

Roll No: 57

Exam Seat No:

**VIVEKANAND EDUCATION SOCIETY'S INSTITUTE
OF TECHNOLOGY**

Hashu Advani Memorial Complex, Collector's Colony, R. C.
Marg, Chembur, Mumbai – 400074. Contact No.
02261532532



Since 1962

CERTIFICATE

Certified that Mr./Miss Ajay Karthikesan Thevar of FYMCA
_____ has satisfactorily completed a course of the necessary
experiments in Advanced Web Technologies Lab under my
supervision in the Institute of Technology in the academic year
2022-2023.

Principal

Head of Department
(Dr. Shiv kumar goel)

External Examiner

Subject Teacher
(Miss. Bhavana Chaudhari)

Name of Student: Ajay Karthikesan			
Roll Number: 57		Assignment Number: 1	
Aim of Assignment: Design UI based applications using basic Windows forms Controls			
DOP: .29.3.23		DOS: 19.4.23	
CO Mapped: CO1	PO Mapped: PO3, PO5, PSO1, PSO2	Faculty Signature:	Marks:

Practical No. 1

Aim: Design UI based applications using basic Windows forms Controls

1. WAP in C# that ask the user to enter a month, a day and a two digit year. The program should then determine whether the month times a day is equal to the year. If so, it should display the message saying the date is magic. Otherwise not a magic
2. WAP to perform Money Conversion.
3. WAP To convert temperature from Fahrenheit to Celsius or vice versa
4. Create a Window application to calculate age of a person by providing input as birth date and current date .Current date and Birth date must be in long string format and display the age in terms of years.

Theory:

- .net framework:
 - .Net Framework is a software development platform developed by Microsoft for building and running Windows applications. The .Net framework consists of developer tools, programming languages, and libraries to build desktop and web applications. It is also used to build websites, web services, and games. The Microsoft .Net framework can be used to create both - Form based and Web-based applications.
- CLR:
 - CLR is the basic and Virtual Machine component of the .NET Framework. It is the run-time environment in the .NET Framework that runs the codes and helps in making the development process easier by providing the various services. Main components of CLR:
- Common Language Specification (CLS)
- Common Type System (CTS)
- Garbage Collection (GC)
- Just In – Time Compiler (JIT)
- Structure of C# programming:
 - C# programs consist of one or more files. Each file contains zero or more namespaces. A namespace contains types such as classes, structs, interfaces, enumerations, and delegates, or other namespaces.
- Namespace and its use in application:
 - A namespace is a declarative region that provides a scope to the identifiers (the names of types, functions, variables, etc.) inside it.
 - It is also referred as named group of classes having common features. The members of a namespace can be namespaces, interfaces, structures, and delegates.
 - Defining a Namespace: To define a namespace in C#, we will use the namespace keyword followed by the name of the namespace and curly braces containing the body of the namespace

Code:

File: MagicDateForm.cs

```
namespace _2Practical
```

```
{
    public partial class MagicDateForm : Form
    {
        public MagicDateForm()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void checkBtn_Click(object sender, EventArgs e)
        {
            int day = Convert.ToInt32(dayTb.Text);
            int month = Convert.ToInt32(monthTb.Text);
            int year = Convert.ToInt32(yearTb.Text);
            string result = "The entered date is ";
            if (year != (day * month))
            {
                result += "not ";
            }
            result += "a magic date";
            MessageBox.Show(result, "caption", MessageBoxButtons.OK,
                MessageBoxIcon.Information);
        }
    }
}
```

File: MoneyConverterForm.cs

```
using _2Practical.utility;
namespace _2Practical
{
    public partial class MoneyConverterForm : Form
    {
        public MoneyConverterForm()
        {
            InitializeComponent();
            inputCurrCbb.DataSource = Enum.GetNames(typeof(Currency));
            outputCurrCbb.DataSource = Enum.GetNames(typeof(Currency));
            outputCurrCbb.SelectedIndex = (int)Currency.INR;
        }
        private void updateOutputAmt()
        {
            Currency sourceCurr =
                (Currency)this.inputCurrCbb.SelectedIndex;
            Currency destCurr =
                (Currency)this.outputCurrCbb.SelectedIndex;
            try
```

```
        {
            double srcAmt =
Convert.ToDouble(this.inputAmtTxt.Text);
            outputAmtTxt.Text = MoneyConverter.convert(sourceCurr,
destCurr, srcAmt).ToString("0.##");
        }
        catch (System.FormatException)
        {
            outputAmtTxt.Text = "";
        }
    }
    private void updateInputAmt()
    {
        Currency sourceCurr =
(Currency)this.outputCurrCbb.SelectedIndex;
        Currency destCurr =
(Currency)this.inputCurrCbb.SelectedIndex;
        try
        {
            double srcAmt =
Convert.ToDouble(this.outputAmtTxt.Text);
            inputAmtTxt.Text = MoneyConverter.convert(sourceCurr,
destCurr, srcAmt).ToString("0.##");
        }
        catch (System.FormatException)
        {
            inputAmtTxt.Text = "";
        }
    }

    private void inputCurr_SelectedIndexChanged(object sender,
EventArgs e)
    {
        updateOutputAmt();
    }
    private void inputAmt_TextChanged(object sender, EventArgs e)
    {
        if (ActiveControl == outputAmtTxt) return;
        updateOutputAmt();
    }
    private void outputAmt_TextChanged(object sender, EventArgs e)
    {
        if (ActiveControl == inputAmtTxt) return;
        updateInputAmt();
    }
    private void outputCurr_SelectedIndexChanged(object sender,
EventArgs e)
    {
        updateInputAmt();
    }
}
```

```
    }  
  }  
}
```

File: TemperatureConverterForm.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 _2Practical  
{  
    public partial class TemperatureConverterForm : Form  
    {  
        public TemperatureConverterForm()  
        {  
            InitializeComponent();  
        }  
        private static string FahrenheitToCelsius(double f)  
        {  
            double c = ((f - 32) * 5) / 9;  
            return c.ToString("0.##");  
        }  
        private static string CelsiusToFahrenheit(double c)  
        {  
            double f = ((c * 9) / 5) + 32;  
            return f.ToString("0.##");  
        }  
        private void updateFhrnTxt()  
        {  
            try  
            {  
                double c = Convert.ToDouble(celsTxt.Text);  
                fhrnTxt.Text = CelsiusToFahrenheit(c);  
            }  
            catch (FormatException)  
            {  
                fhrnTxt.Text = "";  
            }  
        }  
        private void updateCelsTxt()  
        {  
            try  
            {  
                double f = Convert.ToDouble(fhrnTxt.Text);
```

```
        celsTxt.Text = FahrenheitToCelsius(f);
    }
    catch (FormatException)
    {
        celsTxt.Text = "";
    }
}

private void fhrnTxt_TextChanged(object sender, EventArgs e)
{
    if (ActiveControl == celsTxt) return;
    updateCelsTxt();
}

private void celsTxt_TextChanged(object sender, EventArgs e)
{
    if (ActiveControl == fhrnTxt) return;
    updateFhrnTxt();
}
}
```

File: utility/Currency.cs

```
namespace _2Practical.utility
{
    enum Currency
    {
        USD,
        INR,
        YEN,
        EUR
    }
}
```

File: MoneyConverter.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _2Practical.utility
{
    internal class MoneyConverter
    {
        private static double ToINR(Currency srcCurr, double srcAmt)
        {
            switch(srcCurr)
            {
                case Currency.USD: return srcAmt * 82.04;
                case Currency.YEN: return srcAmt * 0.62;
                case Currency.EUR: return srcAmt * 89.81;
            }
        }
    }
}
```

```
        default: return srcAmt;
    }
}
private static double INRTo(Currency destCurr,double srcAmt)
{
    switch (destCurr)
    {
        case Currency.USD: return srcAmt * 0.012;
        case Currency.YEN: return srcAmt * 1.60;
        case Currency.EUR: return srcAmt * 0.011;
        default: return srcAmt;
    }
}
public static double convert(Currency srcCurr,Currency
destCurr,double srcAmt)
{
    if (srcCurr == destCurr) return srcAmt;
    double inrsrcAmt=ToINR(srcCurr,srcAmt);
    return INRTo(destCurr,inrsrcAmt);
}
}
```

File: AgeCalculatorForm.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 _2Practical
{
    public partial class AgeCalculatorForm : Form
    {
        public AgeCalculatorForm()
        {
            InitializeComponent();
        }
        private void showResultMessageBox()
        {
            try
            {
                DateTime dob = DateTime.Parse(dobTxt.Text);
                TimeSpan age = DateTime.Now - dob;
            }
        }
    }
}
```

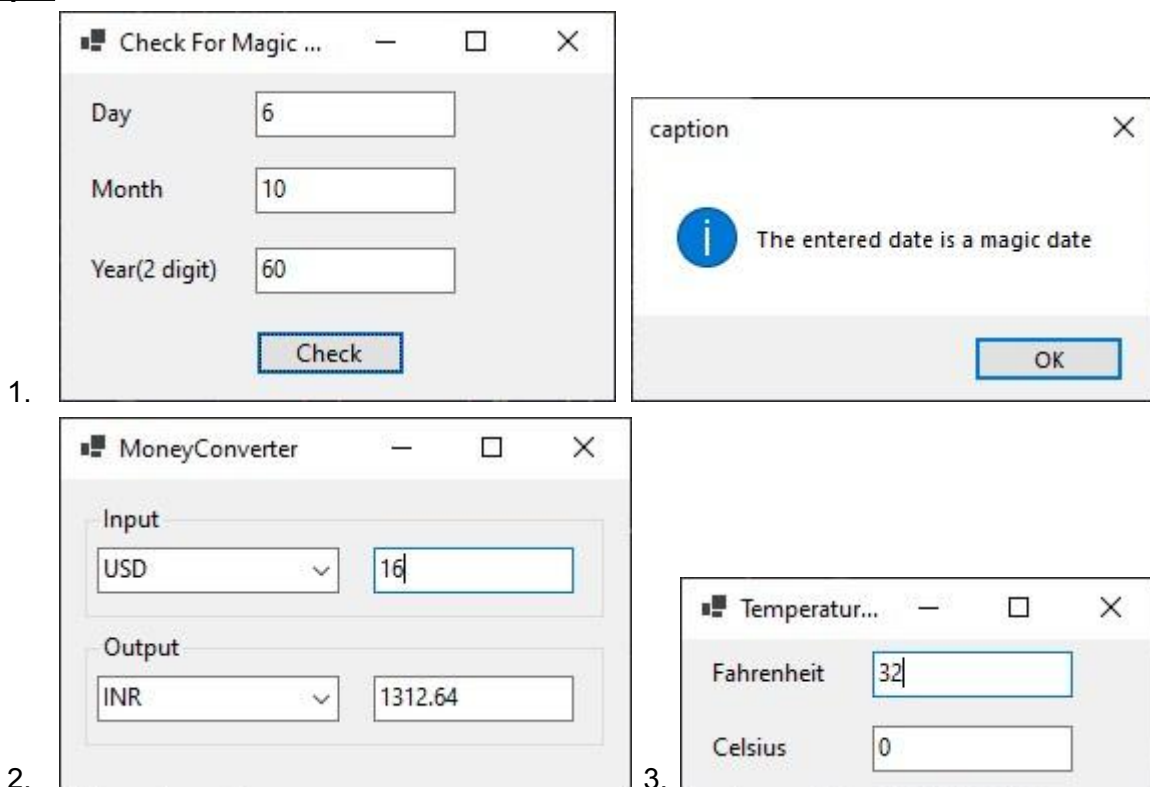


```

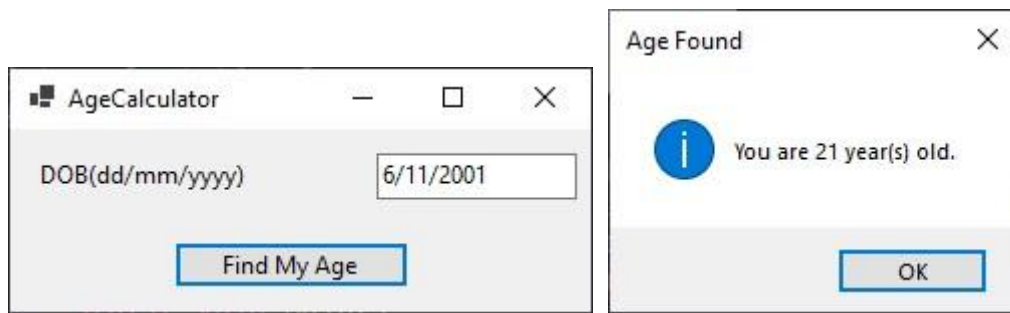
        MessageBox.Show($"You are
{Math.Round((age.TotalDays/365))} year(s) old.", "Age Found",
MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch
    {
        MessageBox.Show("Please Enter the date in the right
format", "Wrong Date Format", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
}

private void findAgeBtn_Click(object sender, EventArgs e)
{
    showResultMessageBox();
}
}
}

```

Output:

4.



Conclusion: I learnt how to design UI based applications using basic Windows forms Controls

Practical No. 2

Aim: Design Applications using Classes and Objects.

1. Write a program to declare a class "staff" having data members as name and post. Accept this data 5 staffs and display names of staff who are HOD.
2. Define a class "salary" which will contain member variable Basic, TA, DA, HRA. Write a program using Constructor with default values for DA and HRA and calculate the salary of the employee.

Theory:

1. Class: A Class is like an object constructor, or a "blueprint" for creating objects. It can have fields, methods, constructors etc.
2. Object: Object is an entity that has state and behavior. Here, state means data and behavior means functionality. Object is a runtime entity, it is created at runtime. Object is an instance of a class. All the members of the class can be accessed through object.
3. Array: An array is a collection of items stored at contiguous memory locations. Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.
4. Array of Object: In C#, arrays are actually objects, and not just addressable regions of contiguous memory as in C and C++. Object Array can store an element of various types in a single collection.
5. Constructor: Constructor is a special method which is invoked automatically at the time of object creation. It is used to initialize the data members of new object. The constructor has the same name as class. There can be two types of constructors in C#.
 - a. Default constructor.
 - b. Parameterized constructor
 - c. Copy constructor
 - d. Static constructor
6. Types of Constructor:
 - a. Default Constructor: A constructor without any parameters called default constructor. In this constructor every instance of the class will be initialized without any parameter values.
 - b. Parameterized constructor: A constructor with at least one parameter is called as parameterized constructor. In parameterized constructor we can initialize each instance of the class to different values.
 - c. Copy constructor: A parameterized constructor that contains a parameter of same class type is called as copy constructor. Main purpose of copy constructor is to initialize new instance to the values of an existing instance.
 - d. Static constructor: When we declared constructor as static it will be invoked only once for any number of instances of the class and it's during the creation of first instance of the class or the first reference to a static member in the class. Static constructor is used to initialize static fields of the class and to write the code that needs to be executed only once.

Code:

File: 1problem/StaffRegistrationForm.cs

```
using System;
using System.Collections;
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 _2APractical._1problem
{
    public partial class StaffRegistrationForm : Form
    {
        private uint _noOfStaffs = 0;
        private List<Staff> staffs = new();
        public StaffRegistrationForm()
        {
            InitializeComponent();
            postCbb.DataSource = Enum.GetValues(typeof(Post));
        }
        private void IncrementNoOfStaffs()
        {
            ++_noOfStaffs;
            noOfStaffs.Text = _noOfStaffs.ToString();
        }
        private void AddStaff()
        {
            Post post = (Post)postCbb.SelectedIndex;
            Staff staff = new Staff(nameTxt.Text, post);
            staffs.Add(staff);
            IncrementNoOfStaffs();
        }
        private void addStaffBtn_Click(object sender, EventArgs e)
        {
            AddStaff();
            nameTxt.Text = "";
        }
        private void ShowHODS(List<Staff> hods)
        {
            List<string> hodsNames = new();
            foreach (Staff staff in hods)
            {
                hodsNames.Add(staff.Name);
            }
            HODListForm hODListForm = new(hodsNames);
            hODListForm.Show();
        }
    }
}
```

```
    }
    private void showHodsBtn_Click(object sender, EventArgs e)
    {
        List<Staff> hods = staffs.FindAll(staff => staff.Post ==
Post.HOD);
        if (hods.Count == 0)
        {
            MessageBox.Show("No HODS are there.");
        }
        else
        {
            ShowHODS(hods);
        }
    }
}
```

File: 1problem/Staff.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _2APractical._1problem
{
    internal class Staff
    {
        public string Name { get; private set; }
        public Post Post { get; private set; }

        internal Staff(string name, Post post)
        {
            this.Name = name;
            this.Post = post;
        }
    }
}
```

File: 1problem/Post.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _2APractical._1problem
{
    enum Post
    {

```

```
        Poen,  
        HOD,  
        Principal  
    }  
}
```

File: 1problem/HODListForm.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 _2APractical._1problem  
{  
    public partial class HODListForm : Form  
    {  
        internal HODListForm(List<string> hods)  
        {  
            InitializeComponent();  
            hodsLbx.DataSource = hods;  
        }  
    }  
}
```

File: 2problem/Salary.cs

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace _2APractical._2problem  
{  
    internal class Salary  
    {  
        decimal basic;  
        byte ta;  
        byte da;  
        byte hra;  
        internal Salary(decimal basic, byte ta, byte da=20, byte hra=50)  
        {  
            this.basic = basic;  
            this.ta = ta;  
            this.da = da;  
            this.hra=hra;  
        }  
    }  
}
```

```

        private decimal PercentOfBasic(byte rate)
        {
            return (basic*rate)/100;
        }
        public decimal get()
        {
            return basic + PercentOfBasic(ta) + PercentOfBasic(da) +
PercentOfBasic(hra);
        }
    }
}

```

File: 2problem/SalaryCalculatorForm.cs

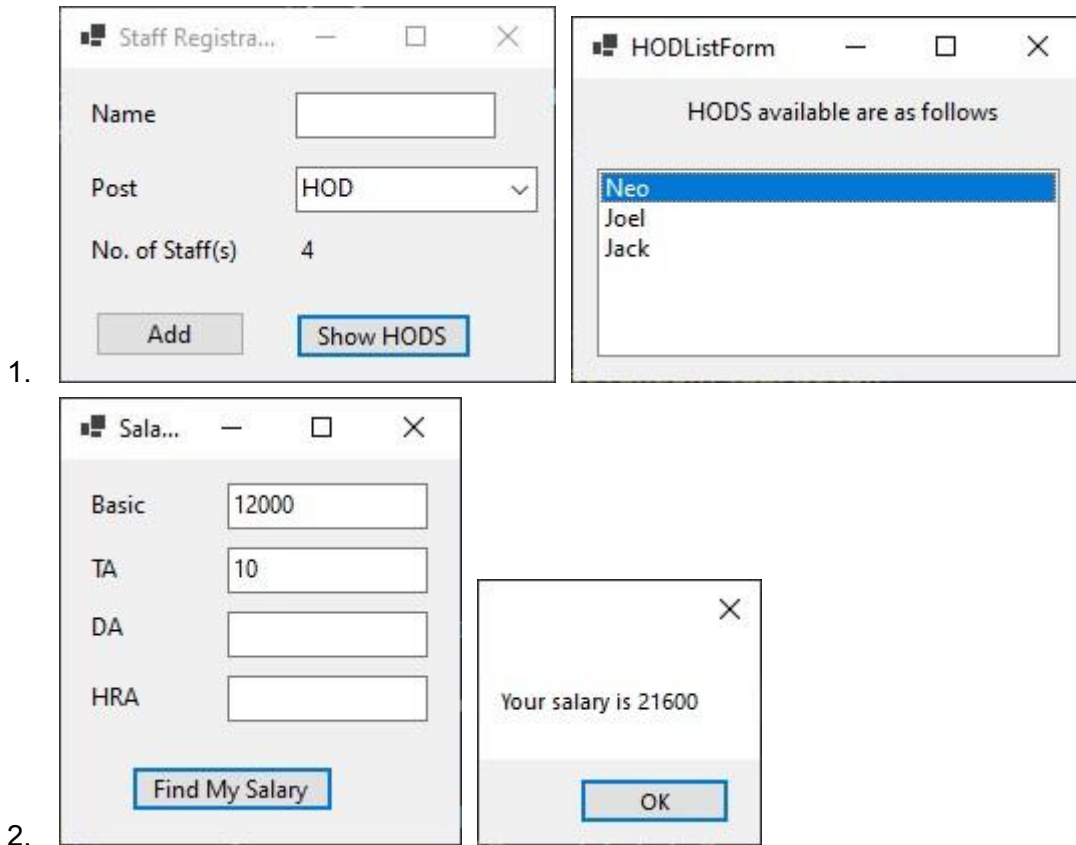
```

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 _2APractical._2problem
{
    public partial class SalaryCalculatorForm : Form
    {
        public SalaryCalculatorForm()
        {
            InitializeComponent();
        }
        private void ShowSalary()
        {
            Salary salary;
            if(daTxt.Text==" " || hraTxt.Text == " ")
            {
                salary = new(Convert.ToDecimal(basicTxt.Text),
Convert.ToByte(taTxt.Text));
            }
            else
            {
                salary = new(Convert.ToDecimal(basicTxt.Text),
Convert.ToByte(taTxt.Text),
Convert.ToByte(daTxt.Text),
Convert.ToByte(hraTxt.Text));
            }
            MessageBox.Show($"Your salary is
{salary.get().ToString("0.##")}");
        }
        private void findSalBtn_Click(object sender, EventArgs e)

```

```
{  
    ShowSalary();  
}  
}
```

Output:

Conclusion: I learnt how to design Applications using Classes and Objects.

Name of Student: Ajay Karthikesan			
Roll Number: 57		Assignment Number: 3	
Aim of Assignment: Design Applications using Inheritance and Abstract Classes.			
DOP: 12.3.23		DOS: 19.4.23	
CO Mapped: CO1	PO Mapped: PO3, PO5, PSO1, PSO2	Faculty Signature:	Marks:

Practical No. 3

Aim: Design Applications using Inheritance and Abstract Classes.

1. Write a program to implement multilevel inheritance from following figure. Accept and display data for one student.
2. Program to calculate to find the area of various shapes: Rectangle, Circle, Ellipse, Square and Triangle using abstract class and abstract method.

Theory:

- Inheritance: Inheritance is a process in which one object acquires all the properties and behaviors of its parent object automatically.
 - Using Inheritance we can reuse, extend or modify the attributes and behaviors which are defined in other class. The class which inherits the members of another class is called derived class and the class whose members are inherited is called base class.
- Types of Inheritance:
 - Single Inheritance
 - Multilevel Inheritance
 - Hierarchical Inheritance
 - Abstraction Classes
- Data abstraction is the process of hiding certain details and showing only essential information to the user. Abstraction can be achieved with either abstract classes or interfaces. The abstract keyword is used for classes and methods.
- Abstract Classes and Abstract Method: An abstract class is an incomplete class or special class we can't be instantiated.
 - The purpose of an abstract class is to provide a blueprint for derived classes and set some rules what the derived classes must implement when they inherit an abstract class. We can use an abstract class as a base class and all derived classes must implement abstract definitions.
 - An abstract method must be implemented in all non-abstract classes using the override keyword. After overriding the abstract method is in the non-Abstract class. We can derive this class in another class and again we can override the same abstract method with it.

Code:

File: 1problem/Result.cs

```
namespace _3Practical._1problem
{
    internal class Result : Test
    {
        int _total;
        public int Total { get => _total; set => _total = value; }
        internal Result(int rollNo, string name, int marks1, int
marks2) : base(rollNo, name, marks1, marks2)
        {
            Total = marks1 + marks2;
        }
        public void ShowInfo()
```

```
        {
            Console.WriteLine($"Student's Info:-\nRoll No.: {RollNo},
Name: {Name}, Marks 1: {Marks1}, Marks 2: {Marks2}, Total: {Total}");
        }
        public static Result Get()
        {
            Console.WriteLine("Enter Student's info:-");
            int rollNo = Utility.GetInt("Roll No. ");
            string name = Utility.GetString("Name: ");
            int marks1 = Utility.GetInt("Marks1: ");
            int marks2 = Utility.GetInt("Marks2: ");
            return new(rollNo, name, marks1, marks2);
        }
    }
}
```

File: 1problem/Student.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _3Practical._1problem
{
    internal class Student
    {
        private int _rollNo;
        private string _name = string.Empty;

        public int RollNo
        {
            get { return _rollNo; }
            set { _rollNo = value; }
        }
        public string Name
        {
            get { return _name; }
            set { _name = value; }
        }
        internal Student(int rollNo, string name)
        {
            RollNo = rollNo;
            Name = name;
        }
    }
}
```

File: Test.cs

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _3Practical._1problem
{
    internal class Test : Student
    {
        private int _marks1, _marks2;
        public int Marks1 { get => _marks1; set => _marks1 = value; }
        public int Marks2 { get => _marks2; set => _marks2 = value; }
        internal Test(int rollNo, string name, int marks1, int marks2)
: base(rollNo, name)
        {
            Marks1 = marks1;
            Marks2 = marks2;
        }
    }
}
```

File: 2problem/Circle.cs

```
namespace _3Practical._2problem
{
    internal class Circle : Shape
    {
        private readonly double _radius;
        public override double Area
        {
            get { return Math.PI * Math.Pow(_radius, 2); }
        }
        public Circle(double radius)
        {
            _radius = radius;
        }
        public static Circle Get()
        {
            Console.WriteLine("Enter Circle's info.");
            double radius = Utility.GetDouble("Radius: ");
            return new(radius);
        }
    }
}
```

File: 2problem/Ellipse.cs

```
namespace _3Practical._2problem
{
    internal class Ellipse : Shape
    {
        private readonly double _majAxisLength;
```

```
        private readonly double _minAxisLength;
        private double SemiMajorAxisLength { get { return
_majAxisLength / 2; } }
        private double SemiMinorAxisLength { get { return
_minAxisLength / 2; } }
        public override double Area
        {
            get
            {
                return Math.PI * SemiMajorAxisLength *
SemiMinorAxisLength;
            }
        }
        public Ellipse(double majAxisLength, double minAxisLength)
        {
            _majAxisLength = majAxisLength;
            _minAxisLength = minAxisLength;
        }
        public static Ellipse Get()
        {
            Console.WriteLine("Enter Ellipse's info.");
            double majAxisLength = Utility.GetDouble("Major Axis
Length: ");
            double minAxisLength = Utility.GetDouble("Minor Axis
Length: ");
            return new(majAxisLength, minAxisLength);
        }
    }
}
```

File: 2problem/Rectangle.cs

```
namespace _3Practical._2problem
{
    internal class Rectangle : Shape
    {
        private readonly double _length;
        private readonly double _breadth;
        public override double Area
        {
            get { return _length * _breadth; }
        }
        Rectangle(double length, double breadth)
        {
            _length = length;
            _breadth = breadth;
        }
        public static Rectangle Get()
        {
            Console.WriteLine("Enter Rectangle's info.");
            double length = Utility.GetDouble("Length: ");
```

```
        double breadth = Utility.GetDouble("Breadth: ");
        return new(length, breadth);
    }
}
```

File: Shape.cs

```
namespace _3Practical._2problem
{
    internal abstract class Shape
    {
        abstract public double Area { get; }
    }
}
```

File: Triangle.cs

```
namespace _3Practical._2problem
{
    internal class Triangle : Shape
    {
        private readonly double _base;
        private readonly double _height;
        public override double Area
        {
            get { return 0.5 * _base * _height; }
        }
        Triangle(double @base, double height)
        {
            _base = @base;
            _height = height;
        }
        public static Triangle Get()
        {
            Console.WriteLine("Enter Triangle's info");
            double @base = Utility.GetDouble("Base: ");
            double height = Utility.GetDouble("Height: ");
            return new(@base, height);
        }
    }
}
```

File: Utility.cs

```
namespace _3Practical
{
    internal class Utility
    {
        public static int GetInt(string message)
        {
            while (true)
            {
                Console.Write(message);
                try
```

```
        {
            return Convert.ToInt32(Console.ReadLine());
        }
        catch (FormatException)
        {
            Console.WriteLine("Please enter a valid integer.");
            continue;
        }
    }
}

public static double GetDouble(string message)
{
    while (true)
    {
        Console.Write(message);
        try
        {
            return Convert.ToDouble(Console.ReadLine());
        }
        catch (FormatException)
        {
            Console.WriteLine("Please enter a valid real
number.");
            continue;
        }
    }
}

public static string GetString(string message)
{
    Console.Write(message);
    return Console.ReadLine() ?? string.Empty;
}
}
```

File: Demo.cs

```
using _3Practical._1problem;
using _3Practical._2problem;

namespace _3Practical
{
    internal class Demo
    {
        internal static class Problem1
        {
            public static void Run()
            {
                Console.WriteLine("Running Problem No.1 Demo...");
                Result.Get().ShowInfo();
            }
        }
    }
}
```

```
    }  
    internal static class Problem2  
    {  
        public static void Run()  
        {  
            Console.WriteLine("Running Problem No.2 Demo...");  
            Console.WriteLine("Area of Circle: " +  
Circle.Get().Area);  
            Console.WriteLine("Area of Ellipse: " +  
Ellipse.Get().Area);  
            Console.WriteLine("Area of Rectangle: " +  
Rectangle.Get().Area);  
            Console.WriteLine("Area of Triangle: " +  
Triangle.Get().Area);  
        }  
    }  
}
```

File: Program.cs

```
using _3Practical;
```

```
Demo.Problem1.Run();
```

```
Demo.Problem2.Run();
```

Output:

```
Running Problem No.1 Demo...  
Enter Student's info:-  
Roll No.: 57  
Name: Ajay Karthikesan  
Marks1: 99  
Marks2: 100  
Student's Info:-  
Roll No.: 57, Name: Ajay Karthikesan, Marks 1: 99, Marks 2: 100, Total: 199  
Running Problem No.2 Demo...  
Enter Circle's info.  
Radius: 12  
Area of Circle: 452.3893421169302  
Enter Ellipse's info.  
Major Axis Length: 10  
Minor Axis Length: 5  
Area of Ellipse: 39.269908169872416  
Enter Rectangle's info.  
Length: 9  
Breadth: 2  
Area of Rectangle: 18  
Enter Triangle's info  
Base: 12  
Height: 3  
Area of Triangle: 18
```

Conclusion:

I learnt how to use to design Applications using Inheritance and Abstract Classes.

Name of Student: Ajay Karthikesan			
Roll Number: 57		Assignment Number: 4	
Aim of Assignment: Design an online registration form using HTML Controls and validation controls			
DOP: 19.4.23		DOS: 26.4.23	
CO Mapped: CO1	PO Mapped: PO3, PO5, PSO1, PSO2	Faculty Signature:	Marks:

Practical No. 1

Aim:

Theory:

There are Six Servers as well as client-side validation controls in ASP.Net

- RequiredFieldValidator control:
 - This control ensures that the control it is used for validation is not empty when the form is submitted. In other words suppose there is one Text Box control and you have used a RequiredFieldValidator to validate that text box; then before submitting the data on the server it checks if the text box is not empty.
- RangeValidator:
 - Checks that the value of the associated control is within a specified range. The value and the range can be numerical, a date or a string. In other words suppose there is one text box and you want to allow only 10 digits or any strings with a specified range using RangeValidator then before submitting the data on the server it ensure that the value is within a specified range.
- CompareValidator:
 - Checks that the value of the associated control matches a specified comparison (less than, greater than, and so on) against another constant value or control.
- RegularExpressionValidator:
 - Checks if the value of the control it has to validate matches the specified regular expression.
- CustomValidator:
 - Allows specification of any client-side JavaScript validation routine and its serverside counterpart to perform your own custom validation logic.
- ValidationSummary:
 - Shows a summary with the error messages for each failed validator on the page (or in a pop-up message box)
- ASP.NET Master Pages: ASP.NET master pages allow you to create a consistent layout for the pages in your application. A single master page defines the look and feel and standard behavior that you want for all of the pages (or a group of pages) in your application. You can then create individual content pages that contain the content you want to display. When users request the content pages, they merge with the master page to produce output that combines the layout of the master page with the content from the content page. Master pages actually consist of two pieces, the master page itself and one or more content pages.
- Master Pages: A master page is an ASP.NET file with the extension .master (for example, MySite.master) with a predefined layout that can include static text, HTML elements, and server controls. The master page is identified by a special @Master directive that replaces the @Page directive that is used for ordinary .aspx pages. Replaceable Content Placeholders In addition to static text and controls that will appear on all pages, the master page also includes one or more ContentPlaceholder

controls. These placeholder controls define regions where replaceable content will appear. In turn, the replaceable content is defined in content pages.

- **Content Pages:** You define the content for the master pages placeholder controls by creating individual content pages, which are ASP.NET pages (.aspx files and, optionally, code-behind files) that are bound to a specific master page. The binding is established in the content page's @Page directive by including a MasterPageFile attribute that points to the master page to be used. In the content page, you create the content by adding Content controls and mapping them to ContentPlaceHolder controls on the master page.
- **Advantages of Master Pages:**
 - Master pages provide functionality that developers have traditionally created by copying existing code, text, and control elements repeatedly; using framesets; using include files for common elements; using ASP.NET user controls; and so on.
- **Advantages of master pages include the following:**
 - They allow you to centralize the common functionality of your pages so that you can make updates in just one place.
 - They make it easy to create one set of controls and code and apply the results to a set of pages. For example, you can use controls on the master page to create a menu that applies to all pages.
 - They give you fine-grained control over the layout of the final page by allowing you to control how the placeholder controls are rendered.
 - They provide an object model that allows you to customize the master page from individual content pages.

Code:

File: success-page.aspx.cs

```
namespace Practical4
{
    public partial class SuccessPage : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            nameLbl.Text = Request.QueryString["name"];
            emailLbl.Text = Request.QueryString["email"];
        }
    }
}
```

File: success-page.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/master.Master"
AutoEventWireup="true" CodeBehind="success-page.aspx.cs"
Inherits="Practical4.SuccessPage" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
runat="server">
    <label>You've Registered Successfully</label><br />
    <label>Name: </label>
```

```
<asp:Label ID="nameLbl" runat="server" Text="Label"></asp:Label><br />
<label>Email ID: </label>
<asp:Label ID="emailLbl" runat="server"
Text="Label"></asp:Label><br />
</asp:Content>
```

File: master.Master

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeBehind="master.master.cs" Inherits="Practical4.master" %>
```

```
<!DOCTYPE html>
```

```
<html>
<head runat="server">
    <title></title>
    <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <header>
        <h1>Nim Inc</h1>
    </header>
    <main>
        <asp:ContentPlaceHolder ID="ContentPlaceHolder1"
runat="server">
        </asp:ContentPlaceHolder>
    </main>
    <footer>
        Website and all its contents belong to Nim Inc.<br />
        Author: Ajay Karthikesan, Roll No.: 57
    </footer>
</body>
</html>
```

File: index.aspx.cs

```
namespace Practical4
{
    public partial class index : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void submitBtn_Click(object sender, EventArgs e)
        {
            if (IsValid)
            {
```

```

Response.Redirect("success-page.aspx?name="+nameLbl.Text+"&email="+emailLbl.Text);
    }
}
}
}

```

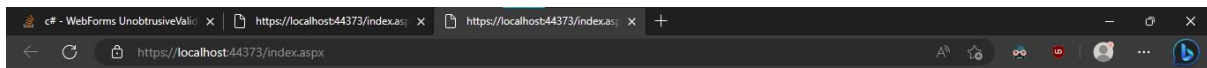
File: index.aspx

```

<%@ Page Title="" Language="C#" MasterPageFile="~/master.Master"
AutoEventWireup="true" CodeBehind="index.aspx.cs"
Inherits="Practical4.index" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
runat="server">
    <form runat="server">
        <label>Name <asp:TextBox ID="nameLbl"
runat="server"></asp:TextBox></label><asp:RequiredFieldValidator
ID="RequiredFieldValidator2" runat="server" ErrorMessage="This is a
required field"
ControlToValidate="nameLbl"></asp:RequiredFieldValidator><br />
        <label>Email ID <asp:TextBox ID="emailLbl"
runat="server"></asp:TextBox></label><asp:RequiredFieldValidator
ID="RequiredFieldValidator1" runat="server" ErrorMessage="This a
required field"
ControlToValidate="emailLbl"></asp:RequiredFieldValidator>
        <asp:RegularExpressionValidator
ID="RegularExpressionValidator1" runat="server" ErrorMessage="Please
enter a valid email" ControlToValidate="emailLbl"
ValidationExpression="\w+([-+.']\w+)*@\w+([-.'\w+)*\.\w+([-.'\w+)*"></a
sp:RegularExpressionValidator><br />
        <asp:Button ID="submitBtn" runat="server" Text="Submit"
OnClick="submitBtn_Click" />
    </form>
</asp:Content>

```

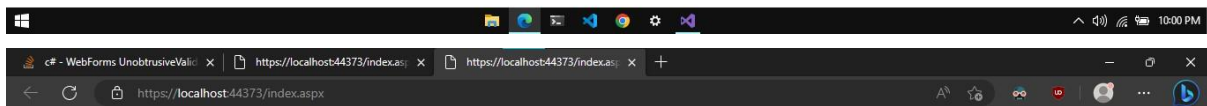
Output:



Nim Inc

Name This is a required field
Email ID 281 Please enter a valid email

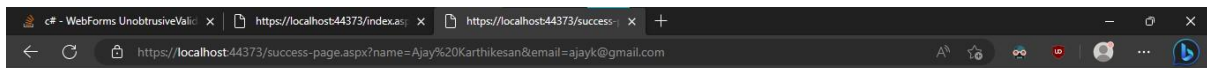
Website and all its contents belong to Nim Inc.
Author: Ajay Karthikesan, Roll No.: 57



Nim Inc

Name Ajay Karthikesan
Email ID ajayk@gmail.com

Website and all its contents belong to Nim Inc.
Author: Ajay Karthikesan, Roll No.: 57



Nim Inc

You've Registered Successfully
Name: Ajay Karthikesan
Email ID: ajayk@gmail.com
Website and all its contents belong to Nim Inc.
Author: Ajay Karthikesan, Roll No.: 57

Conclusion:

I learnt how to design an online registration form using HTML Controls and validation controls.

Name of Student: Ajay Karthikesan			
Roll Number: 57		Assignment Number: 5	
Aim of Assignment: Design a webpage to demonstrate connection to database through ADO.Net			
DOP: 24.4.23		DOS: 7.5.23	
CO Mapped: CO2	PO Mapped: PO3, PO5, PSO1, PSO2	Faculty Signature:	Marks:

Practical No. 5

Aim: Design a webpage to demonstrate connection to database through ADO.Net

Theory:

ADO.Net:

ADO is a rich set of classes, interfaces, structures, and enumerated types that manage data access from various types of data stores.

ADO.NET provides mainly the following two types of architectures:

Connected Architecture

Disconnected Architecture

Connected Architecture:

In the connected architecture, connection with a data source is kept open constantly for data access as well as data manipulation operations.

The ADO.NET Connected architecture considers mainly three types of objects.

SqlConnection con;

SqlCommand cmd;

SqlDataReader dr;

Disconnected Architecture:

Disconnected is the main feature of the .NET framework.

ADO.NET contains various classes that support this architecture. The .NET application does not always stay connected with the database. The classes are designed in a way that they automatically open and close the connection. The data is stored client-side and is updated in the database whenever required.

The ADO.NET Disconnected architecture considers primarily the following types of objects:

- DataSet ds;
- SqlDataAdapter da;
- SqlConnection con;
- SqlCommandBuilder bldr;

1. Connection Object and Connection string

Connection Object

1. One of the first ADO.NET objects is the connection object, that allows you to establish a connection to a data source.
2. The connection objects have the methods for opening and closing connections, for beginning a transaction of data.
3. The .Net Framework provides two types of connection classes: The sqlconnection object, that is designed specially to connect to Microsoft SQL Server and the

OleDbConnection object, that is designed to provide connection to a wide range of databases, such as Microsoft Access and Oracle.

4. A connection is required to interact with the database. A Connection object helps to identify the database server name, user name and password to connect to the database. The Connection object is used by commands on the database.
5. A Connection object has the responsibility of establishing a connection with the data store.

2. Command Object

A Command object executes SQL statements on the database. These SQL statements can be SELECT, INSERT, UPDATE, or DELETE. It uses a connection object to perform these actions on the database.

A Connection object specifies the type of interaction to perform with the database, like SELECT, INSERT, UPDATE, or DELETE.

A Command object is used to perform various types of operations, like SELECT, INSERT, UPDATE, or DELETE on the database.

- SELECT

```
cmd = new SqlCommand("select * from Employee", con);
```

- INSERT

```
cmd = new SqlCommand("INSERT INTO Employee(Emp_ID, Emp_Name) VALUES (' + aa + ',' + bb + '');" , con);
```

- UPDATE

```
SqlCommand cmd = new SqlCommand("UPDATE Employee SET Emp_ID = ' + aa + ' , Emp_Name = ' + bb + ' WHERE Emp_ID = ' + aa + ';" , con);
```

- DELETE

```
cmd = new SqlCommand("DELETE FROM Employee where Emp_ID = ' + aa + ';" , con);
```

- A Command object exposes several execute methods like:

1. ExecuteScalar()

Executes the query, and returns the first column of the first row in the result set returned by the query. Extra columns or rows are ignored.

2. ExecuteReader()

Display all columns and all rows in the client-side environment. In other words, we can say that they display datatables client-side.

3. ExecuteNonQuery()

Something is done by the database but nothing is returned by the database.

3. Data Reader Object

A DataReader object is used to obtain the results of a SELECT statement from a command object. For performance reasons, the data returned from a data reader is a forward-only stream of data. This means that the data can be accessed from the stream in a sequential manner. This is good for speed, but if data needs to be manipulated then a dataset is a better object to work with.

4. Data Adapter Object

A Data Adapter represents a set of data commands and a database connection to fill the dataset and update a SQL Server database.

A Data Adapter contains a set of data commands and a database connection to fill the dataset and update a SQL Server database. Data Adapters form the bridge between a data source and a dataset.

Data Adapters are designed depending on the specific data source. The following table shows the Data Adapter classes with their data source.

Data Source

A Data Adapter object accesses data in a disconnected mode. Its object contains a reference to a connection object.

It is designed in a way that implicitly opens and closes the connection whenever required.

It maintains the data in a DataSet object. The user can read the data if required from the dataset and write back the changes in a single batch to the database. Additionally, the Data Adapter contains a command object reference for SELECT, INSERT, UPDATE, and DELETE operations on the data objects and a data source.

A Data Adapter supports mainly the following two methods:

- Fill ():

The Fill method populates a dataset or a data table object with data from the database. It retrieves rows from the data source

using the SELECT statement specified by an associated select command property.

The Fill method leaves the connection in the same state as it encountered it before populating the data. If subsequent calls to the method for refreshing the data are required then the primary key information should be present.

- Update ()

The Update method commits the changes back to the database. It also analyzes the Row State of each record in the DataSet and calls the appropriate INSERT, UPDATE, and DELETE

statements. A Data Adapter object is formed between a disconnected ADO.NET object and a data source.

5. DataSet Object

In the disconnected scenario, the data retrieved from the database is stored in a local buffer called DataSet. It is explicitly designed to access data from any data source. This class is defined in the System.Data namespace.

A Data Set object is an in-memory representation of the data. It is specially designed to manage data in memory and to support disconnected operations on data.

A Data Set is a collection of DataTable and DataRelations. Each DataTable is a collection of DataColumn, DataRows, and Constraints.

6. Command Builder Object

Automatically generates insert, update, delete queries using the SelectCommand property of a DataAdapter.

A Command Builder Object is used to build commands for data modification from objects based on a single table query.

CommandBuilders are designed depending on the specific data source.

Code:

File: WebForm1.aspx

```
<%@ 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></title>
</head>
<body>
  <form id="form1" runat="server">
    <article>
      <h2>Original Order of table games</h2>
      <asp:GridView ID="GridView1" runat="server"
DataSourceID="SqlDataSource1" AutoGenerateColumns="False"
DataKeyNames="id">
        <Columns>
          <asp:BoundField DataField="id" HeaderText="id"
ReadOnly="True" SortExpression="id"></asp:BoundField>
          <asp:BoundField DataField="name" HeaderText="name"
SortExpression="name"></asp:BoundField>
          <asp:CheckBoxField DataField="is_outdoor"
HeaderText="is_outdoor"
SortExpression="is_outdoor"></asp:CheckBoxField>
```

```

        <asp:BoundField DataField="pcount"
HeaderText="pcount" SortExpression="pcount"></asp:BoundField>
    </Columns>
</asp:GridView>
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString='<%%$ ConnectionStrings:collGamesOgConnectionString %>'
SelectCommand="SELECT * FROM [ajayk57_games]"></asp:SqlDataSource>
</article>
<article>
    <h2>Table games ordered by player count(ascending)</h2>
    <asp:GridView ID="GridView2" runat="server"
DataSourceID="SqlDataSource2" AutoGenerateColumns="False"
DataKeyNames="id">
        <Columns>
            <asp:BoundField DataField="id" HeaderText="id"
ReadOnly="True" SortExpression="id"></asp:BoundField>
            <asp:BoundField DataField="name" HeaderText="name"
SortExpression="name"></asp:BoundField>
            <asp:CheckBoxField DataField="is_outdoor"
HeaderText="is_outdoor"
SortExpression="is_outdoor"></asp:CheckBoxField>
            <asp:BoundField DataField="pcount"
HeaderText="pcount" SortExpression="pcount"></asp:BoundField>
        </Columns>
    </asp:GridView>
    <asp:SqlDataSource ID="SqlDataSource2" runat="server"
ConnectionString='<%%$ ConnectionStrings:collGamesOgConnectionString %>'
SelectCommand="SELECT * FROM [ajayk57_games] ORDER BY
[pcount]"></asp:SqlDataSource>
</article>

</form>
<footer>
    Author: Ajay Karthikesan, Roll No. : 57
</footer>
</body>
</html>

```

Output:

Add Connection ? X

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:
Microsoft SQL Server (SqlClient) Change...

Server name:
MCAB12-26\SQLEXPRESS Refresh

Log on to the server

Authentication: Windows Authentication

User name:

Password:

☐ Save my password

Connect to a database

☒ Select or enter a database name:
ajay_k_57

☐ Attach a database file:
 Browse...

Logical name:

Advanced...

Test Connection OK Cancel

Original Order of table games

id	name	is_outdoor	pcount
1	Cricket	<input checked="" type="checkbox"/>	11
2	Football	<input checked="" type="checkbox"/>	11
3	Carrom	<input type="checkbox"/>	4

Table games ordered by player count(ascending)

id	name	is_outdoor	pcount
3	Carrom	<input type="checkbox"/>	4
1	Cricket	<input checked="" type="checkbox"/>	11
2	Football	<input checked="" type="checkbox"/>	11

Author: Ajay Karthikesan, Roll No. : 57

Conclusion:

Successfully designed a webpage to demonstrate connection to database through ADO.Net

Name of Student: Ajay Karthikesan			
Roll Number: 57		Assignment Number: 6	
Aim of Assignment: Designing webpage to demonstrate use of data bound controls, working of simple stored procedure and parameterized stored procedure.			
DOP: 22.5.23		DOS: 29.5.23	
CO Mapped: CO2	PO Mapped: PO3, PO5, PSO1, PSO2	Faculty Signature:	Marks:

Practical No. 6

Aim: Designing webpage to demonstrate use of data bound controls, working of simple stored procedure and parameterized stored procedure.

Q.1) Create a webpage that demonstrates the use of data bound controls of ASP.NET.

Q.2) Design a webpage to demonstrate the working of a simple stored procedure.

Q.3) Design a webpage to demonstrate the working of parameterized stored procedure.

Theory:

Data Bound Controls:

Databound controls are used to display data to the end-user within the web applications and using databound controls allows you to manipulate the data within the web applications very easily.

Databound controls are bound to the DataSource property.

Databound controls are composite controls that combine other ASP.NET Controls like Text boxes, Radio buttons, Buttons and so on.

Frequently used Databound controls:

- Repeater
- List View
- DataList
- GridView
- Form View

Repeater

- Repeater controls is a Databound control to just display data in the web application, using this we cannot manipulate the data; in other words, a Repeater is a read-only control.

- Repeater is very light-weight and faster to display data compared with other controls, so whenever you just want to display a repeated list of items then use

Repeater Control.

- Repeater control works by repeating using the data source.
- Repeater Control appearance is controlled by its templates.

ListView:

The ListView control displays columns and rows of data and allows sorting and paging. It is by far the most popular data display control, and is ideal for understanding how data display controls interact with data retrieval controls and code.

DataList:

DataList is a Databound control to display and manipulate data in a web application. It is a composite control that can combine other ASP.Net controls and it is present in the form. The

DataList appearance is controlled by its template fields.

The following template fields are supported by the DataList control:

- **ItemTemplate:** It specifies the Items present in the Datasource, it renders itself in the browser as many rows present in the data source collection.
- **EditItemTemplate:** Used to provide edit permissions to the user.
- **HeaderTemplate:** Used to display header text to the data source collection.
- **FooterTemplate:** Used to display footer text to the data source collection.
- **ItemStyle:** Used to apply styles to an ItemTemplate.
- **EditStyle:** Used to apply styles to an EditItemTemplate
- **HeaderStyle:** Used to apply styles to a HeaderTemplate
- **FooterStyle:** Used to apply styles to a FooterTemplate.

GridView:

The GridView control displays the values of a data source in a table. Each column represents a field, while each row represents a record. The GridView control supports the following features:

- Binding to data source controls, such as SqlDataSource.
- Built-in sort capabilities.
- Built-in update and delete capabilities.
- Built-in paging capabilities.
- Built-in row selection capabilities.
- Programmatic access to the GridView object model to dynamically set properties, handle events, and so on.
- Multiple key fields.
- Multiple data fields for the hyperlink columns.
- Customizable appearance through themes and styles.

FormView:

The FormView control is used to display a single record at a time from a data source. When you use the FormView control, you create templates to display and edit data-bound values. The templates contain controls, binding expressions, and formatting that define the look and functionality of the form.

Stored procedures:

Stored procedures (sprocs) are generally an ordered series of Transact-SQL statements bundled into a single logical unit. They allow for variables and parameters, as well as selection and looping constructs. A key point is that sprocs are stored in the database rather than in a separate file.

Advantages over simply sending individual statements to the server include:

- Referred to using short names rather than a long string of text; therefore, less network traffic is required to run the code within the sproc.
- Pre-optimized and precompiled, so they save an incremental amount of time with each sproc call/execution.
- Encapsulate a process for added security or to simply hide the complexity of the database.
- Can be called from other sprocs, making them reusable and reducing code size.

Parameterization Stored Procedure:

A stored procedure gives us some procedural capability, and also gives us a performance boost by using mainly two types of parameters:

- Input parameters
- Output parameters

From outside the sproc, parameters can be passed in either by position or reference.

Declaring Parameters in SQL Server Stored Procedures

- The name
- The datatype
- The default value
- The direction

The syntax is:

```
@parameter_name [AS] datatype [= default|NULL]
[VARYING][OUTPUT|OUT]
```

Code:

File: StudentsService.cs

```
using System;
```

```
using System.Data;
```

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

```
namespace Practical6
```

```
{
```

```
    public class StudentService
```

```
    {
```

```
        readonly SqlCommand cmd = new SqlCommand();
```

```
        readonly SqlParameter sqlpId = new SqlParameter("@id", SqlDbType.Int);
```

```
        readonly SqlParameter sqlpName = new SqlParameter("@Name",
```

```
        SqlDbType.VarChar);
```

```
        readonly SqlParameter sqlpScore = new SqlParameter("@score", SqlDbType.Int);
```

```
        internal StudentService()
```

```
        {
```

```
            cmd.CommandType = CommandType.StoredProcedure;
```

```
            cmd.Parameters.Add(sqlpId);
```

```
            cmd.Parameters.Add(sqlpName);
```

```
            cmd.Parameters.Add(sqlpScore);
```

```

        cmd.CommandText = "ajayk57_sql2";
    }
    internal bool AddStudentEntry(string id, string name, string score)
    {
        using (SqlConnection conn = new SqlConnection("Data
Source=DESKTOP-OUL5TLP\\SQLEXPRESS;Initial Catalog=coll;Integrated
Security=True"))
        {
            cmd.Connection = conn;
            sqlpId.Value = id;
            sqlpName.Value = name;
            sqlpScore.Value = score;
            conn.Open();
            try
            {
                cmd.ExecuteNonQuery();
                return true;
            }
            catch (Exception exc)
            {
                Console.WriteLine(exc.Message);
                return false;
            }
        }
    }
}
}
}
}
File: WebForm1.aspx
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="Practical6.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:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
DataKeyNames="id" DataSourceID="SqlDataSource1">
                <Columns>
                    <asp:BoundField DataField="id" HeaderText="id" ReadOnly="True"
SortExpression="id" />

```

```

        <asp:BoundField DataField="name" HeaderText="name"
SortExpression="name" />
        <asp:BoundField DataField="score" HeaderText="score"
SortExpression="score" />
    </Columns>
</asp:GridView>
<br />
    <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%%$
ConnectionStrings:collConnectionString %>" OnSelecting="SqlDataSource1_Selecting"
ProviderName="<%%$ ConnectionStrings:collConnectionString.ProviderName %>"
SelectCommand="SELECT * FROM [ajayk57_student]"></asp:SqlDataSource>
    <asp:FormView ID="FormView1" runat="server" BorderStyle="Dotted"
DataKeyNames="id" DataSourceID="SqlDataSource1"
OnPageIndexChanging="FormView1_PageIndexChanging">
    <EditItemTemplate>
        id:
        <asp:Label ID="idLabel1" runat="server" Text='<%%# Eval("id") %>' />
        <br />
        name:
        <asp:TextBox ID="nameTextBox" runat="server" Text='<%%# Bind("name") %>' />
        <br />
        score:
        <asp:TextBox ID="scoreTextBox" runat="server" Text='<%%# Bind("score") %>' />
        <br />
        <asp:LinkButton ID="UpdateButton" runat="server" CausesValidation="True"
CommandName="Update" Text="Update" />
        &nbsp;<asp:LinkButton ID="UpdateCancelButton" runat="server"
CausesValidation="False" CommandName="Cancel" Text="Cancel" />
    </EditItemTemplate>
    <InsertItemTemplate>
        id:
        <asp:TextBox ID="idTextBox" runat="server" Text='<%%# Bind("id") %>' />
        <br />
        name:
        <asp:TextBox ID="nameTextBox" runat="server" Text='<%%# Bind("name") %>' />
        <br />
        score:
        <asp:TextBox ID="scoreTextBox" runat="server" Text='<%%# Bind("score") %>' />
        <br />
        <asp:LinkButton ID="InsertButton" runat="server" CausesValidation="True"
CommandName="Insert" Text="Insert" />
        &nbsp;<asp:LinkButton ID="InsertCancelButton" runat="server"
CausesValidation="False" CommandName="Cancel" Text="Cancel" />
    </InsertItemTemplate>
    <ItemTemplate>
        id:

```

```

        <asp:Label ID="idLabel" runat="server" Text='<%=# Eval("id") %>' />
        <br />
        name:
        <asp:Label ID="nameLabel" runat="server" Text='<%=# Bind("name") %>' />
        <br />
        score:
        <asp:Label ID="scoreLabel" runat="server" Text='<%=# Bind("score") %>' />
        <br />

    </ItemTemplate>
</asp:FormView>
<br />
<asp:DataList ID="DataList1" runat="server" BorderStyle="Groove"
DataKeyField="id" DataSourceID="SqlDataSource1" GridLines="Horizontal">
    <ItemTemplate>
        id:
        <asp:Label ID="idLabel" runat="server" Text='<%=# Eval("id") %>' />
        <br />
        name:
        <asp:Label ID="nameLabel" runat="server" Text='<%=# Eval("name") %>' />
        <br />
        score:
        <asp:Label ID="scoreLabel" runat="server" Text='<%=# Eval("score") %>' />
        <br />
    <br />
    </ItemTemplate>
</asp:DataList>
</div>
</form>
</body>
</html>

```

File: WebForm1.aspx.cs

```

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

```

namespace Practical6

```

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

```

```

    }

    protected void SqlDataSource1_Selecting(object sender,
SqlDataSourceSelectingEventArgs e)
    {

    }

    protected void FormView1_PageIndexChanging(object sender,
FormViewPageEventArgs e)
    {

    }
}
}

```

File: WebForm2.aspx

```

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm2.aspx.cs"
Inherits="Practical6.WebForm2" %>

```

```

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>Practical 6-2</title>
</head>
<body>
<form id="form1" runat="server">
    <asp:GridView ID="GridView1" runat="server"></asp:GridView>
</form>
</body>
</html>

```

File: WebForm2.aspx.cs

```

using System;
using System.Data;
using System.Data.SqlClient;

```

```

namespace Practical6

```

```

{
    public partial class WebForm2 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            using (SqlConnection con = new SqlConnection("Data
Source=DESKTOP-OUL5TLP\\SQLEXPRESS;Initial Catalog=coll;Integrated
Security=True"))

```

```
{
    con.Open();
    SqlCommand cmd = new SqlCommand("ajayk57_sql", con);
    cmd.CommandType = CommandType.StoredProcedure;
    SqlDataReader rdr = cmd.ExecuteReader();
    GridView1.DataSource = rdr;
    GridView1.DataBind();
}

}

}
```

File: WebForm3.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm3.aspx.cs"
Inherits="Practical6.WebForm3" %>
```

```
<!DOCTYPE html>
```

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

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

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

```
</head>
```

```
<body>
```

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

```
<div>
```

```
    Please Enter your information:<br />
```

```
    Roll No.:
```

```
    <asp:TextBox ID="TextBox1" runat="server"
```

```
    OnTextChanged="TextBox1_TextChanged"></asp:TextBox>
```

```
    <br />
```

```
    Name:
```

```
    <asp:TextBox ID="TextBox2" runat="server"
```

```
    OnTextChanged="TextBox1_TextChanged"></asp:TextBox>
```

```
    <br />
```

```
    Score:
```

```
    <asp:TextBox ID="TextBox3" runat="server"
```

```
    OnTextChanged="TextBox1_TextChanged"></asp:TextBox>
```

```
    <br />
```

```
    <asp:Button ID="RegBtn" runat="server" Text="Register" OnClick="RegBtn_Click" />
```

```
    <br />
```

```
    Status: <asp:Label ID="Label1" runat="server" Text="----"></asp:Label>
```

```
</div>
```

```
</form>
```

```
</body>
```


</html>

File: WebForm3.aspx.cs

using System;

using System.Data;

using System.Data.SqlClient;

namespace Practical6

{

public partial class WebForm3 : System.Web.UI.Page

{

static StudentService StudentService;

protected void Page_Load(object sender, EventArgs e)

{

StudentService = new StudentService();

}

protected void TextBox1_TextChanged(object sender, EventArgs e)

{

}

protected void RegBtn_Click(object sender, EventArgs e)

{

 //SqlConnection conn = new SqlConnection("Data
Source=DESKTOP-OUL5TLP\\SQLEXPRESS;Initial Catalog=coll;Integrated
Security=True");

 //SqlCommand cmd = new SqlCommand("ajayk57_sql2", conn);

 //cmd.CommandType = CommandType.StoredProcedure;

 //SqlParameter sqlpId = new SqlParameter("@id", SqlDbType.Int);

 //SqlParameter sqlpName = new SqlParameter("@Name", SqlDbType.VarChar);

 //SqlParameter sqlpScore = new SqlParameter("@score", SqlDbType.Int);

 //sqlpId.Value = TextBox1.Text;

 //sqlpName.Value = TextBox2.Text;

 //sqlpScore.Value = TextBox3.Text;

 //cmd.Parameters.Add(sqlpId);

 //cmd.Parameters.Add(sqlpName);

 //cmd.Parameters.Add(sqlpScore);

 //conn.Open();

 //cmd.ExecuteNonQuery();

 //Label1.Text = "Succeeded";

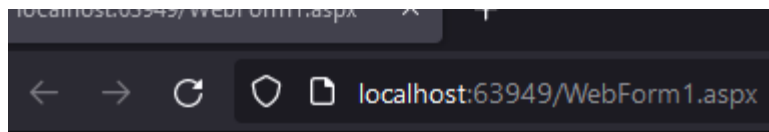
 //Console.WriteLine("huh?");

 //Label1.Text = "Failed";

```
        if (StudentService.AddStudentEntry(TextBox1.Text, TextBox2.Text, TextBox3.Text))
        {
            Label1.Text = "Succeeded";
        }
        else
        {
            Label1.Text = "Failed";
        }
    }
}
```

Output:

Q1)



id	name	score
1	Ajay Karthikesan	100
2	Jim	100
3	Joy	90
4	Jack	91
5	Afzal	95
6	Kishor	92
7	Ram	93

id: 1
name: Ajay Karthikesan
score: 100

id: 1 name: Ajay Karthikesan score: 100
id: 2 name: Jim score: 100
id: 3 name: Joy score: 90
id: 4 name: Jack score: 91
id: 5 name: Afzal score: 95

Q2)

```
SQLQuery1.sql - D...UL5TLP\Admin (70))* -p X
CREATE PROCEDURE ajayk57_sql
AS
BEGIN
select * from ajayk57_student
END
```

100 %

Results

Commands completed successfully.

Completion time: 2023-05-26T16:59:45.7344286+05:30

localhost:63949/WebForm2.aspx

id	name	score
1	Ajay Karthikesan	100
2	Jim	100
3	Joy	90
4	Jack	91
5	Afzal	95
6	Kishor	92
7	Ram	93

Q3)

```
CREATE PROCEDURE ajayk57_sql2(  
    @id INT,  
    @Name VARCHAR(30),  
    @score int  
)  
AS  
BEGIN  
    INSERT INTO ajayk57_student(  
        [id],  
        [name],  
        [score]  
    )  
    VALUES (  
        @id,  
        @name,  
        @score  
    )  
END
```

Messages

Commands completed successfully.

Completion time: 2023-05-26T17:10:42.6576547+05:30

localhost:63949/WebForm3.aspx

Please Enter your information:

Roll No.:

Name:

Score:

Status: Succeeded

localhost:63949/WebForm3.aspx

Please Enter your information:

Roll No.:

Name:

Score:

Status: Failed

Conclusion:

Successfully designed webpage to demonstrate use of data bound controls, working of simple stored procedure and

parameterized stored procedure.

Name of Student: Ajay Karthikesan			
Roll Number: 57		Assignment Number: 7	
Aim of Assignment: Design a webpage to display the use of LINQ			
DOP:		DOS: 3.6.23	
CO Mapped: CO2	PO Mapped: PO3, PO5, PSO1, PSO2	Faculty Signature:	Marks:

Practical No. 7

Aim: Design a webpage to display the use of LINQ

Theory:

LINQ to SQL Classes:

LINQ to SQL (also known as DLINQ) is an electrifying part of Language Integrated Query as this allows querying data in SQL server database by using usual LINQ expressions. It also allows you to update, delete and insert data, but the only drawback from which it suffers is its limitation to the SQL server database. However, there are many benefits of LINQ to SQL over ADO.NET like reduced complexity, few lines of coding and many more. Below is a diagram showing the execution architecture of LINQ to SQL.

Entity Data Model/Framework:

The Entity Data Model (EDM) is a set of concepts that describe the structure of data, regardless of its stored form. The EDM borrows from the EntityRelationship Model described by Peter Chen in 1976, but it also builds on the Entity-Relationship Model and extends its traditional uses.

EDM supports a set of primitive data types that define properties in a conceptual model. We need to consider 3 core parts which form the basis for Entity Framework and collectively it is known as Entity Data Model. Following are the three core parts of EDM.

The Storage Schema Model:

- The Storage Model also called as Storage Schema Definition Layer (SSDL) represents the schematic representation of the backend data store.

The Conceptual Model:

- The Conceptual Model also called as Conceptual Schema Definition Layer (CSDL) is the real entity model, against which we write our queries.

The Mapping Model:

- Mapping Layer is just a mapping between the Conceptual model and the Storage model.
- The logical schema and its mapping with the physical schema is represented as an EDM.
 - Visual Studio also provides Entity Designer, for visual creation of the EDM and the mapping specification.
 - The output of the tool is the XML file (*.edmx) specifying the schema and mapping.
 - The Edmx file contains Entity Framework metadata artifacts.


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

namespace Practical7
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void FindIDBtn_Click(object sender, EventArgs e)
        {
            try
            {
                int id = Convert.ToInt32(IDTxt.Text);

                DataClasses1DataContext dbCon = new
DataClasses1DataContext();
                GridView1.DataSource = (from game in
dbCon.ajayk57_games
                                where
                                    game.id == id
                                select game);

                GridView1.DataBind();
            } catch (Exception) {
                Response.Write("Please enter a valid, integer");
            }
        }

        protected void AllRecBtn_Click(object sender, EventArgs e)
        {
            DataClasses1DataContext dbCon=new
DataClasses1DataContext();
            GridView1.DataSource = (from game in dbCon.ajayk57_games
select game);
            GridView1.DataBind();
        }
    }
}
```

File: WebForm2.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm2.aspx.cs" Inherits="Practical7.WebForm2" %>
```

[illegible]

File: WebForm2.aspx.cs

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

namespace Practical7
{
    public partial class WebForm2 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void FindIDBtn_Click(object sender, EventArgs e)
        {
            try
            {
                int id = Convert.ToInt32(IDTxt.Text);

                DataClasses1DataContext dbCon = new
```

```
DataClasses1DataContext();
        GridView1.DataSource = (from book in
dbCon.ajayk57_libraries
                                where
                                    book.id == id
                                select book);
        GridView1.DataBind();
    }
    catch (Exception)
    {
        Response.Write("Please enter a valid, integer");
    }
}
protected void AllRecBtn_Click(object sender, EventArgs e)
{
    DataClasses1DataContext dbCon = new
DataClasses1DataContext();
    GridView1.DataSource = (from book in
dbCon.ajayk57_libraries select book);
    GridView1.DataBind();
}
}
```

Output:

3

Find ID

Show Every Record

id	name	is_outdoor	pcount
1	Cricket	<input checked="" type="checkbox"/>	11
2	Football	<input checked="" type="checkbox"/>	11
3	Carrom	<input type="checkbox"/>	4

2	Find ID	Show Every Record
id	name	author
2	Carnival	Mitch Albom

← → ↻ ⓘ localhost:2470/WebForm1.aspx

3

id	name	is_outdoor	pcount
3	Carrom	<input type="checkbox"/>	4

← → ↻ ⓘ localhost:2470/WebForm2.aspx

2

id	name	author
1	Tuesdays with Morrie	Mitch Albom
2	Carnival	Mitch Albom
3	HP: 1	JK
4	White Ferry	Mitch Albom
5	Croc Cricket	Ruskin Bond
6	Old Friend	Ruskin Bond
7	Blue Umbrella	Ruskin Bond

Conclusion:

I learnt how to use LINQ to make web applications.

Name of Student: Ajay Karthikesan			
Roll Number: 57		Assignment Number: 8	
Aim of Assignment: State Management			
DOP:		DOS: 3.6.23	
CO Mapped: CO3	PO Mapped: PO3, PO5, PSO1, PSO2	Faculty Signature:	Marks:

Practical No. 8

Aim: State Management

Theory:

State Overview:

As we all know, browsers are generally stateless.

Now the question arises here, what does stateless actually mean?

Stateless means, whenever we visit a website, our browser communicates with the respective server depending on our requested functionality or the request. The browser communicates with the respective server using the HTTP or HTTPs protocol.

But after that response, what's next or what will happen when we visit that website again after closing our web browser?

In this case HTTP/HTTPs doesn't remember what website or URL we visited, or in other words we can say it doesn't hold the state of a previous website that we visited before closing our browser, that is called stateless. Now I guess you have at least an idea of what state and stateless actually means. So our browsers are stateless.

State Outline:

As I said in the beginning, HTTP is a stateless protocol. It just cleans up or we can say removes all the resources/references that were serving a specific request in the past. These resources can be:

- Objects
- Allocated Memory
- Sessions ID's
- Some URL info and so on.

State Management Types:

In ASP.NET there are the following 2 State Management methodologies:

State Management Techniques:

State Management techniques are based on client side and server side. Their functionality differs depending on the change in state, so here is the hierarchy:

Client-Side State Management:

Whenever we use Client-Side State Management, the state related information will directly get stored on the client-side. That specific information will travel back and communicate with every request generated by the user then afterwards provides responses after server-side communication.

This architecture is something like the following,

Hidden Field:

A hidden field is used for storing small amounts of data on the client side. In most simple words it's just a container of some objects but their result is not rendered on our web browser. It is invisible in the browser.

It stores a value for the single variable and it is the preferable way when a variable's value is changed frequently but we don't need to keep track of that every time in our application or web program.

Cookies:

A set of Cookies is a small text file that is stored in the user's hard drive using the client's browser. Cookies are just used for the sake of the user's identity matching as it only stores information such as session's ids, some frequent navigation or post-back request objects.

Whenever we get connected to the internet for accessing a specific service, the cookie file is accessed from our hard drive via our browser for identifying the user. The cookie access depends upon the life cycle or expiration of that specific cookie file.

Query Strings:

Query strings are used for some specific purpose. These in a general case are used for holding some value from a different page and move these values to the different page. The information stored in it can be easily navigated to one page to another or to the same page as well.

View State:

In general, we can say it is used for storing user data in ASP.NET, sometimes in ASP.NET applications the user wants to maintain or store their data temporarily after a post-back. In this case VIEW STATE is the most used and preferred way of doing that.

This property is enabled by default but we can make changes depending on our functionality, what we need to do is just change the EnableViewState value to either TRUE for enabling it or FALSE for the opposite operation.

Server-Side State Management:

Server-Side State Management is different from Client-Side State Management but the operations and working are somewhat the same in functionality. In Server-Side State Management all

the information is stored in the user memory. Due to this functionality, there is more secure domains at the server side in comparison to Client-Side State Management.

The structure is something like the following,

It is another way which ASP.NET provides to store the user's specific information or the state of the application on the server machine. It completely makes use of server resources (the server's memory) to store information.

This management technique basically makes use of the following,

- a. Session State
- b. Application State

Session State:

Session is one of the most common way which is being used by developers to maintain the state of the application. The Session basically stores the values as a dictionary collection in key/value pairs. It completely utilizes server resources to store the data. It is a secure way of storing data, since the data will never be passed to the client.

For each and every user, a separate Session is created, and each and every Session has its Unique ID. This ID is being stored in the client's machine using cookies. If there are multiple users who are accessing a web application, then for each user a separate Session is created. If a single user logs in and logs out the Session is killed, then if the same user again logs into the application, then a new Session ID is being created for the same user.

The Session has a default timeout value (20 minutes). We can also set the timeout value for a session in the web.config file.

There are various ways in which we can store a session and they are as follows:

1. OFF
2. InProc
3. State Server
4. SQL Server

Application State:

If the information that we want to be accessed or stored globally throughout the application, even if multiple users access the site or application at the same time, then we can use an Application Object for such purposes.

It stores information as a Dictionary Collection in key - value pairs. This value is accessible across the pages of the application

/ website.

There are 3 events of the Application which are as follows

- Application_Start
- Application_Error
- Application_End

Example

Just for an example, I am setting the Page title in the

Application Start event of the Global.asax file.

the Key and "Welcome to State Management Application is the value.

Code:

File: a/WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="Practical8.WebForm1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server" Text="Name"></asp:Label>
            <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            <br />
            <asp:Label ID="Label2" runat="server" Text="Age:"></asp:Label>
            <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
            <br />
            <br />
            <asp:Label ID="Label3" runat="server"
Text="Branch:"></asp:Label>
            <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
            <br />
            <br />
            <asp:Button ID="Button2" runat="server" OnClick="Button2_Click"
Text="Button" />
            <asp:Button ID="Button1" runat="server"
PostBackUrl="~/WebForm2.aspx"
Text="Submit" />
            <br />
            <br />
            <asp:HiddenField ID="HiddenField1" runat="server" />
            <asp:HiddenField ID="HiddenField2" runat="server" />
            <asp:HiddenField ID="HiddenField3" runat="server" />
            <br />
        </div>
    </form>
</body>
</html>
```

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

File: a/WebForm1.aspx.cs

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

namespace Practical8
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button2_Click(object sender, EventArgs e)
        {
            HiddenField1.Value = TextBox1.Text;
            HiddenField2.Value = TextBox2.Text;
            HiddenField3.Value = TextBox3.Text;
            Response.Write(HiddenField1.Value);
            Response.Write(HiddenField2.Value);
            Response.Write(HiddenField3.Value);
        }
    }
}
```

File: a/WebForm2.aspx.cs

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm2.aspx.cs" Inherits="Practical8.WebForm2" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
            <asp:Label ID="Label2" runat="server" Text="Label"></asp:Label>
            <asp:Label ID="Label3" runat="server" Text="Label"></asp:Label>
```

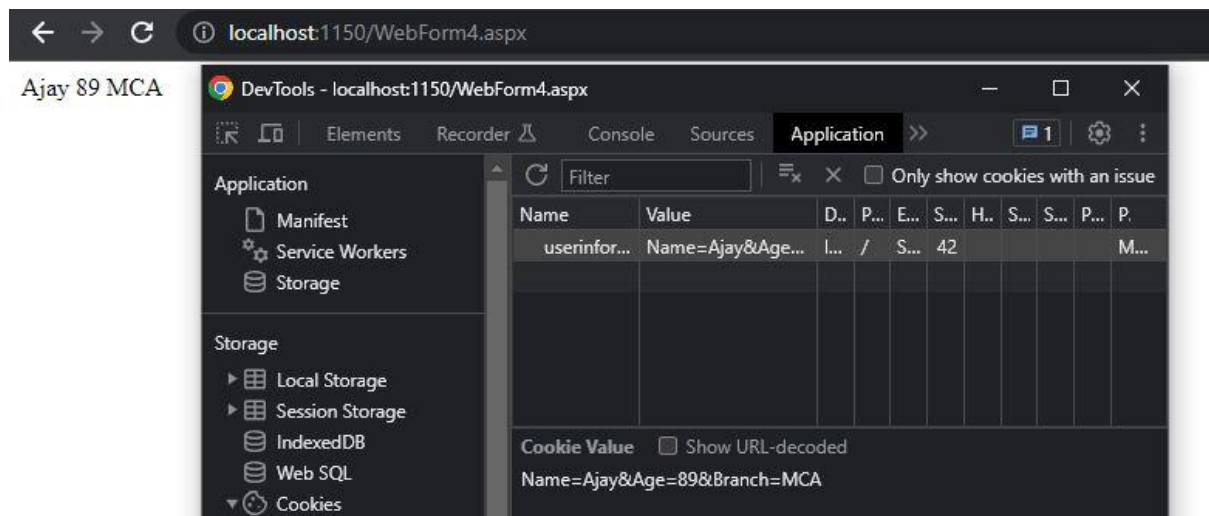
```
</div>
</form>
</body>
</html>
```

File: a/WebForm2.aspx.cs

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

namespace Practical8
{
    public partial class WebForm2 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            String value1 = Request.Form["HiddenField1"];
            String value2 = Request.Form["HiddenField2"];
            String value3 = Request.Form["HiddenField3"];
            Label1.Text = value1;
            Label2.Text = value2;
            Label3.Text = value3;
        }
    }
}
```

Output:

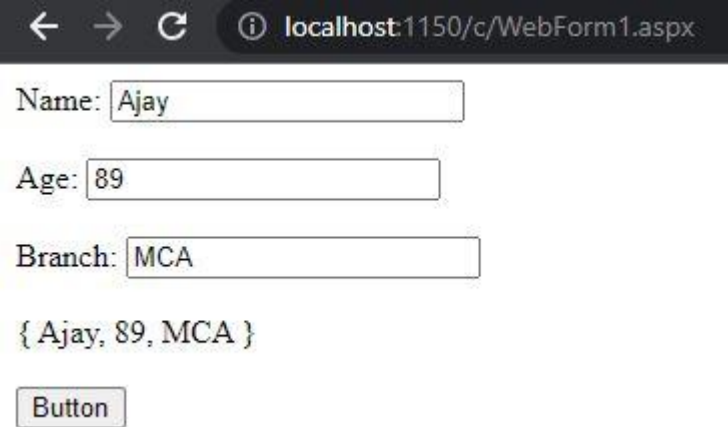


The screenshot shows a web form on `localhost:1150/WebForm1.aspx`. The form contains the following elements:

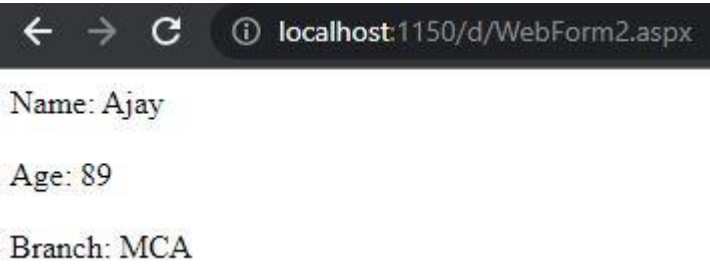
- A label `Ajay89MCA` above the form fields.
- A text input field for 'Name' with the value 'Ajay'.
- A text input field for 'Age' with the value '89'.
- A text input field for 'Branch' with the value 'MCA'.
- Two buttons: 'Button' and 'Submit'.

The screenshot shows a web form on `localhost:1150/WebForm2.aspx`. The form contains the following elements:

- A label `Ajay 89 MCA` above the form fields.



A screenshot of a web browser showing a form titled 'WebForm1.aspx' at 'localhost:1150/c/'. The form contains three input fields: 'Name' with the value 'Ajay', 'Age' with the value '89', and 'Branch' with the value 'MCA'. Below these fields is a text label '{ Ajay, 89, MCA }' and a button labeled 'Button'.



A screenshot of a web browser showing a form titled 'WebForm2.aspx' at 'localhost:1150/d/'. The form displays the values from the previous form: 'Name: Ajay', 'Age: 89', and 'Branch: MCA'.

e

Conclusion:

I learnt how to do state management in ASP.NET.

Name of Student: Ajay Karthikesan			
Roll Number: 57		Assignment Number: 9	
Aim of Assignment: A. Design Web Application to produce and consume a web Service. Write an Application that uses these services. B. Design Web Application to produce and consume a web Service.			
DOP:		DOS: 3.623	
CO Mapped: CO4	PO Mapped: PO3, PO5, PSO1, PSO2	Faculty Signature:	Marks:

Practical No. 9

Aim:

- A. Design Web Application to produce and consume a web Service. Write an Application that uses these services.
- B. Design Web Application to produce and consume a web Service.

Theory:

Web Services:

A Web Service is a software program that uses XML to exchange information with other software via common internet protocols. In a simple sense, Web Services are a way of interacting with objects over the Internet.

A web service is

- Language Independent.
- Protocol Independent.
- Platform Independent.
- It assumes a stateless service architecture.
- Scalable (e.g. multiplying two numbers together to an entire customerrelationship management system).
- Programmable (encapsulates a task).
- Based on XML (open, text-based standard).
- Self-describing (metadata for access and use).
- Discoverable (search and locate in registries)- ability of applications and developers to search for and locate desired Web services through registries. This is based on UDDI.

Web services advantages:-

- Use open, text-based standards, which enable components written in various languages and for different platforms to communicate.
- Promote a modular approach to programming, so multiple organizations can communicate with the same Web service.
- Comparatively easy and inexpensive to implement, because they employ an existing infrastructure and because most applications can be repackaged as Web services.
- Significantly reduce the costs of enterprise application (EAI) integration and B2B communications.
- Implemented incrementally, rather than all at once which lessens the cost and reduces the organizational disruption from an abrupt switch in technologies.
- The Web Services Interoperability Organization (WS-I) consisting of over 100 vendors promotes interoperability.

Web Services Limitations :-

- SOAP, WSDL, UDDI- require further development.

- Interoperability.
- Royalty fees.
- Too slow for use in high-performance situations.
- Increase traffic on networks.
- The lack of security standards for Web services.
- The standard procedure for describing the quality (i.e. levels of performance, reliability, security etc.) of particular Web services – management of Web services.
- The standards that drive Web services are still in draft form (always will be in refinement).
- Some vendors want to retain their intellectual property rights to certain Web services standards.

Web Service Example:-

A web service can perform almost any kind of task.

- Web Portal- A web portal might obtain top news headlines from an Associated press web service.
- Weather Reporting- You can use Weather Reporting web service to display weather information in your personal website.

- Stock Quote- You can display latest update of Share market with Stock Quote on your web site.
- News Headline-You can display latest news update by using News Headline Web Service in your website.

Web Services using http:

If the web service uses an HTTPS connection, then this will communicate using SSL. SSL, or Secure Socket Layer, is a technology which allows the web service client and remote web service server to communicate over a secured connection.

The data is encrypted before being transmitted and decrypted when the data is received before processing. This is a two-way process, meaning that both the server AND the browser encrypt all traffic before sending out data. A certificate is required for SSL communication.

Web Service in ASP.NET:

A Web Service is programmable application logic accessible via standard Web protocols. One of these Web protocols is the Simple Object Access Protocol (SOAP). SOAP is a W3C submitted note (as of May 2000) that uses standards based technologies (XML for data description and HTTP for transport) to encode and transmit application data.

Consumers of a Web Service do not need to know anything about the platform, object model, or programming language used to implement the service; they only need to understand how to send and receive SOAP messages (HTTP and XML).

WCF Service:

Windows Communication Foundation (WCF) is a framework

for building serviceoriented applications. Using WCF, you can send data as asynchronous messages from one service endpoint to another. A service endpoint can be part of a continuously available service hosted by IIS, or it can be a service hosted in an application. An endpoint can be a client of a service that requests data from a service endpoint. The messages can be as simple as a single character or word sent as XML, or as complex as a stream of binary data.

In what scenarios must WCF be used

- A secure service to process business transactions.
- A service that supplies current data to others, such as a traffic report or other monitoring service.
- A chat service that allows two people to communicate or exchange data in real time.
- A dashboard application that polls one or more services for data and presents it in a logical presentation.
- Exposing a workflow implemented using Windows Workflow Foundation as a WCF service.
- A Silverlight application to poll a service for the latest data feeds.

Features of WCF:

- Service Orientation
- Interoperability
- Multiple Message Patterns
- Service Metadata
- Data Contracts
- Security
- Multiple Transports and Encodings
- Reliable and Queued Messages
- Durable Messages
- Transactions
- AJAX and REST Support
- Extensibility

Code:

File: WebForm1.aspx

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
```

```
<body>
    <form id="form1" runat="server">
        <div>
            <asp:GridView ID="GridView1" runat="server">
            </asp:GridView>
            <br />
            <asp:Button ID="Button1" runat="server" OnClick="Button1_Click"
Text="Fetch Data" />
        </div>
    </form>
</body>
</html>
```

File: WebForm1.aspx.cs

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

namespace Practical9
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void Button1_Click(object sender, EventArgs e)
        {
            WebService1 db = new WebService1();
            GridView1.DataSource = db.Get();
            GridView1.DataBind();
        }
    }
}
```

File: WebForm2.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm2.aspx.cs" Inherits="Practical9.WebForm2" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
```

```

<div>
    <asp:GridView ID="GridView1" runat="server">
    </asp:GridView>
    <br />
    <b>Enter Game Information:-</b><br />
    ID&nbsp;
    <asp:TextBox ID="idTxt" runat="server"></asp:TextBox>
    <br />
    Name&nbsp;
    <asp:TextBox ID="nameTxt" runat="server"></asp:TextBox>
    <br />
    <asp:CheckBox ID="outdoorChkBx" runat="server" Text="Is
Outdoor?" />
    <br />
    Player count&nbsp;
    <asp:TextBox ID="pcountTxt" runat="server"></asp:TextBox>
    <br />
    Status&nbsp;
    <asp:Label ID="statusLbl" runat="server">_____</asp:Label>
    <br />
    <asp:Button ID="Button1" runat="server" OnClick="Button1_Click"
Text="Insert" />
</div>
</form>
</body>
</html>

```

File: WebForm2.aspx.cs

```

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

namespace Practical9
{
    public partial class WebForm2 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            WebService1 obj1 = new WebService1();
            GridView1.DataSource = obj1.Get();
            GridView1.DataBind();
        }
        protected void Button1_Click(object sender, EventArgs e)
        {
            int id = Int32.Parse(idTxt.Text);
            String name = nameTxt.Text;

```

```
        bool is_outdoor = outdoorChkBx.Checked;
        int pcount = Convert.ToInt32(pcountTxt.Text);
        WebService1 obj1 = new WebService1();
        obj1.AddData(name, id, is_outdoor, pcount);
        statusLbl.Text = "Success";
        GridView1.DataSource = obj1.Get();
        GridView1.DataBind();
    }
}
```

File: WebService1.asmx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
using System.Data.SqlClient;
using System.Data;

namespace Practical9
{
    /// <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
    {
        private SqlConnection sqlcon = new SqlConnection("Data
Source=SNAKEJAZZ\\SQLEXPRESS;Initial Catalog=coll;Integrated
Security=True");

        [WebMethod]
        public string HelloWorld()
        {
            return "Hello World";
        }

        public DataTable Get()
        {
            using (SqlCommand cmd = new SqlCommand("SELECT * FROM
ajayk57_games"))
            {
                using (SqlDataAdapter sda = new SqlDataAdapter())
                {
                    cmd.Connection = sqlcon;
                    sda.SelectCommand = cmd;
                }
            }
        }
    }
}
```

```
        DataTable dt = new DataTable();
        dt.TableName = "Games";
        sda.Fill(dt);
        return dt;
    }
}

[WebMethod]
public void AddData(String name, int id, bool is_outdoor, int
pcount)
{
    SqlCommand cmd = new SqlCommand("INSERT INTO
ajayk57_games(id, name, is_outdoor, pcount) VALUES( @id,@Name,
@is_outdoor, @pcount)");
    cmd.Parameters.AddWithValue("@name", name);
    cmd.Parameters.AddWithValue("@id", id);
    cmd.Parameters.AddWithValue("@is_outdoor", is_outdoor);
    cmd.Parameters.AddWithValue("@pcount", pcount);
    cmd.Connection = sqlcon;
    sqlcon.Open();
    cmd.ExecuteNonQuery();
    sqlcon.Close();
}
}
```

File: Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Practical9B
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            Service1 obj = new
            Service1();
        }
    }
}
```

```
        resultLbl.Text =
obj.Addition(Convert.ToInt32(textBox1.Text),
        Convert.ToInt32(textBox2.Text)).ToString();
    }
    private void button2_Click(object sender, EventArgs e)
    {
        Service1 obj = new
        Service1();
        resultLbl.Text =
obj.Subtraction(Convert.ToInt32(textBox1.Text),
        Convert.ToInt32(textBox2.Text)).ToString();
    }
    private void button3_Click(object sender, EventArgs e)
    {
        Service1 obj = new Service1();
        resultLbl.Text =
        obj.Multiplication(Convert.ToInt32(textBox1.Text),
        Convert.ToInt32(textBox2.Text)).ToString();
    }
    private void button4_Click(object sender, EventArgs e)
    {
        Service1 obj = new
        Service1();
        resultLbl.Text =
obj.Division(Convert.ToInt32(textBox1.Text),
        Convert.ToInt32(textBox2.Text)).ToString();
    }
}
}
```

File: Service1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Linq;
using System.ServiceProcess;
using System.Text;

namespace Practical9B
{
    partial class Service1 : ServiceBase
    {
        public Service1()
        {
            InitializeComponent();
        }
    }
}
```

```
protected override void OnStart(string[] args)
{
    // TODO: Add code here to start your service.
}

protected override void OnStop()
{
    // TODO: Add code here to perform any tear-down necessary
to stop your service.
}
public int Addition(int num1, int num2)
{
    return num1 + num2;
}
public int Subtraction(int num1, int num2)
{
    return num1 - num2;
}
public int Multiplication(int num1, int num2)
{
    return num1 * num2;
}
public int Division(int num1, int num2)
{
    return num1 / num2;
}
}
```

Output:

A.

← → ↻ ⓘ localhost:4077/WebForm1.aspx

id	name	is_outdoor	pcount
1	Cricket	<input checked="" type="checkbox"/>	11
2	Football	<input checked="" type="checkbox"/>	11
3	Carrom	<input type="checkbox"/>	4

Fetch Data

← → ↻ ⓘ localhost:4077/WebForm2.aspx

id	name	is_outdoor	pcount
1	Cricket	<input checked="" type="checkbox"/>	11
2	Football	<input checked="" type="checkbox"/>	11
3	Carrom	<input type="checkbox"/>	4
4	Hockey	<input checked="" type="checkbox"/>	11

Enter Game Information:-

ID

Name

☒ Is Outdoor?

Player count

Status Success

Insert

← → ↻ ⓘ localhost:4077/WebForm1.aspx

Fetch Data

B.

The image displays four sequential screenshots of a web form titled "Form1". Each screenshot shows the same form with two input fields, "Number 1" and "Number 2", both containing the value "8". Below these fields are four buttons representing arithmetic operations: "+", "-", "x", and "/". The "Answer" field shows the result of the selected operation.

- Top Left Screenshot:** The "+" button is highlighted with a blue border. The "Answer" field displays "10".
- Top Right Screenshot:** The "-" button is highlighted with a blue border. The "Answer" field displays "6".
- Bottom Left Screenshot:** The "x" button is highlighted with a blue border. The "Answer" field displays "16".
- Bottom Right Screenshot:** The "/" button is highlighted with a blue border. The "Answer" field displays "4".

Conclusion:

I learnt how to create and use web services.

Name of Student: Ajay Karthikesan			
Roll Number: 57		Assignment Number: 10	
Aim of Assignment: Model View Controller a. Create a MVC application to demonstrate the use of Httpget and Httppost. b. Create a MVC application to Edit,Create, Delete and to display the data from the database using ADO.net Entity Model.			
DOP:		DOS: 3.6.23	
CO Mapped: CO4	PO Mapped: PO3, PO5, PSO1, PSO2	Faculty Signature:	Marks:

Practical No. 10

Aim: Design MVC based Web applications

- A. Create a MVC application to demonstrate the use of Httpget and Httppost.
- B. Create a MVC application to Edit, Create, Delete and to display the data from the database using ADO.net Entity Model.

Theory:

MVC - Model View Controller:

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components is built to handle specific development aspects of an application. MVC is one of the most frequently used industry standard web development frameworks to create scalable and extensible projects.

MVC Components

- Model

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic related data. For example, a customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

- View

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

- Controller

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component, and interact with the Views to render the final output.

Code:

```
Model: Employee.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
namespace Practical_10A.Models
{
    public class Employee
    {
```

```
public string Name { get; set; }
public string Address { get; set; }
public int Age { get; set; }
}}
```

Controller: HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Practical_10A.Models;
namespace Practical_10A.Controllers
{
    public class HomeController : Controller
    {
        [HttpGet]
        public ActionResult Index()
        {
            Employee emp = new Employee()
            {
                Address = "Nerul",
                Name = "Ajay",
                Age = 89
            };
            return View(emp);
        }
        [HttpPost]
        public ActionResult Index(Employee emp)
        {
            return View("DisplayData", emp);
        }
    }
}
```

Views:

Index.cshtml

```
@model Practical_10A.Models.Employee
```

```
@{
```

```
Layout = null;
```

```
}
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<meta name="viewport" content="width=device-width" />
```

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

```
</head>
```

```
<body>
<div>
@using (Html.BeginForm("Myform"))
{
    @Html.TextBoxFor(x => x.Name) <br /> <br /> <br />
    @Html.TextBoxFor(x => x.Address) <br /> <br /> <br />
    @Html.TextBoxFor(x => x.Age) <br /> <br /> <br />
    <input type="submit" value="submit" />
}
</div>
</body>
</html>
```

DisplayData.cshtml

```
@model Practical_10A.Models.Employee
@{
    Layout = null;
}
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width" />
<title>DisplayData</title>
</head>
<body>
<div>
<h1>Employee Details</h1>
<p>Employee Name: @Model.Name</p>
<p>Employee Age: @Model.Age</p>
<p>Employee Address: @Model.Address</p>
</div>
</body>
</html>
```

B.

Step 1: Create a new ASP.Net application with MVC in core references selected and add ADO.Net Entity Data Model Select an EF designer from the DB model.

Step 2: Make a connection, test connection, and select desired database.

Step 3: Choose the desired Table which shows later in a.edmx file.

Step 4: Right-click on the controller and add a new controller and select the one with MVC5 with views and click add.

Step 5: Ensure the routeConfig.cs file in AppStart folder has correct details of controller and view you wish to run.

```
using System;
```

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

```
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;
namespace Practical_10B
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Students", action =
                    "Index", id = UrlParameter.Optional }
            );
        }
    }
}
```

Step 6: Now run the generated code file.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using Practical_10B;
namespace Practical_10B.Controllers
{
    public class StudentsController : Controller
    {
        private Vesit_40Entities db = new Vesit_40Entities();
        // GET: Students
        public ActionResult Index()
        {
            return View(db.Students.ToList());
        }
        // GET: Students/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {

```

```
return new
 HttpStatusCodeResult(HttpStatusCode.BadRequest);
}
Student student = db.Students.Find(id);
if (student == null)
{
return HttpNotFound();
}
return View(student);
}
// GET: Students/Create
public ActionResult Create()
{
return View();
}
// POST: Students/Create
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include =
"Roll_No,Name,Programme,Course")] Student student)
{
if (ModelState.IsValid)
{
db.Students.Add(student);
db.SaveChanges();
return RedirectToAction("Index");
}
return View(student);
}
// GET: Students/Edit/5
public ActionResult Edit(int? id)
{
if (id == null)
{
return new
 HttpStatusCodeResult(HttpStatusCode.BadRequest);
}
Student student = db.Students.Find(id);
if (student == null)
{
return HttpNotFound();
}
return View(student);
}
// POST: Students/Edit/5
[HttpPost]
```



```
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include =
"Roll_No,Name,Programme,Course")] Student student)
{
    if (ModelState.IsValid)
    {
        db.Entry(student).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(student);
}
```

```
// GET: Students/Delete/5
public ActionResult Delete(int? id)
{
    if (id == null)
    {
        return new
        HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Student student = db.Students.Find(id);
    if (student == null)
    {
        return HttpNotFound();
    }
    return View(student);
}

// POST: Students/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(int id)
{
    Student student = db.Students.Find(id);
    db.Students.Remove(student);
    db.SaveChanges();
    return RedirectToAction("Index");
}

protected override void Dispose(bool disposing)
{
    if (disposing)
    {
        db.Dispose();
    }
    base.Dispose(disposing);
}
```

```
}  
}
```

Output:

A.

Employee Details

Employee Name: Ajay

Employee Age: 57

Employee Address: Nerul

B.

— [Application name](#)

Index

[Create New](#)

Name	Programme	Course	
Alex	MCA	AWT	Edit Details Delete
Lia	BCA	Java	Edit Details Delete
Nora	MCA	UI/UX	Edit Details Delete
Shruti	MCA	AWT	Edit Details Delete
Prathamesh	MCA	IS	Edit Details Delete
Mark	BCA	C# Programming	Edit Details Delete
Jackson	MCA	DMBA	Edit Details Delete

© 2021 - My ASP.NET Application

Conclusion:

We have successfully created MVC Application to:

- A. Demonstrate the usage of HTTP Get & HTTP Post
- B. Edit, Create, Delete, and display the data from the database using ADO.net Entity Model.