Advanced Web Programming Practical
B. Sc. (Information Technology) Semester – V

By munotes.in

B. Sc. (Information Technology)		Semester – V	
Course Name: Advanced Web Programming Practical		**Course Code: USIT5P3**	
Periods per week (1 Period is 50 minutes)		**3**	
Credits		**2**	
		Hours	**Marks**
Evaluation System	**Practical Examination**	**2½**	**50**
	Internal	**--**	**--**

List of Practical	
1.	**Working with basic C# and ASP .NET**
a.	Create an application that obtains four int values from the user and displays the product.
b.	Create an application to demonstrate string operations.
c.	Create an application that receives the (Student Id, Student Name, Course Name, Date of Birth) information from a set of students. The application should also display the information of all the students once the data entered.
d.	Create an application to demonstrate following operations i. Generate Fibonacci series. ii. Test for prime numbers. iii. Test for vowels. iv. Use of foreach loop with arrays v. Reverse a number and find sum of digits of a number.
2.	**Working with Object Oriented C# and ASP .NET**
a.	Create simple application to perform following operations i. Finding factorial Value ii. Money Conversion iii. Quadratic Equation iv. Temperature Conversion
b.	Create simple application to demonstrate use of following concepts i. Function Overloading ii. Inheritance (all types) iii. Constructor overloading iv. Interfaces
c.	Create simple application to demonstrate use of following concepts i. Using Delegates and events ii. Exception handling
3.	**Working with Web Forms and Controls**
a.	Create a simple web page with various sever controls to demonstrate setting and use of their properties. (Example : AutoPostBack)
b.	Demonstrate the use of Calendar control to perform following operations. a) Display messages in a calendar control b) Display vacation in a calendar control c) Selected day in a calendar control using style d) Difference between two calendar dates
c.	Demonstrate the use of Treeview control perform following operations.

	a) Treeview control and datalist b) Treeview operations
4.	**Working with Form Controls**
a.	Create a Registration form to demonstrate use of various Validation controls.
b.	Create Web Form to demonstrate use of Adrotator Control.
c.	Create Web Form to demonstrate use User Controls.
5.	**Working with Navigation, Beautification and Master page.**
a.	Create Web Form to demonstrate use of Website Navigation controls and Site Map.
b.	Create a web application to demonstrate use of Master Page with applying Styles and Themes for page beautification.
c.	Create a web application to demonstrate various states of ASP.NET Pages.
6.	**Working with Database**
a.	Create a web application bind data in a multiline textbox by querying in another textbox.
b.	Create a web application to display records by using database.
c.	Demonstrate the use of Datalist link control.
7.	**Working with Database**
a.	Create a web application to display Databinding using dropdownlist control.
b.	Create a web application for to display the phone no of an author using database.
c.	Create a web application for inserting and deleting record from a database. (Using Execute-Non Query).
8.	**Working with data controls**
a.	Create a web application to demonstrate various uses and properties of SqlDataSource.
b.	Create a web application to demonstrate data binding using DetailsView and FormView Control.
c.	Create a web application to display Using Disconnected Data Access and Databinding using GridView.
9.	**Working with GridView control**
a.	Create a web application to demonstrate use of GridView control template and GridView hyperlink.
b.	Create a web application to demonstrate use of GridView button column and GridView events.
c.	Create a web application to demonstrate GridView paging and Creating own table format using GridView.
10.	**Working with AJAX and XML**
a.	Create a web application to demonstrate reading and writing operation with XML.
b.	Create a web application to demonstrate Form Security and Windows Security with proper Authentication and Authorization properties.
c.	Create a web application to demonstrate use of various Ajax controls.
11.	**Programs to create and use DLL**

1. Working with basic C# and ASP .NET

a. Create an application that obtains four int values from the user and displays the product.

Solution

```
using System;

class Program
{
    static void Main()
    {
        int num1, num2, num3, num4;

        Console.WriteLine("Enter the first number:");
        num1 = Convert.ToInt32(Console.ReadLine());

        Console.WriteLine("Enter the second number:");
        num2 = Convert.ToInt32(Console.ReadLine());

        Console.WriteLine("Enter the third number:");
        num3 = Convert.ToInt32(Console.ReadLine());

        Console.WriteLine("Enter the fourth number:");
        num4 = Convert.ToInt32(Console.ReadLine());

        int product = num1 * num2 * num3 * num4;
        Console.WriteLine("The product of the four numbers is: " + product);

        Console.ReadLine();
    }
}
```

b. Create an application to demonstrate string operations.

Solution

```
using System;

class Program
{
    static void Main()
    {
        string inputString;

        Console.WriteLine("Enter a string:");
        inputString = Console.ReadLine();

        Console.WriteLine("Original string: " + inputString);
        Console.WriteLine("Length: " + inputString.Length);
        Console.WriteLine("Uppercase: " + inputString.ToUpper());
        Console.WriteLine("Lowercase: " + inputString.ToLower());
        Console.WriteLine("Substring (2, 5): " + inputString.Substring(2, 5));
        Console.WriteLine("Replace 'a' with 'e': " + inputString.Replace("a", "e"));
        Console.WriteLine("Concatenation: " + inputString + " is concatenated with " + inputString);
        Console.WriteLine("Contains 'hello': " + inputString.Contains("hello"));
        Console.WriteLine("Starts with 'abc': " + inputString.StartsWith("abc"));
        Console.WriteLine("Ends with 'xyz': " + inputString.EndsWith("xyz"));

        Console.ReadLine();
```

```
    }
}
```

c. Create an application that receives the (Student Id, Student Name, Course Name, Date of
Birth) information from a set of students. The application should also display the
information of all the students once the data entered.

Solution

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

class Student
{
    public string StudentId { get; set; }
    public string StudentName { get; set; }
    public string CourseName { get; set; }
    public DateTime DateOfBirth { get; set; }
}

class Program
{
    static void Main()
    {
        List<Student> students = new List<Student>();

        while (true)
        {
```

```
            Console.WriteLine("Enter student information or type 'exit' to stop:");

            Console.Write("Student ID: ");
            string studentId = Console.ReadLine();

            if (studentId.ToLower() == "exit")
                break;

            Console.Write("Student Name: ");
            string studentName = Console.ReadLine();

            Console.Write("Course Name: ");
            string courseName = Console.ReadLine();

            Console.Write("Date of Birth (dd/mm/yyyy): ");
            DateTime dateOfBirth = DateTime.ParseExact(Console.ReadLine(),
"dd/MM/yyyy", null);

            Student student = new Student
            {
                StudentId = studentId,
                StudentName = studentName,
                CourseName = courseName,
                DateOfBirth = dateOfBirth
            };

            students.Add(student);

            Console.WriteLine("Student information added successfully!\n");
        }

        Console.WriteLine("\nAll Student Information:");
```

```
        foreach (var student in students)
        {
            Console.WriteLine("Student ID: " + student.StudentId);
            Console.WriteLine("Student Name: " + student.StudentName);
            Console.WriteLine("Course Name: " + student.CourseName);
            Console.WriteLine("Date of Birth: " +
student.DateOfBirth.ToString("dd/MM/yyyy"));
            Console.WriteLine("--------------------------------");
        }

        Console.ReadLine();
    }
}
```

d. Create an application to demonstrate following operations
i. Generate Fibonacci series.

Solution

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter the number of terms in the Fibonacci series: ");
        int numTerms = int.Parse(Console.ReadLine());
```

```
        Console.WriteLine("Fibonacci Series:");

        for (int i = 0; i < numTerms; i++)
        {
            int fibonacci = CalculateFibonacci(i);
            Console.Write(fibonacci + " ");
        }

        Console.ReadLine();
    }

    static int CalculateFibonacci(int n)
    {
        if (n <= 1)
            return n;

        int a = 0;
        int b = 1;
        int fibonacci = 0;

        for (int i = 2; i <= n; i++)
        {
            fibonacci = a + b;
            a = b;
            b = fibonacci;
        }

        return fibonacci;
    }
}
```

ii. Test for prime numbers.

Solution

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter a number to check if it's prime: ");
        int number = int.Parse(Console.ReadLine());

        bool isPrime = IsPrime(number);

        if (isPrime)
            Console.WriteLine(number + " is a prime number.");
        else
            Console.WriteLine(number + " is not a prime number.");

        Console.ReadLine();
    }

    static bool IsPrime(int number)
    {
        if (number < 2)
            return false;

        for (int i = 2; i <= Math.Sqrt(number); i++)
        {
            if (number % i == 0)
```

```
            return false;
        }

        return true;
    }
}
```

iii. Test for vowels. iv. Use of foreach loop with arrays

Solution

```
using System;

class Program
{
    static void Main()
    {
        // Test for vowels
        Console.Write("Enter a character to test if it's a vowel: ");
        char character = char.ToLower(Console.ReadKey().KeyChar);

        bool isVowel = IsVowel(character);

        if (isVowel)
            Console.WriteLine("\n" + character + " is a vowel.");
        else
            Console.WriteLine("\n" + character + " is not a vowel.");

        // Use of foreach loop with arrays
        string[] names = { "Alice", "Bob", "Charlie", "Dave", "Eve" };
```

```
        Console.WriteLine("\nNames in the array:");
        foreach (string name in names)
        {
            Console.WriteLine(name);
        }

        Console.ReadLine();
    }

    static bool IsVowel(char character)
    {
        char[] vowels = { 'a', 'e', 'i', 'o', 'u' };

        foreach (char vowel in vowels)
        {
            if (character == vowel)
                return true;
        }

        return false;
    }
}
```

v. Reverse a number and find sum of digits of a number.

Solution

```
using System;
```

```
class Program
{
    static void Main()
    {
        // Reverse a number
        Console.Write("Enter a number to reverse: ");
        int numberToReverse = int.Parse(Console.ReadLine());

        int reversedNumber = ReverseNumber(numberToReverse);
        Console.WriteLine("Reversed number: " + reversedNumber);

        // Find sum of digits of a number
        Console.Write("Enter a number to find the sum of its digits: ");
        int numberToSumDigits = int.Parse(Console.ReadLine());

        int sumOfDigits = SumOfDigits(numberToSumDigits);
        Console.WriteLine("Sum of digits: " + sumOfDigits);

        Console.ReadLine();
    }

    static int ReverseNumber(int number)
    {
        int reversedNumber = 0;

        while (number != 0)
        {
            int digit = number % 10;
            reversedNumber = (reversedNumber * 10) + digit;
            number /= 10;
        }
```

```
        return reversedNumber;
    }

    static int SumOfDigits(int number)
    {
        int sum = 0;

        while (number != 0)
        {
            int digit = number % 10;
            sum += digit;
            number /= 10;
        }

        return sum;
    }
}
```

2. Working with Object Oriented C# and ASP .NET

a. Create simple application to perform following operations

i. Finding factorial Value

Solution

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter a number to find its factorial: ");
        int number = int.Parse(Console.ReadLine());

        long factorial = CalculateFactorial(number);

        Console.WriteLine("Factorial of " + number + " is: " + factorial);

        Console.ReadLine();
    }

    static long CalculateFactorial(int number)
    {
        if (number < 0)
            throw new ArgumentException("Factorial is not defined for negative
numbers.");

        long result = 1;

        for (int i = 1; i <= number; i++)
        {
```

```
            result *= i;
        }

        return result;
    }
}
```

ii. Money Conversion

Solution

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter the amount in USD: ");
        double usdAmount = double.Parse(Console.ReadLine());

        Console.WriteLine("Choose the currency to convert to:");
        Console.WriteLine("1. EUR");
        Console.WriteLine("2. GBP");
        Console.WriteLine("3. JPY");
        Console.WriteLine("4. AUD");
        Console.WriteLine("5. CAD");

        Console.Write("Enter your choice (1-5): ");
        int choice = int.Parse(Console.ReadLine());
```

```
        double convertedAmount = ConvertCurrency(usdAmount, choice);

        Console.WriteLine("Converted amount: " + convertedAmount);

        Console.ReadLine();
    }

    static double ConvertCurrency(double usdAmount, int choice)
    {
        double conversionRate = 0;

        switch (choice)
        {
            case 1: // EUR
                conversionRate = 0.85;
                break;
            case 2: // GBP
                conversionRate = 0.73;
                break;
            case 3: // JPY
                conversionRate = 110.76;
                break;
            case 4: // AUD
                conversionRate = 1.36;
                break;
            case 5: // CAD
                conversionRate = 1.25;
                break;
            default:
                throw new ArgumentException("Invalid choice. Please select a
number from 1 to 5.");
        }
```

```
        double convertedAmount = usdAmount * conversionRate;
        return convertedAmount;
    }
}
```

iii. Quadratic Equation

Solution

```
using System;

class Program
{
    static void Main()
    {
        Console.WriteLine("Enter the coefficients of the quadratic equation
(ax^2 + bx + c = 0):");

        Console.Write("a: ");
        double a = double.Parse(Console.ReadLine());

        Console.Write("b: ");
        double b = double.Parse(Console.ReadLine());

        Console.Write("c: ");
        double c = double.Parse(Console.ReadLine());

        SolveQuadraticEquation(a, b, c);

        Console.ReadLine();
```

```
    }

    static void SolveQuadraticEquation(double a, double b, double c)
    {
        double discriminant = (b * b) - (4 * a * c);

        if (discriminant > 0)
        {
            double root1 = (-b + Math.Sqrt(discriminant)) / (2 * a);
            double root2 = (-b - Math.Sqrt(discriminant)) / (2 * a);

            Console.WriteLine("Root 1: " + root1);
            Console.WriteLine("Root 2: " + root2);
        }
        else if (discriminant == 0)
        {
            double root = -b / (2 * a);

            Console.WriteLine("The equation has a repeated root: " + root);
        }
        else
        {
            Console.WriteLine("The equation has complex roots.");
        }
    }
}
```

iv. Temperature Conversion

Solution

```
using System;

class Program
{
    static void Main()
    {
        Console.WriteLine("Temperature Conversion:");
        Console.WriteLine("1. Celsius to Fahrenheit");
        Console.WriteLine("2. Fahrenheit to Celsius");

        Console.Write("Enter your choice (1 or 2): ");
        int choice = int.Parse(Console.ReadLine());

        double temperature;

        switch (choice)
        {
            case 1:
                Console.Write("Enter the temperature in Celsius: ");
                temperature = double.Parse(Console.ReadLine());
                double celsiusToFahrenheit = CelsiusToFahrenheit(temperature);
                Console.WriteLine("Temperature in Fahrenheit: " +
celsiusToFahrenheit);
                break;
            case 2:
                Console.Write("Enter the temperature in Fahrenheit: ");
                temperature = double.Parse(Console.ReadLine());
                double fahrenheitToCelsius = FahrenheitToCelsius(temperature);
                Console.WriteLine("Temperature in Celsius: " +
fahrenheitToCelsius);
                break;
            default:
```

```
            Console.WriteLine("Invalid choice. Please select either 1 or 2.");
            break;
        }

        Console.ReadLine();
    }

    static double CelsiusToFahrenheit(double celsius)
    {
        double fahrenheit = (celsius * 9 / 5) + 32;
        return fahrenheit;
    }

    static double FahrenheitToCelsius(double fahrenheit)
    {
        double celsius = (fahrenheit - 32) * 5 / 9;
        return celsius;
    }
}
```

b. Create simple application to demonstrate use of following concepts
i. Function Overloading

Solution

```
using System;

class Program
{
    static void Main()
    {
```

```
        Console.WriteLine("Function Overloading Demo:");

        // Calling the Add method with different argument types
        int sum1 = Add(5, 10);
        Console.WriteLine("Sum of integers: " + sum1);

        double sum2 = Add(3.5, 2.7);
        Console.WriteLine("Sum of doubles: " + sum2);

        int sum3 = Add(2, 4, 6);
        Console.WriteLine("Sum of three integers: " + sum3);

        Console.ReadLine();
    }

    static int Add(int a, int b)
    {
        return a + b;
    }

    static double Add(double a, double b)
    {
        return a + b;
    }

    static int Add(int a, int b, int c)
    {
        return a + b + c;
    }
}
```

ii. Inheritance (all types)

Solution

```
using System;

// Base class
class Animal
{
    public void Eat()
    {
        Console.WriteLine("Animal is eating.");
    }
}

// Single Inheritance
class Dog : Animal
{
    public void Bark()
    {
        Console.WriteLine("Dog is barking.");
    }
}

// Multilevel Inheritance
class GermanShepherd : Dog
{
    public void Guard()
    {
        Console.WriteLine("German Shepherd is guarding.");
    }
}
```

```
// Hierarchical Inheritance
class Cat : Animal
{
    public void Meow()
    {
        Console.WriteLine("Cat is meowing.");
    }
}

class Program
{
    static void Main()
    {
        // Single Inheritance
        Dog dog = new Dog();
        dog.Eat();
        dog.Bark();
        Console.WriteLine();

        // Multilevel Inheritance
        GermanShepherd germanShepherd = new GermanShepherd();
        germanShepherd.Eat();
        germanShepherd.Bark();
        germanShepherd.Guard();
        Console.WriteLine();

        // Hierarchical Inheritance
        Cat cat = new Cat();
        cat.Eat();
        cat.Meow();
```

```
        Console.ReadLine();
    }
}
```

iii. Constructor overloading

Solution

```
using System;

class Rectangle
{
    private int length;
    private int width;

    // Default constructor
    public Rectangle()
    {
        length = 0;
        width = 0;
    }

    // Parameterized constructor with one parameter
    public Rectangle(int sideLength)
    {
        length = sideLength;
        width = sideLength;
    }

    // Parameterized constructor with two parameters
    public Rectangle(int rectangleLength, int rectangleWidth)
```

```
    {
        length = rectangleLength;
        width = rectangleWidth;
    }

    // Method to calculate the area of the rectangle
    public int CalculateArea()
    {
        return length * width;
    }
}

class Program
{
    static void Main()
    {
        Rectangle defaultRectangle = new Rectangle();
        Console.WriteLine("Area of default rectangle: " +
defaultRectangle.CalculateArea());

        Rectangle square = new Rectangle(5);
        Console.WriteLine("Area of square: " + square.CalculateArea());

        Rectangle rectangle = new Rectangle(4, 6);
        Console.WriteLine("Area of rectangle: " + rectangle.CalculateArea());

        Console.ReadLine();
    }
}
```

iv. Interfaces

Solution

```
using System;

// Interface
interface IShape
{
    double CalculateArea();
    double CalculatePerimeter();
}

// Class implementing the interface
class Circle : IShape
{
    private double radius;

    public Circle(double circleRadius)
    {
        radius = circleRadius;
    }

    public double CalculateArea()
    {
        return Math.PI * radius * radius;
    }

    public double CalculatePerimeter()
    {
        return 2 * Math.PI * radius;
    }
}
```

```
class Rectangle : IShape
{
    private double length;
    private double width;

    public Rectangle(double rectangleLength, double rectangleWidth)
    {
        length = rectangleLength;
        width = rectangleWidth;
    }

    public double CalculateArea()
    {
        return length * width;
    }

    public double CalculatePerimeter()
    {
        return 2 * (length + width);
    }
}

class Program
{
    static void Main()
    {
        Circle circle = new Circle(5);
        Console.WriteLine("Circle Area: " + circle.CalculateArea());
        Console.WriteLine("Circle Perimeter: " + circle.CalculatePerimeter());

        Rectangle rectangle = new Rectangle(4, 6);
```

```
        Console.WriteLine("Rectangle Area: " + rectangle.CalculateArea());
        Console.WriteLine("Rectangle Perimeter: " +
rectangle.CalculatePerimeter());

        Console.ReadLine();
    }
}
```

c. Create simple application to demonstrate use of following concepts
i. Using Delegates and events

Solution

```
using System;

// Delegate declaration
delegate void NumberChangedEventHandler(int number);

// Event publisher class
class NumberProcessor
{
    // Event declaration
    public event NumberChangedEventHandler NumberChanged;

    public void ProcessNumber(int number)
    {
        Console.WriteLine("Processing number: " + number);

        // Raise the event
        OnNumberChanged(number);
    }
```

```
    protected virtual void OnNumberChanged(int number)
    {
        // Check if there are any subscribers to the event
        if (NumberChanged != null)
        {
            // Invoke the event
            NumberChanged(number);
        }
    }
}

// Event subscriber class
class Display
{
    public void Subscribe(NumberProcessor processor)
    {
        // Subscribe to the event
        processor.NumberChanged += NumberChangedEventHandler;
    }

    public void Unsubscribe(NumberProcessor processor)
    {
        // Unsubscribe from the event
        processor.NumberChanged -= NumberChangedEventHandler;
    }

    private void NumberChangedEventHandler(int number)
    {
        Console.WriteLine("Number has changed: " + number);
    }
}
```

```
class Program
{
    static void Main()
    {
        NumberProcessor processor = new NumberProcessor();
        Display display = new Display();

        // Subscribe to the event
        display.Subscribe(processor);

        // Process numbers
        processor.ProcessNumber(10);
        processor.ProcessNumber(20);
        processor.ProcessNumber(30);

        // Unsubscribe from the event
        display.Unsubscribe(processor);

        // Process number (after unsubscribing)
        processor.ProcessNumber(40);

        Console.ReadLine();
    }
}
```

ii. Exception handling

Solution

```
using System;

class Program
{
    static void Main()
    {
        try
        {
            Console.Write("Enter a number: ");
            int number = int.Parse(Console.ReadLine());

            int result = DivideByTwo(number);
            Console.WriteLine("Result: " + result);
        }
        catch (FormatException)
        {
            Console.WriteLine("Invalid input. Please enter a valid number.");
        }
        catch (DivideByZeroException)
        {
            Console.WriteLine("Error: Cannot divide by zero.");
        }
        catch (Exception ex)
        {
            Console.WriteLine("An error occurred: " + ex.Message);
        }

        Console.ReadLine();
    }

    static int DivideByTwo(int number)
    {
```

```
        if (number == 0)
            throw new DivideByZeroException();

        return number / 2;
    }
}
```

3. Working with Web Forms and Controls

a. Create a simple web page with various sever controls to demonstrate setting and use of
their properties. (Example : AutoPostBack)

Solution

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>WebForm Example</title>
</head>
<body>
    <form id="form1" runat="server">
        <h1>WebForm Example</h1>

        <h2>TextBox:</h2>
        <asp:TextBox ID="txtName" runat="server"></asp:TextBox>

        <h2>Button:</h2>
        <asp:Button ID="btnSubmit" runat="server" Text="Submit"
OnClick="btnSubmit_Click" />

        <h2>Label:</h2>
        <asp:Label ID="lblResult" runat="server"></asp:Label>

        <h2>CheckBox:</h2>
```

```
        <asp:CheckBox ID="chkAgree" runat="server" Text="I agree to the
terms and conditions" />

        <h2>DropDownList:</h2>
        <asp:DropDownList ID="ddlColors" runat="server">
            <asp:ListItem Text="Red" Value="red"></asp:ListItem>
            <asp:ListItem Text="Blue" Value="blue"></asp:ListItem>
            <asp:ListItem Text="Green" Value="green"></asp:ListItem>
        </asp:DropDownList>

        <h2>RadioButtonList:</h2>
        <asp:RadioButtonList ID="rblFruits" runat="server">
            <asp:ListItem Text="Apple" Value="apple"></asp:ListItem>
            <asp:ListItem Text="Banana" Value="banana"></asp:ListItem>
            <asp:ListItem Text="Orange" Value="orange"></asp:ListItem>
        </asp:RadioButtonList>
    </form>
</body>
</html>
```

b. Demonstrate the use of Calendar control to perform following operations.
a) Display messages in a calendar control

Solution

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Calendar Example</title>
</head>
<body>
    <form id="form1" runat="server">
        <h1>Calendar Example</h1>

        <asp:Calendar ID="calendar1" runat="server"
OnDayRender="calendar1_DayRender"></asp:Calendar>

        <asp:Label ID="lblMessage" runat="server"></asp:Label>
    </form>
</body>
</html>
```

To display messages in the Calendar control, we need to handle the DayRender event in the code-behind file:

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

namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
```

```
        protected void calendar1_DayRender(object sender,
DayRenderEventArgs e)
        {
            // Display custom messages for specific dates
            if (e.Day.Date == new DateTime(2023, 7, 15))
            {
                e.Cell.BackColor = System.Drawing.Color.Yellow;
                e.Cell.ToolTip = "Special event!";
                e.Cell.Text = "Event day";
            }
            else if (e.Day.Date == new DateTime(2023, 7, 20))
            {
                e.Cell.BackColor = System.Drawing.Color.Cyan;
                e.Cell.ToolTip = "Another event";
                e.Cell.Text = "Important day";
            }
        }
    }
}
```

b) Display vacation in a calendar Control

Solution

Calendar control in ASP.NET Web Forms to display vacation dates:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Calendar Example</title>
    <style>
        .vacation {
            background-color: #ffd700;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
        <h1>Calendar Example</h1>

        <asp:Calendar ID="calendar1" runat="server"
OnDayRender="calendar1_DayRender"></asp:Calendar>
    </form>
</body>
</html>
```

To display vacation dates in the Calendar control, we need to handle the DayRender event in the code-behind file:

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

namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
```

```
        }

        protected void calendar1_DayRender(object sender,
DayRenderEventArgs e)
        {
            // Display vacation dates
            if (e.Day.Date == new DateTime(2023, 7, 15) || e.Day.Date == new
DateTime(2023, 7, 16) || e.Day.Date == new DateTime(2023, 7, 17))
            {
                e.Cell.CssClass = "vacation";
                e.Cell.ToolTip = "Vacation";
            }
        }
    }
}
```

c) Selected day in a calendar control using style

Solution

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Calendar Example</title>
    <style>
```

```
        .selectedDay {
            background-color: #ff0000;
            color: #ffffff;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
        <h1>Calendar Example</h1>

        <asp:Calendar ID="calendar1" runat="server"
OnDayRender="calendar1_DayRender"
OnSelectionChanged="calendar1_SelectionChanged"></asp:Calendar>
    </form>
</body>
</html>
```

To highlight the selected day with a custom style, we need to handle the DayRender and SelectionChanged events in the code-behind file:

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

namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
```

```
        protected void calendar1_DayRender(object sender,
DayRenderEventArgs e)
        {
            // Highlight the selected day
            if (e.Day.Date == calendar1.SelectedDate)
            {
                e.Cell.CssClass = "selectedDay";
            }
        }

        protected void calendar1_SelectionChanged(object sender, EventArgs
e)
        {
            // Refresh the calendar to reflect the selection
            calendar1.DataBind();
        }
    }
}
```

d) Difference between two calendar Dates

Solution

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

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

```
<head runat="server">
   <title>Calendar Example</title>
</head>
<body>
   <form id="form1" runat="server">
      <h1>Calendar Example</h1>

      <h2>First Date:</h2>
      <asp:Calendar ID="calendar1" runat="server"></asp:Calendar>

      <h2>Second Date:</h2>
      <asp:Calendar ID="calendar2" runat="server"></asp:Calendar>

      <asp:Button ID="btnCalculate" runat="server" Text="Calculate"
OnClick="btnCalculate_Click" />

      <h2>Result:</h2>
      <asp:Label ID="lblResult" runat="server"></asp:Label>
   </form>
</body>
</html>
```

To calculate the difference between the selected dates, we handle the Click event of the Calculate button in the code-behind file:

```
using System;

namespace WebApplication1
{
   public partial class WebForm1 : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
```

```
        {
        }

        protected void btnCalculate_Click(object sender, EventArgs e)
        {
            DateTime firstDate = calendar1.SelectedDate;
            DateTime secondDate = calendar2.SelectedDate;

            if (firstDate != DateTime.MinValue && secondDate !=
DateTime.MinValue)
            {
                TimeSpan difference = secondDate - firstDate;
                int daysDifference = difference.Days;

                lblResult.Text = "The difference in days is: " + daysDifference;
            }
            else
            {
                lblResult.Text = "Please select both dates.";
            }
        }
    }
}
```

c. Demonstrate the use of Treeview control perform following operations.

25

a) Treeview control and datalist

Solution

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>TreeView Example</title>
    <style>
        .selectedNode {
            font-weight: bold;
            color: blue;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
        <h1>TreeView Example</h1>

        <asp:TreeView ID="treeView1" runat="server"
OnSelectedNodeChanged="treeView1_SelectedNodeChanged"
NodeStyle-CssClass="selectedNode">
            <Nodes>
                <asp:TreeNode Text="Fruits" Value="Fruits">
                    <asp:TreeNode Text="Apple" Value="Apple"></asp:TreeNode>
                    <asp:TreeNode Text="Banana"
Value="Banana"></asp:TreeNode>
                    <asp:TreeNode Text="Orange"
Value="Orange"></asp:TreeNode>
                </asp:TreeNode>
                <asp:TreeNode Text="Vegetables" Value="Vegetables">
```

```
                    <asp:TreeNode Text="Carrot"
Value="Carrot"></asp:TreeNode>
                    <asp:TreeNode Text="Broccoli"
Value="Broccoli"></asp:TreeNode>
                    <asp:TreeNode Text="Tomato"
Value="Tomato"></asp:TreeNode>
                </asp:TreeNode>
            </Nodes>
        </asp:TreeView>

        <h2>Selected Item:</h2>
        <asp:Label ID="lblSelectedItem" runat="server"></asp:Label>

        <h2>DataList:</h2>
        <asp:DataList ID="dataList1" runat="server" RepeatColumns="2"
RepeatDirection="Horizontal">
            <ItemTemplate>
                <asp:Label ID="lblItem" runat="server" Text='<%#
Container.DataItem %>'></asp:Label>
            </ItemTemplate>
        </asp:DataList>
    </form>
</body>
</html>
```

To handle the selection of a node in the TreeView control and display the selected item in the DataList control, we need to handle the SelectedNodeChanged event in the code-behind file:

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

```
namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }

        protected void treeView1_SelectedNodeChanged(object sender,
EventArgs e)
        {
            TreeNode selectedNode = treeView1.SelectedNode;
            lblSelectedItem.Text = "Selected Item: " + selectedNode.Text;

            switch (selectedNode.Value)
            {
                case "Fruits":
                    dataList1.DataSource = new string[] { "Apple", "Banana",
"Orange" };
                    break;
                case "Vegetables":
                    dataList1.DataSource = new string[] { "Carrot", "Broccoli",
"Tomato" };
                    break;
                default:
                    dataList1.DataSource = null;
                    break;
            }

            dataList1.DataBind();
        }
    }
```

```
}
```

b) Treeview operations

Solution

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>TreeView Example</title>
    <style>
        .selectedNode {
            font-weight: bold;
            color: blue;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
        <h1>TreeView Example</h1>

        <asp:TreeView ID="treeView1" runat="server"
OnTreeNodeExpanded="treeView1_TreeNodeExpanded"
NodeStyle-CssClass="selectedNode">
            <Nodes>
                <asp:TreeNode Text="Fruits" Value="Fruits">
```

```
                <asp:TreeNode Text="Apple" Value="Apple"></asp:TreeNode>
                <asp:TreeNode Text="Banana"
Value="Banana"></asp:TreeNode>
                <asp:TreeNode Text="Orange"
Value="Orange"></asp:TreeNode>
              </asp:TreeNode>
              <asp:TreeNode Text="Vegetables" Value="Vegetables">
                <asp:TreeNode Text="Carrot"
Value="Carrot"></asp:TreeNode>
                <asp:TreeNode Text="Broccoli"
Value="Broccoli"></asp:TreeNode>
                <asp:TreeNode Text="Tomato"
Value="Tomato"></asp:TreeNode>
              </asp:TreeNode>
            </Nodes>
        </asp:TreeView>

        <h2>Operations:</h2>
        <asp:Button ID="btnExpandAll" runat="server" Text="Expand All"
OnClick="btnExpandAll_Click" />
        <asp:Button ID="btnCollapseAll" runat="server" Text="Collapse All"
OnClick="btnCollapseAll_Click" />
        <asp:Button ID="btnAddNode" runat="server" Text="Add Node"
OnClick="btnAddNode_Click" />
        <asp:Button ID="btnRemoveNode" runat="server" Text="Remove
Node" OnClick="btnRemoveNode_Click" />
    </form>
</body>
</html>
```

To handle the operations, we need to add corresponding event handlers in the code-behind file:

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

namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }

        protected void btnExpandAll_Click(object sender, EventArgs e)
        {
            treeView1.ExpandAll();
        }

        protected void btnCollapseAll_Click(object sender, EventArgs e)
        {
            treeView1.CollapseAll();
        }

        protected void btnAddNode_Click(object sender, EventArgs e)
        {
            TreeNode selectedNode = treeView1.SelectedNode;

            if (selectedNode != null)
            {
                TreeNode newNode = new TreeNode("New Node", "New Node");
                selectedNode.ChildNodes.Add(newNode);
            }
        }
```

```
        protected void btnRemoveNode_Click(object sender, EventArgs e)
        {
            TreeNode selectedNode = treeView1.SelectedNode;

            if (selectedNode != null && selectedNode.Parent != null)
            {
                selectedNode.Parent.ChildNodes.Remove(selectedNode);
            }
        }

        protected void treeView1_TreeNodeExpanded(object sender,
TreeNodeEventArgs e)
        {
            // Add additional child nodes when a parent node is expanded
            if (e.Node.Value == "Fruits")
            {
                TreeNode childNode = new TreeNode("Grapes", "Grapes");
                e.Node.ChildNodes.Add(childNode);
            }
        }
    }
}
```

4. Working with Form Controls

a. Create a Registration form to demonstrate use of various Validation controls.

Solution

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="RegistrationForm.aspx.cs"
Inherits="WebApplication1.RegistrationForm" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Registration Form</title>
    <style>
        .error {
            color: red;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
        <h1>Registration Form</h1>

        <div>
            <label for="txtName">Name:</label>
            <asp:TextBox ID="txtName" runat="server"></asp:TextBox>
            <asp:RequiredFieldValidator ID="rfvName" runat="server"
ControlToValidate="txtName" ErrorMessage="Name is required."
CssClass="error"></asp:RequiredFieldValidator>
        </div>
```

```
        <div>
            <label for="txtEmail">Email:</label>
            <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>
            <asp:RequiredFieldValidator ID="rfvEmail" runat="server"
ControlToValidate="txtEmail" ErrorMessage="Email is required."
CssClass="error"></asp:RequiredFieldValidator>
            <asp:RegularExpressionValidator ID="revEmail" runat="server"
ControlToValidate="txtEmail" ErrorMessage="Invalid email format."
ValidationExpression="^\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*$"
                CssClass="error"></asp:RegularExpressionValidator>
        </div>

        <div>
            <label for="txtPassword">Password:</label>
            <asp:TextBox ID="txtPassword" runat="server"
TextMode="Password"></asp:TextBox>
            <asp:RequiredFieldValidator ID="rfvPassword" runat="server"
ControlToValidate="txtPassword" ErrorMessage="Password is required."
CssClass="error"></asp:RequiredFieldValidator>
        </div>

        <div>
            <label for="txtConfirmPassword">Confirm Password:</label>
            <asp:TextBox ID="txtConfirmPassword" runat="server"
TextMode="Password"></asp:TextBox>
            <asp:RequiredFieldValidator ID="rfvConfirmPassword"
runat="server" ControlToValidate="txtConfirmPassword"
ErrorMessage="Confirm Password is required."
CssClass="error"></asp:RequiredFieldValidator>
```

```
            <asp:CompareValidator ID="cvConfirmPassword" runat="server"
ControlToValidate="txtConfirmPassword"
ControlToCompare="txtPassword"
                ErrorMessage="Passwords do not match."
CssClass="error"></asp:CompareValidator>
        </div>

        <div>
            <label for="txtPhone">Phone:</label>
            <asp:TextBox ID="txtPhone" runat="server"></asp:TextBox>
            <asp:RegularExpressionValidator ID="revPhone" runat="server"
ControlToValidate="txtPhone" ErrorMessage="Invalid phone number
format." ValidationExpression="^\d{10}$"
                CssClass="error"></asp:RegularExpressionValidator>
        </div>

        <div>
            <asp:Button ID="btnSubmit" runat="server" Text="Submit"
OnClick="btnSubmit_Click" />
        </div>
    </form>
</body>
</html>
```

To handle the form submission, we need to handle the Click event of the Submit button in the code-behind file:

```
using System;

namespace WebApplication1
{
    public partial class RegistrationForm : System.Web.UI.Page
```

```
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }

        protected void btnSubmit_Click(object sender, EventArgs e)
        {
            if (Page.IsValid)
            {
                // Perform registration logic
                string name = txtName.Text;
                string email = txtEmail.Text;
                string password = txtPassword.Text;
                string phone = txtPhone.Text;

                // Add code to store the registration data or perform any other
desired operations
                // ...
            }
        }
    }
}
```

b. Create Web Form to demonstrate use of Adrotator Control.

Solution

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>
```

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title>AdRotator Example</title>
</head>
<body>
   <form id="form1" runat="server">
      <h1>AdRotator Example</h1>

      <asp:AdRotator ID="adRotator1" runat="server"
AdvertisementFile="~/AdRotatorFile.xml" />

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

To populate the AdRotator control with advertisements, you need to create an XML file (AdRotatorFile.xml) in the project and define the advertisements in the following format:

```
<Advertisements>
   <Ad>
      <ImageUrl>images/ad1.jpg</ImageUrl>
      <NavigateUrl>https://example.com/ad1</NavigateUrl>
      <AlternateText>Advertisement 1</AlternateText>
   </Ad>
   <Ad>
      <ImageUrl>images/ad2.jpg</ImageUrl>
      <NavigateUrl>https://example.com/ad2</NavigateUrl>
      <AlternateText>Advertisement 2</AlternateText>
   </Ad>
```

```
   <!-- Add more <Ad> elements for additional advertisements -->
</Advertisements>
```

c. Create Web Form to demonstrate use User Controls

Solution

Create a new user control file (UserControl.ascx) with the following content:

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeBehind="UserControl.ascx.cs"
Inherits="WebApplication1.UserControl" %>

<asp:Label ID="lblMessage" runat="server" Text=""></asp:Label>
<br />
<asp:Button ID="btnClickMe" runat="server" Text="Click Me"
OnClick="btnClickMe_Click" />
```

Create a code-behind file (UserControl.ascx.cs) for the user control with the following content:

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

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

```
        protected void btnClickMe_Click(object sender, EventArgs e)
        {
            lblMessage.Text = "Button Clicked!";
        }
    }
}
```

Create a web form file (WebForm1.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>User Control Example</title>
</head>
<body>
    <form id="form1" runat="server">
        <h1>User Control Example</h1>

        <uc1:UserControl ID="userControl1" runat="server" />

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

5. Working with Navigation, Beautification and Master page.

a. Create Web Form to demonstrate use of Website Navigation controls and Site Map.

Solution

Create a new web form file (WebForm1.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title>Website Navigation Example</title>
</head>
<body>
   <form id="form1" runat="server">
      <h1>Website Navigation Example</h1>

      <asp:Menu ID="menu1" runat="server"
DataSourceID="siteMapDataSource" Orientation="Horizontal">
      </asp:Menu>

      <asp:SiteMapDataSource ID="siteMapDataSource"
runat="server" ShowStartingNode="false" />

   </form>
</body>
```

```
</html>
```

In the code-behind file (WebForm1.aspx.cs), add the following using statements:

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

In the code-behind file, add the following code inside the Page_Load event handler:

```
protected void Page_Load(object sender, EventArgs e)
{
   if (!IsPostBack)
   {
      menu1.DataBind();
   }
}
```

Open the Web.sitemap file (if it doesn't exist, you can create a new one) and define the site map structure with menu items. Here's an example:

```
<siteMap
xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
   <siteMapNode url="~/Home.aspx" title="Home" description="Home
Page">
      <siteMapNode url="~/About.aspx" title="About"
description="About Page" />
      <siteMapNode url="~/Contact.aspx" title="Contact"
description="Contact Page" />
   </siteMapNode>
</siteMap>
```

b. Create a web application to demonstrate use of Master Page with applying Styles and Themes for page beautification.

Solution

Create a new master page file (Site.Master) with the following content:

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeBehind="Site.master.cs" Inherits="WebApplication1.SiteMaster"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Master Page Example</title>
    <link href="Styles/Site.css" rel="stylesheet" />
    <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form runat="server">
        <header>
            <h1>My Website</h1>
            <nav>
                <ul>
                    <li><a href="Home.aspx">Home</a></li>
                    <li><a href="About.aspx">About</a></li>
                    <li><a href="Contact.aspx">Contact</a></li>
```

```
                </ul>
            </nav>
        </header>
        <div id="mainContent" class="container">
            <asp:ContentPlaceHolder ID="ContentPlaceHolder1"
runat="server">
            </asp:ContentPlaceHolder>
        </div>
        <footer>
            &copy; 2023 My Website
        </footer>
    </form>
</body>
</html>
```

Create a CSS file (Site.css) in a folder named "Styles" and define the styles for the master page. For example:

```
.container {
    max-width: 960px;
    margin: 0 auto;
    padding: 20px;
}

header {
    background-color: #333;
    color: #fff;
    padding: 10px;
}

nav ul {
    list-style: none;
```

```
    padding: 0;
}

nav ul li {
    display: inline-block;
    margin-right: 10px;
}

nav ul li a {
    color: #fff;
    text-decoration: none;
}

footer {
    background-color: #333;
    color: #fff;
    padding: 10px;
    text-align: center;
}
```

Create a new web form file (Home.aspx) and set the master page to Site.Master. Here's an example:

```
<%@ Page Title="Home" Language="C#"
MasterPageFile="~/Site.Master" AutoEventWireup="true"
CodeBehind="Home.aspx.cs" Inherits="WebApplication1.Home" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head"
runat="server">
    <link href="Styles/Home.css" rel="stylesheet" />
</asp:Content>
```

```
<asp:Content ID="Content2"
ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <h2>Welcome to My Website!</h2>
    <p>This is the home page content.</p>
</asp:Content>
```

Create a CSS file (Home.css) in a folder named "Styles" and define the styles specific to the home page. For example:

```
h2 {
    color: #0099ff;
}

p {
    font-size: 16px;
}
```

c. Create a web application to demonstrate various states of ASP.NET Pages.

Solution

Create a new web form file (WebForm1.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

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

```
<head runat="server">
   <title>Page States Example</title>
</head>
<body>
   <form id="form1" runat="server">
      <h1>Page States Example</h1>

      <asp:Button ID="btnViewState" runat="server" Text="Toggle
ViewState" OnClick="btnViewState_Click" />
      <asp:Button ID="btnPostback" runat="server" Text="Perform
Postback" OnClick="btnPostback_Click" />

      <br /><br />

      <asp:Label ID="lblViewState" runat="server"></asp:Label>
      <br />
      <asp:Label ID="lblPostback" runat="server"></asp:Label>
   </form>
</body>
</html>
```

In the code-behind file (WebForm1.aspx.cs), add the following code:

```
using System;

namespace WebApplication1
{
   public partial class WebForm1 : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {
         if (IsPostBack)
```

```
            {
                lblViewState.Text = "ViewState is enabled.";
            }
            else
            {
                lblViewState.Text = "ViewState is disabled.";
            }

            lblPostback.Text = "Last Postback: " + DateTime.Now.ToString();
        }

        protected void btnViewState_Click(object sender, EventArgs e)
        {
            ViewStateMode = ViewStateMode ==
System.Web.UI.ViewStateMode.Enabled ?
System.Web.UI.ViewStateMode.Disabled :
System.Web.UI.ViewStateMode.Enabled;
            Response.Redirect(Request.RawUrl);
        }

        protected void btnPostback_Click(object sender, EventArgs e)
        {
            // No specific action required, just triggers a postback
        }
    }
}
```

6. Working with Database

a. Create a web application bind data in a multiline textbox by querying in another textbox.

Solution

Create a new web form file (WebForm1.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Data Binding Example</title>
</head>
<body>
    <form id="form1" runat="server">
        <h1>Data Binding Example</h1>

        <label for="txtQuery">Enter Query:</label>
        <asp:TextBox ID="txtQuery" runat="server"></asp:TextBox>
        <asp:Button ID="btnSearch" runat="server" Text="Search"
OnClick="btnSearch_Click" />

        <br /><br />

        <asp:TextBox ID="txtResults" runat="server" TextMode="MultiLine"
Rows="10"></asp:TextBox>
    </form>
```

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

In the code-behind file (WebForm1.aspx.cs), add the following code:

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

namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }

        protected void btnSearch_Click(object sender, EventArgs e)
        {
            string connectionString = 
WebConfigurationManager.ConnectionStrings["YourConnectionString"].ConnectionString;
            string query = txtQuery.Text;

            using (SqlConnection connection = new 
SqlConnection(connectionString))
            {
                connection.Open();

                SqlCommand command = new SqlCommand(query, connection);
                SqlDataReader reader = command.ExecuteReader();
```

```
                txtResults.Text = string.Empty;

                while (reader.Read())
                {
                    for (int i = 0; i < reader.FieldCount; i++)
                    {
                        txtResults.Text += reader[i].ToString() + "\t";
                    }

                    txtResults.Text += Environment.NewLine;
                }

                reader.Close();
            }
        }
    }
}
```

b. Create a web application to display records by using database.

Solution

Create a new web form file (WebForm1.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Record Display Example</title>
</head>
<body>
    <form id="form1" runat="server">
        <h1>Record Display Example</h1>

        <asp:GridView ID="gridView" runat="server"
AutoGenerateColumns="False">
            <Columns>
                <asp:BoundField DataField="Id" HeaderText="ID" />
                <asp:BoundField DataField="Name" HeaderText="Name" />
                <asp:BoundField DataField="Age" HeaderText="Age" />
            </Columns>
        </asp:GridView>

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

In the code-behind file (WebForm1.aspx.cs), add the following code:

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

namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
```

```
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                BindGrid();
            }
        }

        private void BindGrid()
        {
            string connectionString =
ConfigurationManager.ConnectionStrings["YourConnectionString"].Connect
ionString;
            string query = "SELECT Id, Name, Age FROM YourTable";

            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                SqlCommand command = new SqlCommand(query, connection);
                SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
                DataTable dataTable = new DataTable();

                dataAdapter.Fill(dataTable);

                gridView.DataSource = dataTable;
                gridView.DataBind();
            }
        }
    }
}
```

c. Demonstrate the use of Datalist link control.

Solution

Create a new web form file (WebForm1.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title>DataList Example</title>
</head>
<body>
   <form id="form1" runat="server">
      <h1>DataList Example</h1>

      <asp:DataList ID="dataList" runat="server" RepeatColumns="3"
RepeatDirection="Vertical">
         <ItemTemplate>
            <asp:HyperLink ID="hyperLink" runat="server" Text='<%#
Eval("Name") %>' NavigateUrl='<%# Eval("Url") %>' />
         </ItemTemplate>
      </asp:DataList>

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

In the code-behind file (WebForm1.aspx.cs), add the following code:

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

namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                BindData();
            }
        }

        private void BindData()
        {
            List<LinkItem> links = new List<LinkItem>()
            {
                new LinkItem("Google", "https://www.google.com"),
                new LinkItem("Microsoft", "https://www.microsoft.com"),
                new LinkItem("Amazon", "https://www.amazon.com"),
                new LinkItem("OpenAI", "https://www.openai.com")
            };

            dataList.DataSource = links;
            dataList.DataBind();
        }

        public class LinkItem
```

```
        {
            public string Name { get; set; }
            public string Url { get; set; }

            public LinkItem(string name, string url)
            {
                Name = name;
                Url = url;
            }
        }
    }
}
```

7. Working with Database

a. Create a web application to display Databinding using dropdownlist control.

Solution

Create a new web form file (WebForm1.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title>Data Binding Example</title>
</head>
<body>
   <form id="form1" runat="server">
      <h1>Data Binding Example</h1>

      <asp:DropDownList ID="ddlCities" runat="server"
AutoPostBack="true"
OnSelectedIndexChanged="ddlCities_SelectedIndexChanged">
      </asp:DropDownList>

      <br /><br />

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

   </form>
```

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

In the code-behind file (WebForm1.aspx.cs), add the following code:

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

namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                BindCities();
            }
        }

        private void BindCities()
        {
            string connectionString =
ConfigurationManager.ConnectionStrings["YourConnectionString"].Connect
ionString;
            string query = "SELECT Id, CityName FROM Cities";

            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
```

```
            SqlCommand command = new SqlCommand(query, connection);
            SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
            DataTable dataTable = new DataTable();

            dataAdapter.Fill(dataTable);

            ddlCities.DataSource = dataTable;
            ddlCities.DataTextField = "CityName";
            ddlCities.DataValueField = "Id";
            ddlCities.DataBind();
        }
    }

    protected void ddlCities_SelectedIndexChanged(object sender,
EventArgs e)
    {
        lblSelectedCity.Text = "Selected City: " +
ddlCities.SelectedItem.Text;
    }
  }
}
```

b. Create a web application for to display the phone no of an author using database.

Solution

Create a new web form file (WebForm1.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title>Author Phone Number Example</title>
</head>
<body>
   <form id="form1" runat="server">
      <h1>Author Phone Number Example</h1>

      <asp:DropDownList ID="ddlAuthors" runat="server"
AutoPostBack="true"
OnSelectedIndexChanged="ddlAuthors_SelectedIndexChanged">
      </asp:DropDownList>

      <br /><br />

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

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

In the code-behind file (WebForm1.aspx.cs), add the following code:

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

```
using System.Data.SqlClient;

namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                BindAuthors();
            }
        }

        private void BindAuthors()
        {
            string connectionString =
ConfigurationManager.ConnectionStrings["YourConnectionString"].Connect
ionString;
            string query = "SELECT Id, AuthorName FROM Authors";

            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                SqlCommand command = new SqlCommand(query, connection);
                SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
                DataTable dataTable = new DataTable();

                dataAdapter.Fill(dataTable);

                ddlAuthors.DataSource = dataTable;
                ddlAuthors.DataTextField = "AuthorName";
```

```
            ddlAuthors.DataValueField = "Id";
            ddlAuthors.DataBind();
        }
    }

    protected void ddlAuthors_SelectedIndexChanged(object sender,
EventArgs e)
    {
        int selectedAuthorId = Convert.ToInt32(ddlAuthors.SelectedValue);
        string connectionString =
ConfigurationManager.ConnectionStrings["YourConnectionString"].Connect
ionString;
        string query = "SELECT PhoneNumber FROM Authors WHERE Id
= @AuthorId";

        using (SqlConnection connection = new
SqlConnection(connectionString))
        {
            SqlCommand command = new SqlCommand(query, connection);
            command.Parameters.AddWithValue("@AuthorId",
selectedAuthorId);

            connection.Open();
            object result = command.ExecuteScalar();

            if (result != null)
            {
                lblPhoneNumber.Text = "Phone Number: " + result.ToString();
            }
            else
            {
                lblPhoneNumber.Text = "Phone Number not found.";
```

```
                }
            }
        }
    }
}
```

c. Create a web application for inserting and deleting record from a database. (Using Execute-Non Query).

Solution

Create a new web form file (WebForm1.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Insert/Delete Record Example</title>
</head>
<body>
    <form id="form1" runat="server">
        <h1>Insert/Delete Record Example</h1>

        <h3>Insert Record</h3>
        <label for="txtName">Name:</label>
        <asp:TextBox ID="txtName" runat="server"></asp:TextBox>
```

```
        <label for="txtEmail">Email:</label>
        <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>
        <asp:Button ID="btnInsert" runat="server" Text="Insert"
OnClick="btnInsert_Click" />

        <br /><br />

        <h3>Delete Record</h3>
        <label for="ddlRecords">Select Record:</label>
        <asp:DropDownList ID="ddlRecords" runat="server"
AutoPostBack="true"
OnSelectedIndexChanged="ddlRecords_SelectedIndexChanged">
        </asp:DropDownList>
        <asp:Button ID="btnDelete" runat="server" Text="Delete"
OnClick="btnDelete_Click" />

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

In the code-behind file (WebForm1.aspx.cs), add the following code:

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

namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
```

```
    {
        if (!IsPostBack)
        {
            BindRecords();
        }
    }

    private void BindRecords()
    {
        string connectionString =
ConfigurationManager.ConnectionStrings["YourConnectionString"].Connect
ionString;
        string query = "SELECT Id, Name FROM Records";

        using (SqlConnection connection = new
SqlConnection(connectionString))
        {
            SqlCommand command = new SqlCommand(query, connection);
            SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
            DataTable dataTable = new DataTable();

            dataAdapter.Fill(dataTable);

            ddlRecords.DataSource = dataTable;
            ddlRecords.DataTextField = "Name";
            ddlRecords.DataValueField = "Id";
            ddlRecords.DataBind();
        }
    }

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

```
            string name = txtName.Text;
            string email = txtEmail.Text;

            string connectionString =
ConfigurationManager.ConnectionStrings["YourConnectionString"].Connect
ionString;
            string query = "INSERT INTO Records (Name, Email) VALUES
(@Name, @Email)";

            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                SqlCommand command = new SqlCommand(query, connection);
                command.Parameters.AddWithValue("@Name", name);
                command.Parameters.AddWithValue("@Email", email);

                connection.Open();
                int rowsAffected = command.ExecuteNonQuery();

                if (rowsAffected > 0)
                {
                    BindRecords(); // Refresh the DropDownList after successful
insertion
                    ClearInsertForm(); // Clear the insert form fields
                }
            }
        }

        protected void btnDelete_Click(object sender, EventArgs e)
        {
            int selectedRecordId = Convert.ToInt32(ddlRecords.SelectedValue);
```

```
        string connectionString =
ConfigurationManager.ConnectionStrings["YourConnectionString"].Connect
ionString;
        string query = "DELETE FROM Records WHERE Id = @RecordId";

        using (SqlConnection connection = new
SqlConnection(connectionString))
        {
            SqlCommand command = new SqlCommand(query, connection);
            command.Parameters.AddWithValue("@RecordId",
selectedRecordId);

            connection.Open();
            int rowsAffected = command.ExecuteNonQuery();

            if (rowsAffected > 0)
            {
                BindRecords(); // Refresh the DropDownList after successful
deletion
            }
        }
    }

    protected void ddlRecords_SelectedIndexChanged(object sender,
EventArgs e)
    {
        // No specific action required when the selected item changes in the
DropDownList
    }

    private void ClearInsertForm()
    {
```

```
            txtName.Text = string.Empty;
            txtEmail.Text = string.Empty;
        }
    }
}
```

8. Working with data controls

a. Create a web application to demonstrate various uses and properties of SqlDataSource.

Solution

Create a new web form file (WebForm1.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>SqlDataSource Example</title>
</head>
<body>
    <form id="form1" runat="server">
        <h1>SqlDataSource Example</h1>

        <h3>Select Query</h3>
        <asp:DropDownList ID="ddlCountries" runat="server"
DataSourceID="sqlDataSourceCountries" DataTextField="CountryName"
DataValueField="CountryId"></asp:DropDownList>
        <asp:SqlDataSource ID="sqlDataSourceCountries" runat="server"
ConnectionString="<%$ ConnectionStrings:YourConnectionString %>"
SelectCommand="SELECT CountryId, CountryName FROM
Countries"></asp:SqlDataSource>

        <br /><br />
```

```
        <h3>Insert Query</h3>
        <asp:TextBox ID="txtCountryName" runat="server"></asp:TextBox>
        <asp:Button ID="btnInsert" runat="server" Text="Insert"
OnClick="btnInsert_Click" />
        <asp:SqlDataSource ID="sqlDataSourceInsert" runat="server"
ConnectionString="<%$ ConnectionStrings:YourConnectionString %>"
InsertCommand="INSERT INTO Countries (CountryName) VALUES
(@CountryName)">
            <InsertParameters>
                <asp:Parameter Name="CountryName" Type="String" />
            </InsertParameters>
        </asp:SqlDataSource>

        <br /><br />

        <h3>Delete Query</h3>
        <asp:GridView ID="gridViewCountries" runat="server"
DataSourceID="sqlDataSourceGrid" AutoGenerateColumns="False">
            <Columns>
                <asp:BoundField DataField="CountryName"
HeaderText="Country Name" />
                <asp:ButtonField ButtonType="Button" CommandName="Delete"
Text="Delete" />
            </Columns>
        </asp:GridView>
        <asp:SqlDataSource ID="sqlDataSourceGrid" runat="server"
ConnectionString="<%$ ConnectionStrings:YourConnectionString %>"
SelectCommand="SELECT CountryId, CountryName FROM Countries"
DeleteCommand="DELETE FROM Countries WHERE CountryId =
@CountryId">
            <DeleteParameters>
```

```
            <asp:Parameter Name="CountryId" Type="Int32" />
        </DeleteParameters>
    </asp:SqlDataSource>

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

In the code-behind file (WebForm1.aspx.cs), add the following code:

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

namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                BindGridView();
            }
        }

        protected void btnInsert_Click(object sender, EventArgs e)
        {
            string countryName = txtCountryName.Text;
```

```
sqlDataSourceInsert.InsertParameters["CountryName"].DefaultValue = 
countryName;
            sqlDataSourceInsert.Insert();

            txtCountryName.Text = string.Empty;

            BindGridView();
        }

        protected void gridViewCountries_RowDeleting(object sender, 
System.Web.UI.WebControls.GridViewDeleteEventArgs e)
        {
            int countryId = 
Convert.ToInt32(gridViewCountries.DataKeys[e.RowIndex].Value);

            sqlDataSourceGrid.DeleteParameters["CountryId"].DefaultValue = 
countryId.ToString();
            sqlDataSourceGrid.Delete();

            BindGridView();
        }

        private void BindGridView()
        {
            gridViewCountries.DataBind();
        }
    }
}
```

b. Create a web application to demonstrate data binding using DetailsView and FormView Control.

Solution

Create a new web form file (WebForm1.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Data Binding Example</title>
</head>
<body>
    <form id="form1" runat="server">
        <h1>Data Binding Example</h1>

        <h3>DetailsView Control</h3>
        <asp:DetailsView ID="detailsView" runat="server"
AutoGenerateRows="false" DataSourceID="sqlDataSourceDetails">
            <Fields>
                <asp:BoundField DataField="Id" HeaderText="ID"
ReadOnly="true" />
                <asp:BoundField DataField="Name" HeaderText="Name" />
                <asp:BoundField DataField="Email" HeaderText="Email" />
                <asp:CommandField ShowEditButton="true" />
            </Fields>
```

```
        </asp:DetailsView>
        <asp:SqlDataSource ID="sqlDataSourceDetails" runat="server"
ConnectionString="<%$ ConnectionStrings:YourConnectionString %>"
SelectCommand="SELECT Id, Name, Email FROM Users"
UpdateCommand="UPDATE Users SET Name = @Name, Email = @Email
WHERE Id = @Id">
            <UpdateParameters>
                <asp:Parameter Name="Name" Type="String" />
                <asp:Parameter Name="Email" Type="String" />
                <asp:Parameter Name="Id" Type="Int32" />
            </UpdateParameters>
        </asp:SqlDataSource>

        <br /><br />

        <h3>FormView Control</h3>
        <asp:FormView ID="formView" runat="server"
AutoGenerateRows="false" DataSourceID="sqlDataSourceForm">
            <EditItemTemplate>
                <table>
                    <tr>
                        <td><asp:Label ID="lblId" runat="server" Text='<%#
Eval("Id") %>' /></td>
                    </tr>
                    <tr>
                        <td><asp:TextBox ID="txtName" runat="server" Text='<%#
Bind("Name") %>' /></td>
                    </tr>
                    <tr>
                        <td><asp:TextBox ID="txtEmail" runat="server" Text='<%#
Bind("Email") %>' /></td>
                    </tr>
```

```
                <tr>
                    <td><asp:Button ID="btnUpdate" runat="server"
Text="Update" CommandName="Update" /></td>
                </tr>
            </table>
        </EditItemTemplate>
        <ItemTemplate>
            <table>
                <tr>
                    <td><asp:Label ID="lblId" runat="server" Text='<%#
Eval("Id") %>' /></td>
                </tr>
                <tr>
                    <td><asp:Label ID="lblName" runat="server" Text='<%#
Eval("Name") %>' /></td>
                </tr>
                <tr>
                    <td><asp:Label ID="lblEmail" runat="server" Text='<%#
Eval("Email") %>' /></td>
                </tr>
                <tr>
                    <td><asp:Button ID="btnEdit" runat="server" Text="Edit"
CommandName="Edit" /></td>
                </tr>
            </table>
        </ItemTemplate>
    </asp:FormView>
    <asp:SqlDataSource ID="sqlDataSourceForm" runat="server"
ConnectionString="<%$ ConnectionStrings:YourConnectionString %>"
SelectCommand="SELECT Id, Name, Email FROM Users"
UpdateCommand="UPDATE Users SET Name = @Name, Email = @Email
WHERE Id = @Id">
```

```
        <UpdateParameters>
            <asp:Parameter Name="Name" Type="String" />
            <asp:Parameter Name="Email" Type="String" />
            <asp:Parameter Name="Id" Type="Int32" />
        </UpdateParameters>
    </asp:SqlDataSource>

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

In the code-behind file (WebForm1.aspx.cs), add the following code:

```
using System;
using System.Configuration;

namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
    }
}
```

c. Create a web application to display Using Disconnected Data Access and Databinding using GridView.

Solution

Create a new web form file (WebForm1.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title>Disconnected Data Access and Data Binding Example</title>
</head>
<body>
   <form id="form1" runat="server">
      <h1>Disconnected Data Access and Data Binding Example</h1>

      <asp:GridView ID="gridViewEmployees" runat="server"
AutoGenerateColumns="true"></asp:GridView>

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

In the code-behind file (WebForm1.aspx.cs), add the following code:

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

```
namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                BindGridView();
            }
        }

        private void BindGridView()
        {
            string connectionString =
ConfigurationManager.ConnectionStrings["YourConnectionString"].Connect
ionString;
            string query = "SELECT Id, FirstName, LastName, Email FROM
Employees";

            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                SqlDataAdapter dataAdapter = new SqlDataAdapter(query,
connection);
                DataSet dataSet = new DataSet();

                dataAdapter.Fill(dataSet, "Employees");

                gridViewEmployees.DataSource = dataSet.Tables["Employees"];
                gridViewEmployees.DataBind();
```

```
            }
        }
    }
}
```

9. Working with GridView control

a. Create a web application to demonstrate use of GridView control template and GridView Hyperlink.

Solution

Create a new web form file (WebForm1.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title>GridView Control Example</title>
</head>
<body>
   <form id="form1" runat="server">
      <h1>GridView Control Example</h1>

      <asp:GridView ID="gridViewBooks" runat="server"
AutoGenerateColumns="false">
         <Columns>
            <asp:BoundField DataField="BookId" HeaderText="Book ID" />
            <asp:TemplateField HeaderText="Book Title">
               <ItemTemplate>
                  <asp:Label ID="lblTitle" runat="server" Text='<%#
Eval("Title") %>'></asp:Label>
               </ItemTemplate>
```

```
                </asp:TemplateField>
                <asp:BoundField DataField="Author" HeaderText="Author" />
                <asp:HyperLinkField DataTextField="Publisher"
HeaderText="Publisher" DataNavigateUrlFields="PublisherId"
DataNavigateUrlFormatString="Publishers.aspx?PublisherId={0}"
Text="View Publisher" />
            </Columns>
        </asp:GridView>

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

In the code-behind file (WebForm1.aspx.cs), add the following code:

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

namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                BindGridView();
            }
        }
```

```
        private void BindGridView()
        {
            string connectionString =
ConfigurationManager.ConnectionStrings["YourConnectionString"].Connect
ionString;
            string query = "SELECT BookId, Title, Author, Publisher, PublisherId
FROM Books";

            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                SqlDataAdapter dataAdapter = new SqlDataAdapter(query,
connection);
                DataTable dataTable = new DataTable();

                dataAdapter.Fill(dataTable);

                gridViewBooks.DataSource = dataTable;
                gridViewBooks.DataBind();
            }
        }
    }
}
```

b. Create a web application to demonstrate use of GridView button column and GridView Events.

Solution

Create a new web form file (WebForm1.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>GridView Button Column Example</title>
</head>
<body>
  <form id="form1" runat="server">
    <h1>GridView Button Column Example</h1>

    <asp:GridView ID="gridViewBooks" runat="server"
AutoGenerateColumns="false"
OnRowEditing="gridViewBooks_RowEditing"
OnRowUpdating="gridViewBooks_RowUpdating"
OnRowCancelingEdit="gridViewBooks_RowCancelingEdit"
OnRowDeleting="gridViewBooks_RowDeleting">
      <Columns>
        <asp:BoundField DataField="BookId" HeaderText="Book ID"
ReadOnly="true" />
        <asp:BoundField DataField="Title" HeaderText="Title"
ReadOnly="true" />
        <asp:BoundField DataField="Author" HeaderText="Author"
ReadOnly="true" />
        <asp:TemplateField HeaderText="Action">
          <ItemTemplate>
            <asp:Button ID="btnEdit" runat="server" Text="Edit"
CommandName="Edit" />
```

```
                        <asp:Button ID="btnDelete" runat="server" Text="Delete"
CommandName="Delete" OnClientClick="return confirm('Are you sure you
want to delete this record?');" />
                    </ItemTemplate>
                    <EditItemTemplate>
                        <asp:Button ID="btnUpdate" runat="server" Text="Update"
CommandName="Update" />
                        <asp:Button ID="btnCancel" runat="server" Text="Cancel"
CommandName="Cancel" />
                    </EditItemTemplate>
                </asp:TemplateField>
            </Columns>
        </asp:GridView>

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

In the code-behind file (WebForm1.aspx.cs), add the following code:

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

namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
```

```
        if (!IsPostBack)
        {
            BindGridView();
        }
    }

    private void BindGridView()
    {
        string connectionString =
ConfigurationManager.ConnectionStrings["YourConnectionString"].Connect
ionString;
        string query = "SELECT BookId, Title, Author FROM Books";

        using (SqlConnection connection = new
SqlConnection(connectionString))
        {
            SqlDataAdapter dataAdapter = new SqlDataAdapter(query,
connection);
            DataTable dataTable = new DataTable();

            dataAdapter.Fill(dataTable);

            gridViewBooks.DataSource = dataTable;
            gridViewBooks.DataBind();
        }
    }

    protected void gridViewBooks_RowEditing(object sender,
GridViewEditEventArgs e)
    {
        gridViewBooks.EditIndex = e.NewEditIndex;
        BindGridView();
```

```
        }

        protected void gridViewBooks_RowUpdating(object sender,
GridViewUpdateEventArgs e)
        {
            GridViewRow row = gridViewBooks.Rows[e.RowIndex];
            int bookId =
Convert.ToInt32(gridViewBooks.DataKeys[e.RowIndex].Value);
            string title = ((TextBox)row.FindControl("txtTitle")).Text;
            string author = ((TextBox)row.FindControl("txtAuthor")).Text;

            // Update the record in the database using the bookId, title, and
author values

            gridViewBooks.EditIndex = -1;
            BindGridView();
        }

        protected void gridViewBooks_RowCancelingEdit(object sender,
GridViewCancelEditEventArgs e)
        {
            gridViewBooks.EditIndex = -1;
            BindGridView();
        }

        protected void gridViewBooks_RowDeleting(object sender,
GridViewDeleteEventArgs e)
        {
            int bookId =
Convert.ToInt32(gridViewBooks.DataKeys[e.RowIndex].Value);

            // Delete the record from the database using the bookId
```

```
            BindGridView();
        }
    }
}
```

c. Create a web application to demonstrate GridView paging and Creating own table format using GridView.

Solution

Create a new web form file (WebForm1.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>GridView Paging and Custom Table Format Example</title>
</head>
<body>
    <form id="form1" runat="server">
        <h1>GridView Paging and Custom Table Format Example</h1>

        <asp:GridView ID="gridViewBooks" runat="server"
AutoGenerateColumns="false" AllowPaging="true" PageSize="5"
OnPageIndexChanging="gridViewBooks_PageIndexChanging">
```

```
        <Columns>
            <asp:BoundField DataField="BookId" HeaderText="Book ID" />
            <asp:BoundField DataField="Title" HeaderText="Title" />
            <asp:BoundField DataField="Author" HeaderText="Author" />
        </Columns>
        <PagerStyle CssClass="custom-pager" />
    </asp:GridView>

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

In the code-behind file (WebForm1.aspx.cs), add the following code:

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

namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                BindGridView();
            }
        }
```

```
        private void BindGridView()
        {
            string connectionString =
ConfigurationManager.ConnectionStrings["YourConnectionString"].Connect
ionString;
            string query = "SELECT BookId, Title, Author FROM Books";

            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                SqlDataAdapter dataAdapter = new SqlDataAdapter(query,
connection);
                DataTable dataTable = new DataTable();

                dataAdapter.Fill(dataTable);

                gridViewBooks.DataSource = dataTable;
                gridViewBooks.DataBind();
            }
        }

        protected void gridViewBooks_PageIndexChanging(object sender,
GridViewPageEventArgs e)
        {
            gridViewBooks.PageIndex = e.NewPageIndex;
            BindGridView();
        }
    }
}
```

Create a CSS file (styles.css) in the project and add the following CSS code:

```
.custom-pager {
    text-align: right;
    margin-top: 10px;
}

.custom-pager a {
    display: inline-block;
    padding: 5px;
    margin-right: 5px;
    text-decoration: none;
    color: #333;
    background-color: #f0f0f0;
    border: 1px solid #ccc;
}

.custom-pager a:hover {
    background-color: #ccc;
}
```

Include the CSS file in the WebForm1.aspx file by adding the following line inside the <head> tag:

```
<link rel="stylesheet" href="styles.css" />
```

10. Working with AJAX and XML

a. Create a web application to demonstrate reading and writing operation with XML.

Solution

Create a new web form file (WebForm1.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>AJAX and XML Example</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></scri
pt>
    <script>
        $(document).ready(function () {
            $("#btnLoad").click(function () {
                $.ajax({
                    type: "GET",
                    url: "Books.xml",
                    dataType: "xml",
                    success: function (data) {
                        $(data).find("Book").each(function () {
                            var bookId = $(this).find("BookId").text();
                            var title = $(this).find("Title").text();
                            var author = $(this).find("Author").text();
```

```
                        var row = "<tr><td>" + bookId + "</td><td>" + title +
"</td><td>" + author + "</td></tr>";
                        $("#tblBooks tbody").append(row);
                    });
                }
            });
        });

        $("#btnSave").click(function () {
            var xmlData = "<Books>";
            $("#tblBooks tbody tr").each(function () {
                var bookId = $(this).find("td:nth-child(1)").text();
                var title = $(this).find("td:nth-child(2)").text();
                var author = $(this).find("td:nth-child(3)").text();
                xmlData += "<Book><BookId>" + bookId + "</BookId><Title>"
+ title + "</Title><Author>" + author + "</Author></Book>";
            });
            xmlData += "</Books>";

            $.ajax({
                type: "POST",
                url: "SaveXML.aspx",
                data: { xmlData: xmlData },
                success: function () {
                    alert("XML data saved successfully.");
                }
            });
        });
    });
  </script>
</head>
<body>
```

```
  <form id="form1" runat="server">
    <h1>AJAX and XML Example</h1>

    <table id="tblBooks" border="1">
      <thead>
        <tr>
          <th>Book ID</th>
          <th>Title</th>
          <th>Author</th>
        </tr>
      </thead>
      <tbody></tbody>
    </table>

    <br />
    <input id="btnLoad" type="button" value="Load XML" />
    <input id="btnSave" type="button" value="Save XML" />

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

Create a new web form file (SaveXML.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="SaveXML.aspx.cs" Inherits="WebApplication1.SaveXML"
%>

<!DOCTYPE html>

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

```
    <title>Save XML</title>
</head>
<body>
    <form id="form1" runat="server">

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

In the code-behind file (SaveXML.aspx.cs), add the following code:

```
using System;
using System.IO;
using System.Web;

namespace WebApplication1
{
    public partial class SaveXML : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            string xmlData = Request.Form["xmlData"];
            string filePath = Server.MapPath("Books.xml");

            File.WriteAllText(filePath, xmlData);
        }
    }
}
```

Make sure you have a file named Books.xml in your web application's root folder with the following content:

```
<Books>
    <Book>
        <BookId>1</BookId>
        <Title>Book 1</Title>
        <Author>Author 1</Author>
    </Book>
    <Book>
        <BookId>2</BookId>
        <Title>Book 2</Title>
        <Author>Author 2</Author>
    </Book>
</Books>
```

b. Create a web application to demonstrate Form Security and Windows Security with
proper Authentication and Authorization properties.

Solution

N.A.

c. Create a web application to demonstrate use of various Ajax controls.

Solution

Create a new web form file (WebForm1.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>AJAX Controls Example</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></scri
pt>
    <script
src="https://cdn.jsdelivr.net/npm/sweetalert2@10.16.6/dist/sweetalert2.all.
min.js"></script>
    <script>
        $(document).ready(function () {
            $("#btnSubmit").click(function () {
                var name = $("#txtName").val();
                var email = $("#txtEmail").val();

                // Perform AJAX request to save data
                $.ajax({
                    type: "POST",
                    url: "SaveData.aspx",
                    data: { name: name, email: email },
                    success: function () {
                        // Display success message using SweetAlert2
                        Swal.fire({
                            icon: 'success',
                            title: 'Data Saved',
                            text: 'The data has been saved successfully.',
```

```
                    confirmButtonText: 'OK'
                });
            },
            error: function () {
                // Display error message using SweetAlert2
                Swal.fire({
                    icon: 'error',
                    title: 'Error',
                    text: 'An error occurred while saving the data.',
                    confirmButtonText: 'OK'
                });
            }
        });
    });
});
</script>
</head>
<body>
<form id="form1" runat="server">
    <h1>AJAX Controls Example</h1>

    <div>
        <label for="txtName">Name:</label>
        <input type="text" id="txtName" />
    </div>
    <div>
        <label for="txtEmail">Email:</label>
        <input type="text" id="txtEmail" />
    </div>
    <div>
        <input type="button" id="btnSubmit" value="Submit" />
    </div>
```

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

Create a new web form file (SaveData.aspx) with the following content:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="SaveData.aspx.cs" Inherits="WebApplication1.SaveData"
%>

<!DOCTYPE html>

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

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

In the code-behind file (SaveData.aspx.cs), add the following code:

```
using System;
using System.Web;

namespace WebApplication1
{
    public partial class SaveData : System.Web.UI.Page
    {
```

```
        protected void Page_Load(object sender, EventArgs e)
        {
            string name = Request.Form["name"];
            string email = Request.Form["email"];

            // Save the data to the database or perform any desired operation
        }
    }
}
```

11. Programs to create and use DLL

Solution

To create and use a DLL (Dynamic Link Library) in C#, you can follow these steps:

1. Create a new Class Library project:
   - Open Visual Studio.
   - Select "Create a new project."
   - Choose the "Class Library" template under the "Visual C#" or ".NET" category.
   - Provide a name for the project and specify the location.
   - Click "Create" to create the project.

2. Add classes and code to the DLL:
   - Within the Class Library project, add the classes and code that you want to include in the DLL.
   - Define public classes and methods that you want to expose to external applications that will use the DLL.

3. Build the DLL project:
   - Build the Class Library project to compile the code and generate the DLL file.
   - The DLL file will be created in the project's output directory (usually the "bin" folder) with the extension ".dll".

4. Use the DLL in another project:
   - Create a new project (such as a Console Application or ASP.NET project) where you want to use the DLL.
   - Right-click the project and select "Add Reference" to open the Reference Manager.

- Choose the "Browse" tab and locate the DLL file that you built in step 3.
- Select the DLL file and click "Add" to add a reference to the DLL in your project.
- Import or include the necessary namespaces or classes from the DLL in your code files.

5. Use the classes and methods from the DLL:
- Now, you can use the classes and methods defined in the DLL in your application code.
- Instantiate objects of the classes and call the methods exposed by the DLL to utilize its functionality.

Example:
Here's a simple example to demonstrate the creation and usage of a DLL:

1. Create a new Class Library project named "MyMathLibrary" and add the following code to the Class1.cs file:

```
namespace MyMathLibrary
{
    public class MathHelper
    {
        public int Add(int a, int b)
        {
            return a + b;
        }

        public int Multiply(int a, int b)
        {
            return a * b;
        }
    }
```

```
    }
```

2. Build the "MyMathLibrary" project. It will generate a "MyMathLibrary.dll" file in the project's output directory.

3. Create a new Console Application project named "MyApp" and add the following code to the Program.cs file:

```
    using System;
    using MyMathLibrary;

    namespace MyApp
    {
        class Program
        {
            static void Main(string[] args)
            {
                MathHelper mathHelper = new MathHelper();

                int sum = mathHelper.Add(2, 3);
                int product = mathHelper.Multiply(4, 5);

                Console.WriteLine("Sum: " + sum);
                Console.WriteLine("Product: " + product);
            }
        }
    }
```

4. Add a reference to the "MyMathLibrary.dll" in the "MyApp" project as mentioned in step 4.

5. Run the "MyApp" project, and it will use the methods from the DLL to perform addition and multiplication operations, displaying the results on the console.

This is a basic example to demonstrate the process of creating and using a DLL. In practice, DLLs can contain complex functionality, and their usage can vary depending on the specific requirements of your project.