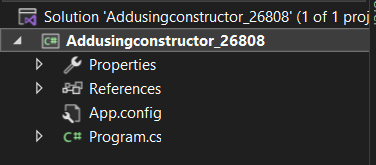
|  |  |
| --- | --- |
| **Lab No.** | **Lab Titles** |
| 01 | Write a C# program to perform addition using constructor. |
| 02 | Write a C# program to initialize and display jagged array elements with sum of each row. |
| 03 | Write a C# program to initialize and display 2D array elements with sum of each row. |
| 04 | Write a C# program to calculate area of rectangle using single interface. |
| 05 | Write a C# program to calculate area and paint cost of rectangle using multiple inheritance. |
| 06 | Write a C# program to call base class constructor using “base” keyword. |
| 07 | Write a C# program using hierarchical inheritance using virtual method. |
| 08 | Write a C# program which takes the length and breadth of 2 rectangle as input and store using an array object. Also print area of each rectangle. |
| 09 | Write a C# program to find the position of a specified word in a given string. |
| 10 | Write a C# program to count the total number of words and characters in a string. |
| 11 | Write a C# program to count the number of alphabets, digits and special characters in string. |
| 12 | Write a C# program to count the number of vowels or consonants in a string. |
| 13 | Write a C# program to calculate sum and difference of two digit using multicast delegates. |
| 14 | Write a C# program to achieve polymorphism using delegates. |
| 15 | Write a C# program to create list with properties for id, name, gender and address. Then creates a list<Student> to store instances of this class. The program adds 10 students to the list, prints the list, and then searches for a student by their address using the findstudentbyaddress function. Finally, it prints the result of the search. |
| 16 | Write a C# program to raise an ApplicationException using Custom MyException class. |
| 17 | Write a C# program to show insert and select student record with given table (tblStudent) with fields (int id, nvarchar(50) name, nvarchar(50) address, nvarchar(50) gender).Also display total no of student from table. |
| 18 | Write a C# program to show insert and fetch student record by Gender from given table (tblStudent) with fields (int id, nvarchar(50) name, nvarchar(50) address, nvarchar(50) gender). |
| 19 | Write a C# program to perform (CRUD) operation from given table (tblStudent) with fields (int id, nvarchar(50) name, nvarchar(50) address, nvarchar(50) gender). |
| 20 | Write a C# Program to insert and display records in table name tblEmployee (id int, name nvarchar(50), address nvarchar(50), salary decimal(18,0), gender nvarchar50) using EntityFramework. |
| 21 | Write a C# program to compute aggregate salary of 5 employee and then display employee record in descending order with respect to employee salary using EntityFramework. |
| 22 | Write a program to select employees whose salary is greater than 20000 and whose address is Kathmandu using EntityFramework. |
| 23 | Write a C# program to connect database Bank and insert 5 costumer records (Account\_no, Name, Address, Balance) in Customer table and display the customer record whose balance is greater than 5000 using EntityFramework. |
| 24 | Create a new project and add a controller with endpoints to read and write values into cookies |
| 25 | Create a controller with endpoints to set and read a value from the session. |
| 26 | Write a simple program to create generic class with generic constructor, generic member variable, generic property and generic method. |
| 27 | Write a asp.net core MVC application to demonstrate in-memory caching. |
| 28 | Write a asp.net core MVC application to demonstrate simple login authentication using Asp.net core identity. |
| 29 | Write a program in C# to implement a generic list data structure. |
| 30 | Write a generic method called ‘Swap’ that takes two parameters of the same type and swaps their values. |

**LAB-01: Write a C# program to perform addition using Constructor.**

**SOURCE CODE:**

using System;

namespace constructorexample\_26830{

internal class Program{

static void Main(string[] args){

Console.WriteLine("enter first digit");

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

Console.WriteLine("enter second digit");

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

AddTwoDigit obj = new AddTwoDigit(a, b);

Console.WriteLine(obj.Add());

Console.ReadLine();

}

}

public class AddTwoDigit{

int first = 0;

int second = 0;

public AddTwoDigit(int x, int y){

first = x;

second = y;

}

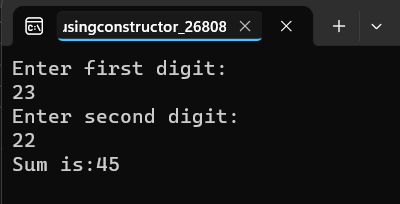
public int Add(){

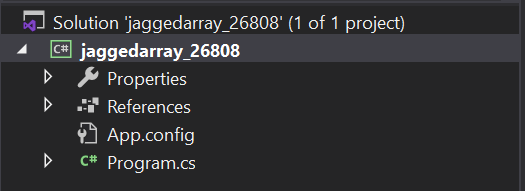
return first + second;

}

}

}

**OUTPUT:**

**LAB-02: Write a C# program to initialize and display jagged array elements with sum of each row.**

**SOURCE CODE:**

using System;

using System.Collections.Generic;

namespace jaggedarray\_26808

{

class Program

{

static void Main(string[] args)

{

int total = 0;

int[][] scores = new int[2][] { new int[] { 92, 93, 94 }, new int[] { 85, 66, 87, 88 } };

for (int i = 0; i < 2; i++){

for (int j = 0; j < scores[i].Length; j++){

Console.Write(scores[i][j] + " ");

total += Convert.ToInt32(scores[i][j]);

}

Console.Write("=" + total);

total = 0;

Console.WriteLine();

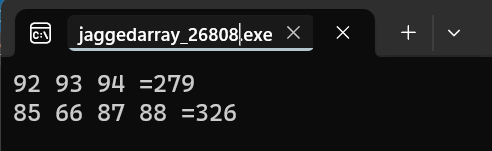
}

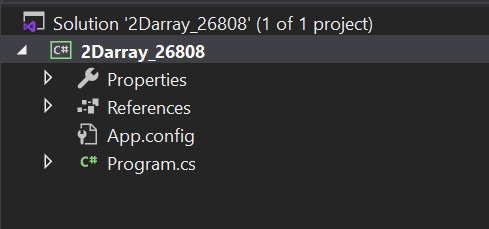
Console.ReadLine();

}

}

}

**OUTPUT:**

**LAB-03: Write a C# program to initialize and display 2D array elements with sum of each row.**

**SOURCE CODE:**

using System;

using System.Collections.Generic;

namespace \_2Darray\_26808

{

class Program

{

static void Main(string[] args)

{

int total = 0;

int[,] scores = new int[3,3] { { 11,10,15 }, { 18,6,30 },{7,23,15 } };

for (int i = 0; i < 3; i++){

for (int j = 0; j < 3; j++){

Console.Write(scores[i,j] + " ");

total += Convert.ToInt32(scores[i,j]);

}

Console.Write("=" + total);

total = 0;

Console.WriteLine();

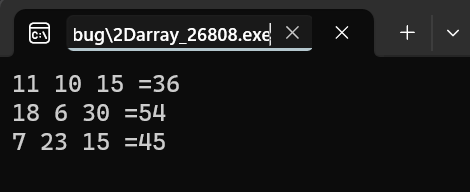
}

Console.ReadLine();

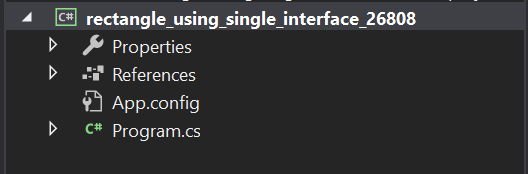
}

}

}

**OUTPUT:**

**LAB-04: Write a C# program to calculate area of rectangle using single interface.**

****

**SOURCE CODE:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace rectangle\_using\_single\_interface

{

class Program

{

static void Main(string[] args)

{

Rectangle rect = new Rectangle();

rect.setHeight(7);

rect.setWidth(10);

Console.WriteLine("Area of Rectangle:{0}",rect.getArea());

Console.ReadLine();

}

}

}

class Shape

{

protected int width;

protected int height;

public void setWidth(int w)

{

width = w;

}

public void setHeight(int h)

{

height = h;

}

}

class Rectangle: Shape

{

public int getArea()

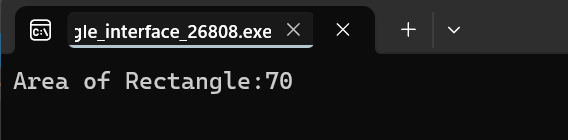
{

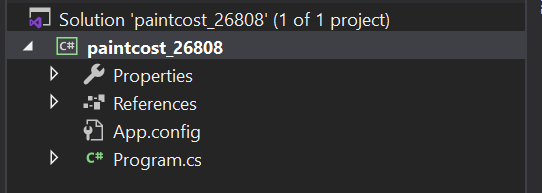
return (width\*height);

}

}

**OUTPUT:**

****

**LAB-05: Write a C# program to calculate area and paint cost of rectangle using multiple inheritance.**

**SOURCE CODE:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace paintcost\_26808

{

class Program

{

static void Main(string[] args)

{

Rectangle rect = new Rectangle();

int area;

rect.setHeight(7);

rect.setWidth(23);

area = rect.getArea();

Console.WriteLine("Total Area:"+ rect.getArea());

Console.WriteLine("Total Paint Cost:"+ rect.getCost(area));

Console.ReadLine();

}

}

}

class Shape

{

protected int width;

protected int height;

public void setWidth(int w)

{

width = w;

}

public void setHeight(int h)

{

height = h;

}

}

//Base Class

public interface PaintCost

{

int getCost(int area);

}

//Derived Class

class Rectangle : Shape, PaintCost

{

public int getArea()

{

return (width \* height);

}

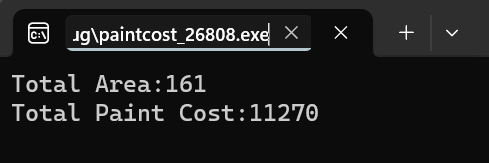
public int getCost(int area)

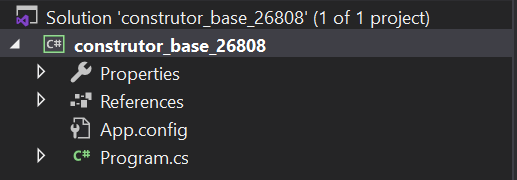
{

return area\*70;

}

}

**OUTPUT:**

**LAB-06: Write a C# program to call base class constructor using “base” keyword.**

**SOURCE CODE:**

using System;

namespace Example\_26808

{

class Program

{

static void Main(string[] args){

Car car = new Car();

Console.WriteLine("Car Model:{0},Speed:{1}", car.model, car.speed);

Console.ReadLine();

}

}

}

class Vehicle

{

public int speed;

public Vehicle(){

this.speed=5;

}

}

class Car : Vehicle

{

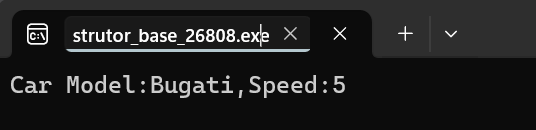
public String model;

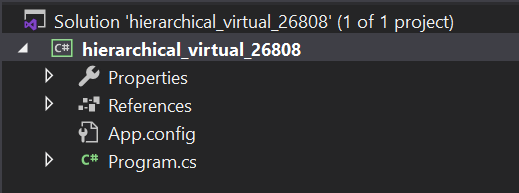
public Car():base(){

this.model = "Bugati";

}

}

**OUTPUT:**

**LAB-07: Write a C# program using hierarchical inheritance using virtual method.**

**SOURCE CODE:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Example\_26808

{

class Program

{

static void Main(string[] args)

{

God g = new Ram();

God g1 = new Sita();

Console.WriteLine(g.GodName());

Console.WriteLine(g1.GodName());

Console.ReadLine();

}

}

}

public class God

{

public virtual string GodName()

{

return "God";

}

}

public class Ram : God

{

public override string GodName()

{

return "Ram";

}

}

public class Sita : God

{

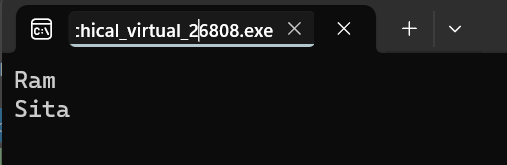
public override string GodName()

{

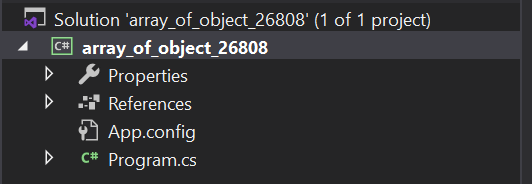
return "Sita";

}

}

**OUTPUT:**

**LAB-08: Write a C# program which takes the length and breadth of 2 Rectangle as input and store using an array objects. Also print area of each rectangle.**

****

**SOURCE CODE:**

using System;

namespace array\_of\_object\_26808

{

class Program

{

static void Main(string[] args)

{

Rectangle[] rect = new Rectangle[2];

for(int i=0;i<2;i++)

{

Console.WriteLine("Rectangle:" + (i + 1));

Console.WriteLine("Enter Length");

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

Console.WriteLine("Enter breadth:");

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

rect[i] = new Rectangle(l, b);

}

for(int i=0;i<2;i++)

{

Console.WriteLine("Area of Rectangle{0}:{1}",(i+1),rect[i].getArea());

}

Console.ReadLine();

}

}

}

public class Rectangle

{

private int length = 0;

private int breadth = 0;

public Rectangle(int l,int b)

{

this.length = l;

this.breadth= b;

}

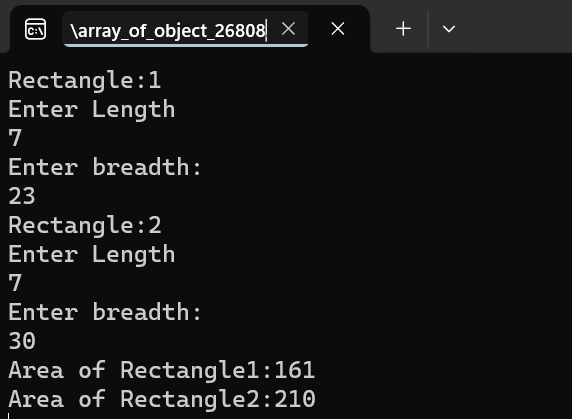
public int getArea()

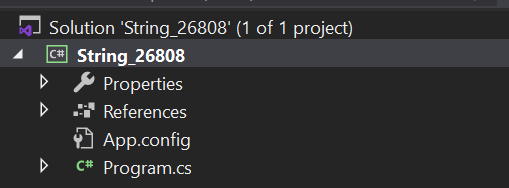
{

return length \* breadth;

}

}

**OUTPUT:**

**LAB-09: Write a C# program to find the position of a specified word in a given string.**

**SOURCE CODE:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace String\_26808

{

class Program

{

static void Main(string[] args)

{

Finder f =new Finder();

string str1 = "The cat sat on the mat";

Console.WriteLine("Original String:"+str1);

Console.Write("Position of the word 'sat' is:" + f.GetResult(str1,"sat"));

Console.Write("\nPosition of the word 'the' is:" + f.GetResult(str1, "the"));

Console.Write("\nPosition of the word 'cat' is:" + f.GetResult(str1, "cat"));

Console.ReadLine();

}

}

}

public class Finder

{

public int GetResult(string text, string word)

{

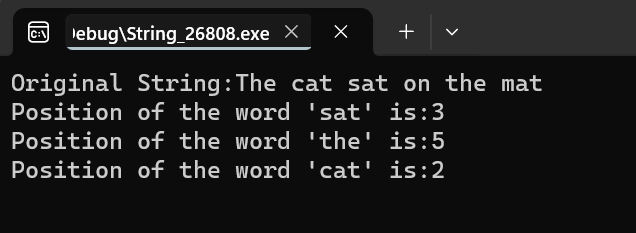
string[] str = text.Split(' ');

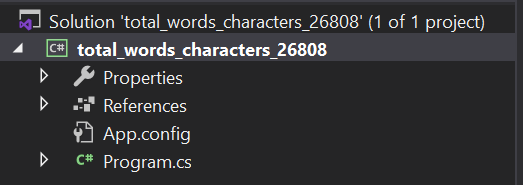
int a = Array.IndexOf(str, word)+1;

return a;

}

}

**OUTPUT:**

**LAB-10: Write a C# program to count the total number of words and characters in a string.**

**SOURCE CODE:**

using System;

namespace total\_words\_characters\_26808

{

class Program

{

static void Main(string[] args)

{

string inputString= "The quick brown fox jumps over the lazy dog.";

int wordCount = CountWords(inputString);

int charCount = inputString.Length;

Console.WriteLine("String: " + inputString);

Console.WriteLine("Total number of words: " + wordCount);

Console.WriteLine("Total number of characters: " + charCount);

Console.ReadLine();

}

static int CountWords(string input)

{

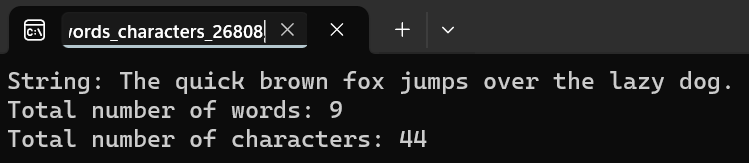
string[] words=input.Split(new char[] {' ','\n','\t','\r' }, StringSplitOptions.RemoveEmptyEntries);

return words.Length ;

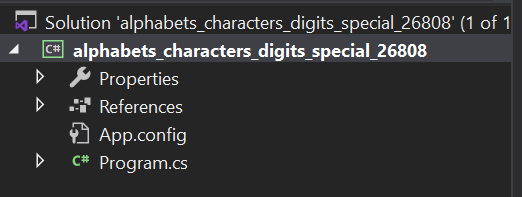
}

}

}

**OUTPUT:**

**LAB-11: Write a C# program to count the number of alphabets, digits and special characters in string.**

****

**SOURCE CODE:**

using System;

namespace alphabets\_characters\_digits\_special\_26808

{

class Program

{

static void Main(string[] args)

{

string inputString = "Hey!Would you like to have 1 cup of tea??";

int alphabetCount = 0;

int digitCount = 0;

int specialCharCount = 0;

foreach (char character in inputString)

{

if(Char.IsLetter(character))

{

alphabetCount++;

}

else if (Char.IsDigit(character))

{

digitCount++;

}

else

{

specialCharCount++;

}

}

Console.WriteLine("Number of alphabets:" + alphabetCount);

Console.WriteLine("Number of digits:" + digitCount);

Console.WriteLine("Number of special characters:" + specialCharCount);

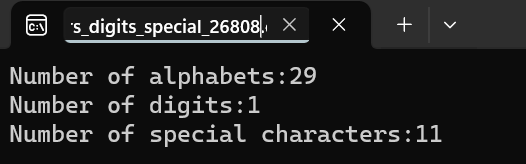
Console.ReadLine();

}

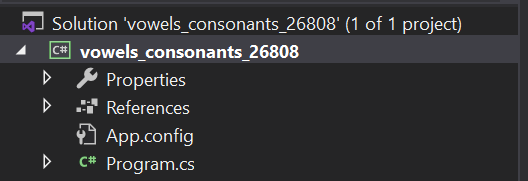
}

}

**OUTPUT:**

****

**LAB-12: Write a C# program to count the number of vowels or consonants in a string.**

****

**SOURCE CODE:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace vowels\_consonants\_26808

{

class Program

{

static void Main(string[] args)

{

string inputString = "Checking of vowel and consonants";

inputString = inputString.ToLower();

int vowelCount = 0;

int consonantCount = 0;

foreach (char character in inputString)

{

if (Char.IsLetter(character))

{

if("aeiou".Contains(character))

{

vowelCount++;

}

else

{

consonantCount++;

}

}

}

Console.WriteLine("Input String:" + inputString);

Console.WriteLine("Number of vowels:"+vowelCount);

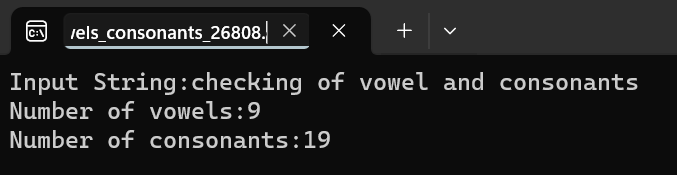
Console.WriteLine("Number of consonants:" + consonantCount);

Console.ReadLine();

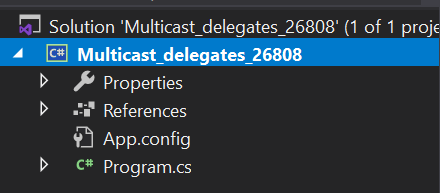
}

}

}

**OUTPUT:**

**LAB-13: Write a C# program to calculate sum and difference of two digit using multicast delegates.**

****

**SOURCE CODE:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Multicast\_delegates\_26808

{

public delegate void delmethod(int x, int y);

public class TestMultipleDelegate

{

public void plus\_Method1(int x,int y)

{

Console.WriteLine("You are in plus\_Method");

Console.WriteLine(x + y);

}

public void subtract\_Method2(int x, int y)

{

Console.WriteLine("You are in subtract\_Method");

Console.WriteLine(x - y);

}

}

class Program

{

static void Main(string[] args)

{

TestMultipleDelegate obj = new TestMultipleDelegate();

delmethod del = new delmethod(obj.plus\_Method1);

del +=new delmethod(obj.subtract\_Method2);

del(30, 15);

Console.WriteLine();

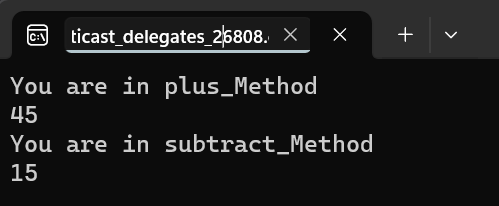
Console.ReadLine();

}

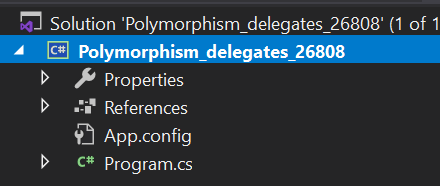
}

}

**OUTPUT:**

****

**LAB-14: Write a C# program to achieve polymorphism using delegates.**

****

**SOURCE CODE:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Polymorphism\_delegates\_26808

{

class Program

{

static void Main(string[] args)

{

GodDelegate godMethod;

godMethod = new Ram().GodName;

godMethod();

godMethod = new Sita().GodName;

godMethod();

Console.ReadLine();

}

}

delegate void GodDelegate();

class Ram

{

public void GodName()

{

Console.WriteLine("God Ram");

}

}

class Sita

{

public void GodName()

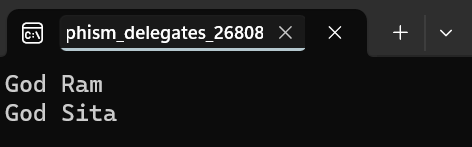
{

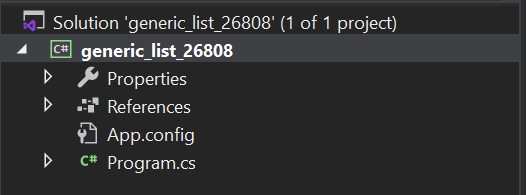
Console.WriteLine("God Sita");

}

}

}

**OUTPUT:**

**LAB-15: Write a C# program to create list with properties for id, name, gender and address. Then creates a list<Student> to store instances of this class. The program adds 10 students to the list, prints the list, and then searches for a student by their address using the findstudentbyaddress function. Finally, it prints the result of the search.**

**SOURCE CODE:**

using System;

namespace generic\_list\_26808

{

class Program

{

static void Main(string[] args)

{

List<Student> st = new List<Student>();

st.Add(new Student() { ID = 1, Name = "Nirdeshika Chuhan", Address = "Kathmandu", Gender = "Female" });

st.Add(new Student() { ID = 2, Name = "Sayal Karki", Address = "Kathmandu", Gender = "Male" });

st.Add(new Student() { ID = 3, Name = "Nirdosh Chuhan", Address = "Jhapa", Gender = "Male" });

st.Add(new Student() { ID = 4, Name = "Nikita Chuhan", Address = "Kathmandu", Gender = "Female" });

st.Add(new Student() { ID = 5, Name = "Samjhana Dhakal", Address = "Ramechap", Gender = "Female" });

st.Add(new Student() { ID = 6, Name = "Sarita Chuhan", Address = "Nuwakot", Gender = "Female" });

st.Add(new Student() { ID = 7, Name = "Chabilal Chuhan", Address = "Kathmandu", Gender = "Male" });

st.Add(new Student() { ID = 8, Name = "Sulav Karki", Address = "Bhaktapur", Gender = "Male" });

st.Add(new Student() { ID = 9, Name = "Saurav Kathayat", Address = "Bhaktapur", Gender = "Male" });

st.Add(new Student() { ID = 10, Name = "Ikita Shrestha", Address = "Kathmandu", Gender = "Female" });

List<Student> filterStudent = FindStudentByAddress(st, "Kathmandu");

foreach (var student in filterStudent)

{

Console.WriteLine("Id:{0} Name:{1} Address:{2} Gender:{3} ", student.ID, student.Name, student.Address, student.Gender);

}

Console.ReadLine();

}

public static List<Student> FindStudentByAddress(List<Student> students, string searchAddress)

{

List<Student> filterStudent = new List<Student>();

foreach (Student item in students)

{

if (item.Address==searchAddress)

{

filterStudent.Add(item);

}

}

return filterStudent;

}

}

}

public class Student

{

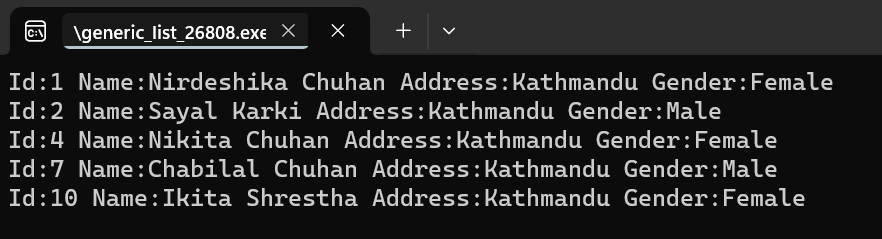
public int ID { get; set; }

public string Name { get; set; }

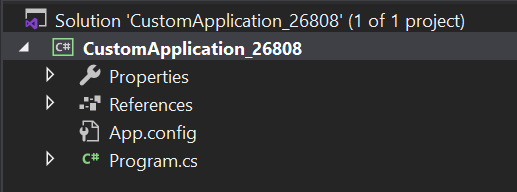
public string Address { get; set; }

public string Gender { get; set; }

}

**OUTPUT:**

**LAB-16: Write a C# program to raise an ApplicationException using Custom MyException class.**

****

**SOURCE CODE:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CustomApplication\_26808

{

class MyException : Exception

{

public void MyCustomException()

{

Console.WriteLine("Exception occurred, divisor should not be zero.");

}

}

class Program

{

static void Main(string[] args)

{

int d, div, res;

d = Int32.Parse(Console.ReadLine());

div = Int32.Parse(Console.ReadLine());

try

{

if (div == 0)

{

throw new MyException();

}

res = d / div;

Console.WriteLine("Result:{0}", res);

Console.ReadLine();

}

catch(MyException e)

{

e.MyCustomException();

}

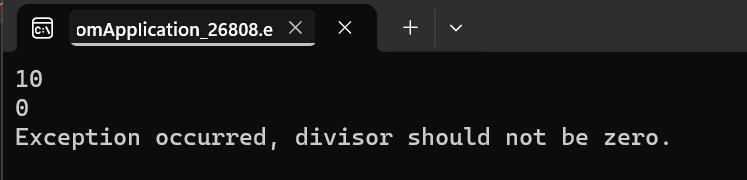
Console.ReadLine();

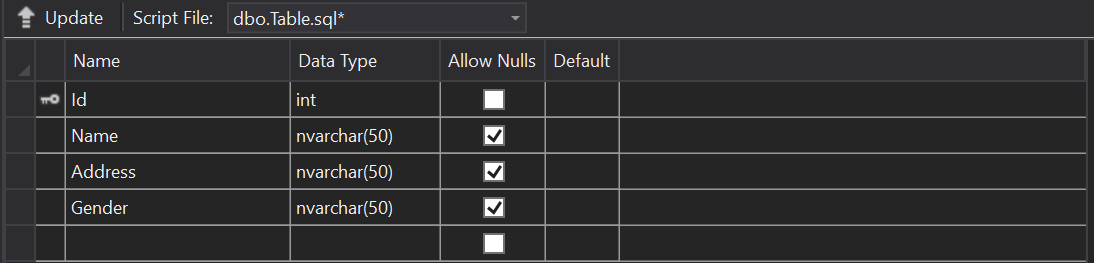
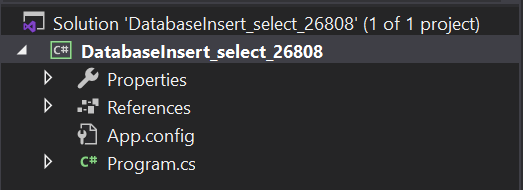
}

}

}

**OUTPUT:**



******LAB-17: Write a C# program to show insert and select student record with given table (tblStudent) with fields (int id, nvarchar(50) name, nvarchar(50) address, nvarchar(50) gender).Also display total no of student from table.**

**SOURCE CODE:**

using System;

using System.Data.SqlClient;//MS SQL Server

using System.Data;

namespace DatabaseInsert\_select\_26808

{

class Program

{

static void Main(string[] args)

{

Student st = new Student();

Console.Write("Enter total number of data:");

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

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

{

Console.WriteLine("Student" + (i + 1));

Console.Write("Enter name:");

string name = Console.ReadLine();

Console.Write("Enter Address:");

string address = Console.ReadLine();

Console.Write("Enter Gender:");

string gender = Console.ReadLine();

st.InsertStudent(name, address, gender);

}

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

st.DisplayStudentData();

Console.ReadLine();

}

}

public class Student

{

public void InsertStudent(string name, string address, string gender)

{

string conStr = "Data Source=(LocalDB)\\MSSQLLocalDB;Database=Samriddhi\_database;Integrated Security=true";

SqlConnection con = new SqlConnection(conStr);

SqlCommand cmd = new SqlCommand("insert into tblStudent values (@name,@address,@gender)", con);

cmd.Parameters.AddWithValue("@name", name);

cmd.Parameters.AddWithValue("@address", address);

cmd.Parameters.AddWithValue("@gender", gender);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

public void DisplayStudentData()

{

string conStr = "Data Source=(LocalDB)\\MSSQLLocalDB;Database=Samriddhi\_database; Integrated Security=true";

SqlConnection con = new SqlConnection(conStr);

SqlCommand cmd = new SqlCommand("select \*from tblStudent", con);

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

da.Fill(dt);

Console.WriteLine("Student Records");

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

{

Console.WriteLine("Name:{0} Address:{1} Gender:{2}", dt.Rows[i]["Name"].ToString(), dt.Rows[i]["Address"].ToString(), dt.Rows[i]["Gender"].ToString());

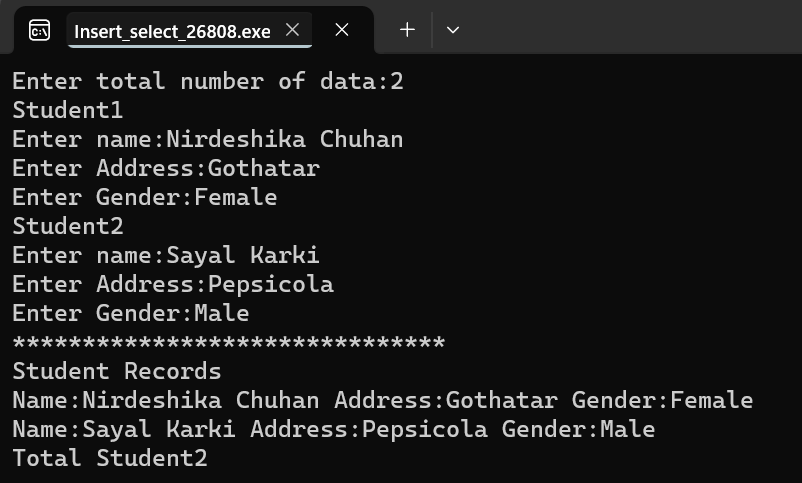
}

Console.Write("Total Student" + dt.Rows.Count);

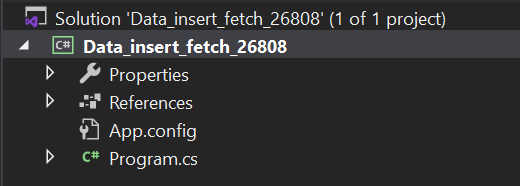
}

}

}

**OUTPUT:**

**LAB-18: Write a C# program to show insert and fetch student record by Gender from given table (tblStudent) with fields (int id, nvarchar(50) name, nvarchar(50) address, nvarchar(50) gender).**



**SOURCE CODE:**

using System;

using System.Data.SqlClient;

using System.Data;

namespace Data\_insert\_fetch\_26808

{

internal class Program

{

static void Main(string[] args)

{

StudentInsert\_SelectRecord obj = new StudentInsert\_SelectRecord();

Console.Write("Enter Name:");

string name = Console.ReadLine();

Console.Write("Enter Address:");

string address = Console.ReadLine();

Console.Write("Enter Gender:");

string gender = Console.ReadLine();

obj.Insert(name, address, gender);

Console.WriteLine("Record Inserted");

Console.WriteLine();

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

Console.WriteLine("All Student By Gender");

DataTable dt = obj.GetAllStudentByGender("Male");

for (int i = 0; i < dt.Rows.Count; i++){

string n = dt.Rows[i]["Name"].ToString();

string a = dt.Rows[i]["Address"].ToString();

string g = dt.Rows[i]["Gender"].ToString();

Console.WriteLine("Name:{0} Address:{1} Gender:{2}", n, a, g);

}

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

Console.WriteLine("Total No of Student: " + dt.Rows.Count);

Console.ReadLine();

}

}

public class StudentInsert\_SelectRecord

{

public void Insert(string name, string address, string gender)

{

string connStr = @"Data Source=(localdb)\MSSqlLocalDB; Database=Samriddhi\_DB; Integrated Security=true";

SqlConnection con = new SqlConnection(connStr);

string sql = "insert into tblStudent values(@name,@address,@gender)";

SqlCommand cmd = new SqlCommand(sql, con);

cmd.Parameters.AddWithValue("@name", name);

cmd.Parameters.AddWithValue("@address", address);

cmd.Parameters.AddWithValue("@gender", gender);

con.Open();

cmd.ExecuteReader();

con.Close();

}

public DataTable GetAllStudentByGender(string gender)

{

string connStr = @"Data Source=(localdb)\MSSqlLocalDB; Database=Samriddhi\_DB; Integrated Security=true";

SqlConnection con = new SqlConnection(connStr);

string sql = "select \*from tblStudent where Gender=@gender";

SqlCommand cmd = new SqlCommand(sql, con);

cmd.Parameters.AddWithValue("@gender", gender);

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();//can hold data in tabular format

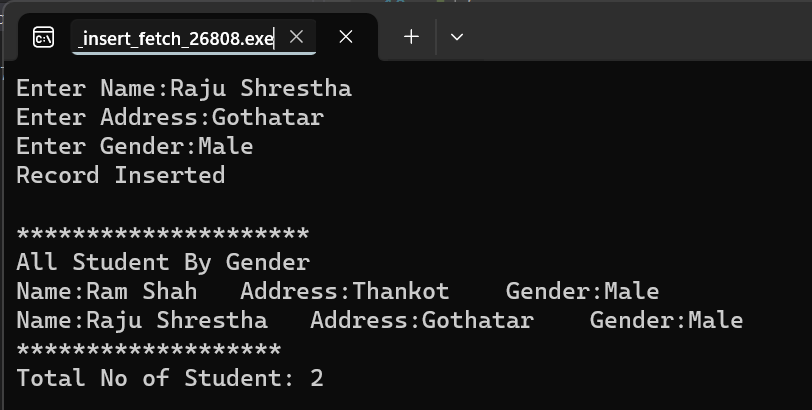
da.Fill(dt);

return dt;

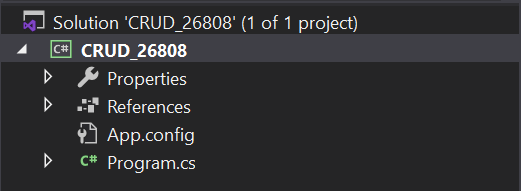
}

}

}

**OUTPUT:**

**LAB-19: Write a C# program to perform (CRUD) operation from given table (tblStudent) with fields (int id, nvarchar(50) name, nvarchar(50) address, nvarchar(50) gender).**

****

**SOURCE CODE:**

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

namespace CRUD\_26808

{

internal class Program

{

static void Main(string[] args)

{

Student st = new Student();

Console.WriteLine("Enter Option");

Console.WriteLine("1 For Insert Student");

Console.WriteLine("2 For Update Student");

Console.WriteLine("3 For Delete Student");

Console.WriteLine("4 For Fetch All Student");

string option = Console.ReadLine();

switch (option)

{

case "1":

Console.Write("Enter Name: ");

string name = Console.ReadLine();

Console.Write("Enter Address: ");

string address = Console.ReadLine();

Console.Write("Enter Gender: ");

string gender = Console.ReadLine();

st.InsertStudent(name, address, gender);

Console.WriteLine("Record Inserted");

break;

case "2":

Console.Write("Enter Id To Update: ");

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

Console.Write("Enter Name: ");

string uname = Console.ReadLine();

Console.Write("Enter Address: ");

string uaddress = Console.ReadLine();

Console.Write("Enter Gender: ");

string ugender = Console.ReadLine();

st.UpdateStudent(uname, uaddress, ugender, id);

Console.WriteLine("Record Updated");

break;

case "3":

Console.Write("Enter Id To Update: ");

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

st.DeleteStudent(did);

Console.WriteLine("Record Deleted");

break;

case "4":

DataTable dt = st.DisplayStudentData();

for (int i = 0; i < dt.Rows.Count; i++){

Console.WriteLine("Name:{0} Address:{1} Gender:{2}",

dt.Rows[i]["Name"], dt.Rows[i]["Address"], dt.Rows[i]["Gender"]);

}

break;

default:

break;

}

Console.ReadLine();

}

}

public class Student

{

public void InsertStudent(string name, string address, string gender)

{

string connStr = @"Data Source=(localdb)\MSSqlLocalDB; Database=Samriddhi\_database; Integrated Security=true";

SqlConnection con = new SqlConnection(connStr);

string sql = "insert into tblStudent values(@name,@address,@gender)";

SqlCommand cmd = new SqlCommand(sql, con);

cmd.Parameters.AddWithValue("@name", name);

cmd.Parameters.AddWithValue("@address", address);

cmd.Parameters.AddWithValue("@gender", gender);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

public void UpdateStudent(string name, string address, string gender, int id)

{

string connStr = @"Data Source=(localdb)\MSSqlLocalDB; Database=Samriddhi\_database; Integrated Security=true";

SqlConnection con = new SqlConnection(connStr);

string sql = "update tblStudent set Name=@name, Address=@address, Gender=@gender where Id=@id";

SqlCommand cmd = new SqlCommand(sql, con);

cmd.Parameters.AddWithValue("@name", name);

cmd.Parameters.AddWithValue("@address", address);

cmd.Parameters.AddWithValue("@gender", gender);

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

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

public void DeleteStudent(int id)

{

string connStr = @"Data Source=(localdb)\MSSqlLocalDB; Database=Samriddhi\_database; Integrated Security=true";

SqlConnection con = new SqlConnection(connStr);

string sql = "delete from tblStudent where Id=@id";

SqlCommand cmd = new SqlCommand(sql, con);

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

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

public DataTable DisplayStudentData()

{

string conStr = "Data Source=(LocalDB)\\MSSQLLocalDB; Database=Samriddhi\_database; Integrated Security=true";

SqlConnection con = new SqlConnection(conStr);//

SqlCommand cmd = new SqlCommand("select \*from tblStudent", con);

SqlDataAdapter da = new SqlDataAdapter(cmd);//works as mediator between datasource=datatable

DataTable dt = new DataTable();//row column

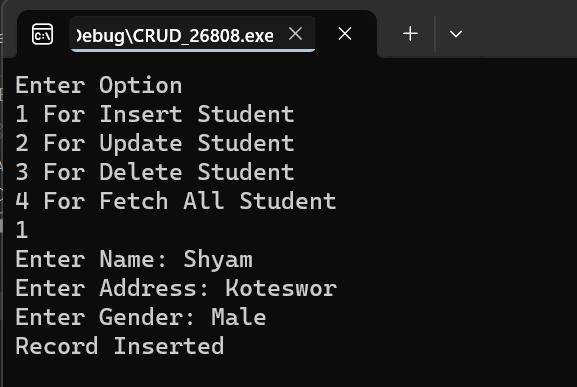
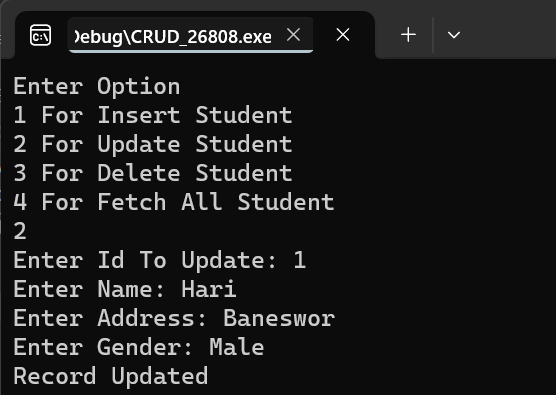
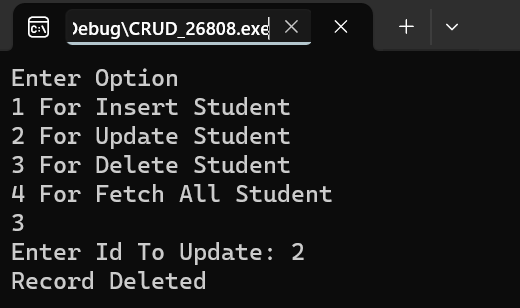
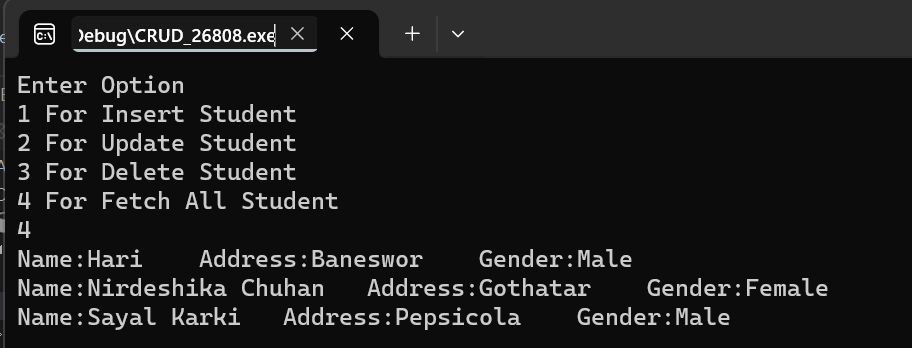
da.Fill(dt);

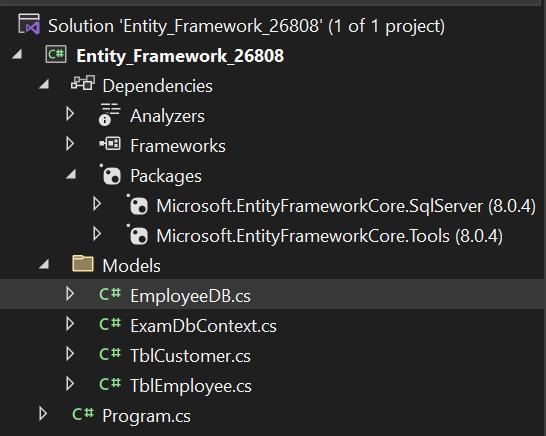
return dt;

}

}

}

**OUTPUT:**

**LAB-20: Write a C# Program to insert and display records in tablename tblEmployee (id int, name nvarchar(50), address nvarchar(50), salary decimal(18,0), gender nvarchar50) using EntityFramework.**

**SOURCE CODE:**

**EmployeeDB.cs:**

using System;

namespace Entity\_Framework\_26808.Models

{

internal class EmployeeDB

{

ExamDbContext db = new ExamDbContext();

public void CreateEmployee(TblEmployee tb)

{

db.TblEmployees.Add(tb);

db.SaveChanges();

Console.WriteLine("Data Saved");

}

public List<TblEmployee> GetAllEmployees()

{

List<TblEmployee> lst = db.TblEmployees.ToList();

return lst;

}

}

}

**Program.cs:**

using Entity\_Framework\_26808.Models;

EmployeeDB emp=new EmployeeDB();

TblEmployee tb = new TblEmployee();

tb.Name = "Ram Shah";

tb.Address = "Bhaktapur";

tb.Salary = 20000;

tb.Gender = "Male";

emp.CreateEmployee(tb);

List<TblEmployee> lst = emp.GetAllEmployees();

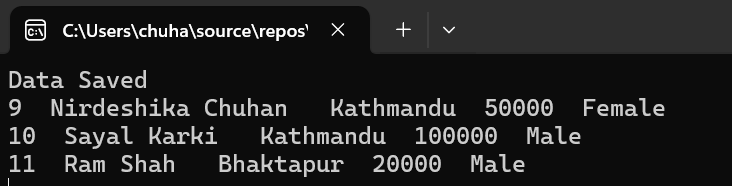
foreach (TblEmployee item in lst)

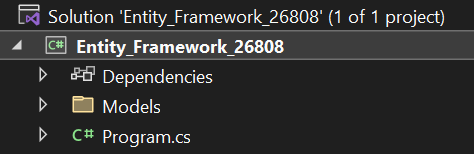
{

Console.WriteLine("{0} {1} {2} {3} {4}", item.Id, item.Name, item.Address, item.Salary, item.Gender);

}

Console.ReadLine();

**OUTPUT:**

**LAB-21: Write a C# program to compute aggregate salary of 5 employee and then display employee record in descending order with respect to employee salary using EntityFramework.**

**SOURCE CODE:**

using Entity\_Framework\_26808.Models;

ExamDbContext db = new ExamDbContext();

string totalsalary = db.TblEmployees.Take(5).Sum(a=>a.Salary).ToString();

decimal averagesalary = Convert.ToDecimal(totalsalary) / 5;

Console.WriteLine("Average Salary:" +averagesalary);

List<TblEmployee> emp = db.TblEmployees.OrderByDescending(a=>a.Salary).ToList();

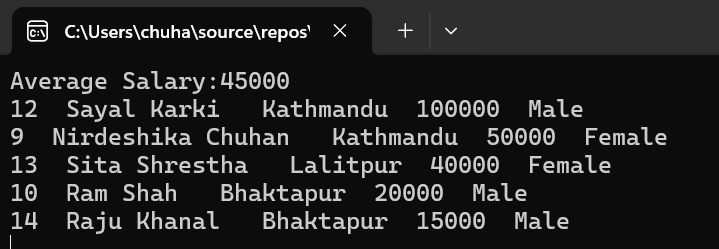
foreach (TblEmployee item in emp)

{

Console.WriteLine("{0} {1} {2} {3} {4}", item.Id, item.Name, item.Address, item.Salary, item.Gender);

}

Console.ReadLine();

**OUTPUT:**

**LAB-22: Write a program to select employees whose salary is greater than 20000 and whose address is Kathmandu using EntityFramework.**

**SOURCE CODE:**

using Entity\_Framework\_26808.Models;

ExamDbContext db = new ExamDbContext();

List<TblEmployee> emp = db.TblEmployees.Where(a=>a.Salary>20000 && a.Address== "Kathmandu").ToList();

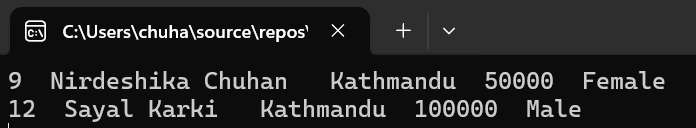
foreach (TblEmployee item in emp)

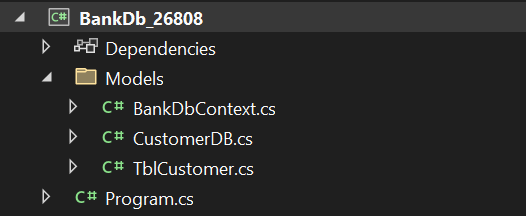
{

Console.WriteLine("{0} {1} {2} {3} {4}", item.Id, item.Name, item.Address, item.Salary, item.Gender);

}

Console.ReadLine();

**OUTPUT:**

**LAB-23: Write a C# program to connect database Bank and insert 5 costumer record (Account\_no, Name, Address, Balance) in Customer table and display the customer record whose balance is greater than 5000 using EntityFramework.**

**SOURCE CODE:**

**CustomerDB.cs:**

namespace BankDb\_26808.Models

{

internal class CustomerDB

{

BankDbContext db = new BankDbContext();

public void CreateCustomer(TblCustomer tb)

{

db.TblCustomers.Add(tb);

db.SaveChanges();

}

public List<TblCustomer> GetAllCustomers()

{

List<TblCustomer> lst = db.TblCustomers.ToList();

return lst;

}

}

}

**Program.cs:**

using BankDb\_26808.Models;

CustomerDB cus = new CustomerDB();

Console.WriteLine("Enter how many records to be inserted:");

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

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

{

TblCustomer tb = new TblCustomer();

Console.WriteLine("Insert Data of Customer No. " + (i + 1));

Console.WriteLine("Account number:");

tb.AccountNo = Console.ReadLine();

Console.WriteLine("Name:");

tb.Name = Console.ReadLine();

Console.WriteLine("Address:");

tb.Address = Console.ReadLine();

Console.WriteLine("Balance:");

tb.Balance = Convert.ToDecimal(Console.ReadLine());

cus.CreateCustomer(tb);

}

Console.WriteLine("All Data Inserted Successfully");

List<TblCustomer> lst = cus.GetAllCustomers().Where(a => a.Balance > 5000).ToList();

foreach (TblCustomer item in lst)

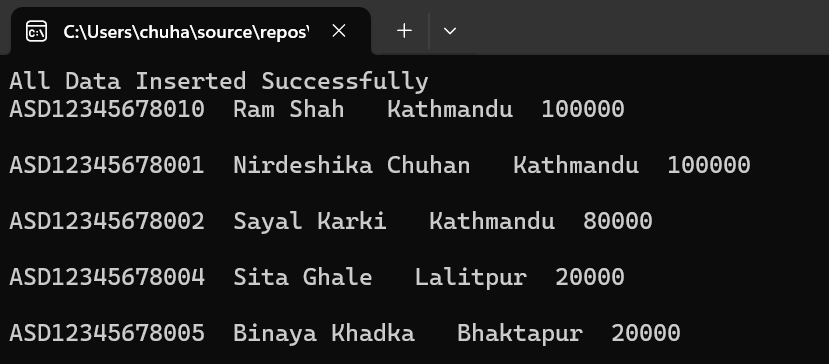
{

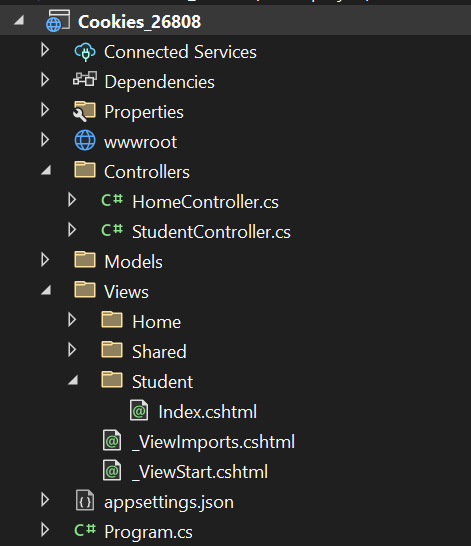
Console.WriteLine("{0} {1} {2} {3}", item.AccountNo, item.Name, item.Address, item.Balance);

Console.WriteLine();

}

Console.ReadLine();

**OUTPUT:**

**LAB-24: Create a new project and add a controller with endpoints to read and write values into cookies.**

**SOURCE CODE:**

**Program.cs:**

app.MapControllerRoute(

name: "default",

pattern: "{controller=Student}/{action=Index}/{id?}"

);

**StudentController.cs:**

using Microsoft.AspNetCore.Mvc;

namespace Cookies\_26808.Controllers

{

public class StudentController : Controller

{

[HttpGet]

public IActionResult Index()

{

//Read cookie from Request Object

string userName = Request.Cookies["UserName"];

return View("Index",userName);

}

[HttpPost]

public IActionResult Index(IFormCollection form)

{

string userName = form["userName"].ToString();

//Set the key value in Cookie

CookieOptions option= new CookieOptions();

option.Expires= DateTime.Now.AddMinutes(10);

Response.Cookies.Append("UserName", userName, option);

return RedirectToAction(nameof(Index));

}

public IActionResult RemoveCookie()

{

//Delete the cookie

Response.Cookies.Delete("UserName");

return View("Index");

}

}

}

**Index.cshtml:**

@model string

@{

ViewData["Title"] = "Cookie";

}

@if(!string.IsNullOrWhiteSpace(Model))

{

<div>Welcome back,@Model</div>

//<a asp-controller="Student" asp-action="Remove Cookie">Forget Me</a>

@Html.ActionLink("Forget Me","RemoveCookie")

}

else

{

<form asp-action="Index">

<label>Please provide your name:</label>

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

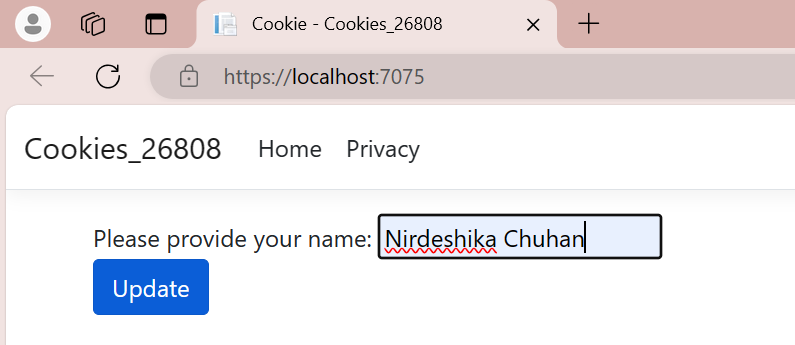
<div class="form-group">

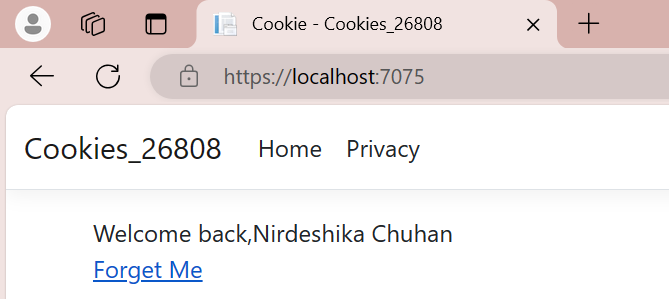
<input type="submit" value="Update" class="btn btn-primary" />

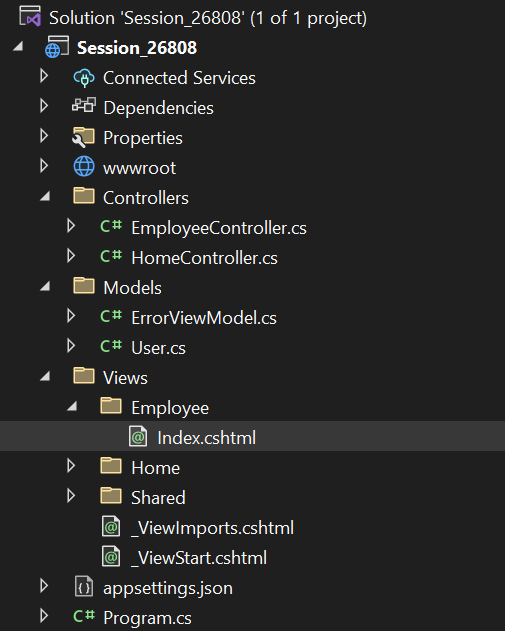
</div>

</form>

}

**OUTPUT:**

****

**LAB-25: Create a controller with endpoints to set and read a value from the session.**

**SOURCE CODE:**

**Program.cs:**

builder.Services.AddSession();

var app = builder.Build();

app.UseSession();

pattern: "{controller=Employee}/{action=Index}/{id?}"

**EmployeeController.cs:**

using Microsoft.AspNetCore.Mvc;

using Session\_26808.Models;

namespace Session\_26808.Controllers

{

public class EmployeeController : Controller

{

public IActionResult Index()

{

//Write session

HttpContext.Session.SetString("Name", "Nirdeshika Chuhan");

HttpContext.Session.SetInt32("Age",36);

User newUser = new User();

//Read Session

newUser.Name = HttpContext.Session.GetString("Name");

newUser.Age=HttpContext.Session.GetInt32("Age").Value;

return View(newUser);

}

}

}

**User.cs:**

namespace Session\_26808.Models

{

public class User

{

public int? Id { get; set; }

public string? Name { get; set; }

public int? Age { get; set; }

}

}

**Index.cshtml:**

@model Session\_26808.Models.User

@{

ViewData["Title"] = "Index";

}

<h1>Index</h1>

<div>

<h4>User</h4>

<hr />

<dl class="row">

<dt class="col-sm-2">

Id

</dt>

<dd class="col-sm-10">

@Model.Id

</dd>

<dt class = "col-sm-2">

Name

</dt>

<dd class = "col-sm-10">

@Model.Name

</dd>

<dt class = "col-sm-2">

Age

</dt>

<dd class = "col-sm-10">

@Model.Age

</dd>

</dl>

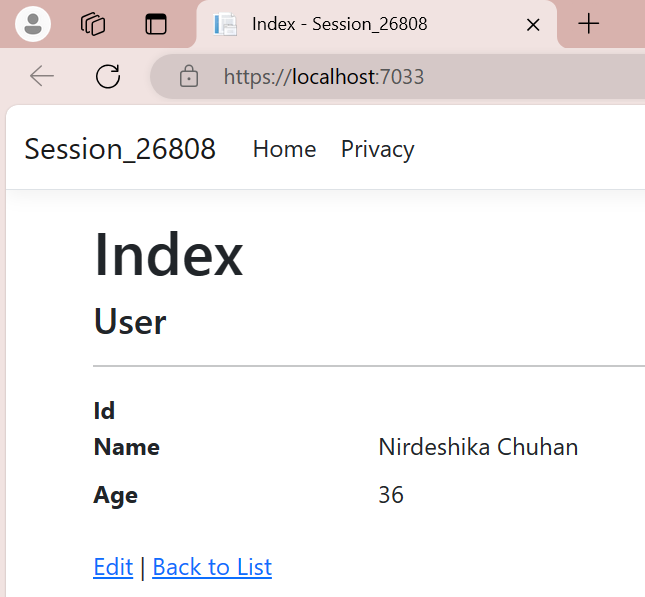
</div>

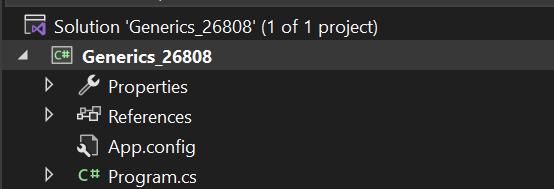
<div>

@Html.ActionLink("Edit", "Edit", new { /\* id = Model.PrimaryKey \*/ }) |

<**a** **asp-action**="Index">Back to List</**a**>

</div>

**OUTPUT:**

**LAB-26: Write a simple program to create generic class with generic constructor, generic member variable, generic property and generic method.**

**SOURCE CODE:**

using System;

namespace Generics\_26808

{

internal class Program

{

static void Main(string[] args)

{

//Create an instanve of MyGenericClass with int type

MyGenericClass<int> intGenericClass = new MyGenericClass<int>(10);

intGenericClass.DisplayGenericValue();

intGenericClass.GenericProperty = 20;

intGenericClass.DisplayGenericValue();

//Create an instance of MyGenericClass with string type

MyGenericClass<string> stringGenericClass = new MyGenericClass<string>("Hello, World!");

stringGenericClass.DisplayGenericValue();

stringGenericClass.GenericProperty = "Goodbye, Samriddhi College!";

stringGenericClass.DisplayGenericValue();

Console.ReadLine();

}

}

public class MyGenericClass<T>

{

private T variable;

//Generic Constructor

public MyGenericClass(T value) {

this.variable = value;

}

//Generic Property

public T GenericProperty

{

get { return variable; }

set { variable = value; }

}

//Generic method

public void DisplayGenericValue()

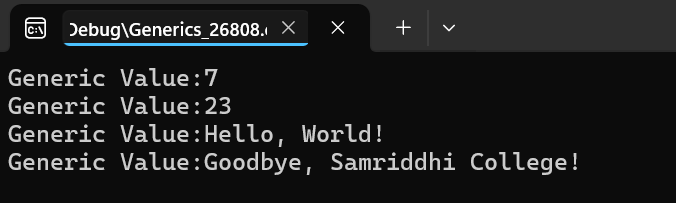
{

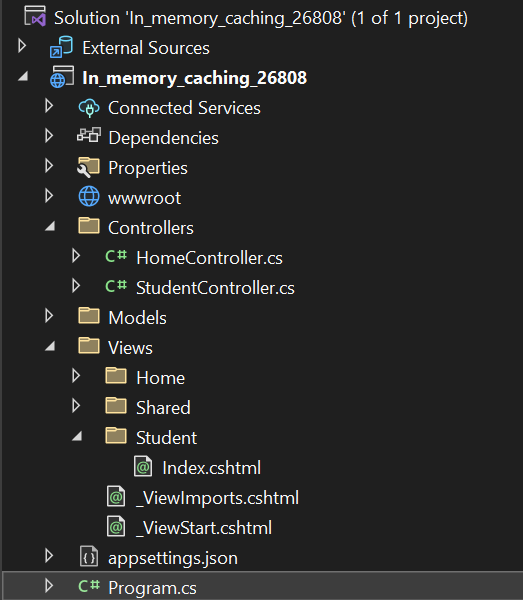
Console.WriteLine($"Generic Value:{variable}");

}

}

}

**OUTPUT:**

**LAB-27: Write a asp.net core mvc application to demonstrate in-memory caching.**

**SOURCE CODE:**

**Program.cs:**

Add line of code to enable in Memorycaching in Program.cs:

builder.Services.AddMemoryCache();

**StudentController.cs:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Caching.Memory;

namespace In\_memory\_caching\_26808.Controllers

{

public class StudentController : Controller

{

private readonly IMemoryCache memoryCache;

public StudentController(IMemoryCache memoryCache)

{

this.memoryCache = memoryCache;

}

public IActionResult Index()

{

DateTime currentTime;

bool isExist =memoryCache.TryGetValue("CacheTime",out currentTime);

if(!isExist)

{

currentTime = DateTime.Now;

memoryCache.Set("CacheTime",currentTime,TimeSpan.FromSeconds(30));

}

return View("Index",currentTime.ToLongTimeString());

}

}

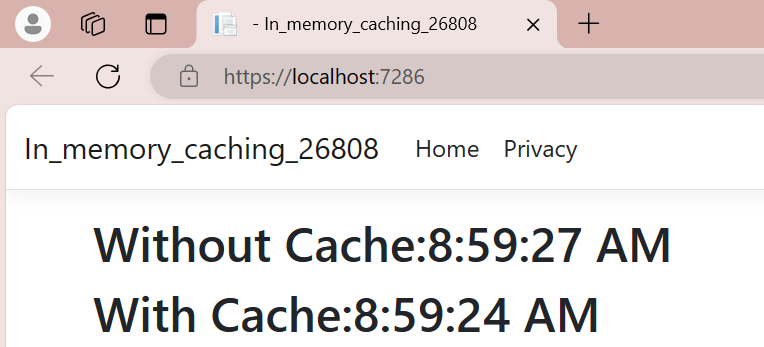
}

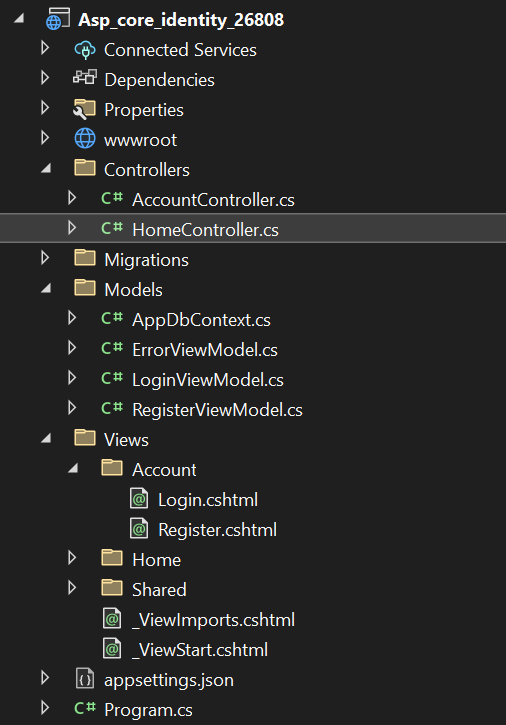
**Index.cshtml:**

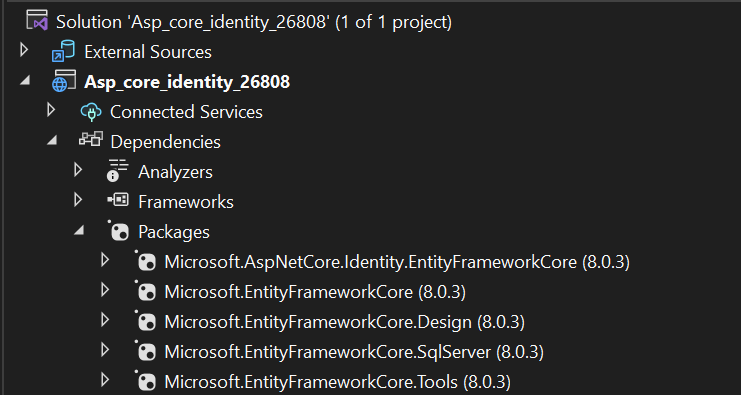
@model string

<h2>Without Cache:@DateTime.Now.ToLongTimeString()</h2>

<h2>With Cache:@Model</h2>

**OUTPUT:**

**LAB-28: Write a asp.net core MVC application to demonstrate simple login authentication using Asp.net core identity.**

**Install All Packages below from Nuget:**

Add a class **AppDbContext.cs** inside **Model** Folder

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

namespace Asp\_core\_identity\_26808.Models

{

public class AppDbContext:IdentityDbContext

{

public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)

{

}

protected override void OnModelCreating(ModelBuilder builder)

{

base.OnModelCreating(builder);

}

}

}

Add **RegisterViewModel** inside **Model** Folder

namespace Asp\_core\_identity\_26808.Models

{

public class RegisterViewModel

{

public string Email { get; set; }

public string Password { get; set; }

public string ConfirmPassword { get; set; }

}

}

Add **LoginViewModel** inside **Model** Folder

using System.ComponentModel.DataAnnotations;

namespace Asp\_core\_identity\_26808.Models

{

public class LoginViewModel

{

[Required]

[EmailAddress]

public string Email { get; set; }

[Required]

[DataType(DataType.Password)]

public string Password { get; set; }

[Display(Name = "Remember me?")]

public bool RememberMe { get; set; }

}

}

Add line of code inside **Program.cs** inside **ConfigureServices Method**

Builder.Services.AddDbContextPool<AppDbContext>(options => options.UseSqlServer("Data Source=.; Initial Catalog=CoreDB; Integrated Security=true"));

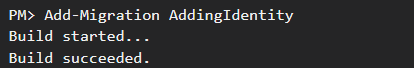
Builder.Services.AddIdentity<IdentityUser,IdentityRole>().AddEntityFrameworkStores<AppDbContext>();

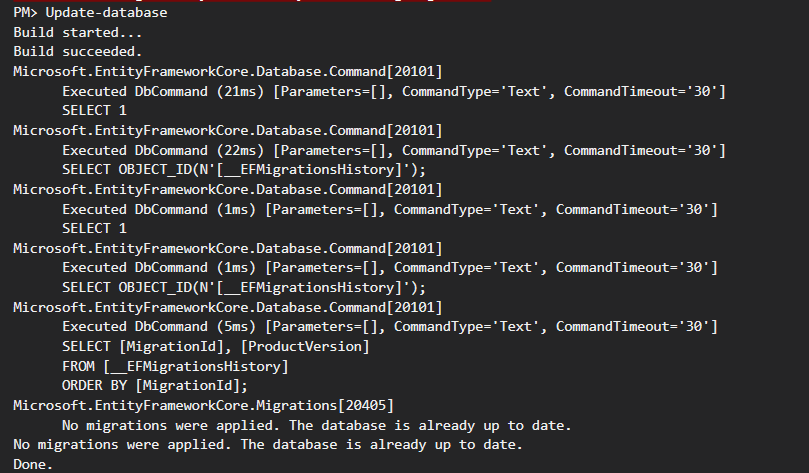
Add below line code inside **Program.cs** inside **Configure** **Method**

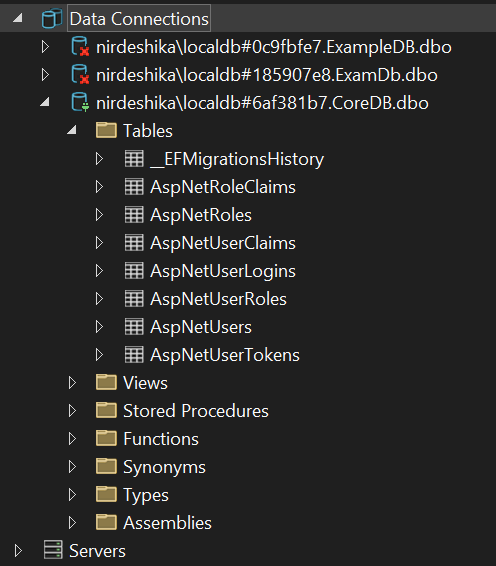
app.UseAuthentication();

**Now, Go To Tools>Nuget Package Manager>Package Manager Console**

**Type: Add-Migration AddingIdentity**

****

**Type: Update-database**

Now, it will generate following tables in databases:

Add **AccountController** inside **Controller** Folder

using Asp\_core\_identity\_26808.Models;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

namespace Asp\_core\_identity\_26808.Controllers

{

public class AccountController : Controller

{

private readonly UserManager<IdentityUser> userManager;

private readonly SignInManager<IdentityUser> signinManager;

public AccountController(UserManager<IdentityUser> userManager, SignInManager<IdentityUser> signinManager)

{

this.userManager = userManager;

this.signinManager = signinManager;

}

[HttpGet]

public IActionResult Login()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Login(LoginViewModel model, string ReturnUrl)

{

var result = await signinManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, false);

if (result.Succeeded)

{

if (!string.IsNullOrEmpty(ReturnUrl))

{

return Redirect(ReturnUrl);

}

else

{

return RedirectToAction("Index", "Home");

}

}

else

{

ModelState.AddModelError("", "Invalid Attempts");

}

return View(model);

}

public IActionResult Register()

{

return View();

}

[HttpPost]

public async Task<IActionResult> Register(RegisterViewModel model)

{

if (ModelState.IsValid)

{

var user = new IdentityUser() { UserName = model.Email, Email = model.Email };

var result = await userManager.CreateAsync(user, model.Password);

if (result.Succeeded)

{

await signinManager.SignInAsync(user, isPersistent: false);

return RedirectToAction("Index", "Home");

}

foreach (var error in result.Errors)

{

ModelState.AddModelError("", error.Description);

}

}

return View();

}

public async Task<IActionResult> Logout()

{

await signinManager.SignOutAsync();

return RedirectToAction("Login", "Account");

}

}

}

Add **Register.cshtml** inside **Account** Folder of **View** Folder

@model Asp\_core\_identity\_26808.Models.RegisterViewModel

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

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

<title>Register</title>

<link href="~/lib/bootstrap/dist/css/bootstrap.css" rel="stylesheet" />

</head>

<body>

<div class="container">

<h4>New User</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form asp-action="Register">

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

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

<div class="form-group">

<input type="submit" value="Register" class="btn btn-primary" />

<a href="@Url.Action("Login")">Login</a>

</div>

</form>

</div>

</div>

</div>

</body>

</html>

Add **Login.cshtml** inside **Account** Folder

@model Asp\_core\_identity\_26808.Models.LoginViewModel

@{

ViewData["Title"] = "Login";

Layout = null;

}

<link href="~/lib/bootstrap/dist/css/bootstrap.css" rel="stylesheet" />

<div class="container">

<h1>Login</h1>

<div class="row">

<div class="col-md-4">

<form asp-action="Login">

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

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

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

<label class="form-check-label">

<input class="form-check-input" asp-for="RememberMe" /> @Html.DisplayNameFor(model => model.RememberMe)

</label>

</div>

<div class="form-group">

<input type="submit" value="Log In" class="btn btn-primary" />

<a href="@Url.Action("Register")" class="btn btn-success">Register</a>

</div>

</form>

</div>

</div>

</div>

For Authorization use [Authorize] attribute inside **HomeController.cs**

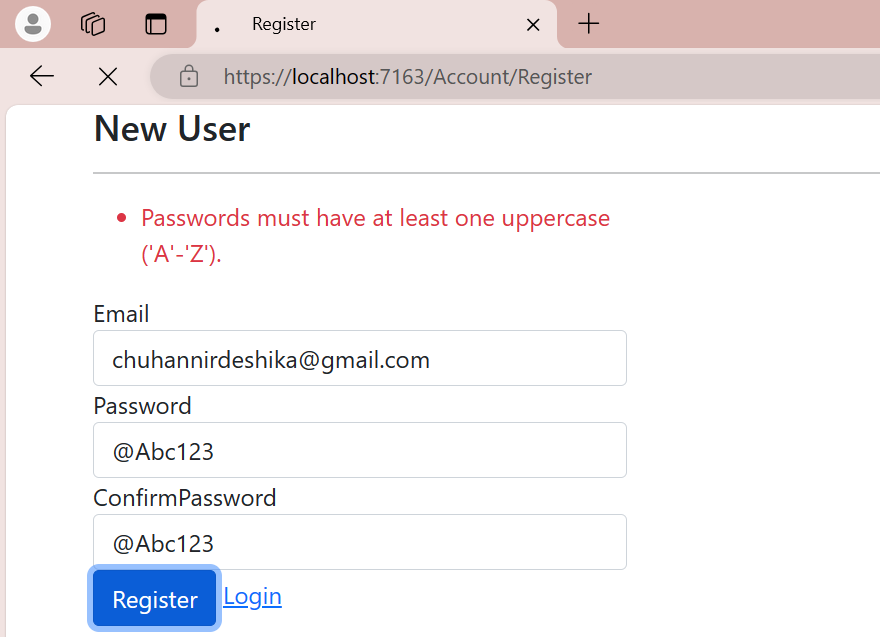
[Authorize]

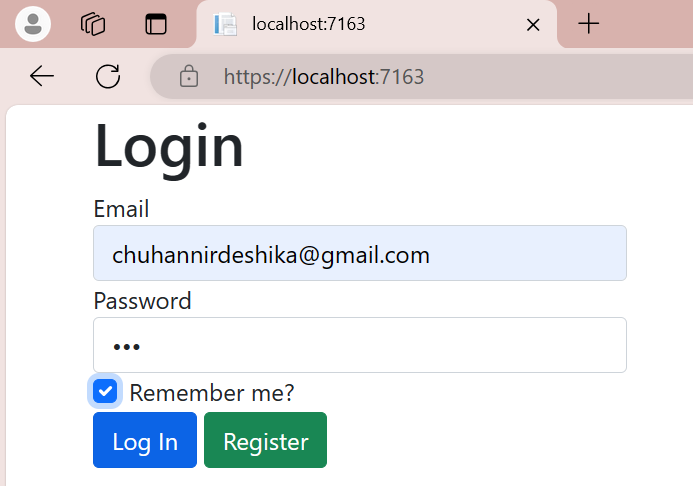
public IActionResult Privacy()

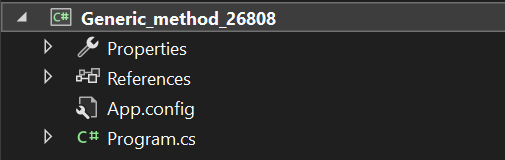
{

return View();

}

**OUTPUT:**

****

**LAB-29: Write a program in C# to** **implement a generic list data structure.**

**SOURCE CODE:**

using System;

using System.Collections.Generic;

namespace Generic\_method\_26808

{

internal class Program

{

static void Main(string[] args)

{

MyList<int> myList = new MyList<int>();

myList.Add(1);

myList.Add(2);

myList.Add(3);

Console.WriteLine("Count: " + myList.Count); // Output: 3

Console.WriteLine("Contains 2: " + myList.Contains(2)); // Output: True

Console.WriteLine("Contains 5: " + myList.Contains(5)); // Output: False

myList.Remove(2);

Console.WriteLine("Count after removal: " + myList.Count); // Output: 2

Console.ReadLine();

}

}

public class MyList<T>

{

private List<T> items = new List<T>();

public void Add(T item)

{

items.Add(item);

}

public bool Remove(T item)

{

return items.Remove(item);

}

public bool Contains(T item)

{

return items.Contains(item);

}

public int Count

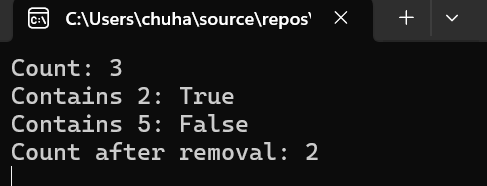
{

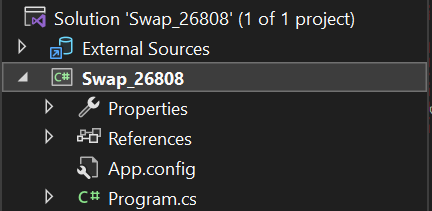
get { return items.Count; }

}

}

}

**OUTPUT:**

**LAB-30: Write a generic method called ‘Swap’ that takes two parameters of the same type and swaps their values.**

**SOURCE CODE:**

using System;

namespace Swap\_26808

{

internal class Program

{

static void Main(string[] args)

{

int a = 7, b = 23;

Console.WriteLine($"Before swap: a = {a}, b = {b}");

GenericExample.Swap(ref a, ref b);

Console.WriteLine($"After swap: a = {a}, b = {b}");

Console.ReadLine();

}

}

public class GenericExample

{

public static void Swap<T>(ref T a, ref T b)

{

T temp = a;

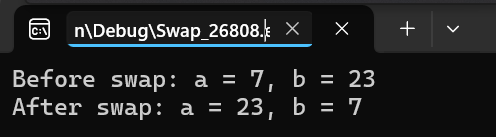
a = b;

b = temp;

}

}

}

**OUTPUT:**