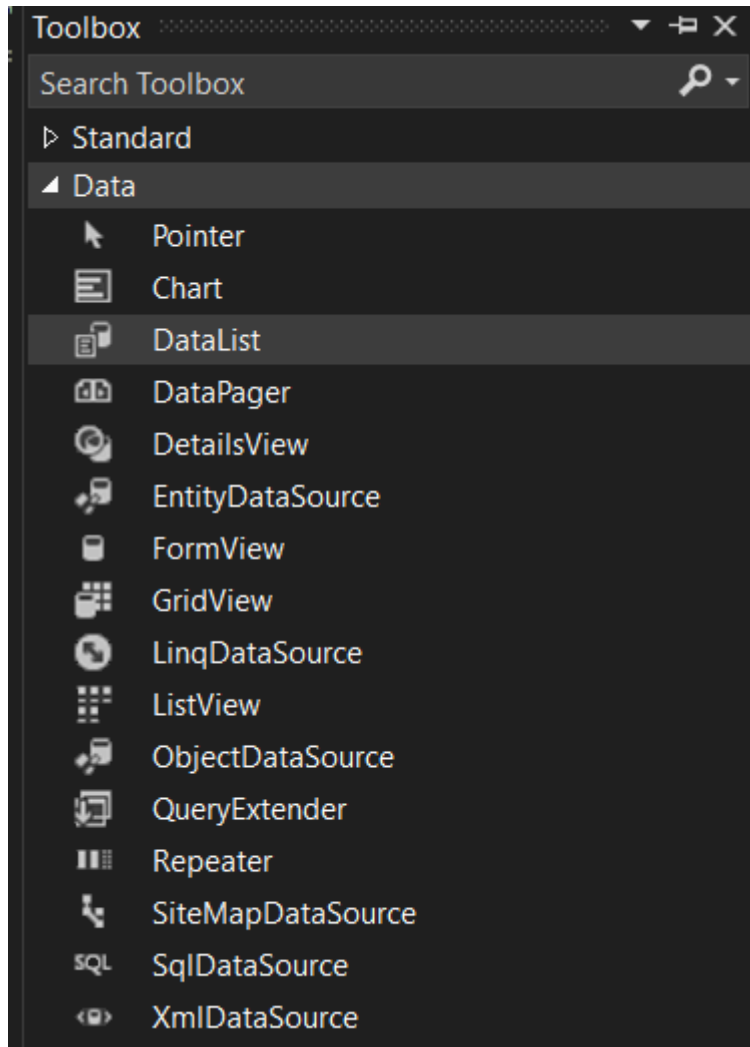
    // Bind data to GridView (data remains in memory)
    UsersGridView.DataSource = ds.Tables["Users"];
    UsersGridView.DataBind();
}
catch (Exception ex)
{
    Response.Write("<script>alert('Error: " + ex.Message +
"');</script>");
}
}
}
}

```



Practical Number 6: Create a webpage that demonstrates the use of data bound controls of ASP.NET.

Data Controls:



DataGridView:

	<u>Databound Col0</u>	<u>Databound Col1</u>	<u>Databound Col2</u>
<u>Select</u>	abc	0	abc
<u>Select</u>	abc	1	abc
<u>Select</u>	abc	2	abc
<u>Select</u>	abc	3	abc
<u>Select</u>	abc	4	abc
<u>Select</u>	abc	5	abc
<u>Select</u>	abc	6	abc
<u>Select</u>	abc	7	abc
<u>Select</u>	abc	8	abc
<u>Select</u>	abc	9	abc
1 2			
SqlDataSource - SqlDataSource1			

localhost:44321/WebForm1.asp
localhost:44321/WebForm1.aspx
New Tab OBE MasterSoft Adobe Acrobat

	<u>EmpID</u>	<u>Name</u>	<u>Department</u>	<u>Salary.</u>
<u>Select</u>	1	John Doe	IT222	60000.00
<u>Select</u>	2	Jane Smith	HR	55000.00
<u>Select</u>	3	hbd	trt	454.00

Practical 7: Design a webpage to demonstrate the working of a simple stored procedure.

The webpage will use **ASP.NET Web Forms** and **SQL Server** to retrieve and display user details.

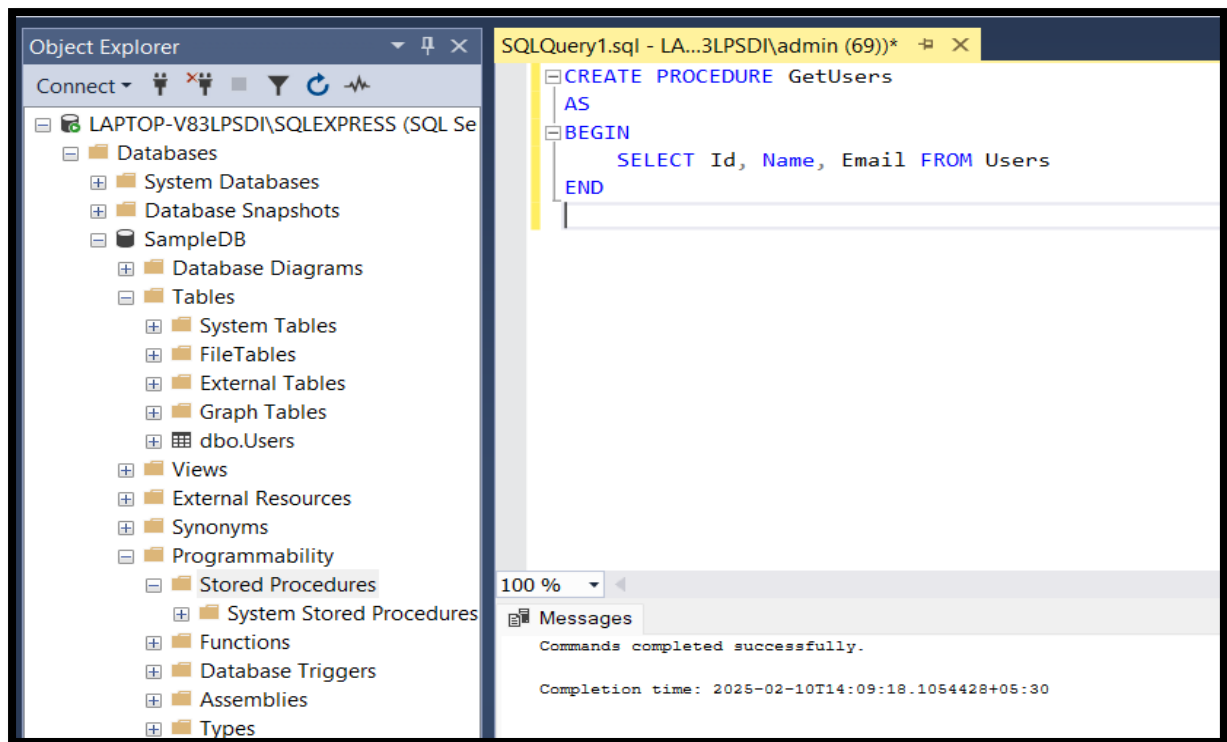
Steps to Implement:

1. **Create a Stored Procedure in SQL Server**
 2. **Design an ASP.NET Web Form (ASPX Page)**
 3. **Connect to the Database and Execute the Stored Procedure**
 4. **Display the Results in a GridView**
-

1. Create the Stored Procedure in SQL Server

Run this SQL script in **SQL Server Management Studio (SSMS)**:

```
CREATE PROCEDURE GetUsers
AS
BEGIN
    SELECT Id, Name, Email FROM Users
END
```

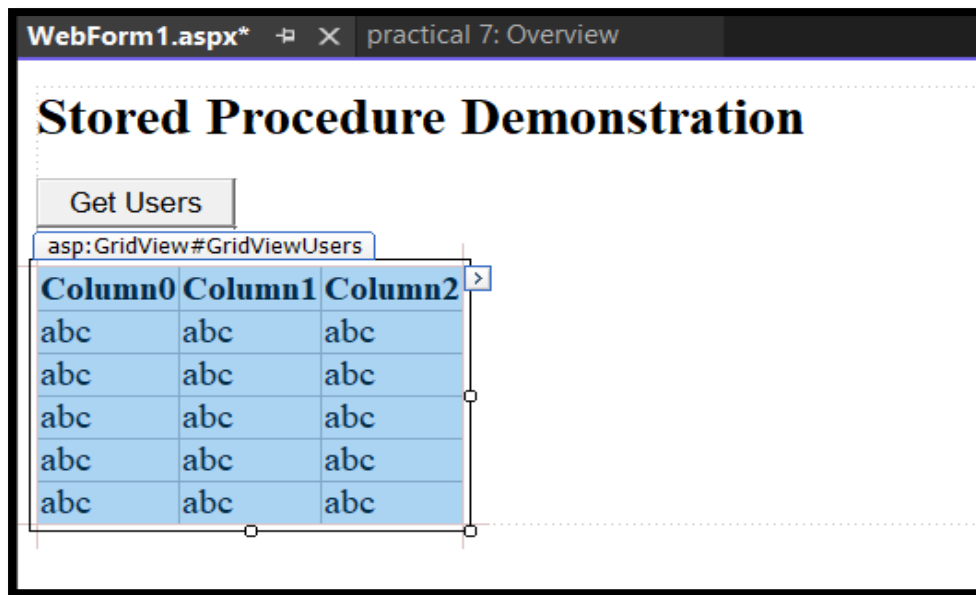


2. Create an ASP.NET Web Application in Visual Studio

- Open **Visual Studio**
- Create a new **ASP.NET Web Forms Application**
- Add a **Web Form (Default.aspx)**

3. Design the Web Form (Default.aspx)

Modify the `WebForm1.aspx` file:



```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="practical_7.WebForm1" %>
```

```
<!DOCTYPE html>
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Stored Procedure Demo</title>
```

```
</head>
```

```
<body>
```

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

```
<h2>Stored Procedure Demonstration</h2>
```

```
<asp:Button ID="btnGetUsers" runat="server" Text="Get Users"
OnClick="btnGetUsers_Click" />
```

```
<br /><br />
```

```
<asp:GridView ID="GridViewUsers" runat="server"
AutoGenerateColumns="true" BorderColor="Black" BorderWidth="1px" />
```

```
</form>
```

```
</body>
```

```
</html>
```

4. Code-Behind File (Default.aspx.cs)

Modify `Default.aspx.cs` to execute the stored procedure and display results in `GridView`:

```
using System;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Web.UI;
```

```
namespace practical_7
{
```

```

public partial class WebForm1 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void btnGetUsers_Click(object sender, EventArgs e)
    {
        // Connection string from Web.config
        string connStr = "Data Source=LAPTOP-V83LPSDI\\SQLEXPRESS;Initial
Catalog=SampleDB;Integrated Security=True";

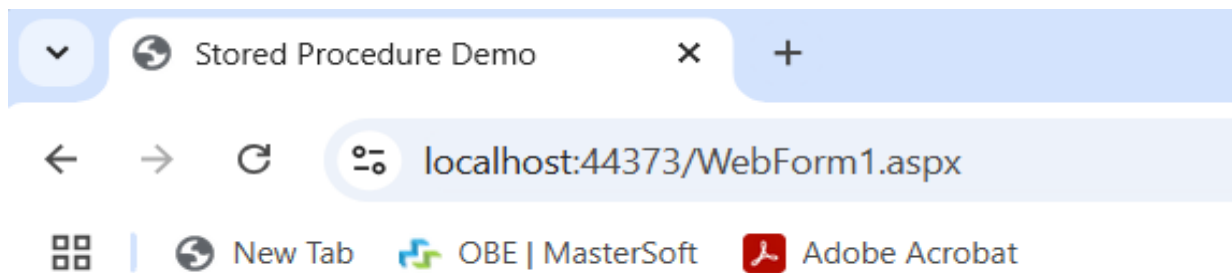
        using (SqlConnection conn = new SqlConnection(connStr))
        {
            using (SqlCommand cmd = new SqlCommand("GetUsers", conn))
            {
                cmd.CommandType = CommandType.StoredProcedure;
                conn.Open();

                SqlDataAdapter da = new SqlDataAdapter(cmd);
                DataTable dt = new DataTable();
                da.Fill(dt);

                GridViewUsers.DataSource = dt;
                GridViewUsers.DataBind();
            }
        }
    }
}

```

OUTPUT:



Stored Procedure Demonstration

Get Users

Id	Name	Email
1	Alice	alice@example.com
2	Bob	bob@example.com
3	Charlie	charlie@example.com

Practical 8: Design a webpage to demonstrate the working of parameterized stored procedure.

✦ Steps to Implement

1. **Create a SQL Server Database** with a stored procedure.
 2. **Design an ASP.NET Web Form** using Visual Studio.
 3. **Use ADO.NET** to call the stored procedure from C#.
 4. **Display the output on the webpage.**
-

1 SQL Server: Create a Database & Stored Procedure

Run the following SQL commands in **SQL Server Management Studio (SSMS)**:

```
CREATE DATABASE EmployeeDB;
USE EmployeeDB;

-- Create Table
CREATE TABLE Employees (
    EmpID INT PRIMARY KEY IDENTITY(1,1),
```

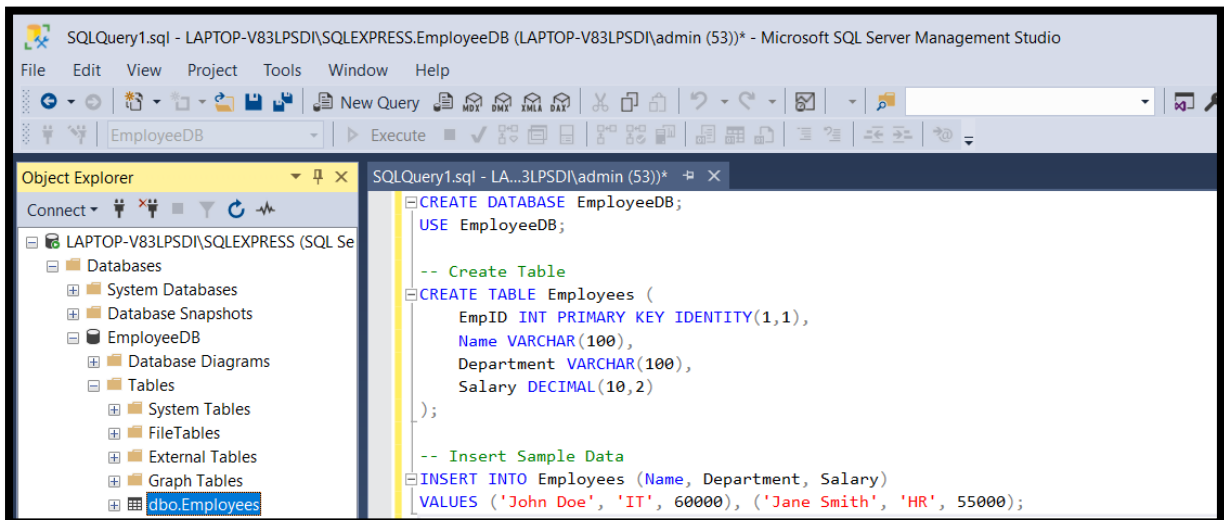
```

        Name VARCHAR(100),
        Department VARCHAR(100),
        Salary DECIMAL(10,2)
    );

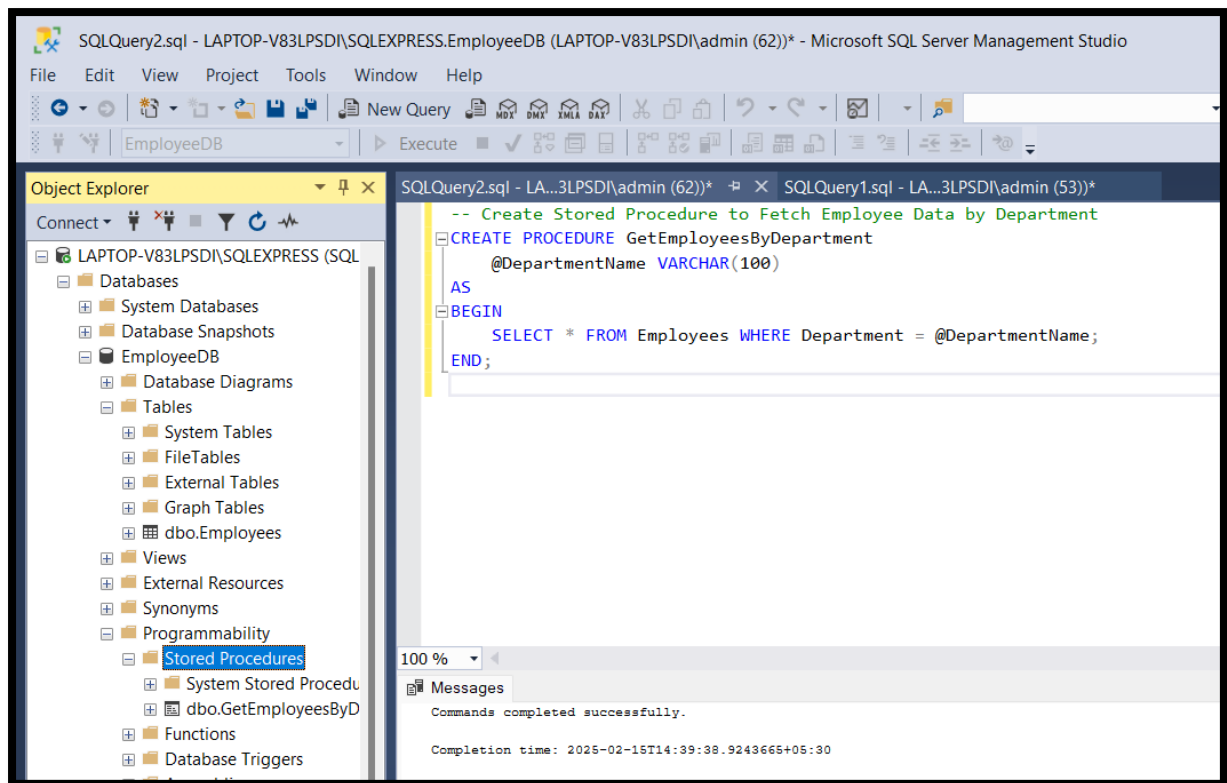
-- Insert Sample Data
INSERT INTO Employees (Name, Department, Salary)
VALUES ('John Doe', 'IT', 60000), ('Jane Smith', 'HR', 55000);

-- Create Stored Procedure to Fetch Employee Data by Department
CREATE PROCEDURE GetEmployeesByDepartment
    @DepartmentName VARCHAR(100)
AS
BEGIN
    SELECT * FROM Employees WHERE Department = @DepartmentName;
END;

```



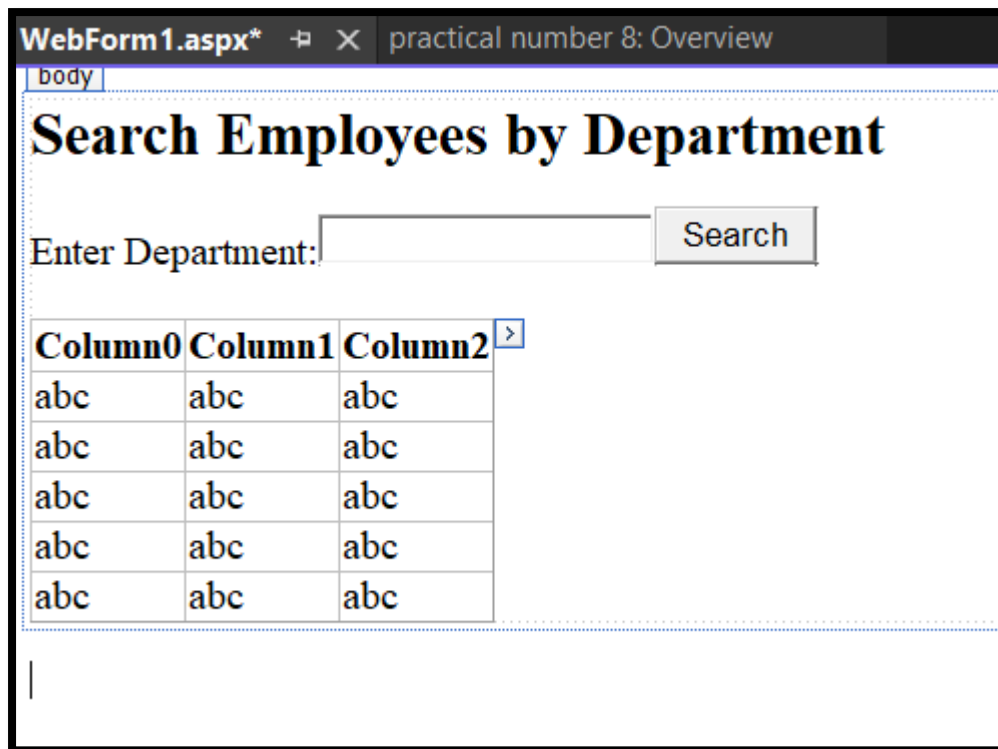
Results		Messages		
	EmpID	Name	Department	Salary
1	1	John Doe	IT	60000.00
2	2	Jane Smith	HR	55000.00



2 Create ASP.NET Web Form

◆ Design a Web Form (WebForm1.aspx)

Go to **Visual Studio** → **ASP.NET Web Forms Application** and modify `WebForm1.aspx` as follows:



```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="practical_number_8.WebForm1" %>
```

```
<!DOCTYPE html>
<html lang="en">
<head runat="server">
    <title>Stored Procedure Demo</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <h2>Search Employees by Department</h2>
            <asp:Label runat="server" Text="Enter Department:"></asp:Label>
            <asp:TextBox ID="txtDepartment" runat="server"></asp:TextBox>
            <asp:Button ID="btnSearch" runat="server" Text="Search"
OnClick="btnSearch_Click"/>

            <br /><br />
            <asp:GridView ID="gvEmployees" runat="server"
AutoGenerateColumns="True"></asp:GridView>
        </div>
    </form>
</body>
</html>
```

3 Backend Code (Default.aspx.cs)

Modify WebForm1.aspx.cs to call the stored procedure using ADO.NET.

```
using System;
using System.Configuration;
using System.Data;
```

```

using System.Data.SqlClient;
using System.Web.UI;

namespace practical_number_8
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void btnSearch_Click(object sender, EventArgs e)
        {
            string connStr = "Data Source=LAPTOP-
V83LPDI\\SQLEXPRESS;Initial Catalog=EmployeeDB;Integrated Security=True";

            {
                SqlConnection conn = new SqlConnection(connStr);
                SqlCommand cmd = new SqlCommand("GetEmployeesByDepartment",
conn);

                cmd.CommandType = CommandType.StoredProcedure;
                cmd.Parameters.AddWithValue("@DepartmentName",
txtDepartment.Text);

                SqlDataAdapter da = new SqlDataAdapter(cmd);
                DataTable dt = new DataTable();
                da.Fill(dt);

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

            }
        }
    }
}

```

Stored Procedure Demo

localhost:44330/WebForm1.aspx

New Tab OBE | MasterSoft Adobe Acrobat

Search Employees by Department

Enter Department:

Stored Procedure Demo

localhost:44330/WebForm1.aspx

New Tab OBE | MasterSoft Adobe Acrobat

Search Employees by Department

Enter Department:

EmpID	Name	Department	Salary
1	John Doe	IT	60000.00

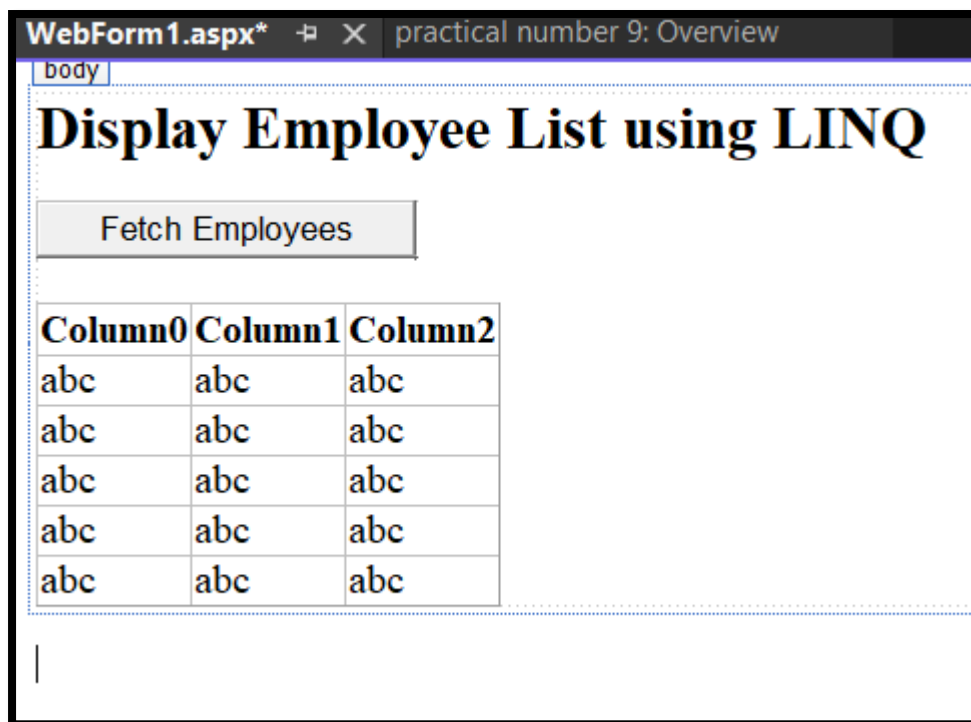
Practical 9: Design a webpage to display the use of LINQ.

✦ Steps to Implement

1. Create an ASP.NET Web Forms project in Visual Studio.
 2. Use LINQ to query a list of employees (In-Memory Collection).
 3. Bind the LINQ results to a GridView for display.
-

1 Create ASP.NET Web Form (WebForm1.aspx)

Modify WebForm1.aspx to include a **Button** to fetch employee data and a **GridView** to display the results.



```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="practical_number_9.WebForm1" %>

<!DOCTYPE html>
<html lang="en">
<head runat="server">
    <title>LINQ Demo</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <h2>Display Employee List using LINQ</h2>
            <asp:Button ID="btnFetchData" runat="server" Text="Fetch Employees"
OnClick="btnFetchData_Click"/>
            <br /><br />
        </div>
    </form>
</body>
</html>
```

```

        <asp:GridView ID="gvEmployees" runat="server"
AutoGenerateColumns="True"></asp:GridView>
    </div>
</form>
</body>
</html>

```

2 Backend Code (Default.aspx.cs)

Modify WebForm1.aspx.cs to use LINQ to query employee data.

```

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

namespace practical_number_9
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        // Define an Employee class
        public class Employee
        {
            public int EmpID { get; set; }
            public string Name { get; set; }
            public string Department { get; set; }
            public decimal Salary { get; set; }
        }

        // Sample Employee Data (In-Memory Collection)
        private List<Employee> employees = new List<Employee>
        {
            new Employee { EmpID = 1, Name = "John Doe", Department = "IT",
Salary = 60000 },
            new Employee { EmpID = 2, Name = "Jane Smith", Department = "HR",
Salary = 55000 },
            new Employee { EmpID = 3, Name = "Mike Johnson", Department = "IT",
Salary = 65000 },
            new Employee { EmpID = 4, Name = "Emily Davis", Department =
"Finance", Salary = 70000 }
        };

        protected void Page_Load(object sender, EventArgs e)
        {

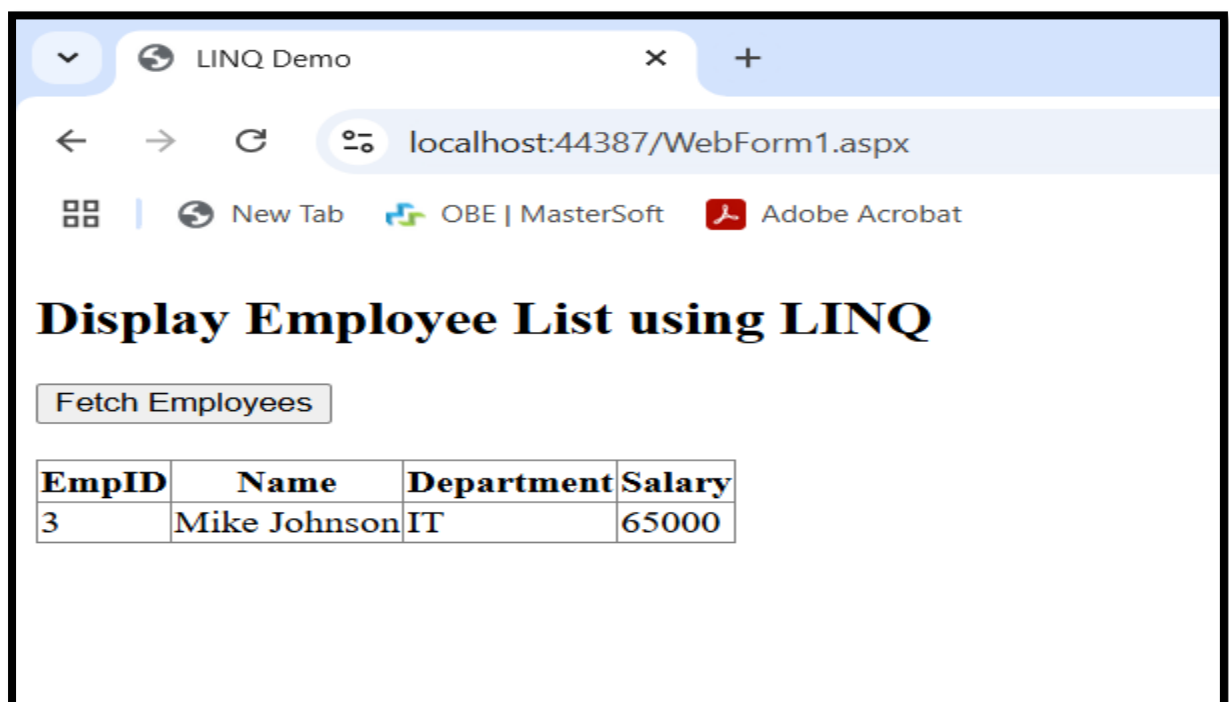
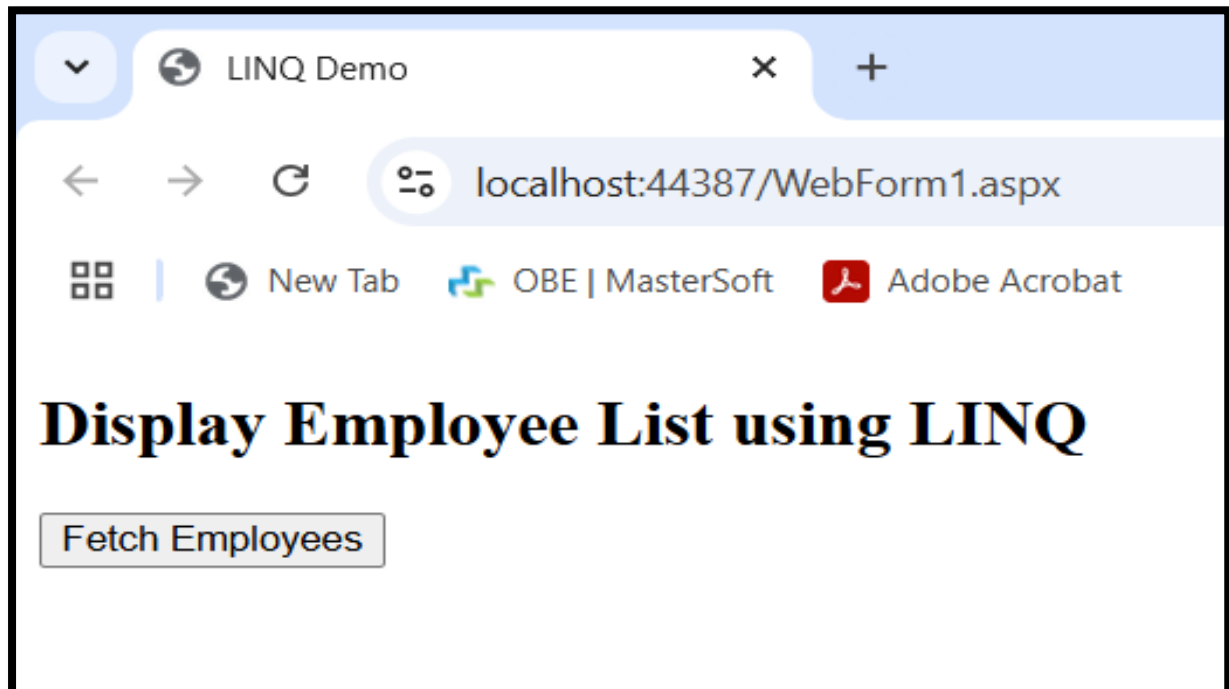
        }

        protected void btnFetchData_Click(object sender, EventArgs e)
        {
            // Use LINQ to fetch IT department employees with salary > 60,000
            var result = from emp in employees
                where emp.Department == "IT" && emp.Salary > 60000
                select emp;

            // Bind data to GridView
            gvEmployees.DataSource = result.ToList();
            gvEmployees.DataBind();
        }
    }
}

```

```
}  
    }  
}
```



Practical 10: Build websites to demonstrate the working of entity frameworks in dot net.

✦ Steps to Implement

1. **Create a SQL Server Database & Table**
 2. **Create an ASP.NET Web Application in Visual Studio**
 3. **Install & Configure Entity Framework (EF) ORM**
 4. **Use EF to perform CRUD operations**
 5. **Display data in GridView & allow users to Add, Edit, Delete records**
-

1 SQL Server: Create a Database & Table

Open **SQL Server Management Studio (SSMS)** and execute:

```
CREATE DATABASE EmployeeDB;
USE EmployeeDB;

CREATE TABLE Employees (
    EmpID INT PRIMARY KEY IDENTITY(1,1),
    Name NVARCHAR(100),
    Department NVARCHAR(100),
    Salary DECIMAL(10,2)
);

INSERT INTO Employees (Name, Department, Salary)
VALUES ('John Doe', 'IT', 60000), ('Jane Smith', 'HR', 55000);
```

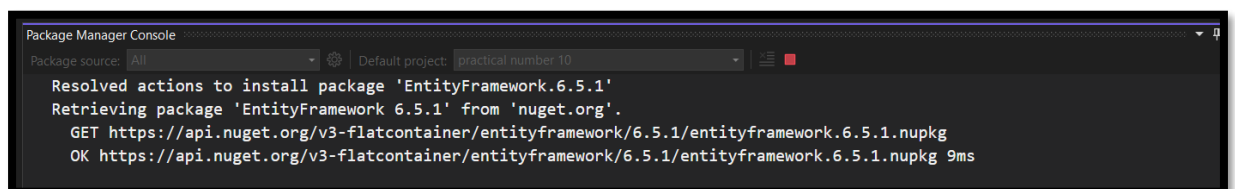
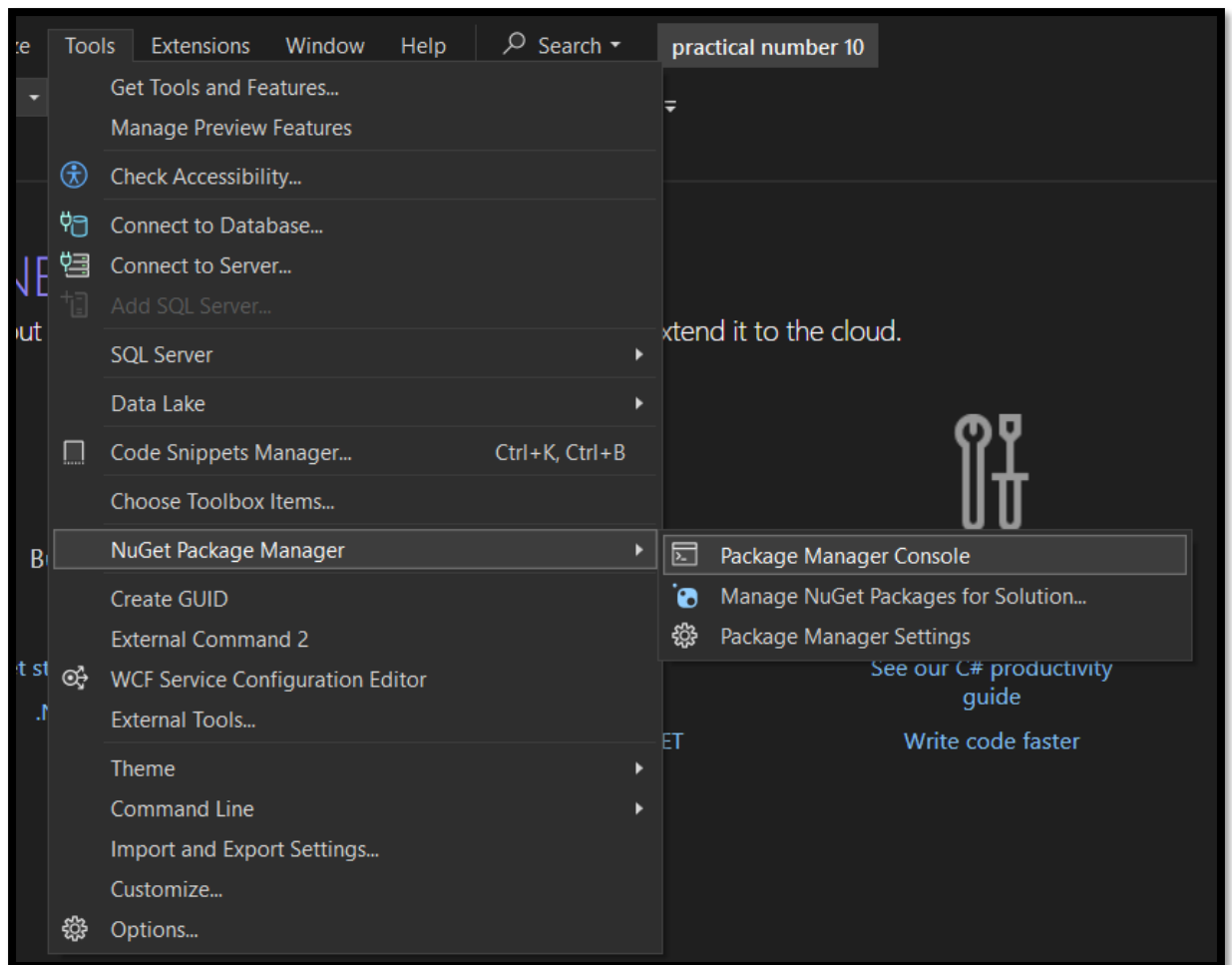
2 Create an ASP.NET Web Application

1. Open **Visual Studio** → Create a **New Project** → Choose **ASP.NET Web Application (.NET Framework)**
 2. Select **Web Forms** and click **Create**
-

3 Install Entity Framework

1. Open **Package Manager Console** (Tools → NuGet Package Manager → Package Manager Console)
2. Run the command:

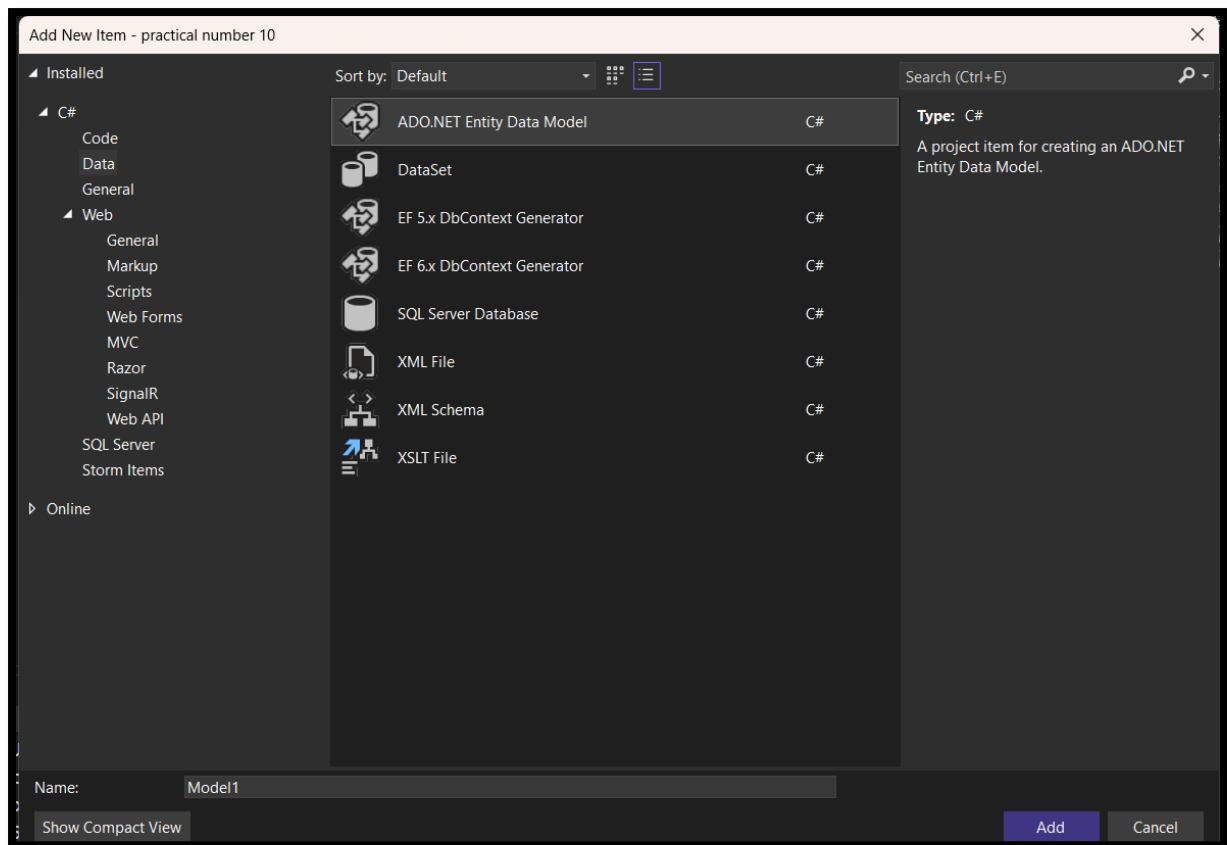
```
Install-Package EntityFramework
```



4 Create Entity Framework Model

1. Right-click the project → Add → New Item → Select **ADO.NET Entity Data Model**
2. Choose **EF Designer from Database**
3. Select **"EmployeeDB"** as the database
4. Select the **Employees** table → Finish

This generates the **Employee.cs model class**.



**Choose Model Contents****What should the model contain?****EF Designer
from
database**Empty EF
Designer
modelEmpty Code
First modelCode First
from
database

Creates a model in the EF Designer based on an existing database. You can choose the database connection, settings for the model, and database objects to include in the model. The classes your application will interact with are generated from the model.

< Previous

Next >

Finish

Cancel



Choose Your Data Connection

Which data connection should your application use to connect to the database?

laptop-v83lpsdi\sqlexpress.EmployeeDB.dbo



New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

- ☐ No, exclude sensitive data from the connection string. I will set it in my application code.
- ☐ Yes, include the sensitive data in the connection string.

Connection string:

```
metadata=res://*/Employee.csdl|res://*/Employee.ssdl|
res://*/Employee.msl;provider=System.Data.SqlClient;provider connection string="data
source=LAPTOP-V83LPSDI\SQLEXPRESS;initial catalog=EmployeeDB;integrated
security=True;trustservercertificate=True;MultipleActiveResultSets=True;App=EntityFramework"
```

☒ Save connection settings in Web.Config as:

EmployeeDBEntities

< Previous

Next >

Finish

Cancel

**Choose Your Database Objects and Settings**

Which database objects do you want to include in your model?

- ✓ ☒ Tables
 - > ☒ dbo
 - ☐ Views
 - > ☐ Stored Procedures and Functions

- ☒ Pluralize or singularize generated object names
- ☒ Include foreign key columns in the model
- ☒ Import selected stored procedures and functions into the entity model

Model Namespace:

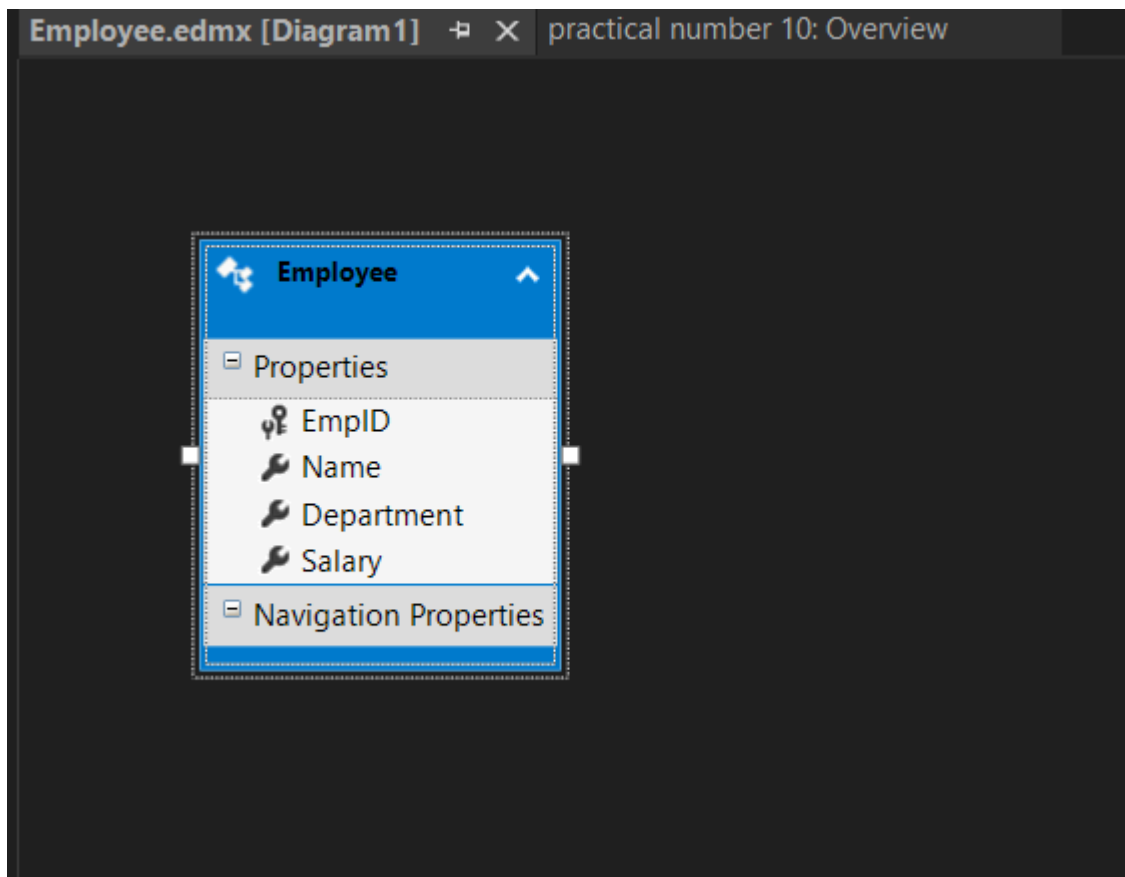
EmployeeDBModel

< Previous

Next >

Finish

Cancel



5 Create an ASP.NET Web Form (WebForm1.aspx)

Modify WebForm1.aspx to include a **GridView** to display records and a **Form** for adding new employees.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="practical_number_10.WebForm1" %>

<!DOCTYPE html>
<html lang="en">
<head runat="server">
    <title>Entity Framework CRUD Demo</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <h2>Employee Management (Entity Framework)</h2>

            <!-- Add Employee Form -->
            <asp:Label runat="server" Text="Name:"></asp:Label>
            <asp:TextBox ID="txtName" runat="server" style="margin-left:
46px"></asp:TextBox>

            <br />

            <asp:Label runat="server" Text="Department:"></asp:Label>
            <asp:TextBox ID="txtDepartment" runat="server"></asp:TextBox> <br />
```

```

        <asp:Label runat="server" Text="Salary:"></asp:Label>
        <asp:TextBox ID="txtSalary" runat="server" style="margin-left:
41px"></asp:TextBox>
        <br />
        <br />
        <br />

        <asp:Button ID="btnAdd" runat="server" Text="Add Employee"
OnClick="btnAdd_Click" />

        <br /><br />

        <!-- Display Employees -->
        <asp:GridView ID="gvEmployees" runat="server"
AutoGenerateColumns="False" DataKeyNames="EmpID"
        OnRowEditing="gvEmployees_RowEditing"
OnRowUpdating="gvEmployees_RowUpdating"
        OnRowCancelingEdit="gvEmployees_RowCancelingEdit"
OnRowDeleting="gvEmployees_RowDeleting" Height="233px" Width="677px">
            <Columns>
                <asp:BoundField DataField="EmpID" HeaderText="EmpID"
ReadOnly="True" />
                <asp:BoundField DataField="Name" HeaderText="Name" />
                <asp:BoundField DataField="Department"
HeaderText="Department" />
                <asp:BoundField DataField="Salary" HeaderText="Salary" />

                <asp:CommandField ShowEditButton="True"
ShowDeleteButton="True" />
            </Columns>
        </asp:GridView>
    </div>
</form>
</body>
</html>

```

WebForm1.aspx* Employee.edmx [Diagram1] practical number 10: Overview

body

Employee Management (Entity Framework)

Name:

Department:

Salary:

EmpID	Name	Department	Salary	
Databound	Databound	Databound	Databound	Edit Delete
Databound	Databound	Databound	Databound	Edit Delete
Databound	Databound	Databound	Databound	Edit Delete
Databound	Databound	Databound	Databound	Edit Delete
Databound	Databound	Databound	Databound	Edit Delete

6 Backend Code (WebForm1.aspx.cs)

Modify WebForm1.aspx.cs to implement CRUD operations using **Entity Framework**.

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

```
namespace practical_number_10
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        EmployeeDBEntities1 db = new EmployeeDBEntities1 ();
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                LoadEmployees();
            }
        }
        private void LoadEmployees()
        {
            gvEmployees.DataSource = db.Employees.ToList();
            gvEmployees.DataBind();
        }

        // Add Employee
        protected void btnAdd_Click(object sender, EventArgs e)
```

```

{
    Employee emp = new Employee
    {
        Name = txtName.Text,
        Department = txtDepartment.Text,
        Salary = Convert.ToDecimal(txtSalary.Text)
    };

    db.Employees.Add(emp);
    db.SaveChanges();
    LoadEmployees();
}

// Edit Employee
protected void gvEmployees_RowEditing(object sender,
GridViewEditEventArgs e)
{
    gvEmployees.EditIndex = e.NewEditIndex;
    LoadEmployees();
}

// Update Employee
protected void gvEmployees_RowUpdating(object sender,
GridViewUpdateEventArgs e)
{
    int empID = Convert.ToInt32(gvEmployees.DataKeys[e.RowIndex].Value);
    Employee emp = db.Employees.Find(empID);

    TextBox txtName =
    (TextBox)gvEmployees.Rows[e.RowIndex].Cells[1].Controls[0];
    TextBox txtDepartment =
    (TextBox)gvEmployees.Rows[e.RowIndex].Cells[2].Controls[0];
    TextBox txtSalary =
    (TextBox)gvEmployees.Rows[e.RowIndex].Cells[3].Controls[0];

    emp.Name = txtName.Text;
    emp.Department = txtDepartment.Text;
    emp.Salary = Convert.ToDecimal(txtSalary.Text);

    db.SaveChanges();
    gvEmployees.EditIndex = -1;
    LoadEmployees();
}

// Cancel Edit
protected void gvEmployees_RowCancelingEdit(object sender,
GridViewCancelEventArgs e)
{
    gvEmployees.EditIndex = -1;
    LoadEmployees();
}

// Delete Employee
protected void gvEmployees_RowDeleting(object sender,
GridViewDeleteEventArgs e)
{
    int empID = Convert.ToInt32(gvEmployees.DataKeys[e.RowIndex].Value);
    Employee emp = db.Employees.Find(empID);

    db.Employees.Remove(emp);
    db.SaveChanges();
    LoadEmployees();
}

```



```

}
}

```

Employee Management (Entity Framework)

Name:

Department:

Salary:

EmpID	Name	Department	Salary	
1	John Doe	IT222	60000.00	Edit Delete
2	Jane Smith	HR	55000.00	Edit Delete

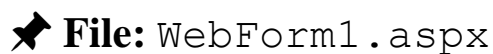
Practical Number 11: Design Web Applications using Client-Side Session Management

Steps to Implement Client-Side State Management

- **View State:** Maintains data across post backs.
- **Query String:** Passes data in the URL.
- **Cookies:** Stores small data persistently.
- **Hidden Fields:** Stores temporary data in forms.

1 View State (ASP.NET Web Forms Only)

Stores data in a hidden field but is accessible only on the same page.



📌 **File:** 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 Practical_Number_11
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (ViewState["UserName"] != null)
            {
```

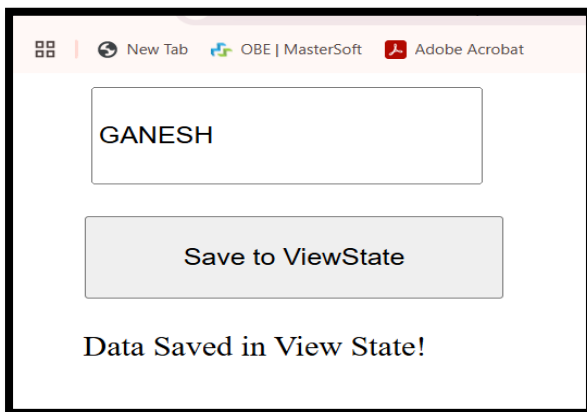
```

        lblMessage.Text = "Stored in View State: " +
ViewState["UserName"].ToString();
    }

}

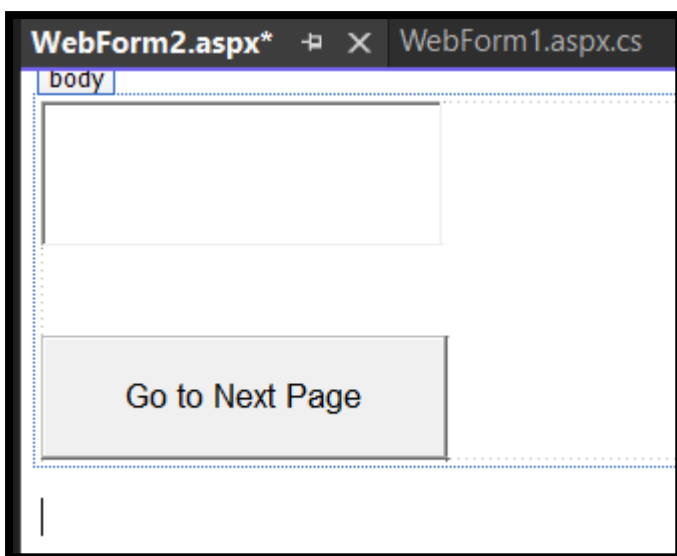
protected void btnSubmit_Click(object sender, EventArgs e)
{
    ViewState["UserName"] = txtName.Text;
    lblMessage.Text = "Data Saved in View State!";
}
}
}

```



2 Query String (Passing Data in URL)

Query strings pass data between pages using the URL.



✦ File: WebForm2.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm2.aspx.cs"
Inherits="Practical_Number_11.WebForm2" %>

<!DOCTYPE html>
<html>
<head>
    <title>Query String Example</title>
</head>
<body>
    <form runat="server">
        <asp:TextBox ID="txtName" runat="server" Height="67px"
Width="193px"></asp:TextBox>
        <br />
        <br />
        <br />
        <asp:Button ID="btnSubmit" runat="server" Text="Go to Next Page"
OnClick="btnSubmit_Click" Height="62px" Width="203px" />
    </form>
</body>
</html>
```

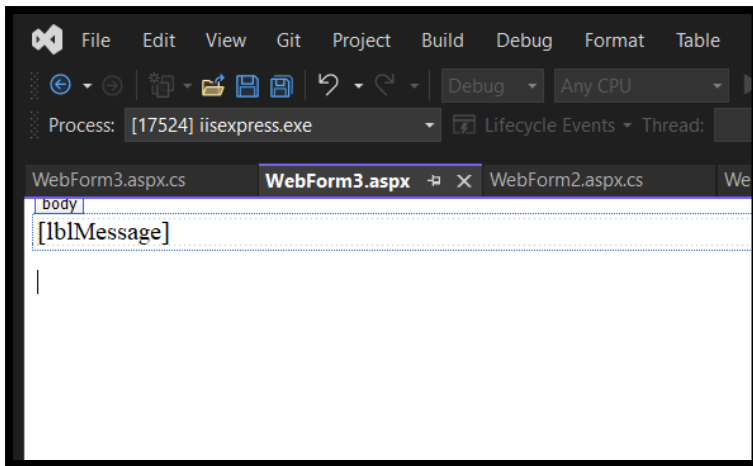
✦ File: WebForm2.aspx.cs

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

namespace Practical_Number_11
{
    public partial class WebForm2 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void btnSubmit_Click(object sender, EventArgs
e)
        {
            Response.Redirect("WebForm3.aspx?name=" +
txtName.Text);
        }
    }
}
```



✦ File: WebForm3.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm3.aspx.cs"
Inherits="Practical_Number_11.WebForm3" %>
```

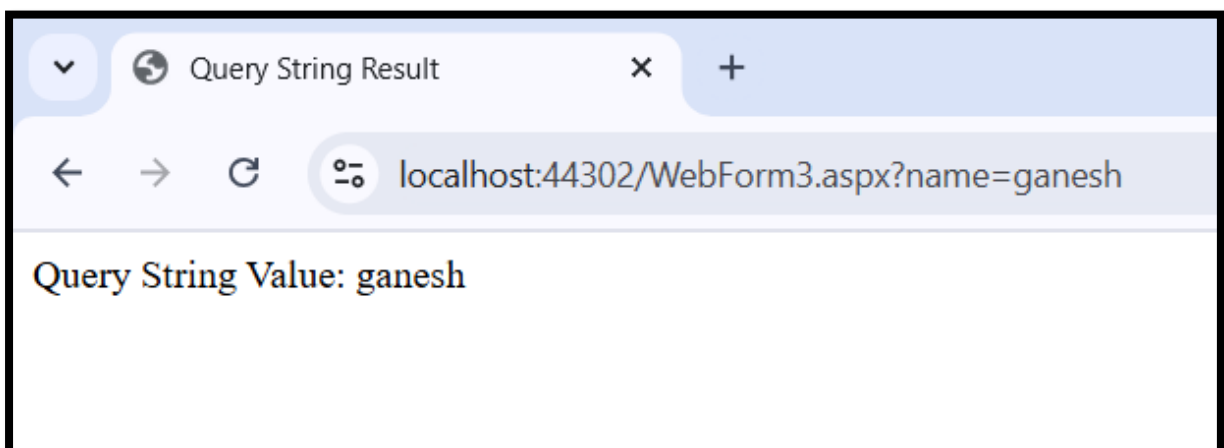
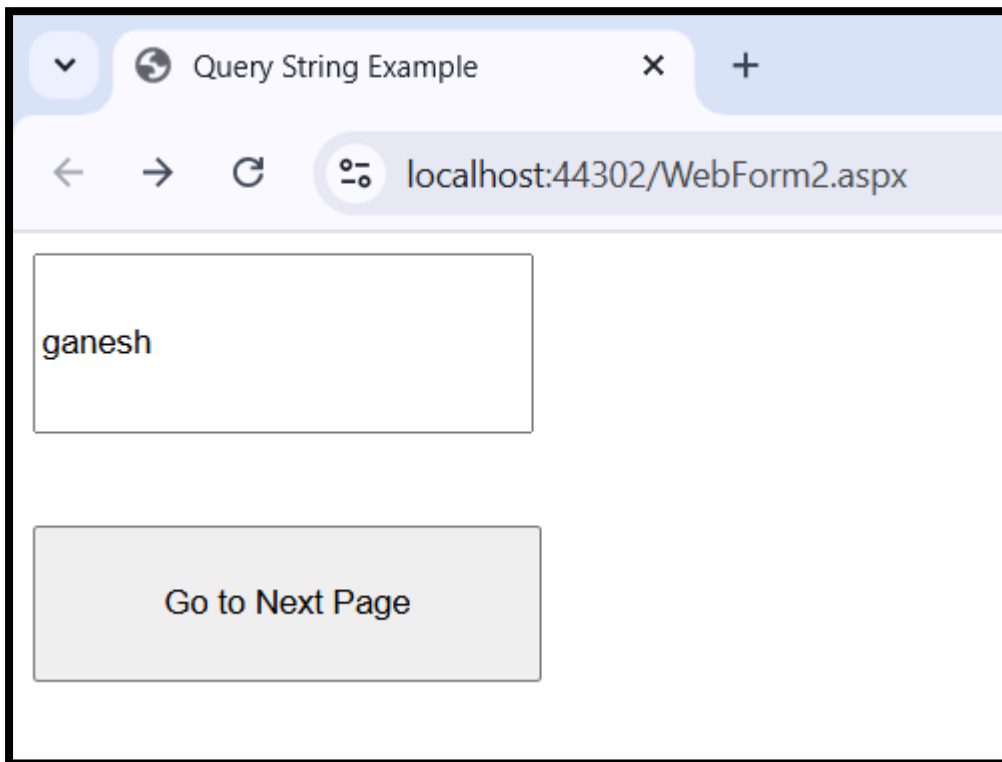
```
<!DOCTYPE html>
<html>
<head>
  <title>Query String Result</title>
</head>
<body>
  <form runat="server">
    <asp:Label ID="lblMessage" runat="server"></asp:Label>
  </form>
</body>
</html>
```

✦ File: WebForm3.aspx.cs

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

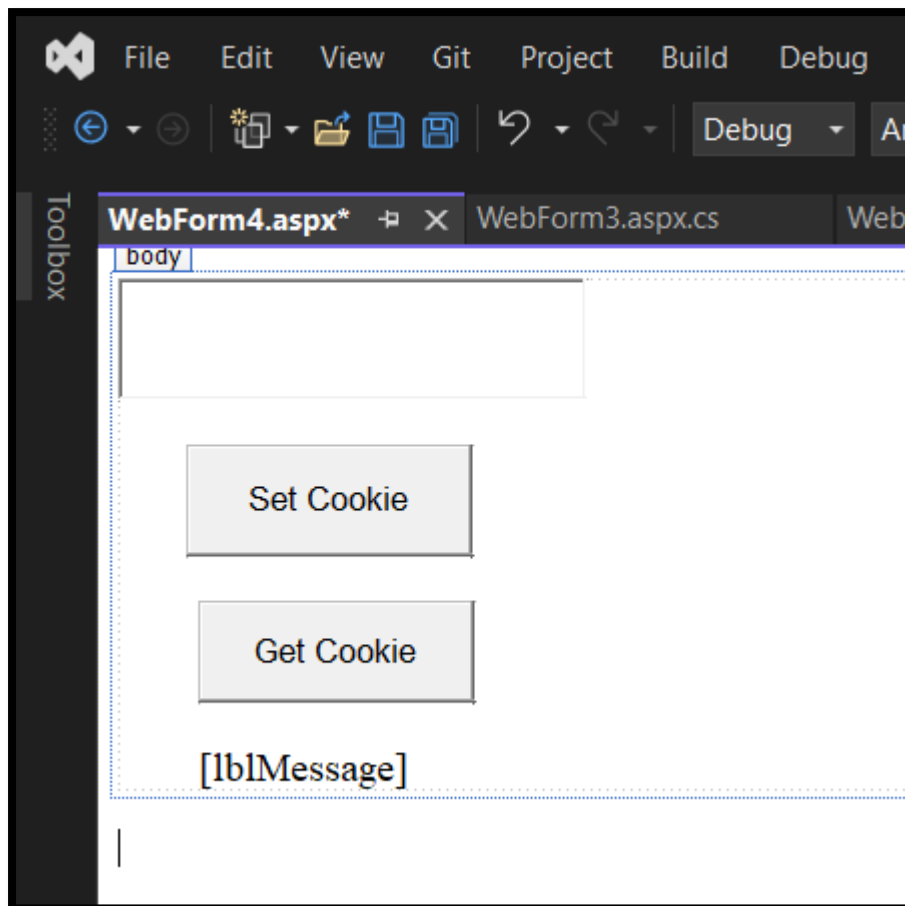
namespace Practical_Number_11
{
    public partial class WebForm3 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (Request.QueryString["name"] != null)
            {
                lblMessage.Text = "Query String Value: " +
Request.QueryString["name"];
            }
        }
    }
}
```

```
}  
    }  
}
```



3 Cookies (Persistent Client-Side Storage)

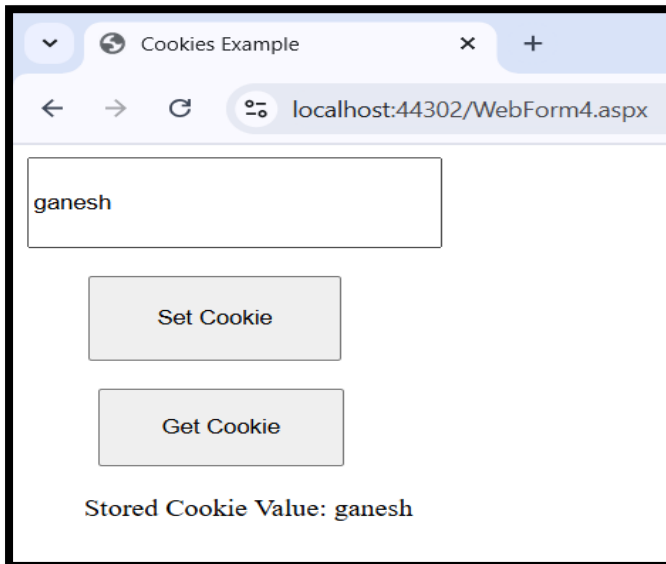
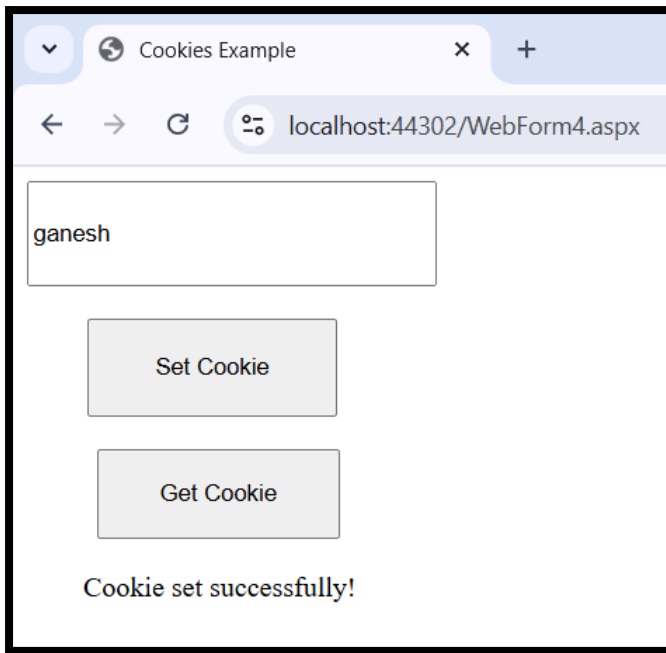
Stores small data on the client-side, accessible across pages.



✦ **File:** WebForm4.aspx

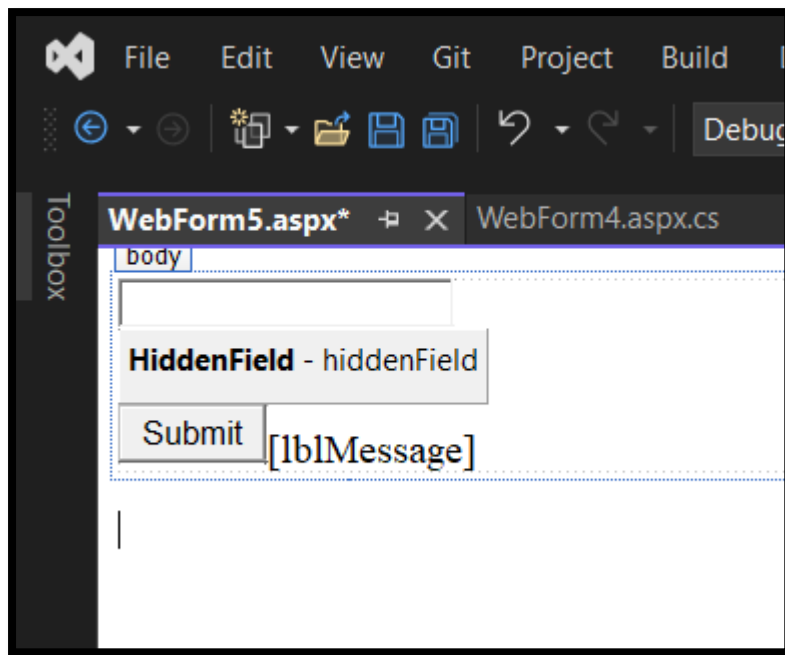
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm4.aspx.cs"
Inherits="Practical_Number_11.WebForm4" %>
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Cookies Example</title>
</head>
<body>
    <form runat="server">
        <asp:TextBox ID="txtName" runat="server" Height="55px"
Width="226px"></asp:TextBox>
        <br />
        <br />
        <asp:Button ID="btnSetCookie" runat="server" Text="Set Cookie"
OnClick="btnSetCookie_Click" Height="56px" style="margin-left: 34px"
Width="143px" />
        <br />
        <br />
        <asp:Button ID="btnGetCookie" runat="server" Text="Get Cookie"
OnClick="btnGetCookie_Click" Height="51px" style="margin-left: 40px"
Width="138px" />
        <br />
    </form>
    <asp:Label ID="lblMessage" runat="server" />
</body>
</html>
```

4 Hidden Fields (Storing Temporary Data)

Stores data in an HTML form but remains invisible to the user.



✦ **File:** WebForm5.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm5.aspx.cs"
Inherits="Practical_Number_11.WebForm5" %>
```

```
<html>
<head>
    <title>Hidden Field Example</title>
</head>
<body>
    <form runat="server">
        <asp:TextBox ID="txtName" runat="server"></asp:TextBox>
        <asp:HiddenField ID="hiddenField" runat="server" Value="12345" />
        <asp:Button ID="btnSubmit" runat="server" Text="Submit"
OnClick="btnSubmit_Click" />
        <asp:Label ID="lblMessage" runat="server"></asp:Label>
    </form>
</body>
</html>
```

✦ **File:** WebForm5.aspx.cs

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

namespace Practical_Number_11
{
    public partial class WebForm5 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

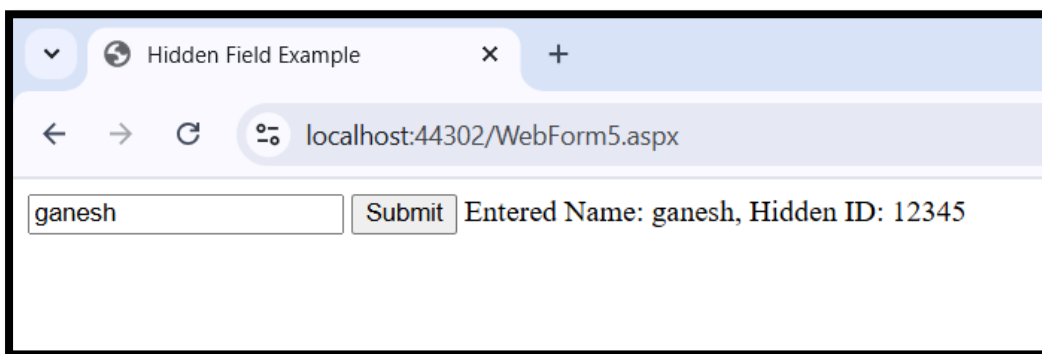
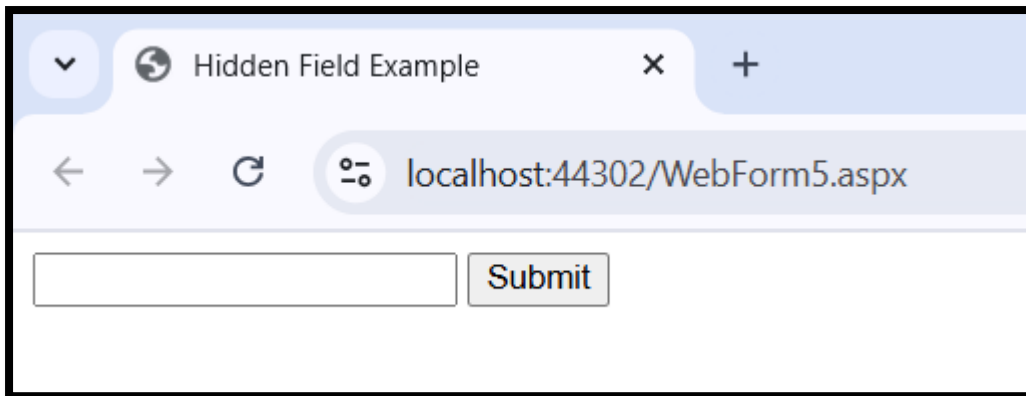
```

```

    }

    protected void btnSubmit_Click(object sender, EventArgs e)
    {
        lblMessage.Text = "Entered Name: " + txtName.Text + ", Hidden ID: " +
hiddenField.Value;
    }
}
}

```



Practical Number 12: Design Web Applications using Server Side Session Management Techniques

Steps to Create a Web Application with Session Management in Visual Studio

1. Create an ASP.NET Web Forms Project

1. Open **Visual Studio**.
2. Click **Create a new project**.
3. Select **ASP.NET Web Application (.NET Framework)** and click **Next**.
4. Name the project (e.g., **SessionManagementDemo**).
5. Choose **Web Forms** and click **Create**.

2. Configure Session Management in Web.config

```
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.7.2" />
    <httpRuntime targetFramework="4.7.2" />
    <sessionState mode="InProc" timeout="20" cookieless="UseCookies" />
  </system.web>
```

Modify Web.config to use **InProc** (default), **StateServer**, or **SQL Server**.

For InProc Session (default):

```
<sessionState mode="InProc" timeout="20" cookieless="UseCookies" />
```

For StateServer Session:

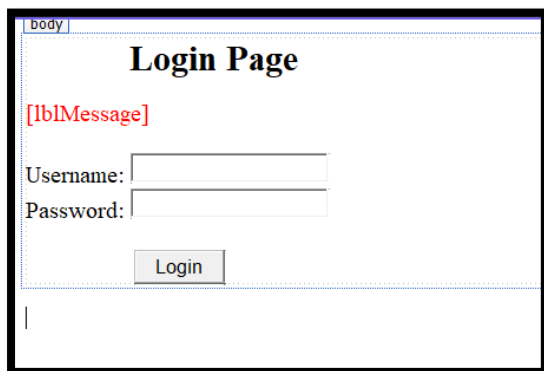
```
<sessionState mode="StateServer" stateConnectionString="tcpip=127.0.0.1:42424" timeout="20"/>
```

For SQL Server Session:

```
<sessionState mode="SqlServer" sqlConnectionString="data source=SQLSERVER;Initial
Catalog=SessionDB;Integrated Security=True" timeout="20"/>
```

3. Implement Login Page (Login.aspx)

Frontend: Login.aspx



```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Login.aspx.cs"
Inherits="SessionManagementDemo.Login" %>
```

```
<!DOCTYPE html>
<html lang="en">
<head runat="server">
  <title>Login</title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
```

```

<h2>Login Page</h2>
<asp:Label ID="lblMessage" runat="server" ForeColor="Red"></asp:Label>
<br />
Username: <asp:TextBox ID="txtUsername" runat="server"></asp:TextBox>
<br />
Password: <asp:TextBox ID="txtPassword" runat="server" TextMode="Password"></asp:TextBox>
<br />
<asp:Button ID="btnLogin" runat="server" Text="Login" OnClick="btnLogin_Click" />
</div>
</form>
</body>
</html>

```

Backend: Login.aspx.cs

```

using System;
using System.Web;

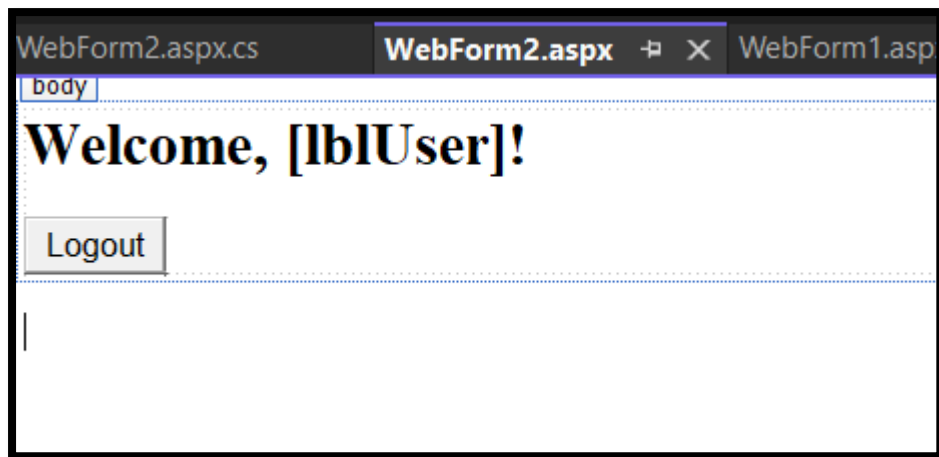
namespace SessionManagementDemo
{
    public partial class Login : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (Session["Username"] != null)
            {
                Response.Redirect("Dashboard.aspx");
            }
        }

        protected void btnLogin_Click(object sender, EventArgs e)
        {
            string username = txtUsername.Text;
            string password = txtPassword.Text;

            // Simulating user authentication
            if (username == "admin" && password == "1234")
            {
                Session["Username"] = username;
                Response.Redirect("Dashboard.aspx");
            }
            else
            {
                lblMessage.Text = "Invalid username or password!";
            }
        }
    }
}

```

4. Create a Dashboard Page (Dashboard.aspx)



Frontend: Dashboard.aspx

```
<% @ Page Language="C#" AutoEventWireup="true" CodeBehind="Dashboard.aspx.cs"
Inherits="SessionManagementDemo.Dashboard" %>

<!DOCTYPE html>
<html lang="en">
<head runat="server">
    <title>Dashboard</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <h2>Welcome, <asp:Label ID="lblUser" runat="server"></asp:Label>!</h2>
            <asp:Button ID="btnLogout" runat="server" Text="Logout" OnClick="btnLogout_Click" />
        </div>
    </form>
</body>
</html>
```

Backend: Dashboard.aspx.cs

```
using System;
using System.Web;

namespace SessionManagementDemo
{
    public partial class Dashboard : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (Session["Username"] == null)
            {
                Response.Redirect("Login.aspx");
            }
            else
            {
                lblUser.Text = Session["Username"].ToString();
            }
        }
    }
}
```

```
}  
  
protected void btnLogout_Click(object sender, EventArgs e)  
{  
    Session.Abandon();  
    Response.Redirect("Login.aspx");  
}  
}  
}
```

Login

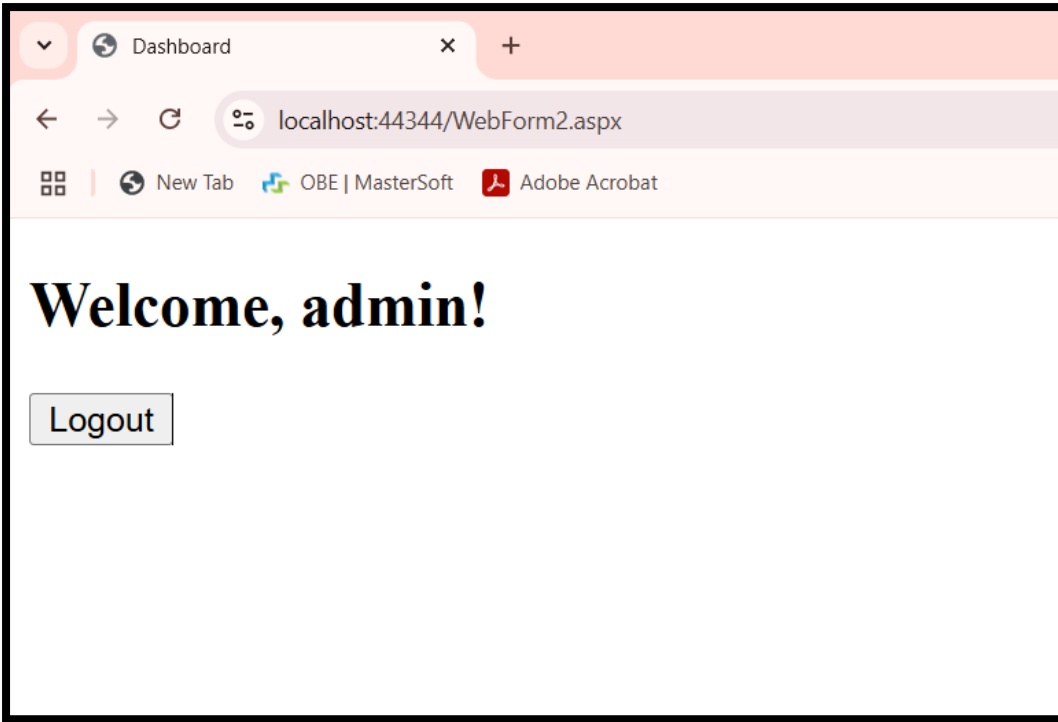
localhost:44344/WebForm1.aspx

Login Page

Username:

Password:

Login



Difference between InProc, StateServer, and SQL Server session modes in ASP.NET Web.config:

Session Mode	Storage Location	Performance	Scalability	Reliability	Use Case
InProc (Default)	Stores session data in web server memory (RAM)	Fastest (since it's in memory)	Not scalable (session lost if app restarts or crashes)	Not reliable (session lost if server restarts)	Best for small applications with a single server
StateServer	Stores session data in a separate ASP.NET State Server (Windows Service)	Slower than InProc (requires serialization)	Scalable (multiple web servers can use the same state server)	More reliable (session survives web server restarts)	Used in web farms where multiple servers share session data
SQL Server	Stores session data in a SQL Server database	Slowest (due to database access)	Highly scalable (multiple servers can share the same DB)	Most reliable (session persists even if the web server crashes)	Best for large applications needing high availability

When to Use Each Mode?

- **Use InProc** → When performance is a priority and you are using a **single server**.
- **Use StateServer** → When using **multiple web servers (Web Farm)** and need better session persistence.
- **Use SQL Server** → When building a **large-scale, enterprise application** that requires **high availability and failover support**.

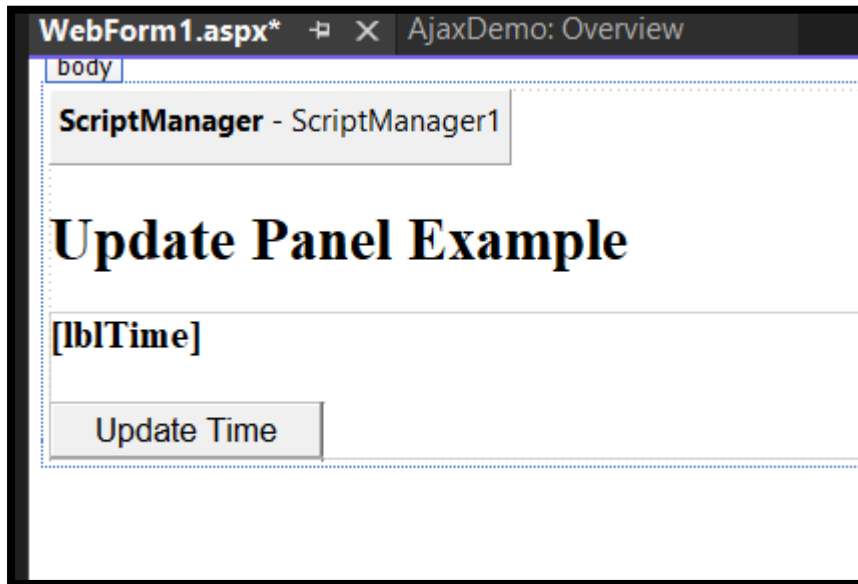
Practical Number 13: Build a web page using AJAX Controls.

1st Program: AJAX UpdatePanel Example (Partial Page Update)

This program updates the **current time** without refreshing the entire page.

Steps:

1. Create an **ASP.NET Web Forms** project in **Visual Studio**.
2. Add a new **Web Form** (WebForm1.aspx).
3. Copy the following code.



Frontend: WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="AjaxDemo.WebForm1" %>

<!DOCTYPE html>
<html lang="en">
<head runat="server">
    <title>AJAX UpdatePanel Example</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:ScriptManager ID="ScriptManager1"
runat="server"></asp:ScriptManager>

        <h2>Update Panel Example</h2>

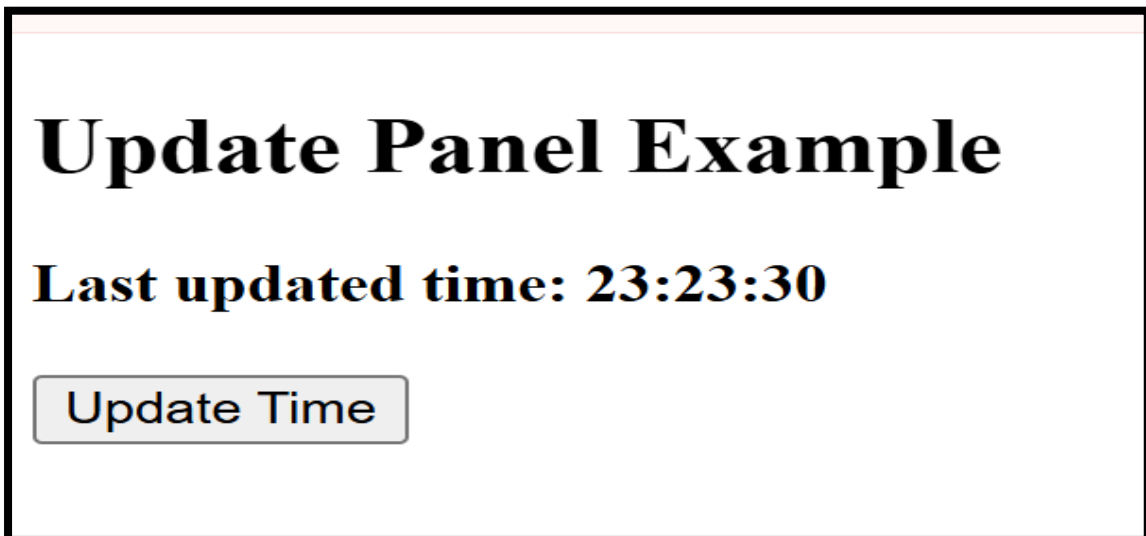
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <asp:Label ID="lblTime" runat="server" Font-
Bold="True"></asp:Label>
                <br /><br />
                <asp:Button ID="btnUpdate" runat="server" Text="Update Time"
OnClick="btnUpdate_Click" />
            </ContentTemplate>
        </asp:UpdatePanel>
    </form>
</body>
</html>
```

Backend: 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 AjaxDemo
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                lblTime.Text = "Last updated time: " +
DateTime.Now.ToString("HH:mm:ss");
            }

            protected void btnUpdate_Click(object sender, EventArgs e)
            {
                lblTime.Text = "Last updated time: " +
DateTime.Now.ToString("HH:mm:ss");
            }
        }
    }
}
```



2nd Program: AJAX Timer Example (Auto Refresh without Full Page Load)

This program automatically updates the time every 5 seconds using an AJAX Timer Control.

Steps:

1. Add another Web Form (TimerDemo.aspx).
2. Copy the following code.



Frontend: WebForm2.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm2.aspx.cs"
Inherits="AjaxDemo.WebForm2" %>

<!DOCTYPE html>
<html lang="en">
<head runat="server">
    <title>AJAX Timer Example</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:ScriptManager ID="ScriptManager1"
runat="server"></asp:ScriptManager>

        <h2>AJAX Timer Example (Auto Refresh)</h2>

        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <asp:Label ID="lblTime" runat="server" Font-
Bold="True"></asp:Label>
            </ContentTemplate>
        </asp:UpdatePanel>

        <asp:Timer ID="Timer1" runat="server" Interval="5000"
OnTick="Timer1_Tick" />
    </form>
</body>
</html>
```

Backend: TimerDemo.aspx.cs

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

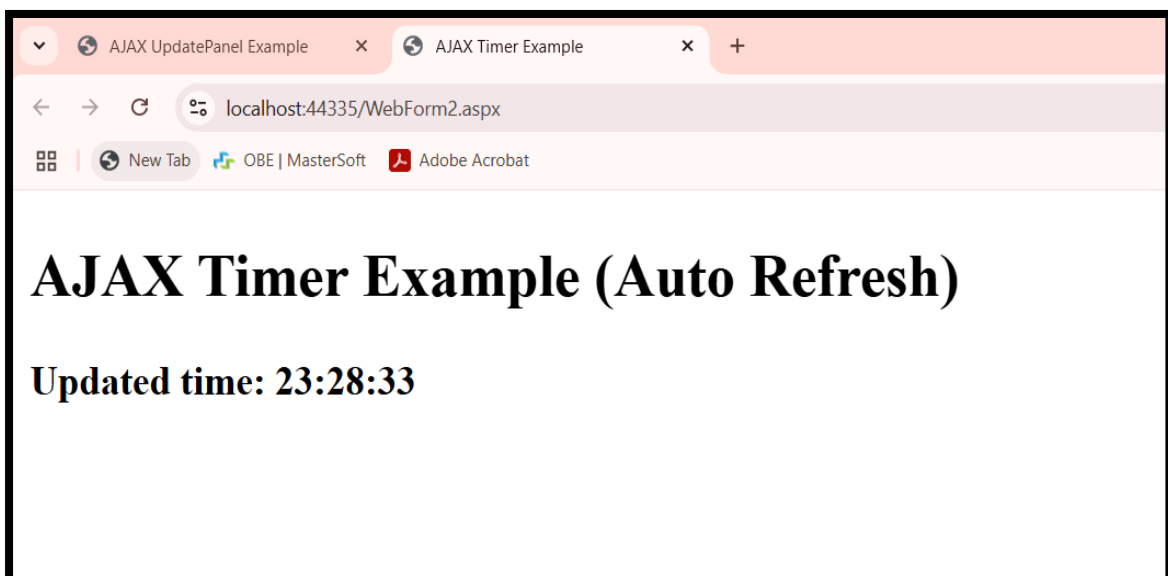
```

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

namespace AjaxDemo
{
    public partial class WebForm2 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                lblTime.Text = "Current time: " +
DateTime.Now.ToString("HH:mm:ss");
            }

            protected void Timer1_Tick(object sender, EventArgs e)
            {
                lblTime.Text = "Updated time: " + DateTime.Now.ToString("HH:mm:ss");
            }
        }
    }
}

```



Practical Number 14: Build a web application to create and use web service in ASP.net

1. Create New ASP.NET Web Forms App

- Open Visual Studio
- Create a new project:
ASP.NET Web Application (.NET Framework)

- Name it: CalculatorApp
- Template: **Empty**
- Check: **Web Forms**
- Click **Create**

2. Add ASMX Web Service

- Right-click project → **Add** → **New Item**
- Select **Web Service (ASMX)**
- Name it: CalculatorService.asmx

CalculatorService.asmx.cs

```
using System.Web.Services;

namespace CalculatorApp
{
    /// <summary>
    /// Web service for basic calculator operations
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    public class CalculatorService : WebService
    {
        [WebMethod]
        public int Add(int a, int b)
        {
            return a + b;
        }

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

3. Add a Web Form

- Right-click project → **Add** → **New Item**
- Choose **Web Form** → Name: Default.aspx

Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="CalculatorApp.Default" %>

<!DOCTYPE html>
<html>
<head runat="server">
    <title>Calculator Web Service Client</title>
```

```

</head>
<body>
  <form id="form1" runat="server">
    <div>
      Number 1: <asp:TextBox ID="txtA" runat="server" /><br />
      Number 2: <asp:TextBox ID="txtB" runat="server" /><br /><br />
      <asp:Button ID="btnAdd" runat="server" Text="Add" OnClick="btnAdd_Click" />
      <asp:Button ID="btnSub" runat="server" Text="Subtract" OnClick="btnSub_Click" /><br /><br />
      Result: <asp:Label ID="lblResult" runat="server" Text="" />
    </div>
  </form>
</body>
</html>

```

4. Add the Service Reference (Self Reference)

- Build the project first (Ctrl+Shift+B)
- Right-click the project → **Add Service Reference**
- Address: `http://localhost:<your-port>/CalculatorService.asmx`
 - (Run the project and copy the URL from the browser)
- Namespace: `CalcRef`
- Click **OK**

5. Code-Behind for Default.aspx.cs

```

using System;
using CalculatorApp.CalcRef;

namespace CalculatorApp
{
    public partial class Default : System.Web.UI.Page
    {
        CalculatorServiceSoapClient client;

        protected void Page_Load(object sender, EventArgs e)
        {
            client = new CalculatorServiceSoapClient();
        }

        protected void btnAdd_Click(object sender, EventArgs e)
        {
            int a = int.Parse(txtA.Text);
            int b = int.Parse(txtB.Text);
            int result = client.Add(a, b);
            lblResult.Text = "Result: " + result;
        }

        protected void btnSub_Click(object sender, EventArgs e)
        {
            int a = int.Parse(txtA.Text);
            int b = int.Parse(txtB.Text);
            int result = client.Subtract(a, b);
            lblResult.Text = "Result: " + result;
        }
    }
}

```

```
}  
}
```

Practical Number 15: Build a web application to create and use WCF service in ASP.net

Configure your new project

WCF Service C# Windows Web Service

Project name
WCFSERVICE5

Location
C:\Users\admin\source\repos

Solution name ⓘ
WCFSERVICE5

☐ Place solution and project in the same directory

Framework
.NET Framework 4.7.2

Project will be created in "C:\Users\admin\source\repos\WCFSERVICE5\WCFSERVICE5"

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  
{  
  
    [OperationContract]  
    string GetData(int value);  
}
```



```

[OperationContract]
double add(double a, double b);

[OperationContract]
double sub(double a, double b);

[OperationContract]
double mul(double a, double b);

[OperationContract]
double div(double a, double b);

[OperationContract]
CompositeType GetDataUsingDataContract(CompositeType composite);

    // TODO: Add your service operations here
}

// Use a data contract as illustrated in the sample below to add composite types
to service operations.
[DataContract]
public class CompositeType
{
    bool boolValue = true;
    string stringValue = "Hello ";

    [DataMember]
    public bool BoolValue
    {
        get { return boolValue; }
        set { boolValue = value; }
    }

    [DataMember]
    public string StringValue
    {
        get { return stringValue; }
        set { stringValue = value; }
    }
}

```

Services.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 string GetData(int value)
    {

```

```

        return string.Format("You entered: {0}", value);
    }

    public double add(double a, double b)
    {
        return a + b;
    }

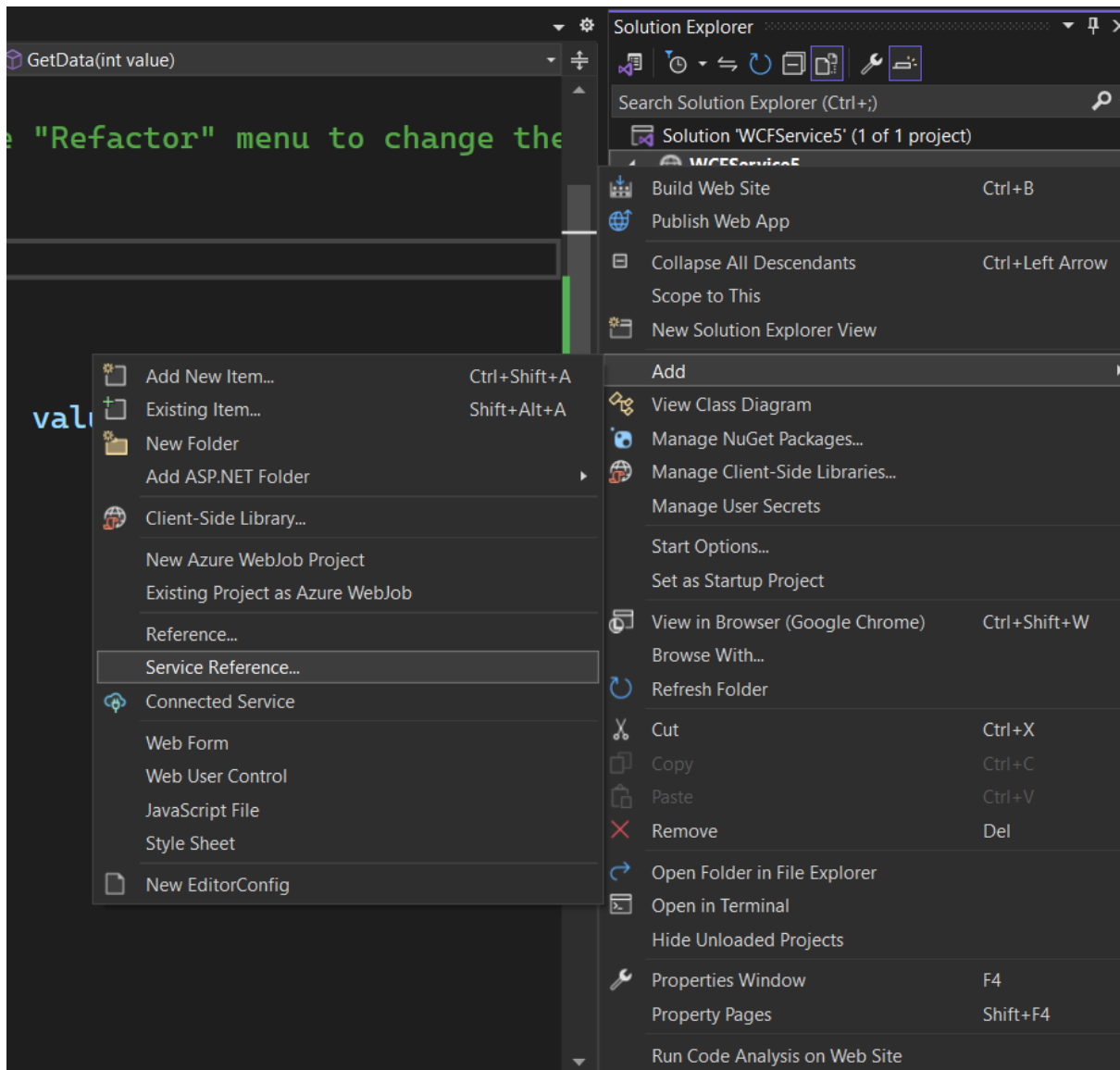
    public double sub(double a, double b)
    {
        return a - b;
    }

    public double mul(double a, double b)
    {
        return a * b;
    }

    public double div(double a, double b)
    {
        return a / b;
    }

    public CompositeType GetDataUsingDataContract(CompositeType composite)
    {
        if (composite == null)
        {
            throw new ArgumentNullException("composite");
        }
        if (composite.BoolValue)
        {
            composite.StringValue += "Suffix";
        }
        return composite;
    }
}

```



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:63146/Service.svc

Go

Discover

Services:

Operations:

▶  Service.svc

Select a service contract to view its operations.

1 service(s) found in the solution.

Namespace:

ServiceReference2

Advanced...

OK

Cancel

WebForm.cs

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

public partial class _Default : 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)
    {
        double a = Convert.ToDouble(TextBox1.Text);
        double b = Convert.ToDouble(TextBox2.Text);

        double result= service.add(a, b);

        Label1.Text = "Addition" + result.ToString();
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        double a = Convert.ToDouble(TextBox1.Text);
        double b = Convert.ToDouble(TextBox2.Text);

        double result = service.sub(a, b);

        Label1.Text = "Subtraction" + result.ToString();
    }

    protected void Button3_Click(object sender, EventArgs e)
    {
        double a = Convert.ToDouble(TextBox1.Text);
        double b = Convert.ToDouble(TextBox2.Text);

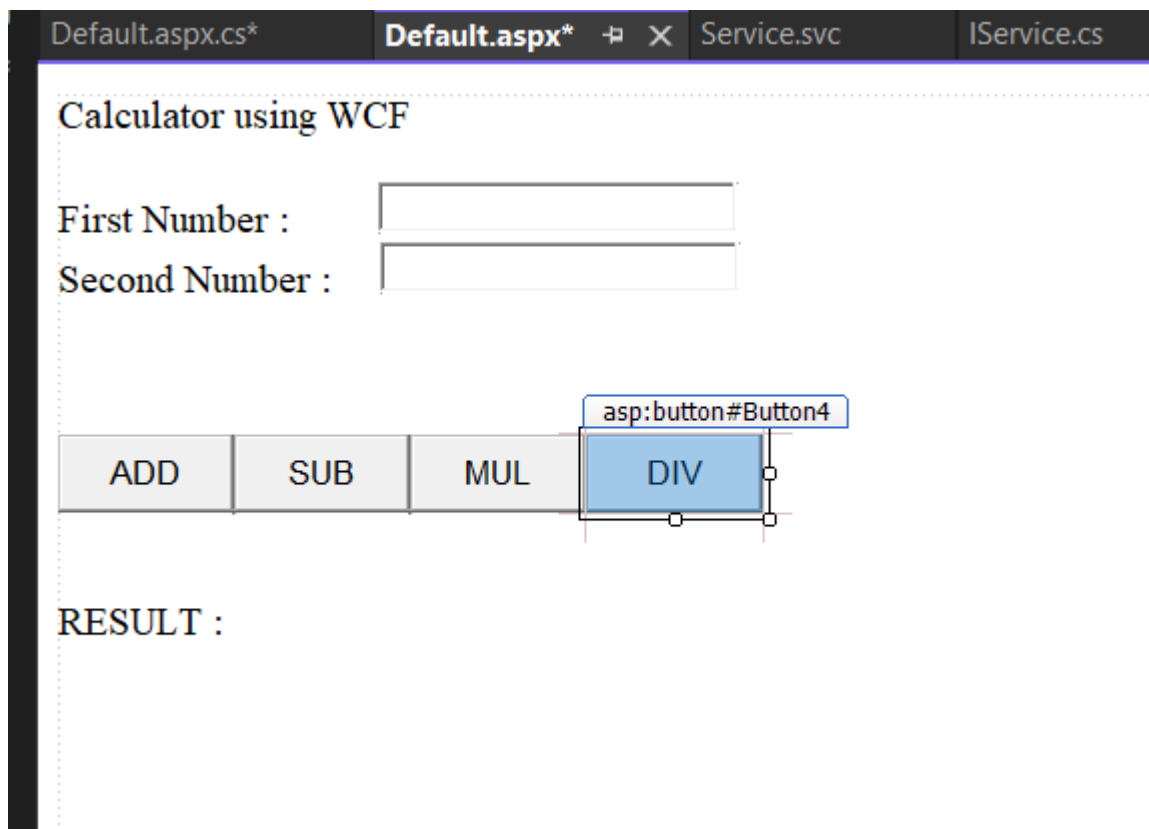
        double result = service.mul(a, b);

        Label1.Text = "Multiplication" + result.ToString();
    }

    protected void Button4_Click(object sender, EventArgs e)
    {
        double a = Convert.ToDouble(TextBox1.Text);
        double b = Convert.ToDouble(TextBox2.Text);

        double result = service.div(a, b);

        Label1.Text = "Division" + result.ToString();
    }
}
```



```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>

<!DOCTYPE html>

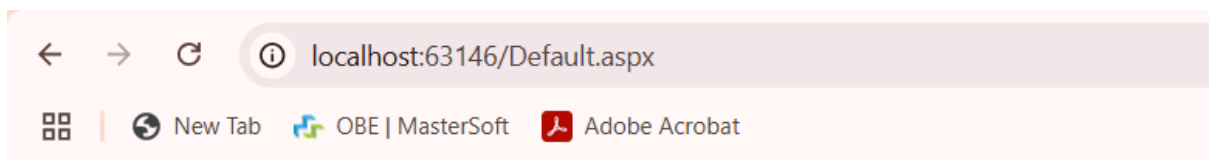
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            Calculator using WCF<br />
            <br />
            First Number :<asp:TextBox ID="TextBox1" runat="server"
style="margin-left: 44px" Width="171px"></asp:TextBox>
            <br />
            Second Number :
            <asp:TextBox ID="TextBox2" runat="server" style="margin-left: 19px"
Width="171px"></asp:TextBox>
            <br />
            <br />
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" Height="39px"
OnClick="Button1_Click" Text="ADD" Width="88px" />
            <asp:Button ID="Button2" runat="server" Height="39px"
OnClick="Button2_Click" Text="SUB" Width="88px" />
            <asp:Button ID="Button3" runat="server" Height="39px"
OnClick="Button3_Click" Text="MUL" Width="88px" />
```

```

        <asp:Button ID="Button4" runat="server" Height="39px"
OnClick="Button4_Click" Text="DIV" Width="88px" />
        <br />
        <br />
        <br />
        <asp:Label ID="Label1" runat="server" Text="RESULT :"></asp:Label>
        <br />
        <br />
        <br />
        <br />
        <br />
        <br />
        <br />
        <br />
        </div>
    </form>
</body>
</html>

```

OUTPUT:



Calculator using WCF

First Number :

Second Number :

ADD	SUB	MUL	DIV
-----	-----	-----	-----

Division3

Practical 16 : MVC Application using Entity Framework

Database Setup

◆ Step 1: Create a Database in SQL Server

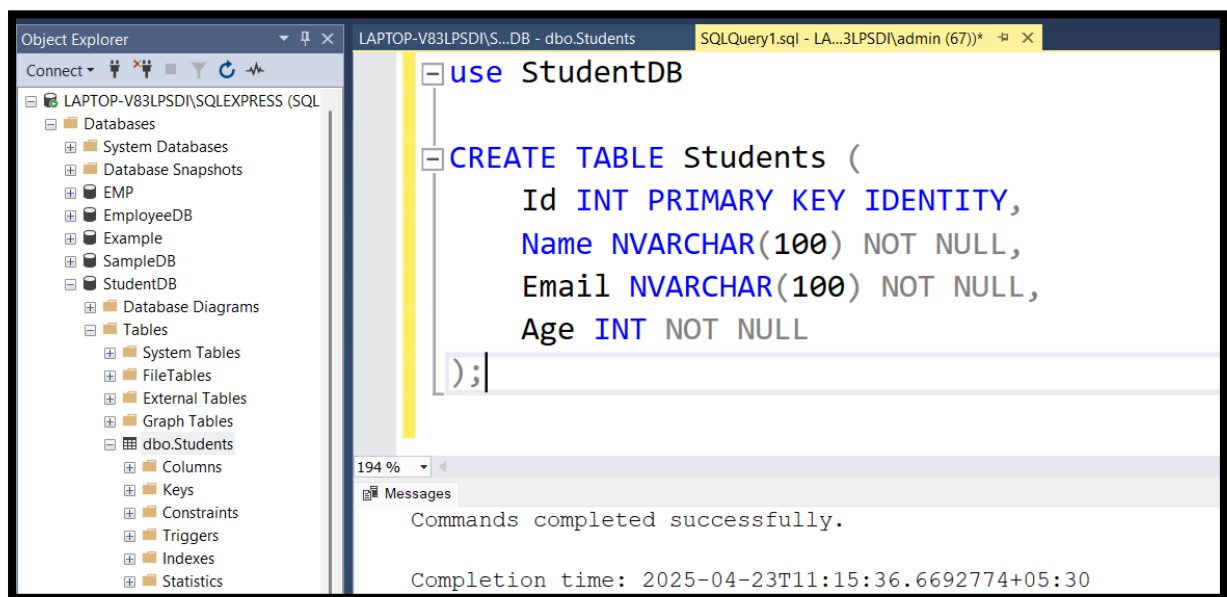
Open SQL Server Management Studio (SSMS) or Visual Studio SQL Server Object Explorer.

Create a Database named StudentDB.

Create a Table using the following SQL: Create Database StudentDB

use StudentDB

```
CREATE TABLE Students (  
    Id INT PRIMARY KEY IDENTITY,  
    Name NVARCHAR(100) NOT NULL,  
    Email NVARCHAR(100) NOT NULL,  
    Age INT NOT NULL  
);
```



The screenshot shows the SQL Server Management Studio interface with the 'Students' table selected. The table contains the following data:

Id	Name	Email	Age
1	ganesh	abc@gmail....	38
2	nilesh	xyz@gmail....	36
▶*	NULL	NULL	NULL

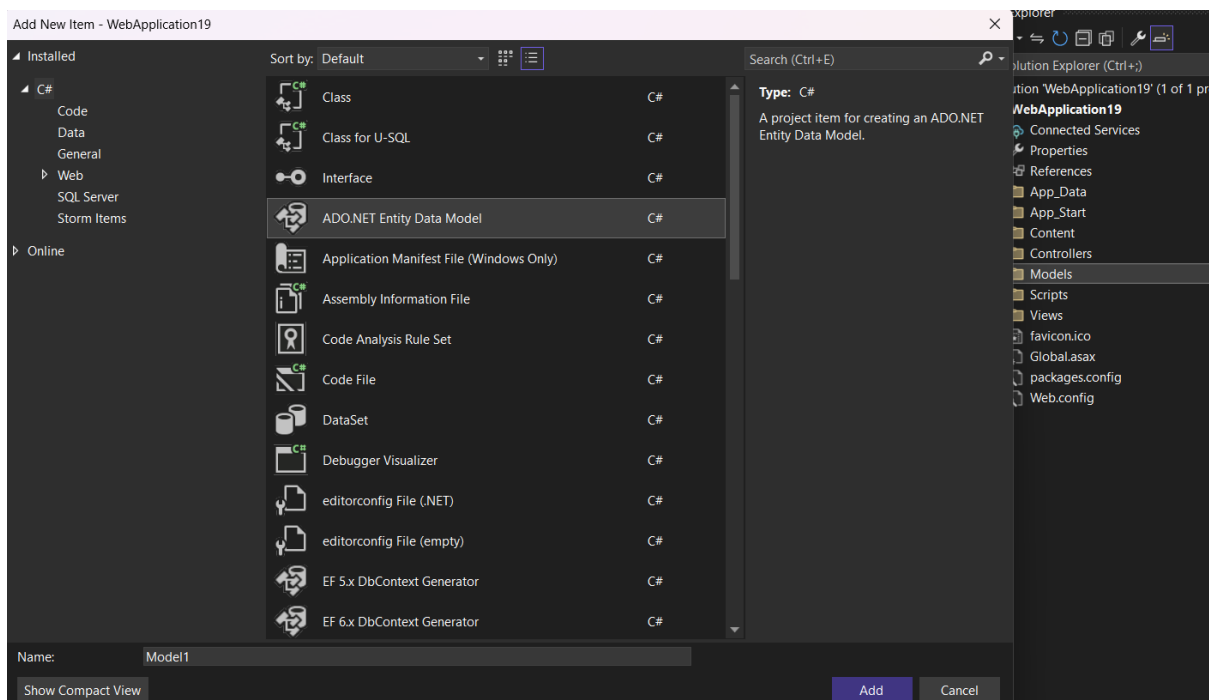
Step 2: Create a New ASP.NET MVC Project

1. Open **Visual Studio**
2. Select **Create a new project**
3. Choose: **ASP.NET Web Application (.NET Framework)**
4. Name: `StudentMVCApp`
5. Choose **MVC** as the template
6. Click **Create**

Entity Framework Model Setup

◆ Step 3: Add Entity Framework Model

1. Right-click the **Models** folder → Add → New Item
2. Choose **ADO.NET Entity Data Model**
3. Name it: `StudentModel.edmx`
4. Choose: "EF Designer from database"
5. Select your SQL Server database (`StudentDB`)
6. Select the `Students` table
7. Finish to generate model classes



**Choose Model Contents****What should the model contain?****EF Designer
from
database**Empty EF
Designer
modelEmpty Code
First modelCode First
from
database

Creates a model in the EF Designer based on an existing database. You can choose the database connection, settings for the model, and database objects to include in the model. The classes your application will interact with are generated from the model.

< Previous

Next >

Finish

Cancel

Connection Properties



Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:

Microsoft SQL Server (SqlClient)

Change...

Server name:

LAPTOP-V83LPDI\SQLEXPRESS



Refresh

Log on to the server

Authentication: Windows Authentication



User name:

Password:

Encrypt:

Mandatory (True)



☒ Trust Server Certificate

☐ Save my password

Connect to a database

☒ Select or enter a database name:

StudentDB



☐ Attach a database file:

Browse...

Logical name:

Advanced...

Test Connection

OK

Cancel

**Choose Your Data Connection****Which data connection should your application use to connect to the database?**

laptop-v83lpsdi\sqlexpress.StudentDB.dbo ▾

New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

- ☐ No, exclude sensitive data from the connection string. I will set it in my application code.
- ☐ Yes, include the sensitive data in the connection string.

Connection string:

```
metadata=res:/*//Models.Model1.csdl|res:/*//Models.Model1.ssdl|
res:/*//Models.Model1.msl;provider=System.Data.SqlClient;provider connection string="data
source=LAPTOP-V83LPSDI\SQLEXPRESS;initial catalog=StudentDB;integrated
security=True;trustservercertificate=True;MultipleActiveResultSets=True;App=EntityFramework"
```

☒ Save connection settings in Web.Config as:

StudentDBEntities

< Previous

Next >

Finish

Cancel

**Choose Your Version**

Which version of Entity Framework do you want to use?

☒ Entity Framework 6.x

☐ Entity Framework 5.0



It is also possible to install and use other versions of Entity Framework.

[Learn more about this](#)

< Previous

Next >

Finish

Cancel

**Choose Your Database Objects and Settings****Which database objects do you want to include in your model?**

- ✓ ☒ Tables
 - > ☒ dbo
- ☐ Views
- ☐ Stored Procedures and Functions

- ☒ Pluralize or singularize generated object names
- ☒ Include foreign key columns in the model
- ☐ Import selected stored procedures and functions into the entity model

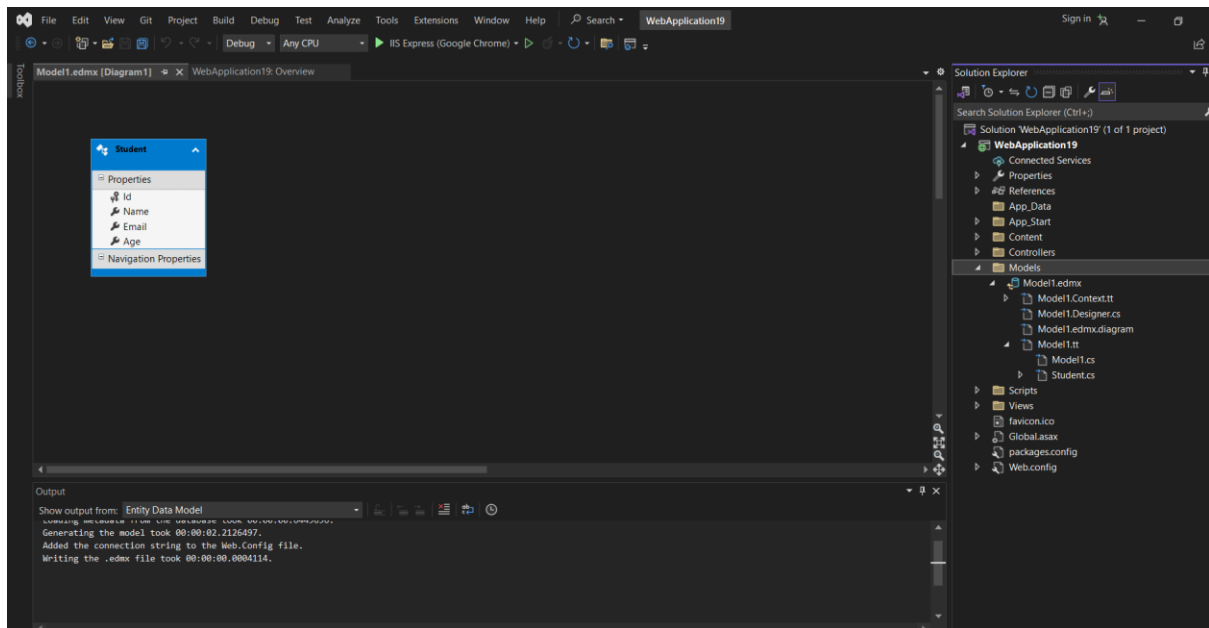
Model Namespace:

< Previous

Next >

Finish

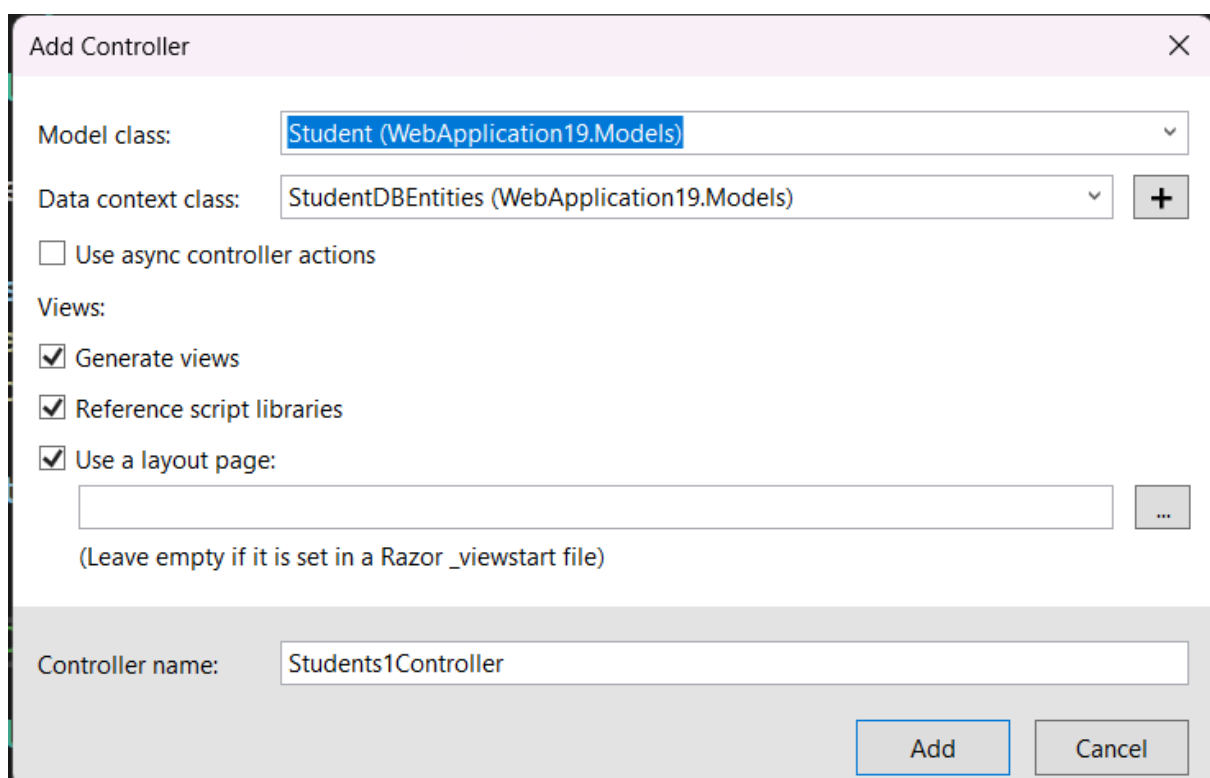
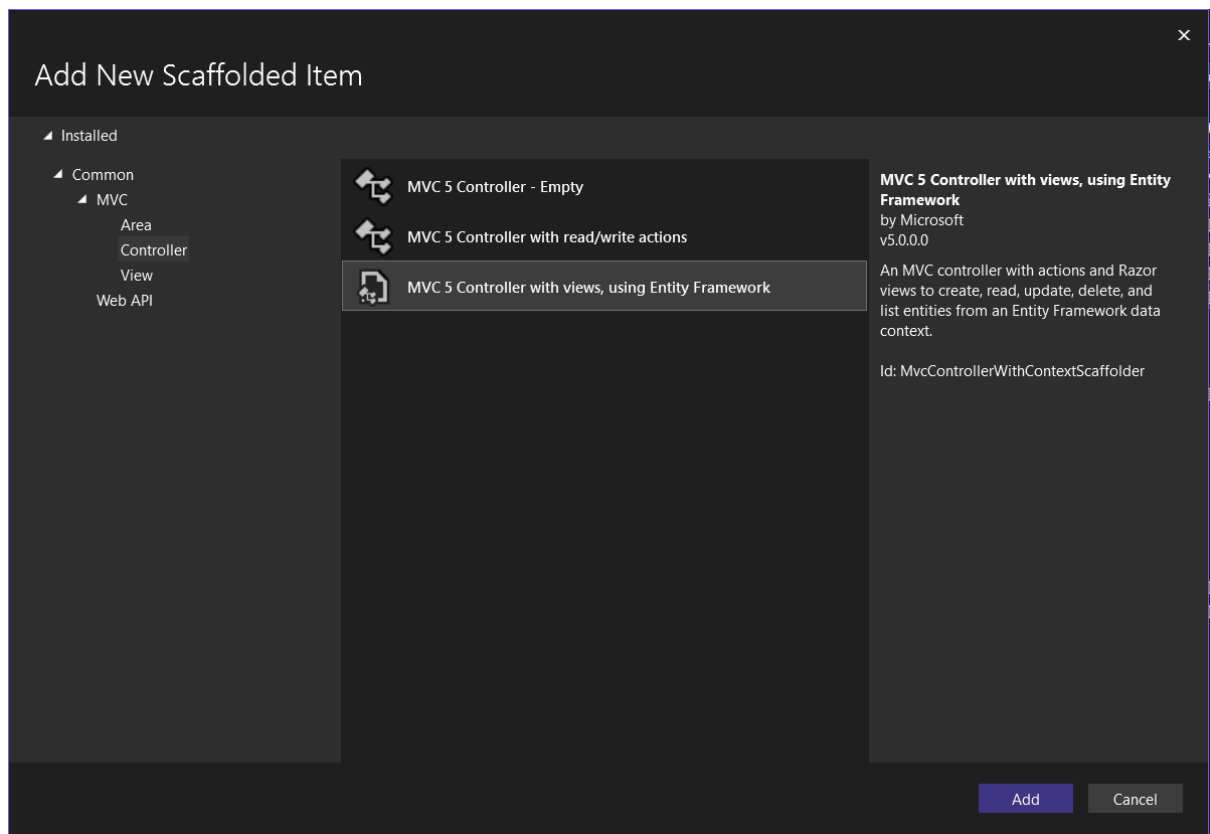
Cancel



++

Step 4: Create Controller

1. Right-click **Controllers** → Add → Controller
2. Choose: **MVC 5 Controller with views, using Entity Framework**
3. Model class: `Student`
4. Data context: `StudentDBEntities` (if using .edmx) or `StudentDBContext`
5. Click **Add**



Step 5: Create Views

You can right-click on each controller action and choose **Add View**, or use the auto-generated ones.

Example: Views/Students/Index.cshtml

```
@model IEnumerable<StudentMVCApp.Models.Student>
@{
    ViewBag.Title = "Student List";
}
<h2>Student List</h2>
<p>@Html.ActionLink("Create New", "Create")</p>

<table class="table">
    <tr>
        <th>@Html.DisplayNameFor(model => model.Name)</th>
        <th>@Html.DisplayNameFor(model => model.Email)</th>
        <th>@Html.DisplayNameFor(model => model.Age)</th>
        <th>Actions</th>
    </tr>

    @foreach (var item in Model) {
        <tr>
            <td>@item.Name</td>
            <td>@item.Email</td>
            <td>@item.Age</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>
    }
```

```
</tr>
}
</table>
```

step 6: Set Default Route

In `App_Start/RouteConfig.cs`, change default route to:

```
csharp
CopyEdit
defaults: new { controller = "Students", action = "Index", id =
UrlParameter.Optional }
```

OUTPUT:

localhost:44359/Students

Application name

Home

About

Contact

Index

[Create New](#)

Name	Email	Age	
ganesh	abc@gmail.com	38	Edit Details Delete
nilesh	xyz@gmail.com	36	Edit Details Delete
ganesh	abc	12	Edit Details Delete

© 2025 - My ASP.NET Application

Create

Student

Name

Email

Age

Create

[Back to List](#)
