Finalpractical-dotnet

NET Centric Computing (Kathmandu Bernhardt College)

Kathmandu Bernhardt College

Bafal, Kathmandu.

Net Centric Computing Lab Report

Submitted by:

Name: Anusha Bhandari

Roll No: 23593/076

Submitted to:

Department of Computer Science and Information Technology

Submission Date: 2080-5-17

Signature

1. Write a program to convert input strings from lower to upper and upper to lower case

```
using System;

namespace ConsoleApp1
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter a string:");
            string input = Console.ReadLine();

            Console.WriteLine("Choose an operation:");
            Console.WriteLine("1. Convert to upper case");
            Console.WriteLine("2. Convert to lower case");

            int choice;
            if (int.TryParse(Console.ReadLine(), out choice))
            {
                string convertedString = "";

                switch (choice)
                {
                    case 1:
                        convertedString = input.ToUpper();
                        break;
                    case 2:
                        convertedString = input.ToLower();
                        break;
                    default:
                        Console.WriteLine("Invalid choice.");
                        break;
                }

                Console.WriteLine("Converted string: " +
convertedString);
            }
            else
            {
                Console.WriteLine("Invalid choice.");
            }

            // Wait for user input before closing the console window
            Console.WriteLine("Press Enter to exit...");
            Console.ReadLine();
        }
    }
}
```

Output:

C:\WINDOWS\system32\cmd.exe

```
Enter a string:
jay shree krishna
Choose an operation:
1. Convert to upper case
2. Convert to lower case
1
Converted string: JAY SHREE KRISHNA
Press Enter to exit...
```

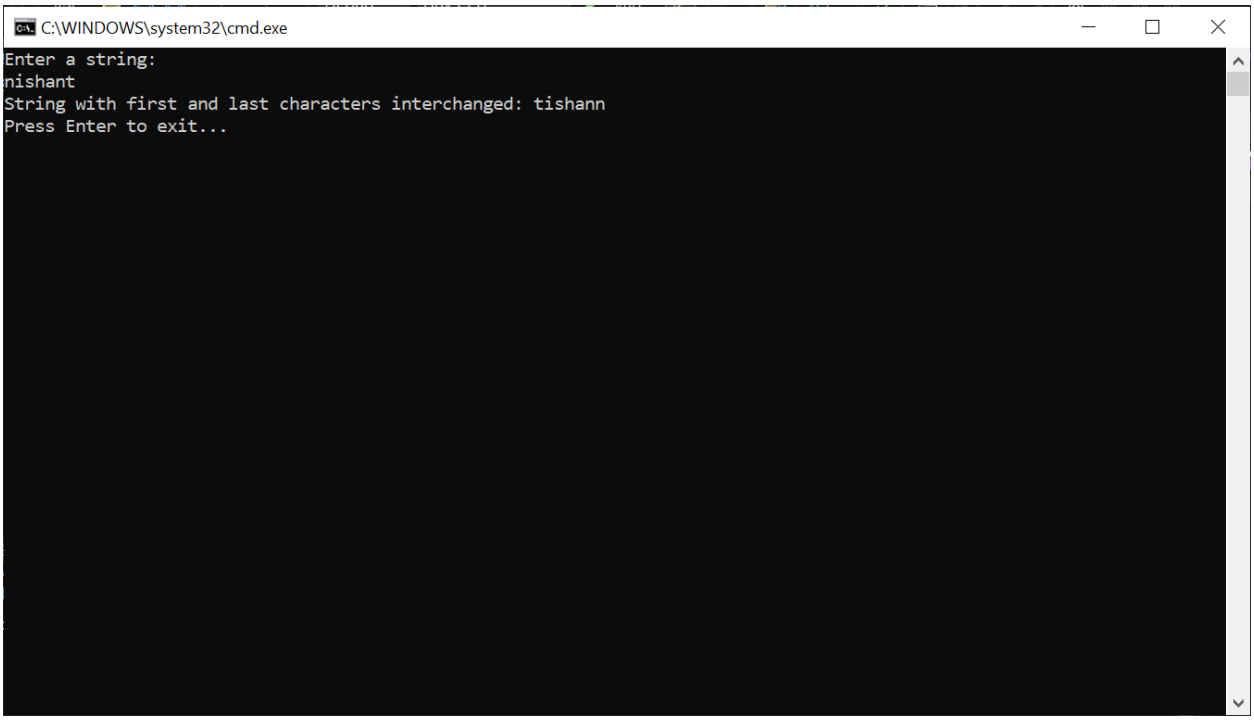
C:\WINDOWS\system32\cmd.exe

```
Enter a string:
RADHE SHYAM
Choose an operation:
1. Convert to upper case
2. Convert to lower case
2
Converted string: radhe shyam
Press Enter to exit...
```

2. Write a program to create a new string from a given string where first and last characters will be interchanged.

```
using System;

namespace StringInterchange
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter a string:");
            string input = Console.ReadLine();

            if (input.Length >= 2)
            {
                char[] charArray = input.ToCharArray();

                // Swap the first and last characters
                char firstChar = charArray[0];
                charArray[0] = charArray[input.Length - 1];
                charArray[input.Length - 1] = firstChar;

                string result = new string(charArray);
                Console.WriteLine("String with first and last
characters interchanged: " + result);
            }
            else
            {
                Console.WriteLine("Input string must contain at least
2 characters.");
            }
            Console.WriteLine("Press Enter to exit...");
            Console.ReadLine();
        }
    }
}
```

Output:

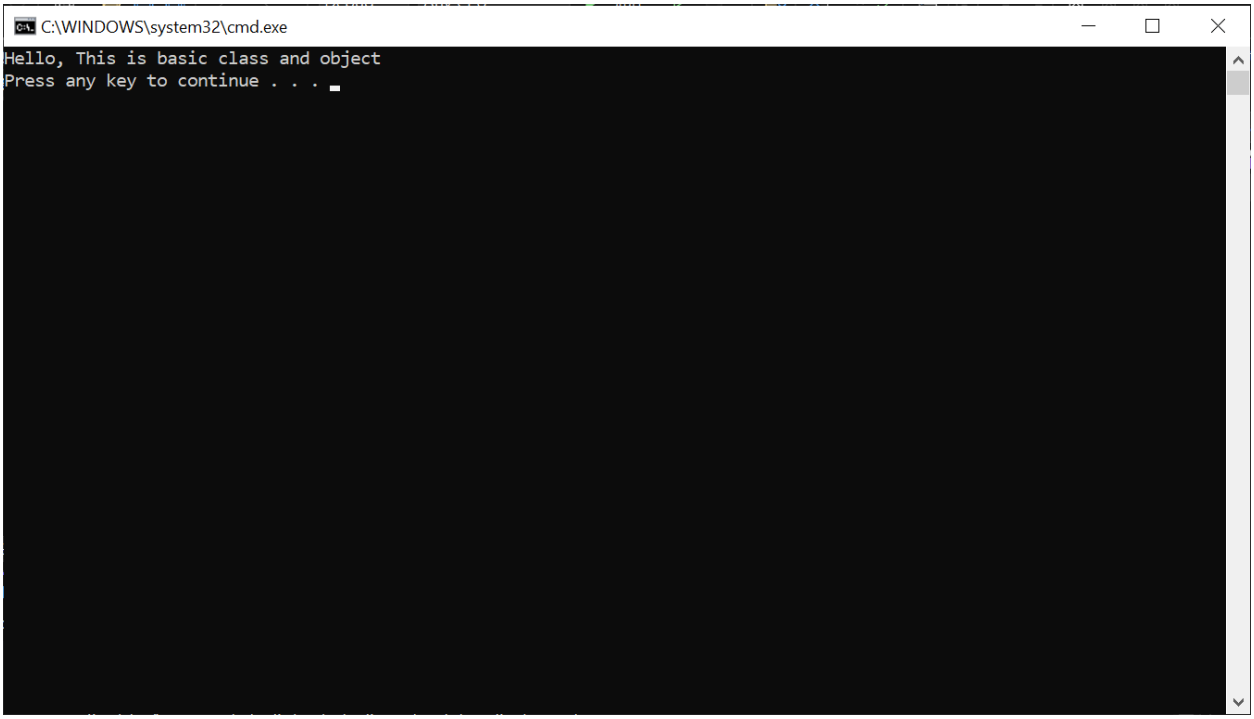


3. Write a program to demonstrate the basics of class and object.

```
using System;

public class Person
{
    public string Name { get; set; }
    public int Age { get; set; }

    public Person(string name, int age)
    {
        Name = name;
        Age = age;
    }

    public void SayHello()
    {
        Console.WriteLine("Hello, This is basic class and object",
Name, Age);
    }
}

public class Program
{
```

```
        public static void Main(string[] args)
        {
            Person person1 = new Person("Anusha Bhandari", 22);
            person1.SayHello();
        }
    }
```

Output:



4. Write a program to illustrate encapsulation with properties and indexers.

```
using System;

public class Student
{
    // Private fields
    private string name;
    private int age;
    private string[] subjects;

    // Properties
    public string Name
    {
        get { return name; }
        set
        {
```

```
            if (!string.IsNullOrEmpty(value))
            {
                name = value;
            }
        }
    }

    public int Age
    {
        get { return age; }
        set
        {
            if (value >= 0)
            {
                age = value;
            }
        }
    }

    // Indexer
    public string this[int index]
    {
        get
        {
            if (index >= 0 && index < subjects.Length)
            {
                return subjects[index];
            }
            return "Invalid Index";
        }
        set
        {
            if (index >= 0 && index < subjects.Length)
            {
                subjects[index] = value;
            }
        }
    }

    // Constructor
    public Student(string name, int age, int numSubjects)
    {
        Name = name;
        Age = age;
        subjects = new string[numSubjects];
    }
}

class Program
{
    static void Main(string[] args)
    {
        // Create a new student object
```

```
        Student student = new Student("Nila", 20, 3);

        // Set properties
        student.Name = "Anusha";
        student.Age = 22;

        // Set subjects using the indexer
        student[0] = "Digital Logic";
        student[1] = "Numerical Method";
        student[2] = "Cryptography";

        // Access properties and indexer
        Console.WriteLine($"Name: {student.Name}");
        Console.WriteLine($"Age: {student.Age}");
        Console.WriteLine($"Subject at index 2: {student[2]}");

        // Try accessing an out-of-bounds index
        Console.WriteLine($"Subject at index 5: {student[5]}");
        Console.WriteLine("Press Enter to exit...");
        Console.ReadLine();

    }
}
```

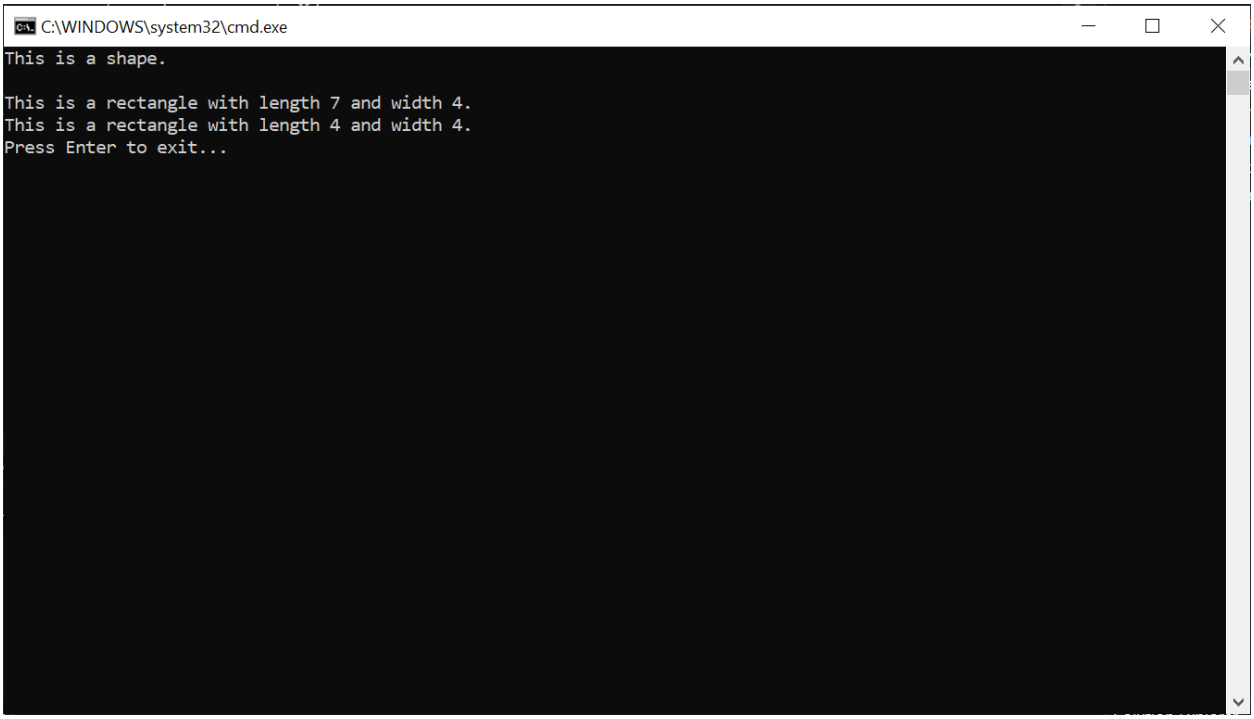
Output:

5. Write a program that reflects the overloading and overriding of constructor and function.

```
using System;

class Shape
{
    public virtual void Display()
    {
        Console.WriteLine("This is a shape.");
    }
}

class Rectangle : Shape
{
    private double length;
    private double width;

    public Rectangle(double length, double width)
    {
        this.length = length;
        this.width = width;
    }

    public Rectangle(double sideLength)
    {
        this.length = sideLength;
        this.width = sideLength;
    }

    public override void Display()
    {
        Console.WriteLine("This is a rectangle with length {0} and
width {1}.", length, width);
    }
}

class Program
{
    static void Main()
    {
        Shape shape = new Shape();
        shape.Display();

        Console.WriteLine();

        Rectangle rectangle1 = new Rectangle(7, 4);
        rectangle1.Display();

        Rectangle rectangle2 = new Rectangle(4);
        rectangle2.Display();
```

```
            Console.WriteLine("Press Enter to exit...");
            Console.ReadLine();

        }
    }
```

Output:

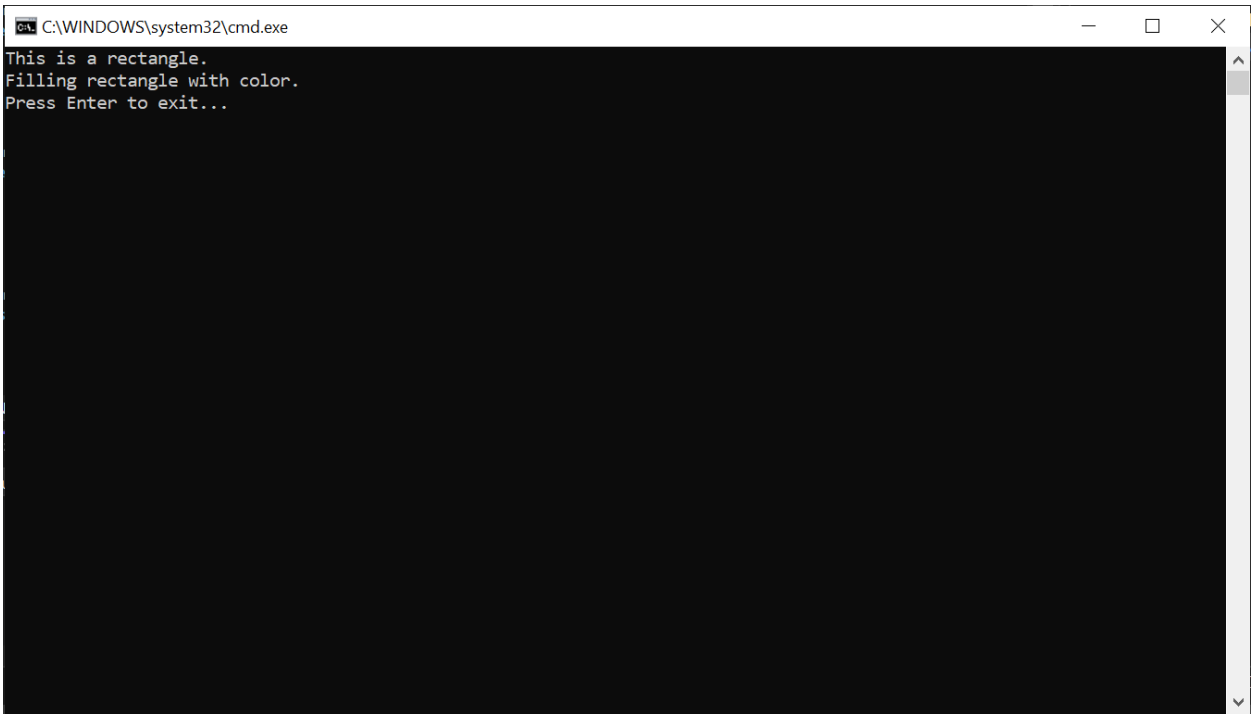


6. Write a program to implement multiple inheritance with the use of interfaces.

```
using System;

interface IShape
{
    void Display();
}

interface IColor
{
    void FillColor();
}

class Rectangle : IShape, IColor
{
    public void Display()
    {
        Console.WriteLine("This is a rectangle.");
```

```
        }

        public void FillColor()
        {
            Console.WriteLine("Filling rectangle with color.");
        }
    }

    class Program
    {
        static void Main()
        {
            Rectangle rectangle = new Rectangle();
            rectangle.Display();
            rectangle.FillColor();

            Console.WriteLine("Press Enter to exit...");
            Console.ReadLine();
        }
    }
```

Output:



7. Write a program to show how to handle exception in C#.

```
using System;

class Program
{
    static void Main(string[] args)
    {
        try
        {
            Console.WriteLine("Enter a number:");
            string userInput = Console.ReadLine();
            int number = int.Parse(userInput);

            int result = 20 / number;

            Console.WriteLine($"Result: {result}");
        }
        catch (DivideByZeroException)
        {
            Console.WriteLine("Error: Division by zero is not
allowed.");
        }
        catch (FormatException)
        {
            Console.WriteLine("Error: Invalid input. Please enter a
valid number.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An unexpected error occurred:
{ex.Message}");
        }
        finally
        {
            Console.WriteLine("Program execution completed.");
        }

        Console.WriteLine("Press Enter to exit...");
        Console.ReadLine();
    }
}
```

Output:

```
C:\WINDOWS\system32\cmd.exe
Enter a number:
4
Result: 5
Program execution completed.
Press Enter to exit...
```

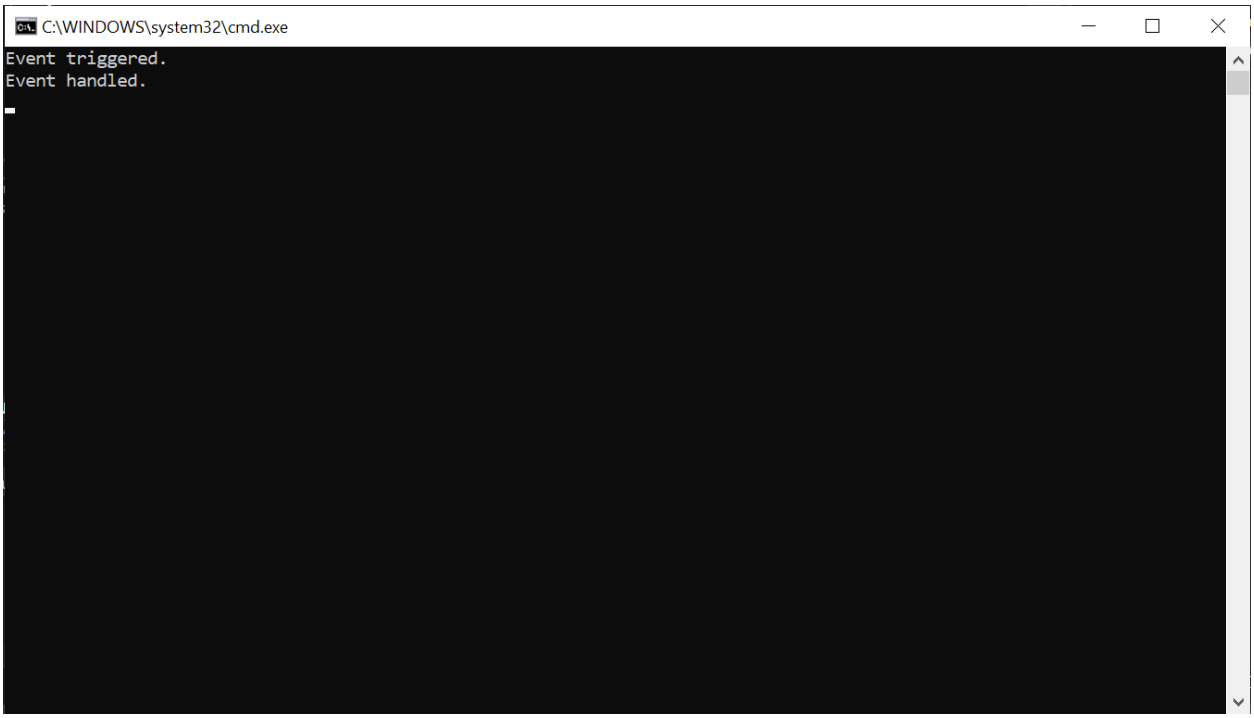
```
C:\WINDOWS\system32\cmd.exe
Enter a number:
hjbh
Error: Invalid input. Please enter a valid number.
Program execution completed.
Press Enter to exit...
```

```
C:\WINDOWS\system32\cmd.exe
Enter a number:
0
Error: Division by zero is not allowed.
Program execution completed.
Press Enter to exit...
```

8. Write a program to demonstrate use of Delegate and Events.

```
using System;


delegate void EventHandler();

class EventPublisher
{
    public event EventHandler MyEvent;

    public void TriggerEvent()
    {

        if (MyEvent != null)
        {
            Console.WriteLine("Event triggered.");

            MyEvent.Invoke();
        }
    }
}

class EventSubscriber
{

    public void HandleEvent()
    {
        Console.WriteLine("Event handled.");
    }
}

class Program
{
    static void Main()
    {

        EventPublisher publisher = new EventPublisher();

        EventSubscriber subscriber = new EventSubscriber();


        publisher.MyEvent += subscriber.HandleEvent;


        publisher.TriggerEvent();

        Console.ReadLine();
    }
}
```

Output:

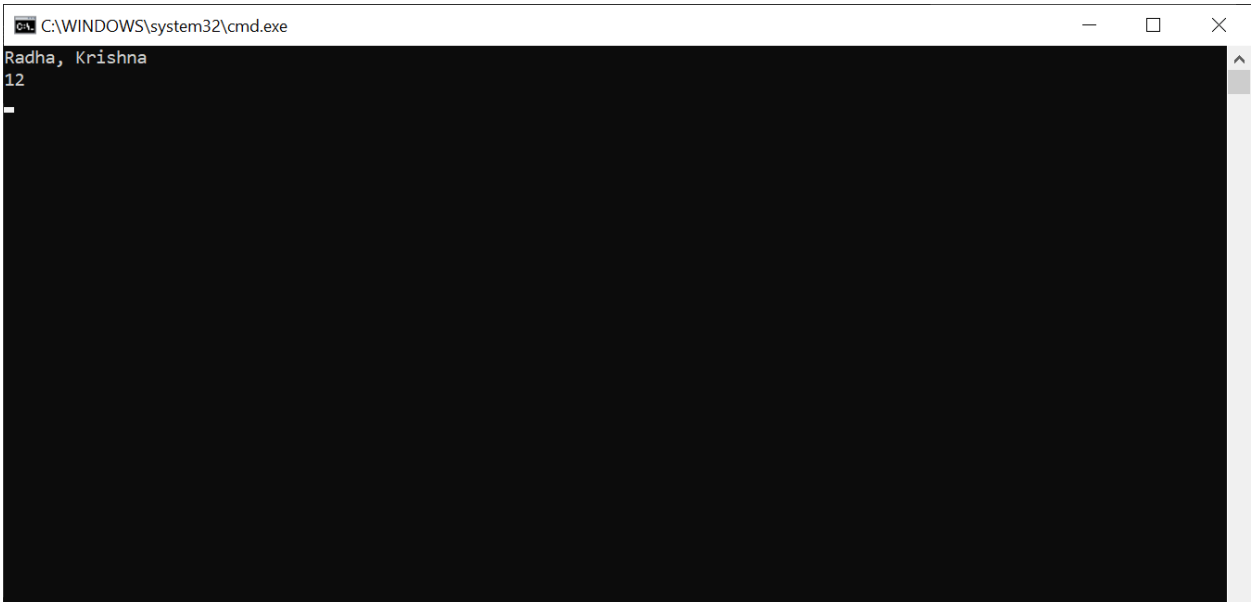


9. Write a program to show the use of generic classes and methods.

```
using System;

public class GenericClass<T>
{
    private T data;

    public GenericClass(T data)
    {
        this.data = data;
    }

    public T getData()
    {
        return data;
    }

    public void setData(T data)
    {
        this.data = data;
    }
}
```

```
public class MainClass
{
    public static void Main()
    {
        GenericClass<string> stringClass = new
GenericClass<string>("Radha, Krishna");
        string data = stringClass.getData();
        Console.WriteLine(data);

        GenericClass<int> intClass = new GenericClass<int>(12);
        int number = intClass.getData();
        Console.WriteLine(number);

        Console.ReadLine();
    }
}
```

Output:

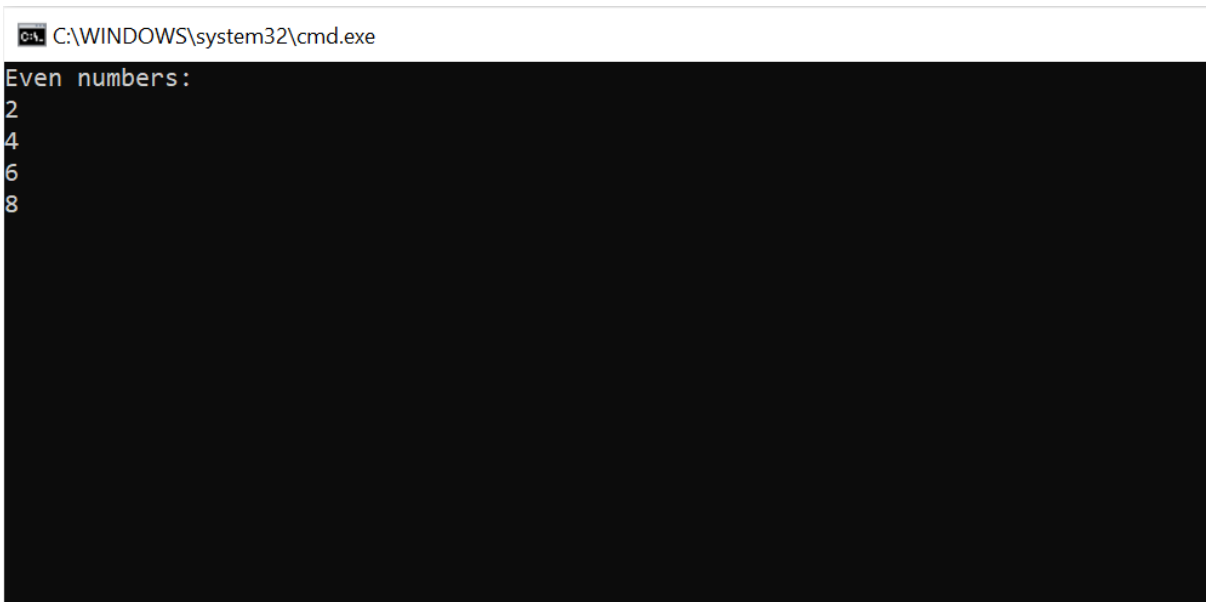


10. Write a program to demonstrate the use of the method as a condition in the LINQ.

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

class Program
{
    static void Main()
    {
        List<int> numbers = new List<int> { 1, 2, 3, 4, 5, 6, 7, 8 };
```

```
            // Use LINQ to filter numbers using a method as a condition
            IEnumerable<int> filteredNumbers = numbers.Where(IsEven);

            Console.WriteLine("Even numbers:");
            foreach (int number in filteredNumbers)
            {
                Console.WriteLine(number);
            }
            Console.ReadLine();
        }

        static bool IsEven(int number)
        {
            return number % 2 == 0;
        }
    }
```

Output:

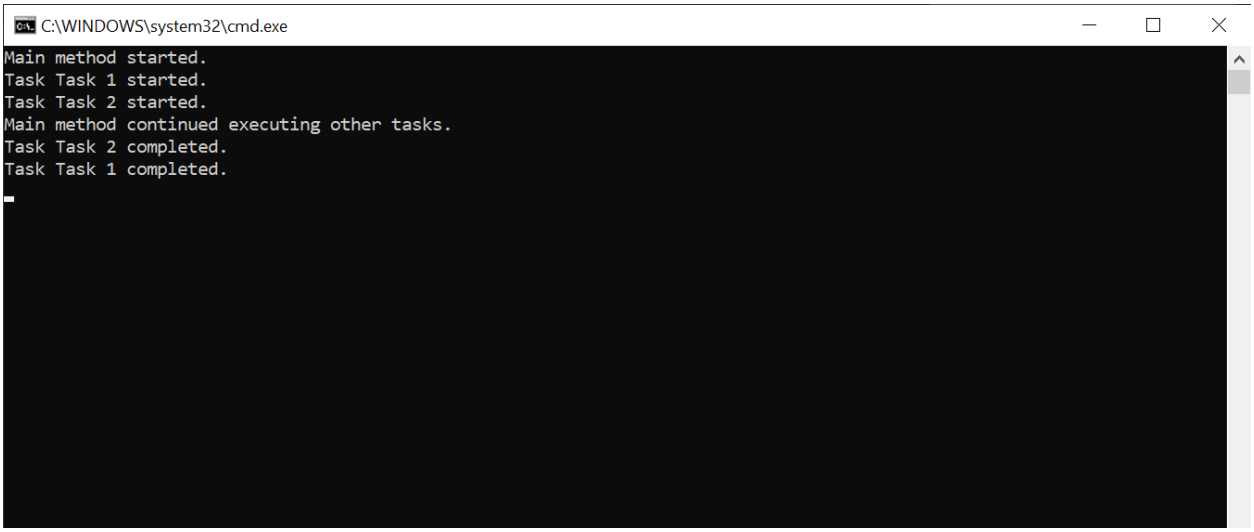


11. Demonstrate Asynchronous programming with async, await. Task in C#.

```
using System;
using System.Threading.Tasks;

class Program
{
    static async Task Main()
    {
        Console.WriteLine("Main method started.");

        Task task1 = TaskMethod("Task 1");
```

```
        Task task2 = TaskMethod("Task 2");


        Console.WriteLine("Main method continued executing other
tasks.");

        await Task.WhenAll(task1, task2);

        Console.WriteLine("Main method completed.");

    }

    static async Task TaskMethod(string name)
    {
        Console.WriteLine($"Task {name} started.");

        await Task.Delay(2000);

        Console.WriteLine($"Task {name} completed.");
        Console.ReadLine();
    }
}
```

Output:



12 Write a program to demonstrate dependency injection in asp . net core.

Program.cs

```
using DependencyInjection.Models;

var builder = WebApplication.CreateBuilder(args);
```

```
builder.Services.AddTransient<IRepository, Repository>();


// Add services to the container.
builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this
for production scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

index.cshtml

```
@if (ViewData.Count > 0)
{
    <table class="table table-bordered table-sm table-striped">
        @foreach (var kvp in ViewData)
        {
            <tr><td>@kvp.Key</td><td>@kvp.Value</td></tr>
        }
    </table>
}
<table class="table table-bordered table-sm table-striped">
    <thead>
        <tr><th>Name</th><th>Price</th></tr>
    </thead>
    <tbody>
        @if (Model == null)
        {
            <tr><td colspan="3" class="text-center">No Model
Data</td></tr>
```

```
        }
        else
        {
            @foreach (var p in Model)
            {
                <tr>
                    <td>@p.Name</td>
                    <td>@string.Format("{0:C2}", p.Price)</td>
                </tr>
            }
        }
    </tbody>
</table>
```

HomeController.cs

```
using DependencyInjection.Models;
using Microsoft.AspNetCore.Mvc;
using System.Diagnostics;

namespace DependencyInjection.Controllers
{
    public class HomeController : Controller
    {
        private IRepository repository;
        public HomeController(IRepository repo)
        {
            repository = repo;
        }

        public IActionResult Index()
        {
            return View(repository.Products);
        }
    }
}
```

Repository.cs

```
namespace DependencyInjection.Models
{
    public class Repository : IRepository
    {
        private Dictionary<string, Product> products;
        public Repository()
        {
            products = new Dictionary<string, Product>();
```

```
            new List<Product> {
                new Product { Name = "Pendrive", Price = 500 },
                new Product { Name = "Hard Disk", Price = 1000 },
                new Product { Name = "SSD", Price = 5000 }
            }.ForEach(p => AddProduct(p));
        }

        public IEnumerable<Product> Products => products.Values;
        public Product this[string name] => products[name];
        public void AddProduct(Product product) =>
products[product.Name] = product;
        public void DeleteProduct(Product product) =>
products.Remove(product.Name);
    }
}
```

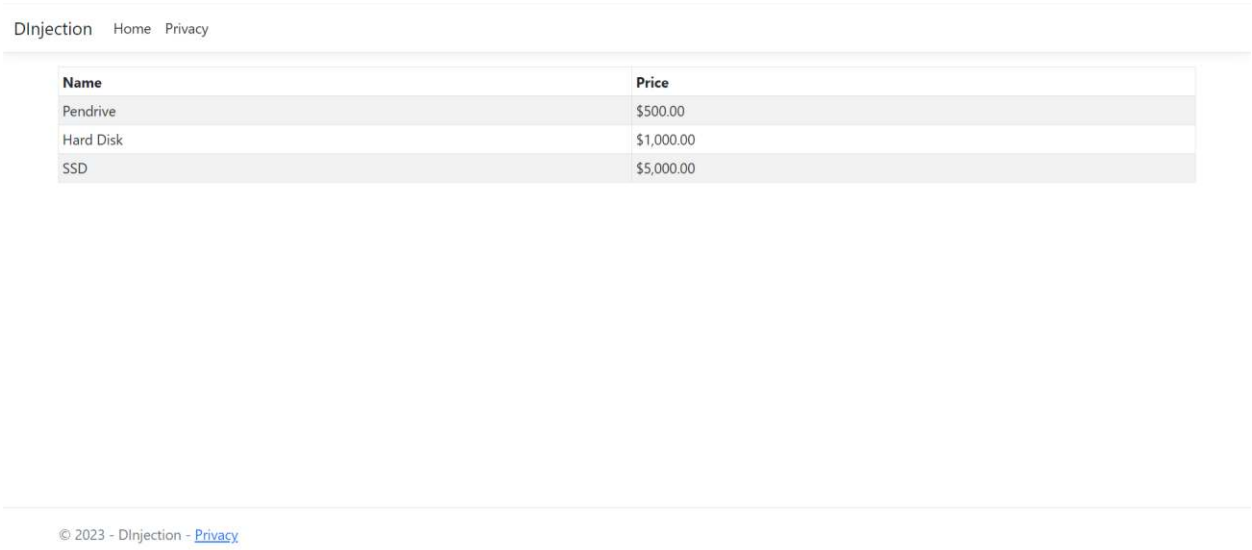
IRepository.cs

```
namespace DependencyInjection.Models
{
    public interface IRepository
    {
        IEnumerable<Product> Products { get; }

        Product this[string name] { get; }

        void AddProduct(Product product);

        void DeleteProduct(Product product);
    }
}
```

Product.cs

```
namespace DependencyInjection.Models
{
    public class Product
    {
        public string Name { get; set; }
        public decimal Price { get; set; }
    }
      }
```

Output:



13. Create an ASP.NET Core application to perform CRUD operation using ADO.NET.

Controllers/HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
//using Microsoft.Data.SqlClient;
using System.Linq;
using System.Threading.Tasks;
using CRUDADO.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;

namespace CRUDADO.Controllers
{
    public class HomeController : Controller
    {
        public IConfiguration Configuration { get; }

        public HomeController(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IActionResult Index()
        {
            List<Teacher> teacherList = new List<Teacher>();
```

```
            string connectionString =
Configuration["ConnectionStrings:DefaultConnection"];
            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                //SqlDataReader
                connection.Open();

                string sql = "Select * From Teacher";
                SqlCommand command = new SqlCommand(sql, connection);

                using (SqlDataReader dataReader =
command.ExecuteReader())
                {
                    while (dataReader.Read())
                    {
                        Teacher teacher = new Teacher();
                        teacher.Id =
Convert.ToInt32(dataReader["Id"]);
                        teacher.Name =
Convert.ToString(dataReader["Name"]);
                        teacher.Skills =
Convert.ToString(dataReader["Skills"]);
                        teacher.TotalStudents =
Convert.ToInt32(dataReader["TotalStudents"]);
                        teacher.Salary =
Convert.ToDecimal(dataReader["Salary"]);
                        teacher.AddedOn =
Convert.ToDateTime(dataReader["AddedOn"]);

                        teacherList.Add(teacher);
                    }
                }

                connection.Close();
            }
            return View(teacherList);
        }

        public IActionResult Create()
        {
            return View();
        }

        [HttpPost]
        public IActionResult Create(Teacher teacher)
        {
            if (ModelState.IsValid)
            {
                string connectionString =
Configuration["ConnectionStrings:DefaultConnection"];
```

```
                using (SqlConnection connection = new
SqlConnection(connectionString))
                {
                    string sql = $"Insert Into Teacher (Name, Skills,
TotalStudents, Salary) Values ('{teacher.Name}',
'{teacher.Skills}','{teacher.TotalStudents}','{teacher.Salary}')";

                    using (SqlCommand command = new SqlCommand(sql,
connection))
                    {
                        command.CommandType = CommandType.Text;

                        connection.Open();
                        command.ExecuteNonQuery();
                        connection.Close();
                    }
                    return RedirectToAction("Index");
                }
            }
            else
                return View();
        }

        public IActionResult Update(int id)
        {
            string connectionString =
Configuration["ConnectionStrings:DefaultConnection"];

            Teacher teacher = new Teacher();
            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                string sql = $"Select * From Teacher Where Id='{id}'";
                SqlCommand command = new SqlCommand(sql, connection);

                connection.Open();

                using (SqlDataReader dataReader =
command.ExecuteReader())
                {
                    while (dataReader.Read())
                    {
                        teacher.Id =
Convert.ToInt32(dataReader["Id"]);
                        teacher.Name =
Convert.ToString(dataReader["Name"]);
                        teacher.Skills =
Convert.ToString(dataReader["Skills"]);
                        teacher.TotalStudents =
Convert.ToInt32(dataReader["TotalStudents"]);
                        teacher.Salary =
Convert.ToDecimal(dataReader["Salary"]);
```

```
                        teacher.AddedOn =
Convert.ToDateTime(dataReader["AddedOn"]);
                    }
                }

                connection.Close();
            }
            return View(teacher);
        }

        [HttpPost]
        [ActionName("Update")]
        public IActionResult Update_Post(Teacher teacher)
        {
            string connectionString =
Configuration["ConnectionStrings:DefaultConnection"];
            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                string sql = $"Update Teacher SET
Name='{teacher.Name}', Skills='{teacher.Skills}',
TotalStudents='{teacher.TotalStudents}', Salary='{teacher.Salary}'
Where Id='{teacher.Id}'";
                using (SqlCommand command = new SqlCommand(sql,
connection))
                {
                    connection.Open();
                    command.ExecuteNonQuery();
                    connection.Close();
                }
            }

            return RedirectToAction("Index");
        }

        [HttpPost]
        public IActionResult Delete(int id)
        {
            string connectionString =
Configuration["ConnectionStrings:DefaultConnection"];
            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                string sql = $"Delete From Teacher Where Id='{id}'";
                using (SqlCommand command = new SqlCommand(sql,
connection))
                {
                    connection.Open();
                    try
                    {
                        command.ExecuteNonQuery();
                    }
                    catch (SqlException ex)
```

```
                    {
                        ViewBag.Result = "Operation got error:" +
ex.Message;
                    }
                    connection.Close();
                }
            }

            return RedirectToAction("Index");
        }
    }
}
```

CustomValidation/SkillsValidate.cs

```
using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CRUDADO.CustomValidation
{
    public class SkillsValidate : Attribute, IModelValidator
    {
        public string[] Allowed { get; set; }
        public string ErrorMessage { get; set; }
        public IEnumerable<ModelValidationResult>
Validate(ModelValidationContext context)
        {

            if (Allowed.Contains(context.Model as string))
                return Enumerable.Empty<ModelValidationResult>();
            else
                return new List<ModelValidationResult> {
                    new ModelValidationResult("", ErrorMessage)
                };
        }
    }
}
```

Models/Teacher.cs

```
using CRUDADO.CustomValidation;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace CRUDADO.Models
```

```
{
    public class Teacher
    {
        public int Id { get; set; }

        [Required]
        public string Name { get; set; }

        [Required]
        [SkillsValidate(Allowed = new string[] { "ASP.NET Core",
"ASP.NET MVC", "ASP.NET Web Forms" }, ErrorMessage = "Your skills are
invalid")]
        public string Skills { get; set; }

        [Range(5, 50)]
        public int TotalStudents { get; set; }

        [Required]
        public decimal Salary { get; set; }

        public DateTime AddedOn { get; set; }
    }
}
```

Views/Home/Create.cshtml

```
@model Teacher

@{
    Layout = "_Layout";
    var title = "CREATE Teacher";
    ViewData["Title"] = title;
}

<style>
    .input-validation-error {
        border-color: red;
    }
</style>

<h2>@title</h2>

<div asp-validation-summary="ModelOnly" class="text-danger"></div>
<form class="m-1 p-1" method="post">
    <div class="form-group">
        <label asp-for="Name"></label>
        <input asp-for="Name" class="form-control" />
        <span asp-validation-for="Name" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Skills"></label>
        <input asp-for="Skills" type="text" class="form-control" />
```

```
        <span asp-validation-for="Skills" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="TotalStudents"></label>
        <input asp-for="TotalStudents" type="text" class="form-
control" />
        <span asp-validation-for="TotalStudents" class="text-
danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Salary"></label>
        <input asp-for="Salary" type="text" class="form-control" />
        <span asp-validation-for="Salary" class="text-danger"></span>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
</form>

<script src="~/lib/jquery/jquery.min.js"></script>
<script src="~/lib/jquery-validate/jquery.validate.min.js"></script>
<script src="~/lib/jquery-validation-
unobtrusive/jquery.validate.unobtrusive.min.js"></script>
```

Views/Home/index.cshtml

```
@model IEnumerable<Teacher>

@{
    Layout = "_Layout";
    var title = "READ Teacher";
    ViewData["Title"] = title;
}

<h2>@title</h2>

<h3><a asp-action="Create" class="btn btn-sm btn-
secondary">Create</a></h3>
<table class="table table-bordered table-sm table-striped">
    <thead>
        <tr><th>Id</th><th>Name</th><th>Skills</th><th>Total
Students</th><th>Salary</th><th>Added
On</th><th>Update</th><th>Delete</th></tr>
    </thead>
    <tbody>
        @if (Model == null)
        {
            <tr><td colspan="7" class="text-center">No Model
Data</td></tr>
        }
        else
        {
            @foreach (var p in Model)
            {
```

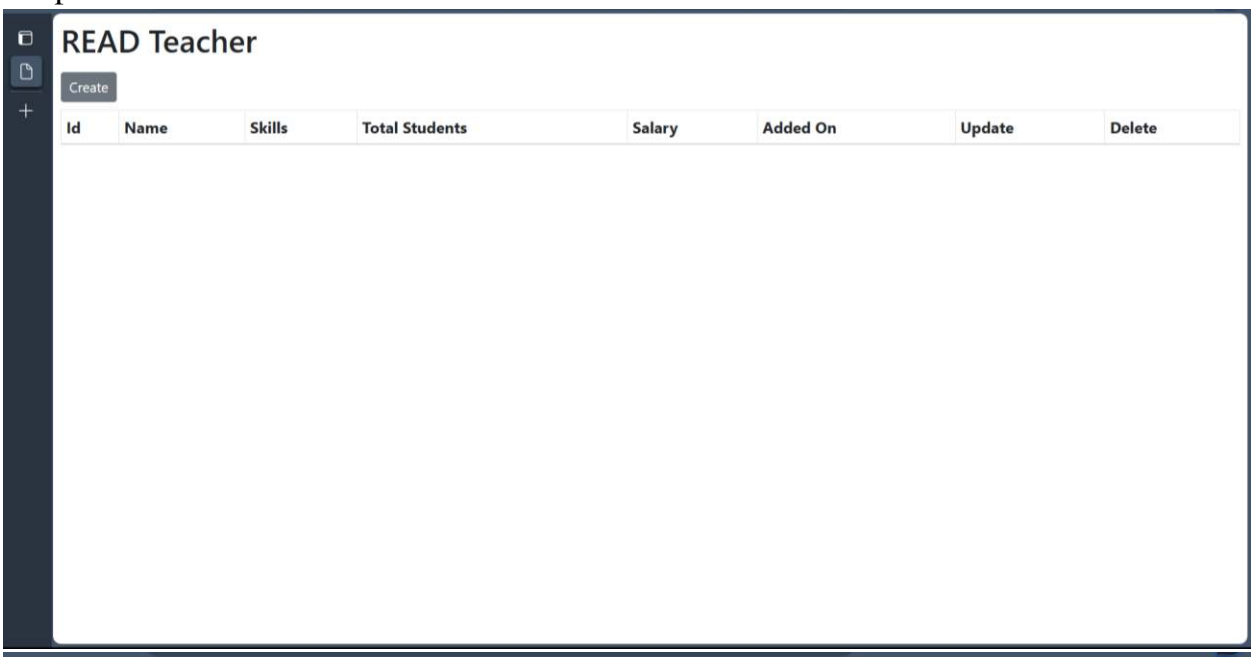
```
                <tr>
                    <td>@p.Id</td>
                    <td>@p.Name</td>
                    <td>@p.Skills</td>
                    <td>@p.TotalStudents</td>
                    <td>@string.Format(new
System.Globalization.CultureInfo("en-US"), "{0:C2}", p.Salary)</td>
                    <td>@string.Format("{0:dddd, dd MMMM yyyy}",
p.AddedOn)</td>
                    <td><a asp-action="Update" asp-route-
id="@p.Id">Update</a></td>
                    <td>
                        <form asp-action="Delete" method="post" asp-
route-id="@p.Id">
                            <button>Delete</button>
                        </form>
                    </td>
                </tr>
            }
        }
    </tbody>
</table>
```

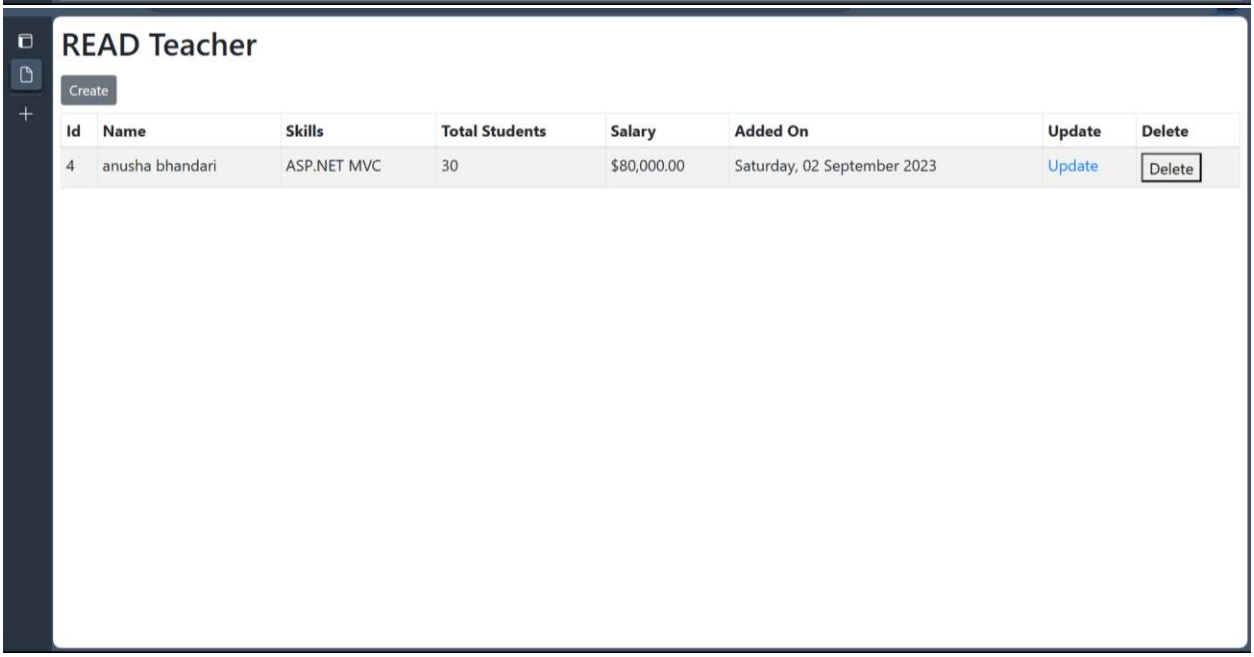
Views/Home/Update.cshtml

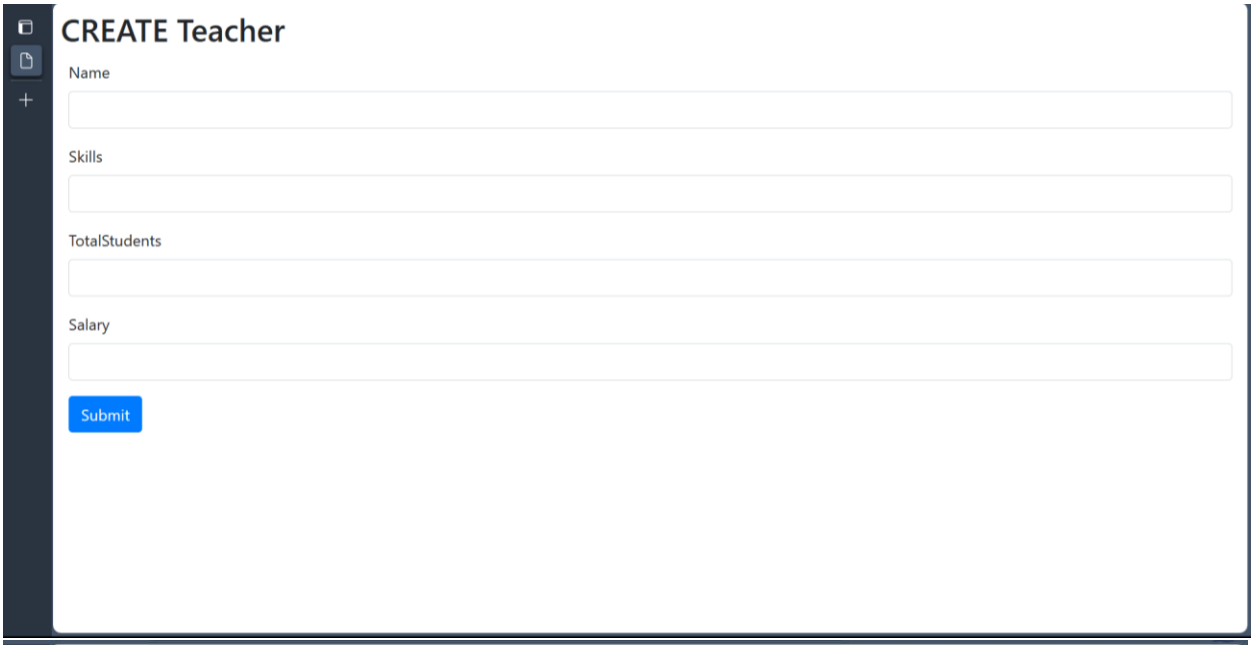
```
@model Teacher

@{
    Layout = "_Layout";
    var title = "UPDATE Teacher";
    ViewData["Title"] = title;
}

<style>
    .input-validation-error {
        border-color: red;
    }
</style>

<h2>@title</h2>

<div asp-validation-summary="ModelOnly" class="text-danger"></div>
<form class="m-1 p-1" method="post">
    <div class="form-group">
        <label asp-for="Id"></label>
        <input asp-for="Id" type="text" readonly class="form-control"
/>
    </div>
    <div class="form-group">
```

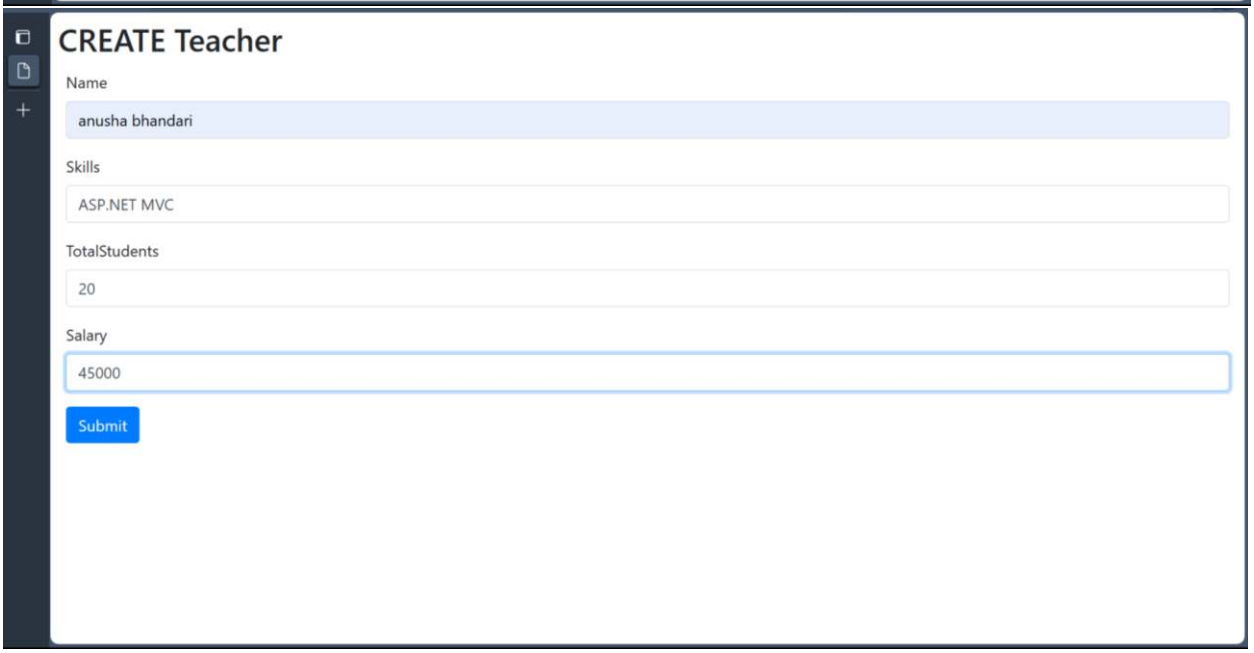
```
        <label asp-for="Name"></label>
        <input asp-for="Name" type="text" class="form-control" />
        <span asp-validation-for="Name" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Skills"></label>
        <input asp-for="Skills" type="text" class="form-control" />
        <span asp-validation-for="Skills" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="TotalStudents"></label>
        <input asp-for="TotalStudents" type="text" class="form-
control" />
        <span asp-validation-for="TotalStudents" class="text-
danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Salary"></label>
        <input asp-for="Salary" type="text" class="form-control" />
        <span asp-validation-for="Salary" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="AddedOn"></label>
        <input asp-for="AddedOn" type="text" class="form-control" asp-
format="{0:d}" />
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
</form>

<script src="~/lib/jquery/jquery.min.js"></script>
<script src="~/lib/jquery-validate/jquery.validate.min.js"></script>
<script src="~/lib/jquery-validation-
unobtrusive/jquery.validate.unobtrusive.min.js"></script>
```

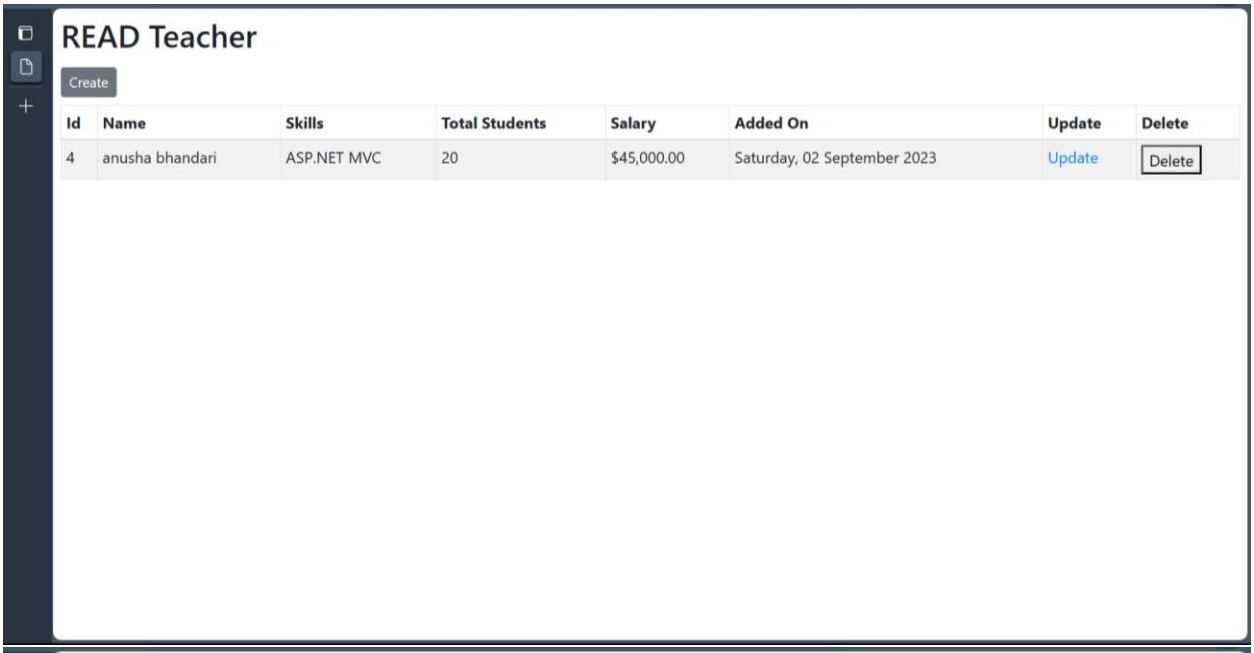
Output:

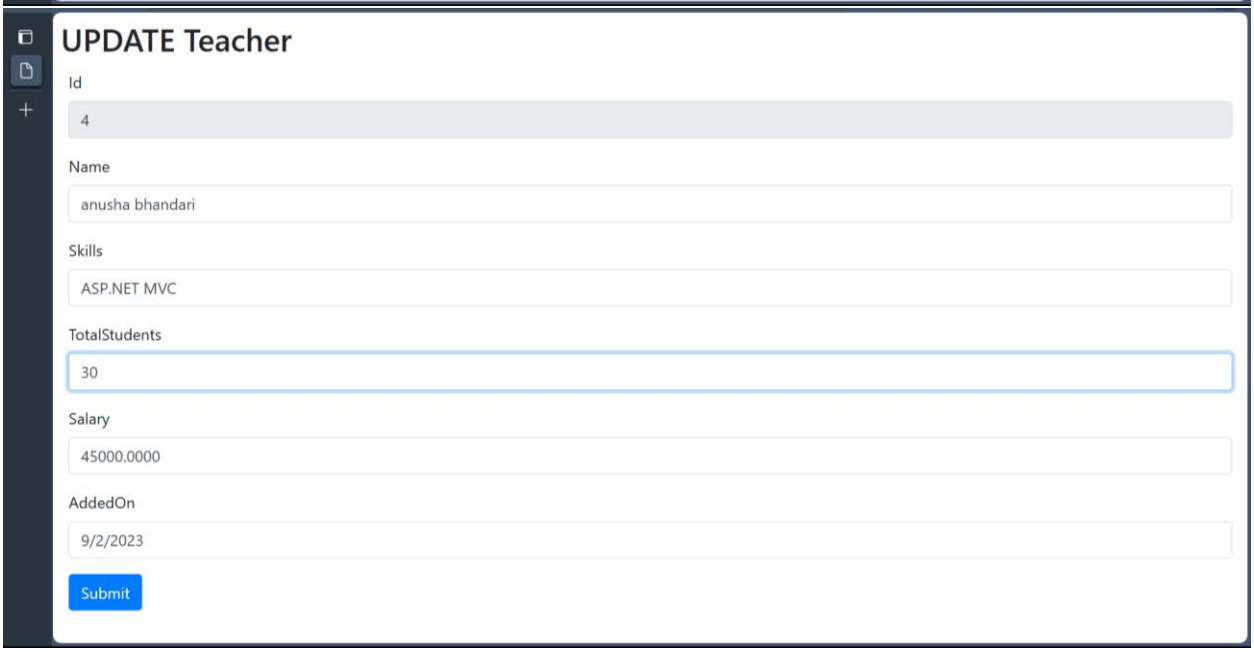












14. Write a program to store and display employee information using DbContext.

Controller/EmployeeController.cs

```
using EmployeeCRUD.Data;
using EmployeeCRUD.Models;
using Microsoft.AspNetCore.Mvc;

namespace EmployeeCRUD.Controllers
{
    public class EmployeeController : Controller
```

```
{
    private readonly ApplicationDbContext _context;
    public EmployeeController(ApplicationDbContext context)
    {
        _context = context;
    }
    public IActionResult Index()
    {
        IEnumerable<Employee> objCatlist = _context.Employees;
        return View(objCatlist);
    }

    public IActionResult Create()
    {
        return View();
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public IActionResult Create(Employee empobj)
    {
        if (ModelState.IsValid)
        {
            var cdate=DateTime.Now;
            empobj.RecordCreatedOn = cdate;

            _context.Employees.Add(empobj);
            _context.SaveChanges();
            TempData["ResultOk"] = "Record Added Successfully !";
            return RedirectToAction("Index");
        }

        return View(empobj);
    }

    public IActionResult Edit(int? id)
    {
        if (id == null || id == 0)
        {
            return NotFound();
        }
        var empfromdb = _context.Employees.Find(id);

        if (empfromdb == null)
        {
            return NotFound();
        }
        return View(empfromdb);
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public IActionResult Edit(Employee empobj)
```

```
        {
            if (ModelState.IsValid)
            {
                _context.Employees.Update(empobj);
                _context.SaveChanges();
                TempData["ResultOk"] = "Data Updated Successfully !";
                return RedirectToAction("Index");
            }

            return View(empobj);
        }

        public IActionResult Delete(int? id)
        {
            if (id == null || id == 0)
            {
                return NotFound();
            }
            var empfromdb = _context.Employees.Find(id);

            if (empfromdb == null)
            {
                return NotFound();
            }
            return View(empfromdb);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult DeleteEmp(int? id)
        {
            var deleterecord = _context.Employees.Find(id);
            if (deleterecord == null)
            {
                return NotFound();
            }
            _context.Employees.Remove(deleterecord);
            _context.SaveChanges();
            TempData["ResultOk"] = "Data Deleted Successfully !";
            return RedirectToAction("Index");
        }


    }
}
```

Data/ApplicationDBcontext.cs

```
using EmployeeCRUD.Models;
using Microsoft.EntityFrameworkCore;

namespace EmployeeCRUD.Data
```

```
{
    public class ApplicationDbContext:DbContext
    {
        public
ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options):base(options)
        {

        }

        public DbSet<Employee> Employees { get; set; }
    }
}
```

Models/Employee.cs

```
using System.ComponentModel.DataAnnotations;

namespace EmployeeCRUD.Models
{
    public class Employee
    {
        [Key]
        public int Id { get; set; }
        [Required]
        [Display(Name ="Employee Name")]
        public string Name { get; set; }
        public string Designation { get; set; }
        [DataType(DataType.MultilineText)]
        public string Address { get; set; }
        public DateTime? RecordCreatedOn { get; set; }

    }
}
```

Views/Employee/Create.cshtml

```
@model Employee


<div class="container shadow p-5">
    <div class="row pb-2">
        <h2>Add Employee</h2>
    </div>

    <form method="post">
        <div asp-validation-summary="All"></div>

        <div class="form-row">
            <div class="form-group col-md-6">
```

```
                <label asp-for="Name">Employee Name</label>
                <input type="text" class="form-control mb-3" asp-
for="Name" placeholder="Enter Name">
                <span asp-validation-for="Name" class=" alert-
danger"></span>
            </div>
            <div class="form-group col-md-6">
                <label asp-for="Designation">Designation</label>
                <input type="text" class="form-control mb-3" asp-
for="Designation" placeholder="Enter Designation">
                <span asp-validation-for="Designation" class=" alert-
danger"></span>
            </div>
        </div>

        <div class="form-row">
            <div class="form-group col-md-6">
                <label asp-for="Address">Address</label>
                <input type="text" class="form-control mb-3" asp-
for="Address" placeholder="Enter Address">
                <span asp-validation-for="Address" class=" alert-
danger"></span>
            </div>
            <div class="form-group col-md-6 mb-3">
                <label asp-for="RecordCreatedOn">Created On</label>
                <input type="datetime-local" class="form-control" asp-
for="RecordCreatedOn">
                <span asp-validation-for="RecordCreatedOn" class="
alert-danger"></span>
            </div>
        </div>


        <button type="submit" class="btn btn-lg btn-primary p-2"><i
class="bi bi-file-plus-fill"></i>Save</button>
        <a asp-controller="Employee" asp-action="Index" class="btn
btn-lg btn-warning p-2">Back To List</a>
    </form>

</div>


@*//for front end validations*@

@section Scripts{
    @{
    <partial name="_ValidationScriptsPartial" />
    }
}
```

Views/Employee/Delete.cshtml

```
@model Employee

<div class="container shadow p-5">
    <div class="row pb-2">
        <h2>Delete Employee</h2>
    </div>

    <form method="post" asp-action="DeleteEmp">
        <input asp-for="Id" hidden />
        <div asp-validation-summary="All"></div>

        <div class="form-row">
            <div class="form-group col-md-6">
                <label asp-for="Name">Employee Name</label>
                <input type="text" class="form-control mb-3" asp-
for="Name" disabled>
                <span asp-validation-for="Name" class=" alert-
danger"></span>
            </div>
            <div class="form-group col-md-6">
                <label asp-for="Designation">Designation</label>
                <input type="text" class="form-control mb-3" asp-
for="Designation" disabled>
                <span asp-validation-for="Designation" class=" alert-
danger"></span>
            </div>
        </div>

        <div class="form-row">
            <div class="form-group col-md-6">
                <label asp-for="Address">Address</label>
                <input type="text" class="form-control mb-3" asp-
for="Address" disabled>
                <span asp-validation-for="Address" class=" alert-
danger"></span>
            </div>
            <div class="form-group col-md-6 mb-3">
                <label asp-for="RecordCreatedOn">Created On</label>
                <input type="datetime-local" class="form-control" asp-
for="RecordCreatedOn" disabled>
                <span asp-validation-for="RecordCreatedOn" class="
alert-danger"></span>
            </div>
        </div>


        <button type="submit" class="btn btn-lg btn-danger p-2"><i
class="bi bi-trash-fill"></i>Delete</button>
        <a asp-controller="Employee" asp-action="Index" class="btn
btn-lg btn-warning p-2">Back To List</a>
    </form>

</div>
```

```
@*//for front end validations*@

@section Scripts{
    @{
    <partial name="_ValidationScriptsPartial" />
    }
}
```

Views/Employee/Edit.cshtml

```
@model Employee

<div class="container shadow p-5">
    <div class="row pb-2">
        <h2>Edit Employee</h2>
    </div>

    <form method="post" asp-action="Edit">
        <div asp-validation-summary="All"></div>

        <div class="form-row">
            <div class="form-group col-md-6">
                <label asp-for="Name">Employee Name</label>
                <input type="text" class="form-control mb-3" asp-
for="Name">
                <span asp-validation-for="Name" class=" alert-
danger"></span>
            </div>
            <div class="form-group col-md-6">
                <label asp-for="Designation">Designation</label>
                <input type="text" class="form-control mb-3" asp-
for="Designation">
                <span asp-validation-for="Designation" class=" alert-
danger"></span>
            </div>
        </div>

        <div class="form-row">
            <div class="form-group col-md-6">
                <label asp-for="Address">Address</label>
                <input type="text" class="form-control mb-3" asp-
for="Address">
                <span asp-validation-for="Address" class=" alert-
danger"></span>
            </div>
            <div class="form-group col-md-6 mb-3">
                <label asp-for="RecordCreatedOn">Created On</label>
                <input type="datetime-local" class="form-control" asp-
for="RecordCreatedOn">
```

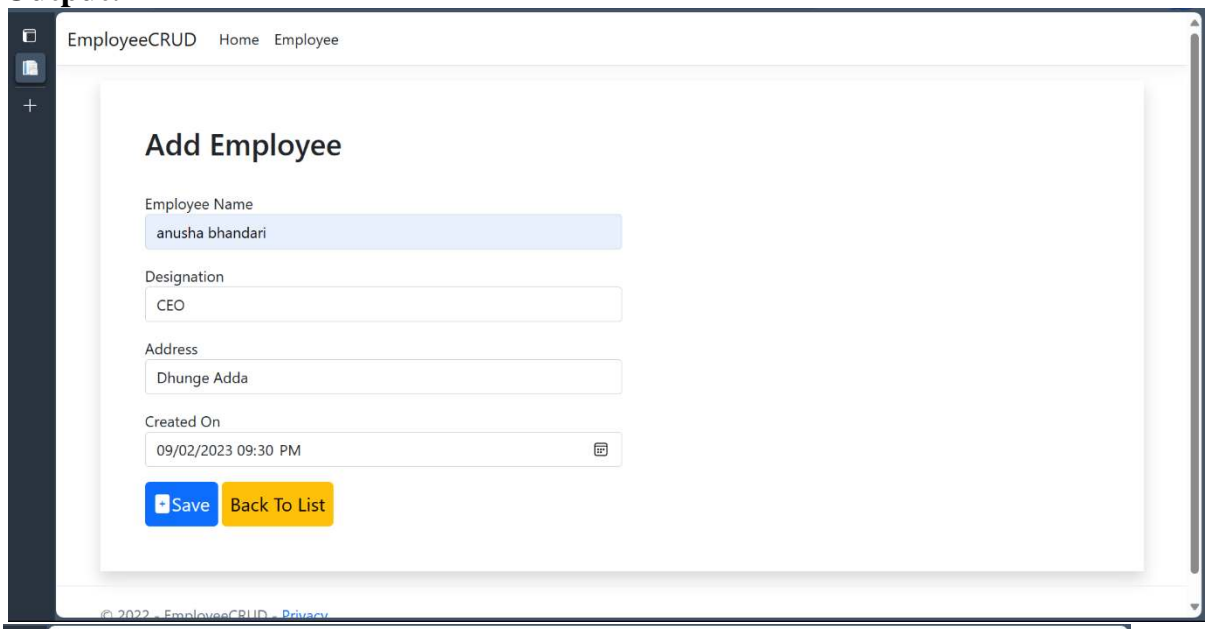
```
                <span asp-validation-for="RecordCreatedOn" class="
alert-danger"></span>
            </div>
        </div>


        <button type="submit" class="btn btn-lg btn-primary p-2"><i
class="bi bi-file-plus-fill"></i>Update</button>
        <a asp-controller="Employee" asp-action="Index" class="btn
btn-lg btn-warning p-2">Back To List</a>
    </form>

</div>


@*//for front end validations*@

@section Scripts{
    @{
    <partial name="_ValidationScriptsPartial" />
    }
}
```

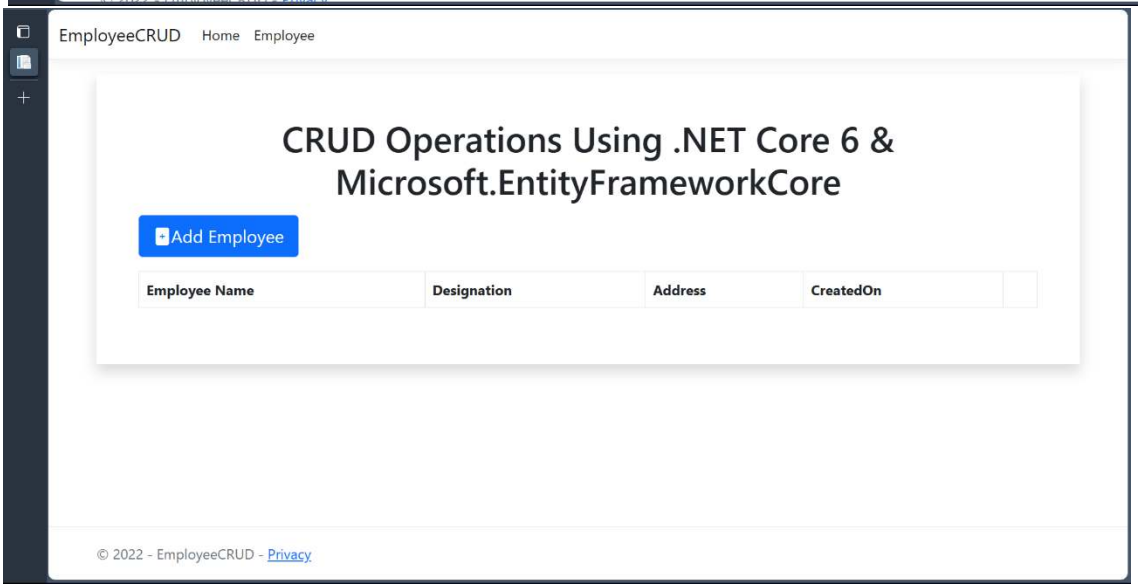
Views/Employee/Index.cshtml

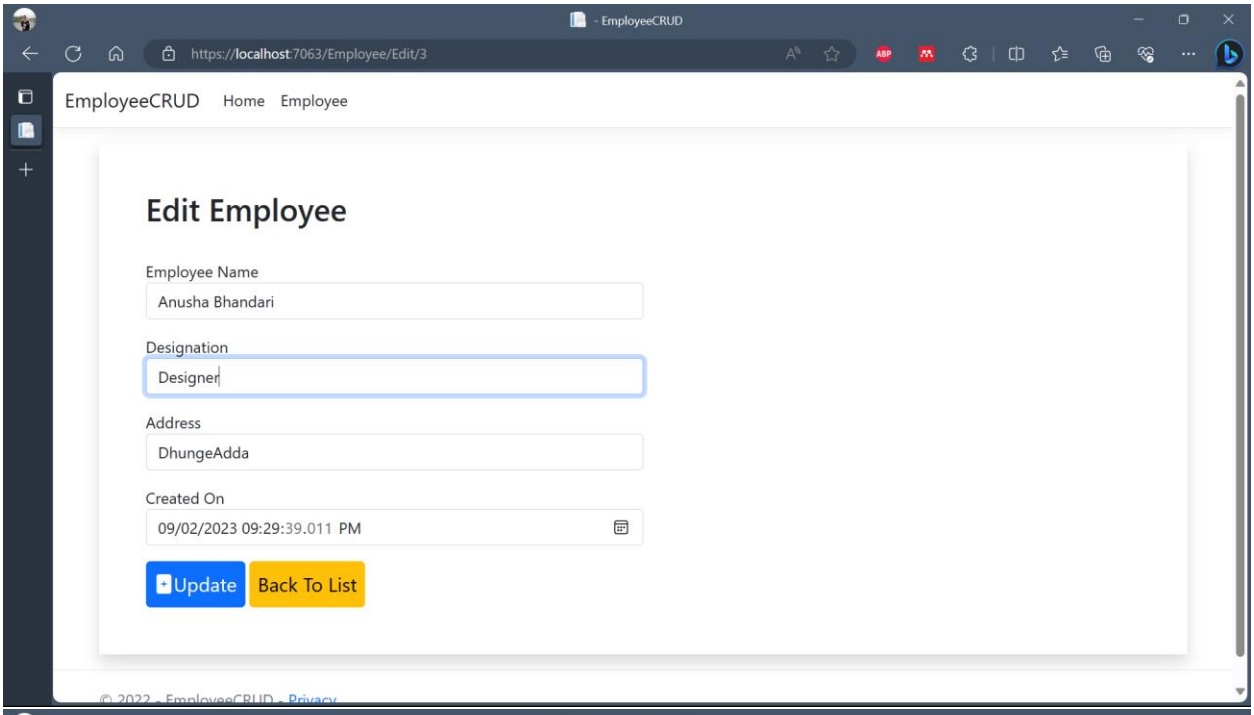
```
@model IEnumerable<Employee>

    @{
    ViewData["Title"] = "Index";
}

@if (TempData["ResultOk"] != null)
{
    <h1 class="alert-success">@TempData["ResultOk"]</h1>
}

<div class="container shadow p-5">

    <h1 class="text-center mb-3">CRUD Operations Using .NET Core 6 &
Microsoft.EntityFrameworkCore </h1>

    <div class="col mb-3">
        <a asp-controller="Employee" asp-action="Create" class="btn
btn-lg btn-primary"><i class="bi bi-file-plus-fill"></i>Add
Employee</a>
    </div>
    <table class="table table-bordered table-hover">
        <thead>
            <tr>
                <th scope="col">Employee Name</th>
```

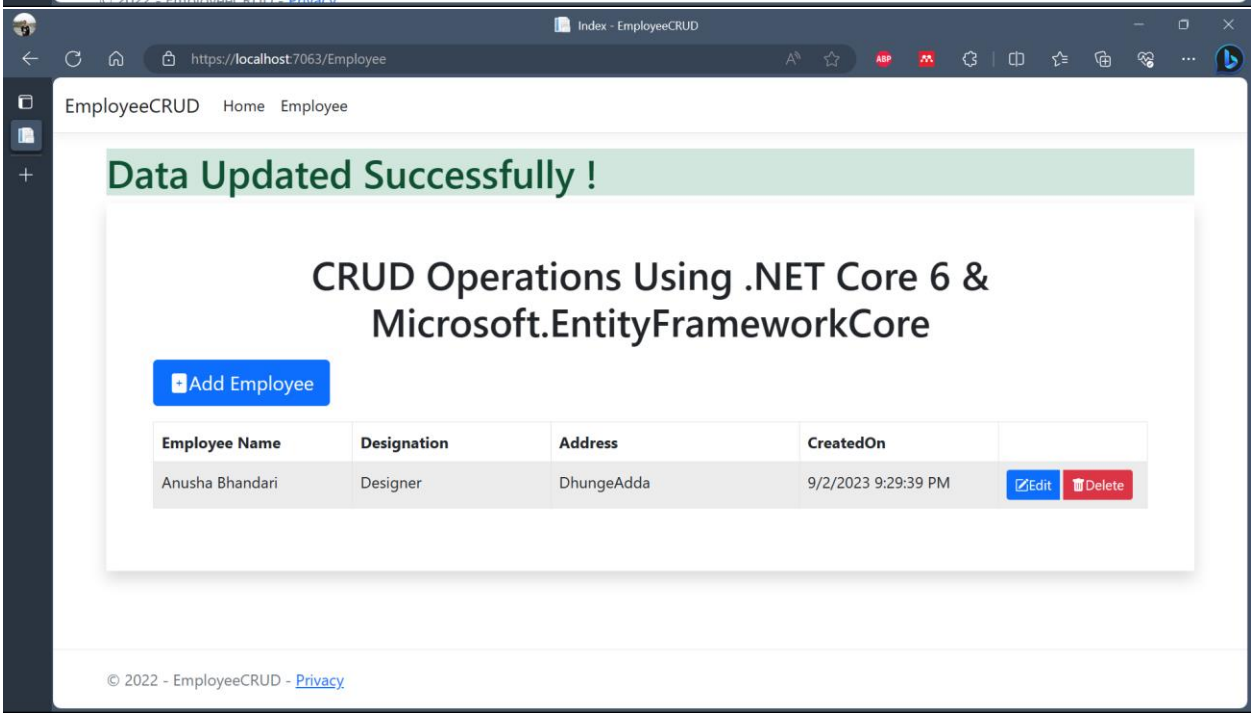
```
                <th scope="col">Designation</th>
                <th scope="col">Address</th>
                <th scope="col">CreatedOn</th>
                <th></th>
            </tr>
        </thead>
        <tbody>

            @foreach (var item in Model)
            {
                <tr>
                    <td width="20%">
                        @item.Name
                    </td>
                    <td width="20%">
                        @item.Designation
                    </td>
                    <td width="25%">
                        @item.Address
                    </td>
                    <td width="20%">
                        @item.RecordCreatedOn
                    </td>
                    <td>
                        <div role="group" class="w-60 btn-group">
                            <a asp-controller="Employee" asp-
action="Edit" asp-route-id="@item.Id" class=" btn btn-sm btn-
primary"><i class="bi bi-pencil-square"></i>Edit</a> 
                            <a asp-controller="Employee" asp-
action="Delete" asp-route-id="@item.Id" class="btn btn-sm btn-
danger"><i class="bi bi-trash-fill"></i>Delete</a>
                        </div>
                    </td>
                </tr>
            }
        </tbody>
    </table>
</div>
```

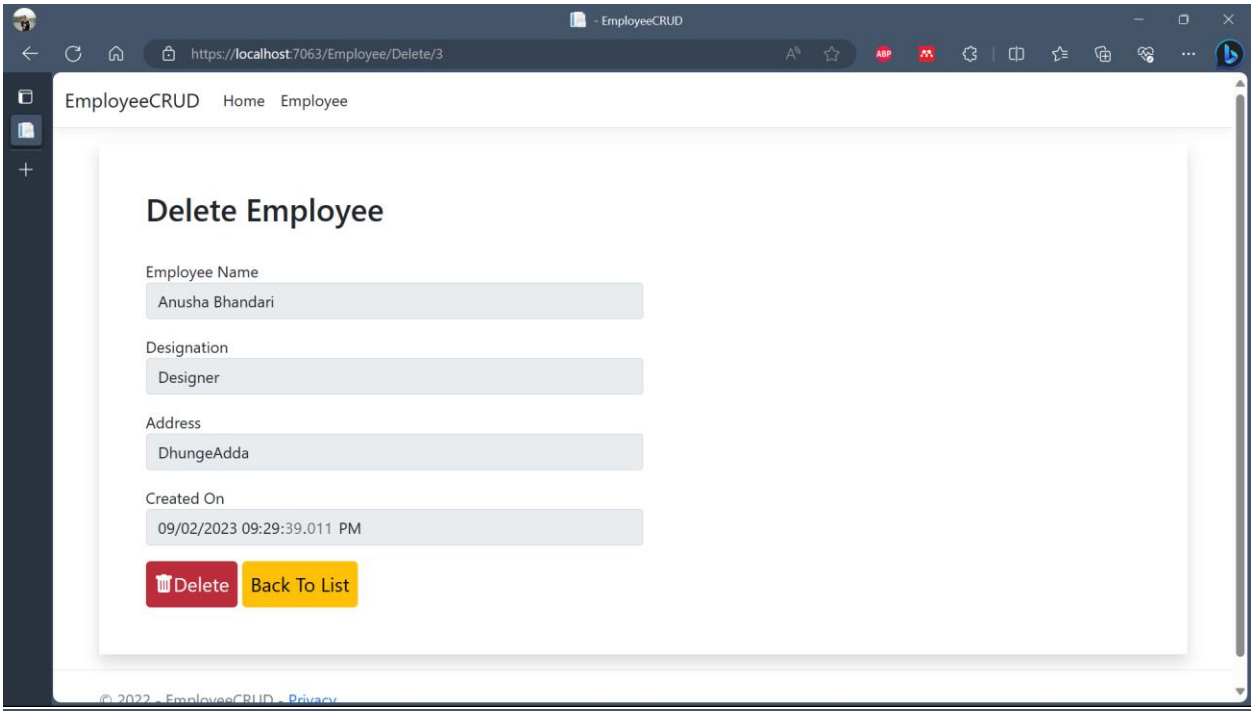
Output:

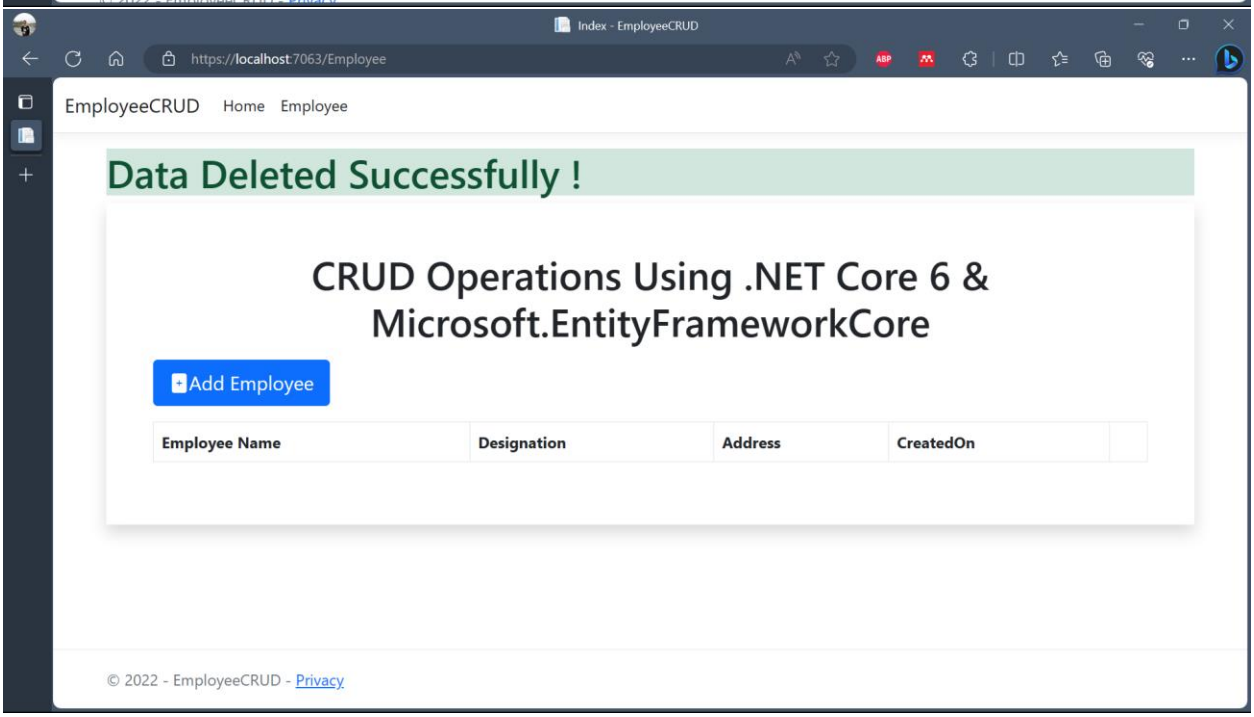












15. Write a program to demonstrate state management server- side in asp. net core application.

Controllers/StateController.cs

```
using Microsoft.AspNetCore.Mvc;

namespace lab15.Controllers
{
    public class StateController : Controller
    {
        /*public IActionResult Index()
        {
            // Session state example
            HttpContext.Session.SetString("Username", "John");

            // TempData example
            TempData["Message"] = "Hello from TempData!";

            return View();
        }

        public IActionResult Display()
        {
            // Retrieve session state
            string username =
HttpContext.Session.GetString("Username");

            // Retrieve TempData
            string message = TempData["Message"] as string;

            ViewBag.Username = username;
            ViewBag.Message = message;

            return View();
        }*/
        public IActionResult Add()
        {
            return View();
        }

        [HttpPost]
        public IActionResult SetUserData(string username, string
message)
        {
            // Session state example
            HttpContext.Session.SetString("Username", username);

            // TempData example
            TempData["Message"] = message;

            return RedirectToAction("Display");
```

```
        }

        public IActionResult Display()
        {
            // Retrieve session state
            string username =
HttpContext.Session.GetString("Username");

            // Retrieve TempData
            string message = TempData["Message"] as string;

            ViewBag.Username = username;
            ViewBag.Message = message;

            return View();
        }
    }
}
```

Views/State/Add.cshtml

```
@model lab15.Controllers.StateController

<form method="post" asp-action="SetUserData">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required><br>

    <label for="message">Message:</label>
    <input type="text" id="message" name="message" required><br>

    <button type="submit">Submit</button>
</form>
```

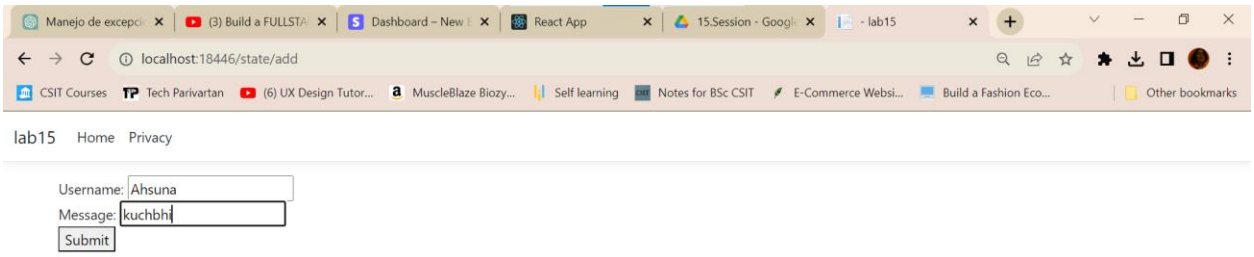
Views/State/Display.cshtml

```
@{
    ViewData["Title"] = "Display";
}

<h2>Display</h2>

<div>
    <p>Username from Session State: @ViewBag.Username</p>
    <p>Message from TempData: @ViewBag.Message</p>
</div>
```

Views/State/Add.cshtml

```
@{
    ViewData["Title"] = "Index";
}

<h2>Index</h2>

<a asp-action="Display">Go to Display Page</a>
```

Output:



16. Write a program to demonstrate state management client side in asp net core application.

Controller/StateController.cs

```
using Microsoft.AspNetCore.Mvc;

namespace lab16.Controllers
{
    public class StateController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }

        [HttpPost]
        public IActionResult SetCookie(string data)
        {
            // Set a cookie with the user-provided data
            CookieOptions option = new CookieOptions();
            option.Expires = DateTime.Now.AddMinutes(30); // Cookie
expiration time

            Response.Cookies.Append("UserData", data, option);

            return RedirectToAction("Index");
        }

        public IActionResult GetCookie()
        {
            // Retrieve the user data from the cookie
            string userData = Request.Cookies["UserData"];

            ViewBag.UserData = userData;

            return View();
        }
    }
}
```

Views/State/Getcookie.cshtml

```
@page
@model lab16.Controllers.StateController

<h2>Stored User Data:</h2>
<p>@ViewBag.UserData</p>
```

Views/State/index.cshtml

```
@page
@model lab16.Controllers.StateController

<form method="post" asp-action="SetCookie">
    <label for="data">Enter data:</label>
    <input type="text" name="data" required />
    <button type="submit">Submit</button>
</form>
```

Output: