Advanced Web Technologies
Module 4: Session Management and AJAX

Assignment 4

1. Write a program to demonstrate the ViewState.

Aim: To demonstrate the use of ViewState in ASP.NET for maintaining the state of web controls across postbacks

Objectives:

1. Understand the concept of ViewState in web development.
2. Demonstrate how ViewState helps in maintaining the state of controls during postbacks.

Theory: ViewState

ViewState is a mechanism used in ASP.NET web applications to persist the state of server-side objects between postbacks. It works by automatically serializing the values of certain controls into hidden fields on the page, which are then sent back and forth between the client and the server with each request/response cycle.

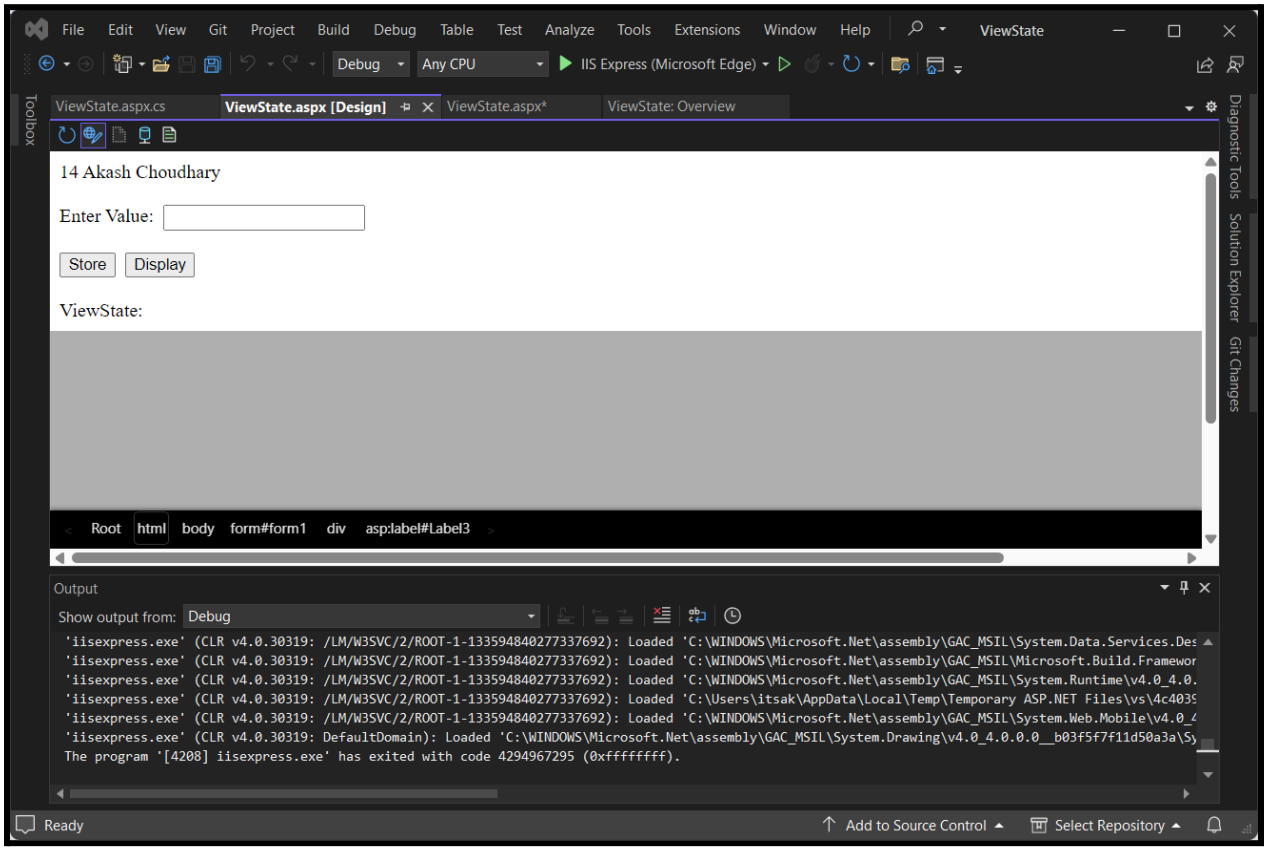
Code:

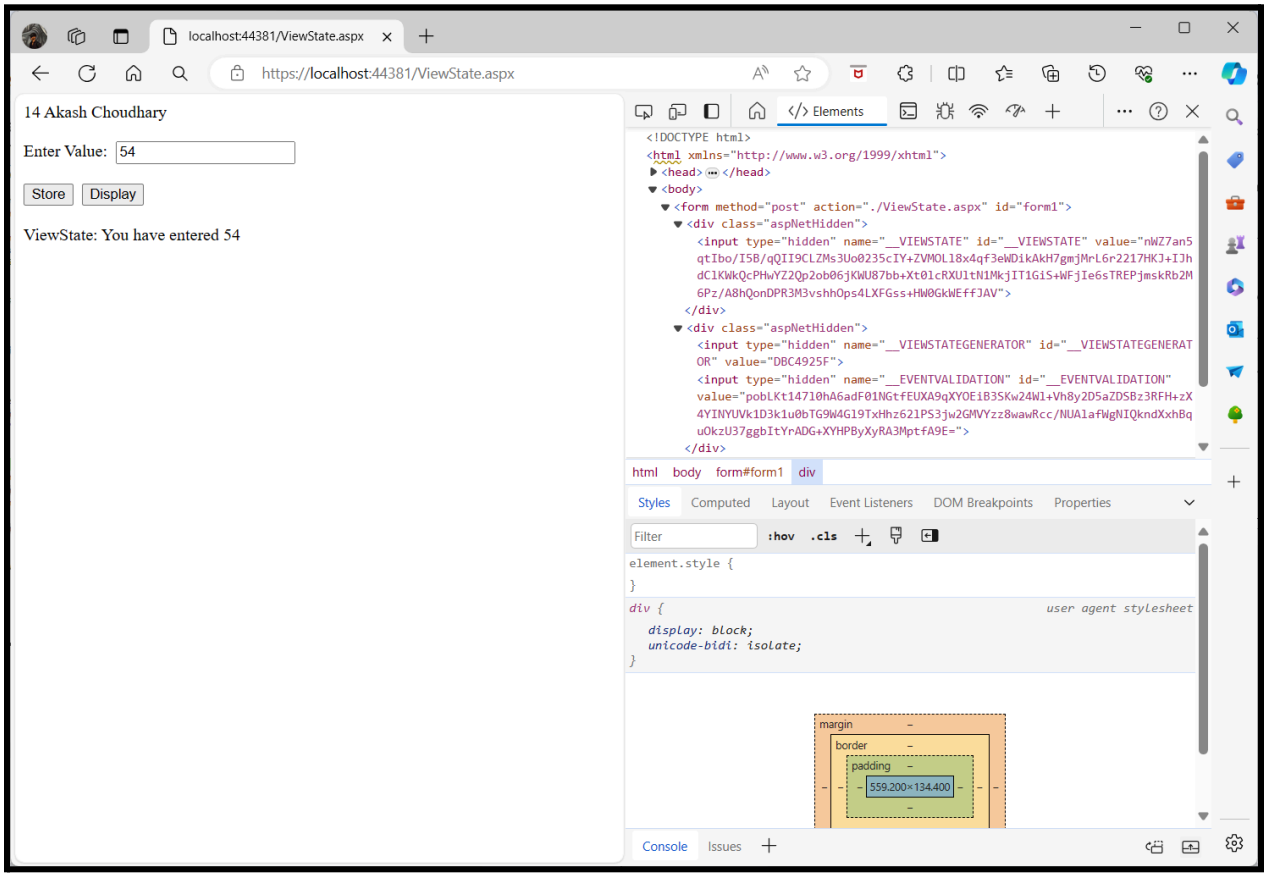
> ViewState.aspx.cs

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

namespace ViewState
{
   public partial class ViewState : 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)
        {
            Label3.Text += "You have entered " + ViewState["msg"].ToString();
        }
    }
}
```

Output:





Conclusion: Understanding ViewState is crucial for managing stateful data in ASP.NET applications, facilitating preservation of data across postbacks.

2. Write a program to demonstrate the Hidden Object.

Aim: To create a program that demonstrates the concept of a Hidden Object

Objectives:

1. To illustrate the concept of a hidden object within a graphical user interface.
2. To demonstrate how objects can be dynamically hidden or revealed based on certain conditions or user interactions.
3. To showcase the importance of user experience design in presenting information effectively while managing visibility of objects.

Theory: Hidden Object

In ASP.NET, a "hidden object" typically refers to a control or element on a web page that is not visible to the user but can hold data or perform some functionality behind the scenes. This is commonly used for passing data between pages or storing temporary information.

This hidden field won't be visible to the user, but you can use it to store data that you need to access or manipulate on the server side. This can be helpful for passing information between pages, storing state information, or performing other tasks without exposing the data to the user directly.

Code:

> HiddenObject.aspx

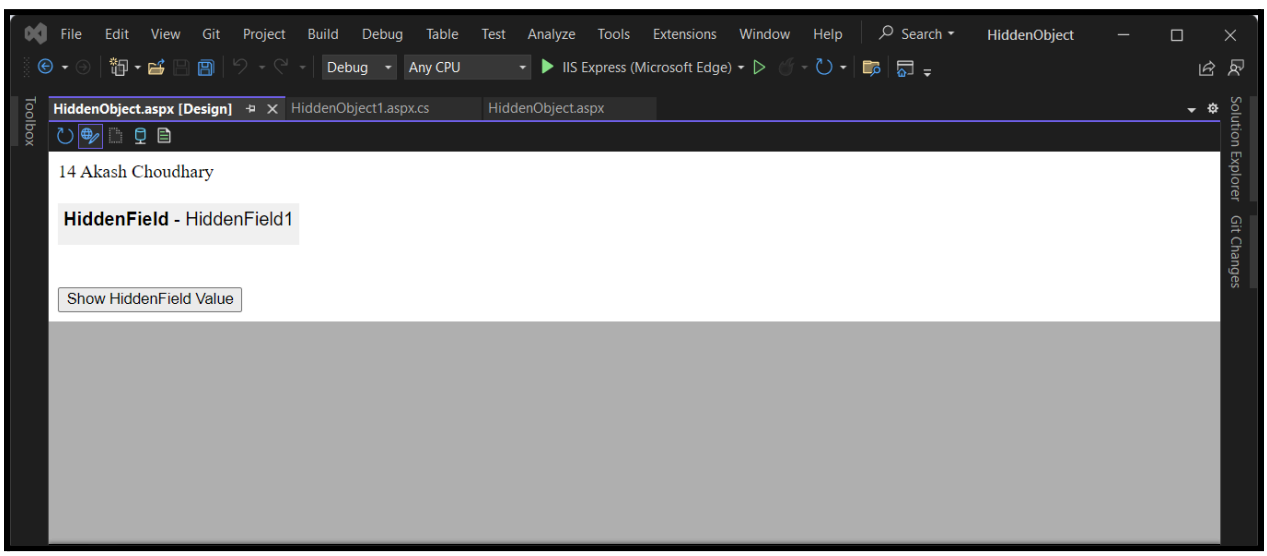
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="HiddenObject.aspx.cs" Inherits="HiddenObject.HiddenObject" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
      <div>
         <asp:Label ID="Label1" runat="server" Text="14 Akash Choudhary"></asp:Label>
         <br /><br />
         <asp:HiddenField ID="HiddenField1" runat="server" Value="14 Akash Choudhary" />
         <br /><br />
                  <asp:Button ID="Button1" runat="server" Text="Show HiddenField Value" OnClick="Button1_Click" PostBackUrl="~/HiddenObject1.aspx" />
```

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

> HiddenObject1.aspx.cs

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

namespace HiddenObject
{
    public partial class HiddenObject1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            Response.Write(Request.Form["HiddenField1"].ToString());
        }
    }
}
```

Output:



Conclusion: Hidden Object programming allows encapsulation of data and functionality, enhancing modularity and reducing complexity in software development.

3. Write a program to demonstrate the Query String.

Aim: To demonstrate passing data between web pages using the query string in ASP.NET

Objectives:

1. Understand the concept of Query String in ASP.NET.
2. Learn how to pass data between pages using Query String parameters.
3. Demonstrate retrieving Query String parameters in the code-behind file.
4. Illustrate how Query String parameters can be used to customize page behavior based on user input or context.

Theory: Query String

A query string is a part of a URL that contains data to be passed to the web server. It's composed of a question mark ? followed by key-value pairs separated by ampersands &. In ASP.NET, you can access these parameters using the Request.QueryString collection. Query strings are commonly used for passing data between pages in a web application, such as passing search parameters, user preferences, or identifying information. However, it's important to note that query strings are visible to users and can potentially be manipulated, so they should not be used to transmit sensitive information or state.

Code:

> QueryString.aspx.cs

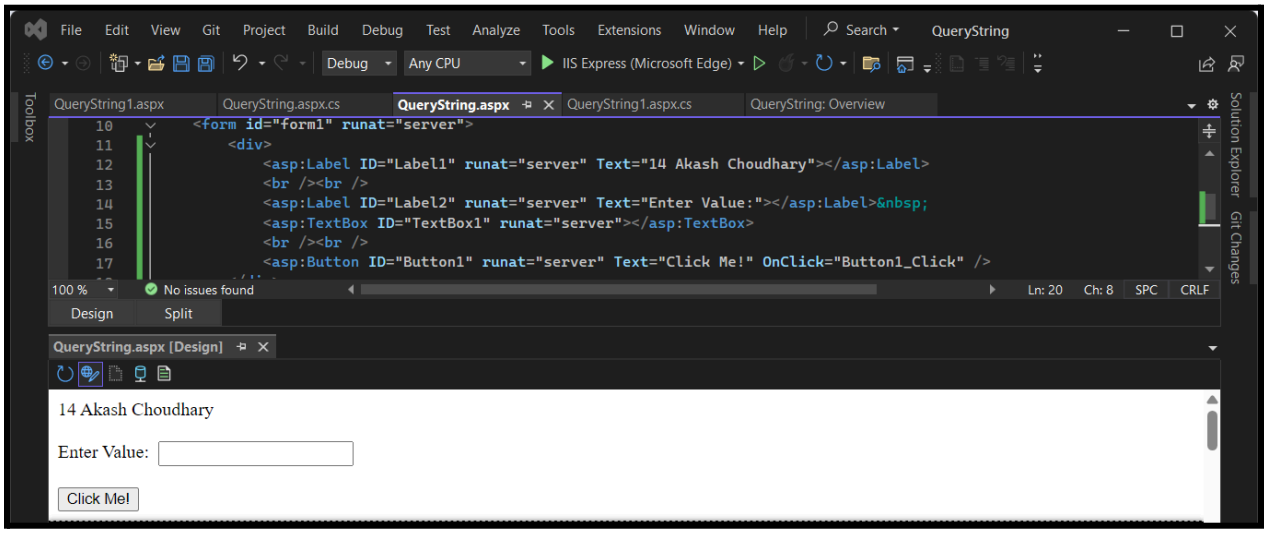
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

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

      }

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

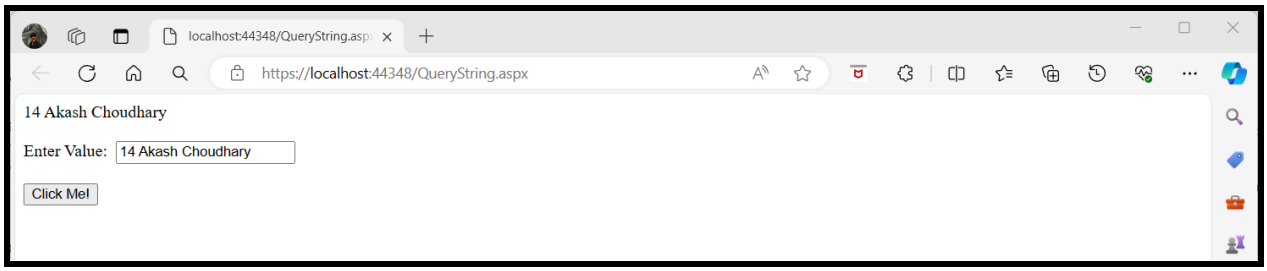
```
            Response.Redirect("QueryString1.aspx?msg=" + TextBox1.Text);
        }
    }
}
```

> QueryString1.aspx.cs

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

namespace QueryString
{
    public partial class QueryString1 : 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.Redirect("QueryString.aspx");
            }
        }
    }
}
```

Output:





Conclusion: Query strings in ASP.NET provide a simple and efficient way to pass data between different web pages or to the same page with varying parameters.

4. Create a web application to obtain the following information from a user: Name, Age, Height, Email, Gender. Validate the input using proper validation controls. If the gender is male, navigate to the Male.aspx web page; if the gender is female, navigate to Female.aspx. Then, depending on 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 (in 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 a web application that collects user information, validates input, navigates to specific pages based on gender, and suggests ideal weight based on gender and height, utilizing cookies for data transfer between pages

Objectives:

1. Collect user information: Name, Age, Height, Email, and Gender.
2. Implement proper validation controls to ensure data integrity and accuracy.
3. Based on the gender input, navigate to the appropriate page (Male.aspx or Female.aspx).
4. Calculate the ideal weight based on the gender and height of the individual.
5. Use cookies to pass the collected information (gender, height) between pages.
6. Display the suggested ideal weight based on the gender and height on the respective pages.

Theory: Cookies

In ASP.NET, cookies are small pieces of data sent from a website and stored on the user's computer by the web browser while the user is browsing. They can be used to store information such as user preferences, shopping cart items, or session identifiers.

Code:

> Cookies.aspx.cs

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

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

namespace ValidationControl
{
   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 = "Female";
         }
         if (RadioButton2.Checked)
         {
            gender = "Male";
         }
         cookie.Values["gender"] = gender;

         Response.Cookies.Add(cookie);

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

> Cookie_Male.aspx.cs

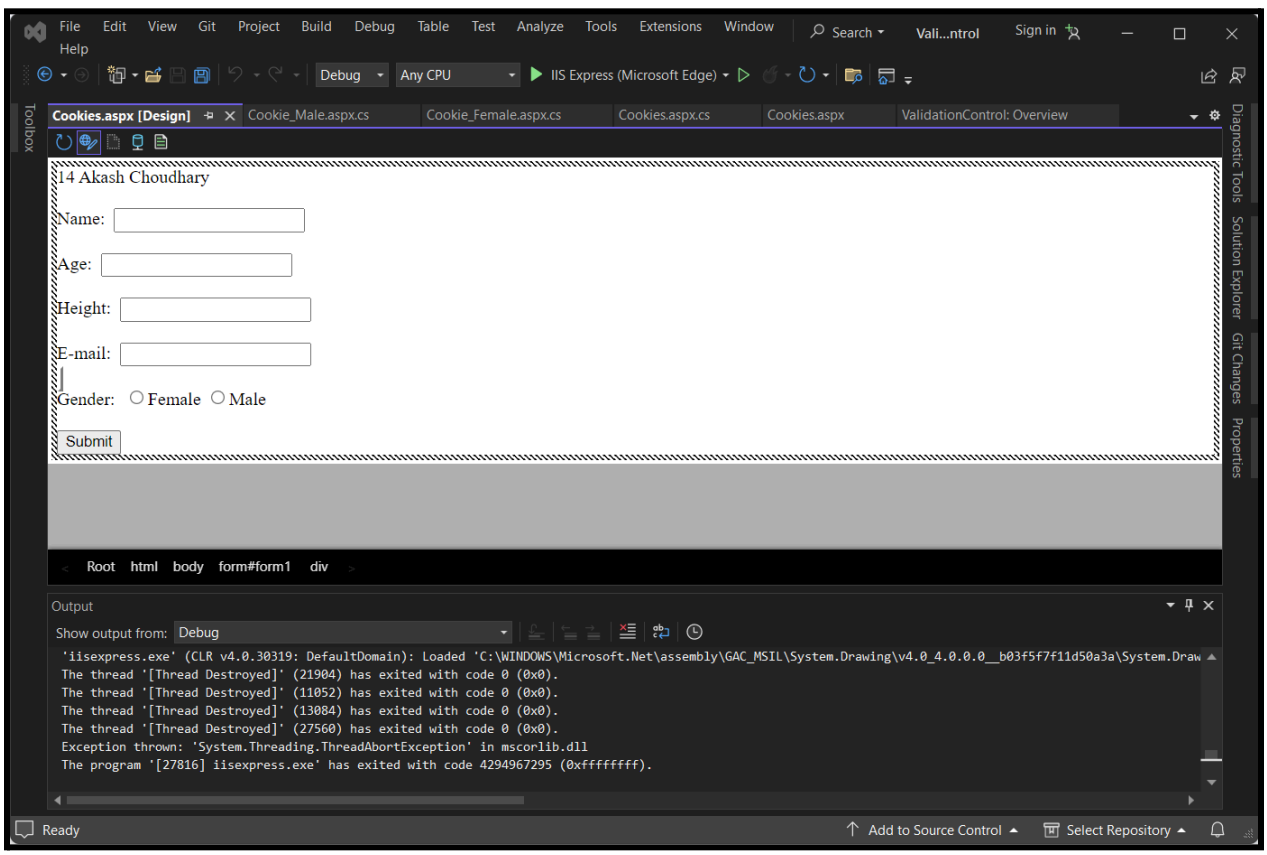
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace ValidationControl
{
   public partial class Cookie_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 height = Request.Cookies["data"].Values["height"];
         string email = Request.Cookies["data"].Values["email"];

         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; }

          Response.Write("Name : " + name + " \n\n\n Age : " + age + " \n\n\n Email : " + email +
" \n\n\n Height : " + height + " \n\n\n Ideal Weight : " + weight);
      }
   }
}
```

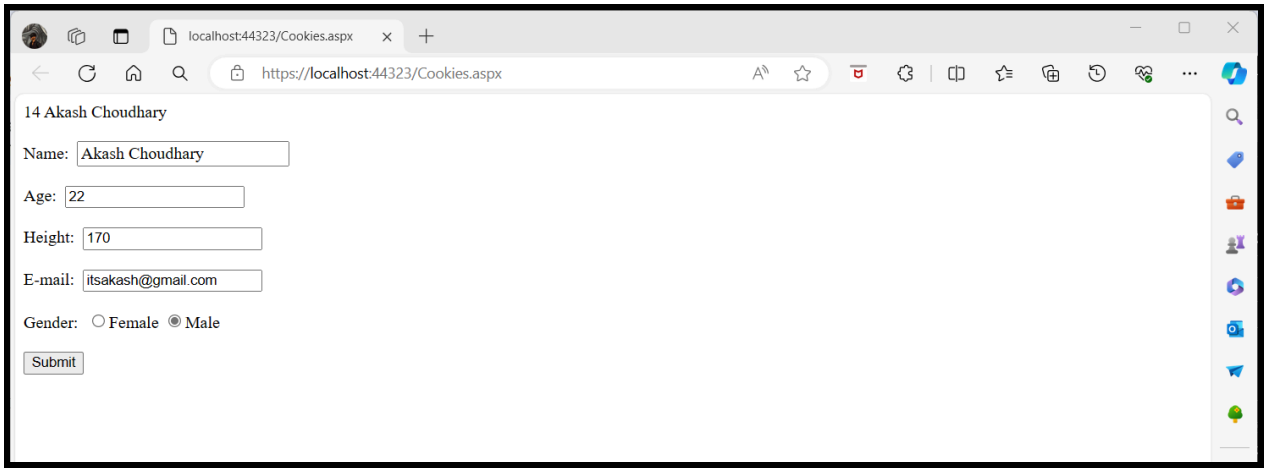
> Cookie_Female.aspx.cs

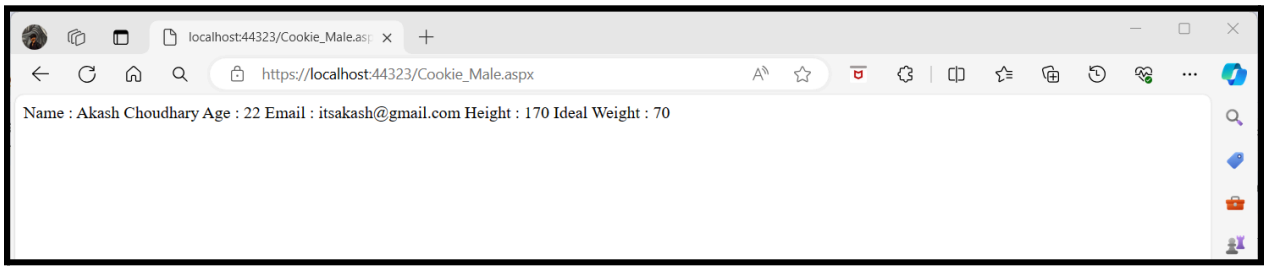
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace ValidationControl
{
   public partial class Cookie_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 height = Request.Cookies["data"].Values["height"];
        string email = Request.Cookies["data"].Values["email"];

        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; }

        Response.Write("Name: " + name + "\nAge: " + age + "\nEmail: " + email + "\nHeight: "
+ height + "\nIdeal Weight: " + weight);
        }
    }
}
```

Output:







Conclusion: Using cookies to pass information between pages in an ASP.NET web application enables personalized experiences and dynamic content presentation based on user input

5. Create an ASP.NET web application for your college fest. Provide login function and registration for a few events. Use cookies to maintain the user's session.

Aim: To create an ASP.NET web application for managing a college fest, featuring user authentication with login functionality and event registration, utilizing cookies to maintain user sessions

Objectives:

1. Implement user authentication: Enable users to log in securely to access fest-related features and content.
2. Offer event registration: Allow users to register for various fest events, providing a seamless registration process.
3. Utilize cookies for session management: Employ cookies to maintain user sessions across different pages, ensuring a consistent and personalized browsing experience.
4. Enhance user engagement: Provide interactive features and content related to the college fest to enhance user engagement and participation.
5. Ensure data security: Implement appropriate security measures to protect user data and maintain privacy throughout the fest application.

Theory: Session management

In ASP.NET, session management refers to the process of maintaining state information for each user across multiple requests to a web application. This ensures that user-specific data is preserved throughout the user's interaction with the application.

Code:

> SQL Query

```
CREATE TABLE [dbo].[RegisterData]
(
        [Id] INT NOT NULL PRIMARY KEY IDENTITY,
   [Name] VARCHAR(50) NOT NULL,
   [Password] VARCHAR(50) NOT NULL,
   [Email] VARCHAR(50) NOT NULL,
   [Mobile] BIGINT NOT NULL,
   [Event] VARCHAR(50) NOT NULL
)
```

> Register.aspx.cs

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

namespace UserCookies
{
   public partial class Register : System.Web.UI.Page
   {
                                    SqlConnection conn = new SqlConnection(@"Data
Source=(localdb)\MSSQLLocalDB;Initial Catalog=EventDB;Integrated Security=True;");

      protected void Page_Load(object sender, EventArgs e)
      {

      }

      protected void Button1_Click(object sender, EventArgs e)
      {
         conn.Open();
         string Name = TextBox1.Text;
         string Password = TextBox2.Text;
         string Email = TextBox3.Text;
         string Mobile = TextBox4.Text;
         string EventName = DropDownList1.SelectedItem.ToString();
             SqlCommand cmd = new SqlCommand("INSERT INTO RegisterData VALUES('" +
Name + "', '" + Password + "', '" + Email + "', '" + Mobile + "', '" + EventName + "')", conn);
         int result = cmd.ExecuteNonQuery();
         if (result > 0)
         {
            Response.Redirect("~/Login.aspx");
         }
         else
         {
            Response.Write("Something went wrong!");
         }
         conn.Close();
         TextBox1.Text = string.Empty;
         TextBox2.Text = string.Empty;
         TextBox3.Text = string.Empty;
         TextBox4.Text = string.Empty;
```

```
        }

        protected void Button2_Click(object sender, EventArgs e)
        {
            Response.Redirect("~/Login.aspx");
        }
    }
}
```

> Login.aspx.cs

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

namespace UserCookies
{
    public partial class Login : System.Web.UI.Page
    {
                                    SqlConnection conn = new SqlConnection(@"Data
Source=(localdb)\MSSQLLocalDB;Initial Catalog=EventDB;Integrated Security=True;");

        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            conn.Open();
                SqlCommand cmd = new SqlCommand("SELECT * FROM RegisterData WHERE
Name = '" + TextBox1.Text + "' AND Password = '" + TextBox2.Text + "'", conn);
            SqlDataAdapter sda = new SqlDataAdapter(cmd);
            DataTable dt = new DataTable();
            sda.Fill(dt);
            conn.Close();
            if (dt.Rows.Count > 0)
            {
                Session["user"] = TextBox1.Text;
                Response.Redirect("~/Homepage.aspx");
            }
```

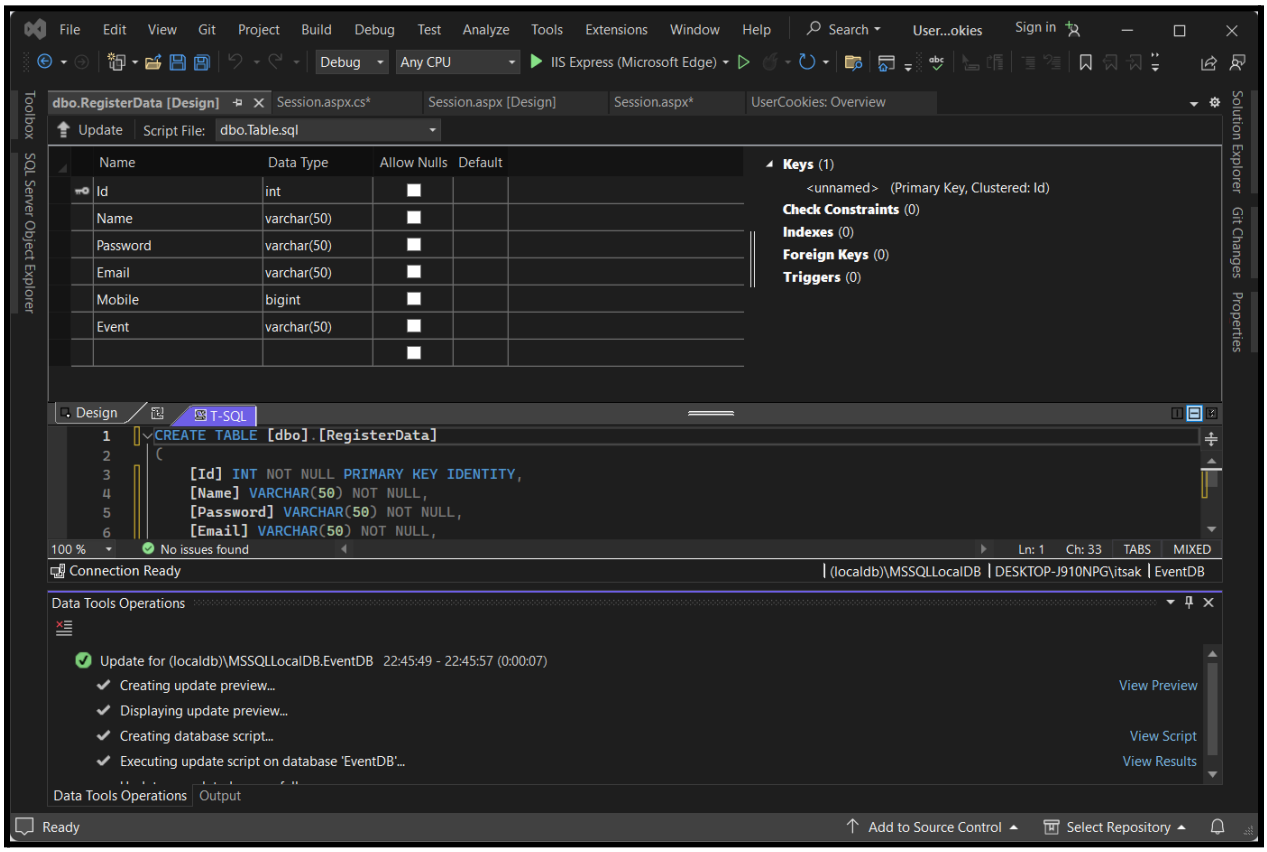
```
            else
            {
                Response.Write("Invalid Username and Password");
            }
        }

        protected void Button2_Click(object sender, EventArgs e)
        {
            Response.Redirect("~/Register.aspx");
        }
    }
}
```

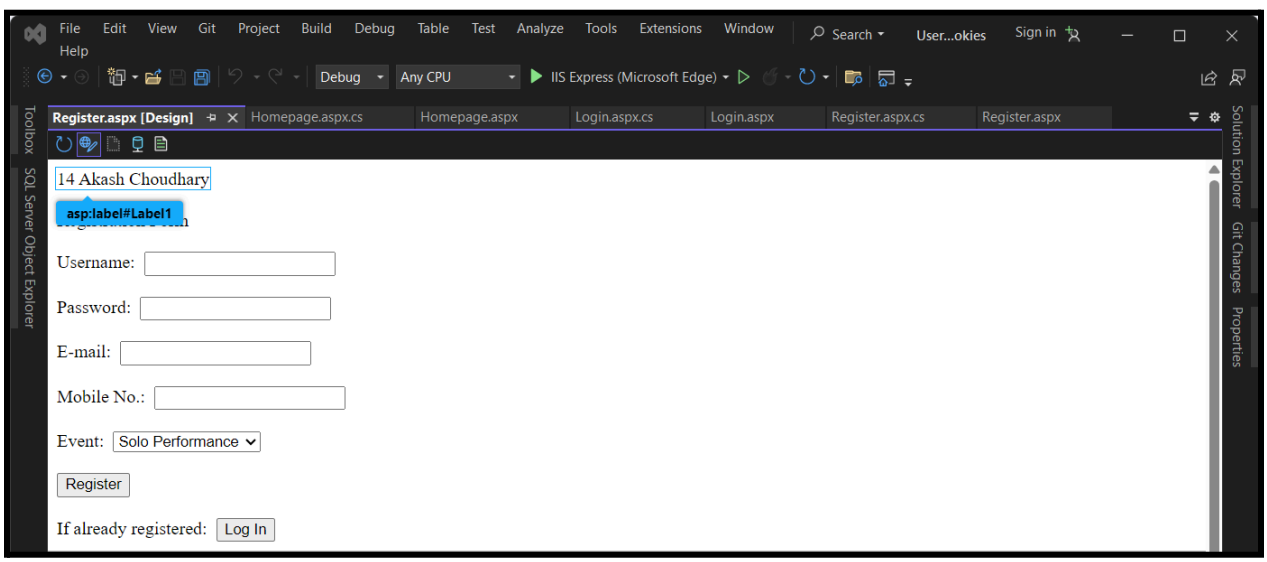
> Homepage.aspx.cs

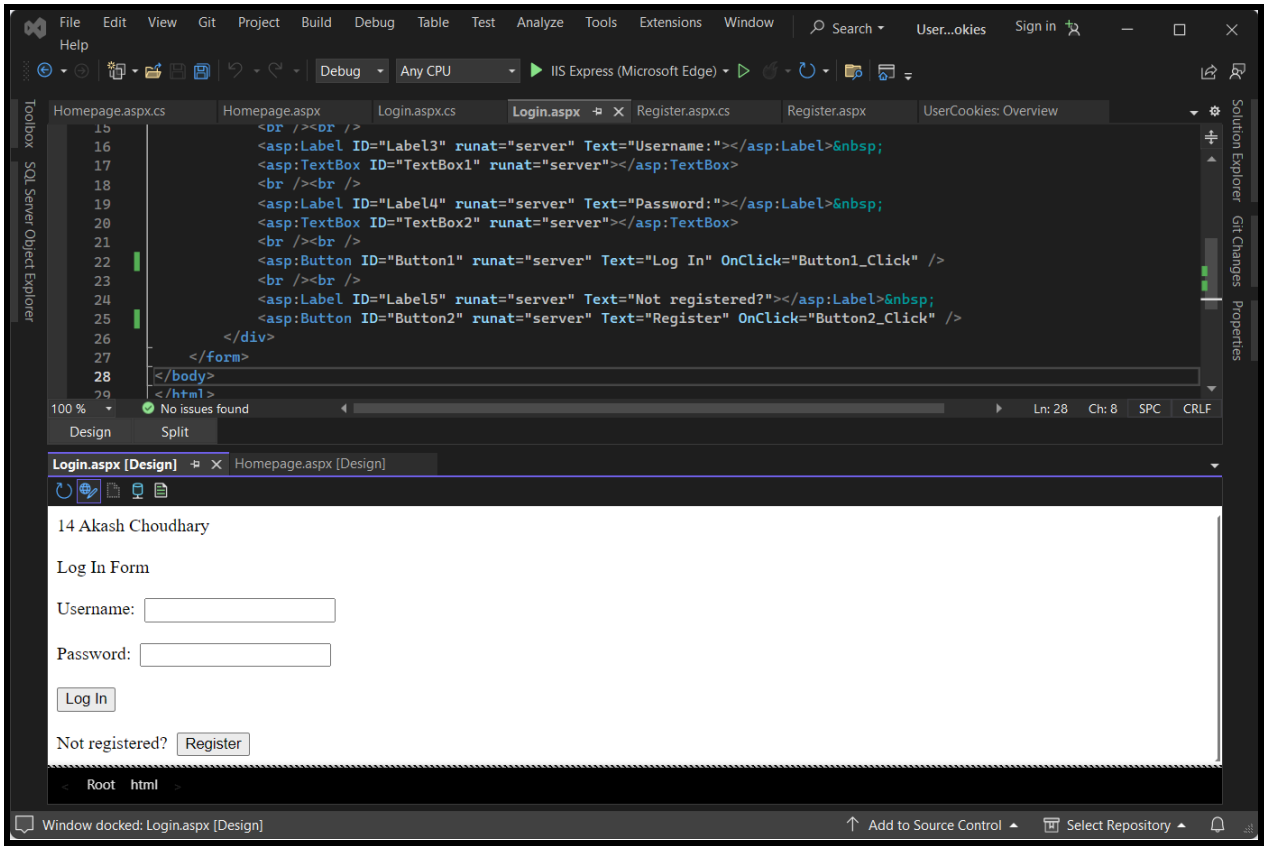
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace UserCookies
{
    public partial class Homepage : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (Session["user"] != null)
            {
                Response.Write("Welcome " + Session["user"].ToString());
                Label2.Text = "You are successfully registered for the college fest.";
            }
            else
            {
                Response.Redirect("~/Register.aspx");
            }

        }

        protected void LinkButton1_Click(object sender, EventArgs e)
        {
            Session.Abandon();
            Session.Remove("user");
            Response.Redirect("~/Login.aspx");

        }
```

```
    }
}
```

Output:


dbo.RegisterData [Design]
Session.aspx.cs*
Session.aspx [Design]
Session.aspx*
UserCookies: Overview
Update
Script File: dbo.Table.sql
Name
Data Type
Allow Nulls
Default
Id int
Name varchar(50)
Password varchar(50)
Email varchar(50)
Mobile bigint
Event varchar(50)
Keys (1)
<unnamed> (Primary Key, Clustered: Id)
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)
Design
T-SQL
CREATE TABLE [dbo].[RegisterData]
(
[Id] INT NOT NULL PRIMARY KEY IDENTITY,
[Name] VARCHAR(50) NOT NULL,
[Password] VARCHAR(50) NOT NULL,
[Email] VARCHAR(50) NOT NULL,
No issues found
Connection Ready
(localdb)\MSSQLLocalDB | DESKTOP-J910NPG\itsak | EventDB
Data Tools Operations
Update for (localdb)\MSSQLLocalDB.EventDB 22:45:49 - 22:45:57 (0:00:07)
Creating update preview...
Displaying update preview...
Creating database script...
Executing update script on database 'EventDB'...
View Preview
View Script
View Results
Ready
Add to Source Control
Select Repository

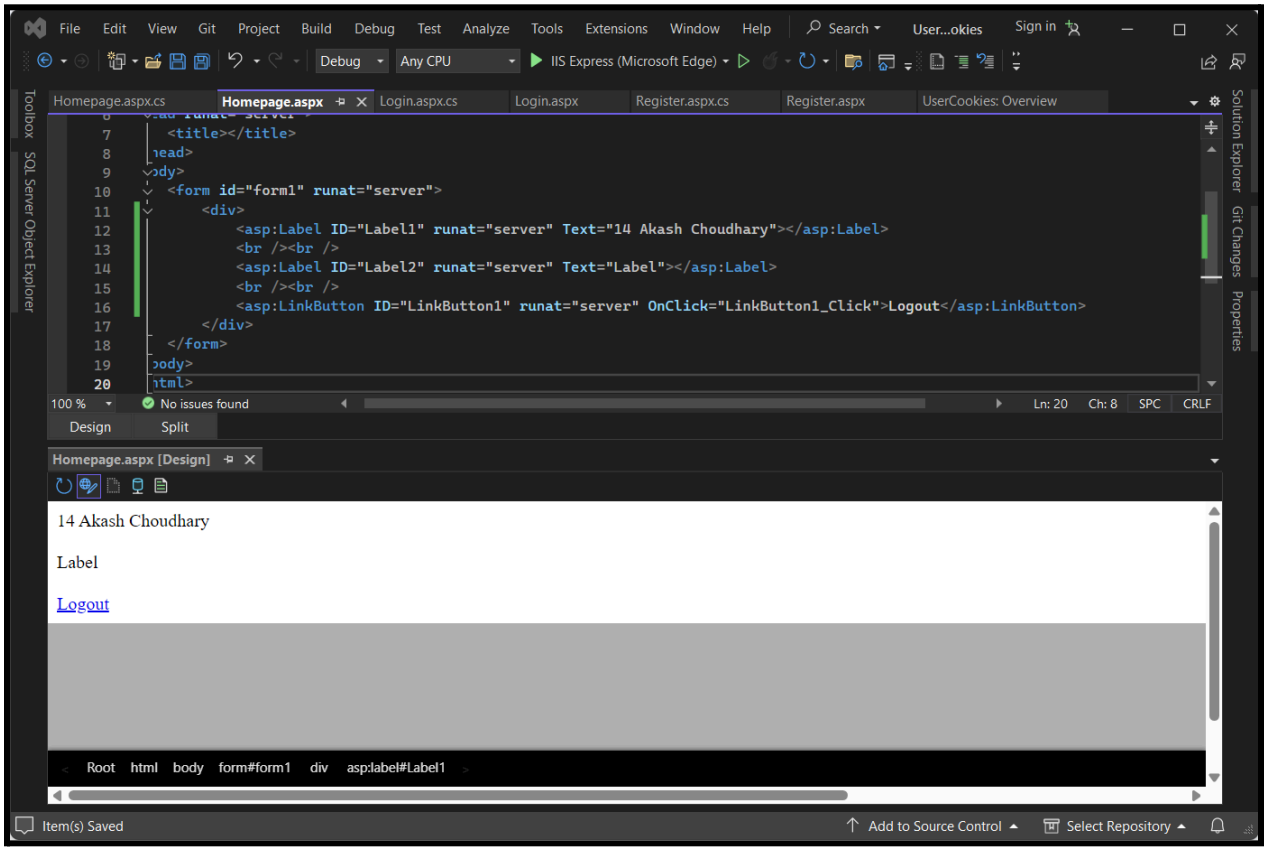

Register.aspx [Design]
Homepage.aspx.cs
Homepage.aspx
Login.aspx.cs
Login.aspx
Register.aspx.cs
Register.aspx
14 Akash Choudhary
asp:label#Label1
Username:
Password:
E-mail:
Mobile No.:
Event:
Solo Performance
Register
If already registered:
Log In


Login.aspx
<asp:Label ID="Label3" runat="server" Text="Username:"></asp:Label>
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

<asp:Label ID="Label4" runat="server" Text="Password:"></asp:Label>
<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>

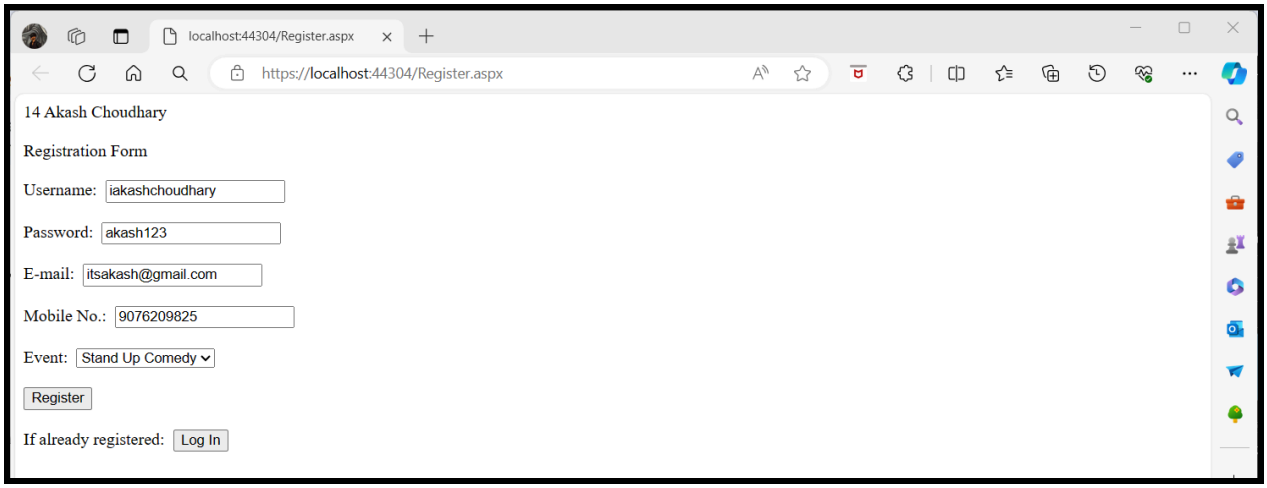
<asp:Button ID="Button1" runat="server" Text="Log In" OnClick="Button1_Click" />

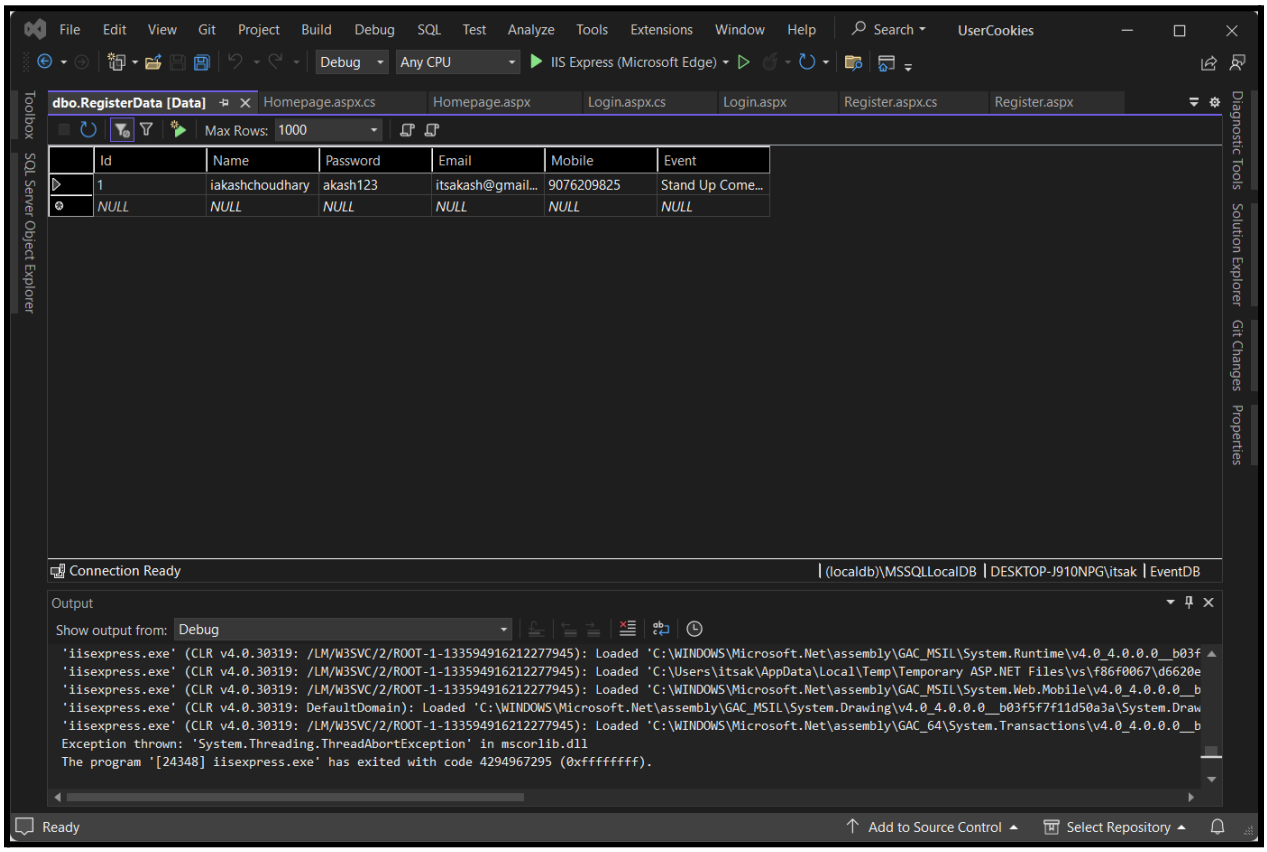
<asp:Label ID="Label5" runat="server" Text="Not registered?"></asp:Label>
<asp:Button ID="Button2" runat="server" Text="Register" OnClick="Button2_Click" />
14 Akash Choudhary
Log In Form
Username:
Password:
Log In
Not registered?
Register

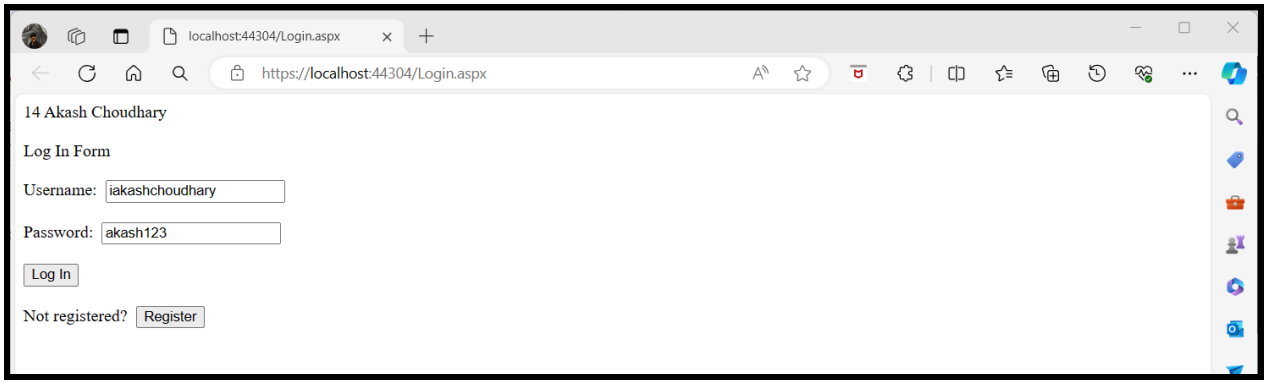

Homepage.aspx
<asp:Label ID="Label1" runat="server" Text="14 Akash Choudhary"></asp:Label>

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

<asp:LinkButton ID="LinkButton1" runat="server" OnClick="LinkButton1_Click">Logout</asp:LinkButton>
14 Akash Choudhary
Label
Logout


localhost:44304/Register.aspx
https://localhost:44304/Register.aspx
14 Akash Choudhary
Registration Form
Username: iakashchoudhary
Password: akash123
E-mail: itsakash@gmail.com
Mobile No.: 9076209825
Event: Stand Up Comedy
Register
If already registered: Log In


File Edit View Git Project Build Debug SQL Test Analyze Tools Extensions Window Help
Search
UserCookies
Debug
Any CPU
IIS Express (Microsoft Edge)
dbo.RegisterData [Data]
Homepage.aspx.cs
Homepage.aspx
Login.aspx.cs
Login.aspx
Register.aspx.cs
Register.aspx
Max Rows: 1000
Id Name Password Email Mobile Event
1 iakashchoudhary akash123 itsakash@gmail... 9076209825 Stand Up Come...
NULL NULL NULL NULL NULL NULL
Connection Ready
(localdb)\MSSQLLocalDB | DESKTOP-J910NPG\itsak | EventDB
Output
Show output from: Debug
'iisexpress.exe' (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-133594916212277945): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\System.Runtime\v4.0_4.0.0.0__b03f
'iisexpress.exe' (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-133594916212277945): Loaded 'C:\Users\itsak\AppData\Local\Temp\Temporary ASP.NET Files\vs\f86f0067\d6620e
'iisexpress.exe' (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-133594916212277945): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\System.Web.Mobile\v4.0_4.0.0.0__b
'iisexpress.exe' (CLR v4.0.30319: DefaultDomain): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\System.Drawing\v4.0_4.0.0.0__b03f5f7f11d50a3a\System.Draw
'iisexpress.exe' (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-133594916212277945): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC_64\System.Transactions\v4.0_4.0.0.0__b
Exception thrown: 'System.Threading.ThreadAbortException' in mscorlib.dll
The program '[24348] iisexpress.exe' has exited with code 4294967295 (0xffffffff).
Ready
Add to Source Control
Select Repository



Conclusion: ASP.NET web applications can efficiently handle user authentication, event registration, and session management using cookies, enhancing the user experience for college fests.

6. Write a program to count the number of live users in your web application.

Aim: To develop a program that tracks and counts the number of active users currently accessing the web application.

Objectives:

1. Track Active Sessions: Monitor and record the active sessions in the web application.
2. Identify Live Users: Determine the number of users currently interacting with the application in real-time.

Theory:

1. Session Management: Each user session is tracked by the web application. When a user accesses the application, a session is created, and a unique session identifier (SessionID) is assigned to that user.
2. Tracking Active Sessions: The web application keeps track of active sessions, usually by maintaining a session state store. This store can be in-memory, database-backed, or using external services like Redis.
3. Counting Live Users: To count the number of live users, the application periodically checks the number of active sessions. This can be done by iterating through the session state store and counting the active sessions.
4. Session Timeout Handling: Sessions have a timeout period after which they expire due to user inactivity. The application needs to handle session timeouts gracefully and remove expired sessions from the count of live users.
5. Displaying Live User Count: Once the count of live users is determined, it can be displayed on a dashboard or admin panel of the web application. This provides real-time insights into the level of user activity.

Code:

> WebForm1.aspx.cs

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

```
namespace VisitorCount
{
   public partial class WebForm1 : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {
         Label2.Text = "TOTAL Visitors " + Application["sessioncount"].ToString();
      }
   }
}
```

> Global.asax.cs

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

namespace VisitorCount
{
   public class Global : System.Web.HttpApplication
   {

      protected void Application_Start(object sender, EventArgs e)
      {
         Application["sessioncount"] = 0;
      }

      protected void Session_Start(object sender, EventArgs e)
      {
         Application.Lock();
         int count = (int)Application["sessioncount"];
         Application["sessioncount"] = count + 1;
         Application.UnLock();
      }

      protected void Application_BeginRequest(object sender, EventArgs e)
      {

      }

      protected void Application_AuthenticateRequest(object sender, EventArgs e)
      {

      }
```

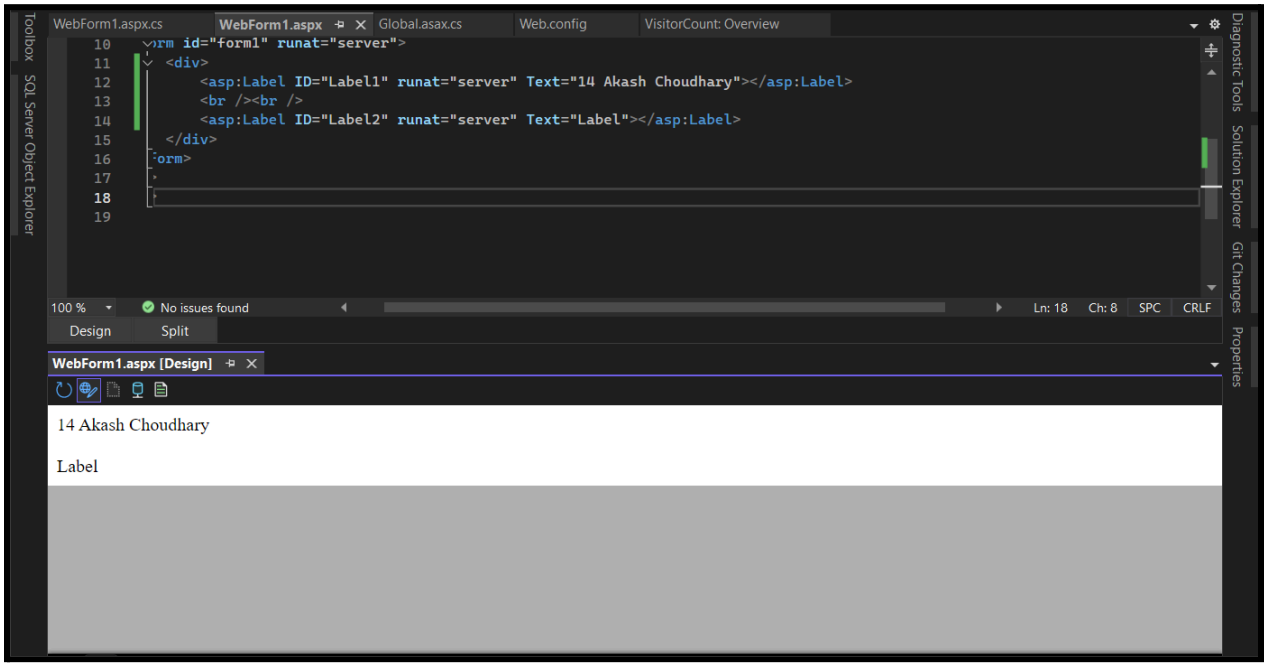
```
        protected void Application_Error(object sender, EventArgs e)
        {

        }

        protected void Session_End(object sender, EventArgs e)
        {
            Application.Lock();
            int count = (int)Application["sessioncount"];
            Application["sessioncount"] = count - 1;
            Application.UnLock();
        }

        protected void Application_End(object sender, EventArgs e)
        {

        }
    }
}
```

> Web.config

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  https://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
  <system.web>
            <sessionState cookieless="UseCookies" mode="InProc" timeout="1"></sessionState>
            <compilation debug="true" targetFramework="4.7.2" />
    <httpRuntime targetFramework="4.7.2" />
  </system.web>
  <system.codedom>
    <compilers>
                                        <compiler language="c#;cs;csharp" extension=".cs"
type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider,
Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.1.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" warningLevel="4"
compilerOptions="/langversion:default /nowarn:1659;1699;1701" />
                                <compiler language="vb;vbs;visualbasic;vbscript" extension=".vb"
type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider,
Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.1.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" warningLevel="4"
compilerOptions="/langversion:default /nowarn:41008 /define:_MYTYPE=\&quot;Web\&quot;
/optionInfer+" />
```

```
    </compilers>
  </system.codedom>
</configuration>
```

Output:



Conclusion: Tracking and counting live users in a web application can be achieved by implementing session management techniques such as tracking active sessions or utilizing real-time monitoring tools, enabling better understanding of user engagement and resource allocation.

7. Write a web application to demonstrate page output caching.

Aim: To showcase how page output caching can improve performance by storing the generated HTML output of a page and serving it to subsequent requests without re-executing the page logic

Objectives:

1. Understanding Page Output Caching: Demonstrate the concept of caching the output generated by a web page to improve performance and reduce server load.
2. Configuring Cache Duration: Implement caching with different cache durations to observe the impact on subsequent page loads.
3. Verifying Cache Effectiveness: Show how cached content is served directly from the cache without executing the page code again, thereby reducing response time.
4. Handling Dynamic Content: Address scenarios where certain parts of the page are dynamic and should not be cached, ensuring accurate and up-to-date information for users.
5. Monitoring Cache Usage: Utilize tools or logs to monitor cache hits and misses, aiding in optimizing cache configuration for better performance.

Theory: Page Output Caching

Page output caching in web applications involves storing the generated output of a page in memory or on disk so that subsequent requests for the same page can be served from the cache instead of regenerating the page content. This improves performance by reducing the processing and rendering time for frequently accessed pages, enhancing the overall responsiveness and scalability of the application.

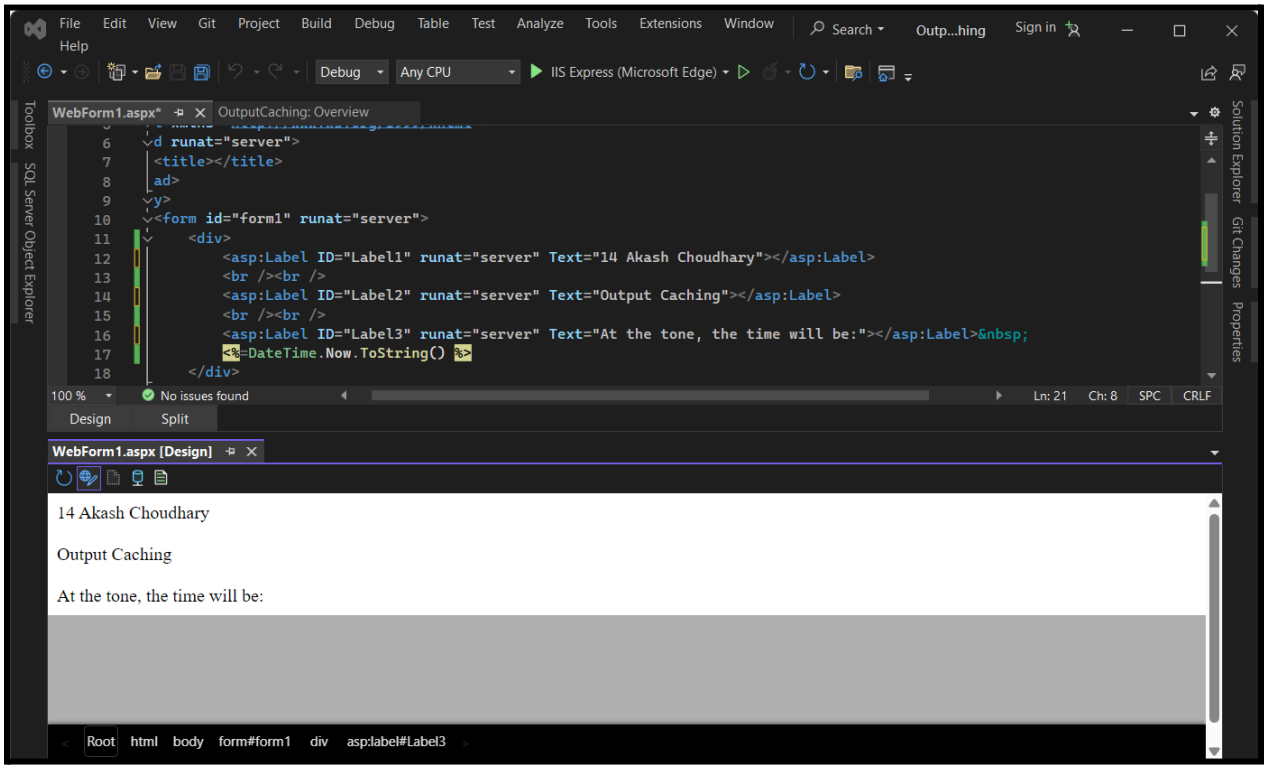
Code:

> WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="OutputCaching.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:Label ID="Label1" runat="server" Text="14 Akash Choudhary"></asp:Label>
        <br /><br />
        <asp:Label ID="Label2" runat="server" Text="Output Caching"></asp:Label>
        <br /><br />
                <asp:Label ID="Label3" runat="server" Text="At the tone, the time will be:"></asp:Label> 
        <%=DateTime.Now.ToString() %>
      </div>
   </form>
</body>
</html>
```

Output:



Conclusion: Page output caching optimizes web application performance by reducing server load and response times, especially for pages with static or infrequently changing content.

8. Write a web application to display a digital clock using AJAX.

Aim: To develop a web application to display a digital clock using AJAX for real-time updating without refreshing the entire page

Objectives:

1. Implementing AJAX: Understand and implement AJAX to asynchronously update the clock without refreshing the entire page, providing a smoother user experience.
2. Real-Time Updates: Ensure that the clock updates in real-time without requiring the user to manually refresh the page, enhancing user engagement and convenience.

Theory:

A digital clock using AJAX involves using JavaScript to periodically make asynchronous requests to the server to fetch the current time, without needing to refresh the entire page. This enables the clock to update dynamically without user intervention, providing a seamless and responsive user experience.

Code:

> WebForm1.aspx

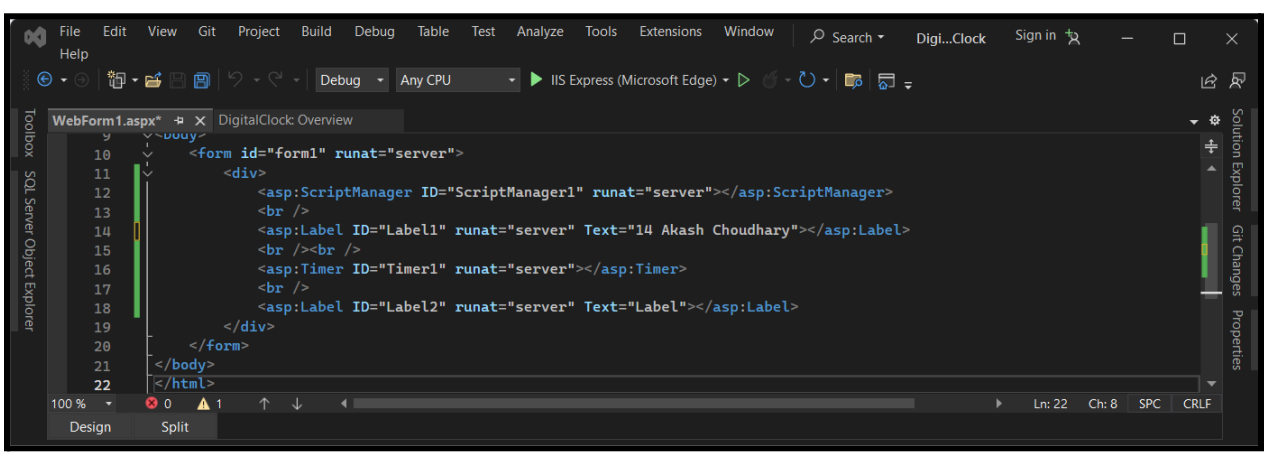
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="DigitalClock.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
      <div>
         <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
         <br />
         <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
               <asp:Label ID="Label1" runat="server" Text="14 Akash Choudhary"></asp:Label>
               <br /><br />
                                    <asp:Timer ID="Timer1" runat="server" Interval="1" OnTick="Timer1_Tick"></asp:Timer>
               <br />
               <asp:Label ID="Label2" runat="server" Text="Label"></asp:Label>
            </ContentTemplate>
```

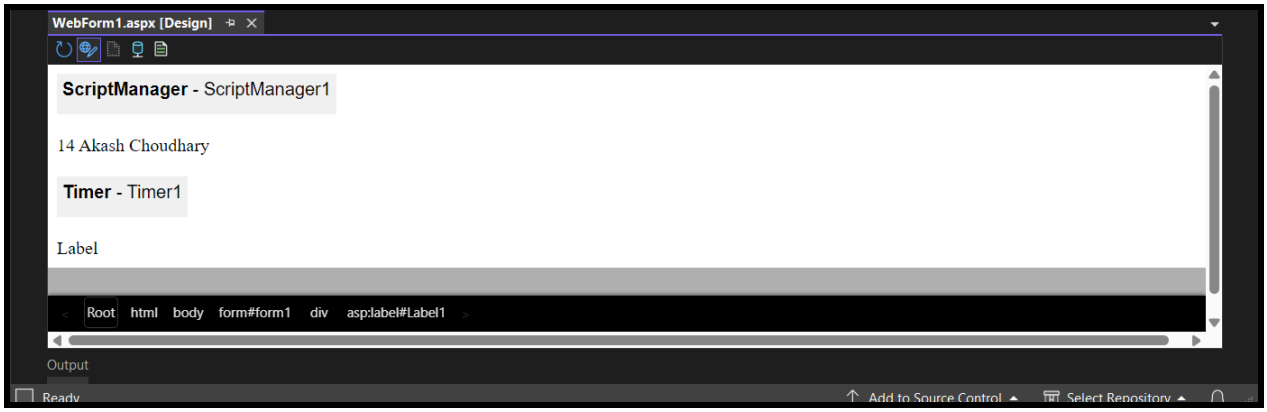
```
            <Triggers>
                <asp:AsyncPostBackTrigger ControlID="Timer1" EventName="tick" />
            </Triggers>
        </asp:UpdatePanel>
    </div>
  </form>
</body>
</html>
```

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

        }

        protected void Timer1_Tick(object sender, EventArgs e)
        {
            Label2.Text = DateTime.Now.ToString();
        }
    }
}
```

Output:





Conclusion: Implementing a digital clock with AJAX enables dynamic and seamless time updates, enhancing user experience without page reloads.

9. Write a web application to create an image slider using AJAX.

Aim: To create an image slider web application using AJAX for dynamic and seamless image transitions

Objectives:

1. Implement AJAX to fetch images asynchronously from the server.
2. Utilize JavaScript for smooth transitions between images.
3. Provide user controls for navigation through the images.

Theory:

AJAX is used to dynamically load images from the server without refreshing the entire page, and JavaScript is employed to create smooth transitions between the images, resulting in an interactive image slider.

Code:

> WebForm1.aspx

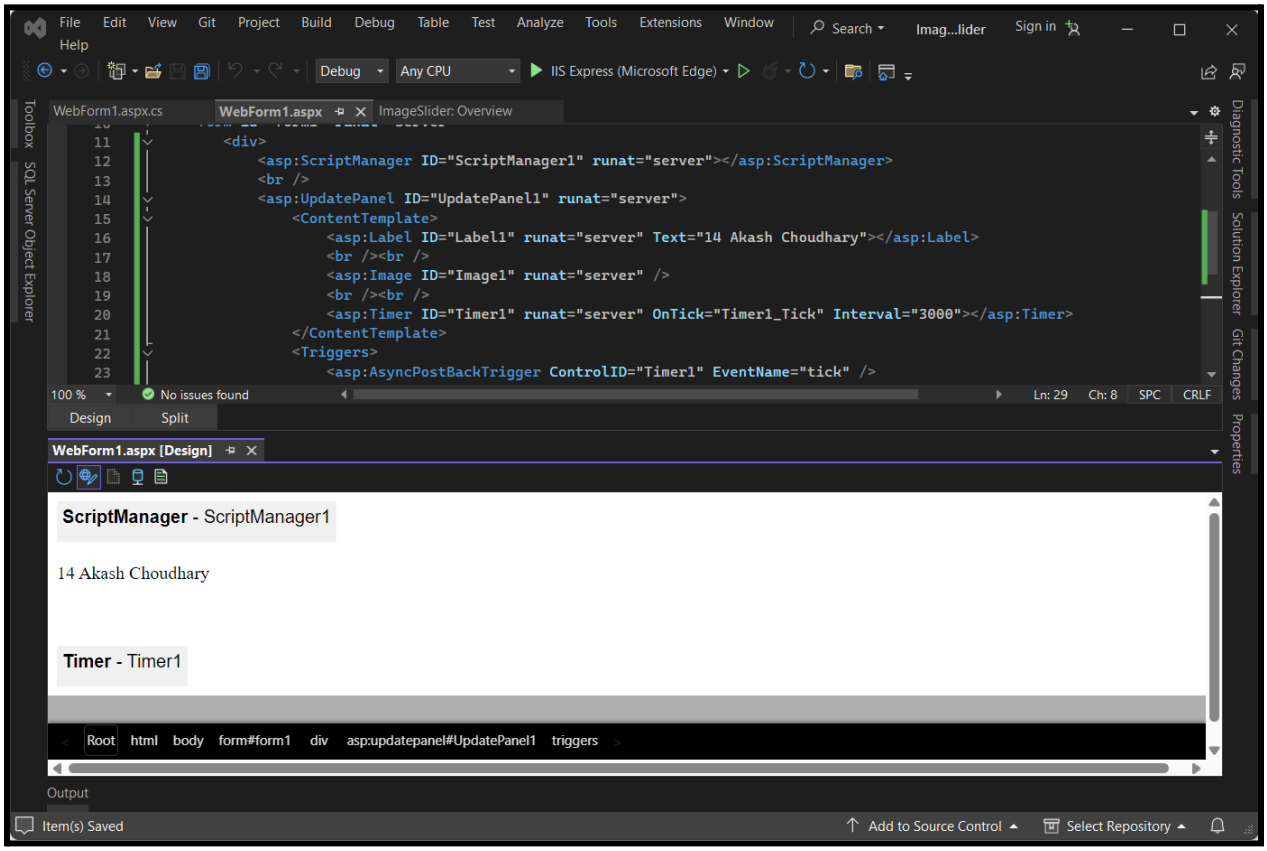
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="ImageSlider.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
      <div>
         <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
         <br />
         <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
               <asp:Label ID="Label1" runat="server" Text="14 Akash Choudhary"></asp:Label>
               <br /><br />
               <asp:Image ID="Image1" runat="server" />
               <br /><br />
                                    <asp:Timer ID="Timer1" runat="server" OnTick="Timer1_Tick" Interval="3000"></asp:Timer>
            </ContentTemplate>
            <Triggers>
               <asp:AsyncPostBackTrigger ControlID="Timer1" EventName="tick" />
            </Triggers>
```

```
        </asp:UpdatePanel>
      </div>
    </form>
</body>
</html>
```

> 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 ImageSlider
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        static int i = 0;

        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Timer1_Tick(object sender, EventArgs e)
        {
            if (i == 0)
            {
                Image1.ImageUrl = "~/img/up.jpeg";
                i = 1;
            }
            else if (i == 1)
            {
                Image1.ImageUrl = "~/img/car.jpeg";
                i = 2;
            }
            else
            {
                Image1.ImageUrl = "~/img/bighero6.jpeg";
                i = 0;
            }

        }
    }
}
```

Output:

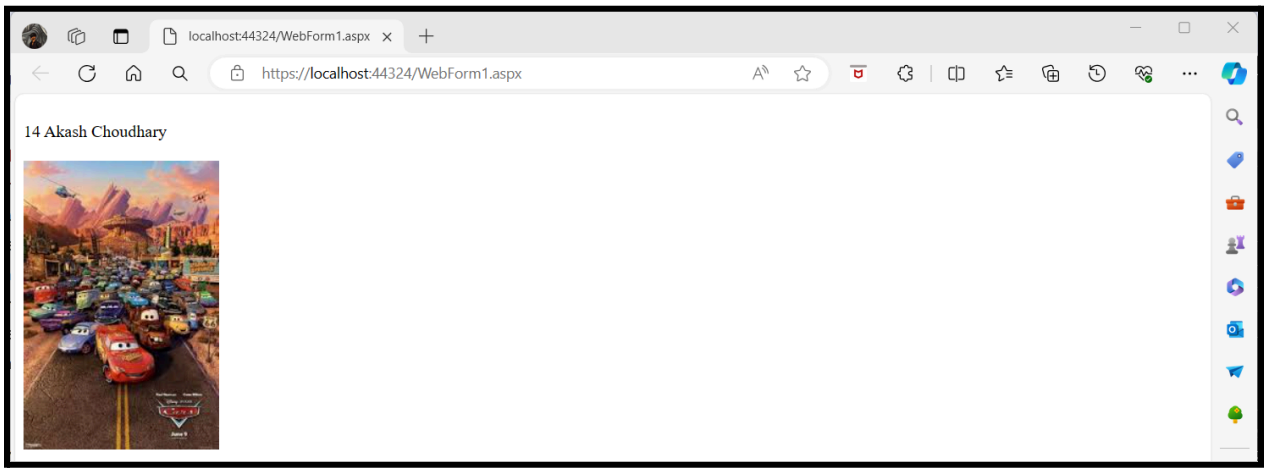

File Edit View Git Project Build Debug Table Test Analyze Tools Extensions Window Help
Search
Imag...lider
Sign in
Debug
Any CPU
IIS Express (Microsoft Edge)
WebForm1.aspx.cs
WebForm1.aspx
ImageSlider: Overview

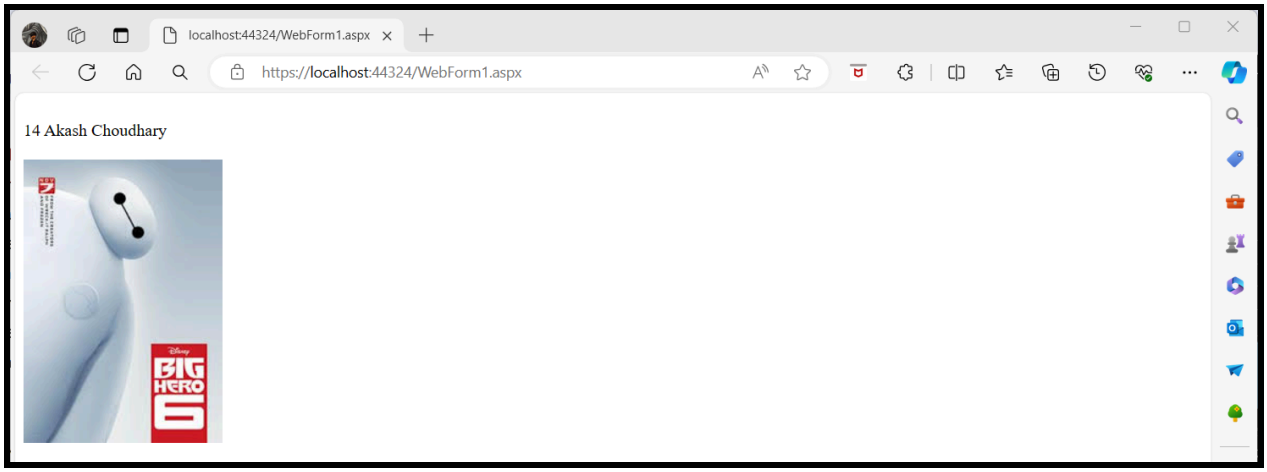
<asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>

<asp:UpdatePanel ID="UpdatePanel1" runat="server">
<ContentTemplate>
<asp:Label ID="Label1" runat="server" Text="14 Akash Choudhary"></asp:Label>

<asp:Image ID="Image1" runat="server" />

<asp:Timer ID="Timer1" runat="server" OnTick="Timer1_Tick" Interval="3000"></asp:Timer>
</ContentTemplate>
<Triggers>
<asp:AsyncPostBackTrigger ControlID="Timer1" EventName="tick" />
100 %
No issues found
Ln: 29
Ch: 8
SPC
CRLF
Design
Split
WebForm1.aspx [Design]
ScriptManager - ScriptManager1
14 Akash Choudhary
Timer - Timer1
Root html body form#form1 div asp:updatepanel#UpdatePanel1 triggers
Output
Item(s) Saved
Add to Source Control
Select Repository


localhost:44324/WebForm1.aspx
https://localhost:44324/WebForm1.aspx
14 Akash Choudhary



Conclusion: Implementing an image slider using AJAX enhances user experience by enabling dynamic loading of images and smooth transitions between them, contributing to a more engaging web application.