

Assignment - 1 Introduction to ASP.NET Controls & State Management

Module - 1

1. Write a program to display greeting messages on button click events.

Aim : To write a C# program that displays a greeting message when a button is clicked.

Objective : To understand how to handle button click events in a C# Windows Forms Application. To display output messages in response to user interactions.

Theory : In event-driven programming, program execution is determined by user actions such as clicks or key presses. In C#, GUI applications can be created using Windows Forms. Each control (like a button or label) can generate events, and event handlers (methods) can be written to define what should happen when the event occurs — such as when a button is clicked.

Code :

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

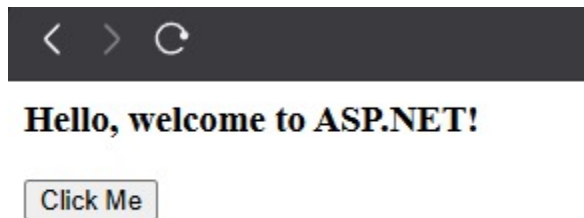
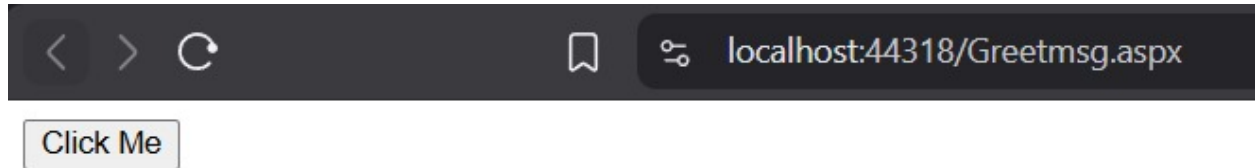
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Greeting Message Demo</title>
</head>
<body>
  <form id="form1" runat="server">
    <asp:Button ID="btnGreet" runat="server" Text="Click Me" OnClick="btnGreet_Click" />
    <br /><br />
    <asp:Label ID="lblMessage" runat="server" Text=""></asp:Label>
  </form>
</body>
</html>
```

```
using System;
```

```
public partial class _Default : System.Web.UI.Page
{
    protected void btnGreet_Click(object sender, EventArgs e)
```

```
{  
    Response.Write("<h3>Hello, welcome to ASP.NET!</h3>");  
}  
}
```

Output :



2. Write a program to print student information in table format and when you submit it displays a greeting message.

Aim: To write a C# program that displays student information in a table format and shows a greeting message when the user clicks the Submit button.

Objective: To understand how to collect and display user input in a GUI form.
To use DataGridView in C# Windows Forms for displaying tabular data.
To handle button click events and display messages interactively.

Theory: Event-driven programming is widely used in GUI applications.

In C# Windows Forms, controls such as TextBox, Button, and DataGridView are used to collect and display data.

- A TextBox allows users to input data.
- A DataGridView is used to display data in a table format.
- A Button generates an event when clicked, which can be handled using an event handler. When the Submit button is clicked, the event handler processes the data entered by the user, adds it to the table, and displays a greeting message using `MessageBox.Show()`.

Code:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm2.aspx.cs" Inherits="Project1.WebForm2" %>
```

```
<!DOCTYPE html>
```

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

```
<head runat="server">
```

```
<title></title>
```

```
<style type="text/css">
```

```
.auto-style1 {
```

```
Width: 100%;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

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

```
<div>
```

```
</div>
```

```
<table class="auto-style1">
```

```
<tr>
```

```
<td>Name</td>
```

```
<td>
```

```

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

            </td>

        </tr>

        <tr>

            <td>Roll No</td>

            <td>

                <asp:Label ID="Label2" runat="server"
Text="11"></asp:Label>

            </td>

        </tr>

        <tr>

            <td>Course</td>

            <td>

                <asp:Label ID="Label3" runat="server" Text="B.Sc.
C.S."></asp:Label>

            </td>

        </tr>

        <tr>

            <td>

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

            </td>

        </tr>

    </table>

```

</form>

</body>

</html>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.web.UI.WebControls;

namespace Project1

{

public partial class WebForm2 : System.Web.UI.Page

{

protected void Page_Load(object sender, EventArgs e)

{

}

protected void Button_click(object sender, EventArgs e)

{

**Response.Write("Thank You" + Label1.Text + "for
Submitting Information");**

}

}

}

Design:

Body	
Name	Himanshu
Roll No	11
Course	B.Sc. C.S.
<input type="button" value="Submit"/>	

Output:

Thank You Himanshu for Submitting Information	
Name	Himanshu
Roll No	11
Course	B.Sc. C.S.
<input type="button" value="Submit"/>	

3. A basic contact form where the user enters their name, email, password and mobile number on Register button click it will display all the information.

Aim :

To write a C# program that creates a basic contact form where the user enters their Name, Email, Password, and Mobile Number, and displays all the entered information upon clicking the Register button.

Objective :

- To learn how to create a basic GUI form using Windows Forms in C#.
- To understand how to collect user input using TextBox and PasswordBox controls.
- To handle button click events and display entered information.
- To enhance understanding of event-driven programming concepts in C#.

Theory :

In C# Windows Forms, applications are based on event-driven programming, where the program flow is determined by user interactions such as button clicks, key presses, etc. GUI controls like TextBox, Label, and Button are used to create interactive interfaces.

- TextBox: Used for text input (e.g., name, email, mobile).
- PasswordChar: Property used to mask the password field for security.
- Button: Generates a **Click** event when pressed.

- `MessageBox.Show()`: Used to display a message dialog with custom text.

When the Register button is clicked, the event handler retrieves values from all the input fields and displays them in a message box.

Code :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="mod1app3.aspx.cs"
Inherits="ModB100.WebForm1" %>
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Register Now</title>
    <style type="text/css">
        .auto-style1 {
            height: 262px;
        }
        .auto-style2 {
            height: 219px;
        }
        .auto-style3 {
            width: 100%;
            height: 100%;
        }
        .auto-style4 {
            text-align: center;
        }
        .auto-style5 {
            width: 115px;
        }
        .auto-style6 {
            width: 415px;
            height: 29px;
        }
        .auto-style7 {
            height: 29px;
        }
    </style>
</head>
```

```

<body class="auto-style1">
  <form id="form1" runat="server">
    <div class="auto-style2">
      <table class="auto-style3">
        <tr>
          <td class="auto-style4" colspan="2">
            <h2>Register Now</h2>
          </td>
        </tr>
        <tr>
          <td class="auto-style5">Name</td>
          <td class="auto-style6">
            <asp:TextBox ID="TextBox1" runat="server"
TextMode="SingleLine"></asp:TextBox>
          </td>
        </tr>
        <tr>
          <td class="auto-style5">E-mail</td>
          <td class="auto-style6">
            <asp:TextBox ID="TextBox2" runat="server"
TextMode="Email"></asp:TextBox>
          </td>
        </tr>
        <tr>
          <td class="auto-style5">Password</td>
          <td class="auto-style6">
            <asp:TextBox ID="TextBox3" runat="server"
TextMode="Password"></asp:TextBox>
          </td>
        </tr>
        <tr>
          <td class="auto-style5">Mobile Number</td>
          <td class="auto-style6">
            <asp:TextBox ID="TextBox4" runat="server"
TextMode="Number"></asp:TextBox>
          </td>
        </tr>
        <tr>
          <td colspan="2" class="auto-style4">

```

```

                                <asp:Button ID="Button1" runat="server" Text="Register"
OnClick="Button1_Click" />
                            </td>
                        </tr>
                        <tr>
                            <td colspan="2" class="auto-style4">
                                <asp:Label ID="Label1" runat="server" Text=""></asp:Label>
                            </td>
                        </tr>
                    </table>
                    <asp:Label ID="Label1" runat="server"></asp:Label>
                </div>
            </form>
        </body>
    </html>

```

```

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

namespace ModulB100
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            Label1.Text = "Name : " + TextBox1.Text +
                "<br>Email : " + TextBox2.Text +
                "<br>Mobile Number : " + TextBox4.Text;
        }
    }
}

```

```

protected void TextBox1_TextChanged(object sender, EventArgs e)
{
}
}
}

```

Output

Register Now

Name	<input type="text" value="Himanshu Pandey"/>
E-mail	<input type="text" value="mca-reg@gmail.com"/>
Password	<input type="text"/>
Mobile number	<input type="text" value="123456789"/>

Name : Himanshu Pandey
 Email : mca-reg@gmail.com
 Mobile Number : 123456789

4. A basic contact form where the user enters their name, and on button click, the name is displayed in a label.

Aim :

To write a C# ASP.NET program that accepts a user's name from a textbox and displays it in a label when the button is clicked.

Objective :

To understand how to handle Button Click events in ASP.NET Web Forms.

To learn how to capture user input using a TextBox control.

To display dynamic output using a Label control.

Theory :

ASP.NET Web Forms is an event-driven framework used to build dynamic web applications using the .NET platform.

A web form consists of server controls such as `TextBox`, `Button`, and `Label` which can interact with the server.

When a user enters their name in a TextBox and clicks a Button, a Click event is triggered. The event handler method defined in the code-behind (C# file) executes, retrieves the text from the TextBox, and sets it as the content of the Label.

This demonstrates the concept of event handling and server-side control manipulation in ASP.NET.

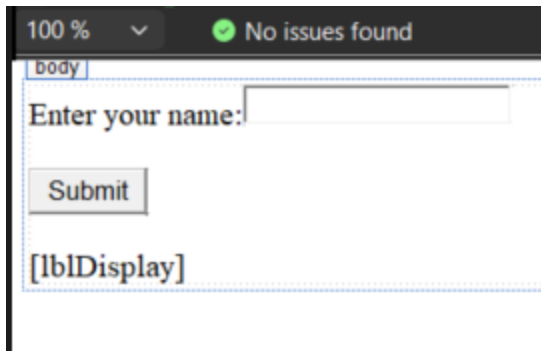
Code :

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

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Contact Form</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:Label ID="lblPrompt" runat="server" Text="Enter your name:"></asp:Label>
        <asp:TextBox ID="txtName" runat="server"></asp:TextBox>
        <br /><br />
        <asp:Button ID="btnSubmit" runat="server" Text="Submit" OnClick="btnSubmit_Click" />
        <br /><br />
        <asp:Label ID="lblDisplay" runat="server"></asp:Label>
    </form>
</body>
</html>

using System;
using System.Xml.Linq;

public partial class _Disname : System.Web.UI.Page
{
    protected void btnSubmit_Click(object sender, EventArgs e)
    {
        string name = txtName.Text;
        lblDisplay.Text = "Hello, " + name + "!";
    }
}
```



Output :

Enter your name:

Hello, Himanshu!

5. Take three Button controls, Red, Green, and Blue, and one Label control. Using CSS class, when a user presses any of the three buttons, the appearance of the label will change accordingly.

Aim :

To write a C# ASP.NET program that uses three buttons — Red, Green, and Blue — to change the appearance (color) of a label using CSS classes.

Objective :

To understand how to apply CSS styling dynamically in ASP.NET Web Forms.

To learn how to handle Button Click events in C#.

To demonstrate how CSS classes can be used to change the look of a web control programmatically.

Theory :

In ASP.NET Web Forms, web pages are built using server-side controls that can respond to user interactions.

When a user clicks a Button, an event (like `Button_Click`) is triggered, and the corresponding event handler in the C# code-behind executes.

Using CSS classes, we can define styles for elements like labels (e.g., background color, text color). By assigning different CSS classes to the label dynamically in code-behind, we can change its appearance at runtime.

Code :

```
<%@Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm2.aspx.cs"
Inherits="Mod1B100.WebForm2" %>
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
<style type="text/css">
.auto-style1 {
height: 261px;
}
.auto-style2 {
height: 162px;
}
.red_bg{
background-color:red;
color:white;
}
.green_bg{
background-color:green;
color:white;
}
.blue_bg{
background-color:blue;
color:white;
}
</style>
</head>
<body class="auto-style1">
<form id="form1" runat="server">
<div class="auto-style2">
<asp:Button ID="Button1" runat="server" Text="Red" class="red_bg"
OnClick="Button1_Click" /><br /><br />
```

```

<asp:Button ID="Button2" runat="server" Text="Green" class="green_bg"
OnClick="Button2_Click" /><br /><br />
<asp:Button ID="Button3" runat="server" Text="Blue" class="blue_bg"
OnClick="Button3_Click" /><br />
<br />
<asp:Label ID="Label1" runat="server"></asp:Label>
<br />
</div>

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

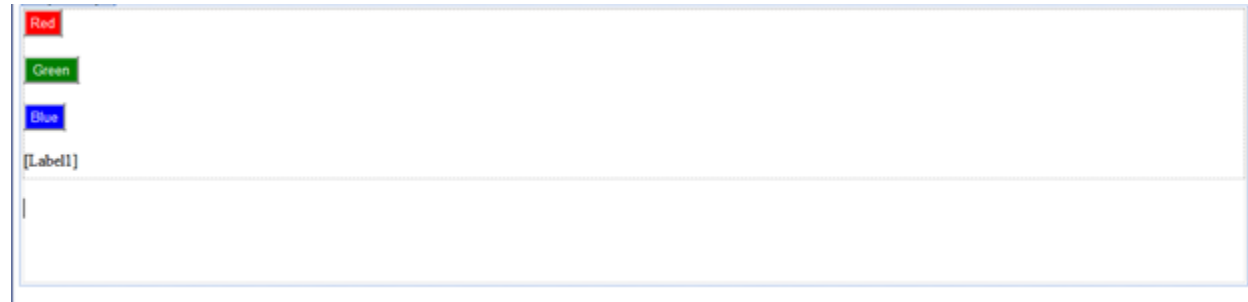
```

```

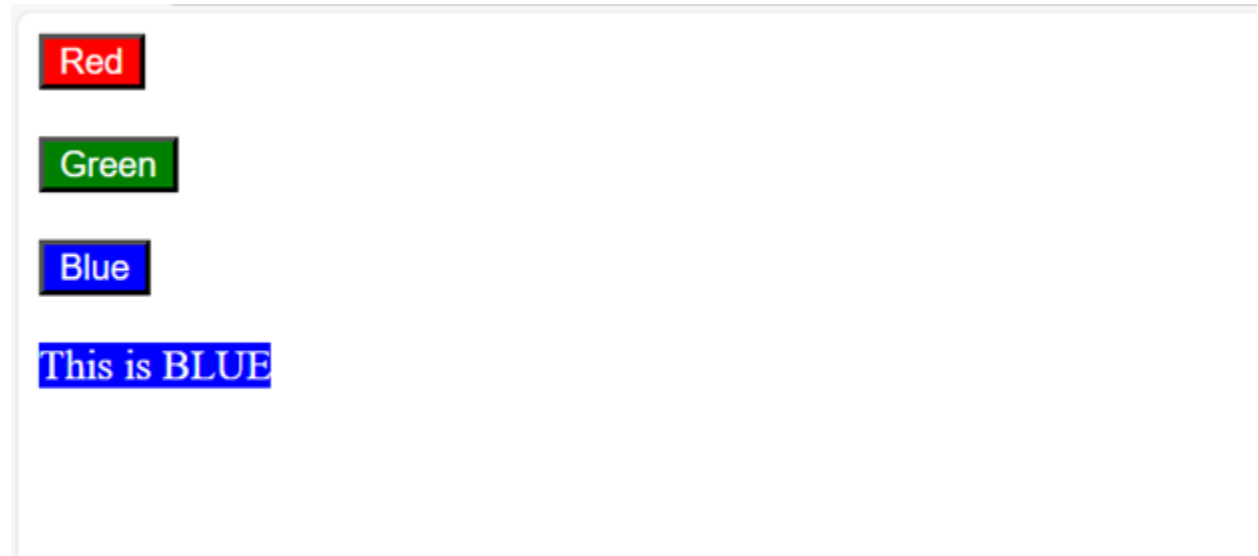
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace Mod1B100
{
    public partial class WebForm2 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void Button3_Click(object sender, EventArgs e)
        {
            Label1.Text = "This is BLUE";
            Label1.CssClass = "blue_bg";
        }
        protected void Button2_Click(object sender, EventArgs e)
        {
            Label1.Text = "This is GREEN";
            Label1.CssClass = "green_bg";
        }
        protected void Button1_Click(object sender, EventArgs e)
        {
            Label1.Text = "This is Red";
            Label1.CssClass = "red_bg";
        }
    }
}

```

```
}  
}  
}
```



Output :





6. Write a program to give font effects (name, size, effect) to the text (without using Button control).

Aim :

To write a C# ASP.NET program that changes the font name, size, and style (bold/italic/underline) of text dynamically without using a Button control.

Objective :

To understand how to apply font effects (name, size, and style) in ASP.NET Web Forms.

To learn how to use DropDownList and CheckBox controls to modify text appearance dynamically.

To demonstrate event-driven programming without requiring a Button click.

Theory :

In ASP.NET Web Forms, user interface elements such as DropDownList, CheckBox, and Label are server controls that can respond to user interactions through events.

Each control triggers specific events (like SelectedIndexChanged for DropDownList or CheckChanged for CheckBox).

By enabling AutoPostBack, these events are automatically sent to the server when the user changes the selection — allowing the page to update dynamically without a button.

We can modify the Label's properties like:

- `Font.Name` → Changes font family
- `Font.Size` → Changes text size
- `Font.Bold`, `Font.Italic`, `Font.Underline` → Toggle text effects

Code :

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication2.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
Select Font :
<asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged">
<asp:ListItem>Times New Roman</asp:ListItem>
<asp:ListItem>Bahnschrift Light</asp:ListItem>
<asp:ListItem>Broadway</asp:ListItem>
<asp:ListItem>Californian FB</asp:ListItem>
<asp:ListItem>Eras Demi ITC</asp:ListItem>
</asp:DropDownList>
<br />
<br />
Select Font Size :
<asp:DropDownList ID="DropDownList2" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="DropDownList2_SelectedIndexChanged" style="height:
22px">
<asp:ListItem>1</asp:ListItem>
<asp:ListItem>4</asp:ListItem>
<asp:ListItem>7</asp:ListItem>
<asp:ListItem>11</asp:ListItem>
<asp:ListItem>15</asp:ListItem>
<asp:ListItem>17</asp:ListItem>
<asp:ListItem>19</asp:ListItem>
<asp:ListItem>20</asp:ListItem>
```

[illegible]

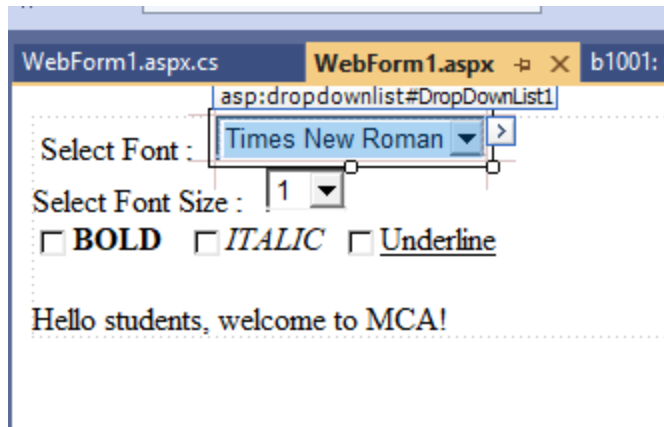
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace WebApplication2
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

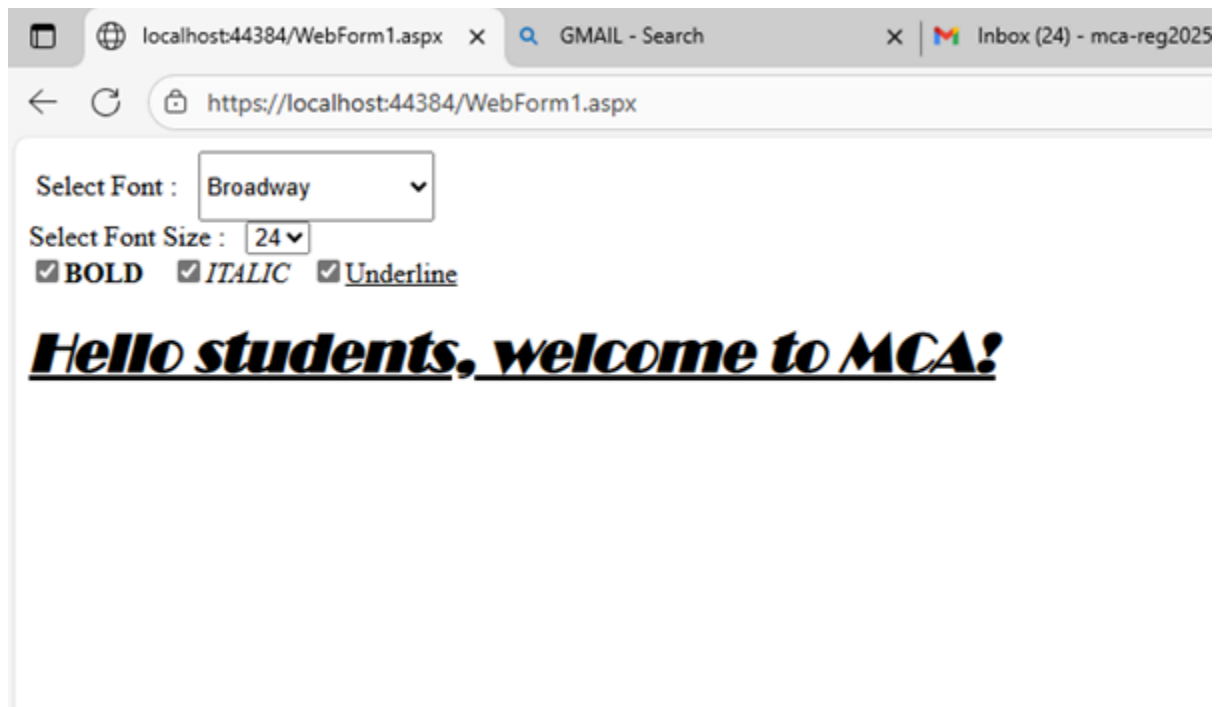
        protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
        {
            Label1.Font.Name=DropDownList1.SelectedItem.Text.ToString();
        }

        protected void DropDownList2_SelectedIndexChanged(object sender, EventArgs e)
        {
            Label2.Font.Name=DropDownList2.SelectedItem.Text.ToString();
        }
    }
}
```

```
{
Label1.Font.Size = Convert.ToInt32(DropDownList1.SelectedItem.Text.ToString());
}
protected void Bold_CheckedChanged(object sender, EventArgs e)
{
if (CheckBox1.Checked)
{
Label1.Font.Bold = true;
}
else
{
Label1.Font.Bold = false;
}
}
protected void Underline_CheckedChanged(object sender, EventArgs e)
{
if (CheckBox3.Checked)
{
Label1.Font.Underline = true;
}
else
{
Label1.Font.Underline = false;
}
}
protected void Italic_CheckedChanged(object sender, EventArgs e)
{
if (CheckBox2.Checked)
{
Label1.Font.Italic = true;
}
else
{
Label1.Font.Italic = false;
}
}
}
```



Output :



7. Design a web application as follows. On submitting the form data confirm the selection made in the following format on one LABEL Control.

Thank you very much _____.

You have chosen _____ for breakfast. I will prepare it for you _____.

Aim : To design a C# ASP.NET web application that accepts user input for name, breakfast choice, and preparation time, and displays a confirmation message on a Label control after submitting the form.

Objective :

To understand how to design and handle input forms in ASP.NET Web Forms.

To learn how to retrieve values from controls like TextBox, DropDownList, and RadioButtonList.

To display a dynamic message using a Label control on button click.

Theory :

ASP.NET Web Forms follow an event-driven programming model, where web controls (like buttons and textboxes) trigger events handled by server-side code.

In this program:

- The user enters their name in a `TextBox`.
- Selects their breakfast item using a `DropDownList`.
- Chooses a time (like “now” or “after 10 minutes”) using a `RadioButtonList`.
- When the Submit button is clicked, the `Button_Click` event is triggered in the code-behind file.
- The selected values are combined into a formatted message and displayed on a `Label`.

Code :

```
<%@PageLanguage="C#"           AutoEventWireup="true"           CodeBehind="pract7.aspx.cs"
Inherits="b1001.pract7" %>
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
  <style type="text/css">
    body {
      height: 557px;
    }
  </style>
</head>
<body>
  <form id="form1" runat="server">
    <div style="height: 425px">
```

```

<asp:Label ID="Label1" runat="server" Text="Please enter your name :"></asp:Label>
 
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<br />
<br />
<asp:Label ID="Label2" runat="server" Text="What would you like for
breakfast?"></asp:Label>
<br />
<br />
<asp:CheckBox ID="CheckBox1" runat="server" Text="Cereal"></asp:CheckBox>
<br />
<asp:CheckBox ID="CheckBox2" runat="server" Text="Fruits"></asp:CheckBox>
<br />
<asp:CheckBox ID="CheckBox4" runat="server" Text="Pancakes"></asp:CheckBox>
<br />
<br />
<asp:Label ID="Label3" runat="server" Text="Supply me:"></asp:Label>
<br />
<br />
<asp:RadioButton ID="RadioButton1" runat="server" Text="Now"
GroupName="gp1"></asp:RadioButton>
<br />
<asp:RadioButton ID="RadioButton2" runat="server" Text="Later"
GroupName="gp1"></asp:RadioButton>
<br />
<br />
<asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Thank
you!"></asp:Button>
<br />
<br />
<asp:Label ID="Label4" runat="server"></asp:Label>
<br />
</div>
</form>
</body>
</html>

```

```

using System;
using System.Collections.Generic;

```

```

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

namespace b1001
{
    public partial class pract7 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            string name = TextBox1.Text.Trim();
            string order = "";
            string time = "";

            if (RadioButton1.Checked)
            {
                time = RadioButton1.Text;
            }

            if (RadioButton2.Checked)
            {
                time = RadioButton2.Text;
            }

            if (CheckBox1.Checked)
            {
                order += CheckBox1.Text + ", ";
            }
            if (CheckBox2.Checked)
            {
                order += CheckBox2.Text + ", ";
            }
            if (CheckBox4.Checked)

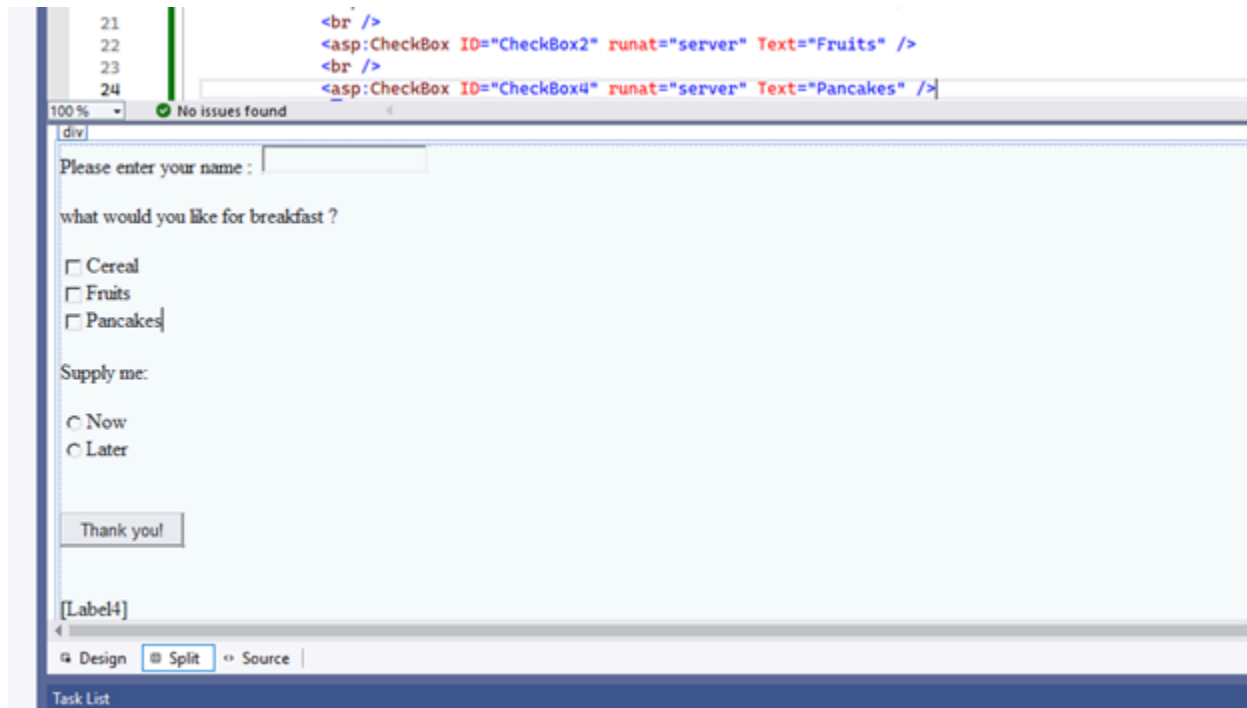
```

```

    {
        order += CheckBox4.Text + ", ";
    }

    Label4.Text = "Thank you very much " + name + "<br>" + "You have chosen " +
order.TrimEnd(',', ' ') + " for breakfast. I will supply the order " + time + ".";
    }
}
}
}

```



Output :

Please enter your name :

what would you like for breakfast ?

☐ Cereal
☒ Fruits
☐ Pancakes

Supply me:

☐ Now
☒ Later

Thank you very much himanshu
You have chosen Fruitsfor breakfast .I will prepare it for you Later

8. Write a program to demonstrate Postback.

Aim : To write a C# ASP.NET program to demonstrate the concept of Postback.

Objective :

To understand how Postback works in ASP.NET Web Forms.

To learn how to handle events triggered by server-side controls that cause a Postback to the server.

To demonstrate how the Page_Load() and IsPostBack properties work.

Theory :

In ASP.NET Web Forms, a Postback is the process of submitting a page to the server for processing.

When a user interacts with a server control (like a Button or DropDownList), the entire page can be sent to the server, processed, and then re-rendered in the browser.

The Page_Load event executes every time the page is loaded or refreshed.

To differentiate between the first-time page load and a Postback, we use the IsPostBack property:

- IsPostBack == false → The page is being loaded for the first time.
- IsPostBack == true → The page is being reloaded as a result of a Postback.

Code :

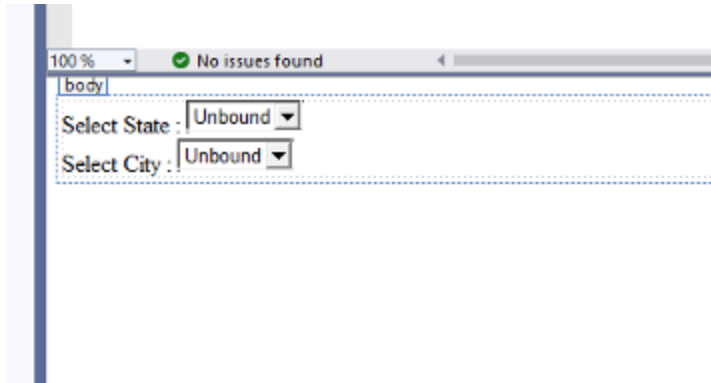
```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm3.aspx.cs" Inherits="WebApplication2.WebForm3" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
Select State :
<asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged">
</asp:DropDownList>
<br />
<br />
Select City :
<asp:DropDownList ID="DropDownList2" runat="server">
</asp:DropDownList>
</div>
</form>
</body>
</html>
```

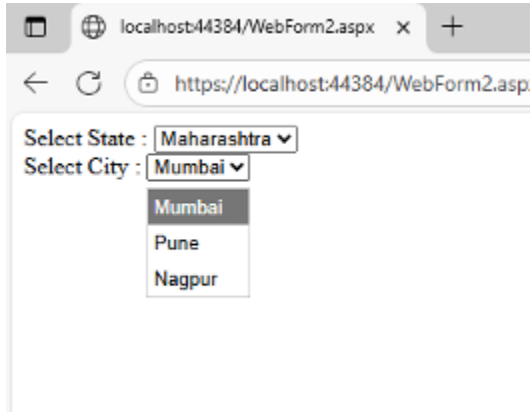
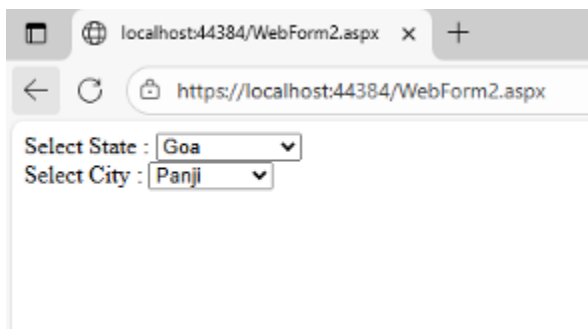
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
namespace WebApplication2
{
    public partial class WebForm3 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!Page.IsPostBack)
            {
                DropDownList1.Items.Add("Maharashtra");
                DropDownList1.Items.Add("Gujarat");

                DropDownList1.Items.Add("Goa");
            }
        }
        protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
        {
            DropDownList2.Items.Clear();
            if(DropDownList1.SelectedItem.Text == "Maharashtra")
            {
                DropDownList2.Items.Add("Mumbai");
                DropDownList2.Items.Add("Pune");
                DropDownList2.Items.Add("Nagpur");
            }
            else if (DropDownList1.SelectedItem.Text == "Gujarat")
            {
                DropDownList2.Items.Add("Surat");
                DropDownList2.Items.Add("Ahemdabad");
                DropDownList2.Items.Add("Valsad");
            }
            else if (DropDownList1.SelectedItem.Text == "Goa")
            {
                DropDownList2.Items.Add("Panaji");
                DropDownList2.Items.Add("Old Goa");
                DropDownList2.Items.Add("Panda");
            }
        }
    }
}
```



Output :



9. Write a program to upload your profile picture and display it in image control (file format should be .jpg, .jpeg or .png format)

Aim:

To write a C# ASP.NET program that allows a user to upload their profile picture and display it in an Image control.

Objective:

- To understand how to use the FileUpload control in ASP.NET.
- To validate and upload image files to the server.
- To display the uploaded image dynamically using the Image control.
- To apply file validation for allowed formats (.jpg, .jpeg, .png).

Theory:

In ASP.NET Web Forms, the FileUpload control allows users to upload files from their local system to the web server.

The uploaded file can then be stored in a specific directory (e.g., /Uploads) and displayed using an Image control.

Common properties and methods:

- `FileUpload.HasFile` → Checks if a file is selected.
- `FileUpload.FileName` → Returns the name of the selected file.
- `FileUpload.SaveAs(Server.MapPath(path))` → Saves the file to the specified server path.

Validation ensures that only image files (.jpg, .jpeg, .png) are uploaded for security and format consistency.

Code:

```
<%@Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication5.WebForm1" %>
```

```
<!DOCTYPE html>
```

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

```
<head runat="server">
```

```
<title></title>
```

```
</head>
```

```
<body>
```

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

```
<div>
```

```
<asp:FileUpload ID="FileUpload1" runat="server" />
```

```
&nbsp;<asp:Image ID="Image1" runat="server" />
```

```
<br />
```

```

        <br />
        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Upload File"
/>
        <br />
        <br />
        <asp:Label ID="Label1" runat="server"></asp:Label>
    </div>
</form>
</body>
</html>

```

```

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

```

```

namespace WebApplication5
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

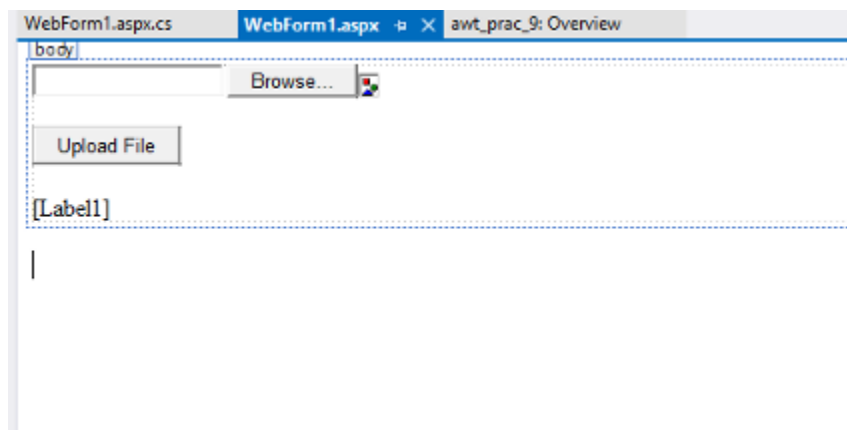
        protected void Button1_Click(object sender, EventArgs e)
        {
            if (FileUpload1.HasFile)
            {
                try
                {
                    string filename = Path.GetFileName(FileUpload1.FileName);
                    string EXT = Path.GetExtension(filename);
                    if(EXT.ToLower() == ".jpg")
                    {
                        FileUpload1.SaveAs(Server.MapPath("~/images/") + filename);
                    }
                }
                catch { }
            }
        }
    }
}

```

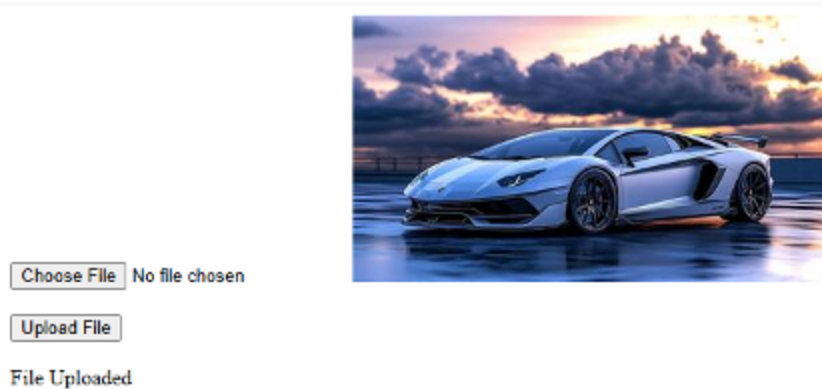
```

        Label1.Text = "File Uploaded"; ;
        Image1.ImageUrl = "~/images/" + filename;
    }
    if (EXT.ToLower() == ".png")
    {
        FileUpload1.SaveAs(Server.MapPath("~/images/") + filename);
        Label1.Text = "File Uploaded"; ;
        Image1.ImageUrl = "~/images/" + filename;
    }
    if (EXT.ToLower() == ".jpeg")
    {
        FileUpload1.SaveAs(Server.MapPath("~/images/") + filename);
        Label1.Text = "File Uploaded"; ;
        Image1.ImageUrl = "~/images/" + filename;
    }
}
catch { }
}
}
}
}

```



Output:



10. Write a program to demonstrate Navigation Controls.

Aim:

To write a C# ASP.NET program to demonstrate the use of Navigation Controls such as HyperLink, LinkButton, and Button for page navigation.

Objective:

To understand how navigation controls help users move between web pages.

To learn the use of HyperLink, LinkButton, and Button controls for page redirection.

To handle navigation events in ASP.NET using Response.Redirect() and server-side events.

Theory:

In ASP.NET Web Forms, navigation controls allow users to move from one page to another within a web application.

The most commonly used navigation controls are:

1. HyperLink Control

- Used for static navigation.
- Has a **NavigateUrl** property that specifies the destination page.
- Example

```
<asp:HyperLink ID="HyperLink1" runat="server"
NavigateUrl="~/Home.aspx">Go to Home</asp:HyperLink>
```

2. LinkButton Control

- Looks like a hyperlink but acts as a server control.
- Can execute server-side code in its `OnClick` event.

Example:

```
<asp:LinkButton ID="LinkButton1" runat="server"
OnClick="LinkButton1_Click">Go to About Page</asp:LinkButton>
```

3. Button Control

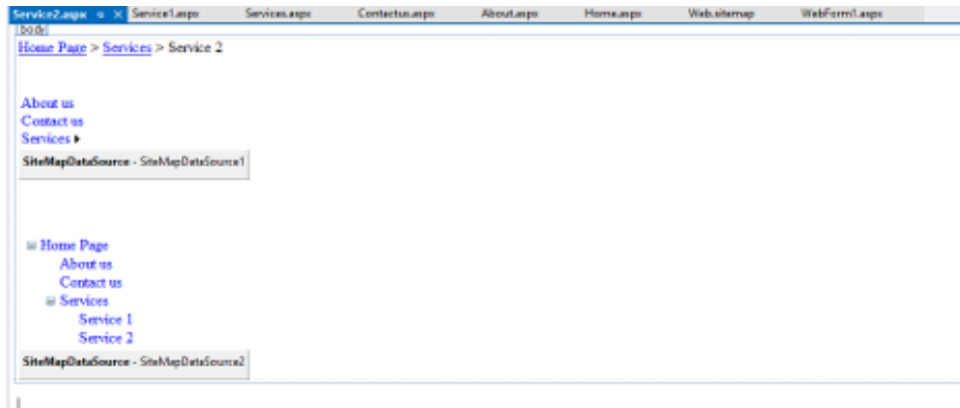
Triggers a Postback to the server and can perform navigation using `Response.Redirect()`.

Example:

```
<asp:Button ID="Button1" runat="server" Text="Go to Contact
Page" OnClick="Button1_Click" />
```

Code:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="Home.aspx" title="Home Page" description="My Home Page">
    <siteMapNode url="About.aspx" title="About us" description="About
Us"></siteMapNode>
    <siteMapNode url="Contactus.aspx" title="Contact us" description="Contact
Us"></siteMapNode>
    <siteMapNode url="Services.aspx" title="Services" description="Services">
      <siteMapNode url="Service1.aspx" title="Service 1" description="Service
1"></siteMapNode>
      <siteMapNode url="Service2.aspx" title="Service 2" description="Service
2"></siteMapNode>
    </siteMapNode>
  </siteMapNode>
</siteMap>
```



Output:

[Home Page](#) > [Services](#) > [Service 2](#)

[Home Page](#) ▶

⊞ [Home Page](#)
 [About us](#)
 [Contact us](#)
 ⊞ [Services](#)
 [Service 1](#)
 [Service 2](#)

11. Write a program to demonstrate Ad Rotator Control.

Aim:

To write a C# ASP.NET program to demonstrate the working of the AdRotator Control.

Objective:

- To understand how to use the AdRotator control in ASP.NET.
- To display rotating advertisements dynamically on a web page.
- To learn how to configure the XML Advertisement File and bind it to the AdRotator control.
- To display different ads randomly or based on specified weightage.

Theory:

The AdRotator control in ASP.NET Web Forms is used to display a series of banner advertisements randomly or according to assigned priority.

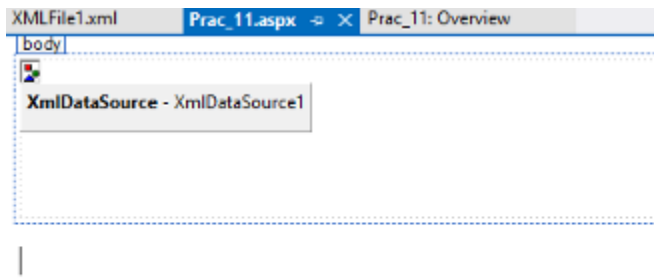
The control reads advertisement details from an XML file, called the Advertisement File, which contains:

- Image file URL
- Navigate URL (link to open when clicked)
- Alternate text
- Keyword
- Impression (weightage or probability of display)

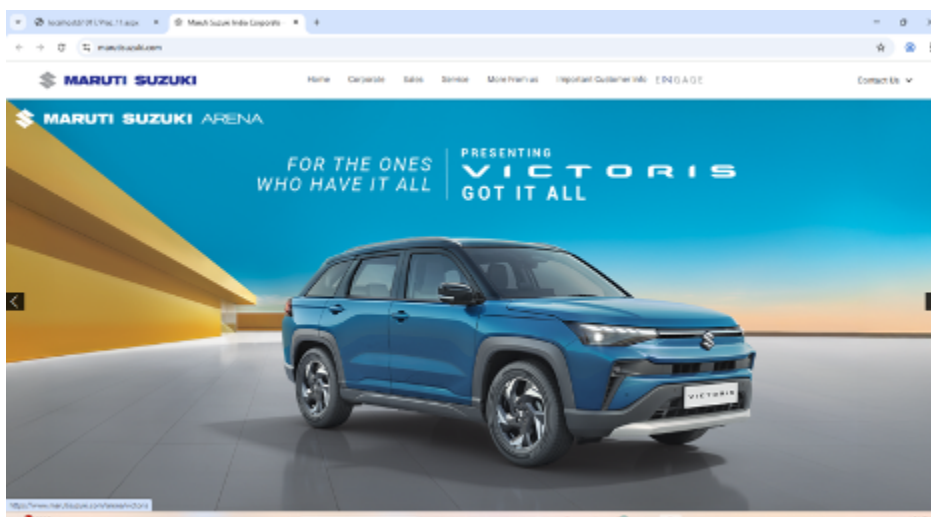
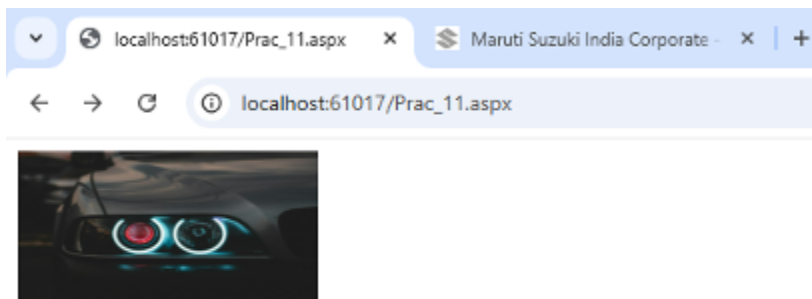
Code:

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements>
  <Ad>
    <ImageUrl>~/images/dell.png</ImageUrl>
    <Width>200</Width>
    <Height>100</Height>
    <NavigateUrl>https://www.dell.com/en-in</NavigateUrl>
    <AlternateText>AD 1</AlternateText>
    <Impression>50</Impression>
    <Keyword>Laptop</Keyword>
  </Ad>
  <Ad>
    <ImageUrl>~/images/hp.png</ImageUrl>
    <Width>200</Width>
    <Height>100</Height>
    <NavigateUrl>https://www.hp.com/in-en/home.html</NavigateUrl>
    <AlternateText>AD 2</AlternateText>
    <Impression>50</Impression>
    <Keyword>Laptop</Keyword>
  </Ad>
  <Ad>
    <ImageUrl>~/images/car.png</ImageUrl>
    <Width>200</Width>
    <Height>100</Height>
    <NavigateUrl>https://www.marutisuzuki.com/</NavigateUrl>
    <AlternateText>AD 3</AlternateText>
    <Impression>50</Impression>
    <Keyword>Cars</Keyword>
  </Ad>
  <Ad>
    <ImageUrl>~/images/iso.png</ImageUrl>
    <Width>200</Width>
    <Height>100</Height>
    <NavigateUrl>https://www.iso.org/home.html</NavigateUrl>
    <AlternateText>AD 4</AlternateText>
    <Impression>50</Impression>
    <Keyword>Cars</Keyword>
  </Ad>
</Advertisements>
```

</Advertisements>



Output:



12. Write a program to demonstrate Calendar control and highlight all holidays of October 2025.

Aim:

To develop a C# Windows Forms application that demonstrates the Calendar control and highlights all holidays in October 2025.

Objective:

- To understand and use the MonthCalendar control in C#.
- To learn how to highlight specific dates (holidays).
- To practice setting BoldedDates in a calendar.

Theory:

The MonthCalendar control in C# displays a monthly calendar that allows users to view and select dates.

You can highlight or mark specific dates using the **BoldedDates** property.

By setting **BoldedDates** with a list of **DateTime** objects, you can visually emphasize holidays or special events.

Key Properties:

- **SelectionStart** – Gets or sets the start date of the selected range.
- **SelectionEnd** – Gets or sets the end date of the selected range.
- **BoldedDates** – An array of **DateTime** objects that appear in bold on the calendar.

Code:

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

namespace b1001
{
    public partial class WebForm3 : System.Web.UI.Page
    {
```

```

protected void Page_Load(object sender, EventArgs e)
{

}

protected void Calendar1_DayRender(object sender, DayRenderEventArgs e)
{
    DateTime d1 = new DateTime(2025, 10, 2);
    if (e.Day.Date.ToString() == d1.ToString())
    {
        e.Cell.Controls.Add(new LiteralControl("\n Gandhi Jayanti"));
        e.Cell.BackColor = System.Drawing.Color.LightPink;
        e.Cell.Font.Bold = true;
        e.Cell.ForeColor = System.Drawing.Color.Beige;
    }

    DateTime d2 = new DateTime(2025, 10, 21);
    if (e.Day.Date.ToString() == d2.ToString())
    {
        e.Cell.Controls.Add(new LiteralControl("\n Diwali"));
        e.Cell.BackColor = System.Drawing.Color.Yellow;
        e.Cell.Font.Bold = true;
        e.Cell.ForeColor = System.Drawing.Color.Red;
    }

    DateTime d3 = new DateTime(2025, 10, 31);
    if (e.Day.Date.ToString() == d3.ToString())
    {
        e.Cell.Controls.Add(new LiteralControl("\n Halloween"));
        e.Cell.BackColor = System.Drawing.Color.Black;
        e.Cell.Font.Bold = true;
        e.Cell.ForeColor = System.Drawing.Color.Orange;
    }
}
}
}

```

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm3.aspx.cs"
Inherits="b1001.WebForm3" %>
```

```
<!DOCTYPE html>
```

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

```
<head runat="server">
```

```
<title></title>
```

```
</head>
```

```
<body>
```

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

```
<div style="height: 384px">
```

```
<asp:Calendar ID="Calendar1" runat="server" BackColor="White"
BorderColor="Black" BorderStyle="Solid" CellSpacing="1" Font-Names="Verdana"
Font-Size="9pt" ForeColor="Black" Height="364px" NextPrevFormat="FullMonth"
OnDayRender="Calendar1_DayRender" OnSelectionChanged="Page_Load"
SelectedDate="2025-11-03" Width="474px">
```

```
<DayHeaderStyle Font-Bold="True" Font-Size="8pt" ForeColor="#333333"
Height="8pt" />
```

```
<DayStyle BackColor="#CCCCCC" />
```

```
<NextPrevStyle Font-Bold="True" Font-Size="8pt" ForeColor="White" />
```

```
<OtherMonthDayStyle ForeColor="#999999" />
```

```
<SelectedDayStyle BackColor="#333399" ForeColor="White" />
```

```
<TitleStyle BackColor="#333399" BorderStyle="Solid" Font-Bold="True"
Font-Size="12pt" ForeColor="White" Height="12pt" />
```

```
<TodayDayStyle BackColor="#999999" ForeColor="White" />
```

```
</asp:Calendar>
```

```
</div>
```

```
</form>
```

```
</body>
```

```
</html>
```

WebForm3.aspx.cs WebForm3.aspx* validationControls.aspx.cs validationControls.aspx WebForm2.aspx.cs

```
19 <TodayDayStyle BackColor="#999999" ForeColor="white" />
20 </asp:Calendar>
21 </div>
22 </form>
23 </body>
```

100% No issues found Ln: 21 Ch: 9 SPC

div

October		November 2025				December	
Sun	Mon	Tue	Wed	Thu	Fri	Sat	
26	27	28	29	30	31	1	
2	3	4	5	6	7	8	
9	10	11	12	13	14	15	
16	17	18	19	20	21	22	
23	24	25	26	27	28	29	
30	1	2	3	4	5	6	

Output:

WebForm3.aspx.cs WebForm3.aspx* validationControls.aspx.cs validationControls.aspx WebForm2.aspx.cs

```
19 <TodayDayStyle BackColor="#999999" ForeColor="white" />
20 </asp:Calendar>
21 </div>
22 </form>
23 </body>
```

100% No issues found Ln: 21 Ch: 9 SPC

div

October		November 2025				December	
Sun	Mon	Tue	Wed	Thu	Fri	Sat	
26	27	28	29	30	31	1	
2	3	4	5	6	7	8	
9	10	11	12	13	14	15	
16	17	18	19	20	21	22	
23	24	25	26	27	28	29	
30	1	2	3	4	5	6	

13 Create a registration form having the following UI.

Registration Form

Full Name

Address

State:

Mobile No.

Email:

Password:

Confirm Password

Enter Class (6-12)

Aim:

To design and implement a Registration Form in C# Windows Forms that collects user information such as name, address, contact details, and login credentials.

Objective:

- To understand how to design forms in C#.
- To learn how to use TextBox, ComboBox, Label, and Button controls.
- To collect and validate user input in a Windows Forms application.
- To perform simple actions such as clearing input fields and validating passwords.

Theory:

Windows Forms is a graphical (GUI) class library included as part of Microsoft .NET Framework. It provides a platform to develop desktop applications with rich user interfaces.

Key controls used:

- Label – Displays static text.
- TextBox – Used for user input.
- ComboBox – Used to select items from a dropdown list.
- Button – Triggers actions such as registration or cancel.

Events like `Click` are handled to perform actions when the user interacts with buttons.

Code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="validationControls.aspx.cs"
Inherits="b1001.validationControls" %>
```

```
<!DOCTYPE html>
```

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

```
<head runat="server">
```

```
    <title>Validation Controls Example</title>
```

```
    <style type="text/css">
```

```
        #TextArea1 {
```

```
            height: 69px;
```

```
        }
```

```
    </style>
```

```
    <script type="text/javascript">
```

```
        function MyValidation(src, args) {
```

```
            var input = document.getElementById('<%= TextBox6.ClientID %>').value;
```

```
            args.IsValid = input.length >= 8; // Password must be at least 8 chars
```

```
        }
```

```
    </script>
```

```
</head>
```

```
<body style="height: 525px">
```

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

```
        <div style="height: 519px">
```

```
            <asp:Panel ID="Panel1" runat="server" GroupingText="Registration Form">
```

```
                &nbsp;Full Name&nbsp; :&nbsp;&nbsp;&nbsp;
```

```
                <asp:TextBox ID="TextBox1" runat="server" Placeholder="Enter full
name"></asp:TextBox>
```

```
                <asp:RequiredFieldValidator
```

```
                    ID="RequiredFieldValidator1"
```

```
                    runat="server"
```

```
                    ControlToValidate="TextBox1"
```

```
                    ErrorMessage="* Full name mandatory"
```

```
                    ForeColor="Red"
```

```
                    ValidationGroup="ValidationGroup">
```

```
                </asp:RequiredFieldValidator>
```


Address:

<asp:TextBox

ID="TextBox2"

```
runat="server"
```

Height="71px"

Width="165px"

```
Placeholder="Enter Address">
```

</asp:TextBox>

<asp:RequiredFieldValidator

ID="RequiredFieldValidator2"

```
runat="server"
```

```
ControlToValidate="TextBox2"
```

```
ErrorMessage="* Address cannot be empty"
```

```
ForeColor="Red"
```

```
ValidationGroup="ValidationGroup">
```

</asp:RequiredFieldValidator>

[illegible]

```
<asp:DropDownList ID="DropDownList1" runat="server">
```

<asp:ListItem>Select State</asp:ListItem>

MaharashtraGujarat`<asp:ListItem>Goa</asp:ListItem>`

</asp:DropDownList>

<asp:RequiredFieldValidator

ID="RequiredFieldValidator3"

```
runat="server"
```

```
ControlToValidate="DropDownList1"
```

```
InitialValue="Select State"
```

```
ErrorMessage="* Select a State"
```

```
ForeColor="Red"
```

```
ValidationGroup="ValidationGroup">
```

</asp:RequiredFieldValidator>

Mobile No. :

```
<asp:TextBox ID="TextBox4" runat="server" placeholder="Enter mobile number"></asp:TextBox>
```

```

<asp:RegularExpressionValidator
    ID="RegularExpressionValidator1"
    runat="server"
    ControlToValidate="TextBox4"
    ErrorMessage="* Enter valid 10-digit mobile number"
    ForeColor="Red"
    ValidationExpression="\d{10}$"
    ValidationGroup="ValidationGroup">
</asp:RegularExpressionValidator>

<br /><br />
Email :
    <asp:TextBox ID="TextBox5" runat="server" placeholder="Enter
Email"></asp:TextBox>
    <asp:RegularExpressionValidator
        ID="RegularExpressionValidator2"
        runat="server"
        ControlToValidate="TextBox5"
        ErrorMessage="* Enter a valid email address"
        ForeColor="Red"
        ValidationExpression="\w+([-+.']\w+)*@\w+([-.\w+)*\.\w+([-.\w+)*"
        ValidationGroup="ValidationGroup">
    </asp:RegularExpressionValidator>

<br /><br />
Password:
    <asp:TextBox ID="TextBox6" runat="server" TextMode="Password"
placeholder="Enter Password"></asp:TextBox>
    <asp:CustomValidator
        ID="CustomValidator1"
        runat="server"
        ClientValidationFunction="MyValidation"
        ControlToValidate="TextBox6"
        ErrorMessage="* Password must be at least 8 characters"
        ForeColor="Red"
        ValidationGroup="ValidationGroup">
    </asp:CustomValidator>

<br /><br />
Confirm Password:

```


[illegible]

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

namespace b1001
{
    public partial class validationControls : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            // Disable unobtrusive validation to avoid jQuery dependency
            ValidationSettings.UnobtrusiveValidationMode = UnobtrusiveValidationMode.None;
        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            // Check if all validations passed
        }
    }
}
```

```

        if (Page.IsValid)
        {
            // You can handle registration logic here
            // Example: Save data to a database or show a confirmation message
            Response.Write("<script>alert('Registration Successful!');</script>");
        }
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        // Clear all fields on Cancel
        TextBox1.Text = string.Empty;
        TextBox2.Text = string.Empty;
        DropDownList1.SelectedIndex = 0;
        TextBox4.Text = string.Empty;
        TextBox5.Text = string.Empty;
        TextBox6.Text = string.Empty;
        TextBox7.Text = string.Empty;
        TextBox8.Text = string.Empty;
    }
}
}

```

The screenshot shows a web browser window with several tabs. The active tab is 'validationControls.aspx', which displays a 'Registration Form'. The form contains the following fields and messages:

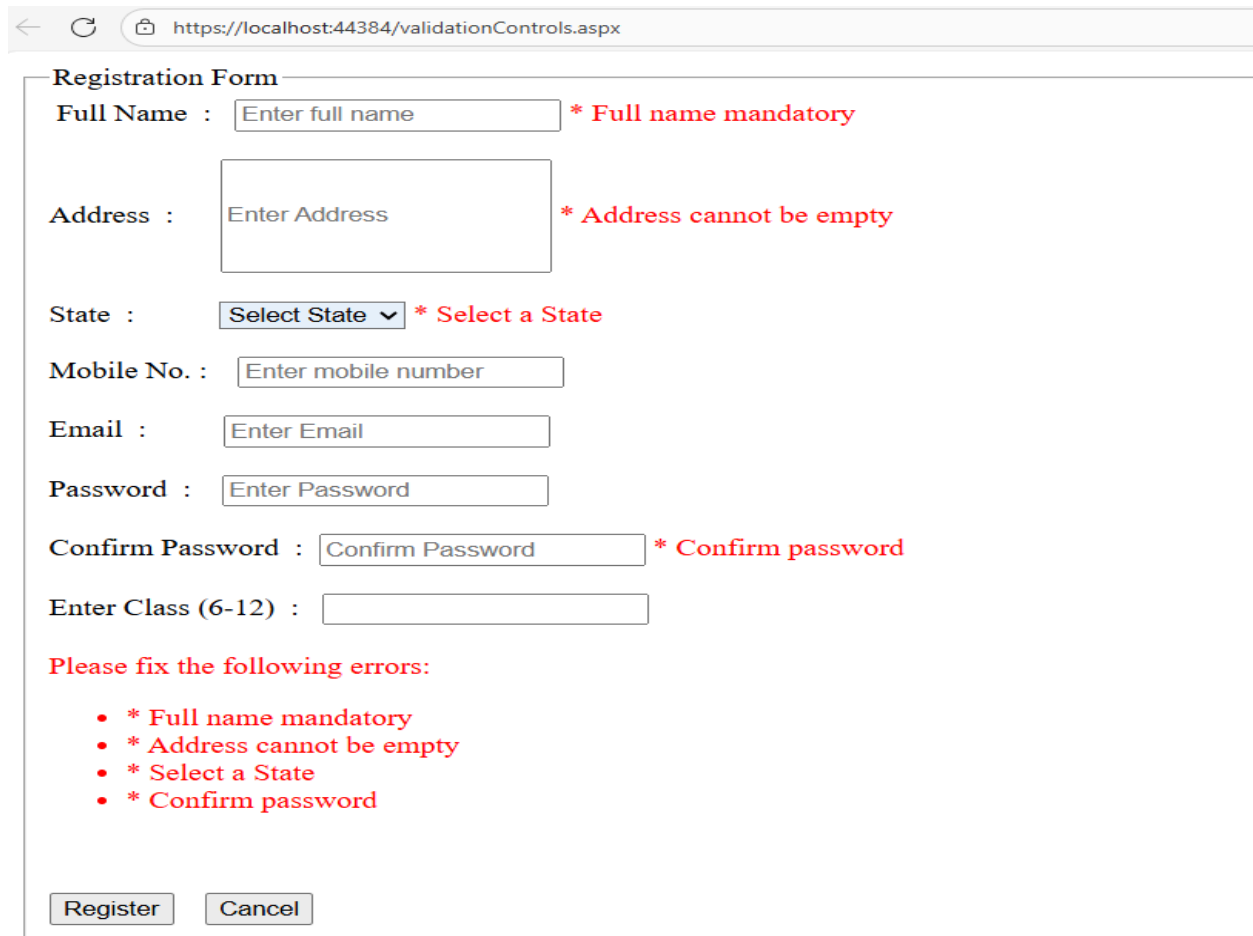
- Full Name :** * Full name mandatory
- Address :** * Address cannot be empty
- State :** * Select a State
- Mobile No. :** * Enter valid 10-digit mobile number
- Email :** * Enter a valid email address
- Password :** * Password must be at least 8 characters
- Confirm Password :** * Confirm password* Passwords do not match
- Enter Class (6-12) :** * Enter a valid class (6-12)

Below the form fields, a red message states: "Please fix the following errors:" followed by a bulleted list:

- Error message 1.
- Error message 2.

At the bottom of the form, there are two buttons: "Register" and "Cancel".

Output:



The screenshot shows a web browser window with the address bar displaying `https://localhost:44384/validationControls.aspx`. The page contains a "Registration Form" with the following fields and validation messages:

- Full Name :** * Full name mandatory
- Address :** * Address cannot be empty
- State :** * Select a State
- Mobile No. :**
- Email :**
- Password :**
- Confirm Password :** * Confirm password
- Enter Class (6-12) :**

Below the form, a red message states: "Please fix the following errors:" followed by a bulleted list of errors:

- * Full name mandatory
- * Address cannot be empty
- * Select a State
- * Confirm password

At the bottom of the form are two buttons: "Register" and "Cancel".

14. Write a program to demonstrate Master Page.

Aim:

To create a program in C# (ASP.NET) that demonstrates the use of a Master Page for consistent layout and design across multiple web pages.

Objective:

To understand how to create and use Master Pages in ASP.NET.

To maintain a uniform look and feel across all pages of a website. To simplify web design by separating layout (Master Page) from content (Content Pages).

Theory:

A Master Page in ASP.NET provides a template for one or more web pages, defining common elements such as the header, footer, and navigation bar.

Each Content Page that uses the Master Page fills in specific sections of the page (called ContentPlaceHolders) with its own unique content.

Advantages:

Reusability of layout.

Easier maintenance — changes in the master page reflect on all linked pages.

Better consistency in website design.

Key Components:

1. Master Page (.master file) – Contains layout, shared design, and one or more `<asp:ContentPlaceholder>` controls.
2. Content Page (.aspx file) – Contains content inside `<asp:Content>` controls that correspond to placeholders in the master page.

Code:

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="Site.master.cs"
Inherits="SiteMaster" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Simple Master Page Example</title>
</head>
<body>
  <form id="form1" runat="server">
    <div style="background-color:lightgray;padding:10px;">
      <h2>Site Header - Appears On All Pages</h2>
      <asp:ContentPlaceholder ID="HeadContent" runat="server" />
    </div>
    <hr />
    <asp:ContentPlaceholder ID="MainContent" runat="server" />
    <hr />
    <div style="background-color:lightgray;padding:10px;">
      Site Footer - Appears On All Pages
    </div>
```

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

```
using System;
```

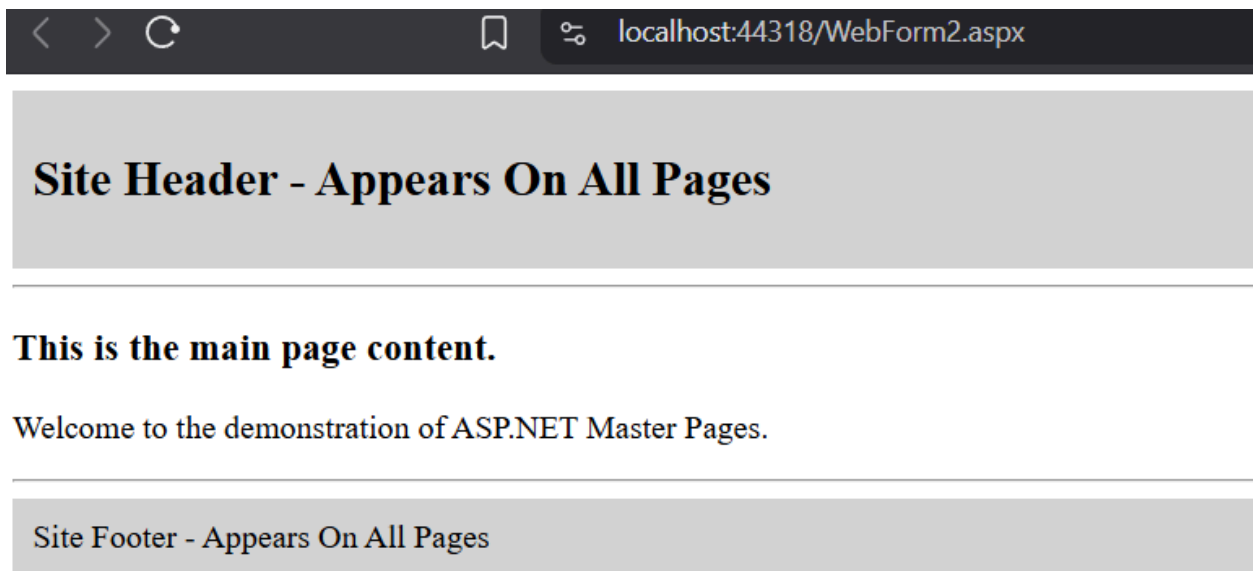
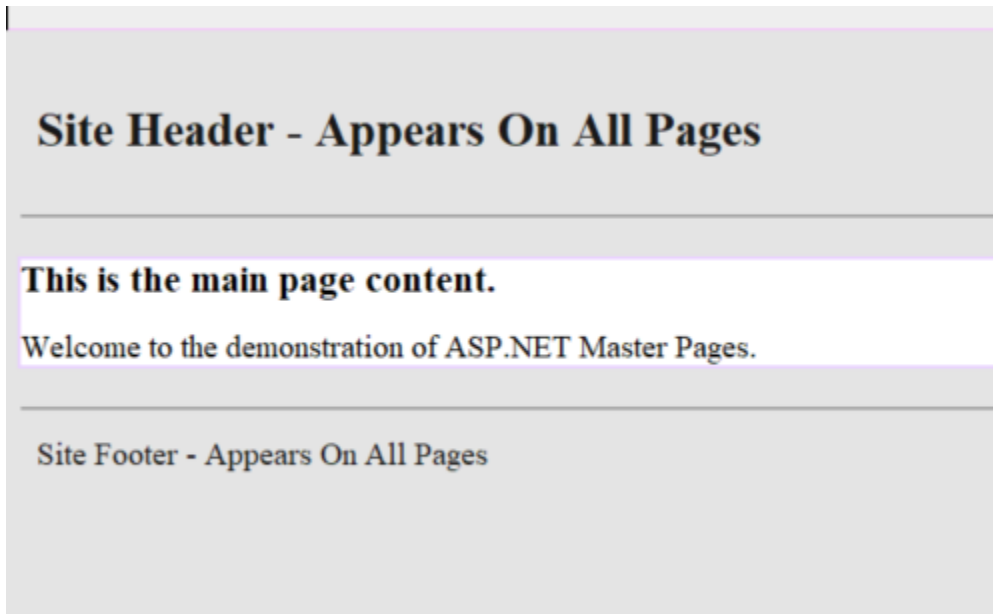
```
public partial class SiteMaster : System.Web.UI.MasterPage
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Optional master page logic
    }
}
<%@ Page Title="" Language="C#" MasterPageFile="~/Site.master" AutoEventWireup="true"
CodeBehind="WebForm2.aspx.cs" Inherits="WebApplication1.WebForm2" %>
<asp:Content ID="Content1" ContentPlaceHolderID="HeadContent" runat="server">
    <!-- Head section custom content (optional) -->
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
    <h3>This is the main page content.</h3>
    <p>Welcome to the demonstration of ASP.NET Master Pages.</p>
</asp:Content>
```

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

```
namespace WebApplication1
{
    public partial class WebForm2 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }
    }
}
```

Output:



Module - 2

1. Write a program to demonstrate the ViewState.

Aim :

To create an ASP.NET Web Form that demonstrates how ViewState maintains the values of controls across postbacks.

Objective :

To understand the concept of state management in ASP.NET.

To learn how ViewState helps retain control values between postbacks.

To implement a simple example using ViewState in C#.

Theory :

In ASP.NET, web pages are stateless — meaning data is not automatically preserved between page requests (postbacks).

To maintain data between postbacks, ASP.NET provides state management techniques, such as:

- ViewState
- Session State
- Cookies
- Query Strings

What is ViewState?

ViewState is a built-in mechanism in ASP.NET that allows the page to retain control values across postbacks.

It stores data in a hidden field (`__VIEWSTATE`) on the page in Base64 encoded format.

Key Features:

- Maintains state at the page level.
- Data is stored in the client-side (browser) as hidden data.
- Automatically managed by ASP.NET (you don't need to explicitly code for most controls).
- Can be manually used to store custom data between postbacks.

Code :

```
using System;
```

```

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

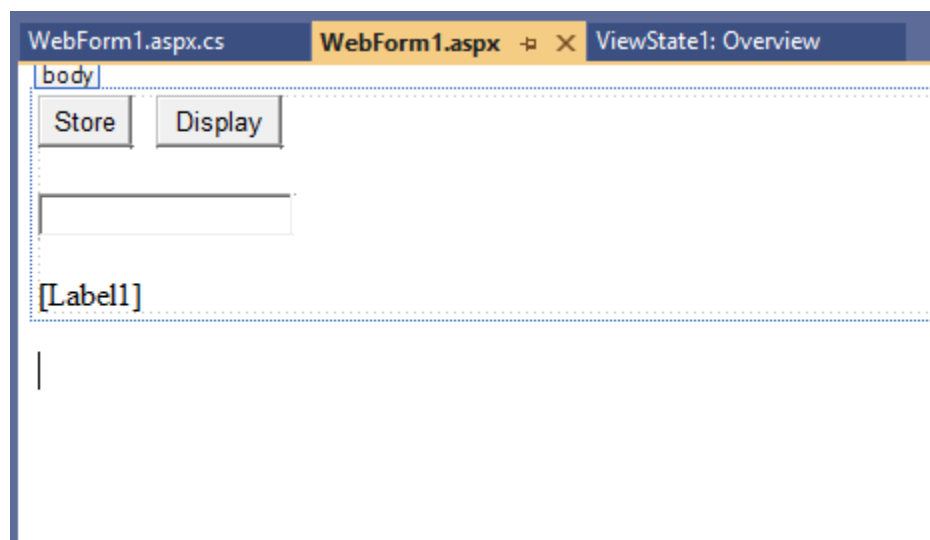
namespace ViewState1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

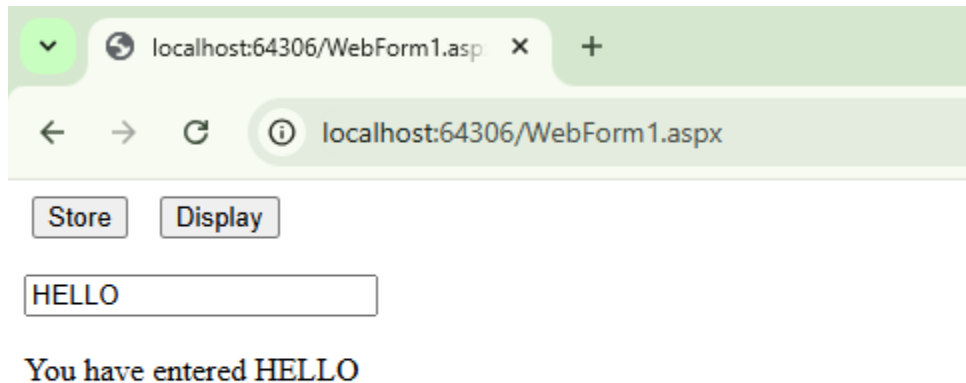
        protected void Button1_Click(object sender, EventArgs e)
        {
            ViewState["msg"] = TextBox1.Text;
        }

        protected void Button2_Click(object sender, EventArgs e)
        {
            Label1.Text += "You have entered " + ViewState["msg"].ToString();
        }
    }
}

```



Output :



2. Write a program to demonstrate the Hidden Object.

Aim :

To develop an ASP.NET program that demonstrates the use of a HiddenField (Hidden Object) to store and retrieve data between postbacks without displaying it to the user.

Objective :

To understand state management using hidden fields in ASP.NET.

To learn how to store temporary data on the client side that is not visible to the user.

To demonstrate how data can persist between postbacks using a HiddenField control.

Theory :

What is a Hidden Field?

A HiddenField (also called Hidden Object) is an ASP.NET server control used to store data that needs to persist between client requests (postbacks) but should not be visible to the user.

It is rendered as an HTML `<input type="hidden">` element, so it's not displayed on the page, but its value is sent back to the server when a postback occurs.

Key Features:

- Stores small pieces of data on the client side.

- Value persists between postbacks.
- Not visible to the user but accessible via code-behind.
- Commonly used for tracking user IDs, session data, or page state information.

Code :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Hidden.aspx.cs"
Inherits="b1001.Hidden" %>
```

```
<!DOCTYPE html>
```

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

```
<head runat="server">
```

```
<title></title>
```

```
</head>
```

```
<body>
```

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

```
<div>
```

```
<asp:HiddenField ID="HiddenField1" runat="server" Value="Helllooo" />
```

```
<br />
```

```
<asp:Button ID="Button1" runat="server"PostBackUrl="~/redirect.aspx" Text="Next"
```

```
/>
```

```
<br />
```

```
</div>
```

```
</form>
```

```
</body>
```

```
</html>
```

```
using System;
```

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

```
using System.Linq;
```

```
using System.Web;
```

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

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

```
namespace b1001
```

```
{
```

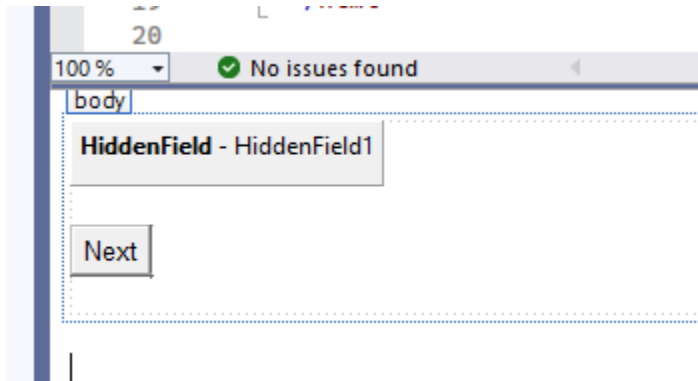
```
public partial class redirect : System.Web.UI.Page
```

```
{
```

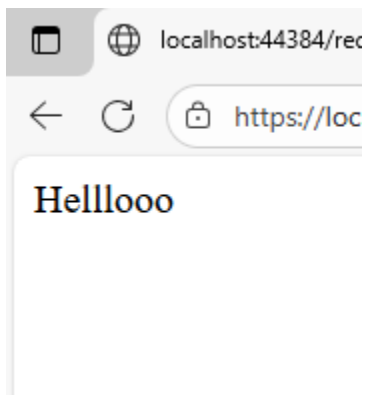
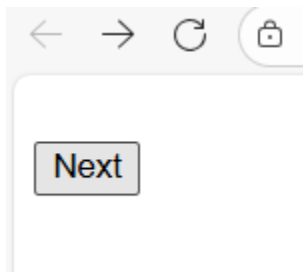
```

protected void Page_Load(object sender, EventArgs e)
{
    Response.Write(Request.Form["HiddenField1"].ToString());
}
}
}

```



Output :



3. Write a program to demonstrate the Query String

Aim:

To develop an ASP.NET program that demonstrates the use of a Query String to pass information between web pages.

Objective:

To understand how to pass data between web pages using a Query String.

To learn how to retrieve Query String values from the URL in C#.

To demonstrate the concept of client-side state management.

Theory :**What is a Query String?**

A Query String is a part of a URL that carries data between web pages in the form of key-value pairs.

It appears after the ? symbol in the URL.

Example:

`https://example.com/Page2.aspx?name=John&age=25`

Here,

- name and age are keys.
- John and 25 are their values.

In ASP.NET, we can use the `Request.QueryString` collection to retrieve these values.

Code:

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

```
<!DOCTYPE html>
```

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

```
<head runat="server">
```

```
<title></title>
```

```
</head>
```

```

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

      Enter Value:
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
      <br />
      <br />
      <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Button" />

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

```

```

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

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

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            Response.Redirect("QueryString2.aspx?msg=" + TextBox1.Text);
        }
    }
}

```

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="QueryString2.aspx.cs"
Inherits="StateManagement.QueryString2" %>
```

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>

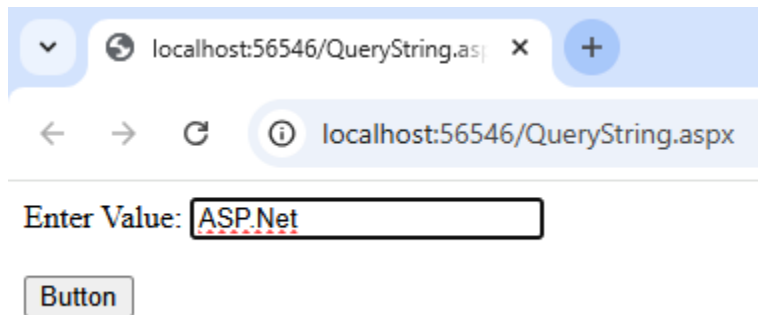
            </div>
        </form>
    </body>
</html>
```

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

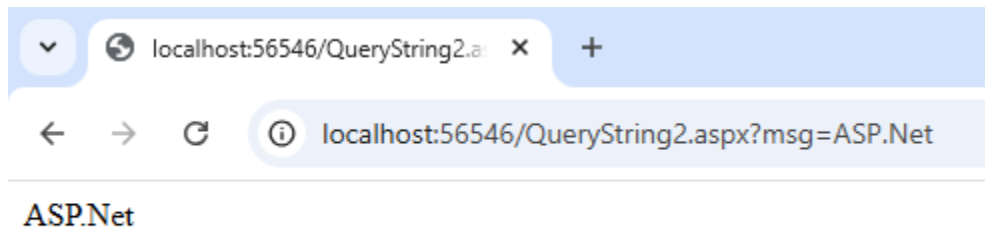
```
namespace StateManagement
{
    public partial class QueryString2 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (Request.QueryString["msg"] != null)
            {
                Response.Write(Request.QueryString["msg"].ToString());
            }
            else {
                Response.Write("3_QueryString.aspx");
            }
        }
    }
}
```

```
}  
}
```

Output:



A screenshot of a web browser window. The address bar shows 'localhost:56546/QueryString.aspx'. Below the address bar, there is a label 'Enter Value:' followed by a text input field containing the text 'ASP.Net'. Below the input field is a button labeled 'Button'.



A screenshot of a web browser window. The address bar shows 'localhost:56546/QueryString2.aspx?msg=ASP.Net'. Below the address bar, the text 'ASP.Net' is displayed.

4. Create a web application to get following information from a user. Name, Age, height, Email, Gender. Validate the input by using proper validation control. If the gender is male then navigate to Male.aspx web page, and if the gender is female then navigate to Female.aspx. Then depending upon the gender and height of the individual suggest the ideal weight.

[Note : use cookies to pass information between pages]

The height to weight ratio is as follows

Height (CM)	150	160	170	180	190
Ideal weight for Male	60	65	70	75	80
Ideal weight for Female	55	60	65	70	75

Aim:

To create an ASP.NET web application that takes user input — Name, Age, Height, Email, and Gender — validates it using Validation Controls, and redirects to separate pages (`Male.aspx` or `Female.aspx`) to display ideal weight using Cookies for data transfer.

Objective:

To learn how to collect user information using Web Forms.

To validate user input using ASP.NET validation controls.

To implement conditional navigation based on gender.

To use Cookies for passing information between web pages.

To calculate and display ideal weight based on gender and height.

Theory:

In ASP.NET, Validation Controls, Cookies, and Page Navigation are commonly used to ensure input accuracy and maintain state between pages.

1. Validation Controls:

ASP.NET provides built-in controls to validate user input before processing:

- `RequiredFieldValidator` → Ensures input is not empty.
- `RangeValidator` → Checks if a value falls within a specified range.
- `RegularExpressionValidator` → Validates format (e.g., email).
- `CompareValidator` → Compares input with another value or data type.
- `ValidationSummary` → Displays all validation errors in one place.

2. Cookies:

A Cookie is a small piece of data stored on the client browser.

In this program, cookies are used to store user details (Name, Age, Height, Gender, etc.) and access them on another page.

</html>

.aspx.cs:

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

namespace StateManagement

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

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            HttpCookie cookie = new HttpCookie("data");
            cookie.Values["name"] = TextBox1.Text;
            cookie.Values["age"] = TextBox2.Text;
            cookie.Values["height"] = TextBox3.Text;
            cookie.Values["email"] = TextBox4.Text;
            string gender = "";
            if (RadioButton1.Checked) { gender = "Male"; }
            if (RadioButton2.Checked) { gender = "Female"; }
            cookie.Values["gender"] = gender;

            Response.Cookies.Add(cookie);

            if (gender == "Male")
                Response.Redirect("~/Male.aspx");
            else
                Response.Redirect("~/Female.aspx");
        }
    }
}
```

male.aspx.cs:

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

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

```
namespace StateManagement
```

```
{
    public partial class Male : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            int weight = 0;
            string name = Request.Cookies["data"].Values["name"];
            string age = Request.Cookies["data"].Values["age"];
            string email = Request.Cookies["data"].Values["email"];
            string height = Request.Cookies["data"].Values["height"];
            if (int.Parse(height) == 150) { weight = 60; }
            if (int.Parse(height) == 160) { weight = 65; }
            if (int.Parse(height) == 170) { weight = 70; }
            if (int.Parse(height) == 180) { weight = 75; }
            if (int.Parse(height) == 190) { weight = 80; }

            Label1.Text = "Name is: " + name + "<br/>Age is: " + age + "<br/>Height is: " + height +
                "<br/>Email is: " + email + "<br/>Weight is: " + weight;
        }
    }
}
```

```
female.aspx.cs:
```

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

```
namespace StateManagement
```

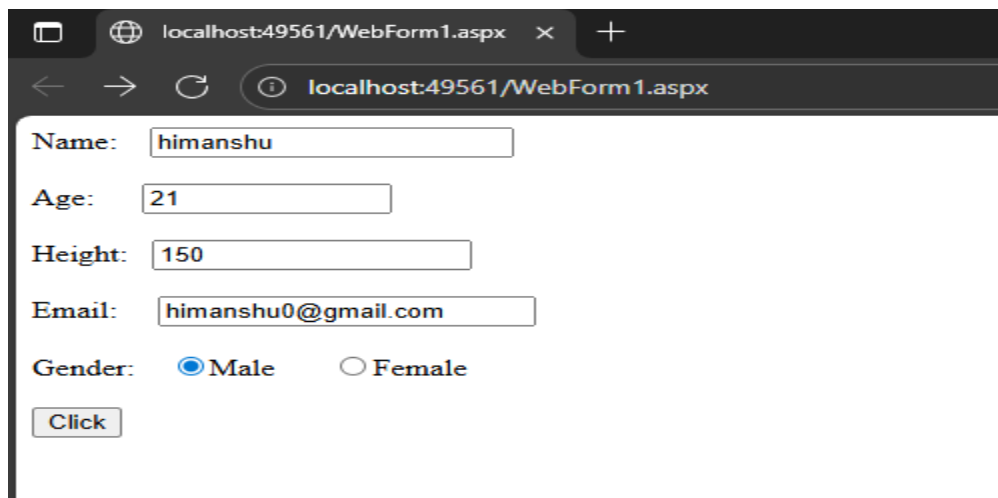
```
{
    public partial class Female : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            int weight = 0;
            string name = Request.Cookies["data"].Values["name"];
```

```

string age = Request.Cookies["data"].Values["age"];
string email = Request.Cookies["data"].Values["email"];
string height = Request.Cookies["data"].Values["height"];
if (int.Parse(height) == 150) { weight = 55; }
if (int.Parse(height) == 160) { weight = 60; }
if (int.Parse(height) == 170) { weight = 65; }
if (int.Parse(height) == 180) { weight = 70; }
if (int.Parse(height) == 190) { weight = 75; }

Label1.Text = "Name is: " + name + "<br/>Age is: " + age + "<br/>Height is: " + height +
"<br/>Email is: " + email + "<br/>Weight is: " + weight;
    }
}
}

```



localhost:49561/WebForm1.aspx

Name:

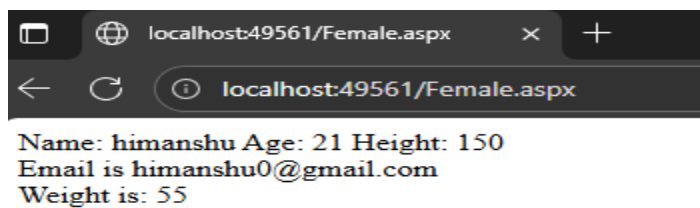
Age:

Height:

Email:

Gender: ☒ Male ☐ Female

Output:



localhost:49561/Female.aspx

Name: himanshu Age: 21 Height: 150
Email is himanshu0@gmail.com
Weight is: 55

