| **Topic No.** | **Topic Name** | **Date** |
| --- | --- | --- |
| 1) | **Introduction to C#**  A] Program to demonstrate reference data types  B] Program using arrays  C] Program using array list  D] collections.  -Generic  - Non Generic | 27/01/24 |
| 2) | **Object Oriented features of C#**  A] Classes and Objects  B] Inheritance  C] Polymorphism | 03/02/24 |
| 3) | **Program to implement the concept of Exception Handling**  A] Built-in  B] User Defined | 10/02/24 |
| 4) | **Program to demonstrate getter and setter method.** | 10/02/24 |
| 5) | **Program based on File and directory handling.**  A] Sequential Access  B] Random Access  C]Demonstrate the use of directories. | 17/02/24 |
| 6) | **Implementation of serialization and deserialization**  **Concepts.** | 02/03/24 |
| 7) | **Windows Form Application. (Calculator)** | 02/03/24 |
| 8) | **Introduction to ASP.NET framework**  A] Adrotator Control  B] FileUpload Control  C] Calendar Control  D]PostBack and Cross Page Posting  E] Common control with Validation  F] Website using MasterPages and Content Pages | 09/03/24  16/03/24 |
| 9) | **ADO.NET**  A] Demonstration of Data Bound Controls  B] Connected Architecture (CRUD) (Using Database)  C] Disconnected Architecture (Using XML) | 23/03/24  30/03/24 |
| 10) | **State Management**  A] Using View state  B] Using Session object  C] Using Cookies  D] Using Page Caching | 06/04/24  13/04/24 |
| 11) | **Creating and deploying Web services**  A] Simple Web Service (Calculator)  B] Web Service with Database  C] Windows Communication Foundation (WCF) Service | 20/04/24  27/04/24 |
| 12) | **Program to demonstrate the implementation of User web controls** | 04/05/24 |
| 13) | **Design a web page using Asynchronous JavaScript and XML (AJAX)** | 04/05/24 |

## 

## **Practical 1 - Introduction to C#**

**Explanation:**

A] Reference Data Types: Reference types store addresses of their data. Modifying data through one reference affects all references pointing to that data.

B] Arrays: Arrays are fixed-size, strongly-typed collections. They store elements of the same type and allow indexed access to these elements.

C] Array List: ArrayList is a non-generic collection that can store any data type. It allows dynamic resizing and stores elements as objects, causing boxing/unboxing for value types.

D] Collections:

Generic: Generic collections, like List<T>, ensure type safety by enforcing that only a specified type can be added. They prevent runtime errors and improve performance by avoiding boxing/unboxing.

Non Generic: Non-generic collections, like ArrayList or Hashtable, store elements as objects, which can lead to runtime errors and inefficient memory usage due to boxing/unboxing.

#### **A] Program to demonstrate reference data types**

**Source Code:**

using System;

using System.IO.Pipes;

namespace Div

{

class Program

{

static void Main(string[] args)

{

int num1, num2, num3, num4, ans;

Console.WriteLine("Enter the number");

num1 = Convert.ToInt32(Console.ReadLine());

num2 = Convert.ToInt32(Console.ReadLine());

num3 = Convert.ToInt32(Console.ReadLine());

num4 = Convert.ToInt32(Console.ReadLine());

ans = num1 + num2 + num3 + num4;

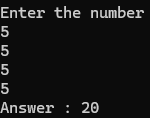
Console.WriteLine("Answer : " + ans);

}

}

}

**Output:**

****

#### **B] Program using arrays**

**Source Code:**

using System;

namespace SimpleArrayExample

{

class Program

{

static void Main(string[] args)

{

int[] numbers = { 1, 2, 3, 4, 5 };

Console.WriteLine("The array elements are:");

for (int i = 0; i < numbers.Length; i++)

{

Console.WriteLine(numbers[i]);

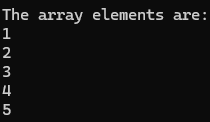
}

}

}

}

**Output:**



#### **C] Program using array list**

**Source Code:**

using System.Collections;

var arlist = new ArrayList();

arlist.Add(1);

arlist.Add("Student 1");

arlist.Add(true);

arlist.Add(4.5);

arlist.Add(null);

foreach (var ar in arlist)

{

Console.WriteLine(ar);

}

// adding elements using object initializer syntax

var arlist2 = new ArrayList()

{

2,"Student 2",false, 5.5, null

};

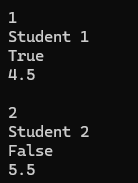
foreach (var ar2 in arlist2)

{

Console.WriteLine(ar2);

}

**Output:**



#### **D] Collections**

-Generic

- Non Generic

**Source Code:**

**Dictionary**

Dictionary<int, string> GenericDictionary = new Dictionary<int, string>();

GenericDictionary.Add(1, "fries");

GenericDictionary.Add(2, "burger");

GenericDictionary.Add(3, "pizza");

foreach (KeyValuePair<int, string> kyp in GenericDictionary)

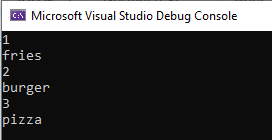
{

Console.WriteLine(kyp.Key);

Console.WriteLine(kyp.Value);

}

**Output:**

****

**Dictionary with user input**

Console.WriteLine("Enter no of students");

int num = Convert.ToInt32(Console.ReadLine());

Dictionary<string, string> student = new Dictionary<string, string>();

for (int i = 0; i < num; i++)

{

Console.WriteLine("Enter First Name");

string fname = Console.ReadLine();

Console.WriteLine("Enter Last Name");

string lname = Console.ReadLine();

student.Add(fname, lname);

}

foreach (KeyValuePair<string, string> kyp in student)

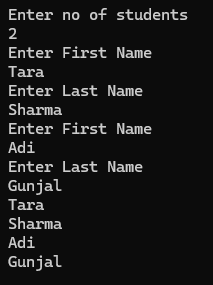
{

Console.WriteLine(kyp.Key);

Console.WriteLine(kyp.Value);

}

**Output:**

****

**Queue**

//create queue

Queue myQueue = new Queue();

//insert elements

myQueue.Enqueue("C++");

myQueue.Enqueue("C");

myQueue.Enqueue("C#");

//print

foreach (var item in myQueue)

{

Console.WriteLine(item);

}

//delete

myQueue.Dequeue();

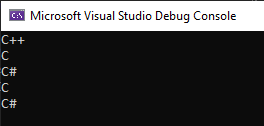
foreach (var item in myQueue)

{

Console.WriteLine(item);

}

**Output:**

****

**Stack**

**Code -**

Stack myStack = new Stack();

myStack.Push("item1");

myStack.Push("item2");

myStack.Push("item3");

myStack.Push("item4");

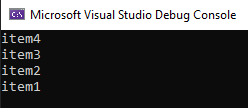
foreach (var item in myStack)

{

Console.WriteLine(item);

}

**Output -**

****

## **Practical 2 - Object Oriented features of C#**

**Explanation:**

A] Classes and Objects: Classes are blueprints for creating objects, encapsulating data (fields) and behavior (methods). Objects are instances of classes, holding specific data and allowing interaction through methods.

B] Inheritance: Inheritance allows a class (derived class) to inherit fields and methods from another class (base class), promoting code reuse and establishing a hierarchical relationship between classes.

C] Polymorphism: Polymorphism enables objects to be treated as instances of their base class rather than their actual derived class, allowing for method overriding and interface implementation, thus facilitating flexible and reusable code.

#### **A] Classes and Objects**

**Source Code:**

using System;

//using System.Collections.Generic;

using System.Text;

namespace ClassObjects

{

class Oops\_demo

{

class Student\_details

{

string name, std;

int rollno;

string div;

public void setdata()

{

Console.WriteLine("enter student details");

Console.WriteLine("enter the name: ");

name = Console.ReadLine();

Console.WriteLine("enter the std: ");

std = Console.ReadLine();

Console.WriteLine("enter the division: ");

div = Console.ReadLine();

Console.WriteLine("enter Roll number: ");

rollno = Convert.ToInt32(Console.ReadLine());

}

public void getdata()

{

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Student Details\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("Name: " + name); Console.WriteLine("Std: " + std);

Console.WriteLine("Div: " + div); Console.WriteLine("Roll No: " + rollno);

}

}

public static void Main()

{

Student\_details s = new Student\_details();

s.setdata();

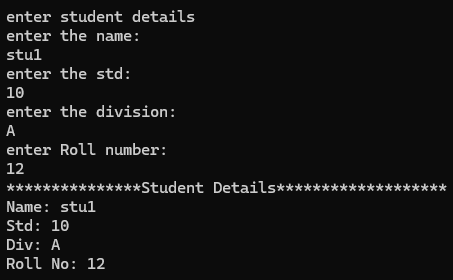
s.getdata();

}

}

}

**Output:**

****

#### **B] Inheritance**

**Source Code:**

using System;

// Parent class

public class Animal

{

public virtual void MakeSound()

{

Console.WriteLine("Animal sound");

}

}

// Child class inheriting from Animal

public class Dog : Animal

{

public override void MakeSound()

{

Console.WriteLine("Woof!");

}

}

class Program

{

static void Main(string[] args)

{

// Object instantiation

Dog myDog = new Dog();

// Accessing method from the parent class

myDog.MakeSound();

}

}

**Output:**

****

#### **C] Polymorphism**

**Source Code:**

**using System;**

// Parent class

public class Shape

{

public virtual double Area()

{

return 0;

}

}

// Child classes inheriting from Shape

public class Rectangle : Shape

{

private double Length { get; set; }

private double Breadth { get; set; }

public Rectangle(double length, double breadth)

{

Length = length;

Breadth = breadth;

}

public override double Area()

{

return Length \* Breadth;

}

}

public class Circle : Shape

{

private double Radius { get; set; }

public Circle(double radius)

{

Radius = radius;

}

public override double Area()

{

return Math.PI \* Math.Pow(Radius, 2);

}

}

class Program

{

static void Main(string[] args)

{

// Object instantiation

Rectangle myRectangle = new Rectangle(5, 4);

Circle myCircle = new Circle(3);

// Polymorphic method calls

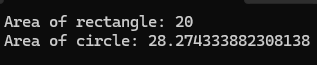
Console.WriteLine("Area of rectangle: " + myRectangle.Area());

Console.WriteLine("Area of circle: " + myCircle.Area());

}

}

**Output:**

****

## **Practical 3 - Program to implement the concept of Exception Handling**

**Explanation:**

A] Built-in Exception Handling: This involves using try, catch, and finally blocks to manage exceptions like IndexOutOfRangeException. The try block contains code that might throw an exception, the catch block handles specific exceptions, and the finally block contains code that runs regardless of whether an exception was thrown.

B] User-Defined Exception Handling: This involves creating a custom exception class by inheriting from the Exception class. Custom exceptions can be thrown using the throw keyword and caught using try and catch blocks. This is useful for handling application-specific error conditions.

#### **A] Built-in**

**Source Code:**

using System;

public class AgeException : Exception

{

public AgeException(String msg) : base(msg)

{

}

}

public class UserDefinedException

{

static void validate()

{

Console.WriteLine("Enter Your Age");

int age = Convert.ToInt32(Console.ReadLine());

if (age < 18)

{

throw new AgeException("Not eligible for voting");

}

else

{

Console.WriteLine("Eligible for voting");

}

}

public static void Main(string[] args)

{

try

{

validate();

}

catch (AgeException e)

{

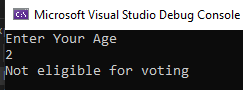
Console.WriteLine("Not eligible for voting");

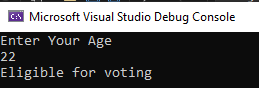
}

}

}

**Output:**

****

****

#### **B] User Defined**

**Source Code:**

using System;

public class InvalidMarksException : Exception

{

public InvalidMarksException(String msg) : base(msg)

{

}

}

public class UserDefinedException

{

static void validate(int marks)

{

if (marks > 100)

{

throw new InvalidMarksException("marks must be less than 100");

}

}

public static void Main(string[] args)

{

try

{

validate(110);

}

catch (InvalidMarksException e)

{

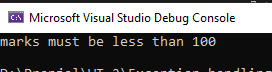
Console.WriteLine("Exception caught");

}

}

}

**Output:**

****

## **Practical 4 - Program to demonstrate getter and setter method.**

**Explanation:**

Getters and setters are used to encapsulate data within a class, providing controlled access and validation. In C#, properties are often used for this purpose, enabling the definition of getters and setters in a concise manner

**Source Code:**

using System;

class Human

{

string name;

string food;

string language;

public string Name

{

get { return name; }

set { name = value; }

}

public string Food

{

get { return food; }

set { food = value; }

}

public string Language

{

get { return language; }

set { language = value; }

}

public void Eat()

{

Console.WriteLine("Eats " + food);

}

public void Speak()

{

Console.WriteLine("Speaks " + language);

}

}

class CreateHuman

{

public static void Main(string[] args)

{

Human human = new Human();

human.Name = "John";

human.Food = "Burger";

human.Language = "English";

human.Eat();

human.Speak();

}

}

**Output:**



## **Practical 5 - Program based on File and directory handling.**

**Explanation:**

A] Sequential Access:

Sequential access reads or writes data to a file in a linear order, starting at the beginning and progressing to the end. A StreamWriter can be used to write lines to a file, and a StreamReader can read lines sequentially from the start to the end.

B] Random Access:

Random access allows reading and writing to any part of a file directly without processing it sequentially. This is typically achieved using the FileStream class, which allows positioning the cursor to any byte in the file for reading or writing.

C] Demonstrate the Use of Directories:

Directory handling involves creating, deleting, and navigating through directories. The Directory and DirectoryInfo classes provide methods to create new directories, list files and subdirectories, and check for the existence of directories. This can be useful for organizing files and managing file system structures programmatically.

#### **A] Sequential Access**

**Source Code:**

using System;

using System.IO;

namespace FH

{

class WritetoFile

{

public void Data()

{

Console.WriteLine("Enter the text that you want to write to the file:");

string str = Console.ReadLine();

using (StreamWriter sw = new StreamWriter("D:\\mydata.txt", true))

{

sw.WriteLine(str);

}

Console.WriteLine("Data written to file successfully.");

}

}

class Program

{

static void Main(string[] args)

{

WritetoFile wr = new WritetoFile();

wr.Data();

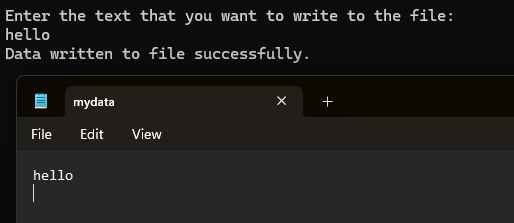
Console.ReadKey();

}

}

}

**Output:**

****

#### **B] Random Access**

**Source Code:**

using System;

using System.IO;

namespace FH

{

class WritetoFile

{

public void Data()

{

Console.WriteLine("Enter the text that you want to write to the file:");

string str = Console.ReadLine();

using (StreamWriter sw = new StreamWriter("D:\\mydata.txt", true))

{

sw.WriteLine(str);

}

Console.WriteLine("Data written to file successfully.");

}

}

class Program

{

static void Main(string[] args)

{

string[] names = new string[] { "abc", "pqr", "xyz" };

using (StreamWriter sw = new StreamWriter("D:\\names.txt"))

{

foreach (string s in names)

{

sw.WriteLine(s);

}

}

string line = "";

using (StreamReader sr = new StreamReader("D:\\names.txt"))

{

while ((line = sr.ReadLine()) != null)

{

Console.WriteLine(line);

}

}

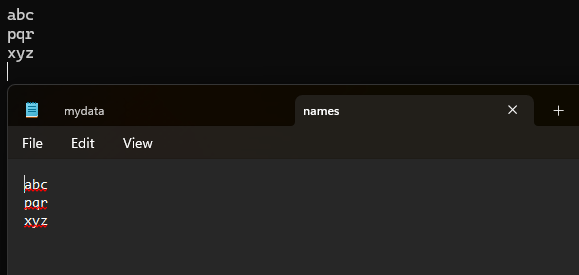
Console.ReadKey();

}

}

}

**Output:**

****

#### **C] Demonstrate the use of directories.**

**Source Code:**

using System;

using System.IO;

class Program

{

static void Main(string[] args)

{

// Create Directory named "Share" in C drive

// Using CreateDirectory() method

Directory.CreateDirectory("C:\\Share");

Console.WriteLine("Directory 'C:\\Share' created.");

string root = @"C:\Temp";

// Delete directory if it exists

if (Directory.Exists(root))

{

Directory.Delete(root, true); // Set true to recursively delete all files and subdirectories

Console.WriteLine("Directory 'C:\\Temp' deleted.");

}

else

{

Console.WriteLine("Directory 'C:\\Temp' does not exist.");

}

// Get files from "C:\\Tempo"

string folder = @"C:\Tempo";

if (Directory.Exists(folder))

{

string[] fileEntries = Directory.GetFiles(folder);

Console.WriteLine("Files in 'C:\\Tempo':");

foreach (string fileName in fileEntries)

{

Console.WriteLine(fileName);

}

}

else

{

Console.WriteLine("Directory 'C:\\Tempo' does not exist.");

}

// Move directory from "C:\\Tempo" to "C:\\NewTempo"

string sourceDirName = @"C:\Tempo";

string destDirName = @"C:\NewTempo";

if (Directory.Exists(sourceDirName))

{

try

{

Directory.Move(sourceDirName, destDirName);

Console.WriteLine("Directory moved from 'C:\\Tempo' to 'C:\\NewTempo' successfully.");

}

catch (IOException exp)

{

Console.WriteLine("Error moving directory: " + exp.Message);

}

}

else

{

Console.WriteLine("Source directory 'C:\\Tempo' does not exist.");

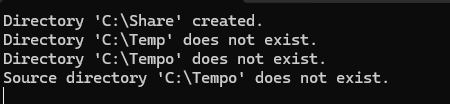
}

Console.ReadLine();

}

}

**Output:**

****

## **Practical 6 - Implementation of serialization and deserialization Concepts.**

**Explanation:**

Serialization: Serialization is the process of converting an object's state into a format that can be stored or transmitted (e.g., a file or a network stream). This allows the object to be recreated later by deserializing the data. In C#, this is commonly achieved using the System.Runtime.Serialization namespace and attributes like [Serializable].

Deserialization: Deserialization is the process of reconstructing an object from its serialized format. This involves reading the stored data and converting it back into an instance of the object. In C#, deserialization is often done using classes such as BinaryFormatter, XmlSerializer, or JsonSerializer, depending on the format used during serialization.

**Source Code:**

using System;

using System.IO;

using System.Text.Json;

class Program

{

static void Main(string[] args)

{

Person person = new Person

{

Name = "John Doe",

Age = 30,

Email = "john@example.com"

};

string jsonString = SerializeToJson(person);

Console.WriteLine("Serialized JSON:");

Console.WriteLine(jsonString);

Person deserializedPerson = DeserializeFromJson<Person>(jsonString);

Console.WriteLine("\nDeserialized Object:");

Console.WriteLine($"Name: {deserializedPerson.Name}");

Console.WriteLine($"Age: {deserializedPerson.Age}");

Console.WriteLine($"Email: {deserializedPerson.Email}");

}

static string SerializeToJson<T>(T obj)

{

return JsonSerializer.Serialize(obj);

}

static T DeserializeFromJson<T>(string jsonString)

{

return JsonSerializer.Deserialize<T>(jsonString);

}

}

class Person

{

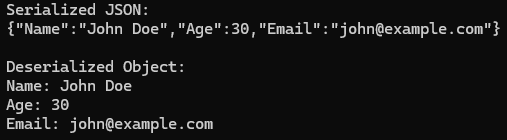
public string Name { get; set; }

public int Age { get; set; }

public string Email { get; set; }

}

**Output:**

****

## **Practical 7 - Windows Form Application (Calculator)**

**Explanation:**

A Windows Form Application is a graphical user interface (GUI) application that runs on the Windows operating system. It uses Windows Forms, a set of managed libraries in the .NET framework, to create desktop applications with rich user interfaces.

**Source Code:**

namespace Calculator

{

partial class Form1

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

private void InitializeComponent()

{

System.ComponentModel.ComponentResourceManager resources = new System.ComponentModel.ComponentResourceManager(typeof(Form1));

this.textBoxCal = new System.Windows.Forms.TextBox();

this.button1 = new System.Windows.Forms.Button();

this.button2 = new System.Windows.Forms.Button();

this.button3 = new System.Windows.Forms.Button();

this.buttonadd = new System.Windows.Forms.Button();

this.buttonminus = new System.Windows.Forms.Button();

this.button6 = new System.Windows.Forms.Button();

this.button5 = new System.Windows.Forms.Button();

this.button4 = new System.Windows.Forms.Button();

this.buttonmul = new System.Windows.Forms.Button();

this.button9 = new System.Windows.Forms.Button();

this.button8 = new System.Windows.Forms.Button();

this.button7 = new System.Windows.Forms.Button();

this.buttondiv = new System.Windows.Forms.Button();

this.clr = new System.Windows.Forms.Button();

this.button0 = new System.Windows.Forms.Button();

this.buttondot = new System.Windows.Forms.Button();

this.buttonequals = new System.Windows.Forms.Button();

this.SuspendLayout();

//

// textBoxCal

//

this.textBoxCal.Location = new System.Drawing.Point(13, 13);

this.textBoxCal.Name = "textBoxCal";

this.textBoxCal.Size = new System.Drawing.Size(318, 20);

this.textBoxCal.TabIndex = 0;

//

// button1

//

this.button1.Location = new System.Drawing.Point(13, 42);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(75, 23);

this.button1.TabIndex = 1;

this.button1.Text = "1";

this.button1.UseVisualStyleBackColor = true;

this.button1.Click += new System.EventHandler(this.button1\_Click);

// button2

this.button2.Location = new System.Drawing.Point(94, 42);

this.button2.Name = "button2";

this.button2.Size = new System.Drawing.Size(75, 23);

this.button2.TabIndex = 2;

this.button2.Text = "2";

this.button2.UseVisualStyleBackColor = true;

this.button2.Click += new System.EventHandler(this.button2\_Click);

// button3

this.button3.Location = new System.Drawing.Point(175, 42);

this.button3.Name = "button3";

this.button3.Size = new System.Drawing.Size(75, 23);

this.button3.TabIndex = 3;

this.button3.Text = "3";

this.button3.UseVisualStyleBackColor = true;

this.button3.Click += new System.EventHandler(this.button3\_Click);

// buttonadd

this.buttonadd.Location = new System.Drawing.Point(256, 42);

this.buttonadd.Name = "buttonadd";

this.buttonadd.Size = new System.Drawing.Size(75, 23);

this.buttonadd.TabIndex = 4;

this.buttonadd.Text = "+";

this.buttonadd.UseVisualStyleBackColor = true;

this.buttonadd.Click += new System.EventHandler(this.buttonadd\_Click);

// buttonminus

this.buttonminus.Location = new System.Drawing.Point(256, 71);

this.buttonminus.Name = "buttonminus";

this.buttonminus.Size = new System.Drawing.Size(75, 23);

this.buttonminus.TabIndex = 8;

this.buttonminus.Text = "-";

this.buttonminus.UseVisualStyleBackColor = true;

this.buttonminus.Click += new System.EventHandler(this.buttonminus\_Click);

// button6

this.button6.Location = new System.Drawing.Point(175, 71);

this.button6.Name = "button6";

this.button6.Size = new System.Drawing.Size(75, 23);

this.button6.TabIndex = 7;

this.button6.Text = "6";

this.button6.UseVisualStyleBackColor = true;

this.button6.Click += new System.EventHandler(this.button6\_Click);

// button5

this.button5.Location = new System.Drawing.Point(94, 71);

this.button5.Name = "button5";

this.button5.Size = new System.Drawing.Size(75, 23);

this.button5.TabIndex = 6;

this.button5.Text = "5";

this.button5.UseVisualStyleBackColor = true;

this.button5.Click += new System.EventHandler(this.button5\_Click);

// button4

this.button4.Location = new System.Drawing.Point(13, 71);

this.button4.Name = "button4";

this.button4.Size = new System.Drawing.Size(75, 23);

this.button4.TabIndex = 5;

this.button4.Text = "4";

this.button4.UseVisualStyleBackColor = true;

this.button4.Click += new System.EventHandler(this.button4\_Click);

// buttonmul

this.buttonmul.Location = new System.Drawing.Point(256, 100);

this.buttonmul.Name = "buttonmul";

this.buttonmul.Size = new System.Drawing.Size(75, 23);

this.buttonmul.TabIndex = 12;

this.buttonmul.Text = "\*";

this.buttonmul.UseVisualStyleBackColor = true;

this.buttonmul.Click += new System.EventHandler(this.buttonmul\_Click);

//

// button9

//

this.button9.Location = new System.Drawing.Point(175, 100);

this.button9.Name = "button9";

this.button9.Size = new System.Drawing.Size(75, 23);

this.button9.TabIndex = 11;

this.button9.Text = "9";

this.button9.UseVisualStyleBackColor = true;

this.button9.Click += new System.EventHandler(this.button9\_Click);

//

// button8

//

this.button8.Location = new System.Drawing.Point(94, 100);

this.button8.Name = "button8";

this.button8.Size = new System.Drawing.Size(75, 23);

this.button8.TabIndex = 10;

this.button8.Text = "8";

this.button8.UseVisualStyleBackColor = true;

this.button8.Click += new System.EventHandler(this.button8\_Click);

//

// button7

//

this.button7.Location = new System.Drawing.Point(13, 100);

this.button7.Name = "button7";

this.button7.Size = new System.Drawing.Size(75, 23);

this.button7.TabIndex = 9;

this.button7.Text = "7";

this.button7.UseVisualStyleBackColor = true;

this.button7.Click += new System.EventHandler(this.button7\_Click);

//

// buttondiv

//

this.buttondiv.Location = new System.Drawing.Point(256, 129);

this.buttondiv.Name = "buttondiv";

this.buttondiv.Size = new System.Drawing.Size(75, 23);

this.buttondiv.TabIndex = 16;

this.buttondiv.Text = "/";

this.buttondiv.UseVisualStyleBackColor = true;

this.buttondiv.Click += new System.EventHandler(this.buttondiv\_Click);

//

// clr

//

this.clr.Location = new System.Drawing.Point(13, 158);

this.clr.Name = "clr";

this.clr.Size = new System.Drawing.Size(237, 23);

this.clr.TabIndex = 15;

this.clr.Text = "CLR";

this.clr.UseVisualStyleBackColor = true;

this.clr.Click += new System.EventHandler(this.clr\_Click);

//

// button0

//

this.button0.Location = new System.Drawing.Point(94, 129);

this.button0.Name = "button0";

this.button0.Size = new System.Drawing.Size(156, 23);

this.button0.TabIndex = 14;

this.button0.Text = "0";

this.button0.UseVisualStyleBackColor = true;

this.button0.Click += new System.EventHandler(this.button0\_Click);

//

// buttondot

//

this.buttondot.Location = new System.Drawing.Point(13, 129);

this.buttondot.Name = "buttondot";

this.buttondot.Size = new System.Drawing.Size(75, 23);

this.buttondot.TabIndex = 13;

this.buttondot.Text = ".";

this.buttondot.UseVisualStyleBackColor = true;

this.buttondot.Click += new System.EventHandler(this.buttondot\_Click);

//

// buttonequals

//

this.buttonequals.Location = new System.Drawing.Point(256, 158);

this.buttonequals.Name = "buttonequals";

this.buttonequals.Size = new System.Drawing.Size(75, 23);

this.buttonequals.TabIndex = 17;

this.buttonequals.Text = "=";

this.buttonequals.UseVisualStyleBackColor = true;

this.buttonequals.Click += new System.EventHandler(this.buttonequals\_Click);

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.BackColor = System.Drawing.SystemColors.ActiveBorder;

this.ClientSize = new System.Drawing.Size(343, 200);

this.Controls.Add(this.buttonequals);

this.Controls.Add(this.buttondiv);

this.Controls.Add(this.clr);

this.Controls.Add(this.button0);

this.Controls.Add(this.buttondot);

this.Controls.Add(this.buttonmul);

this.Controls.Add(this.button9);

this.Controls.Add(this.button8);

this.Controls.Add(this.button7);

this.Controls.Add(this.buttonminus);

this.Controls.Add(this.button6);

this.Controls.Add(this.button5);

this.Controls.Add(this.button4);

this.Controls.Add(this.buttonadd);

this.Controls.Add(this.button3);

this.Controls.Add(this.button2);

this.Controls.Add(this.button1);

this.Controls.Add(this.textBoxCal);

this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));

this.Name = "Form1";

this.Text = "Calculator";

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.TextBox textBoxCal;

private System.Windows.Forms.Button button1;

private System.Windows.Forms.Button button2;

private System.Windows.Forms.Button button3;

private System.Windows.Forms.Button buttonadd;

private System.Windows.Forms.Button buttonminus;

private System.Windows.Forms.Button button6;

private System.Windows.Forms.Button button5;

private System.Windows.Forms.Button button4;

private System.Windows.Forms.Button buttonmul;

private System.Windows.Forms.Button button9;

private System.Windows.Forms.Button button8;

private System.Windows.Forms.Button button7;

private System.Windows.Forms.Button buttondiv;

private System.Windows.Forms.Button clr;

private System.Windows.Forms.Button button0;

private System.Windows.Forms.Button buttondot;

private System.Windows.Forms.Button buttonequals;

}

}

Form.cs

﻿using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Calculator

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

Int32 FirstNum;

String Operator;

// Operands

private void button1\_Click(object sender, EventArgs e)

{

if(textBoxCal.Text == "1" && textBoxCal.Text != null)

{

textBoxCal.Text = "1";

}

else

{

textBoxCal.Text += "1";

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (textBoxCal.Text == "2" && textBoxCal.Text != null)

{

textBoxCal.Text = "2";

}

else

{

textBoxCal.Text += "2";

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (textBoxCal.Text == "3" && textBoxCal.Text != null)

{

textBoxCal.Text = "3";

}

else

{

textBoxCal.Text += "3";

}

}

private void button4\_Click(object sender, EventArgs e)

{

if (textBoxCal.Text == "4" && textBoxCal.Text != null)

{

textBoxCal.Text = "4";

}

else

{

textBoxCal.Text += "4";

}

}

private void button5\_Click(object sender, EventArgs e)

{

if (textBoxCal.Text == "5" && textBoxCal.Text != null)

{

textBoxCal.Text = "5";

}

else

{

textBoxCal.Text += "5";

}

}

private void button6\_Click(object sender, EventArgs e)

{

if (textBoxCal.Text == "6" && textBoxCal.Text != null)

{

textBoxCal.Text = "6";

}

else

{

textBoxCal.Text += "6";

}

}

private void button7\_Click(object sender, EventArgs e)

{

if (textBoxCal.Text == "7" && textBoxCal.Text != null)

{

textBoxCal.Text = "7";

}

else

{

textBoxCal.Text += "7";

}

}

private void button8\_Click(object sender, EventArgs e)

{

if (textBoxCal.Text == "8" && textBoxCal.Text != null)

{

textBoxCal.Text = "8";

}

else

{

textBoxCal.Text += "8";

}

}

private void button9\_Click(object sender, EventArgs e)

{

if (textBoxCal.Text == "9" && textBoxCal.Text != null)

{

textBoxCal.Text = "9";

}

else

{

textBoxCal.Text += "9";

}

}

private void button0\_Click(object sender, EventArgs e)

{

if (textBoxCal.Text == "0" && textBoxCal.Text != null)

{

textBoxCal.Text = "0";

}

else

{

textBoxCal.Text += "0";

}

}

private void buttondot\_Click(object sender, EventArgs e)

{

if (textBoxCal.Text == "0" && textBoxCal.Text != null)

{

textBoxCal.Text = ".";

}

else

{

textBoxCal.Text += ".";

}

}

// Operators

private void buttonadd\_Click(object sender, EventArgs e)

{

FirstNum = Convert.ToInt32(textBoxCal.Text);

textBoxCal.Text = "";

Operator = "+";

}

private void buttonminus\_Click(object sender, EventArgs e)

{

FirstNum = Convert.ToInt32(textBoxCal.Text);

textBoxCal.Text = "";

Operator = "-";

}

private void buttonmul\_Click(object sender, EventArgs e)

{

FirstNum = Convert.ToInt32(textBoxCal.Text);

textBoxCal.Text = "";

Operator = "\*";

}

private void buttondiv\_Click(object sender, EventArgs e)

{

FirstNum = Convert.ToInt32(textBoxCal.Text);

textBoxCal.Text = "";

Operator = "/";

}

private void clr\_Click(object sender, EventArgs e)

{

if (textBoxCal.Text != null)

{

textBoxCal.Text = "";

}

}

private void buttonequals\_Click(object sender, EventArgs e)

{

Int32 SecondNum = Convert.ToInt32(textBoxCal.Text);

switch(Operator)

{

case "+":

textBoxCal.Text = (FirstNum + SecondNum).ToString();

break;

case "-":

textBoxCal.Text = (FirstNum - SecondNum).ToString();

break;

case "/":

if(!SecondNum.Equals(0))

{

textBoxCal.Text = (FirstNum / SecondNum).ToString();

}

else

{

textBoxCal.Text = "Cannot divide by Zero";

}

break;

case "\*":

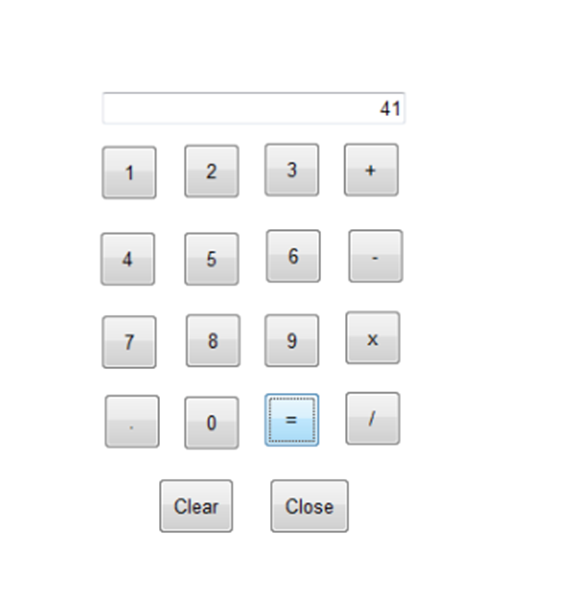
textBoxCal.Text = (FirstNum \* SecondNum).ToString();

break;

}

}

}

**Output:**

## **Practical 8 - Introduction to ASP.NET framework**

**Explanation:**

A] Adrotator Control:

The AdRotator control in ASP.NET is used to display advertisements on a web page. It rotates through a set of images or text ads at regular intervals, providing a simple way to manage and display advertisements dynamically.

B] FileUpload Control:

The FileUpload control allows users to upload files from their local system to the web server. It provides an input field and a browse button, enabling users to select files for uploading, which can then be processed by the server-side code.

C] Calendar Control:

The Calendar control is used to display a calendar interface on a web page. It allows users to select dates interactively, making it useful for applications that require date selection functionality, such as event scheduling or appointment booking systems.

D] PostBack and Cross Page Posting:

PostBack is a mechanism in ASP.NET that allows a web page to post data back to itself, typically in response to user actions such as button clicks. Cross Page Posting, on the other hand, enables a web page to post data to a different page, facilitating communication between pages and passing data between them.

E] Common Control with Validation:

ASP.NET provides a set of common controls with built-in validation capabilities to enforce data integrity and ensure that user input meets specified criteria. Controls like TextBox, DropDownList, and CheckBox can be combined with validation controls like RequiredFieldValidator, RegularExpressionValidator, etc., to perform client-side and server-side validation.

F] Website using MasterPages and Content Pages:

MasterPages provide a template for the layout of a website, defining the common structure and elements (e.g., header, footer, navigation menu) that are shared across multiple pages. Content Pages inherit the layout defined in a MasterPage and allow developers to focus on the unique content of each page while maintaining a consistent look and feel throughout the website. This architecture promotes code reusability, maintainability, and consistency in web application development.

#### **A] Adrotator Control**

**Source Code:**

<?xml version="1.0" encoding="utf-8" ?>

<Advertisements>

<Ad>

<ImageUrl>football.jpg</ImageUrl>

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

<AlternateText>

Football is emotion

</AlternateText>

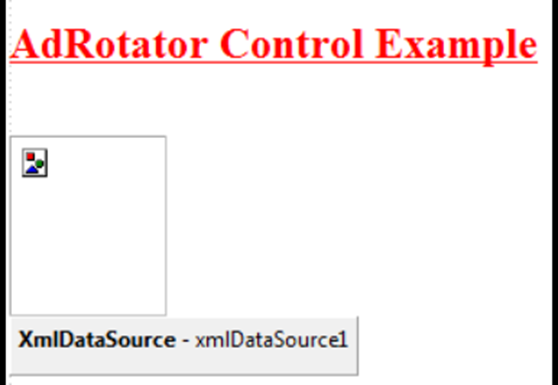
<Impressions>20</Impressions>

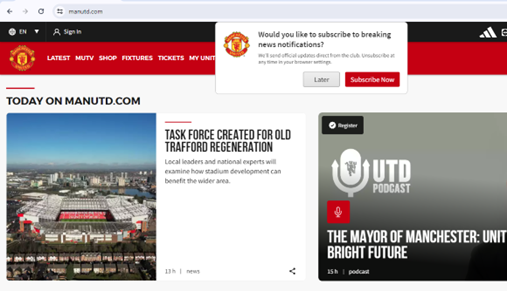
<keyword>football</keyword>

</Ad>

</Advertisements>

**Output:**

****

****

#### **B] FileUpload Control**

**Source Code:**

using System;

using System.IO;

using System.Web.UI;

public partial class Upload : Page

{

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

{

if (FileUpload1.HasFile)

{

// Save the file to the server

string fileName = Path.GetFileName(FileUpload1.FileName);

FileUpload1.SaveAs(Server.MapPath("~/uploads/") + fileName);

StatusLabel.Text = "File uploaded successfully!";

}

else

{

StatusLabel.Text = "Please select a file to upload.";

}

}

}

**Output:**



#### **C] Calendar Control**

**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace WebApplication7

{

public partial class WebForm1 : System.Web.UI.Page

{

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

{

}

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

{

Calendar1.Caption = "My Calendar";

Calendar1.FirstDayOfWeek = FirstDayOfWeek.Monday;

Calendar1.NextPrevFormat = NextPrevFormat.ShortMonth;

Calendar1.TitleFormat = TitleFormat.Month;

lbl2.Text = "Todays Date" + Calendar1.TodaysDate.ToShortDateString();

lbl3.Text = "Vaccation Start: 1-07-2024";

if (Calendar1.SelectedDate.ToShortDateString() == "7-01-2024")

lbl3.Text = "<b> vaccation start </b>";

if (Calendar1.SelectedDate.ToShortDateString() == "7-20-2024")

lbl3.Text = "<b> Vaccation End </b>";

}

protected void Calendar1\_DayRender(object sender, DayRenderEventArgs e)

{

if (e.Day.Date.Day == 5 && e.Day.Date.Month == 9)

{

e.Cell.BackColor = System.Drawing.Color.Yellow;

Label lbl = new Label();

lbl.Text = "<br> Teachers Day";

e.Cell.Controls.Add(lbl);

Image g1 = new Image();

g1.ImageUrl = "td.png";

g1.Height = 20;

g1.Width = 20;

e.Cell.Controls.Add(g1);

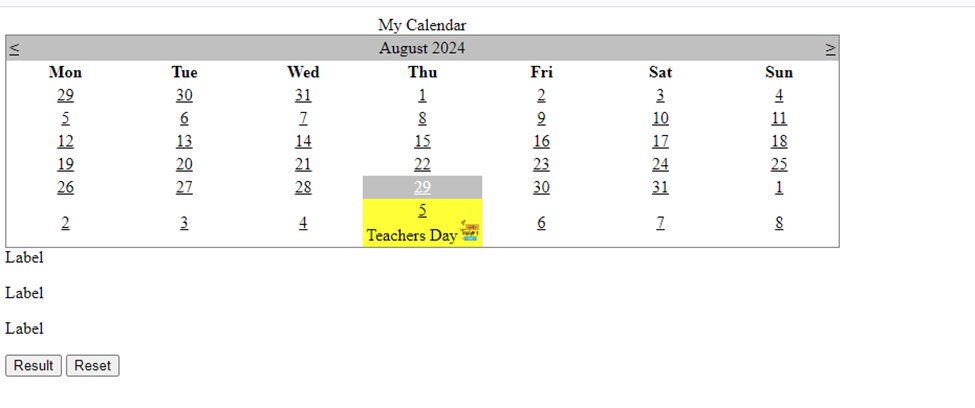
}

}

}

}

**Output:**



#### **D] PostBack and Cross Page Posting**

**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection.Emit;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace WebApplication3

{

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

{

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

{

if (!IsPostBack && Request.QueryString["name"] != null)

{

Label1.Text = HttpUtility.UrlDecode(Request.QueryString["name"]);

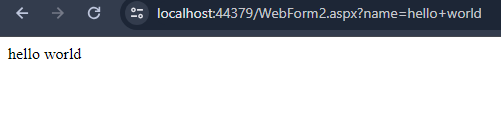
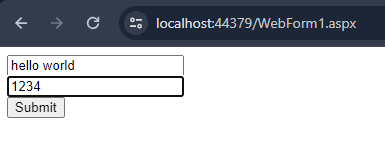
}

}

}

}

**Output:**



#### **E] Common control with Validation**

**Source Code:**

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

<!DOCTYPE html>

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

<head runat="server">

<title>Validation Example</title>

</head>

<body>

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

<div>

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

<asp:RequiredFieldValidator ID="NameValidator" runat="server"

ControlToValidate="NameTextBox"

ErrorMessage="Name is required."

Text="\*"

ForeColor="Red" />

<br />

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

<asp:RequiredFieldValidator ID="EmailValidator" runat="server"

ControlToValidate="EmailTextBox"

ErrorMessage="Email is required."

Text="\*"

ForeColor="Red" />

<asp:RegularExpressionValidator ID="EmailFormatValidator" runat="server"

ControlToValidate="EmailTextBox"

ErrorMessage="Invalid email format."

ValidationExpression="\w+([-+.']\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*"

Text="\*"

ForeColor="Red" />

<br />

<asp:Button ID="SubmitButton" runat="server" Text="Submit" OnClick="SubmitButton\_Click" />

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

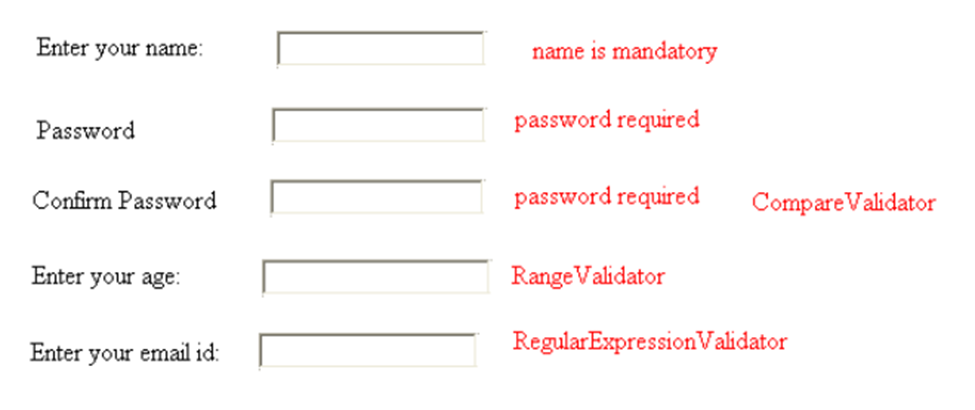
</div>

</form>

</body>

</html>

**Output:**



#### **F] Website using MasterPages and Content Pages**

**Source Code:**

**Default.aspx**

<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="Practical8\_F.\_Default" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">

<h2>Welcome to My Website</h2>

<p>This is the home page.</p>

</asp:Content>

Site.Master

<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site.master.cs" Inherits="Practical8\_F.SiteMaster" %>

<!DOCTYPE html>

<html>

<head runat="server">

<title></title>

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

</asp:ContentPlaceHolder>

<style>

body {

font-family: Arial, sans-serif;

}

.header, .footer {

background-color: #f8f8f8;

padding: 10px;

text-align: center;

}

.content {

margin: 20px;

}

</style>

</head>

<body>

<div class="header">

<h1>My Website</h1>

</div>

<div class="content">

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

</asp:ContentPlaceHolder>

</div>

<div class="footer">

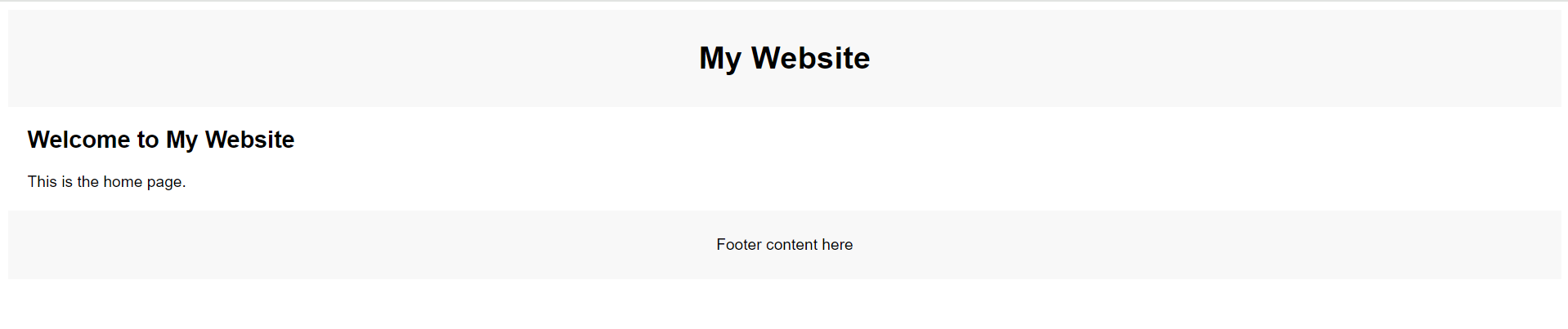
<p>Footer content here</p>

</div>

</body>

</html>

**Output:**



## **Practical 9 - ADO.NET**

**Explanation:**

A] Demonstration of Data Bound Controls

Data bound controls in ADO.NET are UI elements that are linked to data sources, allowing them to display and manipulate data dynamically. These controls are essential for creating data-driven applications, particularly in Windows Forms and ASP.NET applications.

* TextBox: Used for displaying and editing single pieces of data.
* ComboBox: Displays a list of items from a data source.
* DataGridView: Displays and allows manipulation of tabular data.
* ListBox: Displays a list of items from a data source.

B] Connected Architecture (CRUD) (Using Database)

In a connected architecture, the application remains connected to the database while performing operations like Create, Read, Update, and Delete (CRUD). This is achieved using ADO.NET objects like SqlConnection, SqlCommand, SqlDataReader, etc.

C] Disconnected Architecture (Using XML)

In a disconnected architecture, the application does not maintain a constant connection to the database. Instead, it fetches data into a local cache (e.g., a DataSet) and can manipulate the data while being disconnected. The changes can later be synchronized back to the database. XML is often used for data storage and transfer in disconnected scenarios.

#### **A] Demonstration of Data Bound Controls**

**Source Code:**

Add > New Item > SQL Server Database

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Practical9.WebForm1" %>

<!DOCTYPE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

hiiii

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataKeyNames="Id" DataSourceID="SqlDataSource2">

<Columns>

<asp:BoundField DataField="Id" HeaderText="Id" ReadOnly="True" SortExpression="Id" />

<asp:BoundField DataField="username" HeaderText="username" SortExpression="username" />

</Columns>

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource2" runat="server" ConnectionString="<%$ ConnectionStrings:ConnectionString %>" ProviderName="<%$ ConnectionStrings:ConnectionString.ProviderName %>" SelectCommand="SELECT \* FROM [Users]"></asp:SqlDataSource>

<asp:SqlDataSource ID="SqlDataSource1" runat="server"></asp:SqlDataSource>

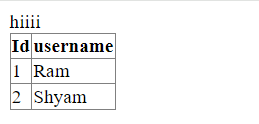
</div>

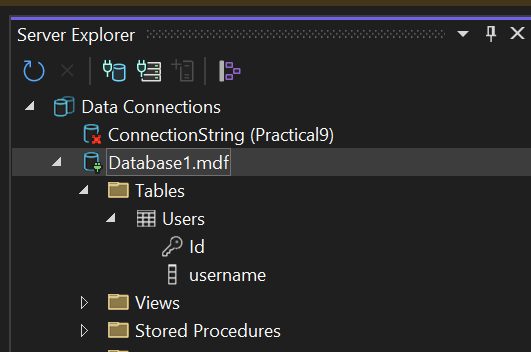
</form>

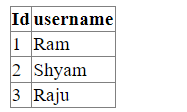
</body>

</html>

**Output:**

****

****

****

#### **B] Connected Architecture (CRUD) (Using Database)**

**Source Code:**

**To get Connection String > DB Properties**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using static System.Net.Mime.MediaTypeNames;

namespace Practical9

{

public partial class WebForm1 : System.Web.UI.Page

{

string connectionString = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\kpanchal\\source\\repos\\Practical9\\Practical9\\App\_Data\\Database1.mdf;Integrated Security=True";

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

{

BindGridView();

}

private void BindGridView()

{

string query = "SELECT \* FROM Users";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

GridView1.DataSource = reader;

GridView1.DataBind();

connection.Close();

}

}

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

{

string username = UsernameTextBox.Text;

int id = Convert.ToInt32(IdTextBox.Text);

string query = "INSERT INTO Users (Id,username) VALUES (@id,@username)";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@username", username);

command.Parameters.AddWithValue("@id", id);

connection.Open();

command.ExecuteNonQuery();

connection.Close();

BindGridView();

}

}

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

{

int id = Convert.ToInt32(IdTextBox.Text);

string username = UsernameTextBox.Text;

string query = "UPDATE Users SET username = @username WHERE Id = @Id";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Id", id);

command.Parameters.AddWithValue("@username", username);

connection.Open();

command.ExecuteNonQuery();

connection.Close();

BindGridView();

}

}

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

{

int id = Convert.ToInt32(IdTextBox.Text);

string query = "DELETE FROM Users WHERE Id = @Id";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Id", id);

connection.Open();

command.ExecuteNonQuery();

connection.Close();

BindGridView();

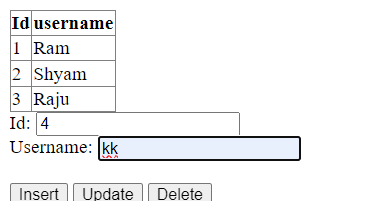
}

}

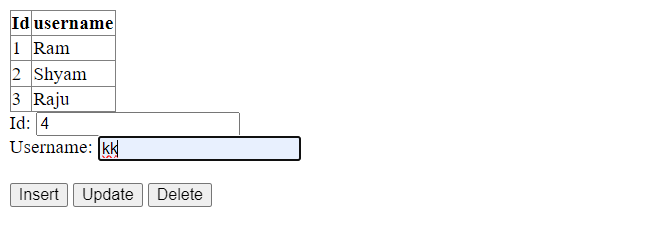
}

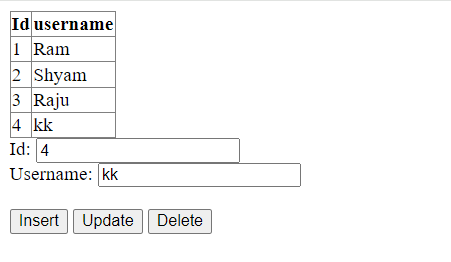
}

**Output:**

****

**—**

****

****

#### **C]** **Disconnected Architecture (Using XML)**

**Source Code:**

<?xml version="1.0" encoding="utf-8" ?>

<catalog>

<book id="bk103">

<author>Corets, Eva</author>

<title>Maeve Ascendant</title>

<genre>Fantasy</genre>

<price>5.95</price>

<publish\_date>2000-11-17</publish\_date>

<description>

After the collapse of a nanotechnology

society in England, the young survivors lay the

foundation for a new society.

</description>

</book>

<book id="bk104">

<author>Corets, Eva</author>

<title>Oberon's Legacy</title>

<genre>Fantasy</genre>

<price>5.95</price>

<publish\_date>2001-03-10</publish\_date>

<description>

In post-apocalypse England, the mysterious

agent known only as Oberon helps to create a new life

for the inhabitants of London. Sequel to Maeve

Ascendant.

</description>

</book>

<book id="bk105">

<author>Corets, Eva</author>

<title>The Sundered Grail</title>

<genre>Fantasy</genre>

<price>5.95</price>

<publish\_date>2001-09-10</publish\_date>

<description>

The two daughters of Maeve, half-sisters,

battle one another for control of England. Sequel to

Oberon's Legacy.

</description>

</book>

<book id="bk106">

<author>Randall, Cynthia</author>

<title>Lover Birds</title>

<genre>Romance</genre>

<price>4.95</price>

<publish\_date>2000-09-02</publish\_date>

<description>

When Carla meets Paul at an ornithology

conference, tempers fly as feathers get ruffled.

</description>

</book>

<book id="bk107">

<author>Thurman, Paula</author>

<title>Splish Splash</title>

<genre>Romance</genre>

<price>4.95</price>

<publish\_date>2000-11-02</publish\_date>

<description>

A deep sea diver finds true love twenty

thousand leagues beneath the sea.

</description>

</book>

<book id="bk108">

<author>Knorr, Stefan</author>

<title>Creepy Crawlies</title>

<genre>Horror</genre>

<price>4.95</price>

<publish\_date>2000-12-06</publish\_date>

<description>

An anthology of horror stories about roaches,

centipedes, scorpions and other insects.

</description>

</book>

</catalog>

**WebForm2.aspx.cs**

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Practical9

{

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

{

DataSet ds = new DataSet();

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

{

ds.ReadXml(Server.MapPath("~/XMLFile1.xml"));

GridView1.DataSource = ds;

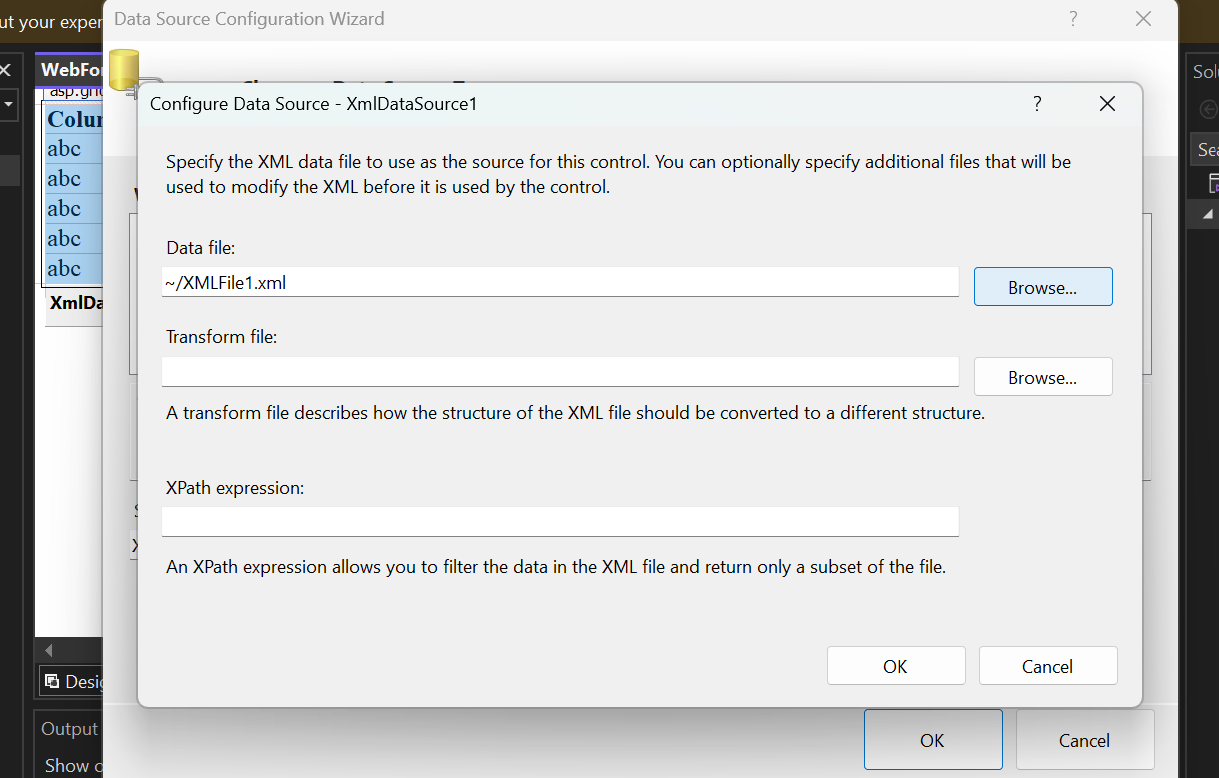
GridView1.DataBind();

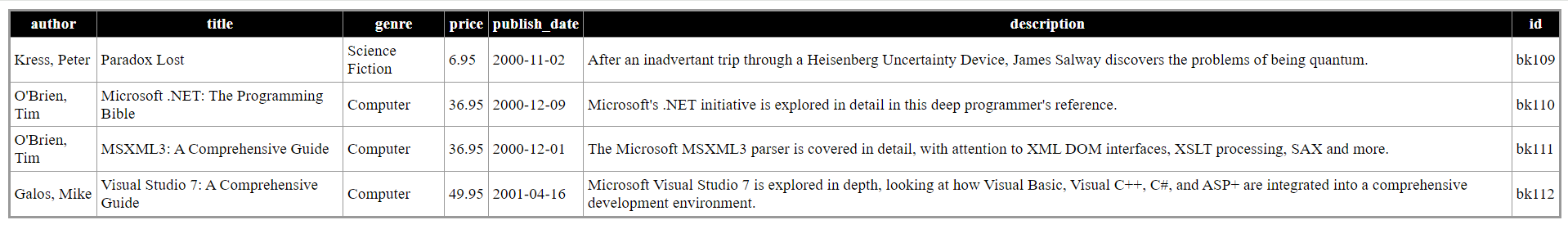
}

}

}

**Output:**

****

****

## 

## **Practical 10 - State Management**

**A] Using ViewState**

ViewState is used to preserve page and control values between postbacks. It's stored in a hidden field on the page and serialized into a string, making it a lightweight method for maintaining the state of a web form.

**B] Using Session Object**

The Session object stores user data across multiple pages during a user session. It allows for storing objects and is useful for retaining user-specific information like login credentials, shopping cart items, etc., until the session expires or is abandoned.

**C] Using Cookies**

Cookies are small text files stored on the client-side that hold user-specific data. They persist across browser sessions and can store user preferences, login details, and other information, with a defined expiration date.

**D] Using Page Caching**

Page caching stores a web page's content and serves it for subsequent requests without reprocessing. It improves performance by reducing server load and response time, particularly for pages that do not change frequently.

#### **A] Using View state**

**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Practical\_10

{

public partial class WebForm1 : System.Web.UI.Page

{

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

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

ViewState["name"] = txtname.Text;

ViewState["add"] = txtadd.Text;

ViewState["edu"] = txtedu.Text;

txtname.Text = txtadd.Text = txtedu.Text = String.Empty;

}

protected void Button2\_Click(object sender, EventArgs e)

{

if (ViewState["name"] != null)

{

txtname.Text = ViewState["name"].ToString();

}

if (ViewState["add"] != null)

{

txtadd.Text = ViewState["add"].ToString();

}

if (ViewState["edu"] != null)

{

txtedu.Text = ViewState["edu"].ToString();

}

}

}

}

**WebForm1.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Practical\_10.WebForm1" %>

<!DOCTYPE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<div>

<asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="XX-Large" Text="Your Name:"></asp:Label>

<asp:TextBox ID="txtname" runat="server" Font-Bold="False"></asp:TextBox>

</div>

<p>

<asp:Label ID="Label2" runat="server" Font-Bold="True" Font-Size="XX-Large" Text="Address:"></asp:Label>

<asp:TextBox ID="txtadd" runat="server" Font-Overline="False" ValidateRequestMode="Disabled"></asp:TextBox>

</p>

<p>

<asp:Label ID="Label3" runat="server" Font-Bold="True" Font-Size="XX-Large" Text="Education:"></asp:Label>

<asp:TextBox ID="txtedu" runat="server" Height="22px"></asp:TextBox>

</p>

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

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

</div>

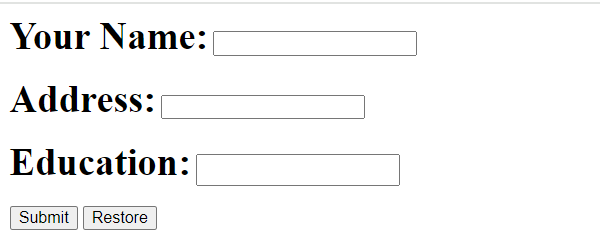
</div>

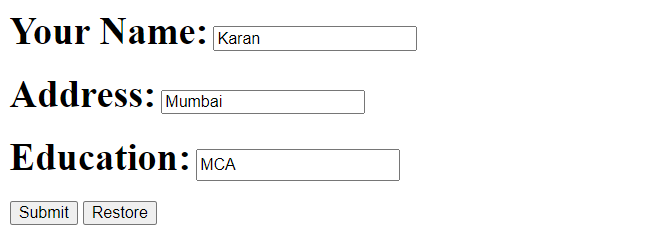
</form>

</body>

</html>

**Output:**

****

****

#### **B] Using Session object**

**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Practical\_10

{

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

{

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

{

lblwelcome.Text = "Welcome:" + Session["un"].ToString();

lblmsg.Text = "Welcome to MCA " + Session["pl"].ToString() + "!!";

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Practical\_10

{

public partial class WebForm1 : System.Web.UI.Page

{

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

{

}

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

{

//create a session

Session["un"] = txtname.Text;

Session["pl"] = txtprog.Text;

//redirect to page 2

Response.Redirect("~/WebForm2.aspx");

}

}

}

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Practical\_10.WebForm1" %>

<!DOCTYPE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="XX-Large" Text="Your Name:"></asp:Label>

<asp:TextBox ID="txtname" runat="server" Height="25px" Width="220px"></asp:TextBox>

<br />

<br />

<asp:Label ID="Label2" runat="server" Font-Bold="True" Font-Size="XX-Large" Text="Fav Programming Language:"></asp:Label>

&nbsp;&nbsp;&nbsp;

<asp:TextBox ID="txtprog" runat="server" Height="25px" Width="220px"></asp:TextBox>

<br />

<br />

<br />

<asp:Button ID="btnsub" runat="server" BorderColor="#6699FF" Font-Bold="True" Font-Size="Medium" Height="35px" OnClick="btnsub\_Click" Text="Submit" Width="90px" />

</div>

</form>

</body>

</html>

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm2.aspx.cs" Inherits="Practical\_10.WebForm2" %>

<!DOCTYPE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<asp:Label ID="lblwelcome" runat="server" Font-Bold="True" Font-Size="XX-Large" ForeColor="#000000" Text="Label"></asp:Label>

<br />

<br />

<asp:Label ID="lblmsg" runat="server" Font-Bold="True" Font-Italic="True" Font-Size="XX-Large" ForeColor="#000000" Text="Label"></asp:Label>

<br />

<br />

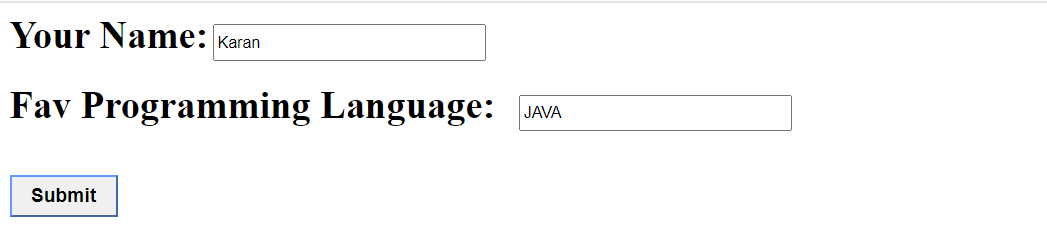
</div>

</form>

</body>

</html>

**Output:**

****

****

#### **C] Using Cookies**

**Source Code:**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="books.aspx.cs" Inherits="Practical\_10.books" %>

<!DOCTYPE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="XX-Large" Text="Select a book:"></asp:Label>

<br />

<br />

<asp:RadioButtonList ID="rblbooks" runat="server">

</asp:RadioButtonList>

<br />

<br />

<asp:Button ID="btnadd" runat="server" Font-Bold="True" Font-Size="Large" OnClick="btnadd\_Click" Text="Add Book" />

<asp:Button ID="btnview" runat="server" Font-Bold="True" Font-Size="Medium" PostBackUrl="~/book\_details.aspx" Text="View Details" />

<br />

<br />

<br />

<asp:Label ID="lblmsg" runat="server" Font-Bold="True" ForeColor="#6666FF" Text="Label"></asp:Label>

</div>

</form>

</body>

</html>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Practical\_10

{

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

{

Dictionary<string, string> books\_dt;

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

{

books\_dt = new Dictionary<string, string>();

books\_dt.Add("C++", "Bjarne stroupstrup");

books\_dt.Add("Java", "Ivor Horton");

books\_dt.Add("C", "Dennis Ritchie");

books\_dt.Add("Machine Learning", "Tom micthell");

books\_dt.Add("Python", "Allen Downey");

rblbooks.Items.Clear();

foreach (String s in books\_dt.Keys)

{

rblbooks.Items.Add(s);

}

}

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

{

}

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

{

if (rblbooks.SelectedIndex > 0)

{

//get the selected index

String bn = rblbooks.SelectedValue;

String bdes = books\_dt[bn];

//create a coookies

HttpCookie cl = new HttpCookie(bn, bdes);

//add cookie

Response.Cookies.Add(cl);

}

else

{

lblmsg.Text = "Please Select a book";

}

}

}

}

**book\_details.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="book\_details.aspx.cs" Inherits="Practical\_10.book\_details" %>

<!DOCTYPE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<asp:Label ID="Label1" runat="server" Font-Bold="True" ForeColor="#000000" Text="Hey,You have Selected"></asp:Label>

</div>

<asp:ListBox ID="lstdetails" runat="server" BackColor="#F6F6F6" Font-Bold="True" Font-Size="XX-Large" ></asp:ListBox>

<p>

<asp:LinkButton ID="LinkButton1" runat="server" Font-Bold="True" Font-Italic="True" PostBackUrl="~/books.aspx">Select Another book</asp:LinkButton>

</p>

</form>

</body>

</html>

**book\_details**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Practical\_10

{

public partial class book\_details : System.Web.UI.Page

{

HttpCookieCollection cookies;

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

{

//Fecth all cookies

cookies = Request.Cookies;

//display cookies in a listbox

if (cookies.Count > 0)

{

for (int i = 0; i < cookies.Count; i++)

{

lstdetails.Items.Add("Book:" + cookies[i].Name + "Author:" + cookies[i].Value);

}

}

else

{

lstdetails.Items.Add("No books selected");

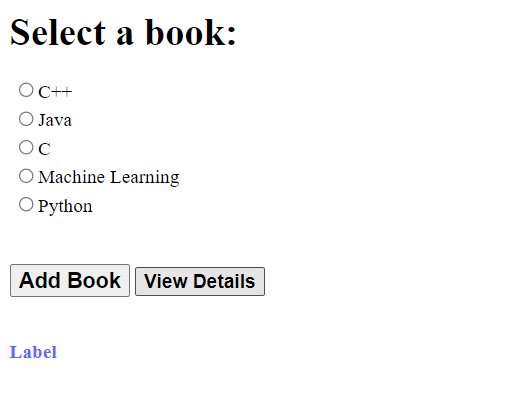
}

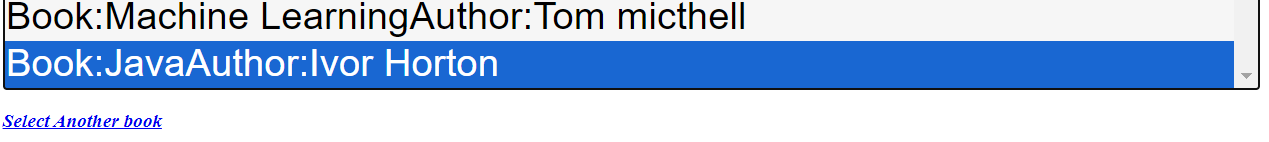
}

}

}

**Output:**

****

****

#### **D] Using Page Caching**

**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Practical\_10

{

public partial class WebForm1 : System.Web.UI.Page

{

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

{

Label1.Text = DateTime.Now.ToLongTimeString();

}

}

}

**ASPX**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Practical\_10.WebForm1" %>

<!DOCTYPE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<h3>Time in cache page</h3>

<asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="Large" Text="Time in cache page:"></asp:Label>

</div>

<p>

&nbsp;</p>

</form>

</body>

</html>

**Output:**

****

## **Practical 11 - Create and Deploy Web Service**

**Explanation:**

**A] Simple Web Service (Calculator)**

A simple web service for a calculator provides basic arithmetic operations like addition, subtraction, multiplication, and division over the web. Implemented using ASP.NET, it exposes methods as web service operations that can be consumed by clients over HTTP, allowing interoperability across different platforms and languages.

**Source:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Services;

namespace Practical\_11

{

/// <summary>

/// Summary description for WebService1

/// </summary>

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

[System.ComponentModel.ToolboxItem(false)]

// To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.

// [System.Web.Script.Services.ScriptService]

public class WebService1 : System.Web.Services.WebService

{

[WebMethod]

public int Add(int fn, int sn)

{

return fn + sn;

}

[WebMethod]

public int Subtract(int fn, int sn)

{

return fn - sn;

}

[WebMethod]

public int Multiply(int fn, int sn)

{

return fn \* sn;

}

[WebMethod]

public int Divide(int fn, int sn)

{

return fn / sn;

}

}

}

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm2.aspx.cs" Inherits="Practical\_11.WebForm2" %>

<!DOCTYPE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

</div>

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

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

<p>

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

<asp:TextBox ID="txtsn" runat="server" OnTextChanged="txtsn\_TextChanged"></asp:TextBox>

</p>

<p>

<asp:Label ID="Label3" runat="server" Text="Result"></asp:Label>

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

<asp:Label ID="lblres" runat="server" Text="[lblres]"></asp:Label>

</p>

<asp:Button ID="btnadd" runat="server" OnClick="btnadd\_Click" Text="Add" />

&nbsp;

<asp:Button ID="btnsub" runat="server" OnClick="btnsub\_Click" Text="Sub" />

&nbsp;

<asp:Button ID="btnmul" runat="server" OnClick="btnmul\_Click" Text="Mul" />

&nbsp;

<asp:Button ID="btndiv" runat="server" OnClick="btndiv\_Click" Text="Div" />

</form>

</body>

</html>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Practical\_11

{

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

{

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

{

}

protected void txtsn\_TextChanged(object sender, EventArgs e)

{

}

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

{

WebService1.CalculatorSoapClient client = new WebService1.CalculatorSoapClient();

int res = client.Add(Convert.ToInt32(txtfn.Text), (Convert.ToInt32(txtsn.Text)));

lblres.Text = res.ToString();

}

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

{

WebService1.CalculatorSoapClient client = new WebService1.CalculatorSoapClient();

int res = client.Subtract(Convert.ToInt32(txtfn.Text), Convert.ToInt32(txtsn.Text));

lblres.Text = res.ToString();

}

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

{

WebService1.CalculatorSoapClient client = new WebService1.CalculatorSoapClient();

int res = client.Multiply(Convert.ToInt32(txtfn.Text), Convert.ToInt32(txtsn.Text));

lblres.Text = res.ToString();

}

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

{

WebService1.CalculatorSoapClient client = new WebService1.CalculatorSoapClient();

int res = client.Divide(Convert.ToInt32(txtfn.Text), Convert.ToInt32(txtsn.Text));

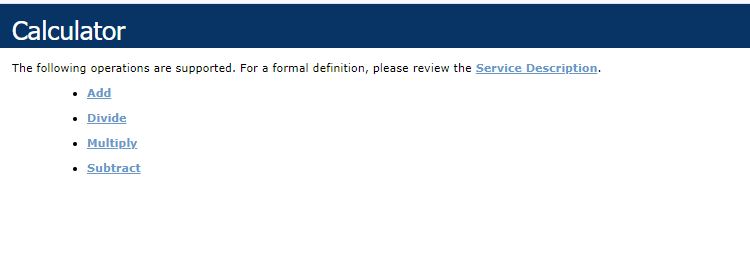
lblres.Text = res.ToString();

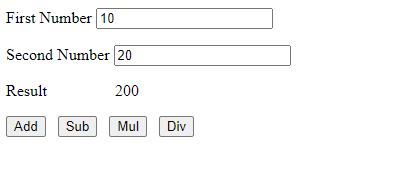
}

}

}

**Output:**

****

****

**B] Web Service with Database**

A web service with a database backend allows clients to perform CRUD operations on a database. For example, a product management service can let clients add, update, retrieve, and delete product information. This service uses ADO.NET or Entity Framework to interact with the database, ensuring data is dynamically managed and accessed.

**Source**

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Xml.Linq;

namespace Practical\_12\_B

{

public partial class WebForm1 : System.Web.UI.Page

{

ServiceReference1.Db\_serviceSoapClient serviceSoapClient;

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

{

if (!IsPostBack)

{

serviceSoapClient = new ServiceReference1.Db\_serviceSoapClient();

DataSet data = serviceSoapClient.Get\_Details();

GridView1.DataSource = data;

GridView1.DataBind();

}

}

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

{

serviceSoapClient = new ServiceReference1.Db\_serviceSoapClient();

DataSet data = serviceSoapClient.Get\_Product\_name(txtname.Text);

GridView2.DataSource = data;

GridView2.DataBind();

}

}

}

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Practical\_12\_B.WebForm1" %>

<!DOCTYPE html>

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

<head runat="server">

<title>Student Details</title>

</head>

<body>

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

<div>

<asp:GridView ID="GridView1" runat="server" BackColor="#99FF66" OnSelectedIndexChanged="GridView1\_SelectedIndexChanged">

</asp:GridView>

<br />

<br />

</div>

<asp:TextBox ID="txtname" runat="server" style="width: 128px"></asp:TextBox>

&nbsp;&nbsp;

<asp:Button ID="btnshow" runat="server" BackColor="#CC99FF" OnClick="btnshow\_Click" Text="Show" />

<p>

<asp:GridView ID="GridView2" runat="server" BackColor="#FFCC66">

</asp:GridView>

</p>

</form>

</body>

</html>

asmx.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Services;

namespace Practical\_12\_B

{

/// <summary>

/// Summary description for WebService1

/// </summary>

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

[System.ComponentModel.ToolboxItem(false)]

// To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.

// [System.Web.Script.Services.ScriptService]

public class WebService1 : System.Web.Services.WebService

{

SqlConnection myconn;

SqlCommand cmd;

SqlDataAdapter adpt;

DataSet myds;

[WebMethod]

public DataSet Get\_Details()

{

myconn = new SqlConnection();

myconn.ConnectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\exam.202STUD125\Documents\Stud.mdf;Integrated Security=True;Connect Timeout=30";

cmd = new SqlCommand("Select \* from dbo.Student\_Info", myconn);

adpt = new SqlDataAdapter(cmd);

myds = new DataSet();

adpt.Fill(myds);

return myds;

}

[WebMethod]

public DataSet Get\_Product\_name(string n)

{

myconn = new SqlConnection();

myconn.ConnectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\exam.202STUD125\Documents\Stud.mdf;Integrated Security=True;Connect Timeout=30";

cmd = new SqlCommand("Select \* from dbo.Student\_Info WHERE Name = @Name", myconn);

cmd.Parameters.AddWithValue("@Name", n);

adpt = new SqlDataAdapter(cmd);

myds = new DataSet();

adpt.Fill(myds);

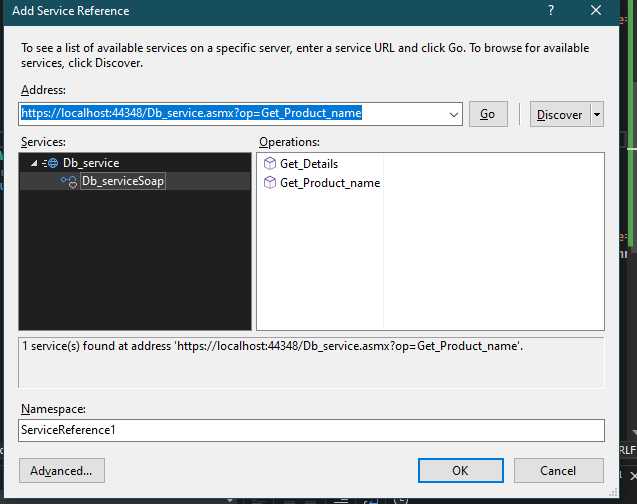
return myds;

}

}

}

**Output:**



**C] Windows Communication Foundation (WCF) Service**

WCF is a framework for building service-oriented applications. A WCF service can communicate over various protocols (HTTP, TCP, etc.) and supports multiple transport and message encoding options. It provides more advanced features than basic web services, such as security, reliability, and transaction support, making it suitable for complex and distributed enterprise applications.

**Source**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.ServiceModel.Web;

using System.Text;

namespace WcfService3

{

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the interface name "IService1" in both code and config file together.

[ServiceContract]

public interface IService1

{

[OperationContract]

string GetData(int value);

[OperationContract]

string Addtion(int value1, int value2);

[OperationContract]

CompositeType GetDataUsingDataContract(CompositeType composite);

// TODO: Add your service operations here

}

// Use a data contract as illustrated in the sample below to add composite types to service operations.

[DataContract]

public class CompositeType

{

bool boolValue = true;

string stringValue = "Hello ";

[DataMember]

public bool BoolValue

{

get { return boolValue; }

set { boolValue = value; }

}

[DataMember]

public string StringValue

{

get { return stringValue; }

set { stringValue = value; }

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.Text;

namespace WcfService3

{

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the interface name "IService2" in both code and config file together.

[ServiceContract]

public interface IService2

{

[OperationContract]

public double Add(double x, double y)

{

return x + y;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.ServiceModel.Web;

using System.Text;

namespace WcfService3

{// NOTE: You can use the "Rename" command on the "Refactor" menu to change the class name "Service1" in code, svc and config file together.

// NOTE: In order to launch WCF Test Client for testing this service, please select Service1.svc or Service1.svc.cs at the Solution Explorer and start debugging.

public class Service1 : IService1

{

public string GetData(int value)

{

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

}

public string Addtion(int value1, int value2)

{

return string.Format("Addition is {0}", (value1 + value2));

}

public string CalculateDays(int day, int month, int year)

{

DateTime date = new DateTime(year, month, day);

int diffdate = DateTime.Now.Subtract(date).Days;

diffdate = Math.Abs(diffdate);

return string.Format("Date is {0} Differnce is {1}", date.ToShortDateString(), diffdate);

}

public CompositeType GetDataUsingDataContract(CompositeType composite)

{

if (composite == null)

{

throw new ArgumentNullException("composite");

}

if (composite.BoolValue)

{

composite.StringValue += "Suffix";

}

return composite;

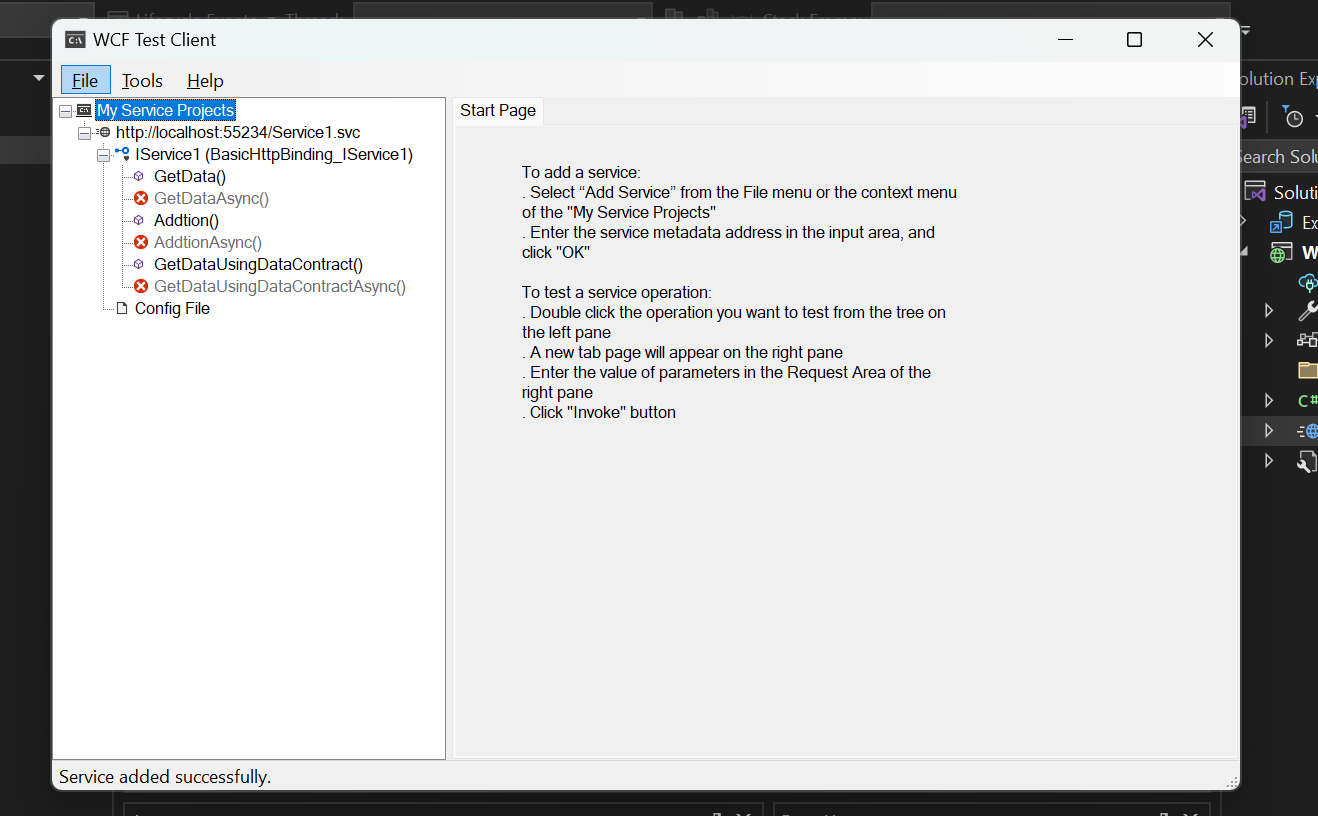
}

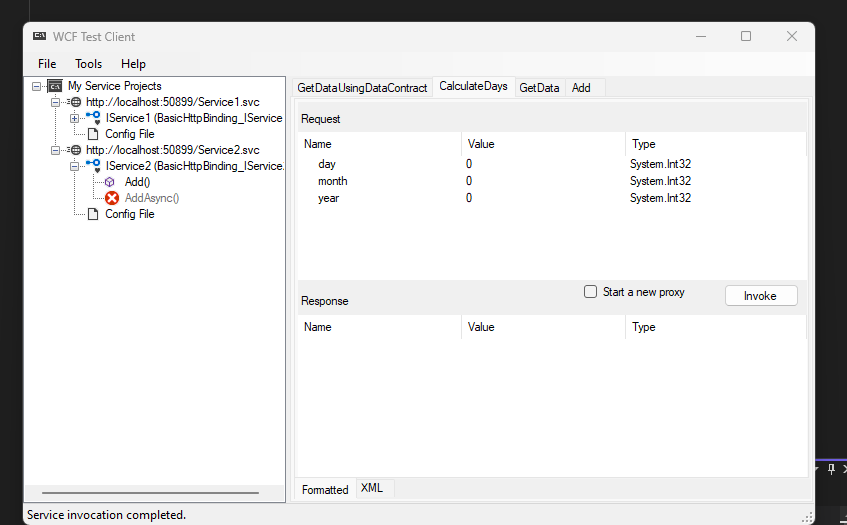
}

}

**Output:**

## 





## **Practical 12 - Program to demonstrate the implementation of User web controls**

User Web Controls, also known as User Controls, are reusable components in ASP.NET that encapsulate a piece of user interface and related functionality into a single unit. These controls are useful for modularizing and reusing UI components across multiple web pages in an ASP.NET application. User Controls have the extension .ascx and are similar to web forms but cannot be accessed directly via a URL.

**Source**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Practical\_12.WebForm1" %>

<%@ Register Src="~/WebUserControl1.ascx" TagName="WebUserControl1" TagPrefix="ctrl" %>

<!DOCTYPE html>

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

<head >

<title>User Control Example</title>

</head>

<body>

<div>

<ctrl:WebUserControl1 ID="WebUserControl1" runat="server" />

</div>

</body>

</html>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Practical\_12

{

public partial class WebForm1 : System.Web.UI.Page

{

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

{

}

}

}

**User Control**

<%@ Control Language="C#" AutoEventWireup="true" CodeBehind="WebUserControl1.ascx.cs" Inherits="Practical\_12.WebUserControl1" %>

<form runat="server">

<asp:Label ID="Label1" runat="server" Text="Name"></asp:Label> &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<asp:TextBox ID="TextBox1" runat="server" Height="34px" Width="143px"></asp:TextBox>

<br />

<br />

<asp:Label ID="Label2" runat="server" Text="City"></asp:Label> &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<asp:TextBox ID="TextBox2" runat="server" Height="37px" Width="144px"></asp:TextBox>

<br />

<br />

<asp:Button ID="Button1" runat="server" Text="Submit" Height="51px" Width="90px" OnClick="Button1\_Click" />

<br />

<br />

<asp:Label ID="Label7" runat="server" Text="Label"></asp:Label>

</form>

**CodeBehind**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Practical\_12

{

public partial class WebUserControl1 : System.Web.UI.UserControl

{

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

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

Label7.Text = TextBox1.Text;

}

}

}

**Output:**

****

## **Practical 13 - Design a web page using Asynchronous JavaScript and XML (AJAX)**

Asynchronous JavaScript and XML (AJAX) is a technique used in web development to create dynamic, interactive web pages. AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

**Source:**

<!DOCTYPE html>

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

<head runat="server">

<title></title>

</head>

<body>

<form id="form1" runat="server" style="height:100vh">

<div>

<h1><asp:Label ID="Label1" runat="server" Text="WEB TECH"/></h1>

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

<asp:Timer ID="Timer1" runat="server" Interval="6000" OnTick="Tick"></asp:Timer>

<asp:Timer ID="Timer2" runat="server" Interval="1000" OnTick="Timer2\_Tick"></asp:Timer>

<asp:UpdatePanel ID="UpdatePanel1" runat="server">

<ContentTemplate>

<asp:Label ID="Label2" runat="server" Text="Quotes"/></h1>

<br />

<br />

<asp:Label ID="Label3" runat="server" Text=""/></h1>

</ContentTemplate>

<Triggers>

<asp:AsyncPostBackTrigger ControlID="Timer2" EventName="Tick" />

<asp:AsyncPostBackTrigger ControlID="Timer1" EventName="Tick" />

</Triggers>

</asp:UpdatePanel>

</div>

</form>

</body>

</html>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace ajax

{

public partial class WebForm1 : System.Web.UI.Page

{

List<String> quotes = new List<String>();

Random random;

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

{

random = new Random();

quotes.Add("Hello");

quotes.Add("World!!");

Label2.Text = quotes[random.Next(0, quotes.Count)];

}

protected void Tick(object sender, EventArgs e)

{

Label2.Text = $"Quote : {quotes[random.Next(0, quotes.Count)]}";

}

protected void Timer2\_Tick(object sender, EventArgs e)

{

Label3.Text = DateTime.Now.ToString();

}

}

}

**Output:**



