

**Practical No:- 1**

**Aim:** Design ui based application using basic windows forms controls, using classes and objects design applications, using inheritance and abstract classes.

### Theory:

UI-based application using basic Windows Forms controls, along with incorporating classes, objects, inheritance, and abstract classes, revolve around the principles of Object-Oriented Programming (OOP) and Graphical User Interface (GUI) design.

Object-Oriented Programming is a programming paradigm based on the concept of "objects", which can contain data in the form of fields like attributes or properties and code in the form of procedures such as methods or functions.

**Classes:**They define the properties and behaviors that objects of the class will have.

**Objects:** Objects are instances of classes. They represent real-world entities and encapsulate data and behavior.

**Encapsulation:** Encapsulation is the bundling of data and methods that operate on that data within a single unit. It hides the internal state of an object and only exposes necessary functionalities.

**Inheritance:** Inheritance is a mechanism where a new class is derived from an existing class. The subclass inherits properties and methods from its superclass and can also have its own additional properties and methods.

**Polymorphism:** Polymorphism allows objects of different classes to be treated as objects of a common superclass.

GUI design involves creating interfaces with graphical elements (controls) such as buttons, textboxes, labels, etc., to interact with users.

Windows Forms Controls: Windows Forms provide a set of controls e.g., Button, TextBox, Label, etc. that can be placed on a form to create a user interface. These controls have properties that can be set at design time or runtime and events that can be handled to respond to user interactions.

Combining OOP with GUI design involves creating classes to represent elements of the user interface, such as windows, forms, and controls, and using inheritance to define relationships between them. Objects instantiated from these classes interact with the GUI controls to handle user input and perform operations, promoting code organization and reusability.

### Code:

1. design UI based application using basic windows forms controls.
   1. Write a program in c# to print( console base(CUI)):
      1. Factorial of a number
      2. even or odd
      3. prime number or not
      4. reverse of a number

using System;

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

using System.Text;

using System.Threading.Tasks;

namespace Practical\_1

{

internal class Program

{

static void Main(string[] args)

{

int n, f = 1;

Console.WriteLine("Enter the Number:"); n=Convert.ToInt32(Console.ReadLine()); for (int i = 1; i<n;i++)

{

f = f \* i;

}

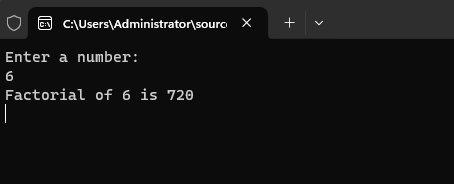
Console.WriteLine("Factorial " + n + " is a:" + f); Console.ReadKey();

}

}

}

### Output:



using System;

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

using System.Text;

using System.Threading.Tasks;

namespace practical1

{

internal class Program

{

static void Main(string[] args)

{

int n;

Console.WriteLine("Enter a number:"); n=Convert.ToInt32(Console.ReadLine());

if ( n % 2 == 0)

{

Console.WriteLine("it is a even number");

}

else

{

Console.WriteLine("it is a odd number");

}

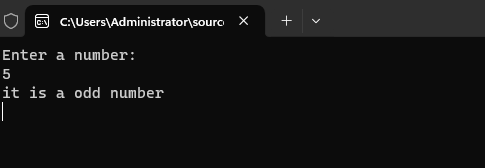
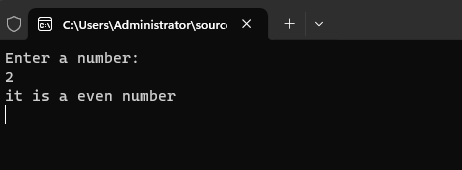
Console.ReadKey();

}

}

}

### Output:



using System;

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

using System.Text;

using System.Threading.Tasks;

namespace practical1

{

internal class Program

{

static void Main(string[] args)

{

int n;

int x = 0; int f = 0;

Console.WriteLine("Enter a number:"); n=Convert.ToInt32(Console.ReadLine()); x = n / 2;

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

{

if (n % 1 == 0)

{

Console.WriteLine(n + "is not prime"); f = 1; break;

}

}

if (f == 0)

{

Console.WriteLine(n + " number is prime");

}

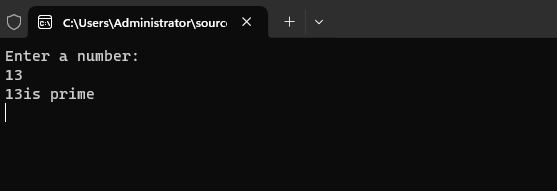
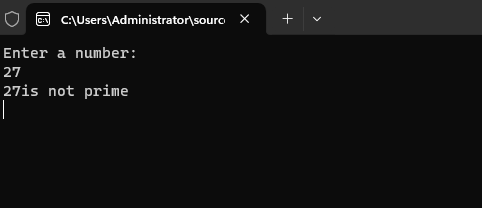
Console.ReadKey();

}

}

}

### Output:



using System; class Program

{

static void Main()

{

int n, r = 0, m;

Console.WriteLine("Mitali Pawar 23MCA35"); Console.WriteLine("Enter a number ");

n = Convert.ToInt32(Console.ReadLine()); while (n != 0)

{

m = n % 10;

r = r \* 10 + m; n /= 10;

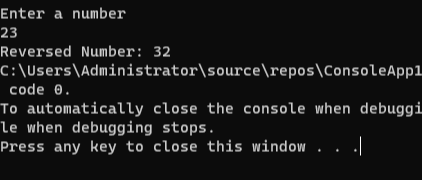
}

Console.Write("Reversed Number: " + r);

}

}

### Output:



* 1. write a program in c# to make a simple calculator for addition, subtraction, division, multiplication and modulus using GUI in c#.net

### Code:

namespace pract1b

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

int a = Convert.ToInt32(textBox1.Text); int b = Convert.ToInt32(textBox2.Text); int c = a % b;

label3.Text = c.ToString();

}

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

{

int a = Convert.ToInt32(textBox1.Text);

int b = Convert.ToInt32(textBox2.Text); int c = a + b;

label3.Text = c.ToString();

}

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

{

int a = Convert.ToInt32(textBox1.Text); int b = Convert.ToInt32(textBox2.Text); int c = a - b;

label3.Text = c.ToString();

}

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

{

int a = Convert.ToInt32(textBox1.Text); int b = Convert.ToInt32(textBox2.Text); int c = a \* b;

label3.Text = c.ToString();

}

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

{

int a = Convert.ToInt32(textBox1.Text); int b = Convert.ToInt32(textBox2.Text); int c = a / b;

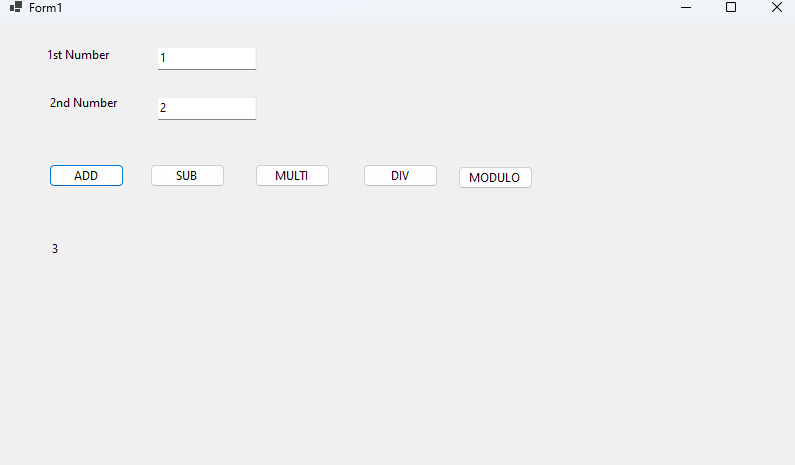
label3.Text = c.ToString();

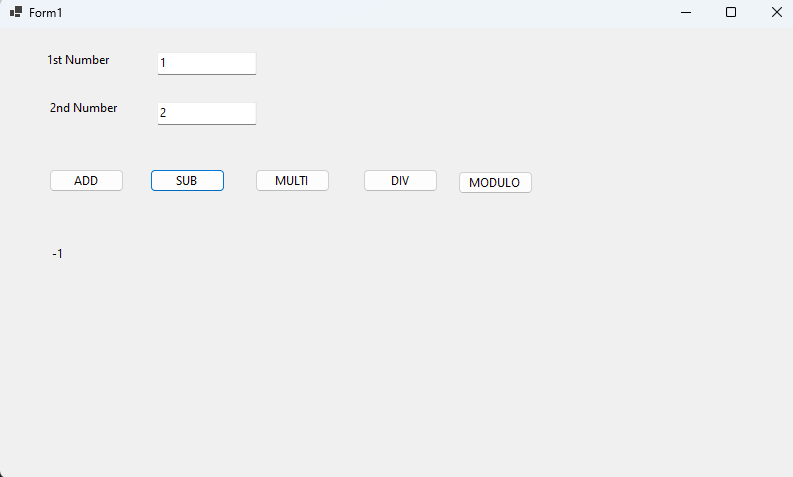
}

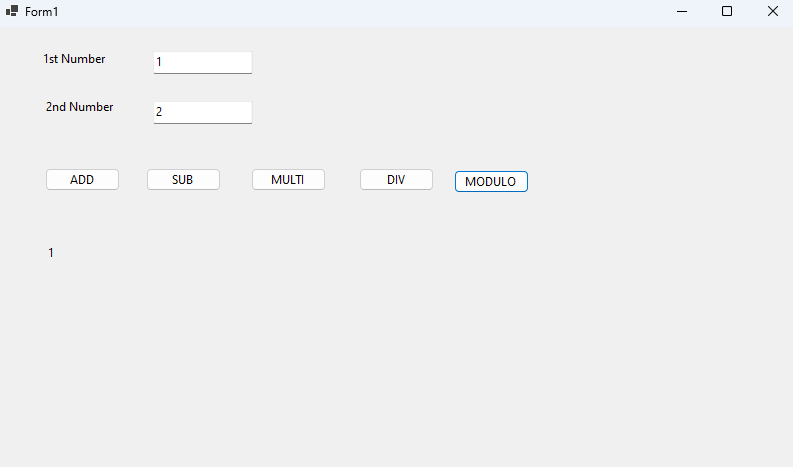
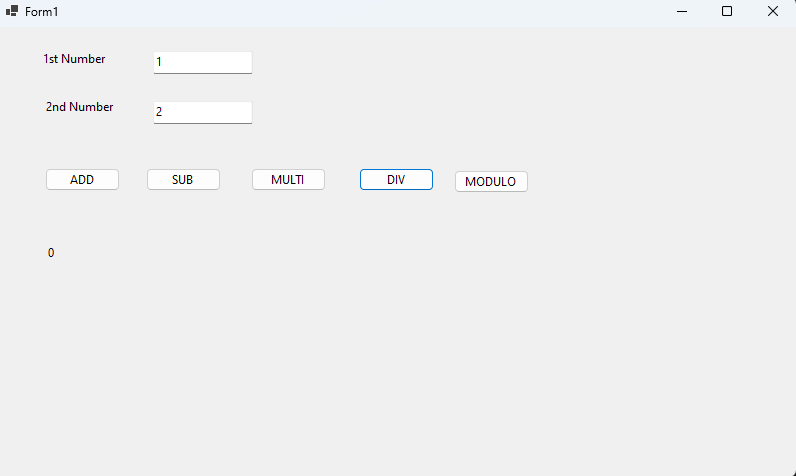
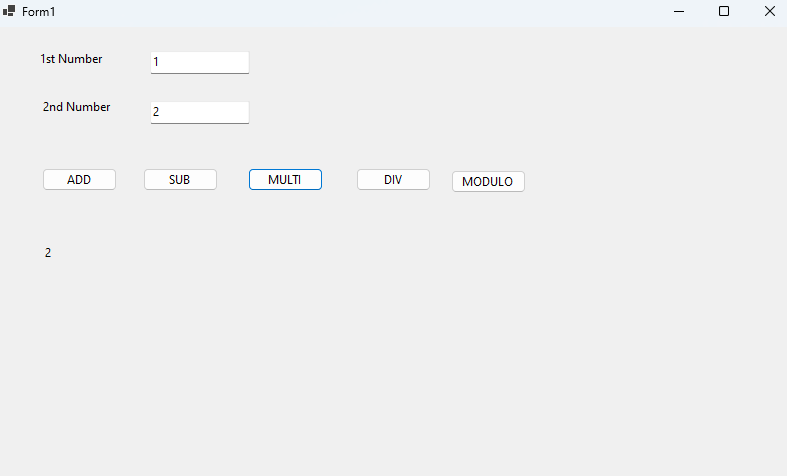
}

}

Output:







1. Design Application using classes and objects
   1. write a program in c#.net using classes and objects to print addition of two matrix take input from the user

Code:

using System;

using System.Collections.Generic; using System.Data;

using System.Linq; using System.Text;

using System.Threading.Tasks;

class Matrix

{

private int row; private int column; private int[,] data;

public Matrix(int row, int column)

{

this.row = row; this.column = column;

this.data = new int[row, column];

}

public void initialize()

{

Console.WriteLine($"Enter element for a {row}x{column} matrix:"); for (int i = 0; i < row; i++)

{

for (int j = 0; j < column; j++)

{

Console.Write($"Enter element at position ({i + 1},{j + 1}): "); if (int.TryParse(Console.ReadLine(), out int value))

{

data[i, j] = value;

}

else

{

Console.WriteLine("Invalid input. Please enter an integer."); j--;

}

}

}

}

public void Display()

{

Console.WriteLine("Matrix:"); for (int i = 0; i < row; i++)

{

for (int j = 0; j < column; j++)

{

Console.Write($"{data[i, j]} ");

}

Console.WriteLine();

}

}

public static Matrix Add(Matrix matrix1, Matrix matrix2)

{

if (matrix1.row != matrix2.row || matrix1.column != matrix2.column)

{

throw new InvalidOperationException("Matrices must have the same dimensions for addition.");

}

Matrix result = new Matrix(matrix1.row, matrix1.column); for (int i = 0; i < matrix1.row; i++)

{

for (int j = 0; j < matrix1.column; j++)

{

result.data[i, j] = matrix1.data[i, j] + matrix2.data[i, j];

}

}

return result;

}

}

Output:



class Program

{

static void Main()

{

Console.WriteLine("Enter dimensions for the first matrix:"); Console.Write("Rows: ");

int rows1 = Convert.ToInt32(Console.ReadLine());

Console.Write("Columns: ");

int columns1 = Convert.ToInt32(Console.ReadLine());

Matrix matrix1 = new Matrix(rows1, columns1); matrix1.initialize();

matrix1.Display();

Console.WriteLine("\nEnter dimensions for the second matrix:"); Console.Write("Rows: ");

int rows2 = Convert.ToInt32(Console.ReadLine());

Console.Write("Columns: ");

int columns2 = Convert.ToInt32(Console.ReadLine());

Matrix matrix2 = new Matrix(rows2, columns2); matrix2.initialize();

matrix2.Display();

try

{

Matrix resultMatrix = Matrix.Add(matrix1, matrix2); Console.WriteLine("\nResult after matrix addition:"); resultMatrix.Display();

}

catch (InvalidOperationException ex)

{

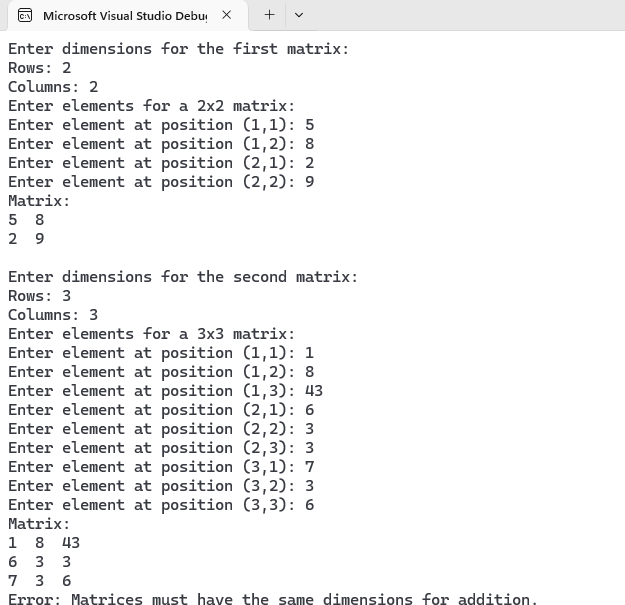
Console.WriteLine($"Error: {ex.Message}");

}

Console.ReadKey();

}

}



3: Design Application using Inheritance and abstract classes

a.Calculate percentage of Student from marks entered by user using c# .NET using Inheritance and Abstract classes .It should have three classes.

Parent class: student

Derived class: UNdergraduate (derived from student class) Derived Class: PostGraduate(derived from student class)

The postgraduate student has a bonus of 1.2%.using switch case statement perform the above program.

Code:

using System;

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

using System.Text;

using System.Threading.Tasks;

abstract class Student

{

protected string name; protected int[] marks;

public Student(string name, int[] marks)

{

this.name = name; this.marks = marks;

}

public abstract double CalculatePercentage();

public void DisplayDetails()

{

Console.WriteLine($"Student Name: {name}"); Console.WriteLine("Marks:");

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

{

Console.WriteLine($"Subject {i + 1}: {marks[i]}");

}

}

}

class Undergraduate : Student

{

public Undergraduate(string name, int[] marks) : base(name, marks)

{

}

public override double CalculatePercentage()

{

int totalMarks = 0;

foreach (int mark in marks)

{

totalMarks += mark;

}

return (double)totalMarks / marks.Length;

}

}

class Postgraduate : Student

{

public Postgraduate(string name, int[] marks) : base(name, marks)

{

}

public override double CalculatePercentage()

{

int totalMarks = 0;

foreach (int mark in marks)

{

totalMarks += mark;

}

return ((double)totalMarks / marks.Length) + 1.2;

}

}

class Program

{

static void Main()

{

Console.Write("Enter student name: "); string name = Console.ReadLine();

Console.Write("Enter the number of subjects: ");

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

int[] marks = new int[numSubjects]; for (int i = 0; i < numSubjects; i++)

{

Console.Write($"Enter marks for Subject {i + 1}: "); marks[i] = Convert.ToInt32(Console.ReadLine());

}

Console.WriteLine("\nSelect student type:"); Console.WriteLine("1. Undergraduate"); Console.WriteLine("2. Postgraduate");

int choice = Convert.ToInt32(Console.ReadLine()); Student student;

switch (choice)

{

case 1:

student = new Undergraduate(name, marks); break;

case 2:

student = new Postgraduate(name, marks); break;

default:

Console.WriteLine("Invalid choice. Defaulting to Undergraduate."); student = new Undergraduate(name, marks);

break;

}

Console.WriteLine("\nStudent Details:"); student.DisplayDetails();

double percentage = student.CalculatePercentage(); Console.WriteLine($"\nPercentage: {percentage}%");

Console.ReadKey();

}

}

Output:



**Practical No:- 2**

**Aim:** Design a Web Application for an Organization with Registration forms and advanced controls also Create website using master page concept.

2.1 Design Registration Form for volunteers in an NGO having following properties/attributes: v\_id, v\_name, v\_emailid, v\_phone, v\_address, v\_skills, v\_availability Display the attributes entered by the user on a different page

### Theory:

NET Framework consists of the common language runtime (CLR) and the .NET Framework class library. The common language runtime is the foundation of .NET Framework.

Think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and remoting, while also enforcing strict type safety and other forms of code accuracy that promote security and robustness.

In web development, mastering the fundamentals is essential, encompassing data types, variables, and operators for effective server-side and client-side scripting.

Server-side controls play a crucial role in user interaction, facilitating the implementation of functionalities like form submissions and event handling.

Concepts such as cross-page posting, postback, and autopostback are pivotal for seamless data transfer and dynamic updates within web applications.

Advanced web server controls augment application functionality, providing capabilities for input validation, date selection, navigation, and file management.

frameworks like AngularJS, jQuery, and Node.js expands development horizons, empowering developers to build dynamic and scalable applications with enhanced user interactivity and responsiveness.

The utilization of master pages in ASP.NET enables the creation of cohesive layouts and facilitates content management across multiple pages, streamlining development efforts and ensuring consistency in design.

### Code:

Detail.aspx- using System;

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

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls;

namespace pract2A

{

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

{

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

{

if(!IsPostBack)

{

Label1.Text = "Name:" + Session["Name"]; Label2.Text = "Email:" + Session["Email"]; Label3.Text = "Phone:" + Session["Phone"]; Label4.Text = "Address:" + Session["Address"]; Label5.Text = "Skills:" + Session["Skills"]; Label6.Text = "Availability:" + Session["Availability"];

}

}

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

{

}

}

}

Registration.aspx- using System;

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

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls;

namespace pract2A

{

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

{

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

{

}

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

{

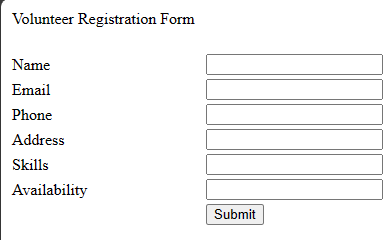
Session["Name"] = TextBox1.Text; Session["Email"] = TextBox2.Text; Session["Phone"] = TextBox3.Text; Session["Address"] = TextBox4.Text; Session["Skills"] = TextBox5.Text; Session["Availability"] = TextBox6.Text; Response.Redirect("Details.aspx");

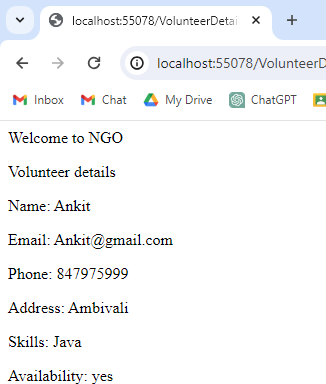
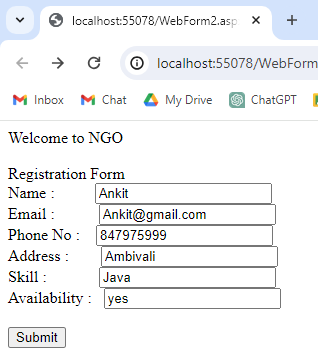
}

}

}

Output:





2.2. Create a simple website using ASP.NET Web Forms and the concept of a master page to accept student details and calculate/display the percentage of the student.

Code:

Webform 1

using System;

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

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls;

namespace Prac2B

{

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

{

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

{

}

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

{

try

{

int marks1 = Convert.ToInt32(TextBox1.Text); int marks2 = Convert.ToInt32(TextBox2.Text);

double total = marks1 + marks2; double perc = (total \* 100) / 200;

Label3.Text = "percentage:" + perc.ToString("0.00") + "%"; Label3.Visible = true;

}

catch

{

Label3.Text = "wrong data is entered";

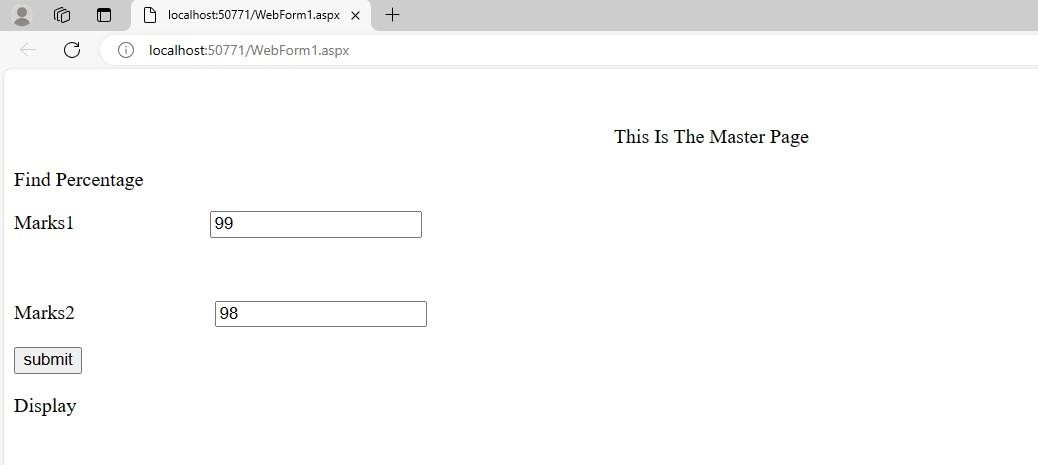
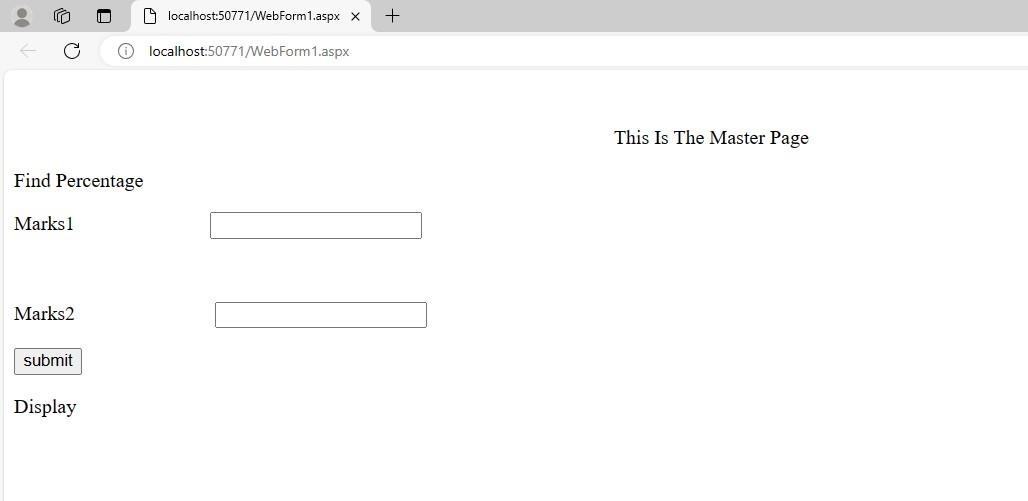
}

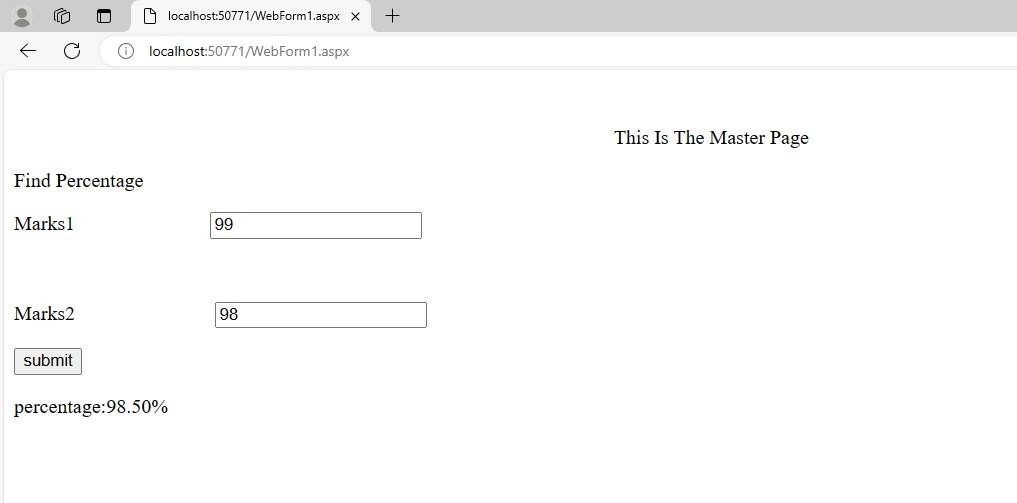
}

}

}

Output:





**Practical No:- 3**

**Aim:** Design a web-pages to demonstrate a connection-oriented architecture and Disconnected architecture. (Install-Package MySql.Data)

### Theory:

In software architecture, the choice between connection-oriented and disconnected architectures often hinges on how data is managed between an application and a database.

In a connection-oriented architecture, a continuous connection is established between the application and the database server throughout the data retrieval process.

This approach ensures real-time access to data but may lead to performance issues due to constant connection overhead.

disconnected architecture involves retrieving data from the database into a local data structure, such as a DataTable or DataSet, and then severing the connection to the database.

This allows for greater scalability and reduces the load on the database server, making it suitable for applications with heavy read operations or intermittent connectivity. it requires careful management of data synchronization to ensure consistency between the local data and the database.

Both architectures have their strengths and weaknesses, and the choice between them depends on factors such as application requirements, scalability needs, and network conditions.

By understanding the trade-offs between connection-oriented and disconnected architectures, developers can design systems that efficiently manage data while meeting performance and reliability goals.

### Code:

1. Create a student table in mysql having attributes: s\_id, s\_rollno s\_name, s\_email, s\_city,s\_country. Using connection-oriented architecture insert data into the table using ASP.Net and Display the Data.

studentData.aspx:

using System;

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

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls;

namespace practical\_3\_1

{

public partial class studentdata : System.Web.UI.flage

{

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

{

if (!IsflostBack)

{

BindGridView();

}

}

private void BindGridView()

{

DatabaseConnection dbConnection = new DatabaseConnection(); List<Student> students = dbConnection.GetStudents(); GridViewStudents.DataSource = students; GridViewStudents.DataBind();

}

}

}

Student.cs:

using System;

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

using System.Web;

namespace practical\_3\_1

{

public class Student

{

public int Id { get; set; } public string Roll o { get; set; } public string ame { get; set; } public string Email { get; set; } public string City { get; set; }

public string Country { get; set; }

}

}

Databaseconnection.cs using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq; using System.Web;

using MySql.Data.MySqlClient;

namespace practical\_3\_1

{

public class DatabaseConnection

{

public MySqlConnection GetConnection()

{

string connectionString = "server=localhost;database=studentdb;username=root;password=howareyou";

MySqlConnection connection = new MySqlConnection(connectionString); return connection;

}

public void InsertStudent(string roll o, string name, string email, string city, string country)

{

using (MySqlConnection connection = GetConnection())

{

connection.Open();

string query = "I SERT I TO student (s\_rollno, s\_name, s\_email, s\_city, s\_country) VALUES (@roll o, @name, @email, @city, @country)";

MySqlCommand command = new MySqlCommand(query, connection); command.flarameters.AddWithValue("@roll o", roll o); command.flarameters.AddWithValue("@name", name); command.flarameters.AddWithValue("@email", email); command.flarameters.AddWithValue("@city", city); command.flarameters.AddWithValue("@country", country); command.Execute onQuery();

}

}

public List<Student> GetStudents()

{

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

using (MySqlConnection connection = GetConnection())

{

connection.Open();

string query = "SELECT \* FROM student";

MySqlCommand command = new MySqlCommand(query, connection); using (MySqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

students.Add(new Student

{

});

}

}

}

Id = Convert.ToInt32(reader["s\_id"]), Roll o = reader["s\_rollno"].ToString(), ame = reader["s\_name"].ToString(), Email = reader["s\_email"].ToString(), City = reader["s\_city"].ToString(), Country = reader["s\_country"].ToString()

return students;

}

}

}

Studentdetails.aspx.cs:

using System;

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

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls;

namespace practical\_3\_1

{

public partial class studentdetails : System.Web.UI.flage

{

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

{

}

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

{

string roll o = txtRoll o.Text; string name = txt ame.Text; string email = txtEmail.Text; string city = txtCity.Text; string country = txtCountry.Text;

DatabaseConnection dbConnection = new DatabaseConnection(); dbConnection.InsertStudent(roll o, name, email, city, country);

lblMessage.Text = "Student added successfully!"; ClearFields();

}

private void ClearFields()

{

txtRoll o.Text = ""; txt ame.Text = ""; txtEmail.Text = ""; txtCity.Text = ""; txtCountry.Text = "";

}

}

}

Studentdetails.aspx:

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="studentdetails.aspx.cs" Inherits="practical\_3\_1.studentdetails" %>

<!DOCTYflE html>

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

<head runat="server">

<title>Add Student</title>

</head>

<body>

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

<div>

<h2>Add Student</h2>

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

<br />

<asp:TextBox ID="txtRoll o" runat="server" placeholder="Roll o"></asp:TextBox>

<br />

<asp:TextBox ID="txt ame" runat="server" placeholder=" ame"></asp:TextBox>

<br />

<asp:TextBox ID="txtEmail" runat="server" placeholder="Email"></asp:TextBox>

<br />

<asp:TextBox ID="txtCity" runat="server" placeholder="City"></asp:TextBox>

<br />

<asp:TextBox ID="txtCountry" runat="server" placeholder="Country"></asp:TextBox>

<br />

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

</div>

</form>

</body>

</html>

Index.aspx:

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="Index.aspx.cs" Inherits="practical\_3\_1.Index" %>

<!DOCTYflE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<h2>This is the Main flage: Use the links below to add or view

data</h2>

<p>

<asp:HyperLink ID="add\_student" runat="server"

avigateUrl="~/studentdetails.aspx">Add Student</asp:HyperLink> finbsp;<asp:HyperLink ID="view\_data" runat="server" avigateUrl="~/studentdata.aspx">View Student Data</asp:HyperLink>

finbsp;finbsp;

<asp:HyperLink ID="HyperLink1" runat="server" avigateUrl="~/Index.aspx">home</asp:HyperLink>

</p>

</div>

</form>

</body>

</html>

Studentdata.aspx:

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="studentdata.aspx.cs" Inherits="practical\_3\_1.studentdata" %>

<!DOCTYflE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<h2>Student Data</h2>

<asp:GridView ID="GridViewStudents" runat="server" AutoGenerateColumns="false">

<Columns>

<asp:BoundField DataField="Roll o" HeaderText="Roll o" />

<asp:BoundField DataField=" ame" HeaderText=" ame" />

<asp:BoundField DataField="Email" HeaderText="Email" />

<asp:BoundField DataField="City" HeaderText="City" />

<asp:BoundField DataField="Country" HeaderText="Country" />

</Columns>

</asp:GridView>

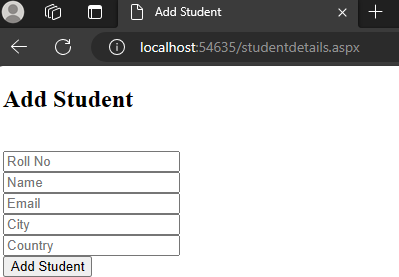
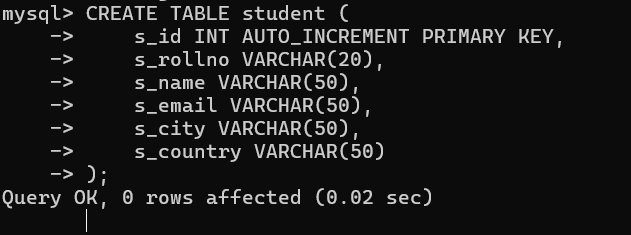
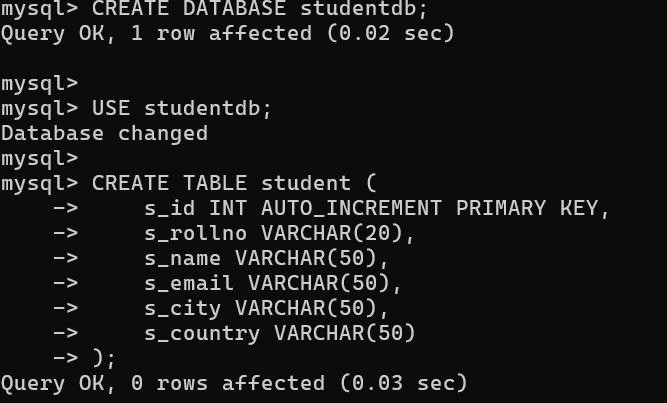
</div>

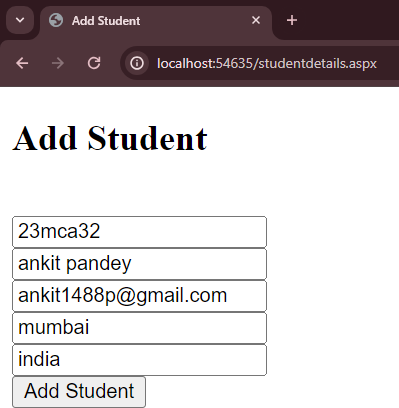
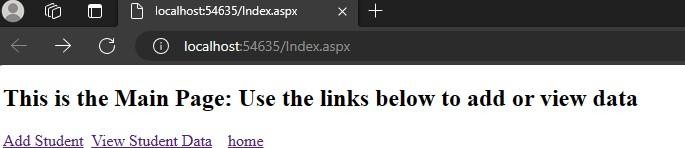
</form>

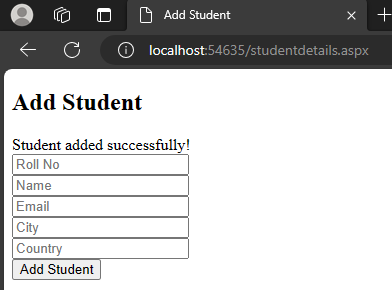
</body>

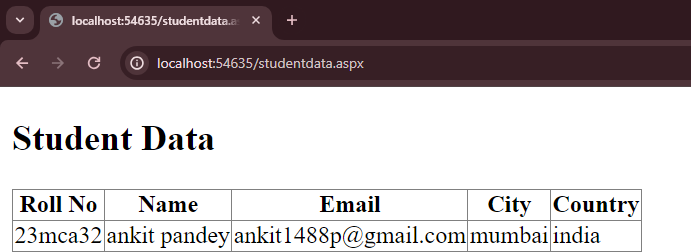
</html>

Output:









1. Create a customer table in mysql having attributes: c\_id, c\_name, c\_email, c\_city,c\_country,c\_pincode. Using Disconnected architecture insert data into the table using ASP.Net and Display the Data which user has inserted.

### Code:

DatabaseConnection.cs:

using System;

using System.Collections.Generic; using System.Data.SqlClient; using System.Data;

using System.Linq; using System.Web;

using MySql.Data.MySqlClient;

namespace practical\_3\_2

{

public class DatabaseConnection

{

private string connectionString = "server=localhost;database=customerdb;username=root;password=howareyou";

public DataTable GetCustomers()

{

DataTable dt = new DataTable();

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

string query = "SELECT c\_name, c\_email, c\_city, c\_country, c\_pincode FROM customer";

MySqlCommand command = new MySqlCommand(query, connection); MySqlDataAdapter adapter = new MySqlDataAdapter(command);

adapter.Fill(dt);

}

return dt;

}

public void InsertCustomer(string name, string email, string city, string country, string pincode)

{

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

string query = "I SERT I TO customer (c\_name, c\_email, c\_city, c\_country, c\_pincode) VALUES (@name, @email, @city, @country, @pin)";

MySqlCommand command = new MySqlCommand(query, connection); command.flarameters.AddWithValue("@name", name); command.flarameters.AddWithValue("@email", email); command.flarameters.AddWithValue("@city", city); command.flarameters.AddWithValue("@country", country); command.flarameters.AddWithValue("@pin", pincode);

connection.Open(); command.Execute onQuery();

}

}

}

}

customerDetails.aspx:

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="CustomerDetail.aspx.cs" Inherits="practical\_3\_2.CustomerDetail" %>

<!DOCTYflE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<h2>Kindly Fill the form to Add your details</h2>

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

<br />

<asp:TextBox ID="txt ame" runat="server" placeholder=" ame"></asp:TextBox>

<br />

<asp:TextBox ID="txtEmail" runat="server" placeholder="Email"></asp:TextBox>

<br />

<asp:TextBox ID="txtCity" runat="server" placeholder="City"></asp:TextBox>

<br />

<asp:TextBox ID="txtCountry" runat="server" placeholder="Country"></asp:TextBox>

<br />

<asp:TextBox ID="txtflincode" runat="server" placeholder="flincode"></asp:TextBox>

<br />

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

</div>

</div>

</form>

</body>

</html>

customerDetails.aspx.cs:

using System;

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

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls;

namespace practical\_3\_2

{

public partial class CustomerDetail : System.Web.UI.flage

{

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

{

}

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

{

string name = txt ame.Text; string email = txtEmail.Text; string city = txtCity.Text; string country = txtCountry.Text; string pincode = txtflincode.Text;

DatabaseConnection dbConnection = new DatabaseConnection(); dbConnection.InsertCustomer(name, email, city, country, pincode);

lblMessage.Text = "Customer added successfully!";

ClearFields();

}

private void ClearFields()

{

txt ame.Text = ""; txtEmail.Text = ""; txtCity.Text = ""; txtCountry.Text = ""; txtflincode.Text = "";

}

}

}

DisplayCustomer.aspx.cs:

using System;

using System.Collections.Generic; using System.Data;

using System.Linq; using System.Web; using System.Web.UI;

using System.Web.UI.WebControls;

namespace practical\_3\_2

{

public partial class DisplayCustomer : System.Web.UI.flage

{

DatabaseConnection dbConnection = new DatabaseConnection(); DataTable dt;

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

{

dt = dbConnection.GetCustomers(); if (!IsflostBack)

{

BindGridView();

}

}

protected void BindGridView()

{

GridViewCustomers.DataSource = dt; GridViewCustomers.DataBind();

}

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

{

DataView dv = new DataView(dt); if (city\_list.SelectedIndex == 0)

{

dv.RowFilter = "c\_city = 'Mumbai'";

}

if (city\_list.SelectedIndex == 1)

{

dv.RowFilter = "c\_city = 'flune'";

}

GridViewCustomers.DataSource = dv; GridViewCustomers.DataBind();

}

}

}

Index.aspx:

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="Index.aspx.cs" Inherits="practical\_3\_2.Index" %>

<!DOCTYflE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

This is the main page: Choose the operation you want to perform:<br />

<asp:HyperLink ID="addcustomer" runat="server" avigateUrl="~/CustomerDetail.aspx">Add Customer</asp:HyperLink> finbsp;

<asp:HyperLink ID="displaycustomer" runat="server" avigateUrl="~/DisplayCustomer.aspx">View Customer</asp:HyperLink>

finbsp;<asp:HyperLink ID="HyperLink1" runat="server" avigateUrl="~/customer\_mumbai.aspx">Mumbai Customers</asp:HyperLink>

</form>

</body>

</html>

DisplayCustomer.aspx:

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="DisplayCustomer.aspx.cs" Inherits="practical\_3\_2.DisplayCustomer" %>

<!DOCTYflE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<h2>The following are the customers available</h2>

<asp:GridView ID="GridViewCustomers" runat="server" AutoGenerateColumns="false">

<Columns>

<asp:BoundField DataField="c\_name" HeaderText=" ame" />

<asp:BoundField DataField="c\_email" HeaderText="Email" />

<asp:BoundField DataField="c\_city" HeaderText="City" />

<asp:BoundField DataField="c\_country" HeaderText="Country" />

<asp:BoundField DataField="c\_pincode" HeaderText="flincode" />

</Columns>

</asp:GridView>

</div>

<asp:DropDownList ID="city\_list" runat="server">

<asp:ListItem Value="mumbai"></asp:ListItem>

<asp:ListItem>pune</asp:ListItem>

</asp:DropDownList>

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

</form>

</body>

</html>

Customer\_mumbai.aspx:

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="customer\_mumbai.aspx.cs" Inherits="practical\_3\_2.customer\_mumbai" %>

<!DOCTYflE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<div>

<h2>Customer Data</h2>

<asp:GridView ID="GridViewCustomers" runat="server" AutoGenerateColumns="false">

<Columns>

<asp:BoundField DataField="c\_name" HeaderText=" ame" />

<asp:BoundField DataField="c\_email" HeaderText="Email" />

<asp:BoundField DataField="c\_city" HeaderText="City" />

<asp:BoundField DataField="c\_country" HeaderText="Country" />

<asp:BoundField DataField="c\_pincode" HeaderText="flincode" />

</Columns>

</asp:GridView>

</div>

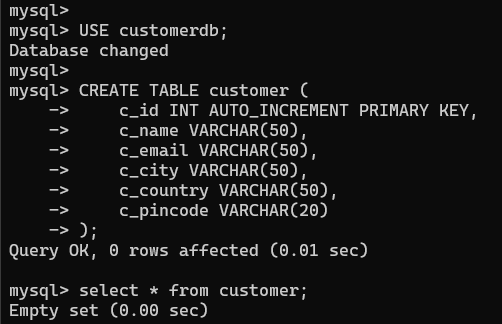
</div>

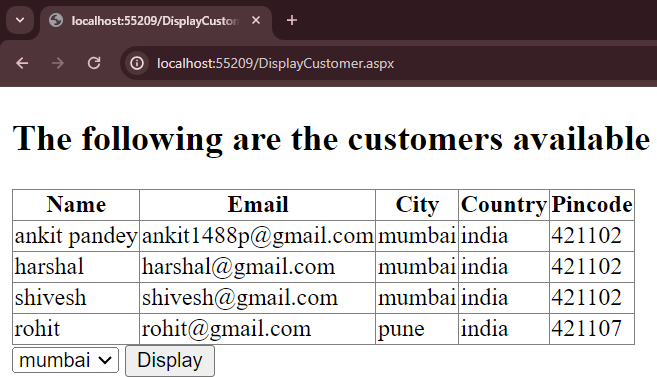
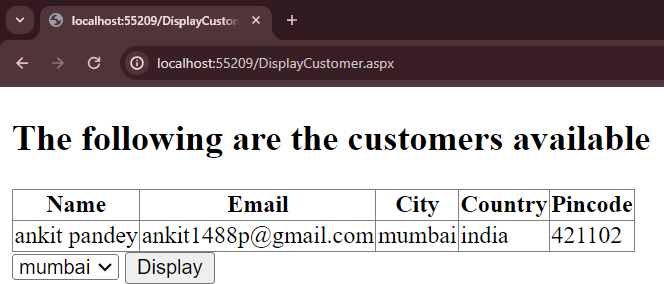
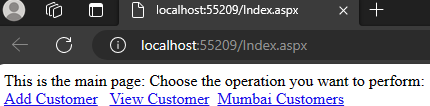
</form>

</body>

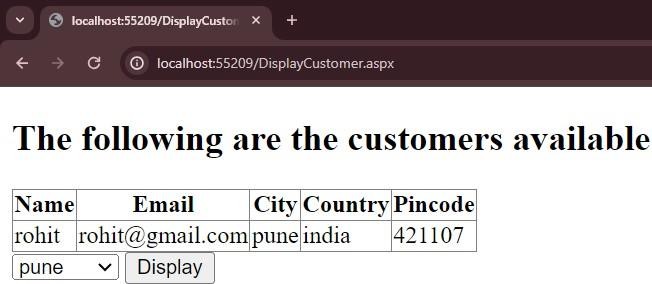
</html>

**Output:**





## V



**Practical No:- 4**

**Aim:** Create a webpage that demonstrates the use of data bound controls of ASP.NET.

### Theory:

web development, effective session management and AJAX (Asynchronous JavaScript and XML) integration are pivotal for delivering dynamic and responsive user experiences.

Client-side state management techniques such as View State, Query String, Cookies, and Hidden Fields play a crucial role in persisting user data across multiple requests, ensuring continuity and personalization.

These mechanisms empower developers to maintain stateful interactions with users, facilitating smoother navigation and data persistence within web applications.

various state management techniques such as Profiles, Session State, Application State, and caching mechanisms offer robust solutions for storing and retrieving application data.

Profiles enable personalized user experiences by storing user-specific information, while Session State preserves session-specific data across multiple requests.

Application State and caching mechanisms optimize application performance by storing frequently accessed data and resources, reducing database queries and enhancing scalability. ASP.NET applications achieve efficient data management and improved responsiveness.

AJAX into ASP.NET applications elevates user interactivity and responsiveness by enabling asynchronous communication between the client and server.

AJAX Controls provide pre-built components for seamless integration of AJAX functionalities, enhancing user experience without requiring full page reloads.

### Code:

1. Develop a web application that displays employee data retrieved from a MySQL database using various data-bound controls in ASP.NET. The application should include functionalities to view employee data in a GridView, a Repeater, a ListView, and a FormView. Additionally, users should be able to filter employees by department using a DropDownList, and select multiple employees using a ListBox. make sure you already have data in the employee table in mysql.

**Employee.aspx:-**

using System;

using System.Collections.Generic; using System.Configuration; using MySql.Data.MySqlClient; using System.Linq;

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls;

namespace Practical\_4

{

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

{

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

{

if (!IsPostBack)

{

BindGridView(); BindRepeater(); BindListView(); BindFormView(); BindDropDownList(); BindListBox();

}

}

private void BindGridView()

{

string connectionString = ConfigurationManager.ConnectionStrings["constring"].ConnectionString;

string department = ddlDepartments.SelectedValue;

// Construct the SQL query with a WHERE clause to filter by department string query = "SELECT \* FROM Employees";

if (!string.IsNullOrEmpty(department))

{

query += " WHERE Department = @Department";

}

using (MySqlConnection con = new MySqlConnection(connectionString))

{

MySqlCommand cmd = new MySqlCommand(query, con); if (!string.IsNullOrEmpty(department))

{

cmd.Parameters.AddWithValue("@Department", department);

}

con.Open();

gvEmployees.DataSource = cmd.ExecuteReader(); gvEmployees.DataBind();

}

}

private void BindRepeater()

{

string connectionString = ConfigurationManager.ConnectionStrings["constring"].ConnectionString;

string query = "SELECT \* FROM Employees"; string department = ddlDepartments.SelectedValue; if (!string.IsNullOrEmpty(department))

{

query += " WHERE Department = @department";

}

using (MySqlConnection con = new MySqlConnection(connectionString))

{

MySqlCommand cmd = new MySqlCommand(query, con); if (!string.IsNullOrEmpty(department))

{

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

}

con.Open();

rptEmployees.DataSource = cmd.ExecuteReader(); rptEmployees.DataBind();

}

}

private void BindListView()

{

string connectionString = ConfigurationManager.ConnectionStrings["constring"].ConnectionString;

string query = "SELECT \* FROM Employees"; string department = ddlDepartments.SelectedValue; if (!string.IsNullOrEmpty(department))

{

query += " WHERE Department = @department";

}

using (MySqlConnection con = new MySqlConnection(connectionString))

{

MySqlCommand cmd = new MySqlCommand(query, con); if (!string.IsNullOrEmpty(department))

{

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

}

con.Open();

lvEmployees.DataSource = cmd.ExecuteReader(); lvEmployees.DataBind();

}

}

private void BindFormView()

{

string connectionString = ConfigurationManager.ConnectionStrings["constring"].ConnectionString;

string query = "SELECT \* FROM Employees"; string department = ddlDepartments.SelectedValue; if (!string.IsNullOrEmpty(department))

{

query += " WHERE Department = @department";

}

using (MySqlConnection con = new MySqlConnection(connectionString))

{

MySqlCommand cmd = new MySqlCommand(query, con); if (!string.IsNullOrEmpty(department))

{

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

}

con.Open();

fvEmployee.DataSource = cmd.ExecuteReader(); fvEmployee.DataBind();

}

}

private void BindDropDownList()

{

string connectionString = ConfigurationManager.ConnectionStrings["constring"].ConnectionString;

using (MySqlConnection con = new MySqlConnection(connectionString))

{

MySqlCommand cmd = new MySqlCommand("SELECT DISTINCT department FROM Employees", con);

con.Open();

MySqlDataReader reader = cmd.ExecuteReader(); ddlDepartments.DataSource = reader; ddlDepartments.DataTextField = "Department"; ddlDepartments.DataValueField = "Department"; ddlDepartments.DataBind();

ddlDepartments.Items.Insert(0, new ListItem("-- Select Department --", ""));

}

}

private void BindListBox()

{

string connectionString = ConfigurationManager.ConnectionStrings["constring"].ConnectionString;

string query = "SELECT \* FROM Employees"; string department = ddlDepartments.SelectedValue; if (!string.IsNullOrEmpty(department))

{

query += " WHERE Department = @department";

}

using (MySqlConnection con = new MySqlConnection(connectionString))

{

MySqlCommand cmd = new MySqlCommand(query, con); if (!string.IsNullOrEmpty(department))

{

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

}

con.Open();

MySqlDataReader reader = cmd.ExecuteReader(); lbEmployees.DataSource = reader; lbEmployees.DataTextField = "f\_name"; lbEmployees.DataValueField = "emp\_id"; lbEmployees.DataBind();

}

}

protected void ddlDepartments\_SelectedIndexChanged(object sender, EventArgs e)

{

lblEmployeesInDepartment.Text = "Employee in " + ddlDepartments.SelectedValue; BindGridView();

BindRepeater(); BindListView(); BindListBox();

}

}

}

Web.config:-

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

<!--

For more information on how to configure your ASP.NET application, please visit https://go.microsoft.com/fwlink/?LinkId=169433

-->

<configuration>

<connectionStrings>

<add name="constring" connectionString="Server=localhost;Database=myemp;Uid=root;Pwd=root;" providerName="MySql.Data.MySqlClient"/>

</connectionStrings>

<system.web>

<compilation debug="true" targetFramework="4.8"/>

<httpRuntime targetFramework="4.8"/>

</system.web>

<system.codedom>

<compilers>

<compiler language="c#;cs;csharp" extension=".cs" type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider, Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.1.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" warningLevel="4" compilerOptions="/langversion:default /nowarn:1659;1699;1701"/>

<compiler language="vb;vbs;visualbasic;vbscript" extension=".vb" type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider, Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.1.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" warningLevel="4" compilerOptions="/langversion:default /nowarn:41008 /define:\_MYTYPE=\&quot;Web\&quot;

/optionInfer+"/>

</compilers>

</system.codedom>

<runtime>

<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">

<dependentAssembly>

<assemblyIdentity name="System.Runtime.CompilerServices.Unsafe" publicKeyToken="B03F5F7F11D50A3A" culture="neutral"/>

<bindingRedirect oldVersion="0.0.0.0-6.0.0.0"

newVersion="6.0.0.0"/>

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="System.Memory" publicKeyToken="CC7B13FFCD2DDD51" culture="neutral"/>

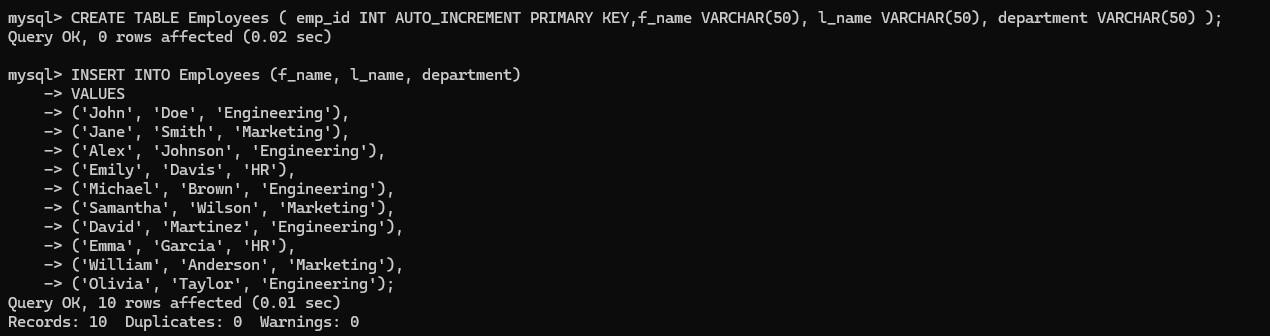
<bindingRedirect oldVersion="0.0.0.0-4.0.1.2"

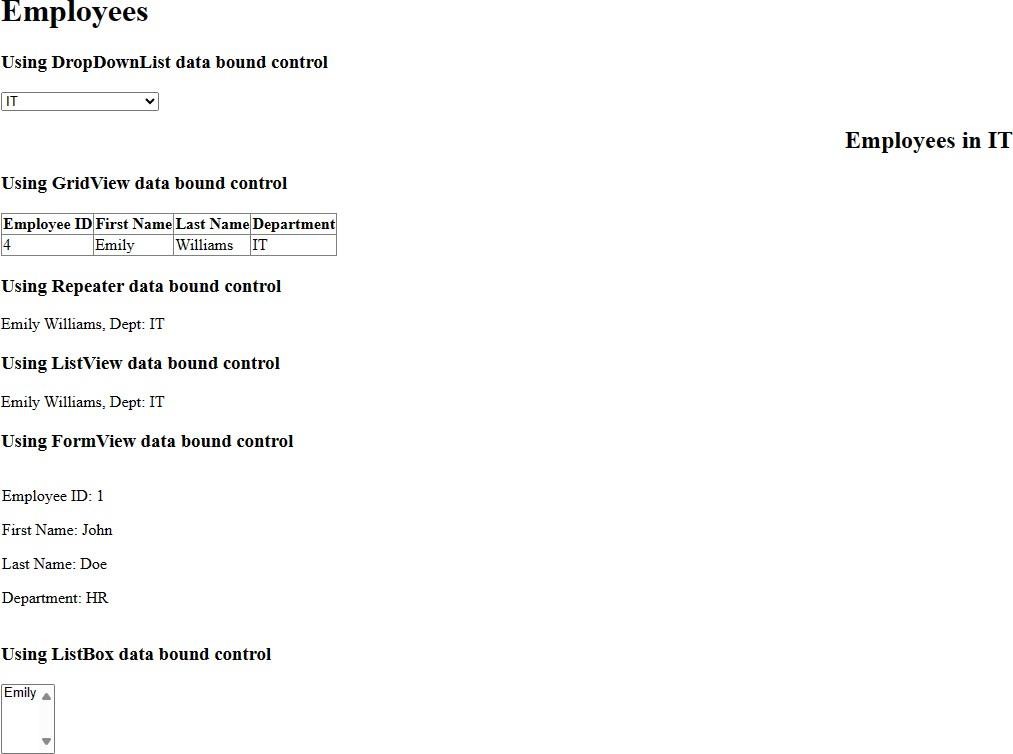
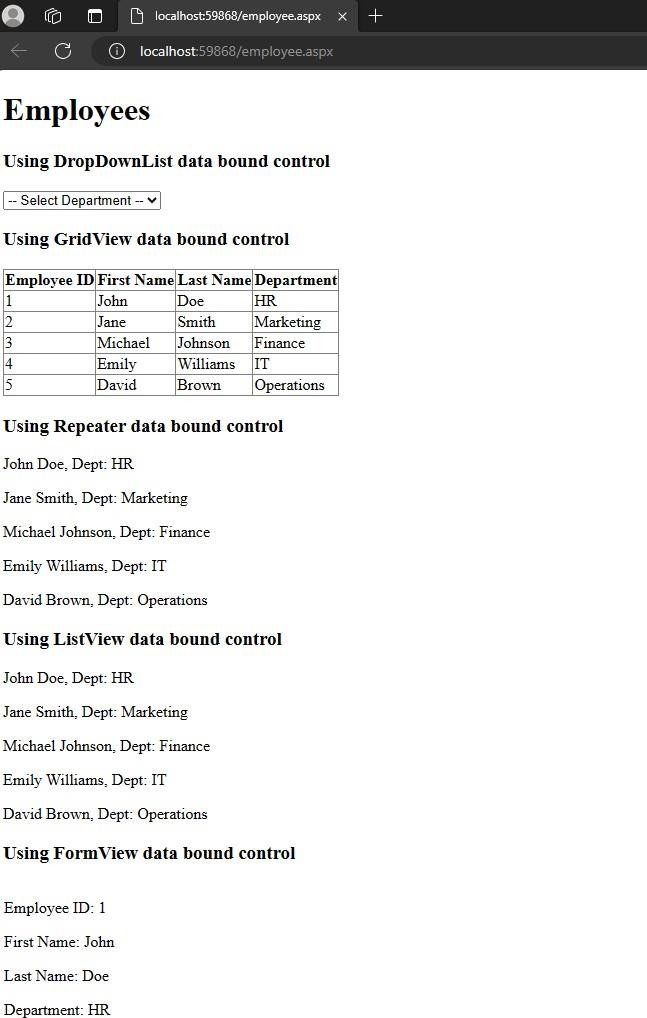
newVersion="4.0.1.2"/>

</dependentAssembly></assemblyBinding></runtime>

</configuration>

### Output:





1. Develop a web application using ASP.NET and MySQL to managecustomer orders and product inventory. Design a user-friendly interface allowing users to view customer details and product information. Utilize data-bound controls such as GridView, Repeater, and DropDownList to display data dynamically. Implement filtering functionalities to filter customers by country and products by category. Ensure data retrieval from MySQL database tables "Customers" and "Products" and provide sample data for testing.

### Code:

using System;

using System.Collections.Generic; using System.Configuration; using System.Data.SqlClient; using System.Data;

using System.Linq; using System.Web; using System.Web.UI;

using System.Web.UI.WebControls; using MySql.Data.MySqlClient;

namespace FinalPract4

{

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

{

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

{

if (!IsPostBack)

{

BindCustomers(); BindProducts();

}

}

protected void rptProducts\_ItemCommand(object source, RepeaterCommandEventArgs e)

{

}

protected void ddlCountries\_SelectedIndexChanged(object sender, EventArgs e)

{

BindCustomers();

}

protected void ddlCategories\_SelectedIndexChanged(object sender, EventArgs e)

{

BindProducts();

}

private void BindCustomers()

{

string constr = ConfigurationManager.ConnectionStrings["MySqlConnection"].ConnectionString;

using (MySqlConnection con = new MySqlConnection(constr))

{

using (MySqlCommand cmd = new MySqlCommand("SELECT \* FROM Customers" + (ddlCountries.SelectedValue != "" ? " WHERE Country=@Country" : ""), con))

{

if (ddlCountries.SelectedValue != "") cmd.Parameters.AddWithValue("@Country", ddlCountries.SelectedValue);

using (MySqlDataAdapter sda = new MySqlDataAdapter(cmd))

{

DataTable dt = new DataTable(); sda.Fill(dt); gvCustomers.DataSource = dt; gvCustomers.DataBind();

}

}

}

}

private void BindProducts()

{

string constr = ConfigurationManager.ConnectionStrings["MySqlConnection"].ConnectionString;

using (MySqlConnection con = new MySqlConnection(constr))

{

using (MySqlCommand cmd = new MySqlCommand("SELECT \* FROM Products"

+ (ddlCategories.SelectedValue != "" ? " WHERE Category=@Category" : ""), con))

{

if (ddlCategories.SelectedValue != "") cmd.Parameters.AddWithValue("@Category", ddlCategories.SelectedValue);

using (MySqlDataAdapter sda = new MySqlDataAdapter(cmd))

{

DataTable dt = new DataTable(); sda.Fill(dt); rptProducts.DataSource = dt; rptProducts.DataBind();

}

}

}

}

protected void gvCustomers\_SelectedIndexChanged(object sender, EventArgs e)

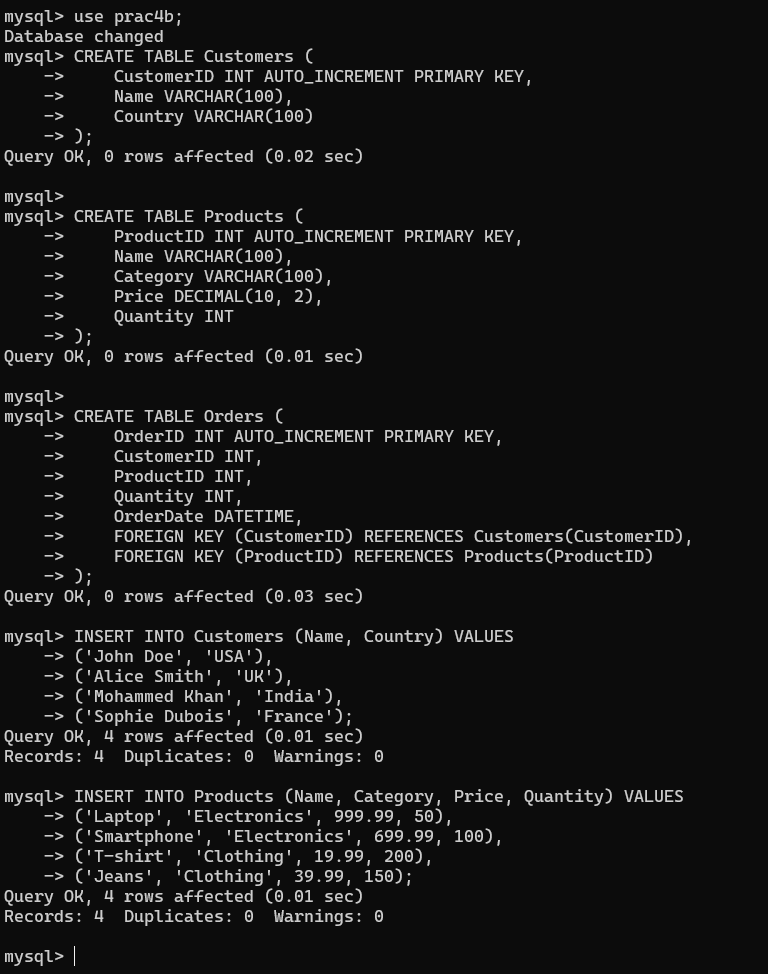
{

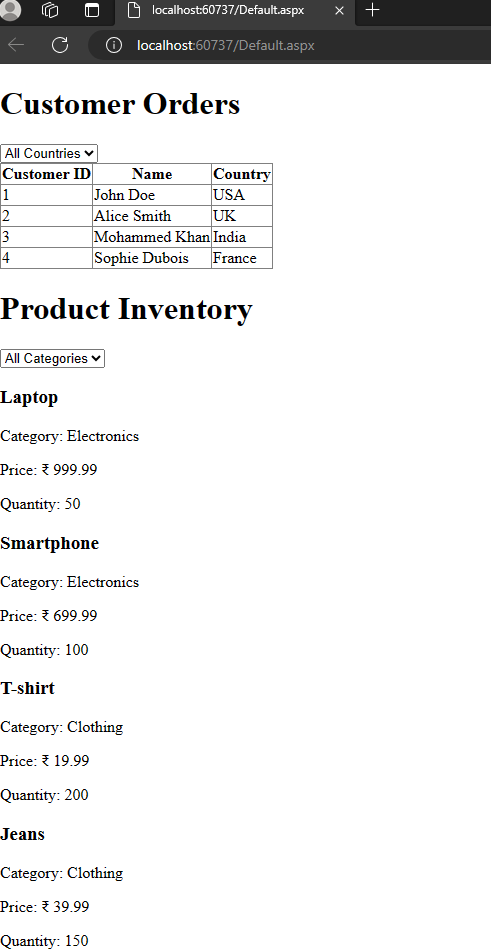
}

}

}

### Output:





**Practical No:- 5**

**Aim:** Design a webpage to demonstrate the working of a simple stored procedure and parameterized stored procedure.

### Theory:

Stored procedures are essential components of database-driven web applications, offering a means to encapsulate complex SQL logic within the database itself.

A simple stored procedure typically performs a specific database operation, such as retrieving data or updating records, and can be invoked directly from a webpage's backend code. This direct execution reduces network traffic and database load by minimizing the need for multiple SQL statements to be sent back and forth between the web server and the database server.

Parameterized stored procedures further enhance the capabilities of stored procedures by accepting input parameters. These parameters allow for dynamic customization of SQL operations based on user-provided values, such as search criteria or user preferences. By parameterizing stored procedures, developers can achieve greater flexibility in their database interactions while also improving security.

Designing a webpage to demonstrate both simple and parameterized stored procedures provides a practical illustration of their utility in real-world scenarios. The webpage can feature interactive elements where users can input values or select options, triggering the execution of the corresponding stored procedures.

The results returned by the stored procedures can then be displayed on the webpage, showcasing how stored procedures streamline database interactions and empower developers to build efficient, secure, and scalable web applications.

This demonstration reinforces the importance of leveraging stored procedures in database- driven web development and highlights best practices for maximizing performance, security, and flexibility in data management.

### Code:

1. **Consider a scenario where you are developing a web application using ASP.NET and MySQL.**
   * Use a stored procedure to retrieve data from MYSQL.
   * You have a table named in your MySQL database with , and

Users

|  |  |
| --- | --- |
| columns | Username |
| GetUsers |  |

Email. Initially, you have to create a stored procedure named to retrieve the list of all users from the table.

Users

* + This procedure is used in ASP.Net page to Display all data on browser

Prac5.aspx.cs using System; using System.Data;

using MySql.Data.MySqlClient;

namespace practical\_5

{

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

{

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

{

if (!IsPostBack)

{

BindGridView();

}

}

private void BindGridView()

{

string connectionString = "server=localhost;database=awt;uid=root;password=root;"; using (MySqlConnection connection = new MySqlConnection(connectionString))

{

using (MySqlCommand command = new MySqlCommand("GetUsers", connection))

{

command.CommandType = CommandType.StoredProcedure;

using (MySqlDataAdapter adapter = new MySqlDataAdapter(command))

{

DataTable dtUsers = new DataTable(); adapter.Fill(dtUsers); GridViewUsers.DataSource = dtUsers; GridViewUsers.DataBind();

}

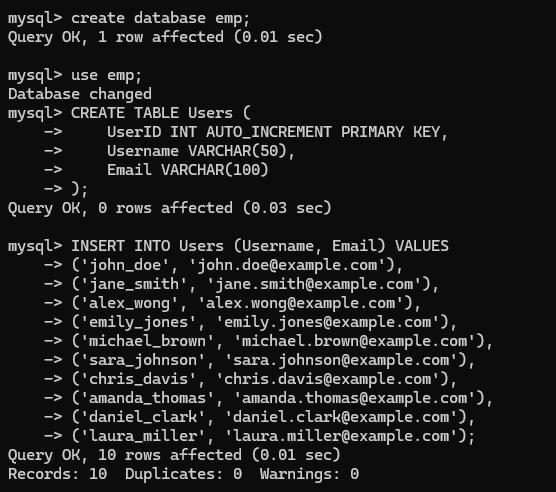
}

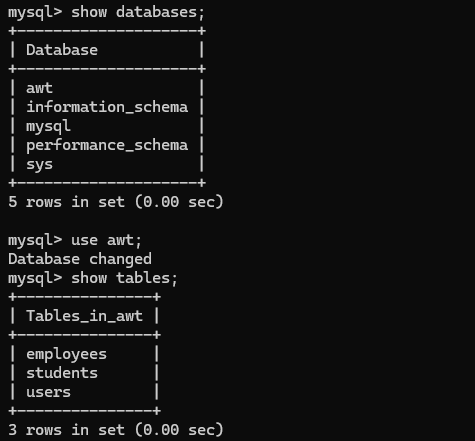
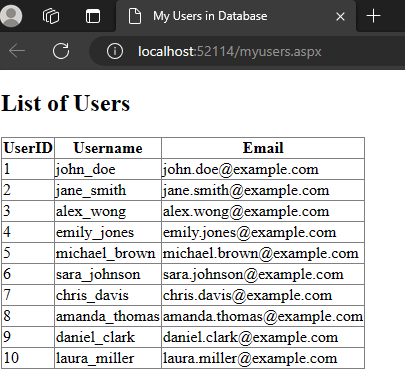
}

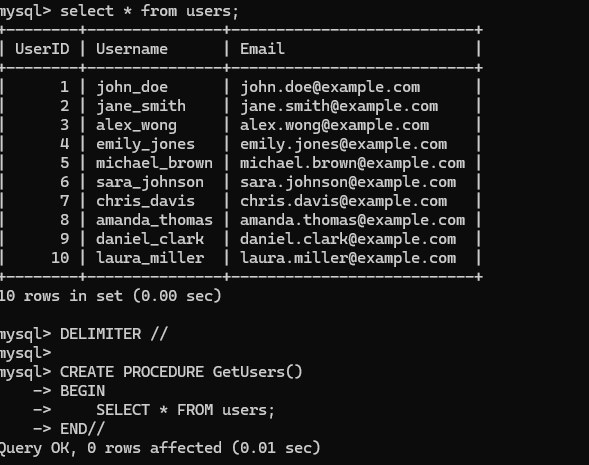
}

}

### Output:







1. **Consider a scenario where you are developing a web application using ASP.NET and MySQL. (self Study)**
   * Use a stored procedure to retrieve data from MYSQL.
   * You have a table named

mobile,city

Fname,Lname,

in your MySQL database with columns

. Initially, you have to create a stored procedure

Email

employees

named Getemp to retrieve the list of all employtees from the

Getempcity

Users

* + Then create one more stored procedure named

table.

to retrieve the list of all

employee full name and city from the table.

Users

* + This procedure is used in ASP.Net pages to Display all data on browser. Create separate pages for Getemp and Getempcity.

### Code:

**myusers.aspx**

using System; using System.Data;

using MySql.Data.MySqlClient;

namespace practical\_5

{

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

{

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

{

if (!IsPostBack)

{

}

}

void GetStudentDetails(int studentId)

{

string connectionString = "server=localhost;database=db;uid=root;password=root;"; using (MySqlConnection connection = new MySqlConnection(connectionString))

{

using (MySqlCommand command = new MySqlCommand("Getempdetails", connection))

{

command.CommandType = CommandType.StoredProcedure; command.Parameters.AddWithValue("@E\_ID", studentId);

using (MySqlDataAdapter adapter = new MySqlDataAdapter(command))

{

DataTable dtStudent = new DataTable(); adapter.Fill(dtStudent);

if (dtStudent.Rows.Count == 0)

{

lbl\_err.Text = "No Such ID found";

}

GridViewEmp.DataSource = dtStudent; GridViewEmp.DataBind();

}

}

}

}

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

{

int studentId;

if (int.TryParse(txtStudentId.Text, out studentId))

{

GetStudentDetails(studentId);

}

else

{

lbl\_err.Text = "you have entered wrong ID";

}

}

}

}

**empDeatils.aspx:**

using MySql.Data.MySqlClient; using System;

using System.Collections.Generic; using System.Data;

using System.Linq; using System.Web; using System.Web.UI;

using System.Web.UI.WebControls;

namespace practical\_5

{

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

{

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

{

if (!IsPostBack)

{

GetStudentDetails();

}

}

void GetStudentDetails()

{

string connectionString = "server=localhost;database=db;uid=root;password=root;"; using (MySqlConnection connection = new MySqlConnection(connectionString))

{

using (MySqlCommand command = new MySqlCommand("Getempcity", connection))

{

command.CommandType = CommandType.StoredProcedure;

using (MySqlDataAdapter adapter = new MySqlDataAdapter(command))

{

DataTable dtStudent = new DataTable(); adapter.Fill(dtStudent);

GridViewEmp.DataSource = dtStudent; GridViewEmp.DataBind();

}

}

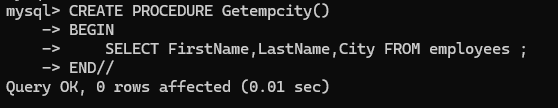
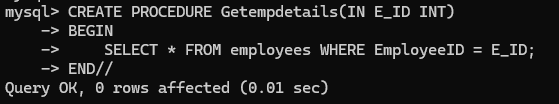
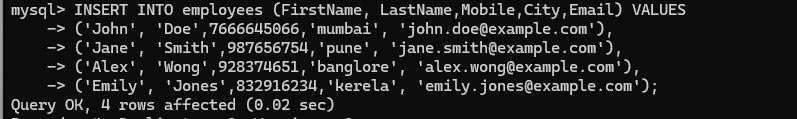
}

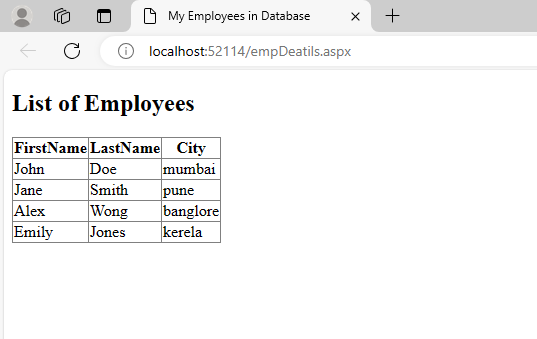
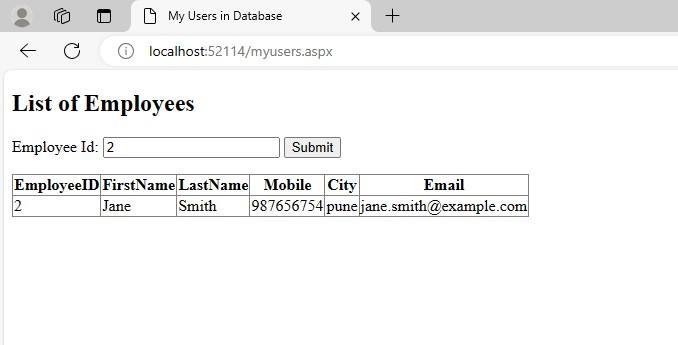
}

}

}

### Output:





**3 Consider a scenario where you are developing a web application using ASP.NET and MySQL.**

* Develop a web Page using ASP.NET and MySQL for managing student records. You have a table named Students in your MySQL database with columns StudentID, FirstName, LastName, and Email.
* implemented a parameterized stored procedure named GetStudentByID to retrieve the details of a student by their StudentID.

### Code:

using System;

using MySql.Data.MySqlClient;

using System.Data;

namespace practical\_5\_3

{

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

{

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

{

}

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

{

int studentId;

if (int.TryParse(txtStudentId.Text, out studentId))

{

GetStudentDetails(studentId);

}

else

{

lbl\_err.Text = "you have entered wrong ID";

}

}

private void GetStudentDetails(int studentId)

{

string connectionString = "server=localhost;database=awt;uid=root;password=root;";

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

connection))

using (MySqlCommand command = new MySqlCommand("GetStudentByID",

{

command.CommandType = CommandType.StoredProcedure; command.Parameters.AddWithValue("@S\_ID", studentId);

using (MySqlDataAdapter adapter = new MySqlDataAdapter(command))

{

DataTable dtStudent = new DataTable(); adapter.Fill(dtStudent);

if (dtStudent.Rows.Count==0)

{

lbl\_err.Text = "No Such ID found";

}

GridViewStudentDetails.DataSource = dtStudent; GridViewStudentDetails.DataBind();

}

}

}

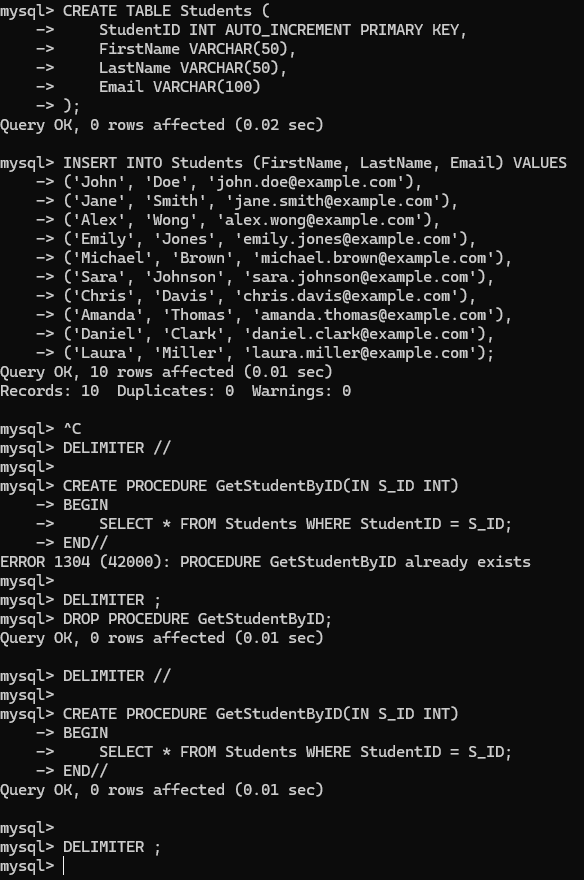
}

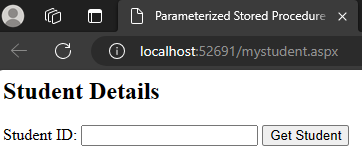
}

}

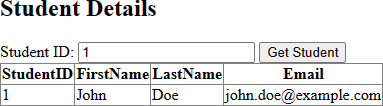
}

### Output:









**Practical No:- 6**

**Aim:** Design a webpage to display the use of LINQ.

### Theory:

LINQ (Language-Integrated Query) is a powerful feature in .NET that allows developers to query and manipulate data from different data sources using a unified syntax within C# or VB.NET code. LINQ provides a concise and expressive way to work with collections, databases, XML, and other data sources.

At its core, LINQ consists of a set of standard query operators that enable developers to perform various operations such as filtering, sorting, grouping, joining, and aggregating data.

These operators are accessible through LINQ methods or query syntax, providing flexibility in how developers construct and execute queries.

In a web development context, a webpage can demonstrate the use of LINQ by showcasing its capabilities in querying and manipulating data retrieved from a database, XML file, or any other data source.

webpage can feature a scenario where a collection of objects representing products is queried using LINQ to filter out products with specific criteria, sort them based on certain properties, or group them by category.

LINQ can be used to perform more complex operations such as joining multiple datasets or calculating aggregate values like sums, averages, or counts.

The webpage can display the results of these operations in a visually appealing format, such as a table or a chart, to illustrate the power and versatility of LINQ in data processing and presentation.

### Code:

1. Consider a scenario where you are developing a web application using ASP.NET and MySQL.

* Extract Data from data from a table named employees in your MySQL database with columns Fname,Lname,mobile,city Email into a gridview.
* Using this gridview extract data using LINQ to display only employees in Mumbai city

Webconfig file :

<connectionStrings>

<add name="constring" connectionString="Server=localhost;Database=awt;Uid=root;Pwd=root;" providerName="MySql.Data.MySqlClient" />

</connectionStrings>

webform1.aspx.cs:

using System;

using System.Collections.Generic; using System.Data.SqlClient; using System.Data;

using System.Linq; using System.Web; using System.Web.UI;

using System.Web.UI.WebControls; using System.Configuration;

using MySql.Data.MySqlClient;

namespace prac666

{

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

{

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

{

BindGridView(); ExtractAndBindFilteredData();

}

private void BindGridView()

{

string connectionString = ConfigurationManager.ConnectionStrings["constring"].ConnectionString;

string query = "SELECT Fname, Lname, mobile, city, email FROM employee";

using (MySqlConnection con = new MySqlConnection(connectionString))

{

MySqlCommand cmd = new MySqlCommand(query, con); con.Open();

DataTable dt = new DataTable(); dt.Load(cmd.ExecuteReader());

GridView1.DataSource = dt; GridView1.DataBind();

}

}

private void ExtractAndBindFilteredData()

{

// Get the data source of the source GridView

DataTable sourceDataTable = (DataTable)GridView1.DataSource;

// Optionally, perform LINQ operations on the source DataTable var filteredData = sourceDataTable.AsEnumerable()

.Where(row => row.Field<string>("City") == "New York")

.CopyToDataTable();

// Bind the filtered data to the destination GridView GridView2.DataSource = filteredData; GridView2.DataBind();

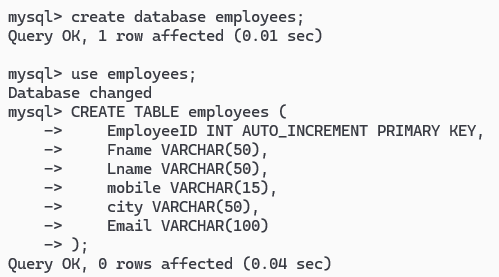
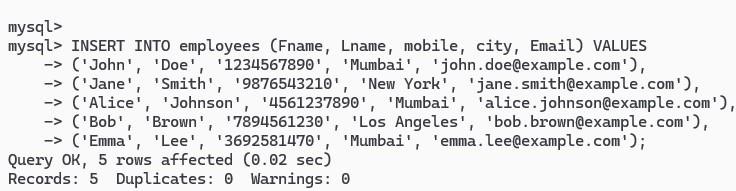
}

}

}

### Output:





1. Consider a scenario where you are developing a web application using ASP.NET and JSON

* Extract Data from data from a JSON String named Product having productname and type
* Extract data using LINQ to display only products of type Electronic Code:

using System;

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

using System.Web.UI.WebControls; using ewtonsoft.Json;

namespace prac6\_2

{

public partial class Yourflage : System.Web.UI.flage

{

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

{

if (!IsflostBack)

{

string jsonString = @"

{

},

""electronic"" },

},

""electronic"" }, electronic"" }

}";

""products"": [

{ ""product ame"": ""Laptop"", ""type"": ""electronic""

{ ""product ame"": ""Smartphone"", ""type"":

{ ""product ame"": ""Book"", ""type"": ""non-electronic""

{ ""product ame"": ""Headphones"", ""type"":

{ ""product ame"": ""Watch"", ""type"": ""non-

]

// Deserialize JSO string into a list of flroduct objects List<flroduct> products =

JsonConvert.DeserializeObject<flroductList>(jsonString).flroducts;

// Filter products of type "electronic"

var electronicflroducts = products.Where(p => p.Type == "electronic").ToList();

// Bind the filtered data to the GridViewflroducts control GridViewflroducts.DataSource = electronicflroducts; GridViewflroducts.DataBind();

}

}

// Define a class to represent a flroduct public class flroduct

{

public string flroduct ame { get; set; } public string Type { get; set; }

}

// Define a class to represent the JSO structure public class flroductList

{

public List<flroduct> flroducts { get; set; }

}

}

}

Web.aspx

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="Yourflage.aspx.cs" Inherits="prac6\_2.Yourflage" %>

<!DOCTYflE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<asp:GridView ID="GridViewflroducts" runat="server" AutoGenerateColumns="false">

<Columns>

<asp:BoundField DataField="flroduct ame" HeaderText="flroduct

ame" />

<asp:BoundField DataField="Type" HeaderText="Type" />

</Columns>

</asp:GridView>

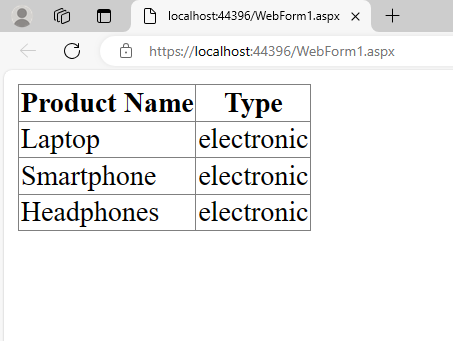
</div>

</form>

</body>

</html>

### Output:



**Practical No:- 7**

**Aim:** Design Web Applications using Client-Side Session Management and Server-Side Session Management Techniques

### Theory:

Web applications utilize session management techniques to maintain user state and data throughout their interactions. Client-side session management involves storing session data, usually in the form of cookies, directly on the user's browser.

These cookies contain session identifiers and other relevant information, allowing the server to identify and track the user's session across multiple page views. While client-side session management is easy to implement, it is susceptible to tampering and has limitations in terms of storage capacity.

server-side session management stores session data securely on the server, typically in memory or in a database.

This approach offers greater security and control over session data, as session information is not accessible to the client. Additionally, server-side management enables more robust session management features such as expiration times and cleanup mechanisms.

client-side session management provides simplicity and convenience, server-side session management offers better security and control over session data.

Depending on the specific requirements and security considerations of a web application, developers must choose the appropriate session management approach or even employ a combination of both for optimal performance and security.

### Code:

1. Create a login page using client-side cookies and maintain the session. Consider a dummy username and password use asp.net

## Login.aspx

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="login.aspx.cs" Inherits="practical\_no\_7\_1.login" %>

<!DOCTYflE html>

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

<head runat="server">

<title>Login flage</title>

</head>

<body>

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

<div>

<h2>Login</h2>

<asp:TextBox ID="txtUsername" runat="server"></asp:TextBox><br />

<asp:TextBox ID="txtflassword" runat="server" TextMode="flassword"></asp:TextBox><br />

<asp:Button ID="btnLogin" runat="server" Text="Login" OnClick="btnLogin\_Click" />

</div>

</form>

</body>

</html>

## Login.aspx.cs

using System;

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

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls;

namespace practical\_no\_7\_1

{

public partial class login : System.Web.UI.flage

{

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

{

}

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

{

string dummyUsername = "Sumeet"; string dummyflassword = "Sumeet123"; string username = txtUsername.Text; string password = txtflassword.Text;

if (username == dummyUsername fifi password == dummyflassword)

{

Response.Cookies["LoggedIn"].Value = "true"; Response.Cookies["username"].Value = txtUsername.Text; Response.Cookies["LoggedIn"].Expires = DateTime. ow.AddHours(1); Response.Redirect("dashboard.aspx");

}

else

{

Response.Write("<script>alert('Invalid username or password');</script>");

}

}

}

}

Dashboard.aspx

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="dashboard.aspx.cs" Inherits="practical\_no\_7\_1.dashboard" %>

<!DOCTYflE html>

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

<head runat="server">

<title>Dashboard</title>

</head>

<body>

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

<div>

<h2>Welcome to Dashboard</h2>

<div>

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

</div>

</div>

</form>

</body>

</html>

Dashboard.aspx.cs

using System;

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

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls;

namespace practical\_no\_7\_1

{

public partial class dashboard : System.Web.UI.flage

{

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

{

if (Request.Cookies["LoggedIn"] != null fifi Request.Cookies["LoggedIn"].Value == "true")

{

string username = Request.Cookies["username"].Value;

}

else

{

}

}

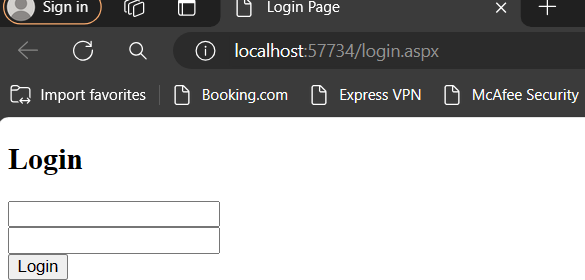
}

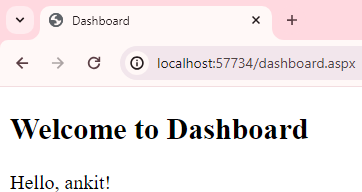
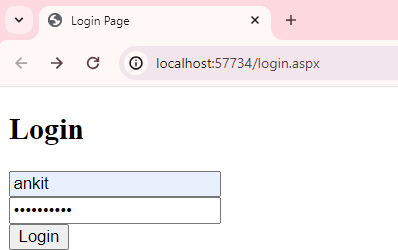
}

lblWelcomeMessage.Text = "Hello, " + username + "!";

Response.Redirect("login.aspx");

### Output:





1. Develop a web application for an online bookstore. The application needs to display a list of books from a database on the homepage (index.aspx). Users should be able to click on a book title to view its details on a separate page (bookdetails.aspx). Implement the necessary server- side code to fetch book data from the database and display it on the homepage. Also, create the book details page to display information about a selected book. using server side session management display the data retrieved from database

# Code:

## Index.aspx

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="index.aspx.cs" Inherits="practical\_7\_2.index" %>

<!DOCTYflE html>

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

<head runat="server">

<title>Bookstore - Home</title>

</head>

<body>

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

<h1>Welcome to Our Online Bookstore</h1>

<div>

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False">

<Columns>

<asp:BoundField DataField="BookID" HeaderText="Book ID"

/>

<asp:BoundField DataField="Title" HeaderText="Title" />

<asp:BoundField DataField="Author" HeaderText="Author" />

<asp:BoundField DataField="Genre" HeaderText="Genre" />

<asp:BoundField DataField="flrice" HeaderText="flrice" />

<asp:TemplateField HeaderText="Details">

<ItemTemplate>

<asp:HyperLink ID="lnkDetails" runat="server"

CssClass="btn btn-primary" Text="View Details" avigateUrl='<% "bookdetails.aspxoBookID=" + Eval("BookID") %>'></asp:HyperLink>

</ItemTemplate>

</asp:TemplateField>

</Columns>

</asp:GridView>

</div>

</form>

</body>

</html>

Index.aspx.cs using System;

using System.Collections.Generic; using MySql.Data.MySqlClient; using System.Linq;

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls; using System.Data;

namespace practical\_7\_2

{

public partial class index : System.Web.UI.flage

{

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

{

if (!IsflostBack fifi GridView1.DataSource == null)

{

// Fetch book data from the database string connectionString =

"Server=localhost;Database=bank;Uid=root;flwd=1234;";

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

FROM Books";

string query = "SELECT BookID, Title, Author, Genre,flrice

MySqlCommand command = new MySqlCommand(query, connection); connection.Open();

DataTable dataTable = new DataTable(); using (MySqlDataAdapter adapter = new

MySqlDataAdapter(command))

{

adapter.Fill(dataTable);

}

GridView1.DataSource = dataTable; GridView1.DataBind();

// Store data in session Session["BookData"] = dataTable;

}

}

}

protected void GridViewBooks\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

Bookdetail.aspx

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="bookdetails.aspx.cs" Inherits="practical\_7\_2.bookdetails" %>

<!DOCTYflE html>

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

<head runat="server">

<title>Bookstore - Book Details</title>

</head>

<body>

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

<h1>Book Details</h1>

<div>

<div class="card-body">

<p class="card-text"><strong>Book Title</strong>:<asp:Label ID="lblTitle" runat="server" Text=""></asp:Label>

<p class="card-text"><strong>Author:</strong> <asp:Label ID="lblAuthor" runat="server" Text=""></asp:Label></p>

<p class="card-text"><strong>Genre:</strong> <asp:Label ID="lblGenre" runat="server" Text=""></asp:Label></p>

<p class="card-text"><strong>flrice: </strong> <asp:Label ID="lblprice" runat="server"></asp:Label></p>

</div>

</div>

</form>

</body>

</html>

Bookdetail.aspx.cs

using System;

using System.Collections.Generic; using System.Data;

using System.Linq; using System.Web; using System.Web.UI;

using System.Web.UI.WebControls; using MySql.Data.MySqlClient;

namespace practical\_7\_2

{

public partial class bookdetails : System.Web.UI.flage

{

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

{

if (!IsflostBack)

{

// Retrieve book ID from query string

int bookID = Convert.ToInt32(Request.QueryString["BookID"]);

// Retrieve book data from session

DataTable dataTable = (DataTable)Session["BookData"]; if (dataTable != null)

{

foreach (DataRow row in dataTable.Rows)

{

if (Convert.ToInt32(row["BookID"]) == bookID)

{

lblTitle.Text = row["Title"].ToString(); lblAuthor.Text = row["Author"].ToString(); lblGenre.Text = row["Genre"].ToString(); lblprice.Text= row["flrice"].ToString(); break;

}

}

}

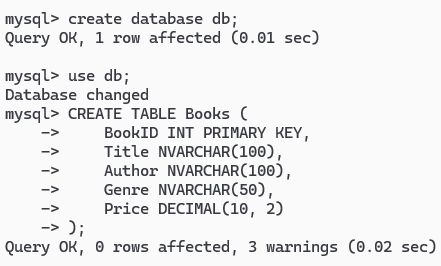
}

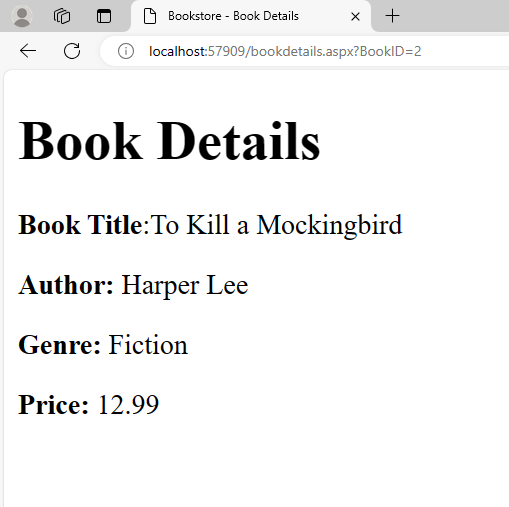
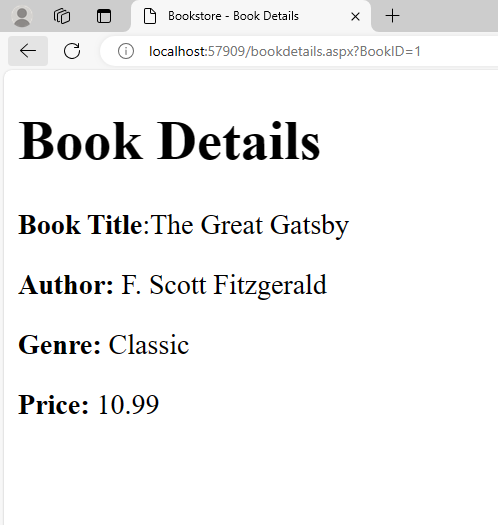
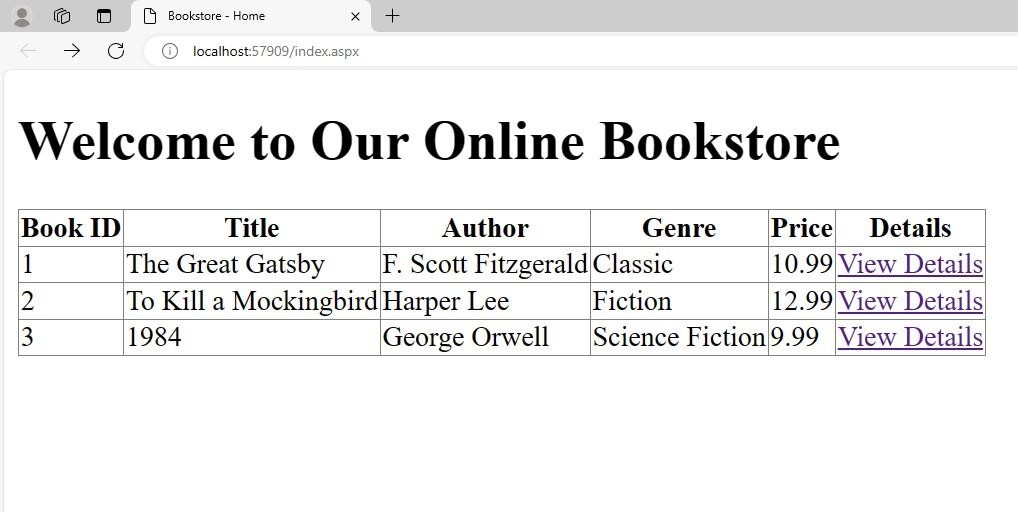
}

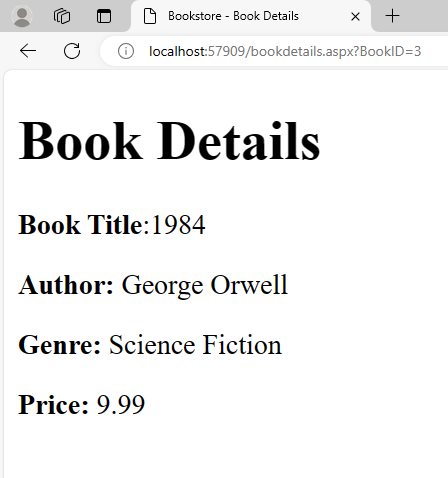
}

}

# Output:







1. You are tasked with implementing a user registration form for a social networking website. The registration form (register.aspx) should collect information such as username, email, password, and date of birth from the user. Implement server-side validation to ensure that all required fields are filled, the email address is in the correct format, and the password meets the complexity requirements. Upon successful registration, store user information in the database and redirect them to a welcome page (welcome.aspx).(Self Study)

Code:

Register.aspx

<!DOCTYflE html>

<html>

<head>

<title>User Registration</title>

</head>

<body>

<h2>User Registration Form</h2>

<form action="register.aspx" method="post">

<label for="username">Username:</label>

<input type="text" id="username" name="username" required><br><br>

<label for="email">Email:</label>

<input type="email" id="email" name="email" required><br><br>

<label for="password">flassword:</label>

<input type="password" id="password" name="password" required><br><b

<label for="dob">Date of Birth:</label>

<input type="date" id="dob" name="dob" required><br><br>

<input type="submit" value="Register">

</form>

</body>

</html> Register.aspx.cs

using System;

using System.Data.SqlClient;

public partial class register : System.Web.UI.flage

{

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

{

if (Request.HttpMethod == "flOST")

{

string username = Request.Form["username"]; string email = Request.Form["email"]; string password = Request.Form["password"]; string dob = Request.Form["dob"];

if (string.Is ullOrEmpty(username) || string.Is ullOrEmpty(email)

|| string.Is ullOrEmpty(password) || string.Is ullOrEmpty(dob))

{

Response.Write("<script>alert('fllease fill in all required fields.');</script>");

return;

}

if (!IsValidEmail(email))

{

Response.Write("<script>alert('fllease enter a valid email address.');</script>");

return;

}

if (!IsStrongflassword(password))

{

Response.Write("<script>alert('flassword must be at least 8 characters long and contain at least one uppercase letter, one lowercase letter, one number, and one special character.');</script>");

return;

}

try

{

SqlConnection conn = new SqlConnection("YourConnectionString");

SqlCommand cmd = new SqlCommand("I SERT I TO Users (Username, Email, flassword, DOB) VALUES (@Username, @Email, @flassword, @DOB)", conn);

cmd.flarameters.AddWithValue("@Username", username); cmd.flarameters.AddWithValue("@Email", email); cmd.flarameters.AddWithValue("@flassword", password);

cmd.flarameters.AddWithValue("@DOB", dob); conn.Open();

cmd.Execute onQuery(); conn.Close(); Response.Redirect("welcome.aspx");

}

catch (Exception ex)

{

Response.Write("<script>alert('An error occurred during registration. fllease try again later.');</script>");

}

}

}

private bool IsValidEmail(string email)

{

try

{

var addr = new System. et.Mail.MailAddress(email); return addr.Address == email;

}

catch

{

return false;

}

}

private bool IsStrongflassword(string password)

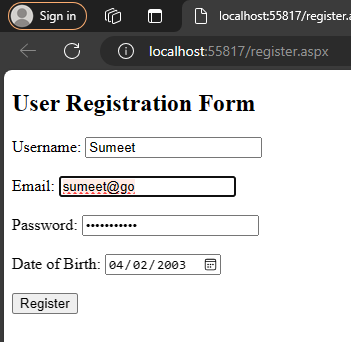
{

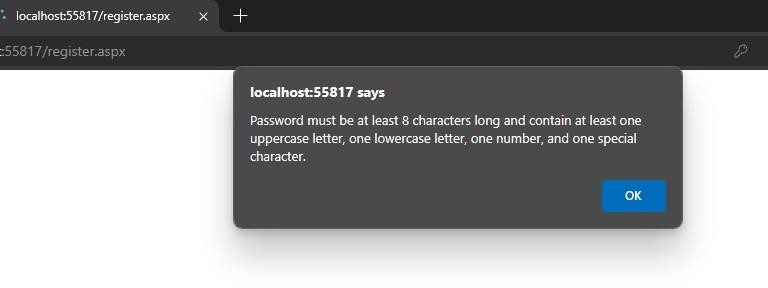
return password.Length >= 8 fifi password.Any(char.IsUpper) fifi password.Any(char.IsLower) fifi password.Any(char.IsDigit) fifi password.Any(char.IsSymbol);

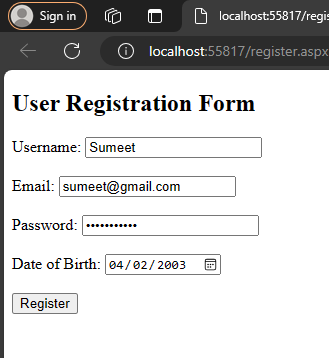
}

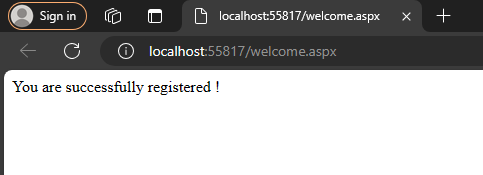
}

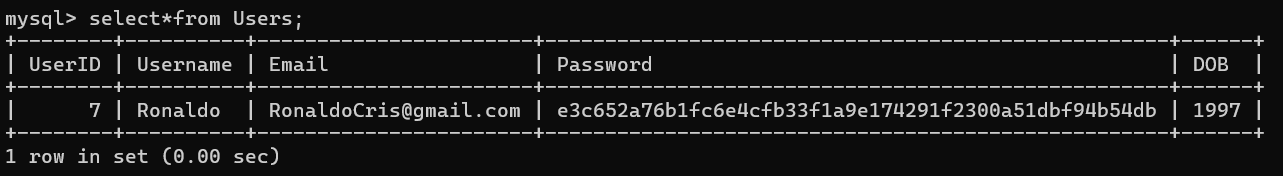
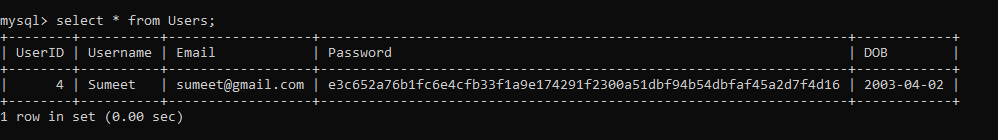
Output:











**Practical No:- 8**

**Aim:** Design Web Application to produce and consume a web Service, WCF Service

### Theory:

Web services are essential components of modern web applications, facilitating communication between different software systems over the internet. A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It enables applications to exchange data and perform actions without knowing the internal details of each other. In the context of temperature conversion, a web service can be created to expose methods that accept temperature values in Fahrenheit or Celsius and return the equivalent values in the other unit.

WCF (Windows Communication Foundation) is a framework for building service-oriented applications. It provides a unified programming model for developing distributed applications, including web services.

WCF enables developers to create web services that are interoperable across different platforms and communication protocols. In the scenario of temperature conversion, WCF can be used to implement a service contract defining methods for converting temperatures between Fahrenheit and Celsius, along with the necessary data contracts for input and output parameters.

To implement a web service for temperature conversion, developers can use ASP.NET Web API or WCF to create a service endpoint that exposes methods for converting temperatures. The service can be hosted on a web server and accessed by client applications through HTTP requests. Clients can consume the service by sending temperature values in either Fahrenheit or Celsius format and receiving the corresponding converted values in the desired unit. This demonstrates the use of web services to provide interoperable and scalable solutions for exchanging data and performing operations across distributed systems.

### Code:

1. Write a program to implement to create a simple web service that converts the temperature from Fahrenheit to Celsius and vice a versa.

Temperature.aspx

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="Temperature.aspx.cs" Inherits="Temperature\_practical\_1.Temperature" %>

<!DOCTYflE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

=<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>

<br />

<asp:RadioButtonList ID="RadioButtonList1" runat="server" AutoflostBack="True" OnSelectedIndexChanged="RadioButtonList1\_SelectedIndexChanged">

<asp:ListItem>celsius</asp:ListItem>

<asp:ListItem>Fahrenhiet</asp:ListItem>

</asp:RadioButtonList>

<br />

</div>

</form>

</body>

</html>

Temperature.aspx.cs

using System;

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

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls;

namespace Temperature\_practical\_1

{

public partial class Temperature : System.Web.UI.flage

{

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

{

}

protected void RadioButtonList1\_SelectedIndexChanged(object sender, EventArgs e)

{

mywebservice ws=new mywebservice(); switch(RadioButtonList1.SelectedIndex)

{

case 0: TextBox2.Text = ws.ftoc(Convert.ToDouble(TextBox1.Text)).ToString();

break;

case 1:TextBox2.Text = ws.ctof(Convert.ToDouble(TextBox1.Text)).ToString();

break;

}

}

}

}

Mywebservices.asmx.cs using System;

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

using System.Web;

using System.Web.Services;

namespace Temperature\_practical\_1

{

/// <summary>

/// Summary description for mywebservice

/// </summary>

[WebService( amespace = "<http://tempuri.org/>")] [WebServiceBinding(ConformsTo = Wsiflrofiles.Basicflrofile1\_1)] [System.ComponentModel.ToolboxItem(false)]

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

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

public class mywebservice : System.Web.Services.WebService

{

[WebMethod]

public string HelloWorld()

{

return "Hello World";

}

[WebMethod]

public double ctof(double c)

{

return ((c \* 1.8) + 32);

}

[WebMethod]

public double ftoc(double f)

{

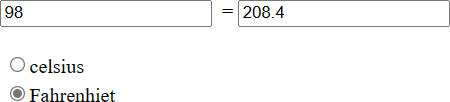
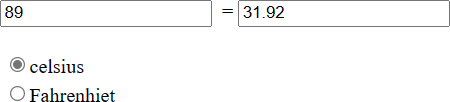
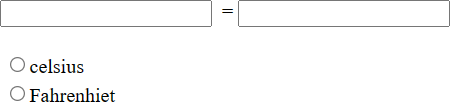
return ((f - 32) \* 0.56);

}

}

}

### Output:



1. Write a program to implement to create a simple web service accepts a number and return the factorial of the number.

### Code:

factorial.cs:

using System;

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

using System.Web;

using System.Web.Services; using System.Web.UI;

using System.Web.UI.WebControls;

namespace prac8b

{

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

{

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

{

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

{

myWebService ws = new myWebService(); double input = 0;

if (double.TryParse(TextBox1.Text, out input))

{

double result = ws.fac(input); Label1.Text = result.ToString();

}

else

{

Label1.Text = "Invalid input";

}

}

}

}

myWebService.asmx.cs:

using System;

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

using System.Web;

using System.Web.Services;

namespace prac8b

{

/// <summary>

/// Summary description for myWebService

/// </summary>

[WebService(Namespace = "<http://tempuri.org/>")] [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)] [System.ComponentModel.ToolboxItem(false)]

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

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

public class myWebService : System.Web.Services.WebService

{

[WebMethod]

public string HelloWorld()

{

return "Hello World";

}

public double fac(double c)

{

double factorial = 1;

for (int i = 1; i <= c; i++)

{

factorial \*= i;

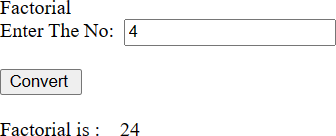
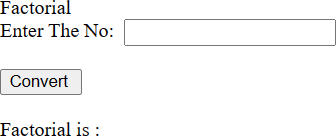
return factorial;

}

}

}

### Output:



1. Write a program to implement to create a WCF accepts a number and return the factors of that number

Code:

**ProduceApp:**

Iservice1.cs

using System;

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

using System.Runtime.Serialization; using System.ServiceModel;

using System.ServiceModel.Web; using System.Text;

namespace produceapp

{

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

[ServiceContract]

public interface IService1

{

[OperationContract]

int[] FindFactors(int number);

}

}

Service1.svc.cs

using System;

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

using System.Runtime.Serialization; using System.ServiceModel;

using System.ServiceModel.Web; using System.Text;

namespace produceapp

{

public class Service1 : IService1

{

public int[] FindFactors(int number)

{

List<int> factors = new List<int>(); for (int i = 1; i <= number; i++)

{

if (number % i == 0)

{

factors.Add(i);

}

}

return factors.ToArray();

}

}

}

**ConsumeApp:**

Factor.aspx

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="factors.aspx.cs" Inherits="consumeapp.factors" %>

<!DOCTYflE html>

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

<head runat="server">

<title>Factorial Calculator</title>

</head>

<body>

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

<div>

<h1>Factorial Calculator</h1>

<asp:Label ID="InputLabel" runat="server" Text="Enter a number:"></asp:Label>

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

<asp:Button ID="CalculateButton" runat="server" Text="Find Factors" OnClick="CalculateButton\_Click" />

<br />

<asp:GridView ID="FactorsGridView" runat="server" AutoGenerateColumns="False">

<Columns>

<asp:BoundField DataField="Factors" HeaderText="Factors" />

</Columns>

</asp:GridView>

</div>

</form>

</body>

</html>

Factor.aspx.cs

using System;

using System.Collections.Generic; using System.Data;

using System.Linq; using System.Web; using System.Web.UI;

using System.Web.UI.WebControls;

namespace consumeapp

{

public partial class factors : System.Web.UI.flage

{

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

{

}

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

{

"Factor"

int number = Convert.ToInt32( umberTextBox.Text); myfactors.Service1Client s=new myfactors.Service1Client(); int[] factors = s.FindFactors(number);

DataTable dt = new DataTable();

dt.Columns.Add("Factors", typeof(int)); // Add a column with the name

foreach (int factor in factors)

{

dt.Rows.Add(factor);

}

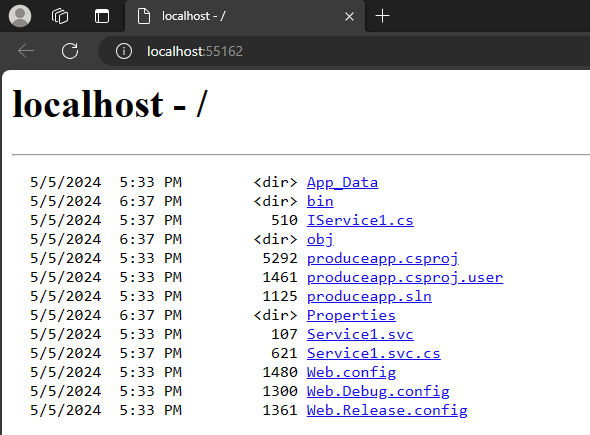
FactorsGridView.DataSource = dt; FactorsGridView.DataBind();

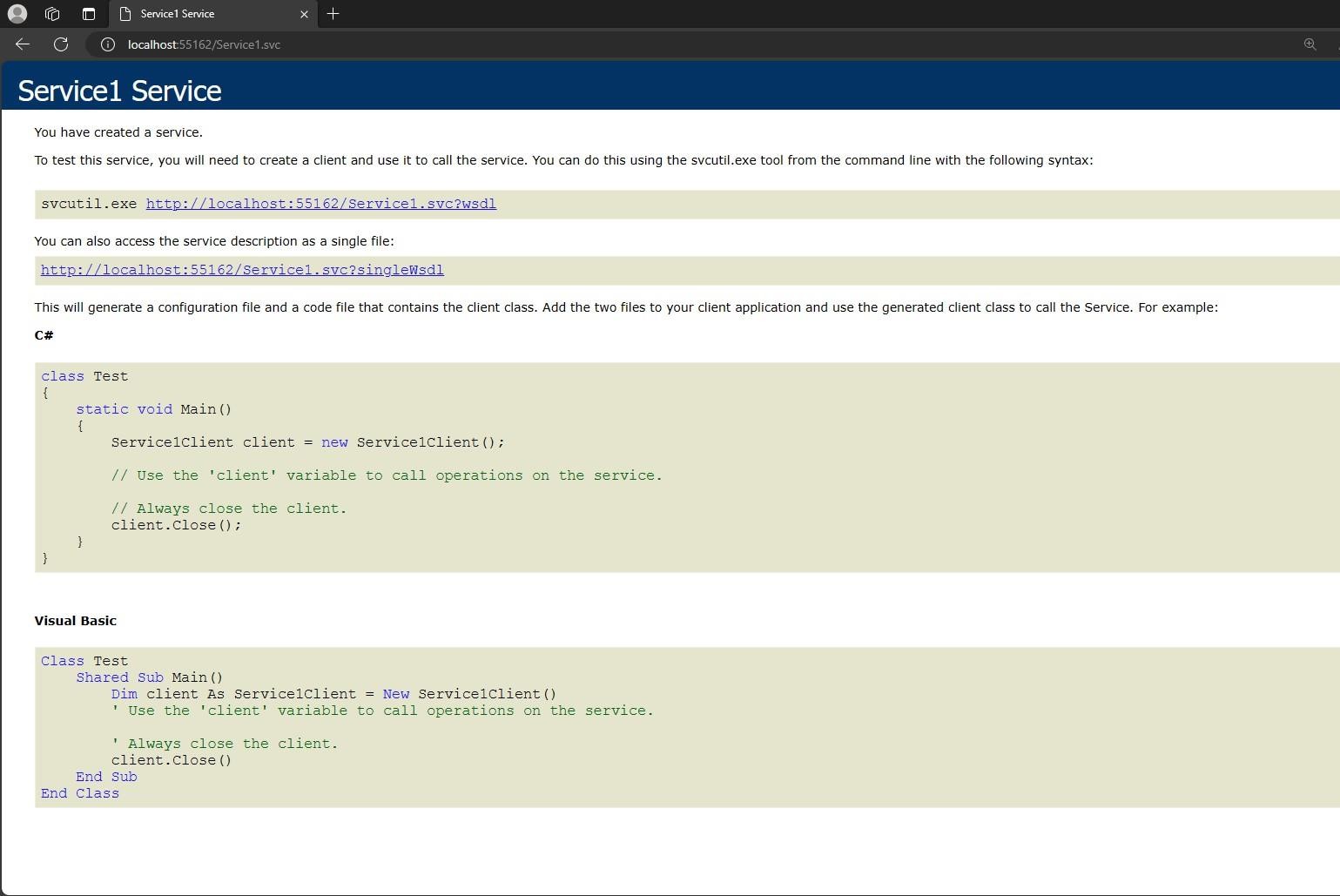
}

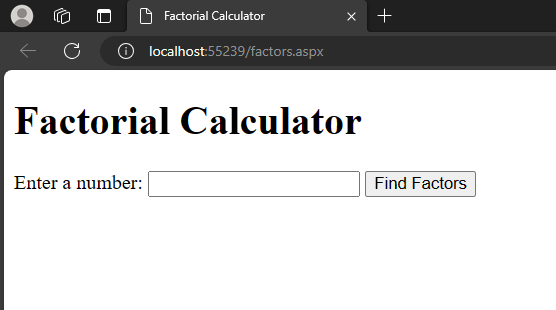
}

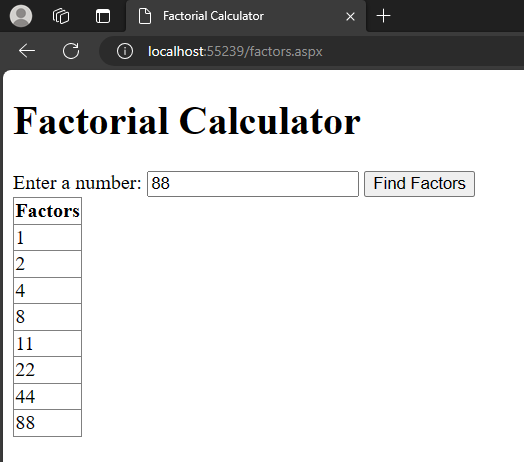
}

Output:









1. **Write a program to implement to create a WCF accepts a String and return reverse (Self-study)**

**CODE:**

**IService1.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization; using System.ServiceModel;

using System.ServiceModel.Web; using System.Text;

namespace produceapp

{

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

[ServiceContract]

public interface IService1

{

[OperationContract]

string ReverseString(string str);

}

}

**IService1.svc**

using System;

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

using System.Runtime.Serialization; using System.ServiceModel;

using System.ServiceModel.Web; using System.Text;

namespace produceapp

{

public class Service1 : IService1

{

public string ReverseString(string input)

{

char[] charArray = input.ToCharArray(); Array.Reverse(charArray);

return new string(charArray);

}

}

}

**WebForm1.aspx**

<%@ flage Language="C " AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="consumerapp.WebForm1" %>

<!DOCTYflE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<h1>Reverse String</h1>

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

<asp:Button ID="btnReverse" runat="server" Text="Reverse" OnClick="btnReverse\_Click" />

<br />

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

</div>

</form>

</body>

</html>

**WebForm1.aspx.cs**

using System;

using System.Collections.Generic; using System.Data;

using System.Linq; using System.Web; using System.Web.UI;

using System.Web.UI.WebControls;

namespace consumerapp

{

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

{

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

{

}

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

{

// Get the input string from the TextBox string input = txtInput.Text;

reverseString.Service1Client s = new reverseString.Service1Client();

// Reverse the input string

string reversed = s.ReverseString(input);

// Display the reversed string in the Label

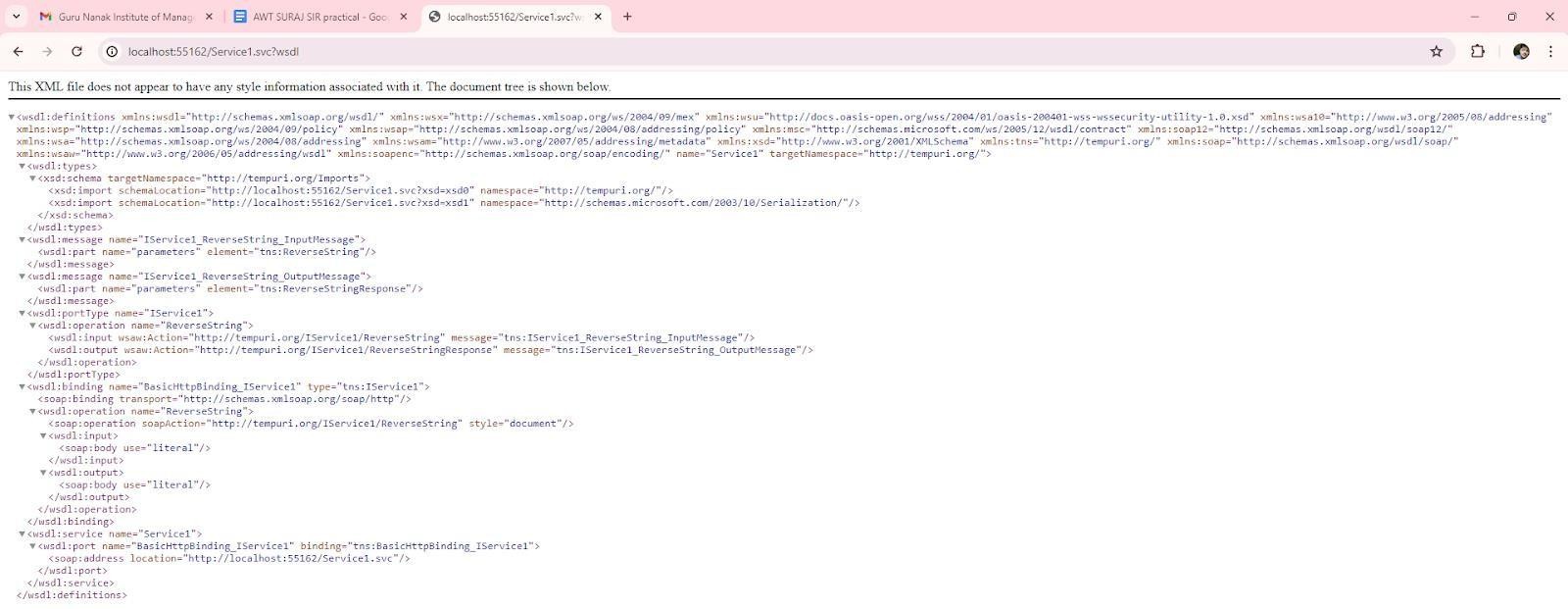
lblReversed.Text = reversed;

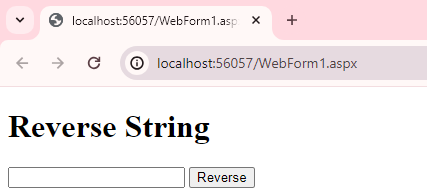
}

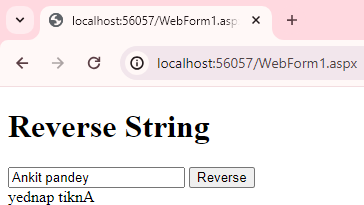
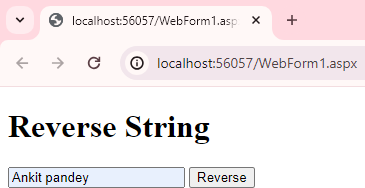
**)**

**)**

**OUHflUH:**







**Practical No:- 9**

**Aim:** Build websites to demonstrate the working of entity framework in dot net.

### Theory:

C# Entity framework is an Object Relational Mapping (ORM) framework that gives developers an automated way to store and access databases. The Entity Framework allows developers to work with data at a higher level of abstraction.

The Database-First approach provides an alternative to the Code First and Model First approaches to the Entity data model, and it creates model codes (Classes, properties, dB Context, etc.) from the database to the project, and those classes become the link between the database and controller.

Entity Framework is an open-source object-relational mapper framework for .NET applications supported by Microsoft. It increases the developer’s productivity as it enables developers to work with data using objects of domain-specific classes without focusing on the underlying database tables and columns where this data is stored. It eliminates the need for most of the data-access code which is used to interact with the database that developers usually need to write.

It provides an abstract level to the developers to work with a relational table and columns by using the domain-specific object. It also reduces the code size of the data specific applications and also the readability of the code increases by using it. This is a new technology for accessing the data for Microsoft application. The latest version for Entity Framework is 6.0.

It provides a connection between the business entity and data tables in the database. It saves data stored in the properties of business entities and also retrieves data from the database and converts it to business entities objects automatically.

When defining the class and features of entity framework first and then entity framework convert it into the conceptual model first and it creates database and objects in the database from the conceptual model this method is called Code First.

### Code:

1.Create a program by using entity framework in dot net to find the GCD of two numbers entered by the user. Create a razor file to take input from user.

### Index.cshtml :

@{

ViewData["Title"] = "GCD Calculator";

}

<h2>@ViewData["Title"]</h2>

@using (Html.BeginForm("CalculateGCD", "GCD", FormMethod.Post))

{

<div>

<label>Number 1:</label>

<input type="number" id="number1" name="number1" required>

</div>

<div>

<label>Number 2:</label>

<input type="number" id="number2" name="number2" required>

</div>

<button type="submit">Calculate GCD</button>

}

@if (ViewBag.GCD != null)

{

<div role="alert">

GCD: @ViewBag.GCD

</div>

}

### GCDcontroller.cs :

using System;

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

using System.Web; using System.Web.Mvc;

namespace GCD.Controllers

{

public class GCDController : Controller

{

// GET: GCD

public ActionResult Index()

{

return View();

}

[HttpPost]

public ActionResult CalculateGCD(int number1, int number2)

{

int gcd = CalculateGCD1(number1, number2); ViewBag.GCD = gcd;

return View("Index");

}

private int CalculateGCD1(int a, int b)

{

while (b != 0)

{

int temp = b; b = a % b;

a = temp;

}

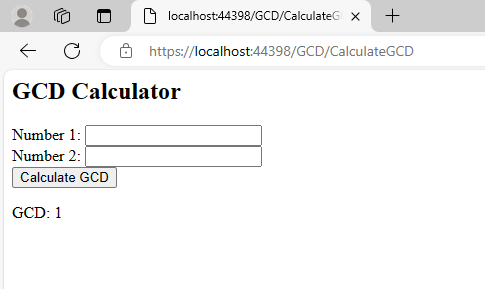
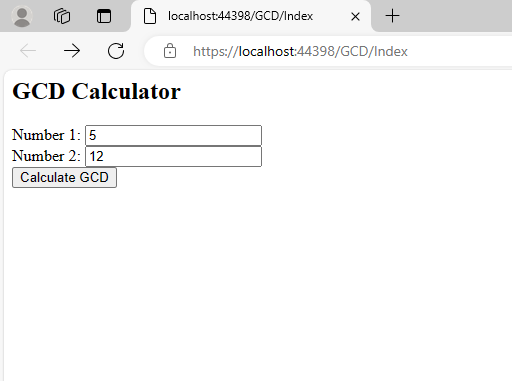
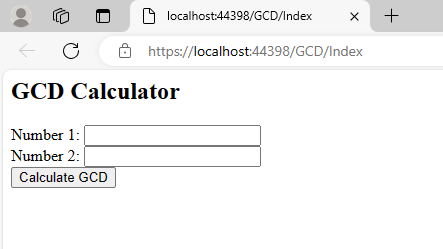
return a;

}

}

}

### Output:



**Practical No:- 10**

**AIM: Design MVC based Web applications.**

**1. Create a MVC based Web Application to find the factors of number entered by the user. Create a View to take input from the user and display the answer.**

**CODE:**

**Index.cshtml :**

@{

ViewData["Title"] = "GCD Calculator";

}

<h2>@ViewData["Title"]</h2>

@using (Html.BeginForm("CalculateGCD", "GCD", FormMethod.Post))

{

<div>

<label>Number 1:</label>

<input type="number" id="number1" name="number1" required>

</div>

<div>

<label>Number 2:</label>

<input type="number" id="number2" name="number2" required>

</div>

<button type="submit">Calculate GCD</button>

}

@if (ViewBag.GCD != null)

{

<div role="alert">

GCD: @ViewBag.GCD

</div>

}

### GCDcontroller.cs :

using System;

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

using System.Web; using System.Web.Mvc;

namespace GCD.Controllers

{

public class GCDController : Controller

{

// GET: GCD

public ActionResult Index()

{

return View();

}

[HttpPost]

public ActionResult CalculateGCD(int number1, int number2)

{

int gcd = CalculateGCD1(number1, number2); ViewBag.GCD = gcd;

return View("Index");

}

private int CalculateGCD1(int a, int b)

{

while (b != 0)

{

int temp = b; b = a % b;

a = temp;

}

return a;

}

}

}

**OUTPUT:**



