

Content

Write a program to demonstrate the basic C# structure.	3
Write a program to demonstrate the basics of class and object.	4
Write a program to illustrate encapsulation with properties and indexers and constructor.	5
Write a program that reflects the overloading and overriding of constructor and function.	6
WAP to show the use case of the Method Hiding and the Method Overloading along with the new Keyword.	7
Write a Program to show the struct and the Enum	8
WAP to show the implementation of the Abstract class and the sealed class	9
Write a program to implement multiple inheritance with the use of interfaces.	10
Write the Program to show the Error Handling in dotnet using the try catch and finally Block.	11
Write a program to demonstrate use of Delegate and Events.	12
Write a program to show the use of generic classes and methods.	13
Demonstrate Asynchronous programming with async, await. Task in C#.	15
Write a program to demonstrate the use of the method as a condition in the LINQ(Language Integrated Query).	16
Write a program to show the file handing in the C#	16
Demonstrate how the Asp.Net MVC core application is build and Run.	18
WAP to the Project structure of the ASP.NET MVC Core application.	19
Create an ASP.NET Core application to perform CRUD operation using ADO.NET.	20
Create an ASP.NET Core application to perform CRUD operation using Entity Framework Core.	31
Write a program to demonstrate state management server- side in asp. net core application.	42
Write a program to demonstrate state management Client- side in asp. net core application	43
WAP to show the prevention of the Cross-Site Request Forgery (CSRF) in ASP.NET Core MVC using AntiForgeryToken.	45
Demonstrate the Client-side validation in ASP.NET Core MVC	48

Write a program to showcase the Dependency Injection50

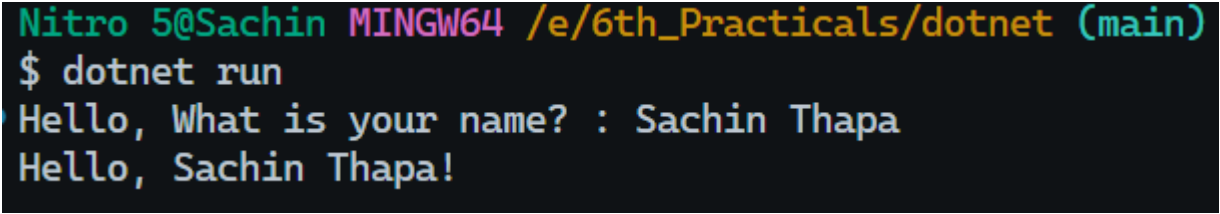
Write a program to demonstrate the basic C# structure.

```
using System;

namespace Practical.Codes
{
    public class Labs
    {
        public static void PrintName()
        {
            System.Console.Write("Hello, What is your name? : ");

            string name = Console.ReadLine() ?? string.Empty;

            System.Console.WriteLine($"Hello, {name}!");
        }
    }
}
```

A terminal window with a dark background and colorful text. The prompt is 'Nitro 5@Sachin MINGW64 /e/6th_Practicals/dotnet (main)'. The user enters '\$ dotnet run'. The program outputs 'Hello, What is your name? : Sachin Thapa' and then 'Hello, Sachin Thapa!'.

```
Nitro 5@Sachin MINGW64 /e/6th_Practicals/dotnet (main)
$ dotnet run
Hello, What is your name? : Sachin Thapa
Hello, Sachin Thapa!
```

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

```
public class ClassDemo
{
    public string Name { get; set; }
    public int Age { get; set; }
    public ClassDemo(string name, int age){
        Name = name;
        Age = age;
    }
    public void Display(){
        Console.WriteLine($"Name: {Name} and Age: {Age}");
    }
}

public class ObjectDemo {
    public static void ClassObject()
    {
        ClassDemo obj = new ClassDemo("Sachin Thapa", 25);
        obj.Display();
    }
}
```

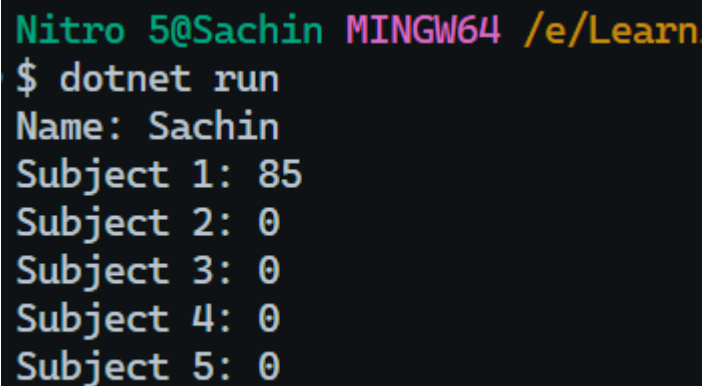
```
Nitro 5@Sachin MINGW64 /e/6th_Practicals/dotnet (main)
$ dotnet run
Name: Sachin Thapa and Age: 25
```

Write a program to illustrate encapsulation with properties and indexes and constructor.

```
using System;

class Person {
    private string name;
    private int[] marks = new int[5];
    public Person(string name) { this.name = name; }
    public string Name { get; set; }
    public int this[int index] { get => marks[index]; set => marks[index] = value; }
    public void ShowInfo()
    {
        Console.WriteLine($"Name: {Name}");
        for (int i = 0; i < marks.Length; i++) Console.WriteLine($"Subject {i + 1}: {marks[i]}");
    }
}

class Program{
    static void Main(){
        Person p = new Person("Sachin");
        p[0] = 85;
        p.ShowInfo();
    }
}
```



```
Nitro 5@Sachin MINGW64 /e/Learn
$ dotnet run
Name: Sachin
Subject 1: 85
Subject 2: 0
Subject 3: 0
Subject 4: 0
Subject 5: 0
```

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

```
class Parent {  
    public static void Show()  
    {  
        Console.WriteLine("Parent static Show() (Method Hiding)");  
    }  
    public void Greet(string name){  
        Console.WriteLine($"Hello, {name}!");  
    }  
}  
  
class Child : Parent {  
    public new static void Show()  
    {  
        Console.WriteLine("Child static Show() (Method Hiding)");  
    }  
    public void Greet(string name, int age)  
    {  
        Console.WriteLine($"Hello, {name}! You are {age} years old.");  
    }  
}
```

```
Nitro 5@Sachin MINGW64 /e/LearningThings/RandomThings/app  
• $ dotnet run  
Hello, Sachin!  
Hello, Sachin! You are 22 years old.  
Parent static Show() (Method Hiding)  
Child static Show() (Method Hiding)
```

WAP to show the use case of the Method Hiding and the Method Overloading along with the new Keyword

```
class Parent
{
    public static void Show()
    {
        Console.WriteLine("Parent static Show()");
    }
}

class Child : Parent
{
    public new static void Show()
    {
        Console.WriteLine("Child static Show()");
    }

    public void Greet()
    {
        Console.WriteLine("Hello!");
    }

    public void Greet(string name)
    {
        Console.WriteLine($"Hello, {name}!");
    }
}
```

```
Nitro 5@Sachin MINGW64 /e/LearningThings/RandomThings/app
$ dotnet run
Hello!
Hello, Sachin!
Parent static Show()
Child static Show()
```

Write a Program to show the struct and the Enum

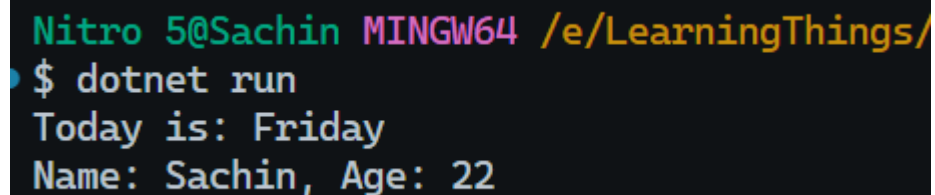
```
enum Day { Sunday, Monday, Tuesday , Thursday , Friday , Saturday }
```

```
struct Person
```

```
{  
    public string Name;  
    public int Age;  
    public void ShowInfo()  
    {  
        Console.WriteLine($"Name: {Name}, Age: {Age}");  
    } }  
}
```

```
class Program
```

```
{  
    static void Main()  
    {  
        Day today = Day.Friday;  
        Console.WriteLine("Today is: " + today);  
        Person p;  
        p.Name = "Sachin";  
        p.Age = 22;  
        p.ShowInfo();  
    }  
}
```



```
Nitro 5@Sachin MINGW64 /e/LearningThings/  
$ dotnet run  
Today is: Friday  
Name: Sachin, Age: 22
```

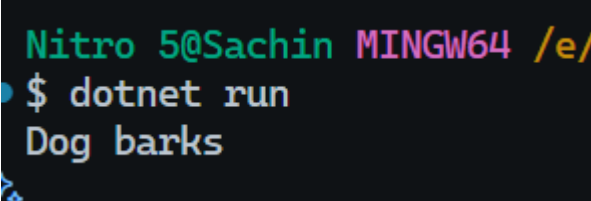

WAP to show the implementation of the Abstract class and the sealed class

```
using System;

abstract class Animal
{
    public abstract void MakeSound();
}

sealed class Dog : Animal
{
    public override void MakeSound()
    {
        Console.WriteLine("Dog barks");
    }
}

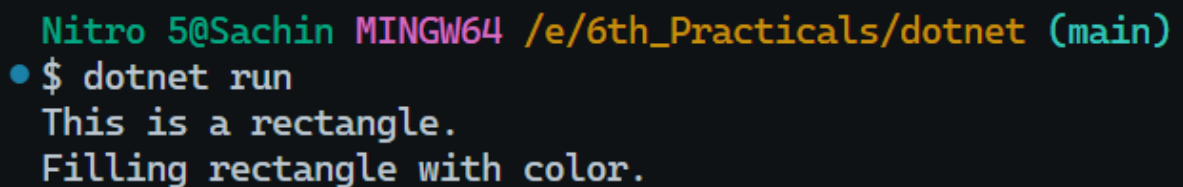
class Program
{
    static void Main()
    {
        Dog dog = new Dog();
        dog.MakeSound();
    }
}
```



A screenshot of a terminal window with a dark background. The prompt is 'Nitro 5@Sachin MINGW64 /e/'. The command '\$ dotnet run' has been entered, and the output 'Dog barks' is displayed on the next line.

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

```
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.");
    } }
public class InterfaceExample{
    public static void RunProgram(){
        Rectangle rectangle = new Rectangle();
        rectangle.Display();
        rectangle.FillColor();
    }
}
```



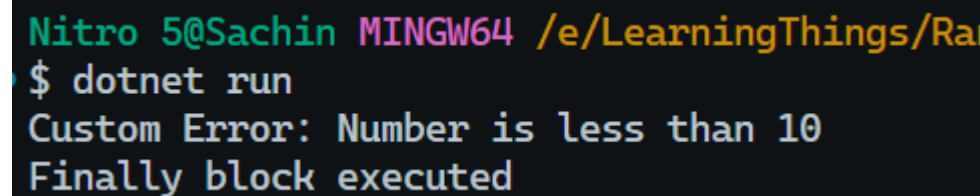
A screenshot of a terminal window with a dark background. The prompt is 'Nitro 5@Sachin MINGW64 /e/6th_Practicals/dotnet (main)'. The command '\$ dotnet run' has been executed, resulting in two lines of output: 'This is a rectangle.' and 'Filling rectangle with color.'.

```
Nitro 5@Sachin MINGW64 /e/6th_Practicals/dotnet (main)
• $ dotnet run
This is a rectangle.
Filling rectangle with color.
```

Write the Program to show the Error Handling in dotnet using the try catch and finally Block.

```
class MyCustomException : Exception
{
    public MyCustomException(string message) : base(message) { }
}

class Program{
    static void Main()
    {
        Try {
            int num = 5;
            if (num < 10) {
                throw new MyCustomException("Number is less than 10");
            }
            int result = num / 0;
            Console.WriteLine(result);
        }
        catch (DivideByZeroException ex) {
            Console.WriteLine("System Error: " + ex.Message);
        }
        catch (MyCustomException ex) {
            Console.WriteLine("Custom Error: " + ex.Message);
        }
        finally {
            Console.WriteLine("Finally block executed");
        }
    }
}
```



```
Nitro 5@Sachin MINGW64 /e/LearningThings/Ra
$ dotnet run
Custom Error: Number is less than 10
Finally block executed
```

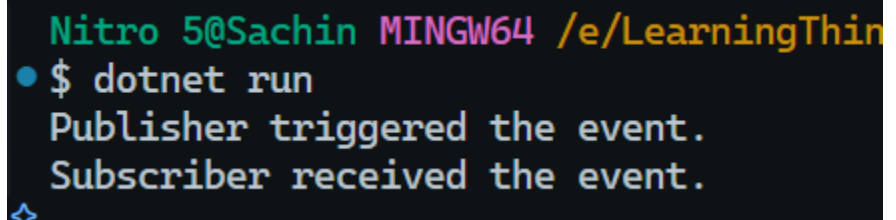
Write a program to demonstrate use of Delegate and Events.

```
delegate void Notify();

class Publisher {
    public event Notify? OnNotify;
    public void TriggerEvent()
    {
        Console.WriteLine("Publisher triggered the event.");
        OnNotify?.Invoke();
    }
}

class Subscriber {
    public void OnEventOccurred() {
        Console.WriteLine("Subscriber received the event.");
    }
}

class Program {
    static void Main() {
        Publisher publisher = new Publisher();
        Subscriber subscriber = new Subscriber();
        publisher.OnNotify += subscriber.OnEventOccurred;
        publisher.TriggerEvent();
    }
}
```



```
Nitro 5@Sachin MINGW64 /e/LearningThin
• $ dotnet run
Publisher triggered the event.
Subscriber received the event.
```

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

```
namespace Collection
```

```
{
```

```
    using System;
```

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

```
    public class StudentModel
```

```
    {
```

```
        public string Name { get; set; } = String.Empty;
```

```
        public int Age { get; set; }
```

```
        public string Grade { get; set; } = String.Empty;
```

```
    }
```

```
    public class StudentList
```

```
    {
```

```
        private List<StudentModel> students = new List<StudentModel>();
```

```
        public void AddStudent(StudentModel student)
```

```
        {
```

```
            students.Add(student);
```

```
        }
```

```
        public void DisplayStudents()
```

```
        {
```

```
            foreach (var student in students)
```

```
            {
```

```
                Console.WriteLine($"Name: {student.Name}, Age: {student.Age}, Grade: {student.Grade}");
```

```
    }  
}  
}
```

```
public class StudentDisplay
```

```
{
```

```
    public static void RunProgram()
```

```
    {
```

```
        // Step 5: Create a StudentList of Student objects
```

```
        StudentList studentList = new StudentList();
```

```
        // Step 6: Add some students to the list
```

```
        studentList.AddStudent(new StudentModel { Name = "John Doe", Age = 20, Grade = "A" });
```

```
        studentList.AddStudent(new StudentModel { Name = "Jane Smith", Age = 22, Grade = "B" });
```

```
        studentList.AddStudent(new StudentModel { Name = "Alice Brown", Age = 19, Grade = "A" });
```

```
        // Step 7: Display all students
```

```
        Console.WriteLine("Student List:");
```

```
        studentList.DisplayStudents();
```

```
    }
```

```
}
```

```
Nitro 5@Sachin MINGW64 /e/6th_Practicals/dotnet (main)  
● $ dotnet run  
Student List:  
Name: Sachin Thapa, Age: 20, Grade: A  
Name: Ram Bahik, Age: 22, Grade: B  
Name: Sita kandel, Age: 19, Grade: A
```

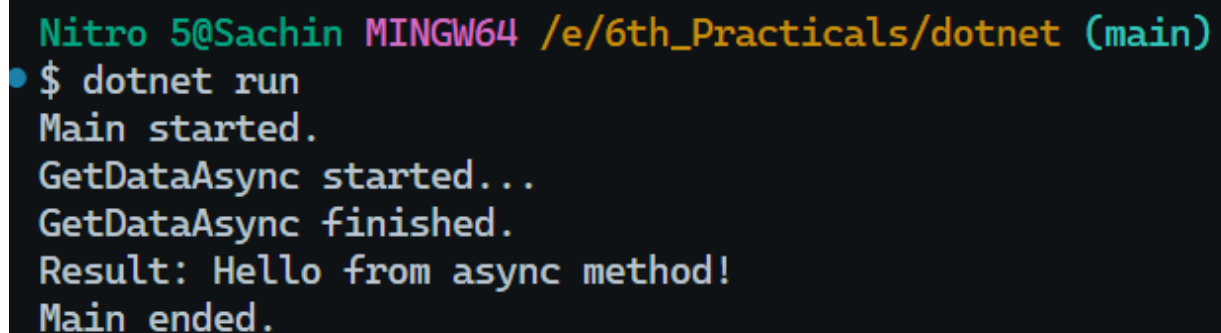
Demonstrate Asynchronous programming with async, await. Task in C#.

```
using System;

using System.Threading.Tasks;

namespace AsyncDemo
{
    class AsyncDataHandler
    {
        static async Task<string> GetDataAsync()
        {
            Console.WriteLine("GetDataAsync started...");
            await Task.Delay(3000);
            Console.WriteLine("GetDataAsync finished.");
            return "Hello from async method!";
        }

        public static async Task Run()
        {
            Console.WriteLine("Main started.");
            string result = await GetDataAsync();
            Console.WriteLine($"Result: {result}");
            Console.WriteLine("Main ended.");
        }
    }
}
```



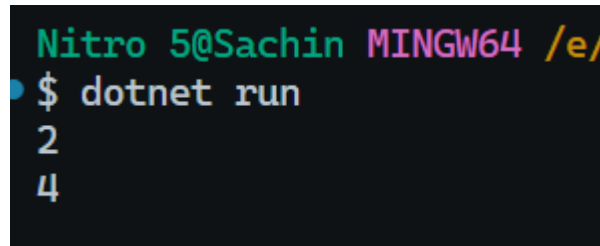
A terminal window with a dark background and light-colored text. The prompt is 'Nitro 5@Sachin MINGW64 /e/6th_Practicals/dotnet (main)'. The command '\$ dotnet run' has been executed. The output shows the program's execution flow: 'Main started.', 'GetDataAsync started...', 'GetDataAsync finished.', 'Result: Hello from async method!', and 'Main ended.'.

```
Nitro 5@Sachin MINGW64 /e/6th_Practicals/dotnet (main)
• $ dotnet run
Main started.
GetDataAsync started...
GetDataAsync finished.
Result: Hello from async method!
Main ended.
```

Write a program to demonstrate the use of the method as a condition in the LINQ(Language Integrated Query).

```
class LINQExample
{
    public void DemonstrateLINQ()
    {

        var numbers = new List<int> { 1, 2, 3, 4, 5 };
        IEnumerable<int> evenNumbers = numbers.Where(n => n % 2 == 0).ToList();
        foreach (var number in evenNumbers)
        {
            Console.WriteLine(number);
        }
    }
}
```



```
Nitro 5@Sachin MINGW64 /e/
$ dotnet run
2
4
```

Write a program to show the file handing in the C#

```
using System;
using System.IO;
namespace FileHandlingStreamDemo
{
    class FileHandling
    {
        public static void RunProgram()
        {
```



```

string filePath = "./note/file.txt";

// Step 1: Write to the file using StreamWriter
using (StreamWriter writer = new StreamWriter(filePath))
{
    writer.WriteLine("Hello, this is line 1.");
    writer.WriteLine("This is line 2.");
    writer.WriteLine("C# file handling using StreamWriter and StreamReader.");
}
Console.WriteLine("Data written to file using StreamWriter.\n");

// Step 2: Read from the file using StreamReader
Console.WriteLine("Reading from file using StreamReader:");

using (StreamReader reader = new StreamReader(filePath))
{
    string? line;
    while ((line = reader.ReadLine()) != null)
    {
        Console.WriteLine(line);
    }
}
}
}
}

```

```

Nitro 5@Sachin MINGW64 /e/6th_Practicals/dotnet (main)
● $ dotnet run
Data written to file using StreamWriter.

Reading from file using StreamReader:
Hello, this is line 1.
This is line 2.
C# file handling using StreamWriter and StreamReader.

```

Demonstrate how the Asp.Net MVC core application is build and Run.

Create a Asp.net core MVC Application

1. Create a new dotnet core web application using the command line or Visual Studio.
2. Choose the "ASP.NET Core Web App (Model-View-Controller)" template.
3. Name the project and solution as "MyMvcApp".
4. Choose the target framework (e.g., .NET 6 or .NET 7).
5. Click "Create" to generate the project.
6. Write the code for the MVC application in the appropriate folders (Models, Views, Controllers).

Using the Command Line Interface (CLI):

```
dotnet new mvc -n MyMvcApp
```

```
cd MyMvcApp
```

```
dotnet run
```

6. Open the project in your preferred IDE (e.g., Visual Studio, Visual Studio Code).

Run the Application

1. Open a terminal or command prompt.
2. Navigate to the project directory (where the `.csproj` file is located).
3. Run the following command to start the application:

```
dotnet run
```

Build and Run the Application

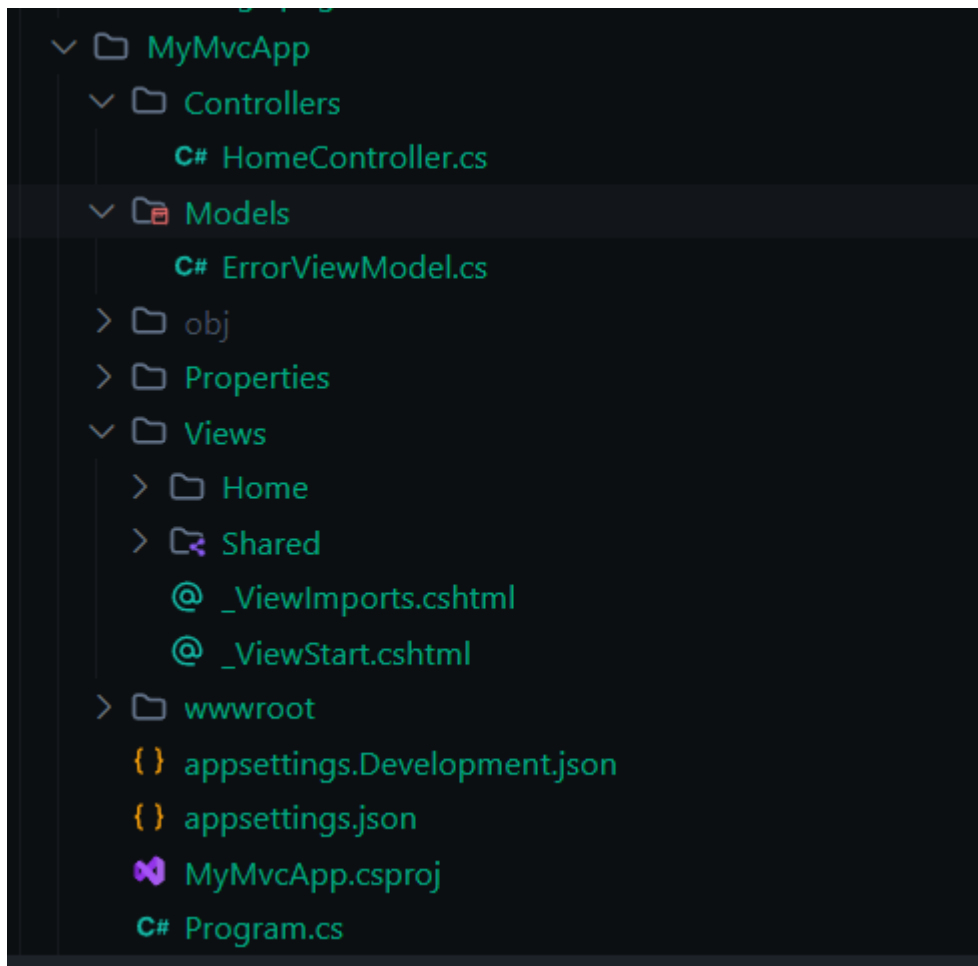
1. Open a terminal or command prompt.
2. Navigate to the project directory (where the `.csproj` file is located).
3. Run the following command to build the application:

dotnet build

4. After a successful build, run the application using:

dotnet run

WAP to the Project structure of the ASP.NET MVC Core application.



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

Models/ProductModel.cs

```
using System.ComponentModel.DataAnnotations;
```

```
namespace MySqlAdoNetCrude.Models
```

```
{
```

```
    public class ProductModel
```

```
    {
```

```
        [Key]
```

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

```
        [Required(ErrorMessage = "Name is required")]
```

```
        [Display(Name = "Product Name")]
```

```
        [StringLength(100)]
```

```
        public string Name { get; set; } = string.Empty;
```

```
        [Required(ErrorMessage = "Price is required")]
```

```
        [Range(0.01, 10000.00, ErrorMessage = "Price must be between 0.01 and 10000.00")]
```

```
        [Display(Name = "Product Price")]
```

```
        [DataType(DataType.Currency)]
```

```
        public decimal Price { get; set; }
```

```
        [Required(ErrorMessage = "Quantity is required")]
```

```

        [Display(Name = "Quantity")]
        [Range(1, 1000, ErrorMessage = "Quantity must be between 1 and 1000")]
        public int Quantity { get; set; }
    }
}

```

Controllers/ProductController.cs

```

using System;
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using MySql.Data.MySqlClient;
using MySqlAdoNetCrude.Models;
namespace MySqlAdoNetCrude.Controllers
{
    public class ProductController : Controller
    {
        private readonly IConfiguration _configuration;
        private readonly string _connectionString;

        public ProductController(IConfiguration configuration)
        {
            _configuration = configuration;
            _connectionString = _configuration.GetConnectionString("DefaultConnection") ??
                throw new InvalidOperationException("Connection string 'DefaultConnection' is not configured.");
        }
    }
}

```

[HttpGet]

public IActionResult Index()

{

List<ProductModel> productsList = new List<ProductModel>();

try

{

using (SqlConnection conn = new SqlConnection(_connectionString))

{

conn.Open();

string query = "SELECT * FROM Products";

using var command = new MySqlCommand(query, conn);

using MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

productsList.Add(new ProductModel

{

Id = reader.GetInt32("Id"),

Name = reader.GetString("Name"),

Price = reader.GetDecimal("Price"),

Quantity = reader.GetInt32("Quantity")

});

}}} catch (Exception ex)

{

ViewData["Result"] = \$"Error: {ex.Message}";

}

return View(productsList);

```

    }

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

    [HttpPost]
    [ValidateAntiForgeryToken]
    public IActionResult Create(ProductModel product)
    {
        if (ModelState.IsValid) { try {

            using (SqlConnection conn = new SqlConnection(_connectionString))
            {
                conn.Open();

                string query = "INSERT INTO Products (Name, Price, Quantity) VALUES (@Name, @Price,
@Quantity)";

                using var command = new MySqlCommand(query, conn);
                command.Parameters.AddWithValue("@Name", product.Name);
                command.Parameters.AddWithValue("@Price", product.Price);
                command.Parameters.AddWithValue("@Quantity", product.Quantity);

                command.ExecuteNonQuery();
            }

            return RedirectToAction(nameof(Index));
        }

        catch (Exception ex)
        {
            ModelState.AddModelError("", $"Error: {ex.Message}");

```

```

    }
}
return View(product);
}
[HttpGet]
public IActionResult Edit(int id)
{
    ProductModel product = new ProductModel();
    try
    {
        using (MySqlConnection conn = new MySqlConnection(_connectionString))
        {
            conn.Open();
            string query = "SELECT * FROM Products WHERE Id=@Id";
            using var command = new MySqlCommand(query, conn);
            command.Parameters.AddWithValue("@Id", id);
            using MySqlDataReader reader = command.ExecuteReader();

            if (reader.Read())
            {
                product.Id = reader.GetInt32("Id");
                product.Name = reader.GetString("Name");
                product.Price = reader.GetDecimal("Price");
                product.Quantity = reader.GetInt32("Quantity");
            }
            else
            {

```



```

        return NotFound();
    } }

    catch (Exception ex)
    {
        ViewData["Result"] = $"Error: {ex.Message}";
    }

    return View(product);
}

[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Edit(ProductModel product) {
    if (ModelState.IsValid) {
        Try {
            using (SqlConnection conn = new SqlConnection(_connectionString))
            {
                conn.Open();

                string query = "UPDATE Products SET Name=@Name, Price=@Price, Quantity=@Quantity
WHERE Id=@Id";

                using var command = new MySqlCommand(query, conn);
                command.Parameters.AddWithValue("@Id", product.Id);
                command.Parameters.AddWithValue("@Name", product.Name);
                command.Parameters.AddWithValue("@Price", product.Price);
                command.Parameters.AddWithValue("@Quantity", product.Quantity);

                int rowsAffected = command.ExecuteNonQuery();

                if (rowsAffected == 0)
                {
                    return NotFound();
                }
            }
        }
    }
}

```

```

        }
    }
    return RedirectToAction(nameof(Index));
}
catch (Exception ex)
{
    ModelState.AddModelError("", $"Error: {ex.Message}");
}
}
return View(product);
}
[ValidateAntiForgeryToken]
public IActionResult Delete(int id)
{
    try
    {
        using (SqlConnection conn = new SqlConnection(_connectionString))
        {
            conn.Open();
            string query = "DELETE FROM Products WHERE Id=@Id";
            using var command = new SqlCommand(query, conn);
            command.Parameters.AddWithValue("@Id", id);
            int rowsAffected = command.ExecuteNonQuery();

            if (rowsAffected == 0)
            {
                ViewData["Result"] = "Product not found or already deleted.";
            }
        }
    }
}

```

```

        }
    }
}
catch (Exception ex)
{
    ViewData["Result"] = $"Error: {ex.Message} ";
}
return RedirectToAction(nameof(Index));
} } }

```

Views/Edit.cshtml

```

@model MySqlAdoNetCrude.Models.ProductModel

<div class="container mt-4">

    <div class="row">

        <div class="col-md-8 offset-md-2">

            <div class="card">

                <div class="card-header bg-info text-white">

                    <h4 class="mb-0">Edit Product</h4>

                </div>

                <div class="card-body">

                    <form asp-action="Edit" method="post">

                        <div asp-validation-summary="ModelOnly" class="text-danger"></div>

                        <input type="hidden" asp-for="Id" />

                        <div class="mb-3">

                            <label asp-for="Name" class="form-label"></label>

                            <input asp-for="Name" class="form-control" />

                        </div>

                    </form>

                </div>

            </div>

        </div>

    </div>

```

```

        <span asp-validation-for="Name" class="text-danger"></span>
    </div>

    <div class="mb-3">
        <label asp-for="Price" class="form-label"></label>
        <input asp-for="Price" class="form-control" type="number" step="0.01" />
        <span asp-validation-for="Price" class="text-danger"></span>
    </div>

    <div class="mb-3">
        <label asp-for="Quantity" class="form-label"></label>
        <input asp-for="Quantity" class="form-control" type="number" />
        <span asp-validation-for="Quantity" class="text-danger"></span>
    </div>

    <div class="d-flex justify-content-between">
        <a asp-action="Index" class="btn btn-secondary">Back to List</a>
        <button type="submit" class="btn btn-primary">Save Changes</button>
    </div>
</form>

```

Product/Create.cshtml

```

@model MySqlAdoNetCrude.Models.ProductModel

<div class="container mt-4">
    <div class="row">
        <div class="col-md-8 offset-md-2">
            <div class="card">
                <div class="card-header bg-primary text-white">
                    <h4 class="mb-0">Add New Product</h4>

```

</div>

<div class="card-body">

<form asp-action="Create" method="post">

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<div class="mb-3">

<label asp-for="Name" class="form-label"></label>

<input asp-for="Name" class="form-control" />

</div>

<div class="mb-3">

<label asp-for="Price" class="form-label"></label>

<input asp-for="Price" class="form-control" type="number" step="0.01" />

</div>

<div class="mb-3">

<label asp-for="Quantity" class="form-label"></label>

<input asp-for="Quantity" class="form-control" type="number" />

</div>

<div class="d-flex justify-content-between">

<a asp-action="Index" class="btn btn-secondary">Back to List

<button type="submit" class="btn btn-success">Create</button>

</div>

</form>

Products

[Add New Product](#)

Id	Product Name	Product Price	Quantity	Actions
6	Apple	\$25.00	5	Edit Delete
7	Banana	\$45.00	60	Edit Delete
8	cheery	\$80.00	56	Edit Delete

Database Home Privacy ADO_Products EFF_Employees D...>Products>Delete Product - Confirm Delete - EM - Client Side - EM

Products

[Add New Product](#)

Id	Product Name	Product Price	Quantity	Actions
6	Apple	\$25.00	5	Edit Delete
7	Banana	\$45.00	60	Edit Delete
8	cheery	\$80.00	56	Edit Delete

Confirm Delete

Are you sure you want to delete the product: **Apple**?

[Cancel](#) [Delete](#)

Add New Product

Product Name

Product Price

Quantity

[Back to List](#)[Create](#)

Create an ASP.NET Core application to perform CRUD operation using Entity Framework Core.

ApplicationDbContext.cs

```
using EmployeeManagement.Models;
using Microsoft.EntityFrameworkCore;

namespace EmployeeManagement.Data
{
    public class ApplicationDbContext : DbContext
    {
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options)
        {
        }

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

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);
        }
    }
}
```

EmployeeModel.cs

```
using System;

using System.ComponentModel.DataAnnotations;

namespace EmployeeManagement.Models
{
    public class Employee
    {
        [Key]
        public int EmployeeId { get; set; }

        [Required(ErrorMessage = "First Name is required")]
        [Display(Name = "First Name")]
        [StringLength(50)]
        public string FirstName { get; set; } = String.Empty;

        [Required(ErrorMessage = "Last Name is required")]
        [Display(Name = "Last Name")]
        [StringLength(50)]
        public string LastName { get; set; } = String.Empty;

        [EmailAddress]
        [Required(ErrorMessage = "Email is required")]
        [StringLength(100)]
        public string Email { get; set; } = String.Empty;
    }
}
```


[Display(Name = "Phone Number")]

[Phone]

[StringLength(15)]

public string PhoneNumber { get; set; } = String.Empty;

[Required(ErrorMessage = "Hire Date is required")]

[Display(Name = "Hire Date")]

[DataType(DataType.Date)]

public DateTime HireDate { get; set; }

[Required(ErrorMessage = "Department is required")]

[StringLength(50)]

public string Department { get; set; } = String.Empty;

[Display(Name = "Salary")]

[DataType(DataType.Currency)]

[Range(0, 1000000, ErrorMessage = "Salary must be between 0 and 1,000,000")]

public decimal Salary { get; set; }

[Display(Name = "Is Active")]

public bool IsActive { get; set; } = true;

}

}

EmployeeController.cs

using System;

```
using System.Linq;
using System.Threading.Tasks;
using EmployeeManagement.Data;
using EmployeeManagement.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace EmployeeManagement.Controllers
{
    public class EmployeeController : Controller
    {
        private readonly ApplicationDbContext _context;

        public EmployeeController(ApplicationDbContext context)
        {
            _context = context;
        }

        public async Task<IActionResult> Index()
        {
            var employees = await _context.Employees.ToListAsync();
            return View(employees);
        }

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

```

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create(Employee employee)
{
    if (ModelState.IsValid)
    {
        _context.Add(employee);
        await _context.SaveChangesAsync();
        return RedirectToAction("Index");
    }
    return View(employee);
}
}

```

Views/Employee

Index.cshtml

```

<div class="container mt-4">
    <div class="row">
        <div class="col-md-12">
            <div class="d-flex justify-content-between align-items-center mb-4">
                <h1>Employee Management</h1>
                <a asp-action="Create" class="btn btn-primary">
                    <i class="bi bi-plus-circle"></i> Add New Employee
                </a>
            </div>
        </div>

        <div class="table-responsive">

```

```

<table class="table table-striped table-hover">
  <thead class="table-dark">
    <tr>
      <th>@Html.DisplayNameFor(model => model.FirstName)</th>
      <th>@Html.DisplayNameFor(model => model.LastName)</th>
      <th>@Html.DisplayNameFor(model => model.Email)</th>
      <th>@Html.DisplayNameFor(model => model.Department)</th>
      <th>@Html.DisplayNameFor(model => model.HireDate)</th>
      <th>@Html.DisplayNameFor(model => model.Salary)</th>
      <th>@Html.DisplayNameFor(model => model.IsActive)</th>
      <th>Actions</th>
    </tr>
  </thead>
  <tbody>
    @foreach (var item in Model)
    {
      <tr>
        <td>@Html.DisplayFor(modelItem => item.FirstName)</td>
        <td>@Html.DisplayFor(modelItem => item.LastName)</td>
        <td>@Html.DisplayFor(modelItem => item.Email)</td>
        <td>@Html.DisplayFor(modelItem => item.Department)</td>
        <td>@Html.DisplayFor(modelItem => item.HireDate)</td>
        <td>@Html.DisplayFor(modelItem => item.Salary)</td>
        <td>
          @if (item.IsActive)
          {
            <span class="badge bg-success">Active</span>
          }
        </td>
      </tr>
    }
  </tbody>
</table>

```

```

    }
    else
    {
        <span class="badge bg-danger">Inactive</span>
    }
</td>
<td>
    <div class="btn-group" role="group">
        <a asp-action="Details" asp-route-id="@item.EmployeeId" class="btn btn-info btn-
sm">
            <i class="bi bi-eye">Info</i>
        </a>
        <a asp-action="Edit" asp-route-id="@item.EmployeeId" class="btn btn-warning btn-
sm">
            <i class="bi bi-pencil">Edit</i>
        </a>
        <a asp-action="Delete" asp-route-id="@item.EmployeeId"
            class="btn btn-danger btn-sm">
            <i class="bi bi-trash">Delete</i>
        </div>

```

```

@if (!Model.Any())
{
    <div class="alert alert-info text-center">
        No employees found. Please add a new employee.
    </div>
}
</div>

```

</div>

</div>

Create.cshtml

@model EmployeeManagement.Models.Employee

@{

ViewData["Title"] = "Create Employee";

}

<div class="container mt-4">

<div class="row">

<div class="col-md-8 offset-md-2">

<div class="card">

<div class="card-header bg-primary text-white">

<h4 class="mb-0">Add New Employee</h4>

</div>

<div class="card-body">

<form asp-action="Create" method="post">

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<div class="row mb-3">

<div class="col-md-6">

<div class="form-group">

<label asp-for="FirstName" class="control-label"></label>

<input asp-for="FirstName" class="form-control" />

</div>

<div class="row mb-3">

<div class="col-md-6">

<div class="form-group">

<label asp-for="Email" class="control-label"></label>

<input asp-for="Email" class="form-control" />

</div>

</div>

<div class="row mb-3">

<div class="col-md-6">

<div class="form-group">

<label asp-for="Department" class="control-label"></label>

<select asp-for="Department" class="form-control">

<option value="">-- Select Department --</option>

<option value="IT">IT</option>

<option value="HR">HR</option>

<option value="Finance">Finance</option>

<option value="Marketing">Marketing</option>

<option value="Operations">Operations</option>

<option value="Sales">Sales</option>

</select>

</div>

</div>

```
<div class="col-md-6">
```

```
<div class="row mb-3">
```

```
<div class="col-md-6">
```

```
<div class="form-group">
```

```
<label asp-for="Salary" class="control-label"></label>
```

```
<input asp-for="Salary" class="form-control" />
```

```
<span asp-validation-for="Salary" class="text-danger"></span>
```

```
</div>
```

```
</div>
```

```
<div class="form-group d-flex justify-content-between mt-4">
```

```
<a asp-action="Index" class="btn btn-secondary">Back to List</a>
```

```
<button type="submit" class="btn btn-success">Create</button>
```

```
</form>
```

```
@section Scripts {
```

```
@{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
```

```
}
```

Database Home Privacy ADO_Products EFF_Employees Dependency_Injection Server Side SM Client Side SM

Employee Management

Add New Employee

First Name	Last Name	Email	Department	Hire Date	Salary	Is Active	Actions
Sachin	Thapa	test@gmail.com	IT	4/25/2025	\$25.00	Active	Info Edit Delete
Sachin	Thapa	test@gmail.com	Operations	4/26/2025	\$2,589.00	Inactive	Info Edit Delete

Add New Employee

First Name Brian	Last Name Ayala
Email jefeh@mailinator.com	Phone Number 9811321108
Department HR	Hire Date 10/09/1992
Salary 125000	<input checked="" type="checkbox"/> Is Active
Back to List	Create

Employee Details

EmployeeId	2
First Name	Sachin
Last Name	Thapa
Email	test@gmail.com
Phone Number	9811921106
Hire Date	4/26/2025
Department	Operations
Salary	\$2,589.00
Is Active	Inactive

[Back to List](#)[Edit](#)

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

```
using Microsoft.AspNetCore.Mvc;

namespace StateManagement.Controllers {

    public class ServerSideSMController : Controller {

        public IActionResult Index() {

            HttpContext.Session.SetString("Name", "Sachin Thapa");

            HttpContext.Session.SetInt32("Age", 22);

            // TempData

            TempData["Faculty"] = "Bsc csit";

            TempData["sem"] = 6;

            return View();

        }

        public IActionResult GetState() {

            ViewData["Name"] = HttpContext.Session.GetString("Name");

            ViewData["Age"] = HttpContext.Session.GetInt32("Age");

            ViewData["Faculty"] = TempData["Faculty"];

            ViewData["sem"] = TempData["sem"];

            return View();

        }
    }
}
```

Server Side State Management

Name: Sachin Thapa

Age: 22

Faculty: Bsc csit

Semester: 6

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

```
using Microsoft.AspNetCore.Mvc;
namespace StateManagement.Controllers
{
    public class ClientSideSMController : Controller
    {
        public IActionResult Index()
        {
            CookieOptions cookieOptions = new CookieOptions()
            {
                HttpOnly = true,
                Secure = true,
                SameSite = SameSiteMode.Strict,
                Expires = DateTimeOffset.Now.AddDays(2)
            };
            // Set cookies Cookies
            HttpContext.Response.Cookies.Append("Name", "Sachin Thapa", cookieOptions);
            HttpContext.Response.Cookies.Append("Age", "22", cookieOptions);
            HttpContext.Response.Cookies.Append("Faculty", "Csit", cookieOptions);
            return View();
        }

        [ValidateAntiForgeryToken]
        public IActionResult GetState(string num)
        {
            
```

```
// Cookies
```

```
ViewData["Name"] = HttpContext.Request.Cookies["Name"];
```

```
ViewData["Age"] = HttpContext.Request.Cookies["Age"];
```

```
ViewData["Faculty"] = HttpContext.Request.Cookies["Faculty"];
```

```
// hidden Field
```

```
var number = int.Parse(num);
```

```
number++;
```

```
ViewData["Number"] = number.ToString();
```

```
return View();
```

```
}
```

```
public IActionResult RemoveState()
```

```
{
```

```
// Remove Cookies
```

```
HttpContext.Response.Cookies.Delete("Name");
```

```
HttpContext.Response.Cookies.Delete("Age");
```

```
HttpContext.Response.Cookies.Delete("Faculty");
```

```
// Remove Hidden Field
```

```
ViewData["Number"] = "0";
```

```
return RedirectToAction(nameof(Index));
```

```
}
```

```
}
```

```
}
```

Client-Side State Data

Name: Sachin Thapa

Age: 22

Faculty: Csit

Hidden Field Incremented Value: 4

Increment Hidden Field

Clear All Client State

WAP to show the prevention of the Cross-Site Request Forgery (CSRF) in ASP.NET Core MVC using AntiForgeryToken.

HomeController.cs

```
using Microsoft.AspNetCore.Mvc;

public class HomeController : Controller
{
    [HttpGet]
    public IActionResult Form()
    {
        return View();
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public IActionResult Form(string name)
    {
        return Content($"Hello, {name}");
    }
}
```

Form.cshtml

```
@using (Html.BeginForm("Form", "Home", FormMethod.Post))
{
    @Html.AntiForgeryToken()

    <input type="text" name="name" />

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

WAP to show the prevention of the Cross-Site Scripting (XSS) in ASP.NET Core MVC.

XssDemoController.cs

```
using Microsoft.AspNetCore.Mvc;
```

```
public class XssDemoController : Controller
{
    [HttpGet]
    public IActionResult EnterText()
    {
        return View();
    }

    [HttpPost]
    public IActionResult EnterText(string userInput)
    {
        return View("SafeDisplay", model: userInput);
    }
}
```

EnterText.cshtml

```
@using (Html.BeginForm("EnterText", "XssDemo", FormMethod.Post))
{
    <label>Enter Message:</label>

    <input type="text" name="userInput" />

    <button type="submit">Send</button>
}
```

```
}
```

SafeDisplay.cshtml

```
@model string
```

```
<h3>Sanitized Output:</h3>
```

```
<p>@Html.Encode(Model)</p>
```

Enter Message:

Sanitized Output:

<script> alert("XSS attack" </script>

Demonstrate the Client-side validation in ASP.NET Core MVC .

```
<body>

  <form id="registerForm">

    <label>Name:</label>

    <input type="text" id="name" />

    <span id="nameError" style="color:red"></span><br />


    <label>Email:</label>

    <input type="text" id="email" />

    <span id="emailError" style="color:red"></span><br />


    <button type="submit">Register</button>

  </form>


  <script>

    $('#registerForm').submit(function (e) {

      e.preventDefault();

      var isValid = true;

      var name = $('#name').val().trim();

      var email = $('#email').val().trim();

      var emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

      $('#nameError').text("");

      $('#emailError').text("");
```



```
if (name === "") {  
    $('#nameError').text('Name is required.');
```



```
    isValid = false;  
}  
  
if (!emailPattern.test(email)) {  
    $('#emailError').text('Valid email is required.');
```



```
    isValid = false;  
}  
  
if (isValid) {  
    alert("Form submitted successfully!");  
}  
});  
</script>  
</body>
```

Name: Name is required.
Email: Valid email is required.

127.0.0.1:5500 says

Form submitted successfully!

OK

Write a program to showcase the Dependency Injection .

// Controllers/HomeController.cs

```
using Microsoft.AspNetCore.Mvc;

[ApiController]
[Route("[controller]")]
public class DiController : ControllerBase
{
    private readonly IGreetingService _greetingService;

    public DiController(IGreetingService greetingService)
    {
        _greetingService = greetingService;
    }

    [HttpGet("{name}")]
    public IActionResult GetGreeting(string name)
    {
        var message = _greetingService.Greet(name);
        return Ok(new { message });
    }
}
```

Program.cs

```
builder.Services.AddScoped<IGreetingService, GreetingService>();
```

```
{
  "message": "Hello, Sachin Thapa! Welcome to ASP.NET Core DI."
}
```