|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Advanced Web Technologies(AWT) Lab (MCAL25)**  **INDEX**  **Name of the faculty: Ganesh Bhagwat** | | | | |
| **Experiment Number** | **Name of the experiment** | **Date** | **CO** | **Sign** |
| **1** | **Design a Web Application for an Organization with Registration forms and advanced controls.** |  | **CO1** |  |
| **2** | **Create a website using the master page concept.** |  | **CO1** |  |
| **3** | **Design a Web Application using advanced controls.** |  | **CO1** |  |
| **4** | **Webpage Demonstrating Connection-Oriented Architecture (ASP.NET Web Forms with SQL Server Database)** |  | **CO2** |  |
| **5** | **Webpage Demonstrating Disconnected Architecture (ASP.NET Web Forms with SQL Server Database)** |  | **CO2** |  |
| **6** | **Create a webpage that demonstrates the use of data bound controls of ASP.NET.** |  | **CO2** |  |
| **7** | **Design a webpage to demonstrate the working of a simple stored**  **procedure.** |  | **CO2** |  |
| **8** | **Design a webpage to demonstrate the working of parameterized stored procedure.** |  | **CO2** |  |
| **9** | **Design a webpage to display the use of LINQ.** |  | **CO2** |  |
| **10** | **Build websites to demonstrate the working of entity frameworks in dot net.** |  | **CO3** |  |
| **11** | **Design Web Applications using Client Side Session Management** |  | **C03** |  |
| **12** | **Design Web Applications using Server Side Session Management Techniques** |  | **CO3** |  |
| **13** | **Build a web page using AJAX Controls.** |  | **CO3** |  |
| **14** | **Build a web application to create and use web service in ASP.net** |  | **CO3** |  |
| **15** | **Build a web application to create and WCF service in ASP.net** |  | **CO3** |  |
| **16** | **Design web application using MVC framework** |  | **CO4** |  |

**Practical Number 1**

**Aim:Design a Web Application for an Organization with Registration forms and advanced controls.**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Practical\_1.WebForm1" %>

<!DOCTYPE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<br />

<strong>Regestration Form C24072</strong><br />

<br />

Full Name:&nbsp;&nbsp;

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

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ControlToValidate="TextBox1" ErrorMessage=" \*Name Required" ForeColor="#CC0000"></asp:RequiredFieldValidator>

<br />

<br />

Email:&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>

<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server" ControlToValidate="TextBox2" ErrorMessage="Must be a valid email format" ForeColor="#CC0000" ValidationExpression="\w+([-+.']\w+)@\w+([-.]\w+)\.\w+([-.]\w+)"></asp:RegularExpressionValidator>

<br />

<br />

Password:&nbsp;&nbsp;&nbsp; <asp:TextBox ID="TextBox3" runat="server" TextMode="Password" ></asp:TextBox>

<asp:RegularExpressionValidator ID="RegularExpressionValidator2" runat="server" ControlToValidate="TextBox2" ErrorMessage="Minimum 8 characters, containing at least one uppercase letter and one number" ForeColor="#CC0000" ValidationExpression="^(?=.[A-Z])(?=.\*\d).{8,}$"></asp:RegularExpressionValidator>

<br />

<br />

Confirm Password:<asp:TextBox ID="TextBox4" runat="server" TextMode="Password" ></asp:TextBox>

<asp:CompareValidator ID="CompareValidator1" runat="server" ControlToCompare="TextBox3" ControlToValidate="TextBox4" ErrorMessage="\*Must match the password " ForeColor="#CC0000"></asp:CompareValidator>

<br />

<br />

Age:&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; <asp:TextBox ID="TextBox5" runat="server"></asp:TextBox>

<asp:RangeValidator ID="RangeValidator1" runat="server" ControlToValidate="TextBox5" ErrorMessage="\*Age Should be between 18 and 60" ForeColor="#CC0000"></asp:RangeValidator>

<br />

<br />

<asp:Button ID="Button1" runat="server" Text="Submit" />

</div>

</form>

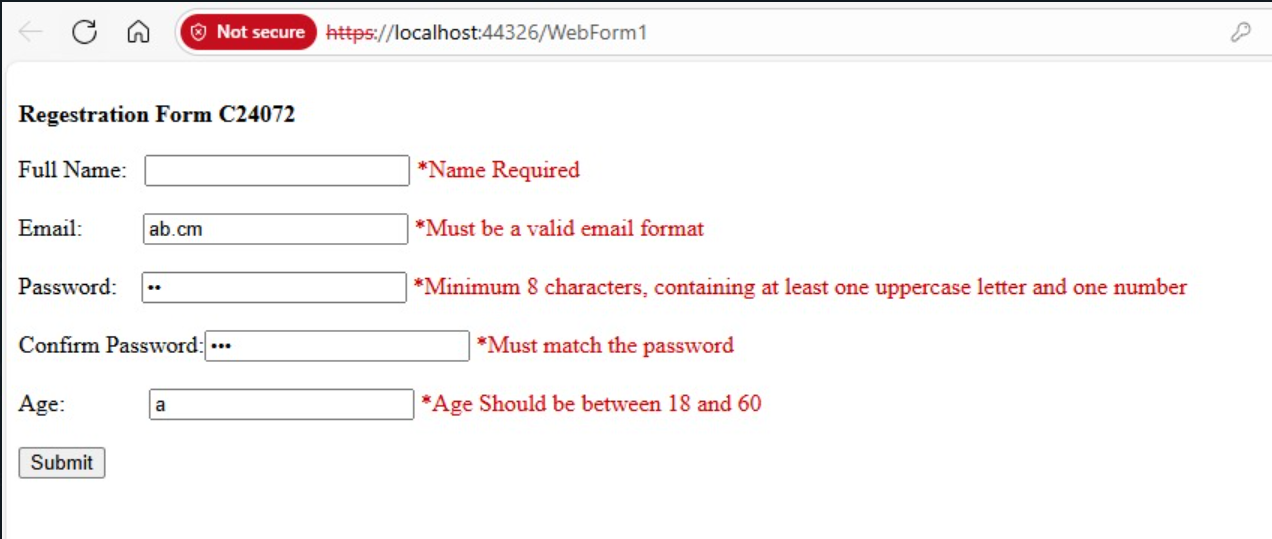
<p>

&nbsp;</p>

</body>

</html>

**Output:**



**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" CodeBehind="Site1.master.cs" Inherits="Practical\_No.\_2.Site1" %>

<!DOCTYPE html>

<html>

<head runat="server">

<title></title>

<asp:ContentPlaceHolder ID="head" runat="server">

</asp:ContentPlaceHolder>

<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**

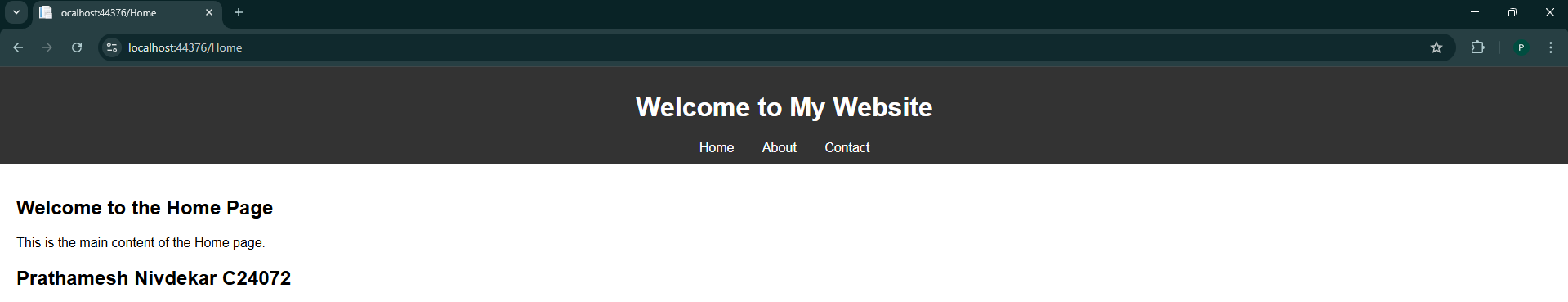
<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master" AutoEventWireup="true" CodeBehind="Home.aspx.cs" Inherits="Practical\_No.\_2.Home" %>

<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**

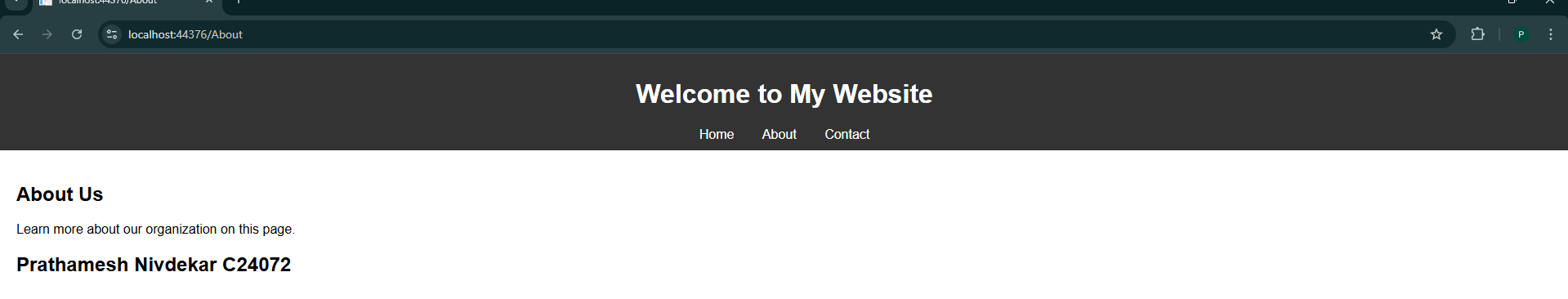
<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master" AutoEventWireup="true" CodeBehind="About.aspx.cs" Inherits="Practical\_No.\_2.About1" %>

<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**

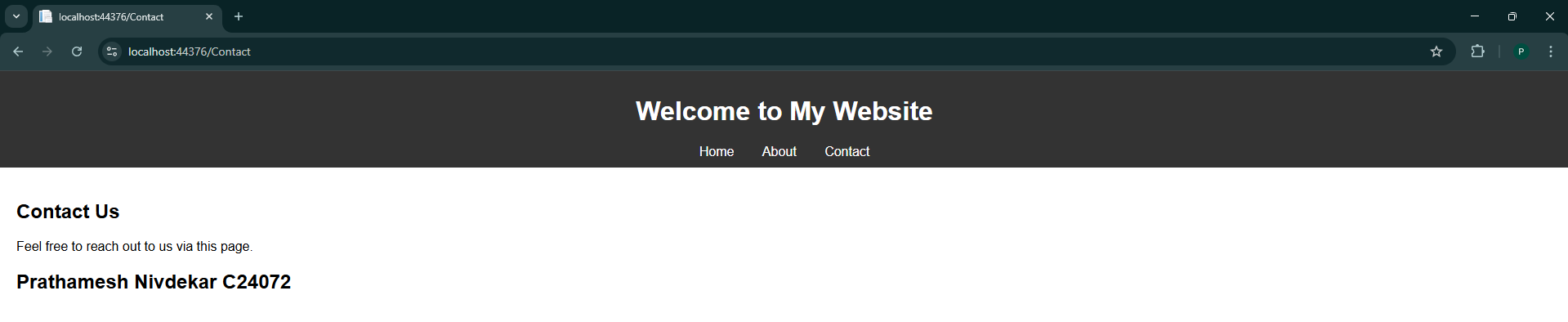
<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master" AutoEventWireup="true" CodeBehind="Contact.aspx.cs" Inherits="Practical\_No.\_2.Contact1" %>

<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**

**Aim: Design a Web Application using advanced controls.**

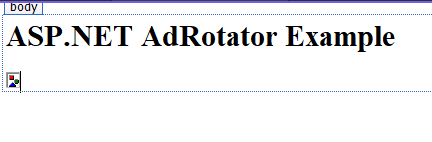
**1. AdRotator Control in ASP.NET**

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>

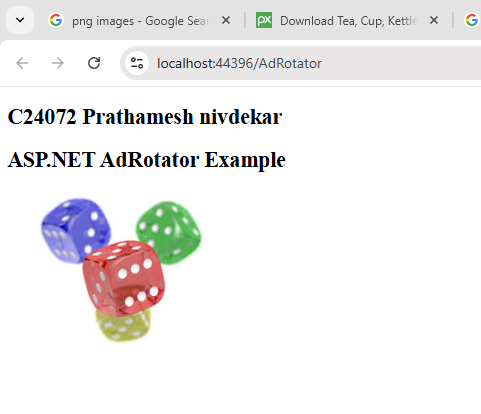
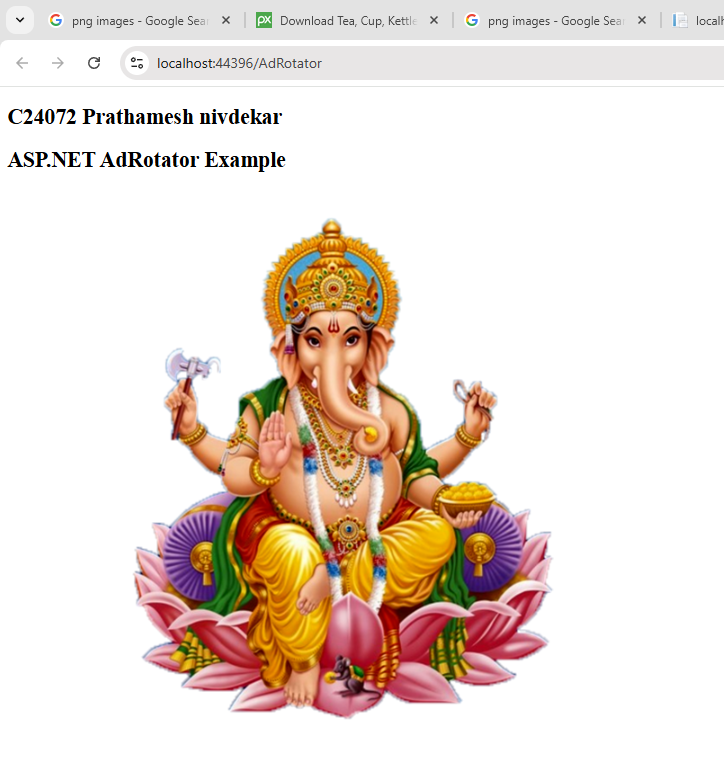
<NavigateUrl>https://www.example1.com</NavigateUrl>

<AlternateText>Second Ad</AlternateText>

<Impressions>30</Impressions>

</Ad>

</Advertisements>



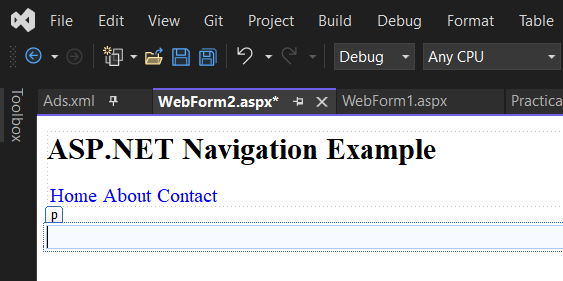
**2. Navigation Control (Menu Navigation)**

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"/>

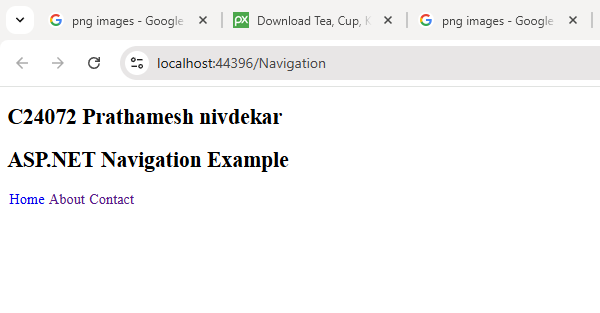
</Items>

</asp:Menu>

</form>

</body>

</html>



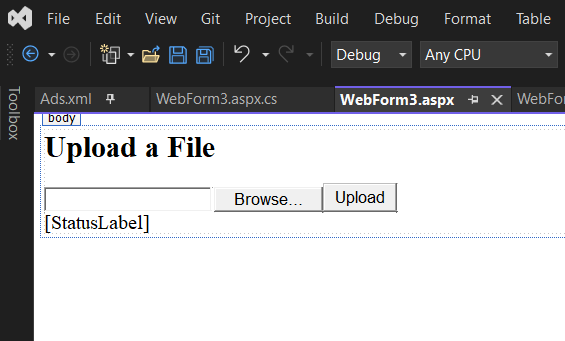
**3. File Upload in ASP.NET**

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

{

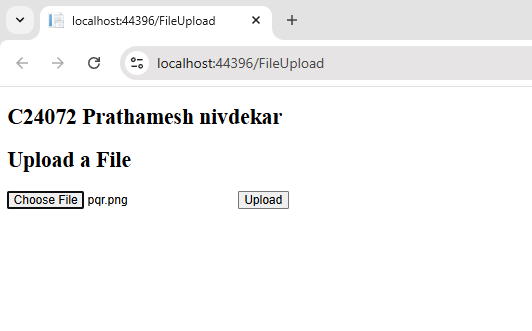
StatusLabel.Text = "Upload status: No file selected.";

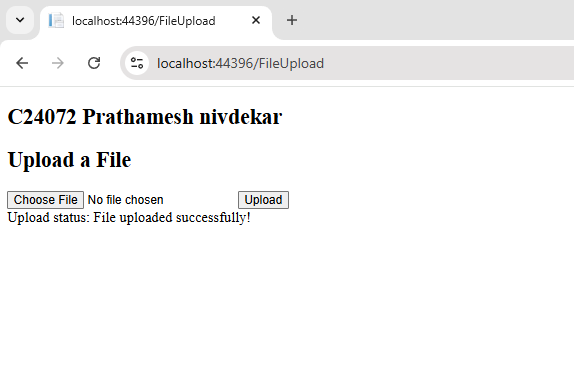
}

}

}

}

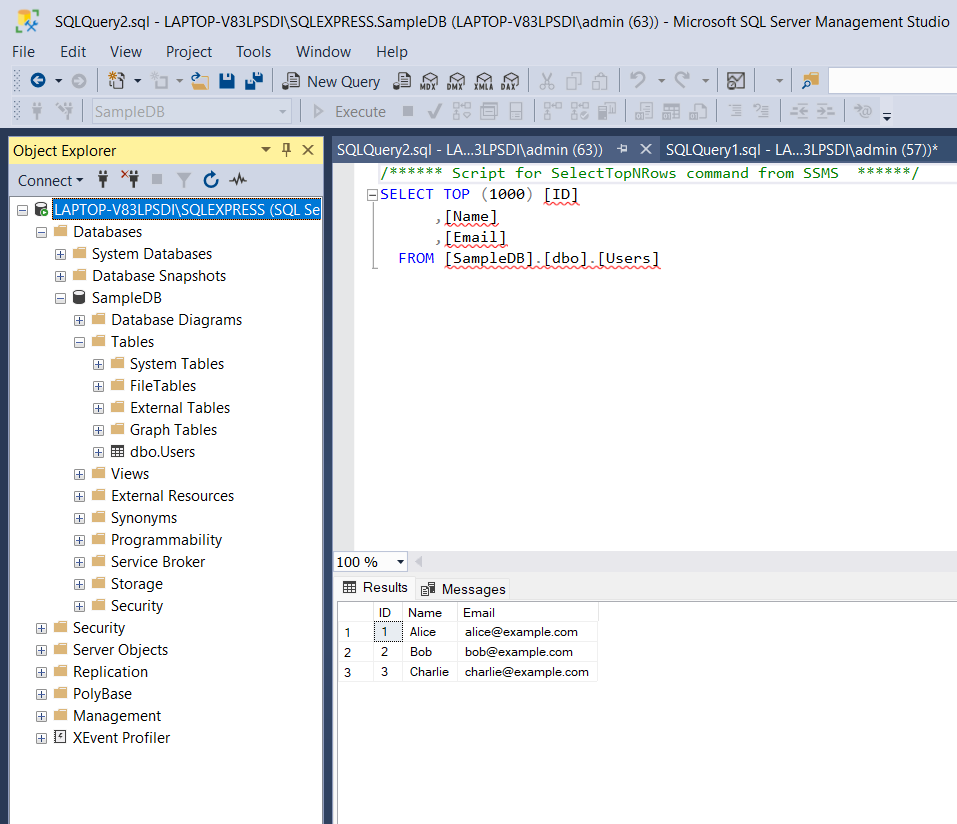




**Practical Number 4**

**:** **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)

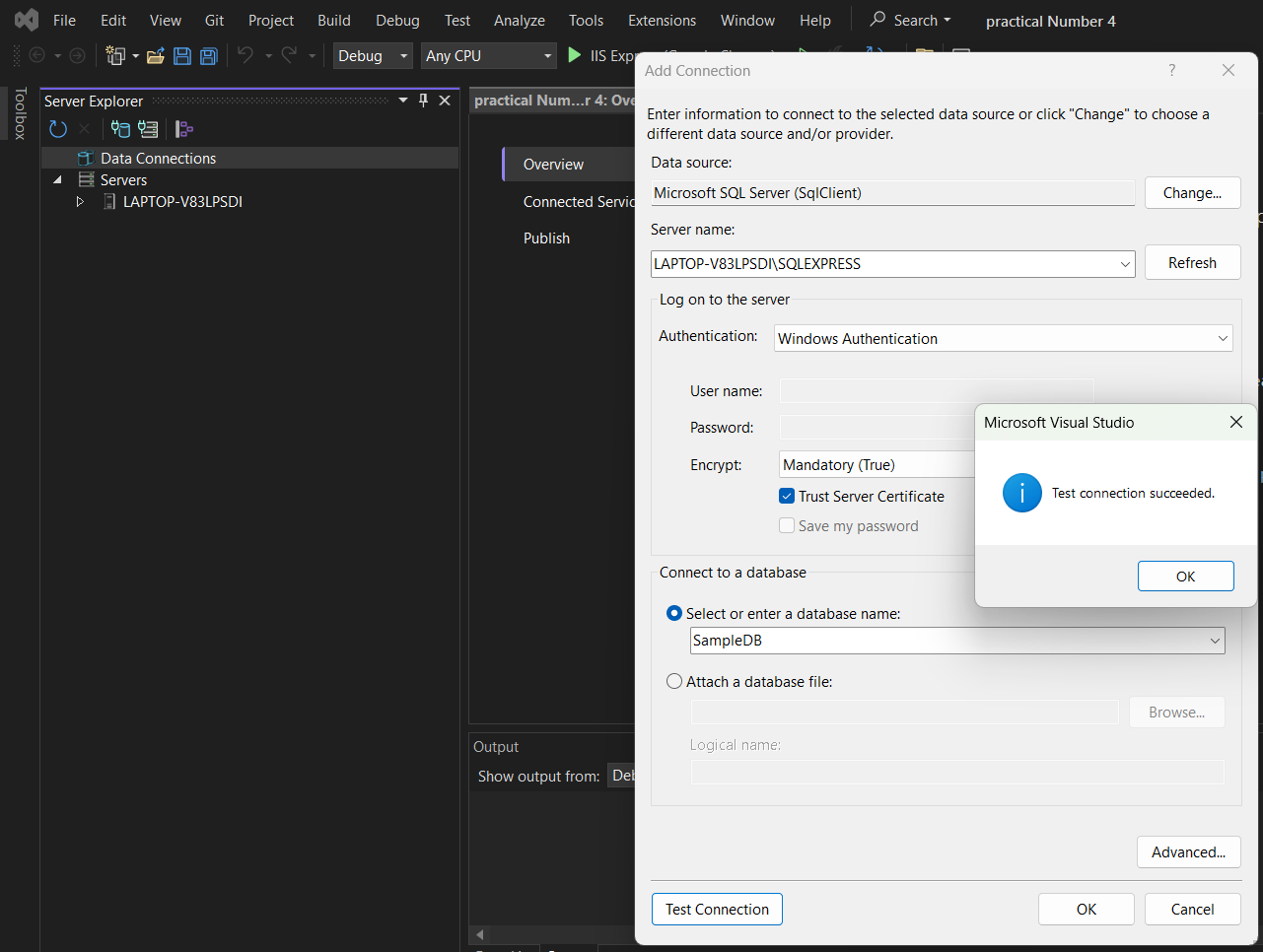
);

INSERT INTO Users (Name, Email) VALUES

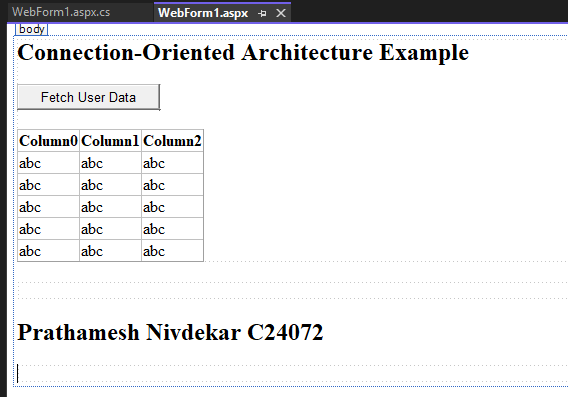
('Alice', 'alice@example.com'),

('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>");

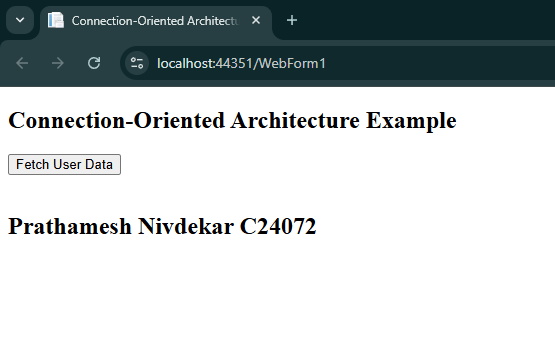
}

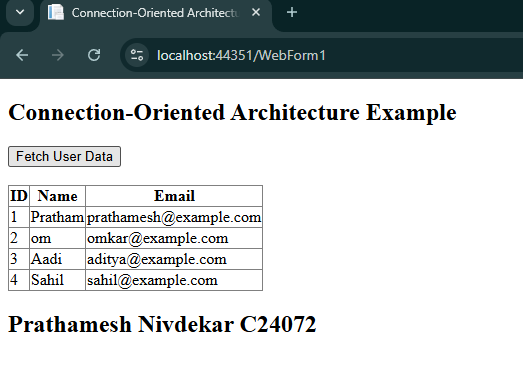
} // Connection closes au

}

}

}





**Practical Number 5**

**Aim: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)

);

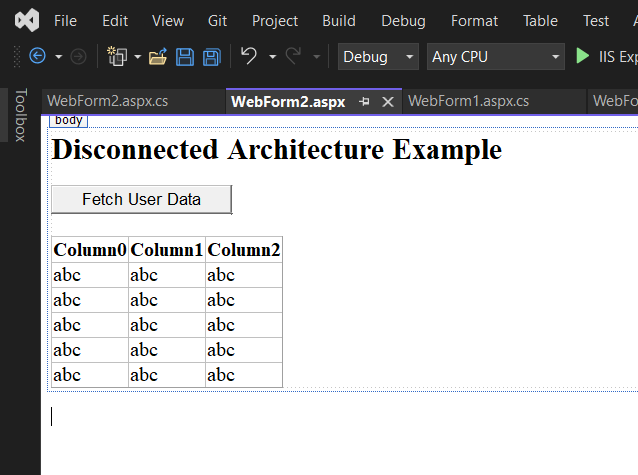
INSERT INTO Users (Name, Email) VALUES

('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-V83LPSDI\\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>");

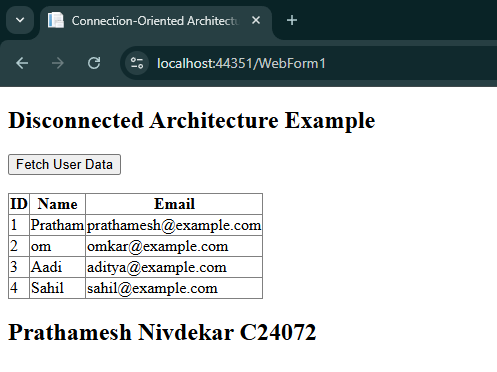
}

}

}

}

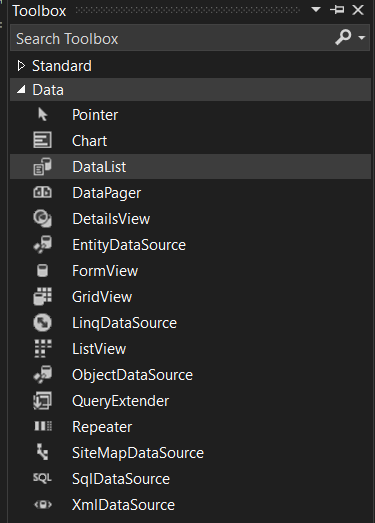




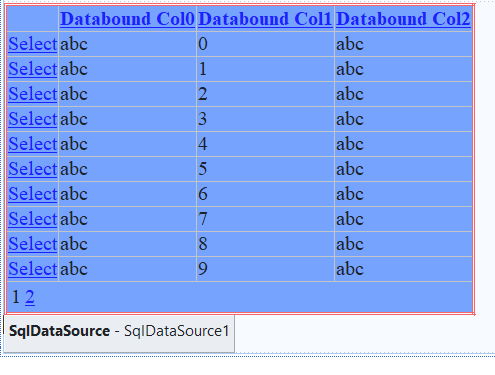
**Practical Number 6**

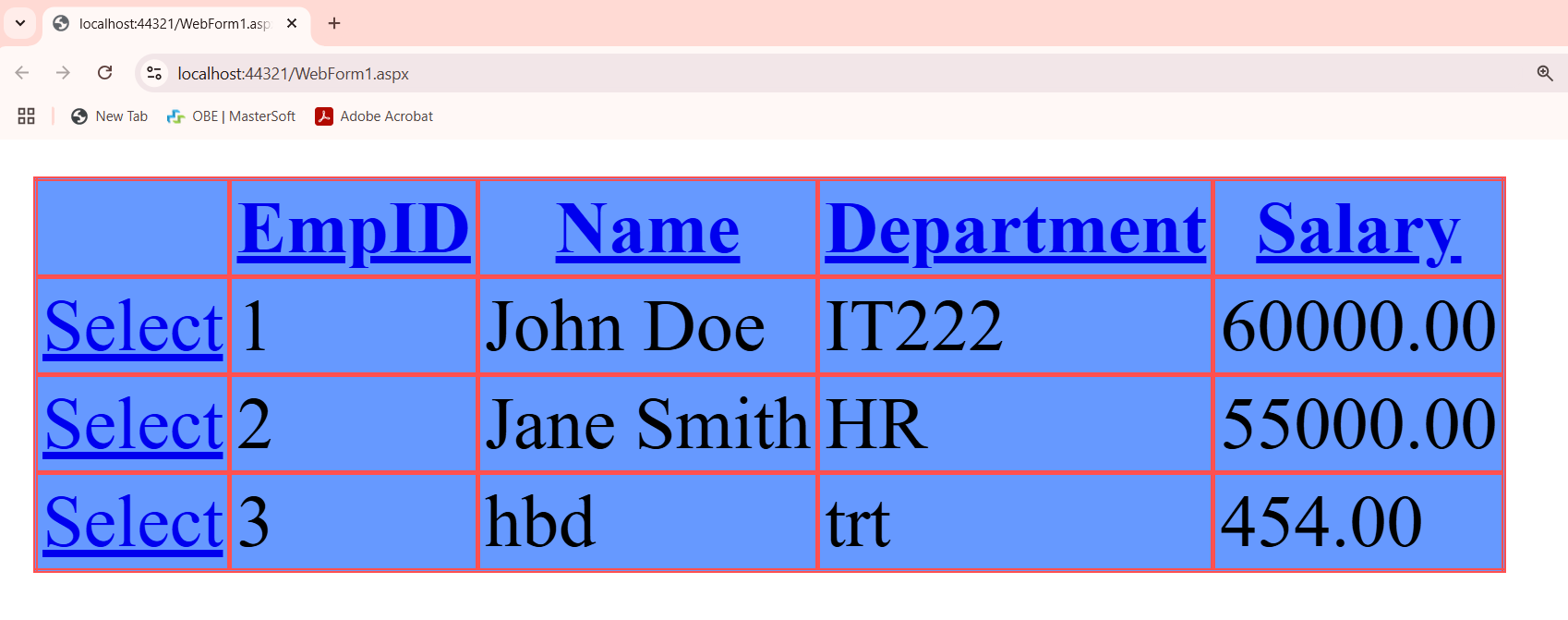
**Aim: Create a webpage that demonstrates the use of data bound controls of ASP.NET.**

Data Controls:



DataGridView:





**Practical No. 7**

**Aim: 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)**:

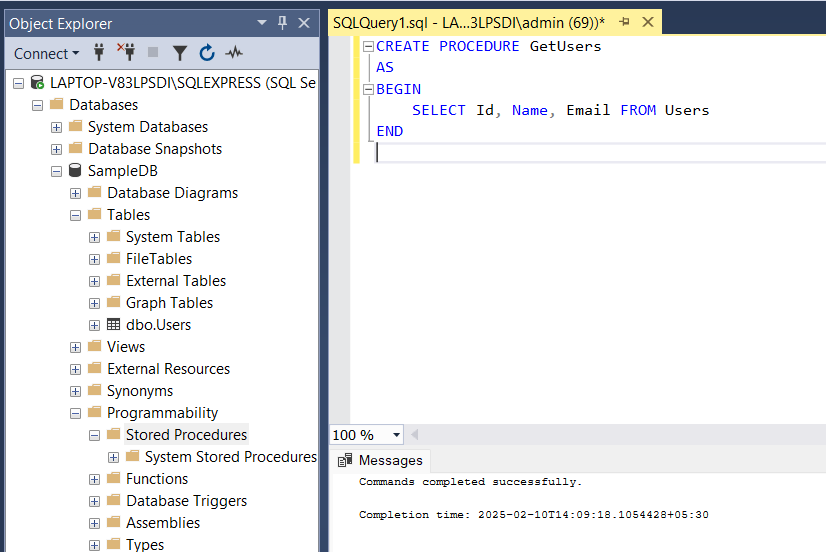
CREATE PROCEDURE GetUsers

AS

BEGIN

SELECT Id, Name, Email FROM Users

END

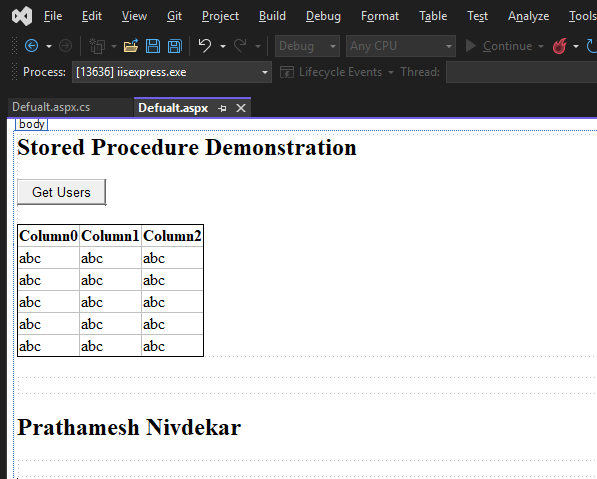


**2. Create an ASP.NET Web Application in Visual Studio**

* 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();

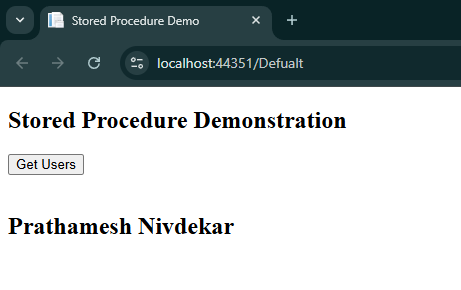
}

}

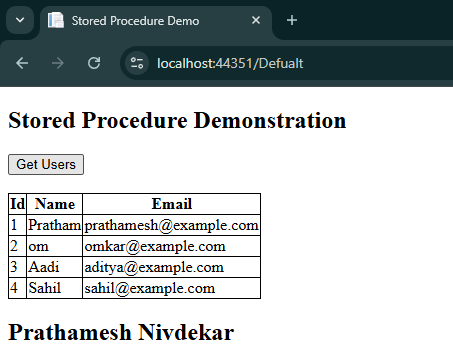
}

}

}



**OUTPUT:**



**Practical No. 8**

**Aim: 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.**

**1️ 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 ('Prathamesh', 'IT', 60000),

('Aditya', 'HR', 55000),

('Omkar', 'Developer', 10000),

('Sahil', 'IT', 3000);

-- Create Stored Procedure to Fetch Employee Data by Department

CREATE PROCEDURE GetEmployeesByDepartment

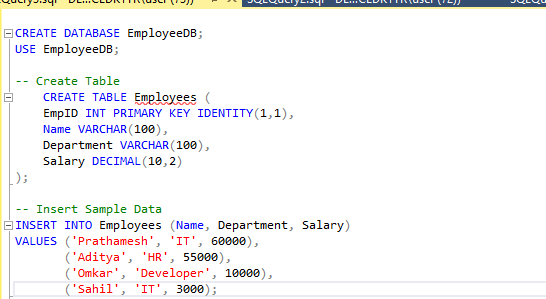
@DepartmentName VARCHAR(100)

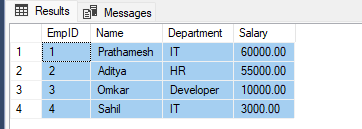
AS

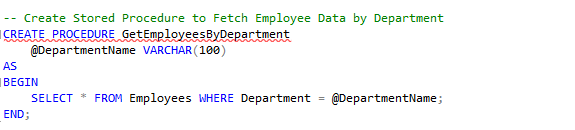
BEGIN

SELECT \* FROM Employees WHERE Department = @DepartmentName;

END;



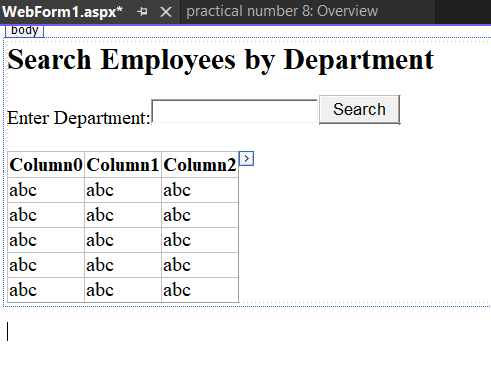




**2️ Create ASP.NET Web Form**

**🔹 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>

**3️ 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-V83LPSDI\\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;

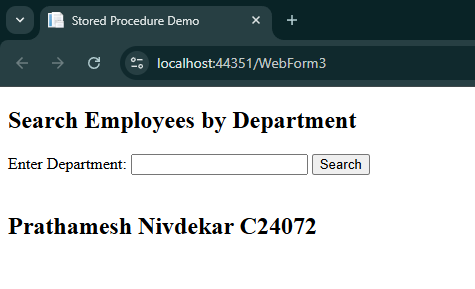
gvEmployees.DataBind();

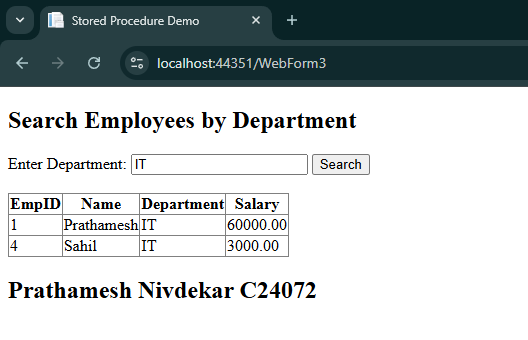
}

}

}

}

****



**Practical No.9**

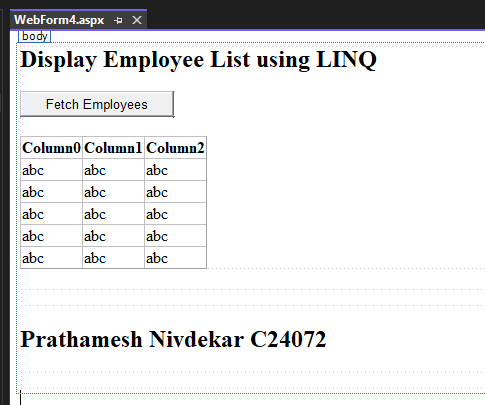
**Aim: Design a webpage to display the use of LINQ.**

**📌 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.

**1️ 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 />

<asp:GridView ID="gvEmployees" runat="server" AutoGenerateColumns="True"></asp:GridView>

</div>

</form>

</body>

</html>

**2️ 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

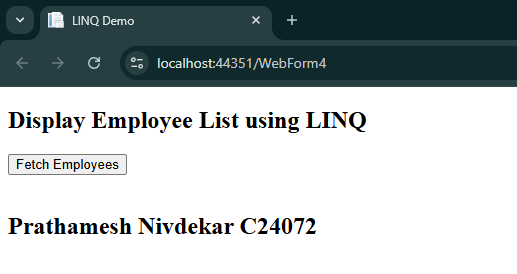
gvEmployees.DataSource = result.ToList();

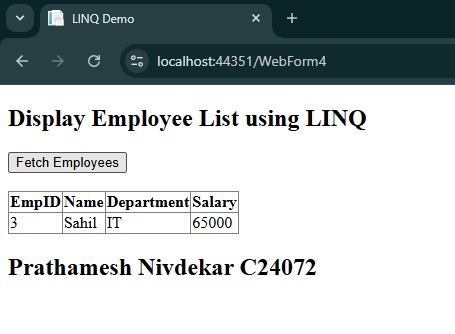
gvEmployees.DataBind();

}

}

}

****

****

**Practical No. 10**

**Aim: 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**

**1️ 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);

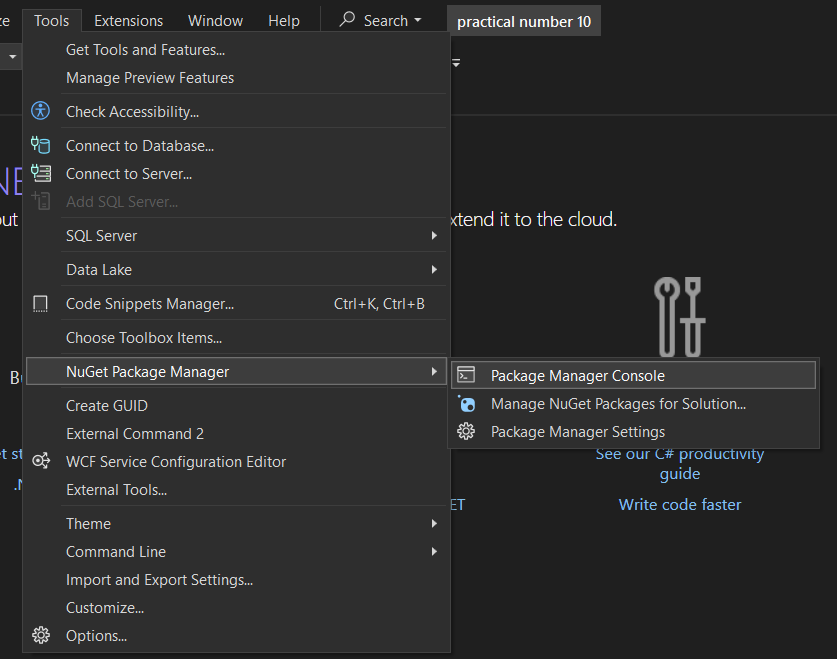
**2️ Create an ASP.NET Web Application**

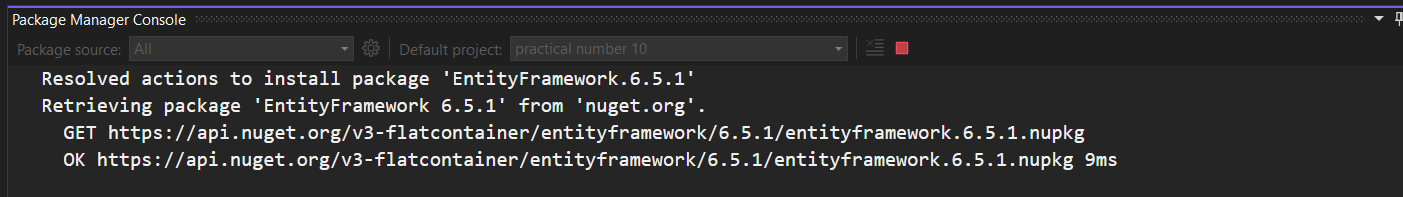
1. Open **Visual Studio** → Create a **New Project** → Choose **ASP.NET Web Application (.NET Framework)**
2. Select **Web Forms** and click **Create**

**3️ Install Entity Framework**

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

Install-Package EntityFramework

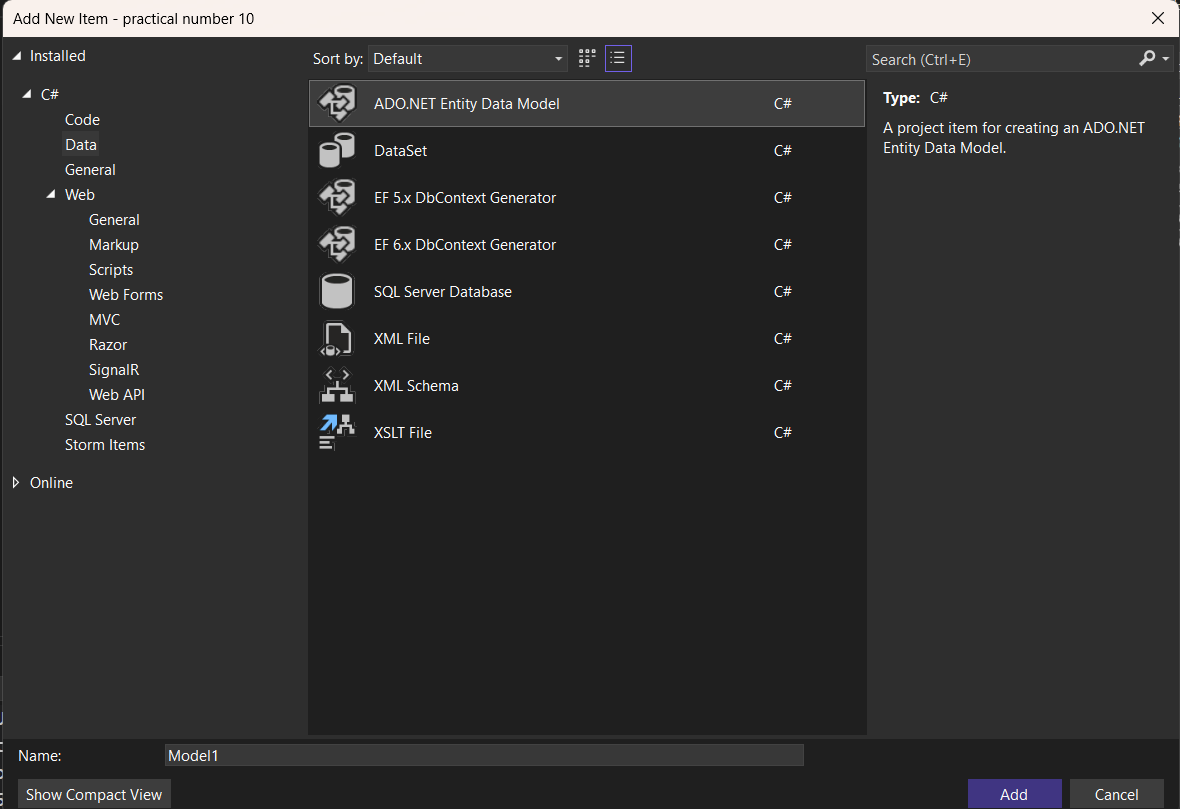


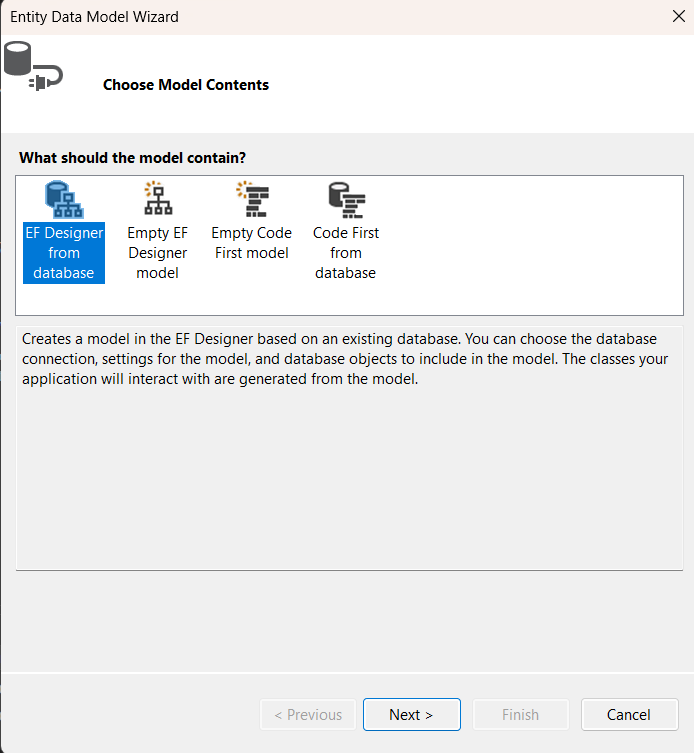


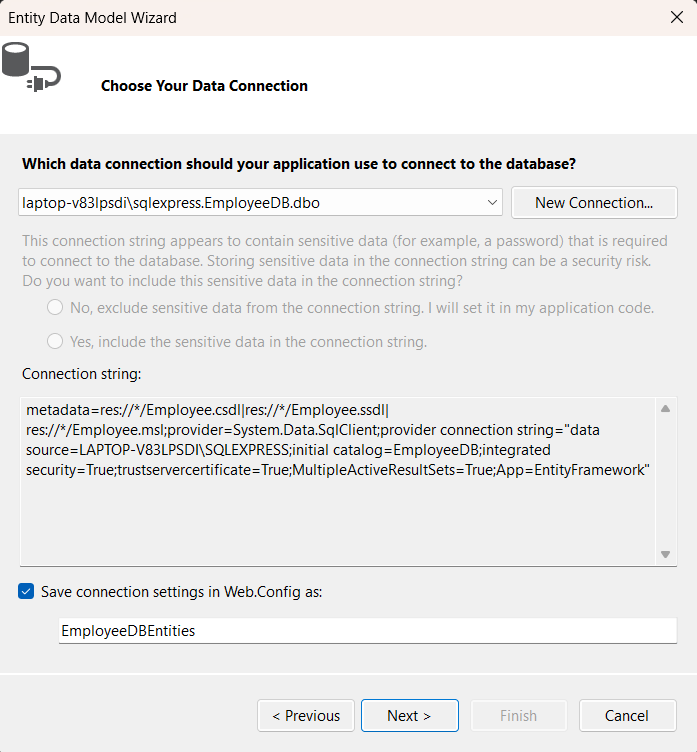
**4️ Create Entity Framework Model**

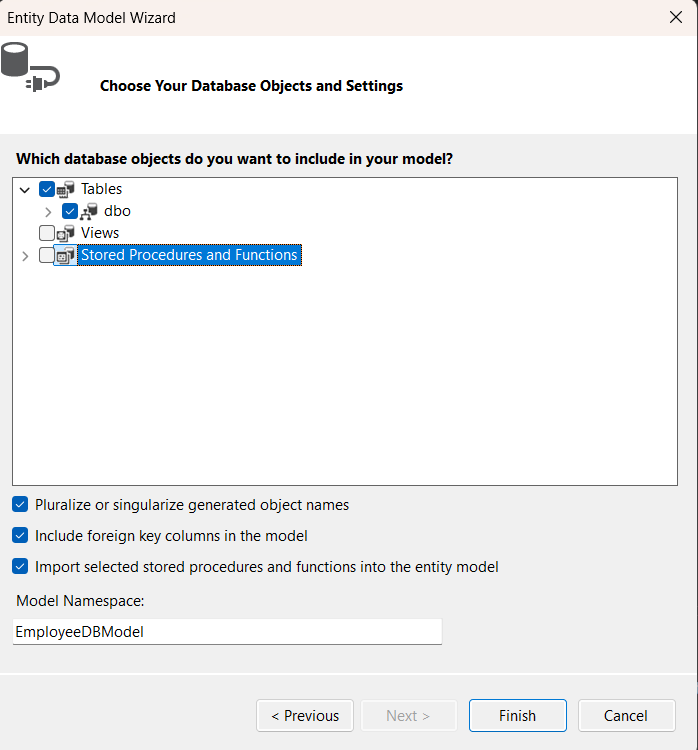
1. Right-click the project → Add → New Item → Select **ADO.NET Entity Data Model**
2. Choose **EF Designer from Database**
3. Select **"EmployeeDB"** as the database
4. Select the **Employees** table → Finish

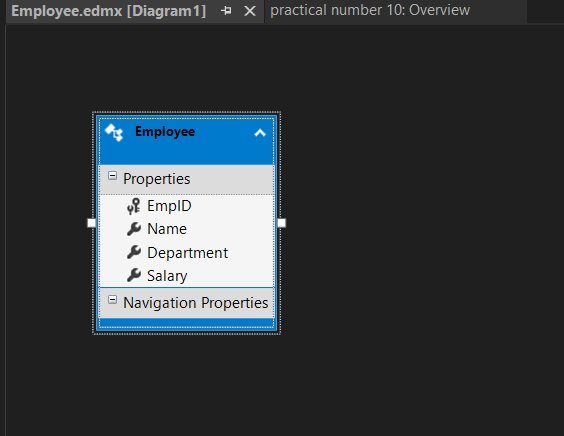
This generates the **Employee.cs model class**.











**5️ 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>

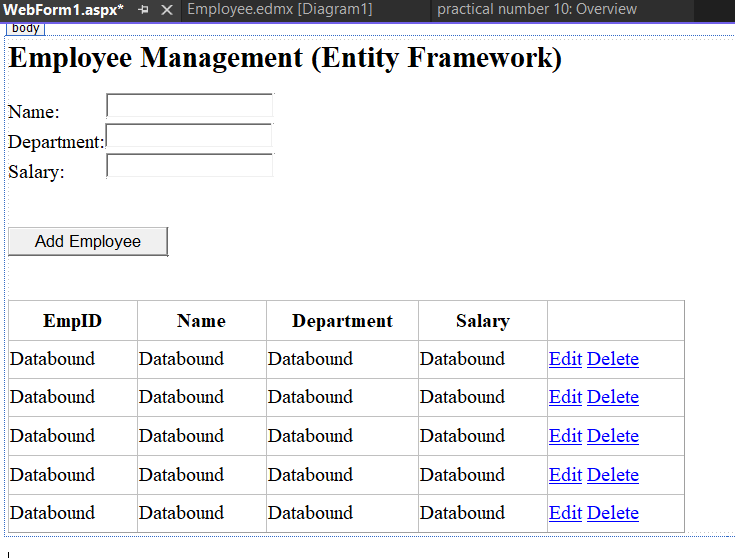
</asp:GridView>

</div>

</form>

</body>

</html>



**6️ 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, GridViewCancelEditEventArgs 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);

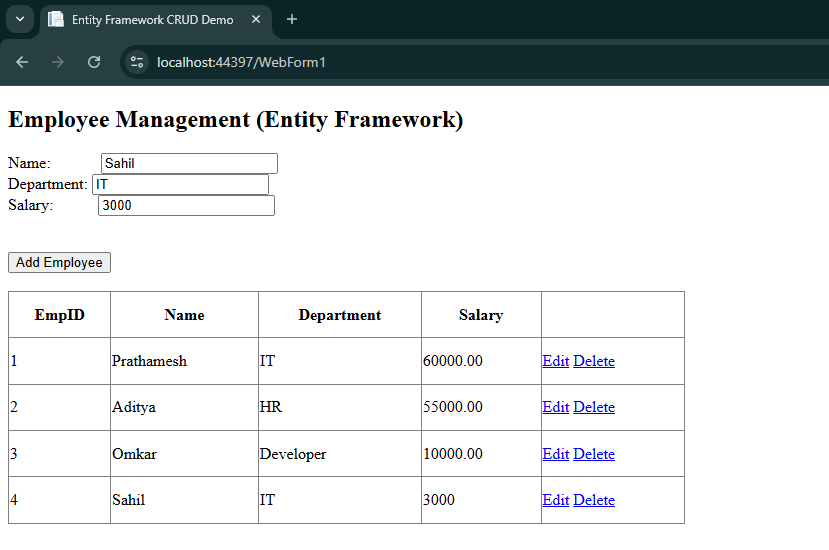
db.SaveChanges();

LoadEmployees();

}

}

}

****

**Practical Number 11**

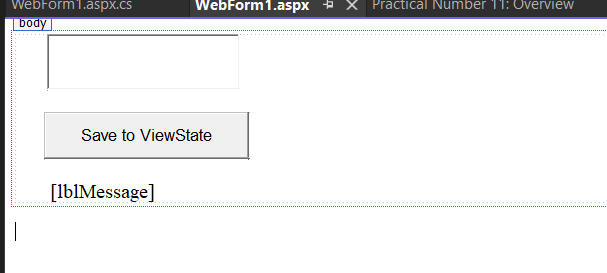
**Aim: Design Web Applications using Client-Side Session Management**

## **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.

## **1️ View State (ASP.NET Web Forms Only)**

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



📌 **File:** WebForm1.aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Practical\_Number\_11.WebForm1" %>

<!DOCTYPE html>

<html>

<head>

<title>View State Example</title>

</head>

<body>

<form runat="server">

<asp:TextBox ID="txtName" runat="server" Height="50px" style="margin-left: 32px" Width="185px"></asp:TextBox>

<br />

<br />

<asp:Button ID="btnSubmit" runat="server" Text="Save to ViewState" OnClick="btnSubmit\_Click" Height="47px" style="margin-left: 29px" Width="205px" />

<br />

<br />

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<asp:Label ID="lblMessage" runat="server"></asp:Label>

</form>

</body>

</html>

📌 **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)

{

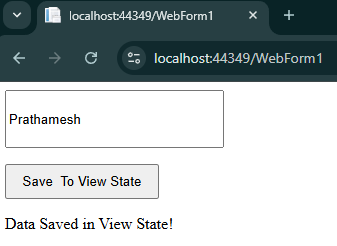
ViewState["UserName"] = txtName.Text;

lblMessage.Text = "Data Saved in View State!";

}

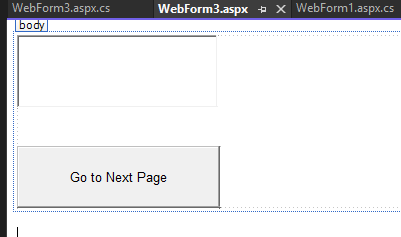
}

}



## **2️ 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)

{

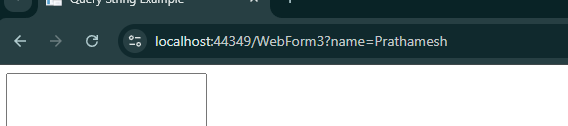
Response.Redirect("WebForm3.aspx?name=" + txtName.Text);

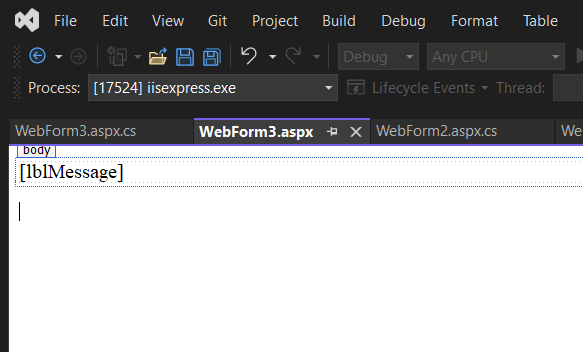
}

}

}

**Output:**





📌 **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)

{

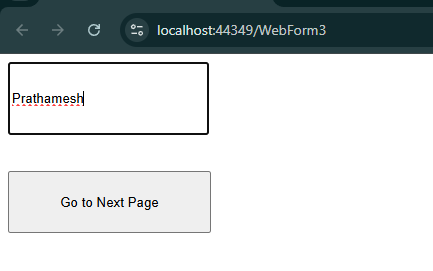
lblMessage.Text = "Query String Value: " + Request.QueryString["name"];

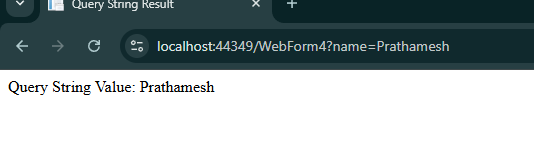
}

}

}

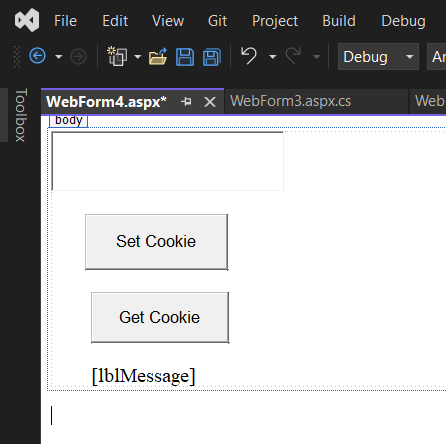
}





## **3️ 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 />

<br />

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<asp:Label ID="lblMessage" runat="server"></asp:Label>

</form>

</body>

</html>

📌 **File:** WebForm4.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 WebForm4 : System.Web.UI.Page

{

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

{

}

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

{

HttpCookie cookie = new HttpCookie("UserName", txtName.Text);

cookie.Expires = DateTime.Now.AddDays(7); // Cookie valid for 7 days

Response.Cookies.Add(cookie);

lblMessage.Text = "Cookie set successfully!";

}

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

{

HttpCookie cookie = Request.Cookies["UserName"];

if (cookie != null)

{

lblMessage.Text = "Stored Cookie Value: " + cookie.Value;

}

else

{

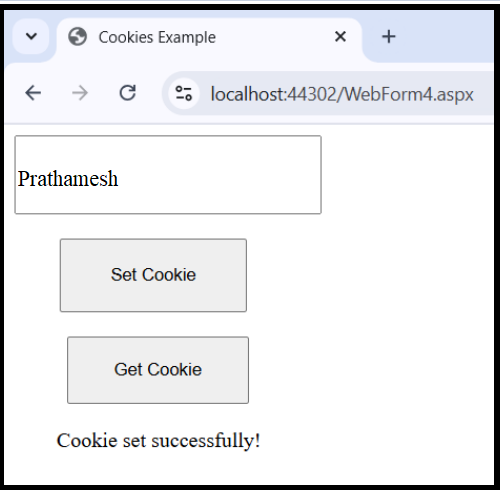
lblMessage.Text = "No Cookie Found!";

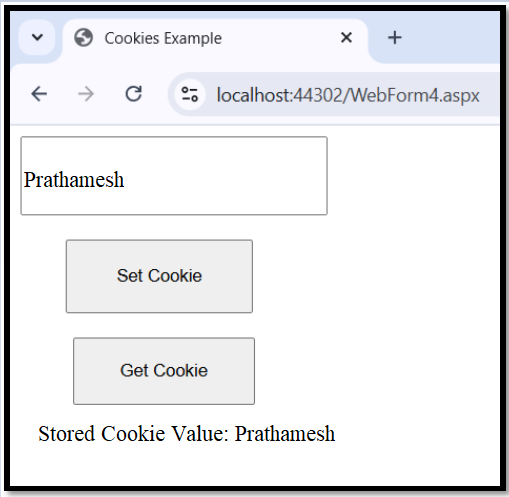
}

}

}

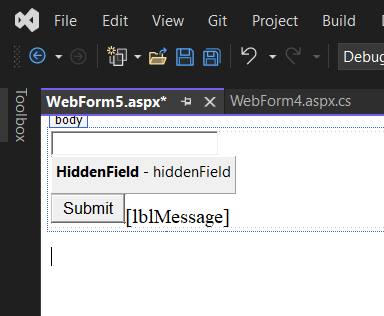
}





## **4️ 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)

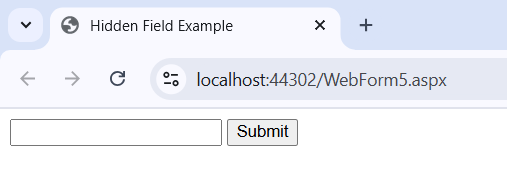
{

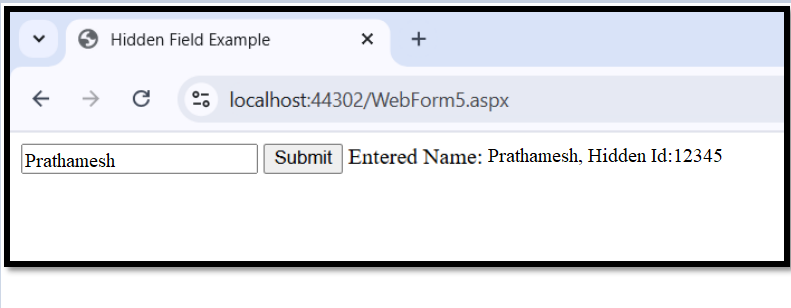
lblMessage.Text = "Entered Name: " + txtName.Text + ", Hidden ID: " + hiddenField.Value;

}

}

}





**Practical Number 12**

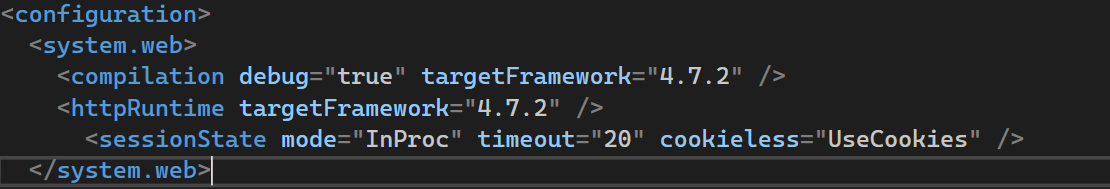
**Aim: Design Web Applications using Server Side Session Management Techniques**

**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**

****

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**:

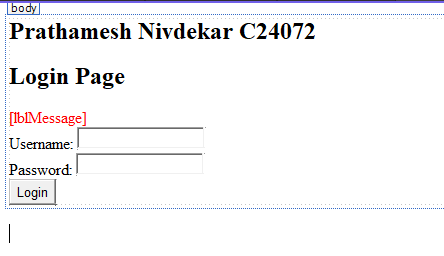
<sessionState mode="StateServer" stateConnectionString="tcpip=127.0.0.1:42424" timeout="20"/>

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="Practical\_No.\_12.Login" %>

<!DOCTYPE html>

<html lang="en">

<head runat="server">

<title>Login</title>

</head>

<body>

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

<div>

<h2>Prathamesh Nivdekar C24072</h2>

<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.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Practical\_No.\_12

{

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 == "Prathamesh" && password == "1234")

{

Session["Username"] = username;

Response.Redirect("Dashboard.aspx");

}

else

{

lblMessage.Text = "Invalid username or password!";

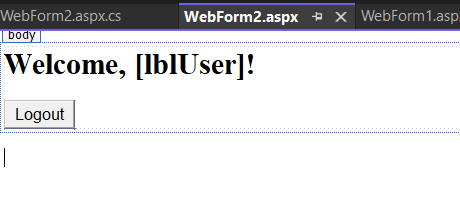
}

}

}

}

**4. Create a Dashboard Page (Dashboard.aspx)**

****

**Frontend: Dashboard.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Dashboard.aspx.cs" Inherits="Practical\_No.\_12.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.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Practical\_No.\_12

{

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();

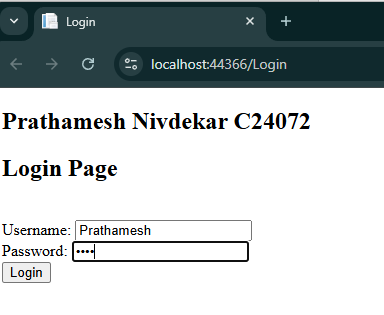
Response.Redirect("Login.aspx");

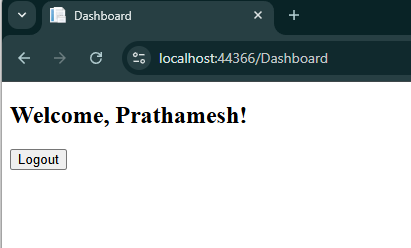
}

}

}

**Output:**





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**

**Aim: 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="Practical\_No\_13.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" />

<br />

<br />

<h2>Prathamesh Nivdekar C24072</h2>

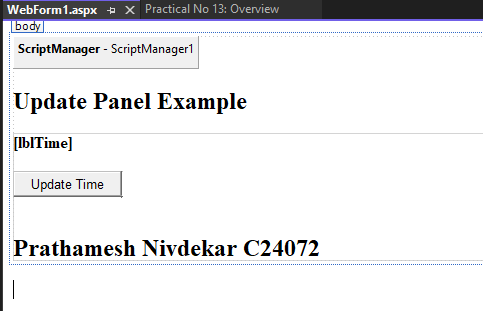
</ContentTemplate>

</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 Practical\_No\_13

{

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");

}

}

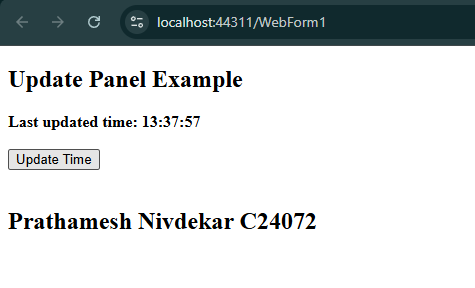
protected void btnUpdate\_Click(object sender, EventArgs e)

{

lblTime.Text = "Last updated time: " + DateTime.Now.ToString("HH:mm:ss");

}

}

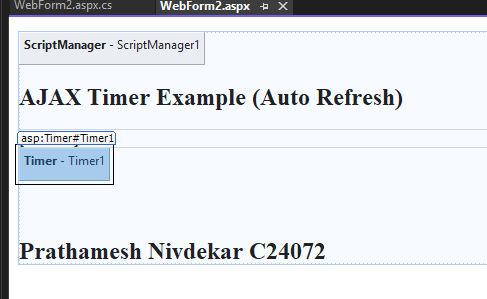
} 

**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="Practical\_No\_13.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="1000" OnTick="Timer1\_Tick" />

<br />

<br />

<h2>Prathamesh Nivdekar C24072</h2>

</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 Practical\_No\_13

{

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)

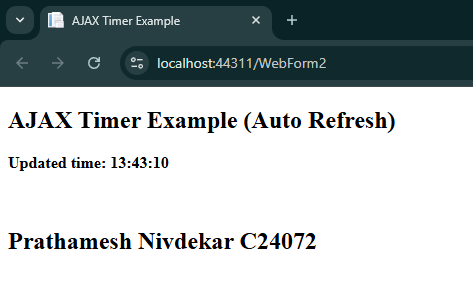
{

lblTime.Text = "Updated time: " + DateTime.Now.ToString("HH:mm:ss");

}

}

}



**Practical Number 14**

**Aim: 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;

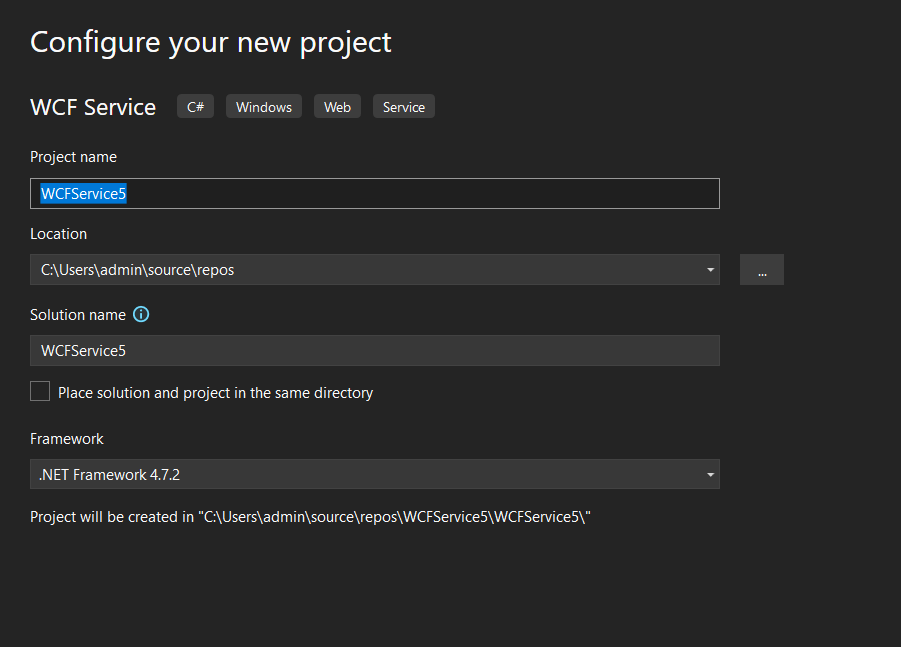
}

}

}

**Practical Number 15**

**Aim: Build a web application to create and use WCF service in ASP.net**



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; }

}

}

Servics.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)

{

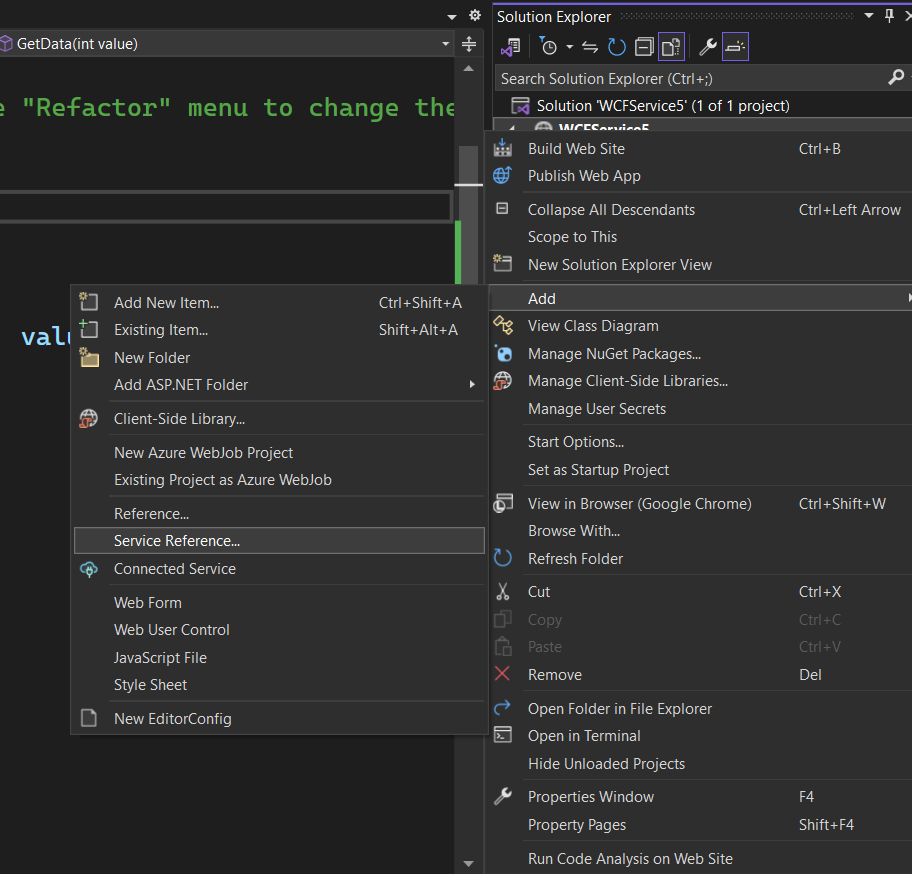
composite.StringValue += "Suffix";

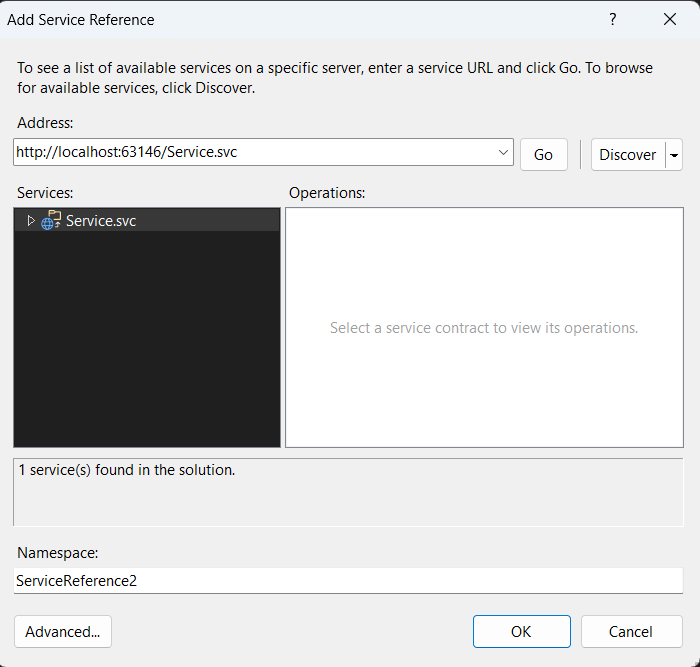
}

return composite;

}

}





WebForm.cs

<%@ 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>

<h2>Prathamesh Nivdekar C24072</h2>

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

<div>

Calculator suing WCF<br />

<br />

First Number:<asp:TextBox ID="TextBox1" runat="server" style="margin-bottom: 0px"></asp:TextBox>

<br />

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

<br />

<br />

<asp:Button ID="Button1" runat="server" OnClick="Button1\_Click" Text="ADD" />

<asp:Button ID="Button2" runat="server" OnClick="Button2\_Click" Text="SUB" />

<asp:Button ID="Button3" runat="server" OnClick="Button3\_Click" Text="MUL" />

<asp:Button ID="Button4" runat="server" OnClick="Button4\_Click" Text="DIV" />

<br />

<br />

Result:

<asp:Label ID="Label1" runat="server"></asp:Label>

</div>

</form>

</body>

</html>



<%@ 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 />

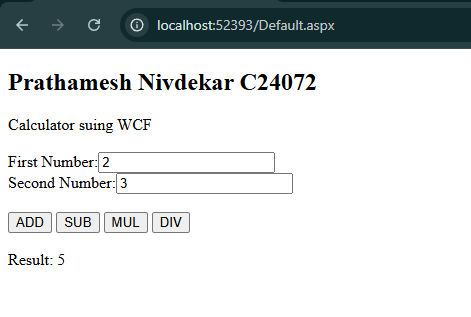
</div>

</form>

</body>

</html>

OUTPUT:



#### Practical No. 16

#### Aim: 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.

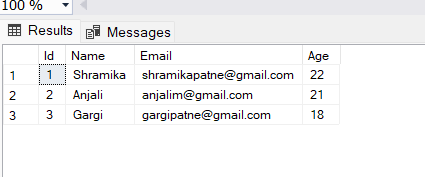
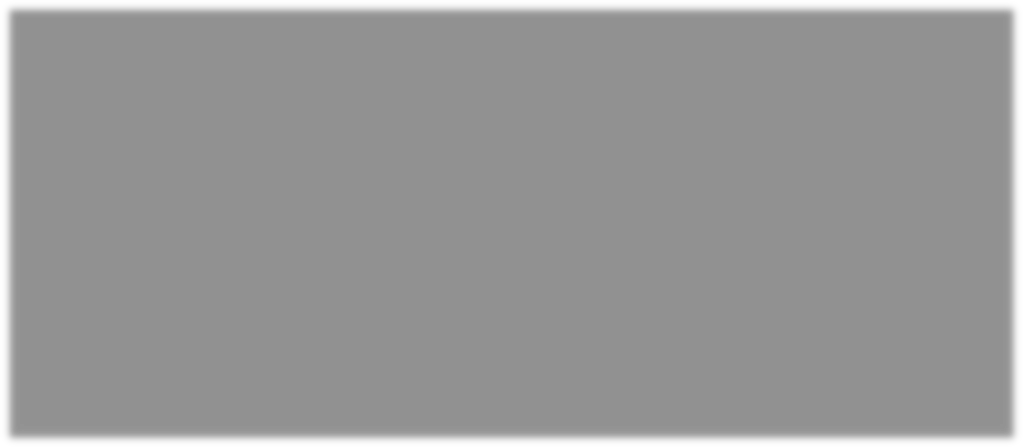
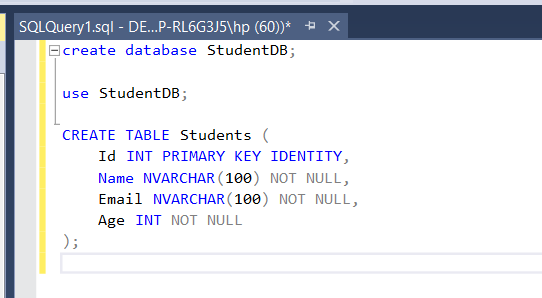
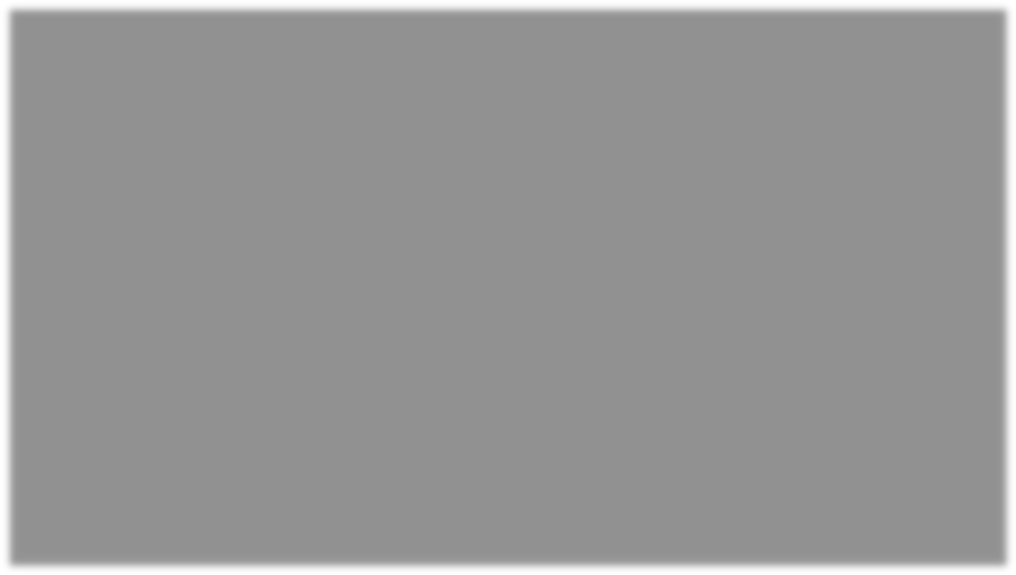
**Create a Database** named StudentDB.

**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



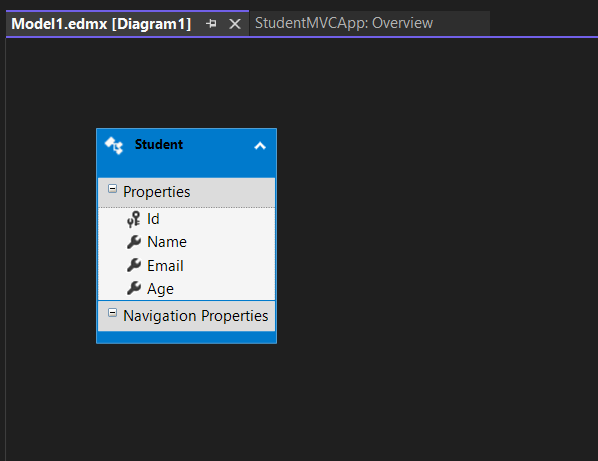
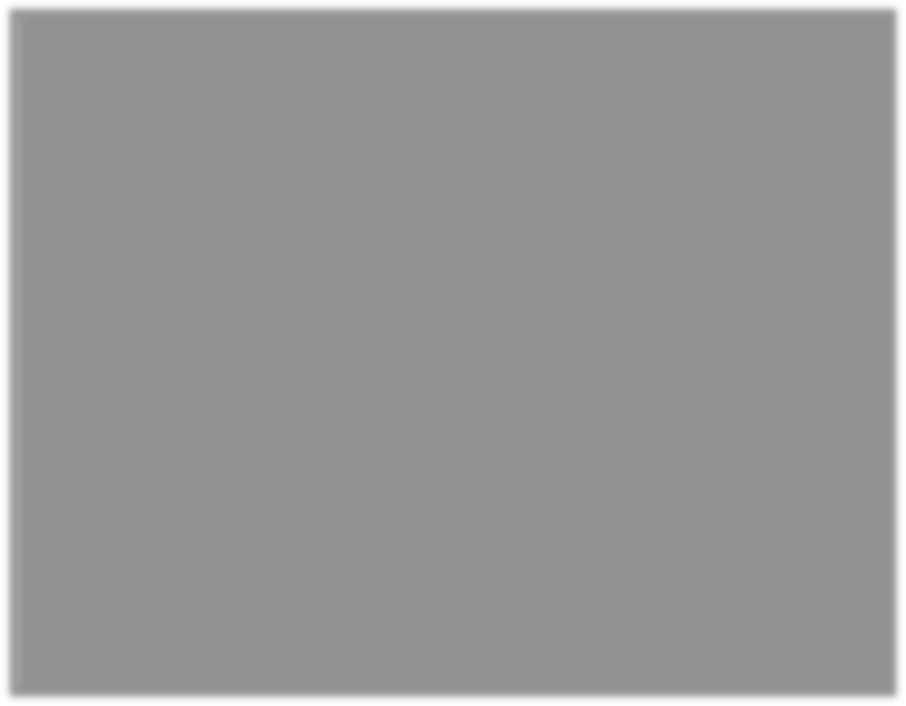
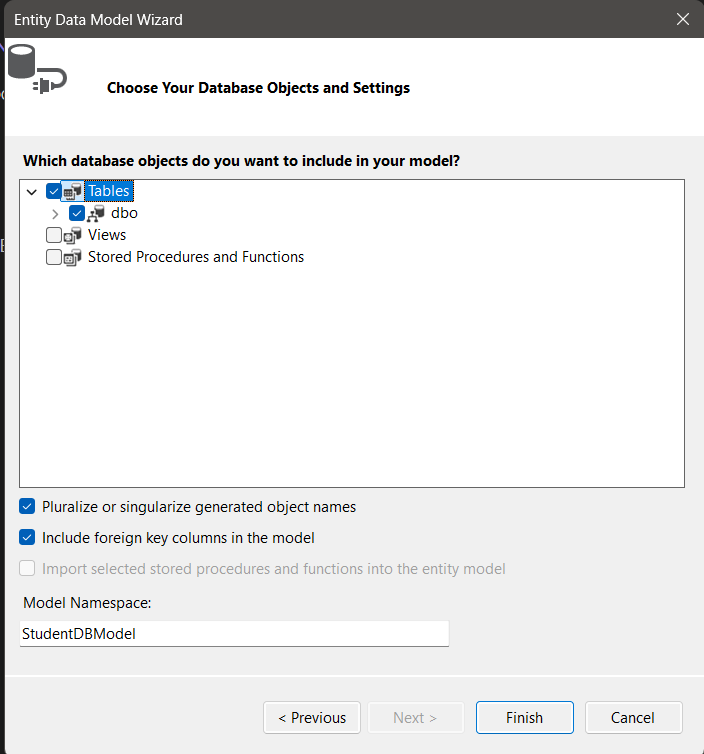
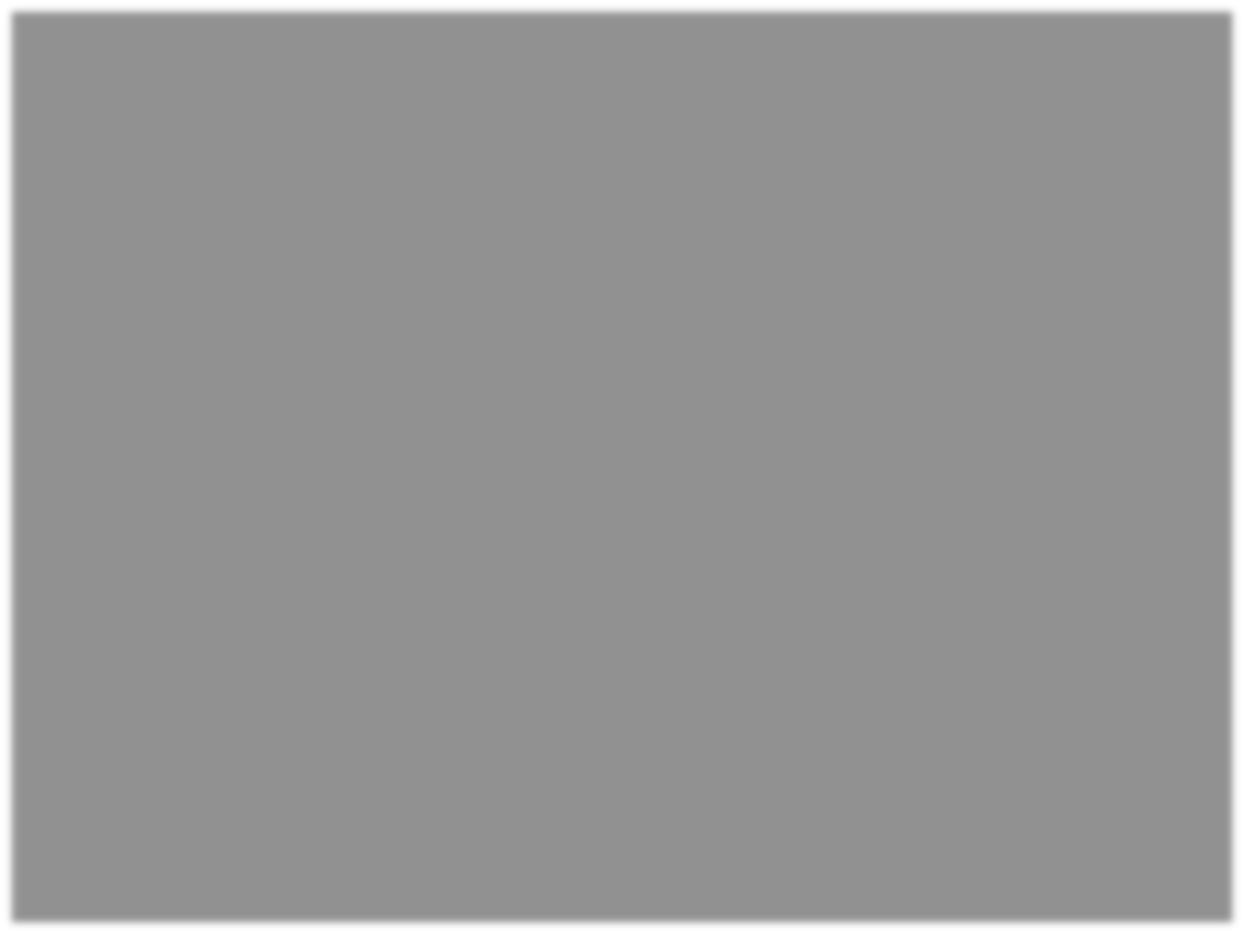
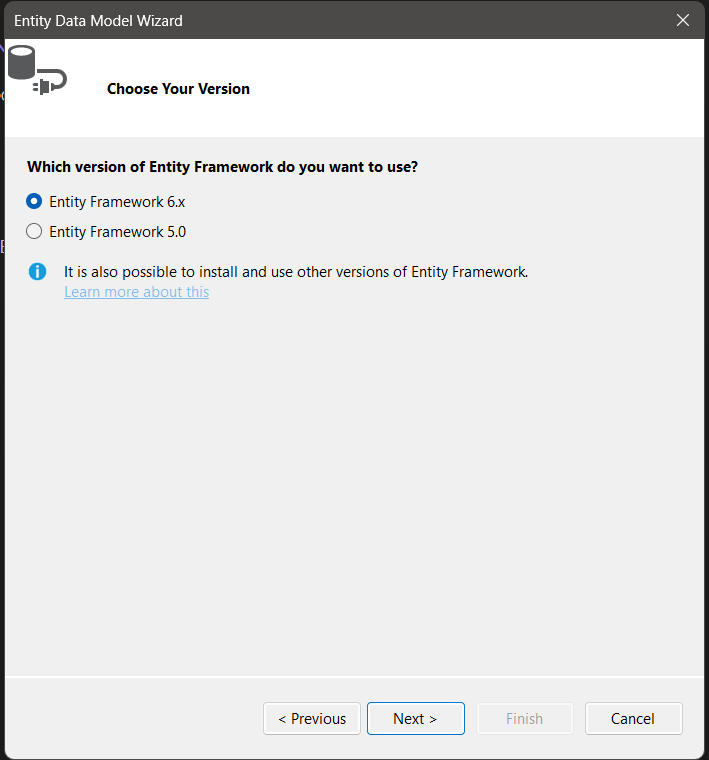
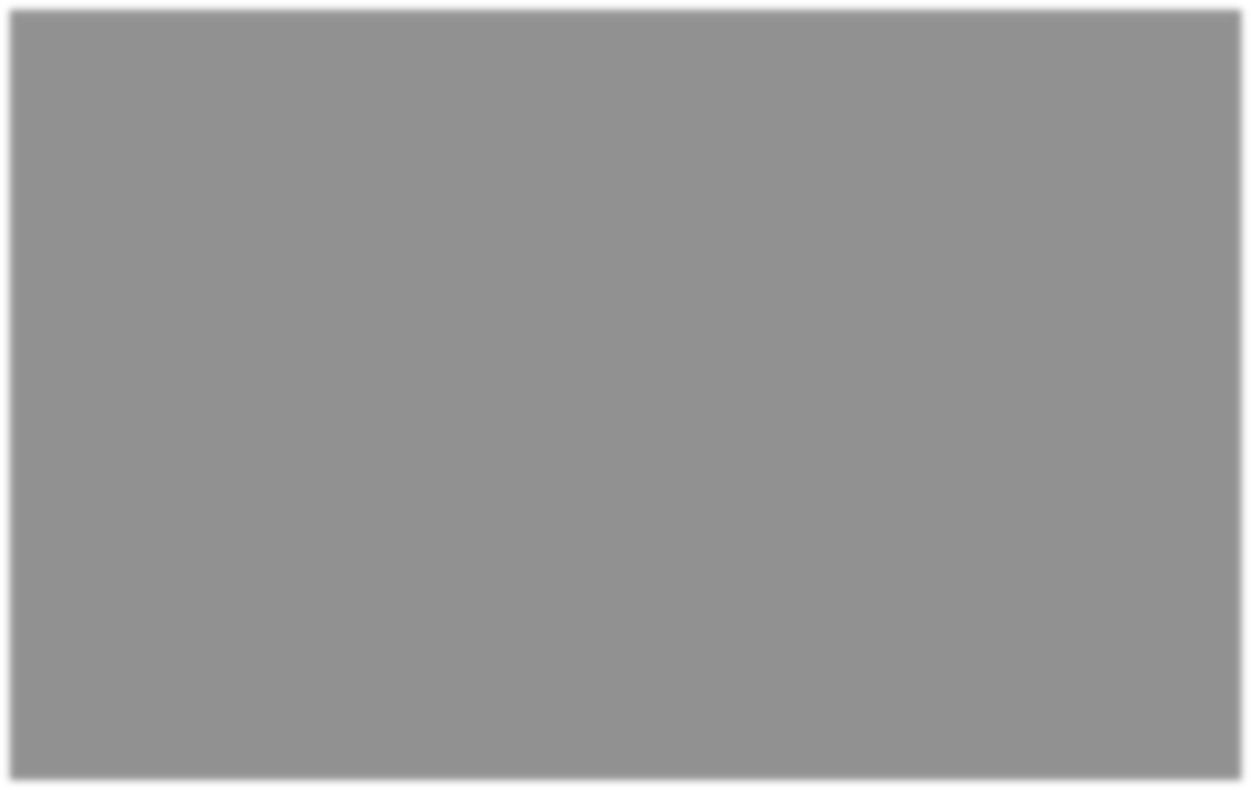
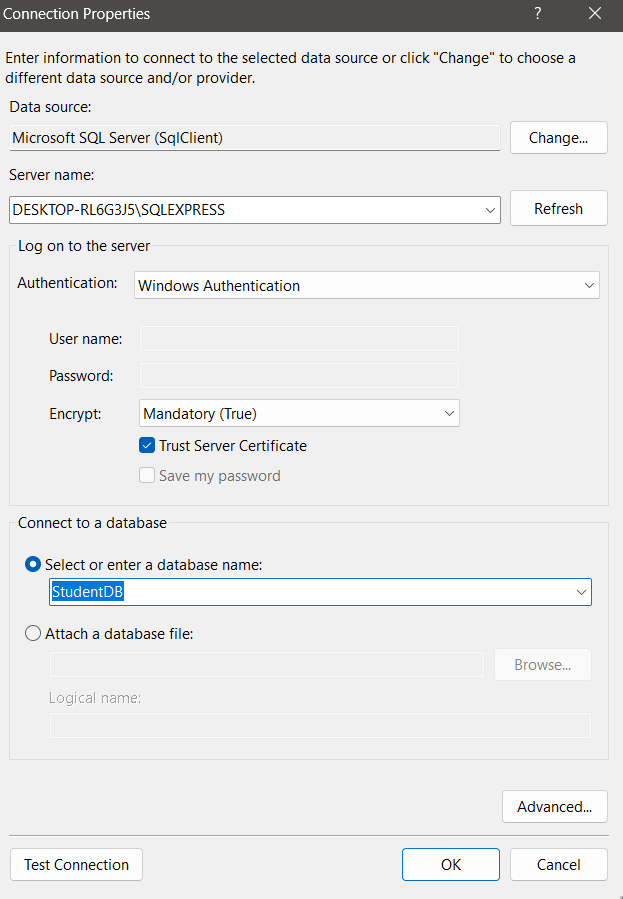
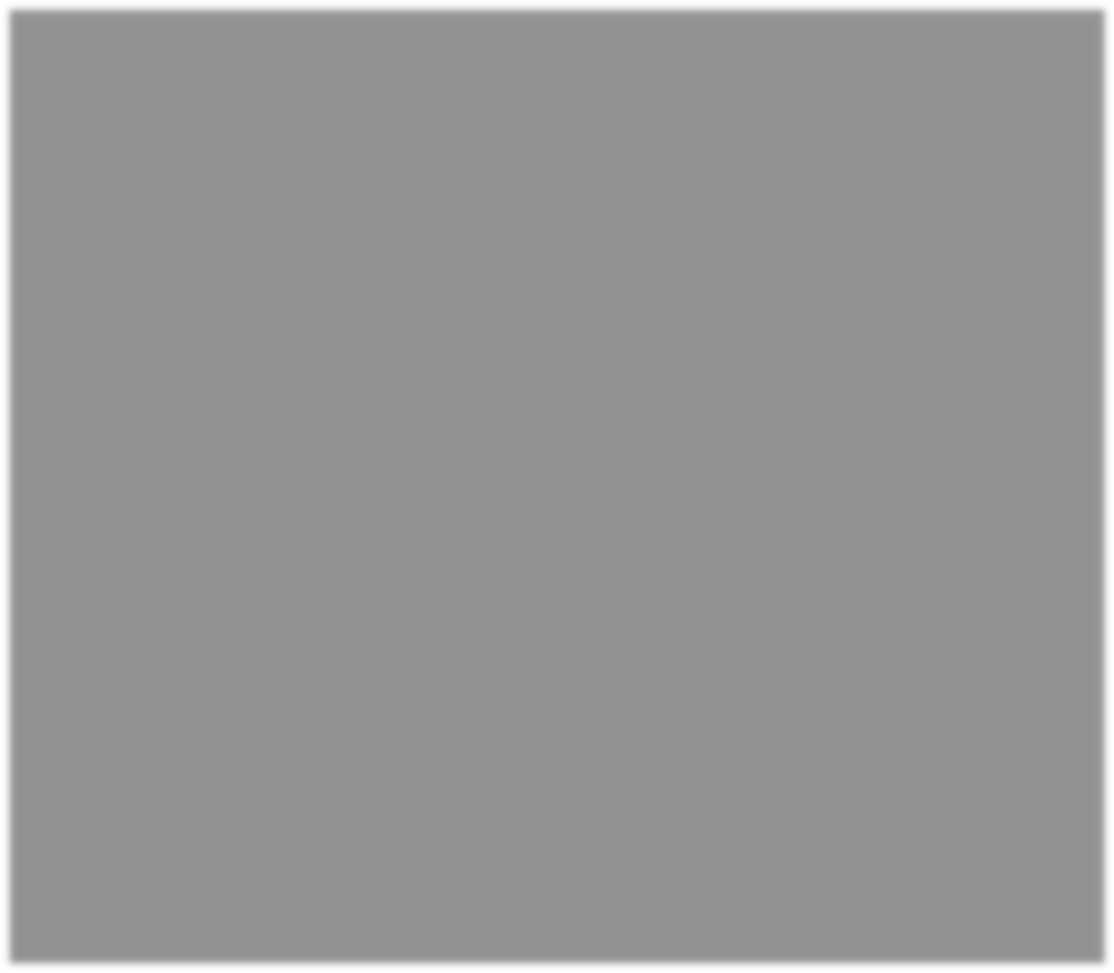
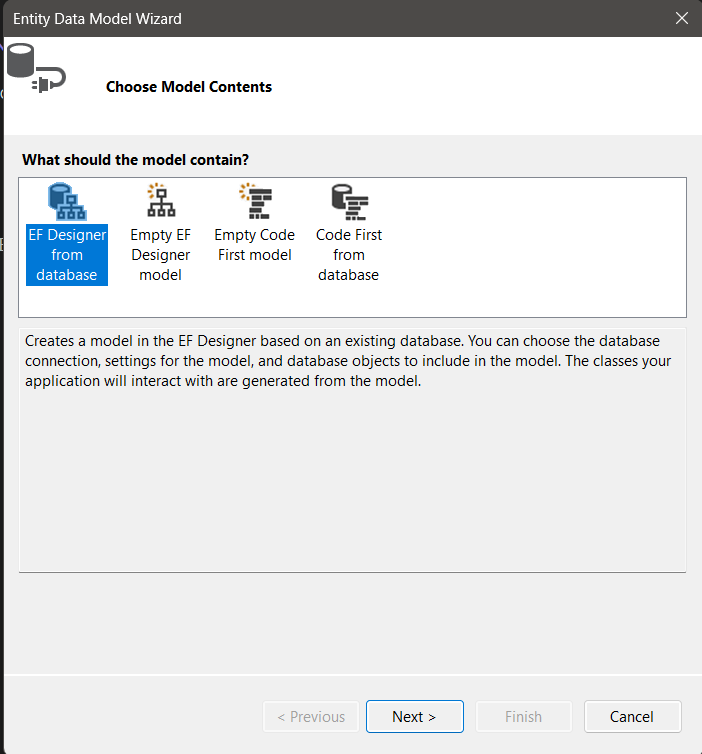
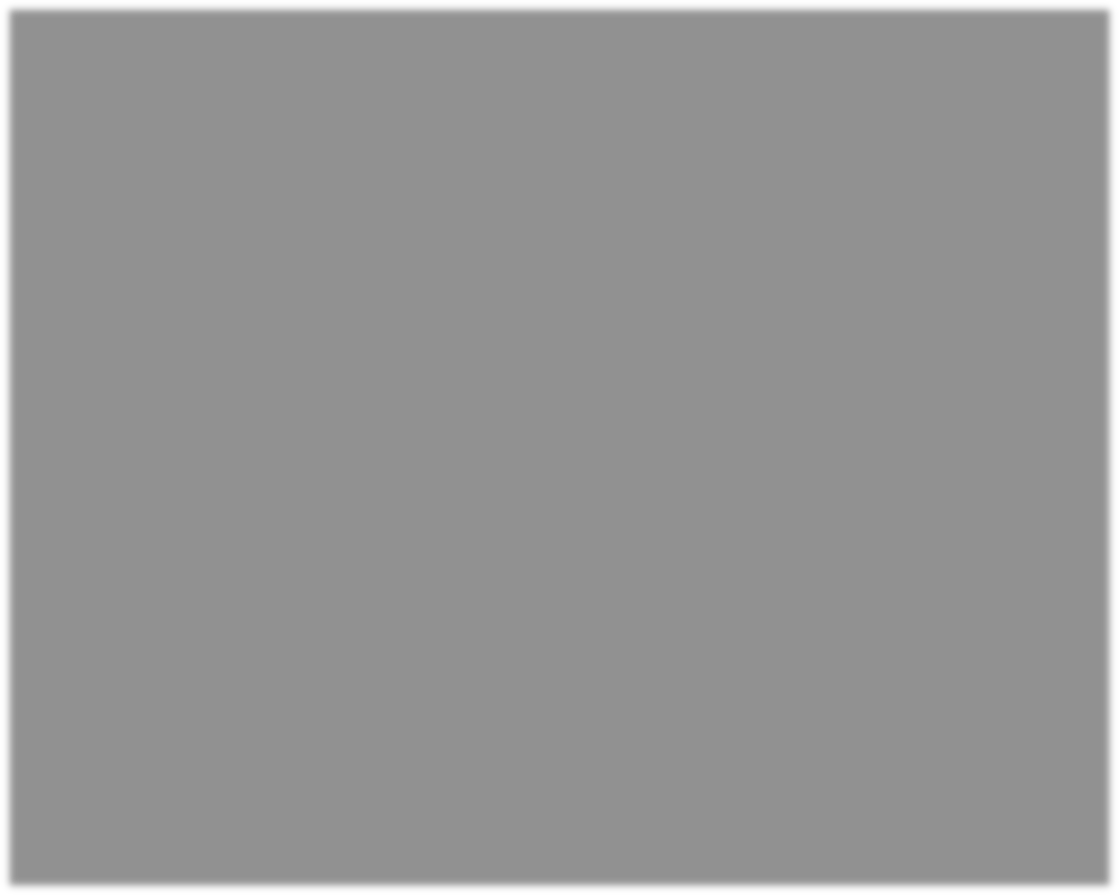
###### Step 2: Create a New ASP.NET MVC Project

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

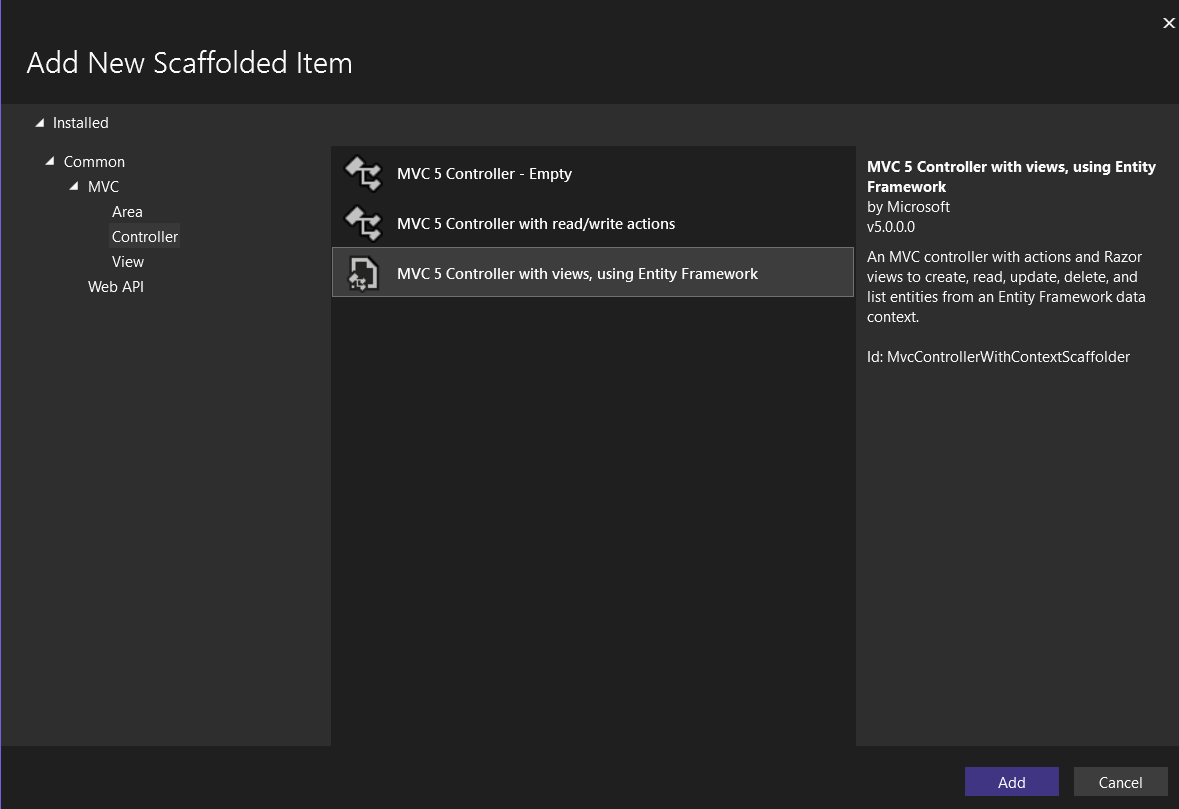
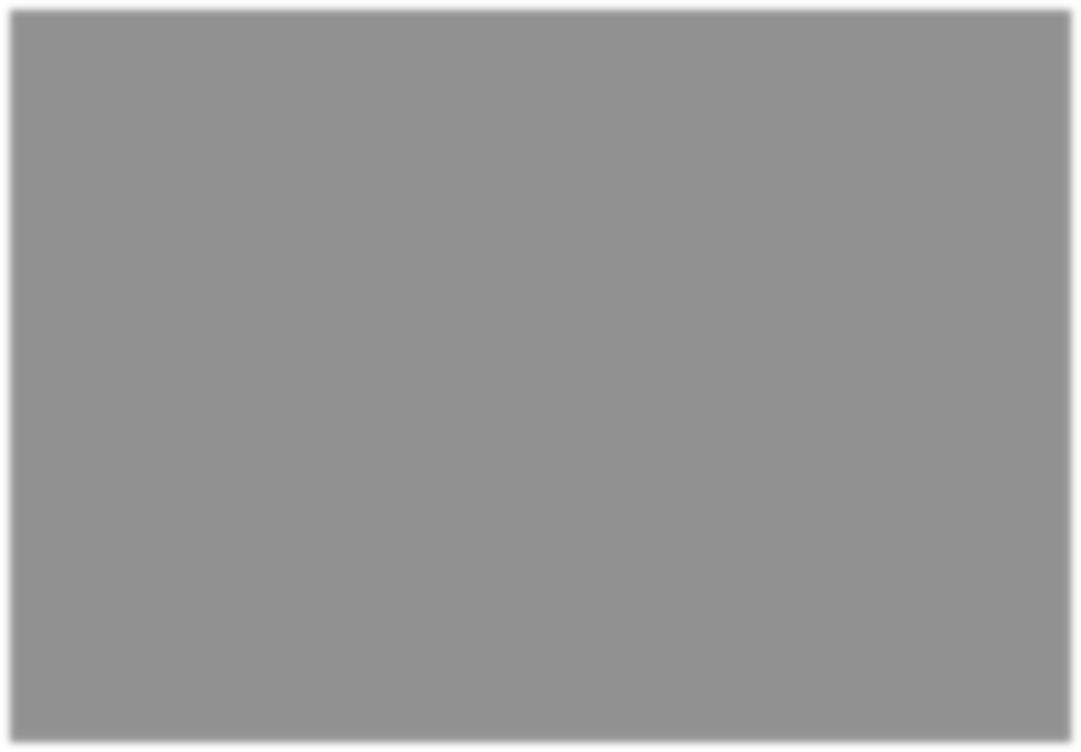
###### Step 3: Add Entity Framework Model

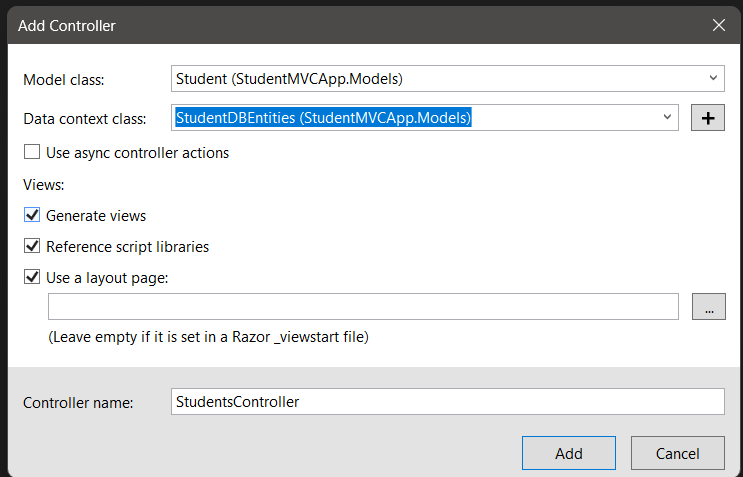
1. Right-click the **Models** folder → Add → New Item
2. Choose **ADO.NET Entity Data Model**
3. Name it: StudentModel.edmx
4. Choose: "EF Designer from database"
5. Select your SQL Server database (StudentDB)
6. Select the Students table
7. Finish to generate model classes



**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>

@tml.ActionLink("Create ew", "Create")

</p>

<table class="table">

<tr>

<th>

@tml.DisplayameFor(model => model.ame)

</th>

<th>

@tml.DisplayameFor(model => model.Email)

</th>

<th>

@tml.DisplayameFor(model => model.Age)

</th>

<th>Actions</th>

</tr>

@foreach (var item in Model) {

<tr>

<td>@item.ame</td>

<td>@item.Email</td>

<td>@item.Age</td>

<td>



@ @ @

</td>

</tr>

}

</table>

tml.ActionLink("Edit", "Edit", new { id = item.Id }) | tml.ActionLink("Details", "Details", new { id = item.Id }) | tml.ActionLink("Delete", "Delete", new { id = item.Id }

**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

