MODULE 3: DATABASE PROGRAMMING IN ASP.NET

EXPERIMENT NO 5:

Aim: Web Application to demonstrate LINQ with object data source.

In your project, create an App_Code folder. Add a Product.cs file in the App_Code folder. This folder is used for placing classes that can be accessed throughout the application.

DataSetHelper.cs:

```
using System;
using System.Data;
public class DataSetHelper {
public static DataSet GetSample Data Set () {
DataSet dataSet = new DataSet ();
Data Table productsTable = new Data Table (" Products ");
productsTable . Columns. Add (" ProductID ", typeof( int));
productsTable . Columns. Add (" ProductName ", typeof( string ));
productsTable . Columns. Add (" Price ", typeof( decimal));
productsTable . Rows. Add (2 , " Smartphone ", 499.99 m); productsTable . Rows. Add (3 , "
Tablet", 299.99 m); productsTable . Rows. Add (4 , " Headphones", 99.99 m); productsTable
. Rows. Add (5 , " Smartwatch ", 199.99 m);
return dataSet;
        }
}
```

Note:
Right-click on the .cs file in the App_Code folder and check its properties. Make sure the Build Action is set to Compile.

Default.aspx:

```
<%@  Page  Title =" Home  Page "  Language =" C#"  Auto EventWireup =" true " Code
Behind =" Default. aspx. cs" Inherits =" _ 17 _LINQWith Dataset . _Default"
% >
<! DOCTYPE  html >
<html  xmlns =" http :// www . w3 . org /1999/ xhtml" >
<head  runat =" server" >
<title > LINQ  with  DataSet </ title >
<style >
table {
```

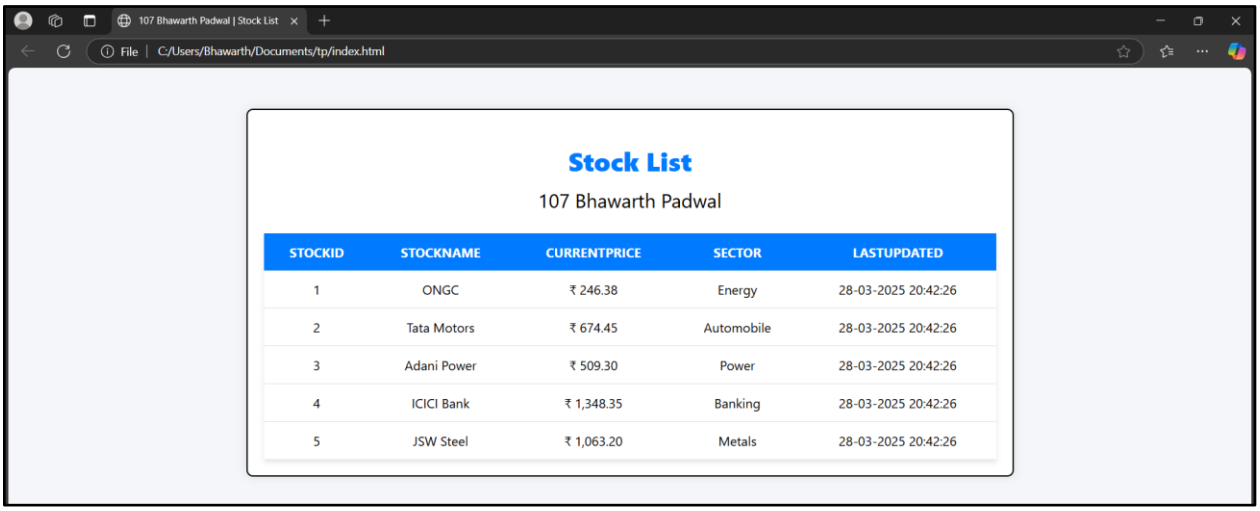
```
width : 50%;
border - collapse :  collapse ;
}
th , td {
padding :  10 px;
border: 1 px solid # ddd ; text - align : center;

}
th {
}
</ style >
</ head >
<body >
background - color:  # f2f2f2 ;
<form  id =" form1 "  runat =" server" >
<div >
<h2 > Product  List </ h2 >
<!-- Grid View Control to Display Products -->
<asp : Grid View ID =" Grid View 1 " runat =" server" Auto Generate Columns =" True " On
Row Data Bound =" Grid View 1 _Row Data Bound " >
</ asp : GridView >
</ div >
</ form >
</ body >
</ html >
```

Default.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. Web Controls;
namespace  _ 17 _LINQWith Dataset {
public partial class _Default : Page {
protected void Page_Load ( object sender , EventArgs e) {
if  (! IsPostBack ) {
DataSet ds = Data SetHelper. GetSample Data Set ();
var products = from product in ds. Tables [" Products "].
AsEnumerable ()
where  product. Field < decimal >(" Price ")  > 100 select  new
{
```

```
ProductID = product. Field <int >(" ProductID "),
ProductName = product. Field <string >(" ProductName "),
Price = product. Field < decimal >(" Price ")
};
Grid View 1 . Data Source = products. To List (); Grid View 1 . Data Bind ();
}}
protected void Grid View 1 _Row Data Bound ( object sender , Grid View Row EventArgs e)
{
if ( e. Row . Row Type == Data ControlRow Type . DataRow )
{
e. Row . Cells [2]. Text = string . Format ("{0: C}", Convert.
To Decimal( e. Row . Cells [2]. Text));
}}}}
```

Output:



STOCKID	STOCKNAME	CURRENTPRICE	SECTOR	LASTUPDATED
1	ONGC	₹ 246.38	Energy	28-03-2025 20:42:26
2	Tata Motors	₹ 674.45	Automobile	28-03-2025 20:42:26
3	Adani Power	₹ 509.30	Power	28-03-2025 20:42:26
4	ICICI Bank	₹ 1,348.35	Banking	28-03-2025 20:42:26
5	JSW Steel	₹ 1,063.20	Metals	28-03-2025 20:42:26

EXPERIMENT NO 6:

Aim: Web Application to demonstrate LINQ with data set.

In your project, create an App_Code folder. Add a Product.cs file in the App_Code folder. This folder is used for placing classes that can be accessed throughout the application.

DataSetHelper.cs:

```
using System ; using System . Data;
public class Data SetHelper{
public static DataSet GetSample Data Set () {
DataSet dataSet = new DataSet ();
Data Table productsTable = new Data Table (" Products ");
productsTable . Columns. Add (" ProductID ", typeof( int));
productsTable . Columns. Add (" ProductName ", typeof( string ));
productsTable . Columns. Add (" Price ", typeof( decimal));
productsTable . Rows. Add (1 , " Laptop ", 799.99 m);
productsTable . Rows. Add (2 , " Smartphone ", 499.99 m);
productsTable . Rows. Add (3 , " Tablet", 299.99 m);
productsTable . Rows. Add (4 , " Headphones", 99.99 m);
productsTable . Rows. Add (5 , " Smartwatch ", 199.99 m);
dataSet. Tables. Add ( productsTable );
return dataSet;
}}
```

Note:
Right-click on the .cs file in the App_Code folder and check its properties. Make sure the Build Action is set to Compile

Default.aspx:

```
<%@  Page  Title =" Home  Page "  Language =" C#"  Auto EventWireup =" true " Code
Behind =" Default. aspx. cs" Inherits =" _ 17 _LINQWith Dataset . _Default"
% >
<! DOCTYPE  html >
<html  xmlns =" http :// www . w3 . org /1999/ xhtml" >
<head  runat =" server" >
<title > LINQ  with  DataSet </ title >
<style >
table {
width : 50%;
```

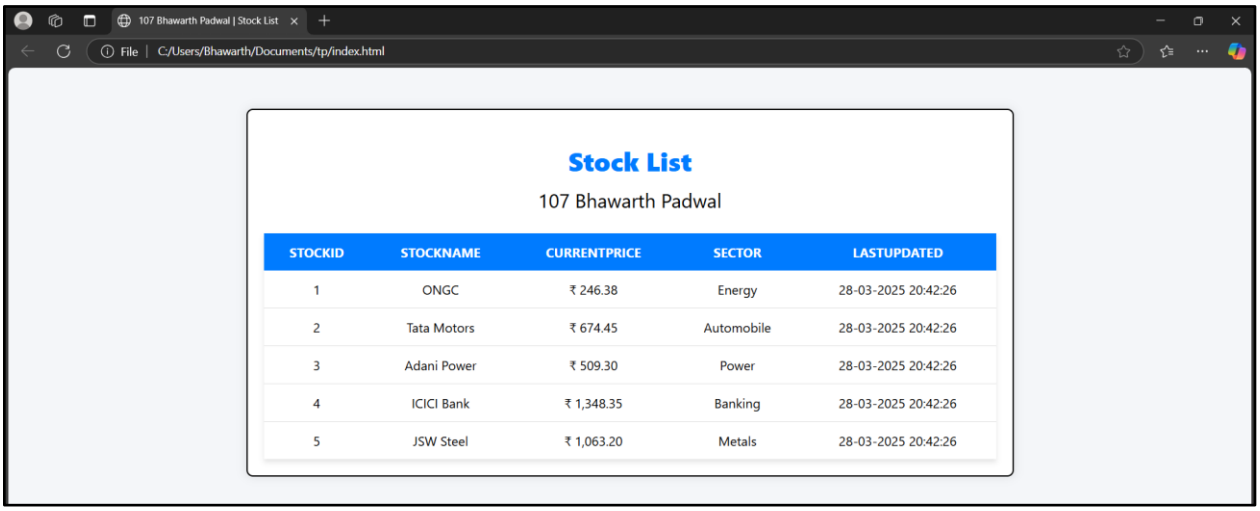
```
border - collapse :  collapse ;
}
th , td {
padding :  10 px;
border: 1 px solid # ddd ; text - align : center;
 }
th {
}
</ style >
</ head >
<body >
background - color:  # f2f2f2 ;
<form  id =" form1 "  runat =" server" >
<div >
<h2 > Product  List </ h2 >
<!-- Grid View Control to Display Products -->
<asp : Grid View ID =" Grid View 1 " runat =" server" Auto Generate Columns =" True " On
Row Data Bound =" Grid View 1 _Row Data Bound " >
</ asp : GridView >
</ div >
</ form >
</ body >
</ html >
```

Default.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. Web Controls;
namespace  _ 17 _LINQWith Dataset{
public partial class _Default : Page{
protected void Page_Load ( object sender , EventArgs e){
if  (! IsPostBack ){
DataSet ds = Data SetHelper. GetSample Data Set ();
var products = from product in ds. Tables [" Products "].
AsEnumerable ()
where  product. Field < decimal >(" Price ")  >
100 // Filter products with price > 100
select  new{
ProductID = product. Field <int >(" ProductID "),
```

```
ProductName = product. Field <string >(" ProductName "),
Price = product. Field < decimal >(" Price ")
};
Data Source = products. To List ();
Data Bind ();
}}
protected void Grid View 1 _Row Data Bound ( object sender , Grid View Row EventArgs e)
{
if ( e. Row . Row Type == Data ControlRow Type . DataRow )
{
e. Row . Cells [2]. Text = string . Format ("{0: C}", Convert.
To Decimal( e. Row . Cells [2]. Text));
}}}}
```

Output:



STOCKID	STOCKNAME	CURRENTPRICE	SECTOR	LASTUPDATED
1	ONGC	₹ 246.38	Energy	28-03-2025 20:42:26
2	Tata Motors	₹ 674.45	Automobile	28-03-2025 20:42:26
3	Adani Power	₹ 509.30	Power	28-03-2025 20:42:26
4	ICICI Bank	₹ 1,348.35	Banking	28-03-2025 20:42:26
5	JSW Steel	₹ 1,063.20	Metals	28-03-2025 20:42:26

EXPERIMENT NO 7:

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

Default.aspx:

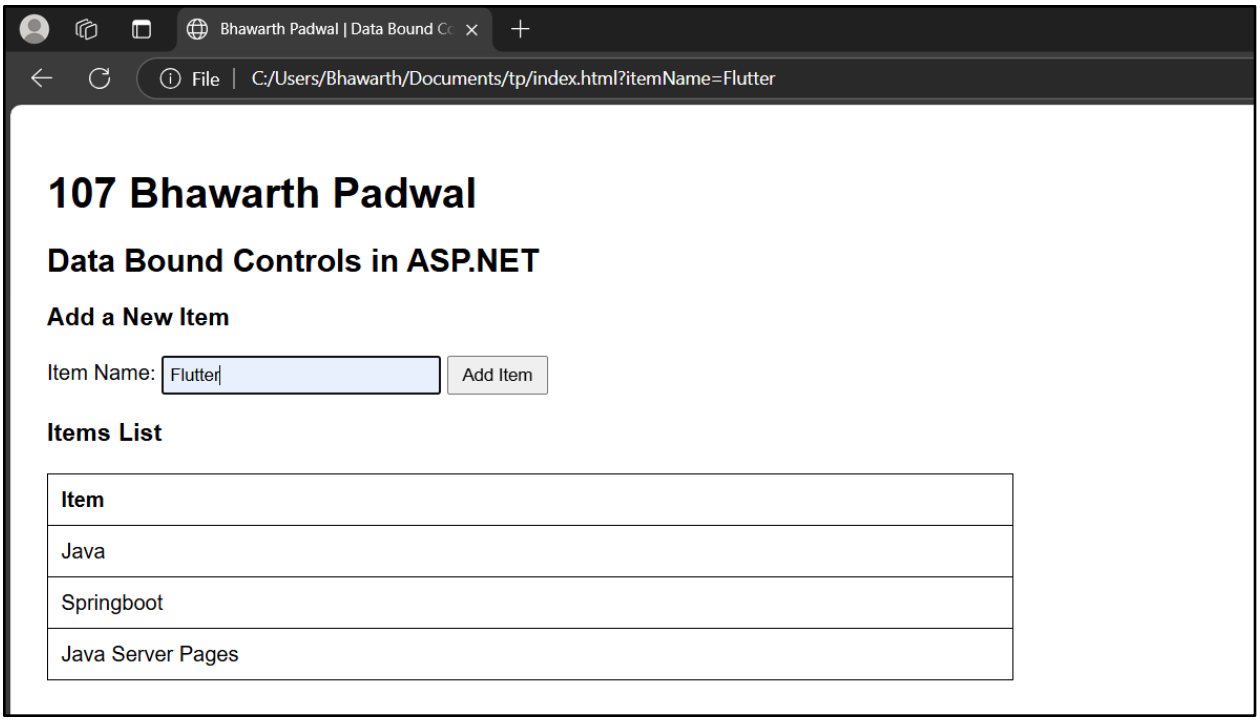
```
<%@ Page Language =" C#" Auto EventWireup =" true " Code Behind =" Default. aspx. cs"
Inherits =" _ 18 _Data Bound Controls . Default" % >
<! DOCTYPE  html >
<html  xmlns =" http :// www . w3 . org /1999/ xhtml" >
<head  runat =" server" >
<title > Data Binding  with  GridView </ title >
<style >
table {
width : 100%;
border - collapse :  collapse ;
}
table , th , td {
border: 1 px solid black;
}
th , td {
padding : 8 px; text - align : left;
}
</ style >
</ head >
<body >
<form  id =" form1 "  runat =" server" >
<div >
<h1 > Data Bound  Controls in ASP . NET </ h1 >
<!-- Add Item Form -->
<h2 > Add  a  Student </ h2 >
<label  for=" txtItem Name " > Student  Name : </ label >
<asp : TextBox ID =" txtItem Name " runat =" server" / >
<asp : Button ID =" btn Add Item " runat =" server" Text =" Add Item " On Click =" btn Add
Item_Click " / >
<br / ><br / >
<!-- Grid View to display data -->
<h2 > Student  List </ h2 >
<asp : Grid View ID =" gvItems" runat =" server" Auto Generate Columns =" True " / >
</ div >
</ form >
```

```
</ body >
</ html >
```

Default.aspx.cs:

```
using System ;
using System . Collections. Generic; using System . Linq ;
using System . Web ; using System . Web . UI;
using System . Web . UI. Web Controls;
namespace  _ 18 _Data Bound Controls {
public partial class Default : Page {
private static List <string > items = new List <string >
{
"Bhawarth 107",
"Dattaram 97",
"Ritesh 128"
};

protected void Page_Load ( object sender , EventArgs e)
{
if  (! IsPostBack )
{
Bind Grid View ();
}}
private void Bind Grid View ()
{
gvItems. Data Source = items; gvItems. DataBind ();
}
protected void btn Add Item_Click ( object sender , EventArgs e)
{
string new Item = txtItem Name . Text. Trim (); if (! string . IsNullOrEmpty ( new Item ))
{
items. Add ( new Item );
 }
else
{
}}}}
```

Output:


Bhawarth Padwal | Data Bound Co
File | C:/Users/Bhawarth/Documents/tp/index.html?itemName=Flutter
107 Bhawarth Padwal
Data Bound Controls in ASP.NET
Add a New Item
Item Name: Flutter
Add Item
Items List
Item
Java
Springboot
Java Server Pages

EXPERIMENT NO 8:

Aim: Build websites to demonstrate the working of entity frameworks.

Create a New MVC Project Install Entity Framework

1. Open Tools → NuGet Package Manager → Package Manager Console.
2. Run the following command: Install - Package EntityFramework

Create the Model

1. In Solution Explorer, right-click the Models folder → Add → Class.
2. Name it Student.cs and add the following code:

Students.cs:

```
using System ;
using System . Collections. Generic; using System . Linq ;
using System . Web ;
namespace _ 19 _Entity Framework . Models {
public class Student {
public int ID { get; set; }
public string Name { get; set; }
public string Email { get; set; }
public string Course { get; set; }
public string Contact { get; set; }
} }
```

Create Database Context

1. Right-click Models → Add → Class.
2. Name it RecordContext.cs and add the following code:

RecordContext.cs:

```
using System ;
using System . Collections. Generic; using System . Linq ;
using System . Web ;
using System . Data. Entity;
using  System . Data. Entity. ModelConfiguration . Conventions;
namespace _ 19 _Entity Framework . Models {
public class Record Context : Db Context {
public Record Context () : base (" Record Context ") { } public DbSet < Student > Students {
get; set; }
protected override void On ModelCreating ( Db ModelBuilder modelBuilder)
{
```

```
modelBuilder. Conventions. Remove < Pluralizing Table Name Convention >();
}}}
```

Enable Migrations

1. Open Tools → NuGet Package Manager → Package Manager Console.
2. Run the following command:

Enable - Migrations - ContextType Name _ 19 _EntityFramework . Models.
Record Context

3. This creates a Migrations folder with a Configuration.cs file.
4. Open Configuration.cs in the Migrations folder and replace the Seed method with:

```
protected  override  void  Seed ( _ 19 _Entity Framework . Models.Record Context context)
{
var students = new List < Student >
{
new  Student { Name =" Bhawarth", Email=" Bhawarth@ example . com ", Course =" Java ",
Contact=" +25 -258628 " },
new Student { Name =" Dattaram", Email=" Dattaram@ example . com "
, Course =". NET", Contact=" +25 -258694 " },
new  Student { Name =" Bhawarth", Email=" bhawarth@ example . com ", Course =" Java ",
Contact=" +25 -258999 " },
new Student { Name =" Ritesh", Email=" ritesh@ example . com ", Course =" Linux",
Contact=" +25 -258111 " },
};
students. ForEach ( s => context. Students. Add ( s)); context. Save Changes ();
}
```

5. Run migrations using:

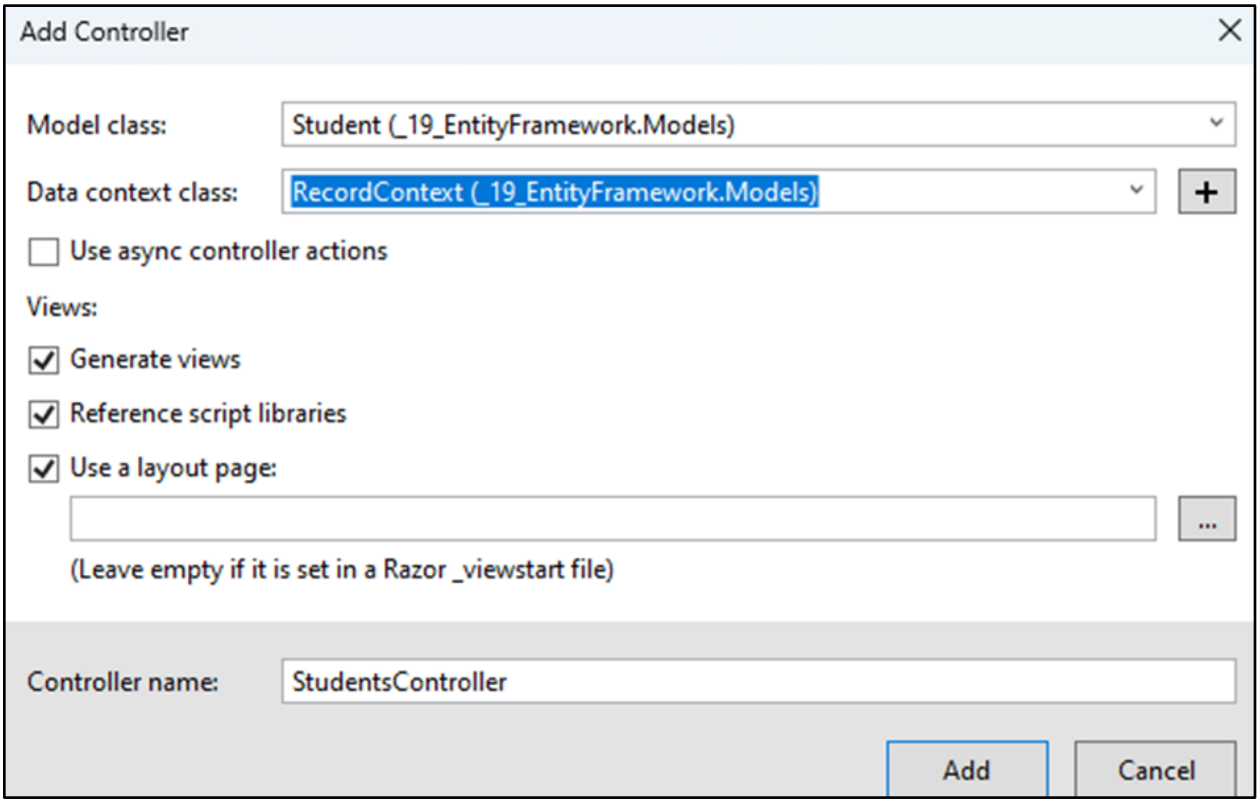
Add - Migration Initial Update - Database
Create a Controller

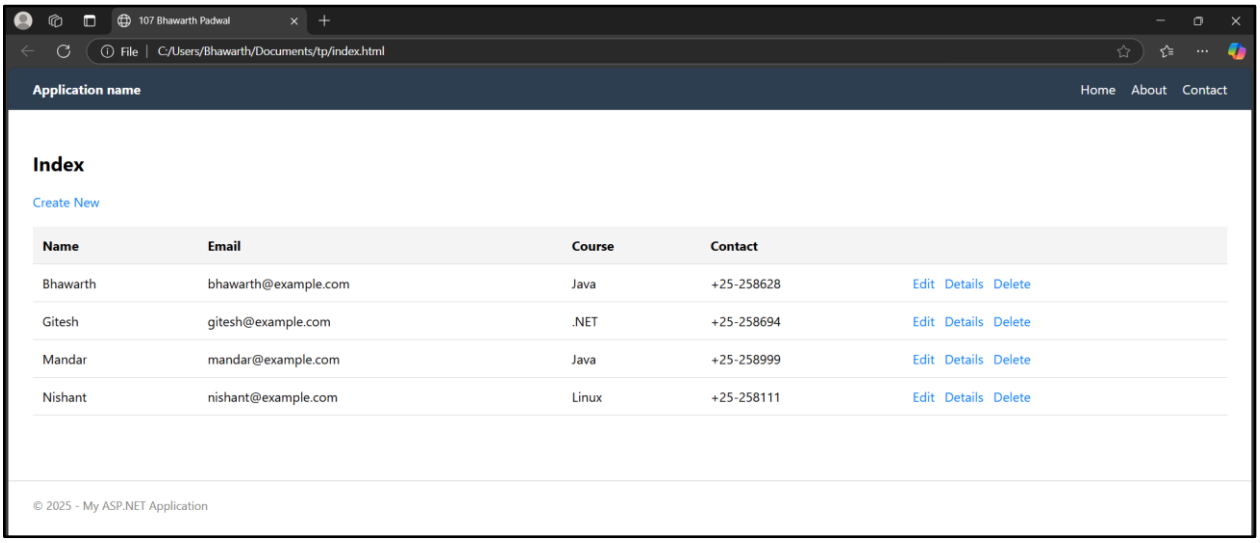
1. Right-click Controllers → Add → Controller.
2. Select MVC 5 Controller with views, using Entity Framework.
3. Choose:
   - Model class: Student (_19_EntityFramework.Models).
   - Data context class: RecordContext (_19_EntityFramework.Models).
4. Click Add.
5. This creates StudentsController.cs with CRUD methods.

Run the Application

1. Run the project.
2. Navigate to http://localhost:portno/Students to see the student list.

Output:


Add Controller
Model class:
Student (_19_EntityFramework.Models)
Data context class:
RecordContext (_19_EntityFramework.Models)
Use async controller actions
Views:
Generate views
Reference script libraries
Use a layout page:
(Leave empty if it is set in a Razor _viewstart file)
Controller name:
StudentsController
Add
Cancel


107 Bhawarth Padwal
File | C:/Users/Bhawarth/Documents/tp/index.html
Application name
Home About Contact
Index
Create New
Name Email Course Contact
Bhawarth bhawarth@example.com Java +25-258628 Edit Details Delete
Gitesh gitesh@example.com .NET +25-258694 Edit Details Delete
Mandar mandar@example.com Java +25-258999 Edit Details Delete
Nishant nishant@example.com Linux +25-258111 Edit Details Delete
© 2025 - My ASP.NET Application

EXPERIMENT NO 9:

Aim: Design a web page to perform CRUD operation using SqlDataSource.

Default.aspx:

```
<%@ Page Language =" C#" Auto EventWireup =" true " Code Behind =" Default. aspx. cs"
Inherits =" practical20 . Default" % >
<! DOCTYPE  html >
<html  xmlns =" http :// www . w3 . org /1999/ xhtml" >
<head  runat =" server" >
<title > Payment Form </ title >
<style >
body {
font - family: Arial , sans - serif; margin : 20 px;
background - color:  # f9f9f9 ;
}
h2 {
}
text - align : center; color: #2 C3E50 ;
form . form - container table { margin : 20 px auto ;
border - collapse : collapse ; background - color: # ffffff; padding : 20 px;
border - radius: 8 px;
box - shadow : 0 px 0 px 10 px rgba (0 ,0 ,0 ,0.1);
}
form . form - container td { padding : 10 px;
font - size : 14 px;
}
. form - container  input[ type =" submit "],
. form - container asp \: Button , # btn Insert {
background - color: #34495 E; color: white ;
padding : 6 px 10 px; border: none ;
border - radius: 5 px; cursor: pointer;
}
# btn Insert: hover {
background - color:  #2 C3E50 ;
}
. aspNet - Grid View { margin : 20 px auto ; width : 90%;
border - collapse :  collapse ;
}
```

```
. aspNet - Grid View th { background - color: #2 C3E50 ; color: white ! important; padding :
10 px;
text - align : center;
}
. aspNet - Grid View th a { color: white ! important; text - decoration : none ;
}
. aspNet - Grid View th a: hover { text - decoration : underline ; color: # f0f0f0 ;
}
. aspNet - Grid View td { padding : 8 px;
text - align : center; color: black;
}
. aspNet - Grid View tr: nth - child ( even ) { background - color: # f2f2f2 ;
}
. aspNet - Grid View tr: nth - child ( odd ) { background - color: # ffffff;
}
</ style >
</ head >
<body >
<h2 >100 Soumen Mondal </ h2 >
<h3 > Payment  Form </ h3 >
<form  class =" form - container"  id =" form1 "  runat =" server" >
<div >
<table >
<tr >
<td > Payment  ID : </ td >
<td >< asp : TextBox  ID =" txtPaymentId "  runat =" server"
/ ></ td >
</ tr >
<tr >
<td > Student  ID : </ td >

<td >< asp : TextBox  ID =" txtStudentId "  runat =" server"
/ ></ td >
</ tr >
<tr >
<td > Student  Name : </ td >
<td >< asp : TextBox  ID =" txtStudentName "  runat =" server"
/ ></ td >
</ tr >
<tr >
```

```
<td > Amount  Paid : </ td >
<td >< asp : TextBox  ID =" txtAmountPaid "  runat =" server"
/ ></ td >
</ tr >
<tr >
<td > Payment  Date : </ td >
<td >< asp : TextBox  ID =" txtPaymentDate "  runat =" server"
/ ></ td >
</ tr >
<tr >
<td > Mode : </ td >
<td >< asp : TextBox  ID =" txtMode "  runat =" server"  / ></ td >
</ tr >
<tr >
<td colspan ="2" >
<asp : Button ID =" btn Insert" runat =" server" Text =" Insert Record " On Click =" btn
Insert_Click " / >
</ td >
</ tr >
</ table >
<asp : Grid View
CssClass =" aspNet - Grid View " ID  ="  Grid View 1  " runat    ="    server" Allow Paging ="
True " Allow Sorting =" True " Auto Generate Columns =" False " Data KeyNames ="
PaymentId "
Data   Source   ID   ="   SqlData   Source   1   " On Selected IndexChanged =" Grid View 1
_Selected IndexChanged " Auto PostBack =" True " >
<Columns >
<asp : Command Field   Show EditButton =" True " Show Delete Button =" True " Show
SelectButton =" True "
/ >
<asp : Bound Field Data Field =" PaymentId " HeaderText =" PaymentId " Read Only =" True
" SortExpression ="
PaymentId " / >
<asp : Bound Field Data Field =" StudentID " HeaderText =" StudentID " SortExpression ="
StudentID " / >
<asp : Bound Field Data Field =" StudentName " HeaderText =" StudentName "
SortExpression =" StudentName " / >
<asp : Bound Field Data Field =" AmountPaid " HeaderText =" AmountPaid "
SortExpression =" AmountPaid " / >
```

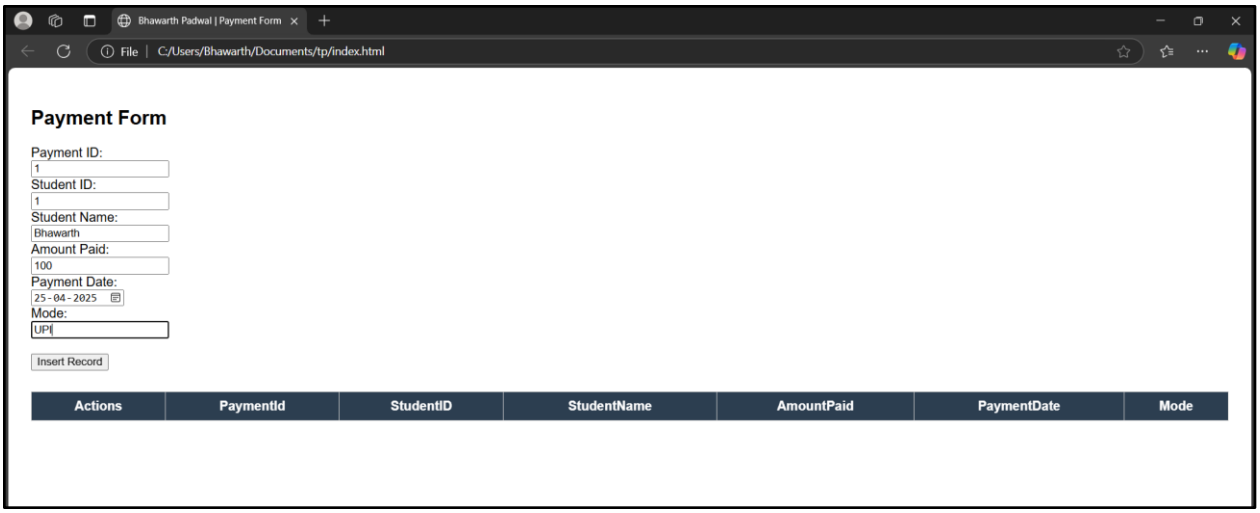
```
<asp : Bound Field Data Field =" PaymentDate " HeaderText =" PaymentDate "
SortExpression =" PaymentDate " / >
<asp : Bound Field Data Field =" Mode " HeaderText =" Mode " SortExpression =" Mode " /
>
</ Columns >
</ asp : GridView >
<asp : SqlData Source ID =" SqlData Source 1 " runat =" server"
Connection String =" <% $ Connection Strings : Payment % >" ProviderName =" <% $
Connection Strings : Payment. ProviderName
% >"
SelectCommand =" SELECT  *  FROM  [ Payment ]" InsertCommand =" INSERT INTO
Payment ( PaymentId , StudentID
, StudentName , AmountPaid , PaymentDate , Mode ) VALUES (@ PaymentId , @ StudentID ,
@ StudentName , @ AmountPaid , @ PaymentDate , @ Mode )"
Update Command =" UPDATE Payment SET StudentID = @ StudentID , StudentName = @
StudentName , AmountPaid = @ AmountPaid , PaymentDate = @ PaymentDate , Mode = @
Mode WHERE PaymentId = @ PaymentId "
Delete Command =" DELETE FROM Payment WHERE PaymentId = @ PaymentId " >
<InsertParameters >
<asp : Parameter  Name =" PaymentId "  Type =" Int32 "  / >
<asp : Parameter Name =" StudentID " Type =" Int32 " / >
<asp : Parameter Name =" StudentName " Type =" String " / >
<asp : Parameter Name =" AmountPaid " Type =" Decimal" / >
<asp : Parameter Name =" PaymentDate " Type =" Date Time " / >
<asp : Parameter Name =" Mode " Type =" String " / >
</ InsertParameters >
<Update Parameters >
<asp : Parameter Name =" StudentID " Type =" Int32 " / >
<asp : Parameter Name =" StudentName " Type =" String " / >
<asp : Parameter Name =" AmountPaid " Type =" Decimal" / >
<asp : Parameter Name =" PaymentDate " Type =" Date Time " / >
<asp : Parameter Name =" Mode " Type =" String " / >
<asp : Parameter  Name =" PaymentId "  Type =" Int32 "  / >
</ Update Parameters >
<Delete Parameters >
<asp : Parameter  Name =" PaymentId "  Type =" Int32 "  / >
</ Delete Parameters >
</ asp : SqlDataSource >
</ div >
</ form >
```

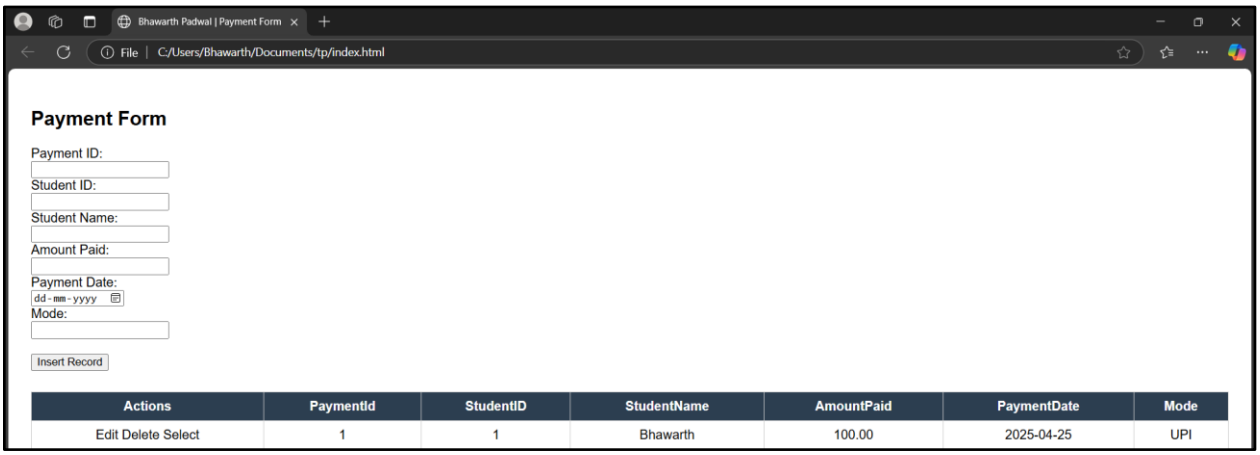
```
</ body >
</ html >
```

Default.aspx.cs:

```
using System ;
using System . Web . UI;
using System . Web . UI. Web Controls;

namespace practical20
{
public partial class Default : System . Web . UI. Page
{
protected void Page_Load ( object sender , EventArgs e)
{
}
protected void btn Insert_Click ( object sender , EventArgs e)
{
SqlData Source 1 . InsertParameters [" PaymentId "]. DefaultValue = txtPaymentId . Text;
SqlData Source 1 . InsertParameters [" StudentID "]. DefaultValue = txtStudentId . Text;
SqlData Source 1 . InsertParameters [" StudentName "]. DefaultValue
=  txtStudentName . Text; SqlData Source 1 . InsertParameters [" AmountPaid "].
DefaultValue  =txtAmountPaid .Text;
SqlData Source 1 . InsertParameters [" PaymentDate "]. DefaultValue= txtPaymentDate
.Text;
SqlDataSource1 . InsertParameters [" Mode "]. DefaultValue = txtMode . Text;
SqlData  Source 1   . Insert   (); Grid View 1 . Data Bind (); // Refresh Grid View
}
protected void Grid View 1 _Selected IndexChanged ( object sender , EventArgs e)
{
Grid View Row row = Grid View 1 . Selected Row ; txtPaymentId . Text = row . Cells [1].
Text;
txtStudentId . Text = row . Cells [2]. Text; txtStudentName . Text = row . Cells [3]. Text;
txtAmountPaid . Text = row . Cells [4]. Text; txtPaymentDate . Text = row . Cells [5]. Text;
txtMode . Text = row . Cells [6]. Text;
}}}
```

Output:


Bhawarth Padwal | Payment Form
File | C:/Users/Bhawarth/Documents/tp/index.html
Payment Form
Payment ID:
1
Student ID:
1
Student Name:
Bhawarth
Amount Paid:
100
Payment Date:
25-04-2025
Mode:
UPI
Insert Record
Actions
PaymentId
StudentID
StudentName
AmountPaid
PaymentDate
Mode


Bhawarth Padwal | Payment Form
File | C:/Users/Bhawarth/Documents/tp/index.html
Payment Form
Payment ID:
Student ID:
Student Name:
Amount Paid:
Payment Date:
dd-mm-yyyy
Mode:
Insert Record
Actions
PaymentId
StudentID
StudentName
AmountPaid
PaymentDate
Mode
Edit Delete Select
1
1
Bhawarth
100.00
2025-04-25
UPI

MODULE 4: STATE MANAGEMENT AND AJAX

EXPERIMENT NO 1:

Aim: Create a web application demonstrating client-side state management using ViewState and Cookies.

StateManagement.aspx:

```
<%@ Page Language =" C#" Auto EventWireup =" true " Code Behind =" State Management
. aspx. cs"  Inherits =" _ 21 _State Management . State Management " % >
<! DOCTYPE  html >
<html  xmlns =" http :// www . w3 . org /1999/ xhtml" >
<head  runat =" server" >
<title >Client - Side  State  Management </ title >
<style >
label {
font - weight: bold ; display: block; margin - top : 10 px;
}
body {
font - family: Arial , sans - serif; margin : 20 px;
}
h1 , h2 {
color: #0078 D7 ;
}
</ style >
</ head >
<body >
<form  id =" form1 "  runat =" server" >
<div >
<h1 > Client - Side State Management with View State and Cookies
</ h1 >
<!-- View State Section -->
<h2 > Store Name using View State </ h2 >
<label for=" txtName View State " > Enter your name : </ label >
<asp : TextBox ID =" txtName View State " runat =" server" ></ asp : TextBox >
<asp : Button ID =" btn Save View State " runat =" server" Text =" Save to View State " On
Click =" btn Save View State_Click " / >
<br / ><br / >
<asp : Label ID =" lblView State " runat =" server" Text =" Your name from View State will
appear here ." ></ asp : Label >
```

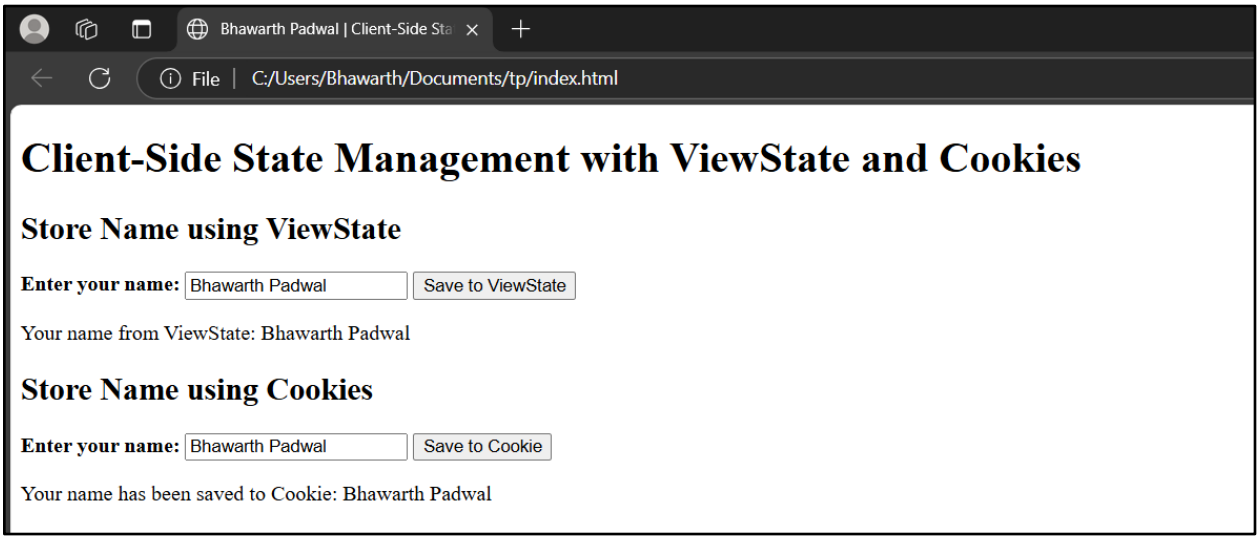
```
<br / ><br / >
 <!-- Cookies Section -->
<h2 > Store Name using Cookies </ h2 >
<label for=" txtName Cookie " > Enter your name : </ label >
<asp : TextBox  ID =" txtName Cookie "  runat =" server" ></ asp : TextBox>
<asp : Button ID =" btn Save Cookie " runat =" server" Text =" Save to Cookie " On Click ="
btn Save Cookie_Click " / >
<br / ><br / >
<asp : Label ID =" lblCookie " runat =" server" Text =" Your name from Cookies will appear
here ." ></ asp : Label >
</ div >
</ form >
</ body >
</ html >
```

StateManagement.aspx.cs:

```
using System ;
using System . Collections. Generic; using System . Linq ;
using System . Web ; using System . Web . UI;
using System . Web . UI. Web Controls;
namespace _ 21 _State Management {
public partial class State Management : System . Web . UI. Page{
protected void Page_Load ( object sender , EventArgs e){
if ( View State [" Name From View State "] != null){
lblView State . Text = " Your name from View State : " + View State [" Name From View State
"]. To String ();
}
if ( Request. Cookies [" Name From Cookie "] != null)
{
lblCookie . Text = " Your name from Cookie : " + Request.
Cookies [" Name From Cookie "]. Value ;
}}
protected void  btn Save View State_Click ( object  sender , EventArgs e) {
string  name From View State  =  txtName View State . Text; View State [" Name From View
State "] = name From View State ; lblView State . Text = " Your name has been saved to View
State :
" + name From View State ;
}
protected void  btn Save Cookie_Click ( object  sender , EventArgs e)
{
```

```
string  name From Cookie  =  txtName Cookie . Text; Http Cookie cookie = new Http Cookie
(" Name From Cookie ",
name From Cookie )
{
Expires = Date Time . Now . Add Days (1) // Cookie will expire in 1 day
};
Response . Cookies. Add ( cookie );
lblCookie . Text = " Your name has been saved to Cookie : " + name From Cookie ;
}}}
```

Output:



EXPERIMENT NO 2:

Aim: Design a session-based login system where users are redirected to a dashboard upon successful login.

User.cs:

```
public class User
{
public string Username { get; set; } public string Password { get; set; }
}
```

AccountController.cs:

```
using System . Collections. Generic; using System . Linq ;
using System . Web . Mvc;
namespace Session Login MVC . Controllers{
public class AccountController : Controller{
// Simulated user database
private List <User > users = new List <User >{
new User { Username = " Soumen ", Password = " password 123 " }
};
// GET: Login Page
public Action Result  Login (){
return  View ();
}
public Action Result Login ( string username , string password ){
var user = users. FirstOrDefault( u => u. Username == username && u. Password ==
password );
if ( user != null){
Session [" Username "] = username ;
return RedirectTo Action (" Dashboard ");
}
else
{
View Bag . ErrorMessage = " Invalid username or password ."; return View ();
}}
// GET: Dashboard Page
public Action Result Dashboard ()
{
if ( Session [" Username "] == null)
{
```

```
return  RedirectTo Action (" Login ");
}
View Bag . Username = Session [" Username "]. To String (); return View ();
}
//  Logout  method
public Action Result Logout ()
{
Session . Clear (); // Clear session return  RedirectTo Action (" Login ");
}}}
```

Login.cshtml:

```
@{
View Bag . Title = " Login ";
}
<h2 >Login </ h2 >
@if ( View Bag . ErrorMessage != null)
{
<p style =" color:  red ;" >@ View Bag . ErrorMessage </ p>
}
@ using ( Html. Begin Form (" Login ", " Account", Form Method . Post))
{
<div >
<label >Username : </ label > @ Html. TextBox(" username ")
</ div >
<div >
<label >Password : </ label > @ Html. Password (" password ")
</ div >

<div >
<button  type =" submit" >Login </ button >
</ div >
}
```

Dashboard.cshtml:

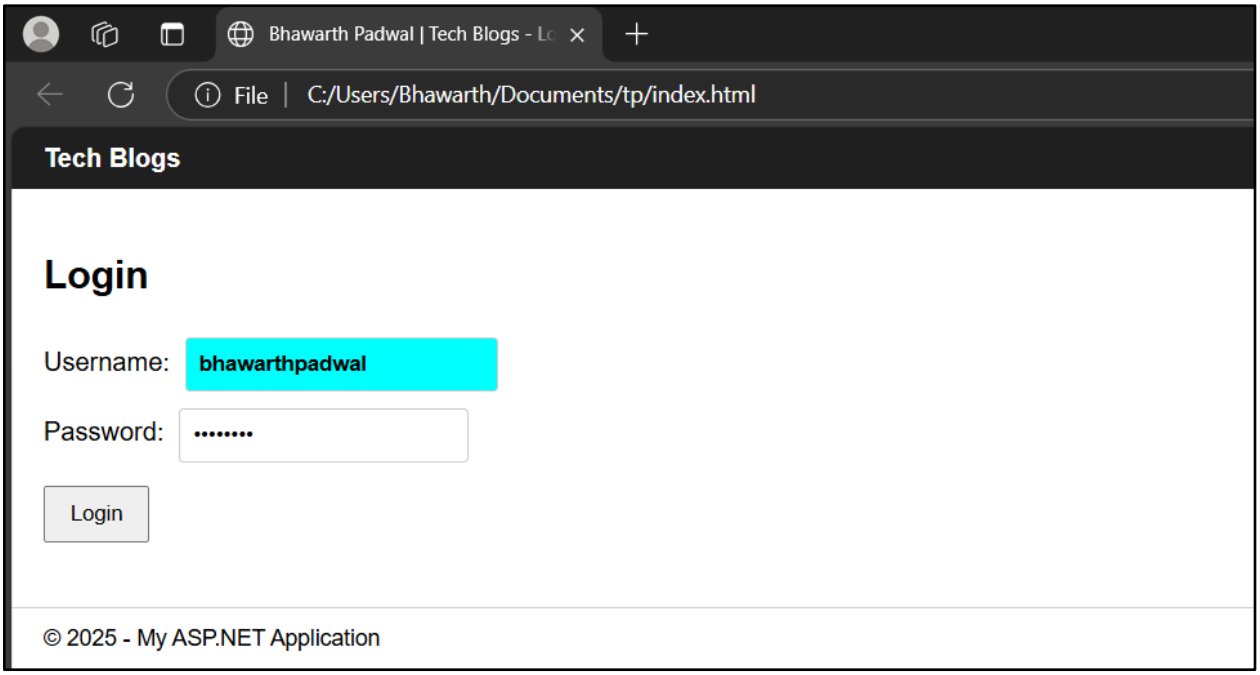
```
@{
View Bag . Title = " Dashboard ";
}
<h2 >Welcome ,  @ View Bag . Username ! </ h2 >
<p>This is your dashboard . </ p>
@ Html. Action Link (" Logout", " Logout", " Account")
```

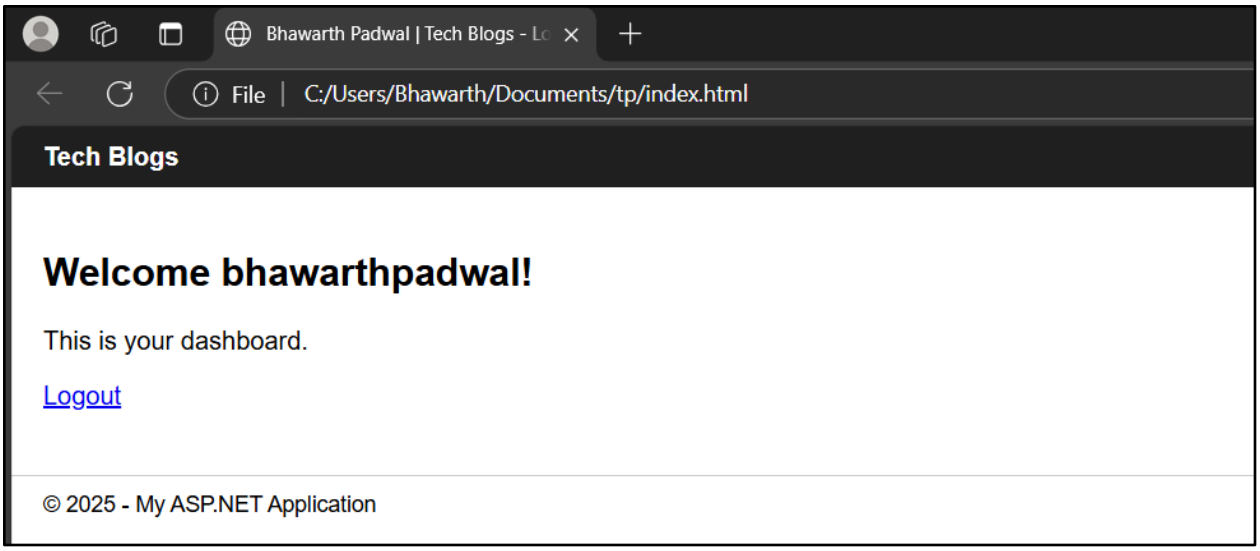
Web.config

```
< configuration >
<system . web >
< session State mode =" In Proc" timeout=" 20 " / >
</ system . web >
</ configuration >
```

Run the Application:

Output:




Bhawarth Padwal | Tech Blogs - Lo
File | C:/Users/Bhawarth/Documents/tp/index.html
Tech Blogs
Welcome bhawarthpadwal!
This is your dashboard.
Logout
© 2025 - My ASP.NET Application

EXPERIMENT NO 3:

Aim: Implement an AJAX-enabled web form to fetch user details without reloading the page.

Default.aspx:

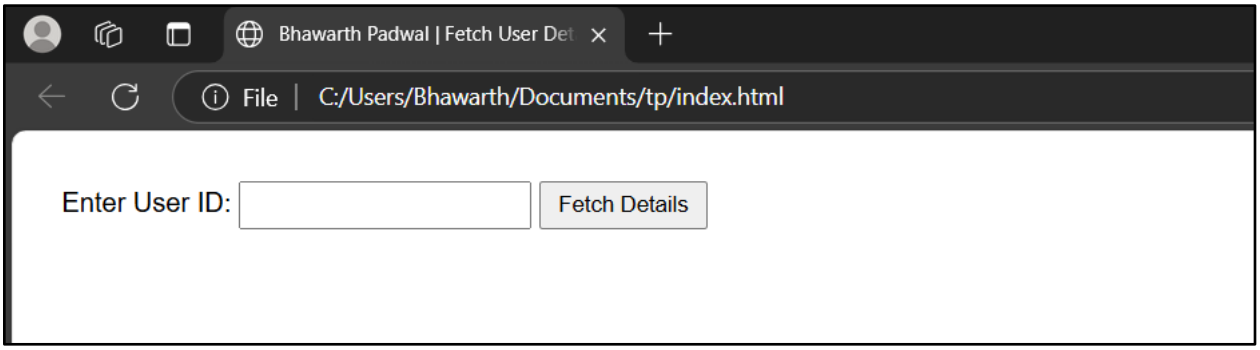
```
<%@ Page Language =" C#" Auto EventWireup =" true " Code Behind =" Default. aspx. cs"
Inherits =" _ 23 _AJAXenabled . Default" % >
<! DOCTYPE  html >
<html >
<head  runat =" server" >
<title > AJAX Fetch User Details </ title >
<script src=" https :// ajax. googleapis. com / ajax/ libs/ jquery /3.5.1/ jquery. min . js" ></
script >
<script type =" text/ javascript" > function fetch UserDetails () {
var userID = $ ("# userID "). val(); // Get the UserID value
$. ajax ({
type : " POST ",
url: " Default. aspx/ GetUserDetails", // Call Web Method data: JSON . stringify ({ userID :
userID }), contentType : " application / json ; charset=utf -8", dataType : " json ",
success: function ( response ) { if ( response . d) {
var user = response . d;
$ ("# userDetails "). html(
" Name : " + user. Name + " <br/ >" + " Email: " + user. Email + " <br/ >" + " Phone : " + user.
Phone
);}},
error: function ( error) { console . log( error);
}});}
</ script >
</ head >
<body >
<form  id =" form1 "  runat =" server" >
<div >
<label for=" userID " > Enter  User  ID : </ label >
<input type =" text" id =" userID " / >
<button type =" button " onclick =" fetch UserDetails ()" > Fetch Details </ button >
</ div >
<div  id =" userDetails" >
<!-- User details will be displayed here -->
</ div >
```

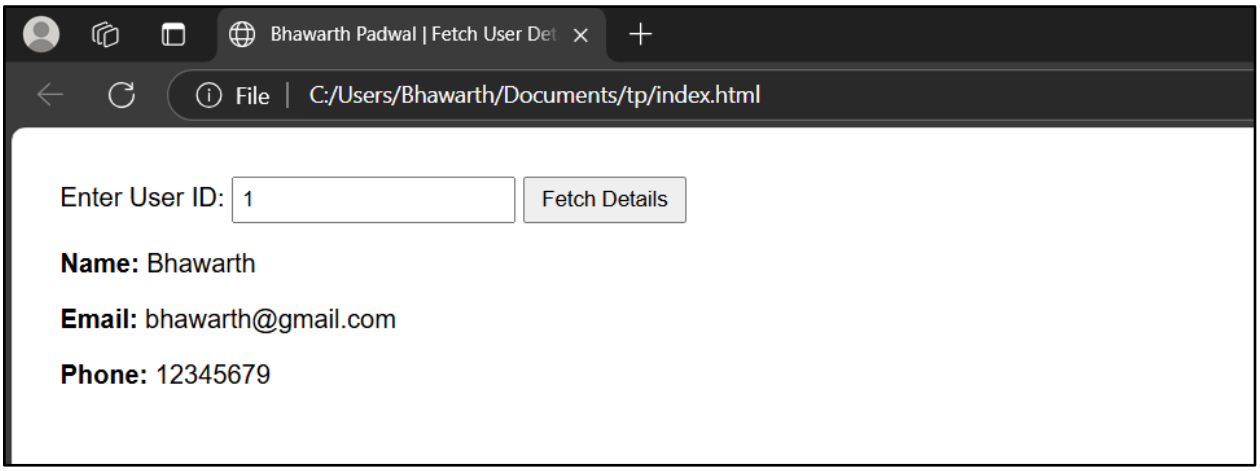
```
</ form >
</ body >
</ html >
```

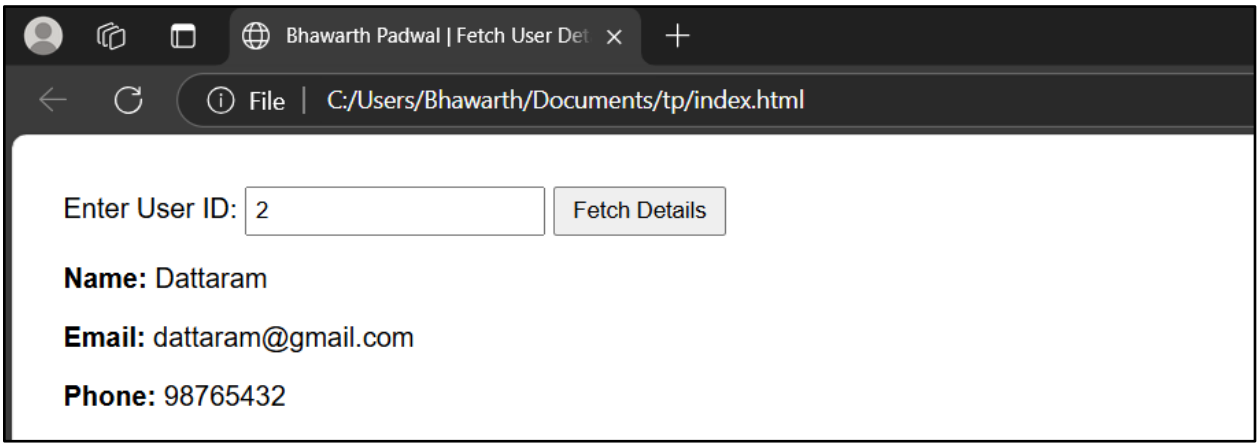
Default.aspx.cs:

```
using System ;
using System . Collections. Generic; using System . Linq ;
using System . Web ;
using System . Web . Services; using System . Web . UI;
using System . Web . UI. Web Controls;
namespace  _ 23 _AJAXenabled{
public partial class Default : Page{
public class User{
public string Name { get; set; } public string Email { get; set; } public string Phone { get; set;
}}
private User GetUserDetailsBy Id ( int userID )
{
if ( userID == 1){
return new User{
Name = " Dattaram",
Email = " Dattaran=m@ example . com ", Phone = "123 -456 -7890"
};}
else if ( userID == 2){
return new User
{
Name = " Bhawarth Padwal",
Email = " bhawarthpadwal@ example . com ", Phone = "987 -654 -3210"
};}
return null; }
[ Web Method ]
public static object GetUserDetails ( string userID ){
int id = Convert. To Int32 ( userID ); Default page = new Default ();
var user = page . GetUserDetailsBy Id ( id);
return user ?? null; // Return user details or null
}}}
```

Output:


Bhawarth Padwal | Fetch User Det
File | C:/Users/Bhawarth/Documents/tp/index.html
Enter User ID:
Fetch Details


Bhawarth Padwal | Fetch User Det
File | C:/Users/Bhawarth/Documents/tp/index.html
Enter User ID: 1
Fetch Details
Name: Bhawarth
Email: bhawarth@gmail.com
Phone: 12345679


Bhawarth Padwal | Fetch User Det
File | C:/Users/Bhawarth/Documents/tp/index.html
Enter User ID: 2
Fetch Details
Name: Dattaram
Email: dattaram@gmail.com
Phone: 98765432

EXPERIMENT NO 4:

Aim: Develop a simple to-do list application using AJAX controls and update panels.

Default.aspx:

```
<%@ Page Language =" C#" Auto EventWireup =" true " Code Behind =" Default. aspx. cs"
Inherits =" _ 24 _AJAXTo Do List . Default" % >
<! DOCTYPE  html >
<html  xmlns =" http :// www . w3 . org /1999/ xhtml" >
<head  runat =" server" >
<title >To - Do List </ title >
<style >
. todo - item {
padding : 5 px;
margin - bottom : 5 px; background - color: # f1f1f1 ;
}
</ style >
<script type =" text/ javascript" > function remove Task ( task) {
Page Methods. Remove Task ( task , function () {
__do PostBack (' <%= Update Panel1 . ClientID % >', '');
});
}
</ script >
</ head >
<body >
<form  id =" form1 "  runat =" server" >
<div >
<h2 >To - Do  List </ h2 >
<!-- Input Section -->
<div >
<label for=" taskInput" > New Task: </ label >
<input type =" text" id =" taskInput" runat =" server" / >
<asp : Button ID =" btn Add Task " runat =" server" Text =" Add Task" On Click =" btn Add
Task_Click " / >
</ div >
<!-- ScriptManager is required for AJAX -->
<asp : ScriptManager ID =" ScriptManager1 " runat =" server" Enable Page Methods =" true
" / >
<!-- Update Panel -->
 <asp : Update Panel  ID =" Update Panel1 "  runat =" server" >
```

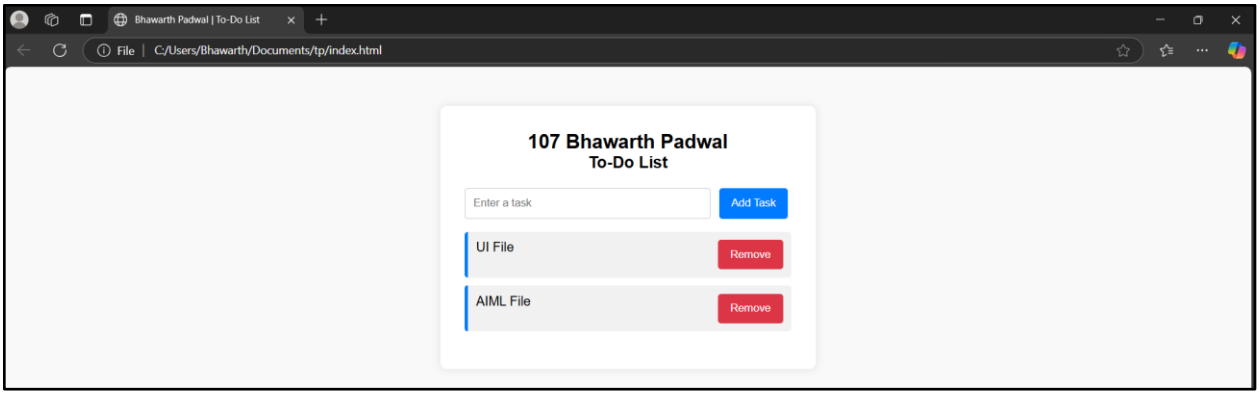
```
<ContentTemplate >
<div id =" taskList" runat =" server" >
<%-- Tasks will appear here --% >
</ div >
</ ContentTemplate >
</ asp : UpdatePanel >
</ div >
</ form >
</ body >
</ html >
```

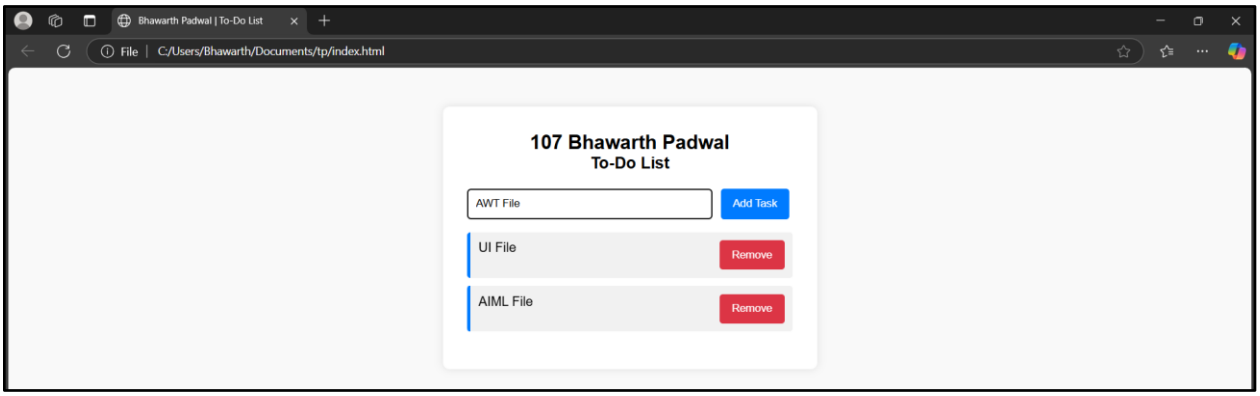
Default.aspx.cs:

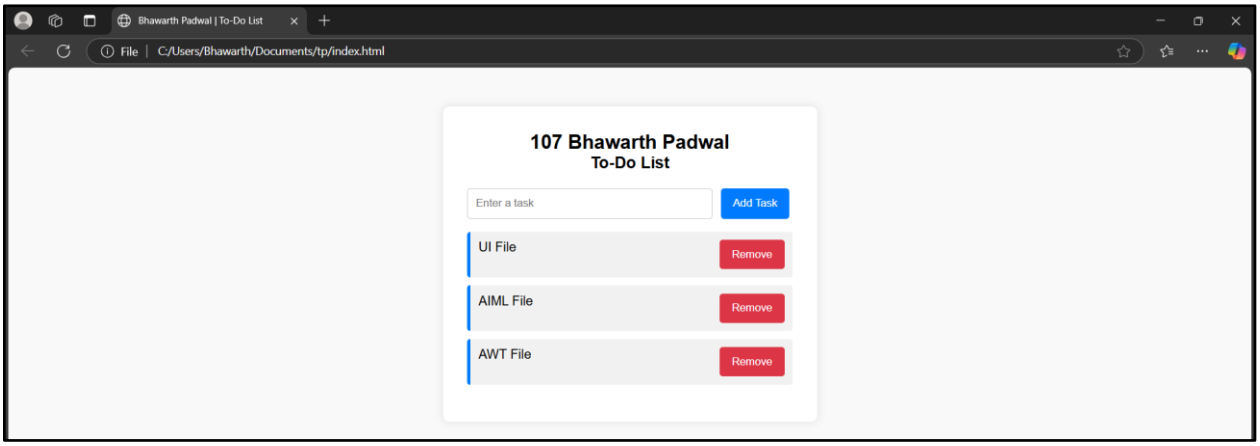
```
using System ;
using System . Collections. Generic; using System . Web . Services;

namespace  _ 24 _AJAXTo Do List{
public partial class Default : System . Web . UI. Page{
private static List <string > tasks = new List <string >();
protected void Page_Load ( object sender , EventArgs e){
if  (! IsPostBack ){
DisplayTasks ();
}}
protected void  btn Add Task_Click ( object sender , EventArgs e){
string new Task = taskInput. Value ; if (! string . IsNullOrEmpty ( new Task )){
tasks. Add ( new Task );
taskInput. Value = ""; // Clear input field DisplayTasks ();}
}
private void  DisplayTasks (){
taskList. InnerHtml = ""; // Clear old list foreach ( string task in tasks) {
string escaped Task = task. Replace ("\"" , "\\\""); string taskHtml = $@"
<div  class=' todo - item ' >
<span >{ task } </ span >
<button type =' button ' onclick =' remove Task (""{ escaped Task }"") '>Remove </ button
>
</ div >";
taskList. InnerHtml += taskHtml;
}
}
[ Web Method ]
public static void Remove Task ( string task)
```

```
{
tasks. Remove ( task);
}}}
```

Output:







EXPERIMENT NO 5:

Aim: Demonstrate the use of the Global.asax file for handling application-level events.

Global.aspx.cs:

```
using System ;
using System . Collections. Generic; using System . Linq ;
using System . Web ;
using System . Web . Optimization ; using System . Web . Routing ; using System . Web .
Security ; using System . Web . Session State ;
namespace  _ 25 _GlobalAsax{
public class Global : Http Application{
void Application_Start ( object sender , EventArgs e)
{
System . Diagnostics. Debug . Write Line ("Application Started ");
}
void Session_Start( object sender , EventArgs e)
{
System . Diagnostics. Debug . Write Line ("Session Started ");
}
void Application_Error ( object sender , EventArgs e){
Exception ex = Server. GetLastError ();
System . Diagnostics. Debug . Write Line ("Application Error: "+ ex. Message );
Server. ClearError ();
}
void Session_End ( object sender , EventArgs e){
System . Diagnostics. Debug . Write Line ("Session Ended ");
 }
void Application_End ( object sender , EventArgs e){
System . Diagnostics. Debug . Write Line ("Application Ended ");
}}}
```

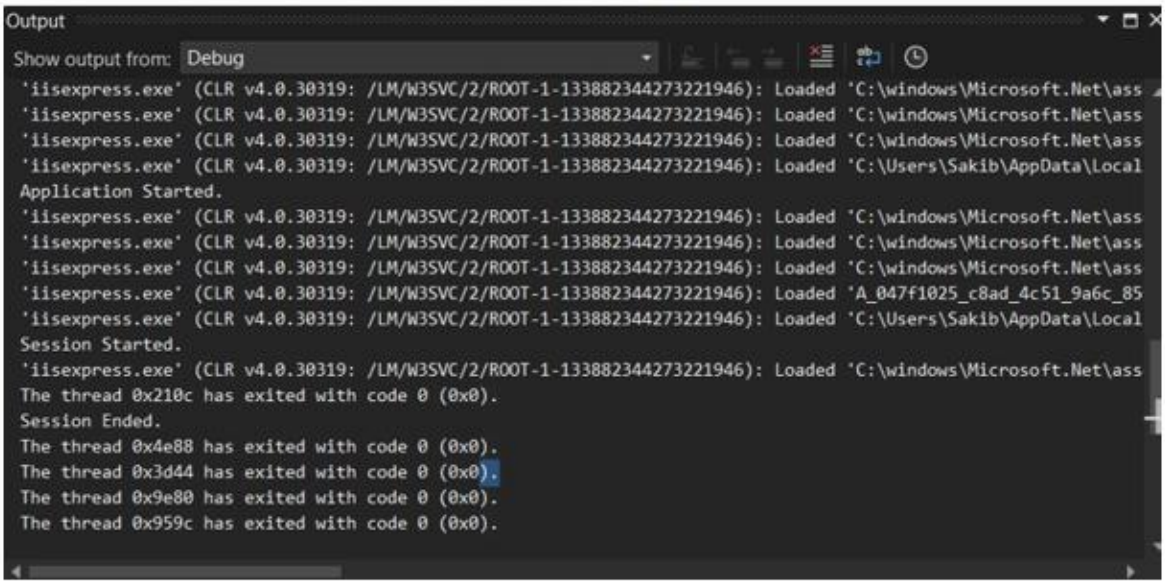
Set the session timeout to something short in Web.config:

```
<system . web >
< session State timeout ="1" / > <!-- 1 minute timeout -->
</ system . web >
```

Output:

107 Bhawarth Padwal

Global.asax Event Handling



MODULE 5: WEB SERVICE AND WCF

EXPERIMENT NO 1:

Aim: Create an XML-based web service to fetch product details and consume it in a web application.

PART 1: Create the XML-Based SOAP Web Service (ASMX)

ProductService.asmx.cs:

```
using System ;
using System . Collections. Generic; using System . Linq ;
using System . Web ;
using System . Web . Services; using System . Xml;
namespace  _ 26 _ProductWeb Service{
[ Web Service ( Namespace = " http :// www . example . com / ProductService ")] [ Web
Service Binding ( ConformsTo = WsiProfiles. BasicProfile 1 _ 1 )] public class
ProductService : Web Service{
[ Web Method ]
public XmlNode GetProductDetails ( int productId ){
XmlDocument  xmlDoc  = new  XmlDocument ();
XmlElement root = xmlDoc. Create Element(" ProductDetails");
xmlDoc. Append Child ( root);
XmlElement product = xmlDoc. Create Element(" Product"); root. Append Child ( product);
XmlElement id = xmlDoc. Create Element(" ProductId "); id. InnerText  =  productId . To
String (); product. Append Child ( id);
XmlElement name = xmlDoc. Create Element(" ProductName "); name . InnerText  =  "
Product  "  +  productId ; product. Append Child ( name );
XmlElement price = xmlDoc. Create Element(" Price "); price . InnerText = " $" + ( productId
* 10); product. Append Child ( price );
return xmlDoc. DocumentElement ;
}}}
```

Run and Test Web Service

PART 2: Create ASP.NET Web Forms App to Consume the Service Create the Web Application

Add Service Reference Create ProductDetails.aspx

ProductDetails.aspx:

```
<%@ Page Language =" C#" Auto EventWireup =" true " Code Behind =" ProductDetails
. aspx. cs" Inherits=" _ 26 _ProductWeb App . ProductDetails " % >
```

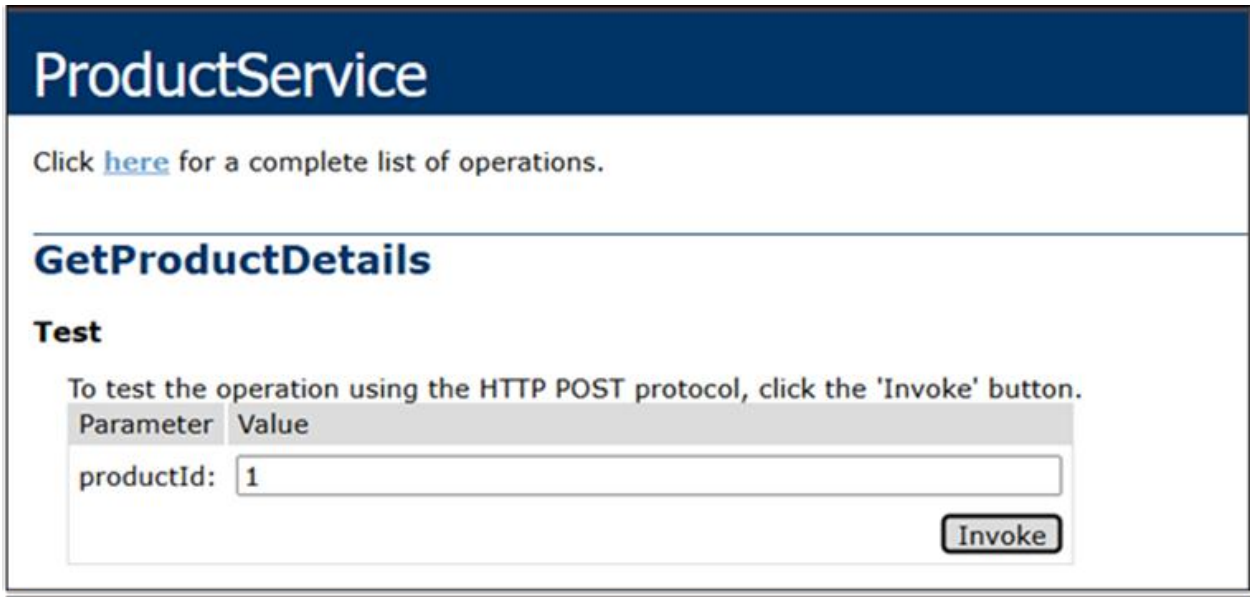
```
<! DOCTYPE  html >
<html  xmlns=" http :// www . w3 . org /1999/ xhtml" >
<head  runat=" server" >
<title >Product  Details </ title >
</ head >
<body >
<form  id=" form1 "  runat=" server" >
<div >
<h2 >Product  Details </ h2 >
<asp : Label ID=" lblProductDetails " runat=" server" Text=" Enter Product ID:" / >
<asp : TextBox  ID=" txtProductId "  runat=" server"  / >
<asp : Button ID=" btn Fetch Product " runat=" server" Text=" Fetch Product" On Click ="
btn Fetch Product_Click " / >
<br / ><br / >
<asp : Grid View ID=" gvProductDetails " runat=" server" Auto Generate Columns =" True "
/ >
</ div >
</ form >
</ body >
</ html >
```

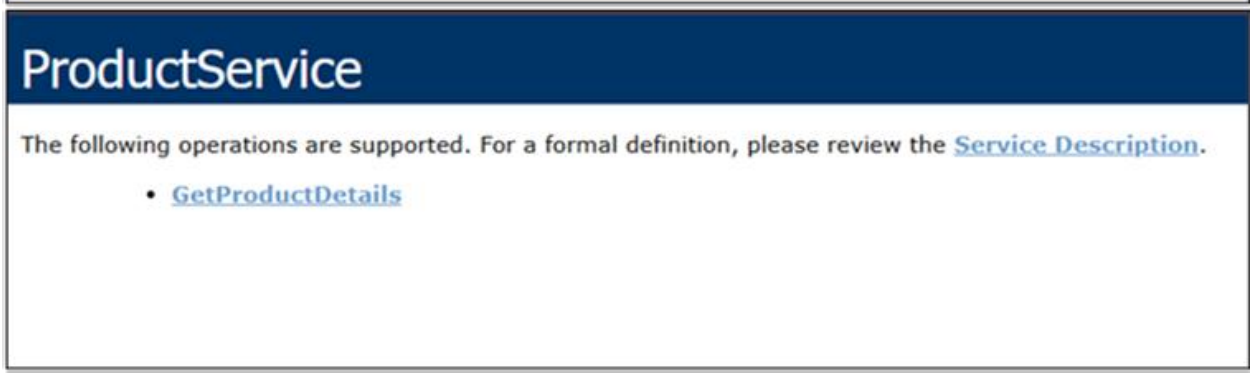
ProductDetails.aspx.cs:

```
using System ;
using System . Collections. Generic; using System . Data;
using System . Linq ; using System . Web ;
using System . Web . UI;
using System . Web . UI. Web Controls;
using _ 26 _ProductWeb App . ProductService ; using System . Xml;
using System . Xml. Linq ;
namespace  _ 26 _ProductWeb App{
public partial class ProductDetails : System . Web . UI. Page{
protected void Page_Load ( object sender , EventArgs e){
}
protected  void  btn Fetch Product_Click ( object  sender ,  EventArgs  e){
int productId ;
if (! int. TryParse ( txtProductId . Text ,  out  productId )){
return return ;
}
var service = new ProductService . ProductService SoapClient ();
 XElement productXml = service . GetProductDetails ( productId );
```

```
DataSet ds = new DataSet ();
using ( var reader = productXml. Create Reader ()){
ds. Read Xml( reader);
gvProductDetails . Data Source = ds;
gvProductDetails . DataBind();
}}}}
```

Output:






```
<ProductDetails>
  <Author> Bhawarth Padwal </Author>
  <Product>
    <ProductId> 1 </ProductId>
    <ProductName> Padwal Product </ProductName>
    <Price> $10 </Price>
  </Product>
</ProductDetails>
```

EXPERIMENT NO 2:

Aim: Develop a database-driven WCF service to fetch and update user details.

Create the Database

```
CREATE DATABASE UserDB ; USE UserDB ;
CREATE TABLE Users (
UserId INT PRIMARY KEY IDENTITY ,
FirstName NVARCHAR (50) , LastName NVARCHAR (50) , Email NVARCHAR (100) ,
```

```
Age INT
);

-- Insert dummy data
INSERT INTO Users ( FirstName , LastName , Email , Age ) VALUES (' Alice ', ' Smith ', '
alice@ example . com ', 25),
(' Bob ', ' Johnson ', ' bob@ example . com ', 30);
```

Interface: IService1.cs:

```
using  System . Service Model;

[ Service Contract ]
public interface IUserService
{
[ Operation Contract ]
User GetUserDetails ( int userId );

[ Operation Contract ]
bool Update UserDetails ( User user);
}
```

svc.cs:

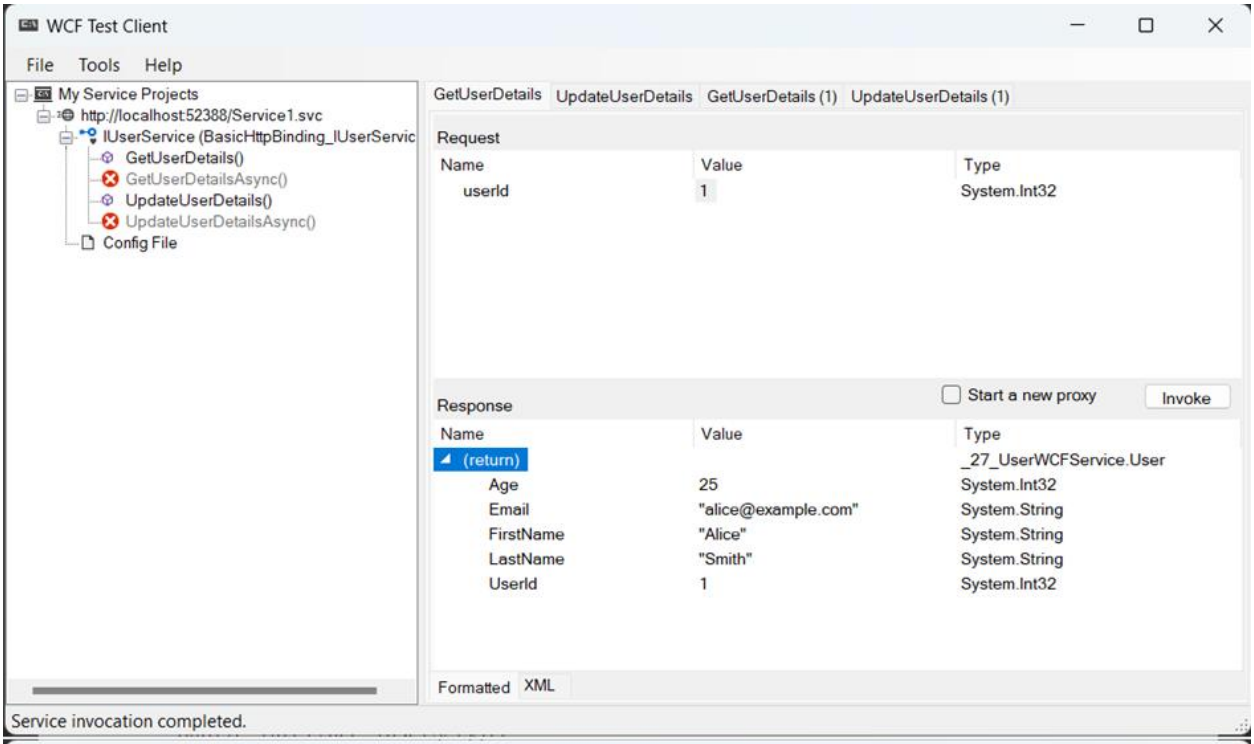
```
public class UserService : IUserService {
private UserDBEntities db = new UserDBEntities ();
public User GetUserDetails( int userId ){
var userEntity = db. Users. SingleOrDefault ( u => u. UserId == userId );
if ( userEntity != null){
return new User {
UserId = userEntity . UserId , FirstName = userEntity . FirstName , LastName = userEntity .
LastName , Email = userEntity . Email ,
Age = userEntity . Age
};}
return null;
}
public bool  Update UserDetails ( User user){
var existing = db. Users. SingleOrDefault ( u => u. UserId == user.
UserId );
if ( existing != null){
existing . FirstName = user. FirstName ; existing . LastName = user. LastName ;
existing . Email = user. Email;
```

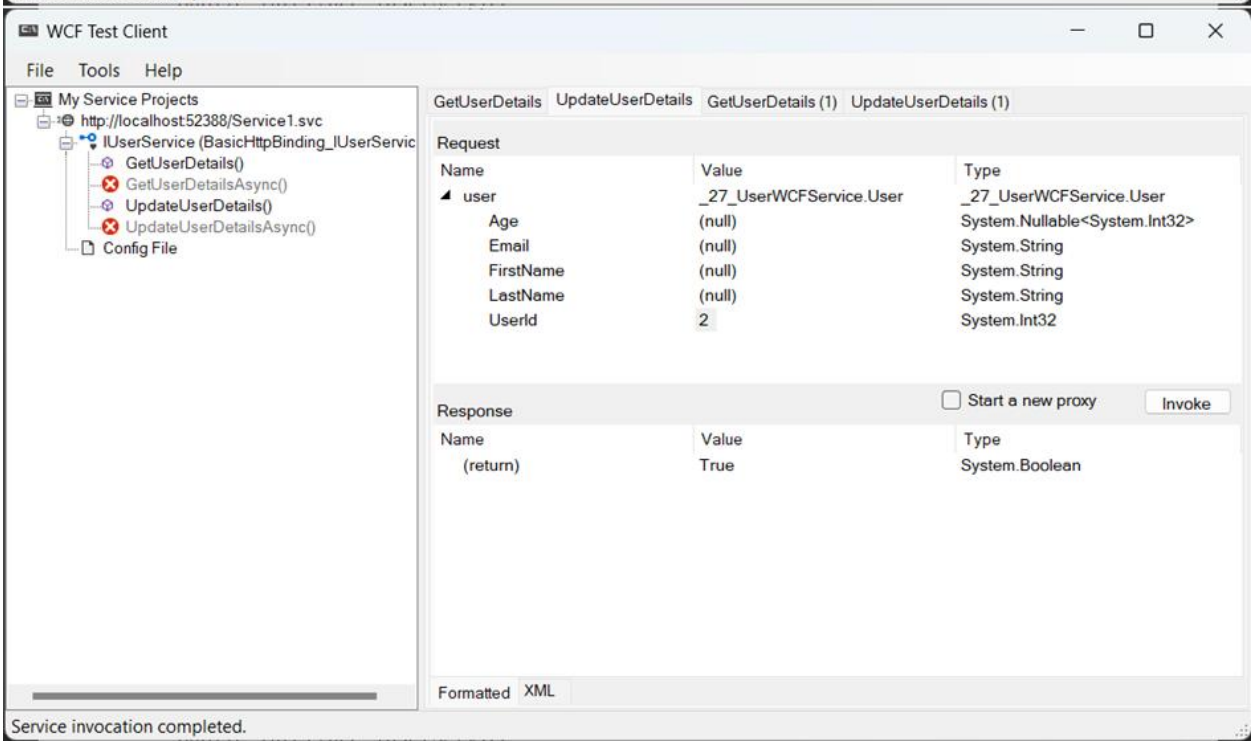
```
existing . Age=user. Age; db. Save Changes ();
return true ;
}
return false ;
}}
```

Web.config:

```
<system . service Model >
<services >
<service  name =" UserWCFService . UserService " >
<endpoint address=""
binding  ="  wsHttp  Binding  " contract=" UserWCFService . IUserService " / >
<host >
< base Addresses >
<add  base Address=" http: // localhost:8000 / UserService " / >
</ base Addresses >
</ host >
</ service >
</ services >
<behaviors >
< service Behaviors >
<behavior >
< service Metadata  http GetEnabled =" true "  / >
< service Debug  include Exception DetailIn Faults =" true "  / >
</ behavior >
</ service Behaviors >
</ behaviors >
< service Hosting Environment multiple Site BindingsEnabled =" true " / >
</ system . service Model >
```

Output:


WCF Test Client
File Tools Help
My Service Projects
http://localhost:52388/Service1.svc
IUserService (BasicHttpBinding_IUserServic
GetUserDetails()
GetUserDetailsAsync()
UpdateUserDetails()
UpdateUserDetailsAsync()
Config File
GetUserDetails UpdateUserDetails GetUserDetails (1) UpdateUserDetails (1)
Request
Name Value Type
userId 1 System.Int32
Start a new proxy
Invoke
Response
Name Value Type
(return) _27_UserWCFService.User
Age 25 System.Int32
Email "alice@example.com" System.String
FirstName "Alice" System.String
LastName "Smith" System.String
UserId 1 System.Int32
Formatted XML
Service invocation completed.


WCF Test Client
File Tools Help
My Service Projects
http://localhost:52388/Service1.svc
IUserService (BasicHttpBinding_IUserServic
GetUserDetails()
GetUserDetailsAsync()
UpdateUserDetails()
UpdateUserDetailsAsync()
Config File
GetUserDetails UpdateUserDetails GetUserDetails (1) UpdateUserDetails (1)
Request
Name Value Type
user _27_UserWCFService.User _27_UserWCFService.User
Age (null) System.Nullable<System.Int32>
Email (null) System.String
FirstName (null) System.String
LastName (null) System.String
UserId 2 System.Int32
Start a new proxy
Invoke
Response
Name Value Type
(return) True System.Boolean
Formatted XML
Service invocation completed.

EXPERIMENT NO 3:

Aim: Create xml based web service to create a calculator and consume it in a website.

CalculatorController.cs:

```
using System . Net;
using System . Net. Http ;
using System . Web . Http ;
namespace CalculatorWeb Service . Controllers{
[ Route Prefix (" api/ calculator ")]
public class CalculatorController : ApiController
{
[  Http  Get] [ Route (" add ")]
public Http Response Message Add ( int a, int b)
{
var  result  = a + b;
return Request. Create Response ( Http StatusCode . OK , new { result });
}
[    Http   Get] [ Route (" subtract ")]
public Http Response Message Subtract( int a, int b)
{
var result = a - b;
return Request. Create Response ( Http StatusCode . OK , new { result });
}
[    Http   Get] [ Route (" multiply ")]
public Http Response Message Multiply ( int a, int b){
var  result  = a * b;
return Request. Create Response ( Http StatusCode . OK , new { result });
}
[   Http  Get] [ Route (" divide ")]
public Http Response Message Divide ( int a, int b)
{
if ( b  ==  0)
return  Request. Create ErrorResponse ( Http StatusCode .Bad Request , " Cannot divide by
zero .");
var  result = a / b;
return Request. Create Response ( Http StatusCode . OK , new { result });
}}}
```

Web.config:

```
<add name =" Api" path =" api /*" verb ="*" type =" System . Web . Http .
Dispatcher. Http ControllerDispatcher , System .
Web . Http " resource Type =" Unspecified " / >
```

Run the Service

- Test the API in browser: http://localhost:yourport/api/calculator/add?a=5&b=3

Create Website to Consume the Service

Default.aspx:

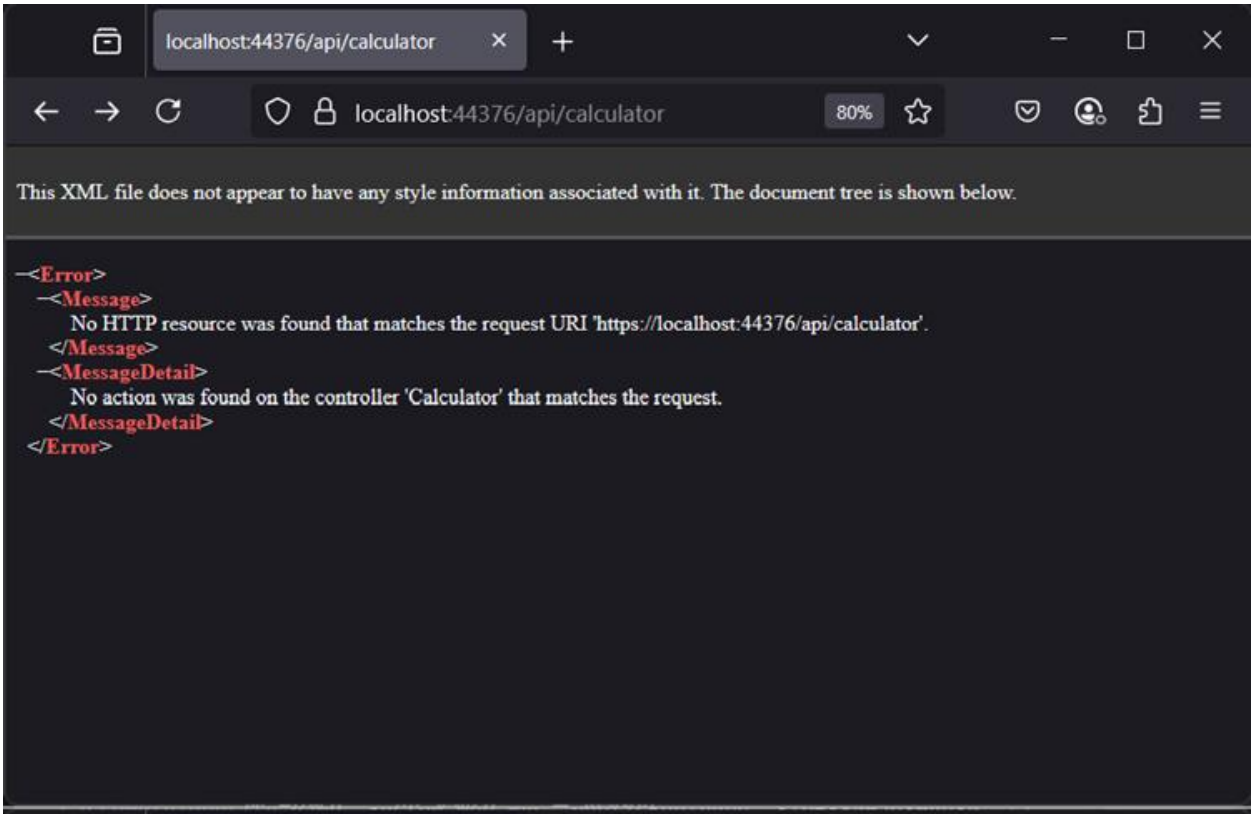
```
<asp : Button ID=" btn Calculate " runat=" server" Text=" Calculate " On Click =" btn
Calculate_Click " / >
<asp : TextBox ID=" txtResult" runat=" server" / >
```

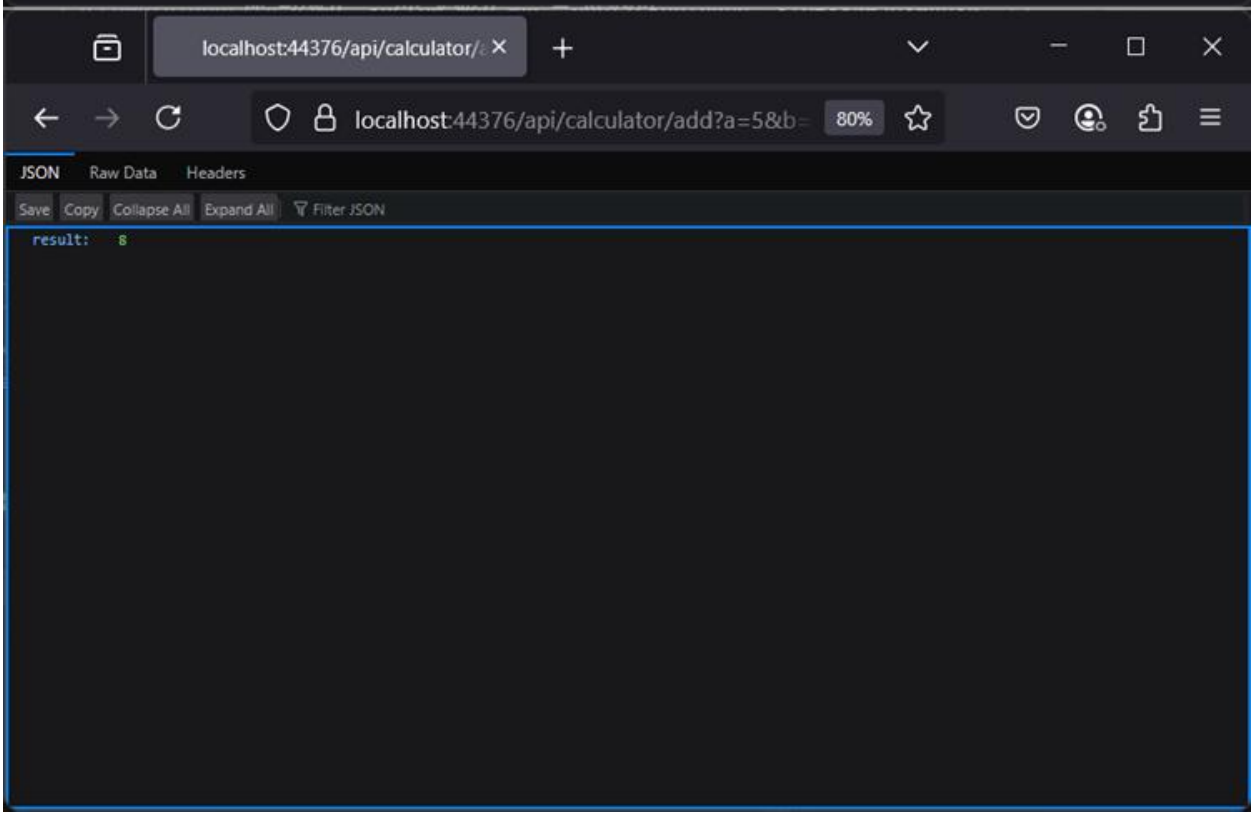
Default.aspx.cs:

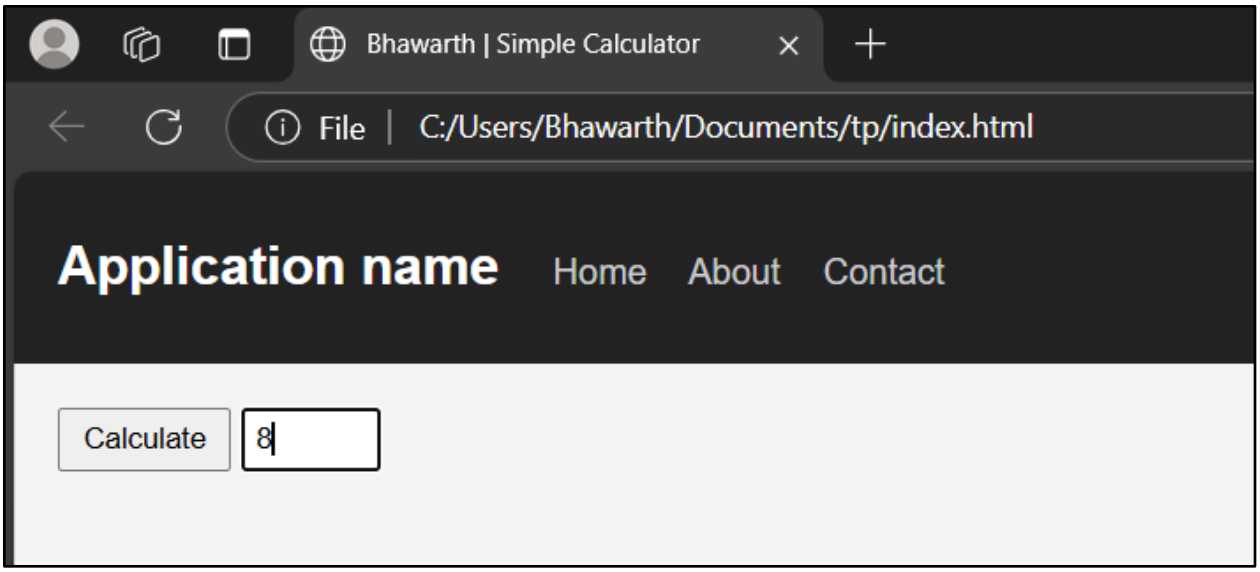
```
using System ;
using System . Net. Http ;
using System . Threading . Tasks;
public partial class _Default : System . Web . UI. Page{
protected async void btn Calculate_Click ( object sender , EventArgs e){
int a = 5; int b = 3;
using ( Http Client client = new Http Client ()){
client. Base Address = new Uri(" http :// localhost: YOURPORT / api/ calculator /");
Http Response Message response = await client. GetAsync( $" add ? a
={ a}& b={ b }");
if ( response . IsSuccessStatusCode )
{
dynamic result = await response . Content. Read AsAsync < dynamic >();
txtResult. Text = result. result. To String ();

}else{
txtResult. Text = " Error: " + response . StatusCode ;
}}}}
```

- Replace "http://localhost:YOURPORT" with the actual port from your Web API project (check browser when it runs).

Output:


localhost:44376/api/calculator
localhost:44376/api/calculator
80%
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<Error>
<Message>
No HTTP resource was found that matches the request URI 'https://localhost:44376/api/calculator'.
</Message>
<MessageDetail>
No action was found on the controller 'Calculator' that matches the request.
</MessageDetail>
</Error>


localhost:44376/api/calculator/
localhost:44376/api/calculator/add?a=5&b=
80%
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON
result: 8


Bhawarth | Simple Calculator
File
C:/Users/Bhawarth/Documents/tp/index.html
Application name
Home
About
Contact
Calculate
8

EXPERIMENT NO 4:

Aim: Create and consume the WCF service to calculate simple interest.

PART 1: Create the WCF Service Application

IService1.cs:

```
using  System . Service Model;
[ Service Contract ]
public interface IService1
{
[ Operation Contract ]
double Calculate Simple Interest ( double principal , double rate , double time );
}
```

Service1.svc.cs:

```
public class Service1 : IService 1{
public double Calculate Simple Interest ( double principal , double rate , double time ){
return ( principal * rate * time ) / 100;
}}
```

Web.config:

```
<? xml version =" 1.0 "? >
< configuration >
<system . web >
<compilation  debug =" true "  targetFramework =" 4.7.2 "  / >
</ system . web >

<system . service Model >
<services >
<service  name =" Simple InterestWCFService . Service1 " >
<endpoint address="" binding =" basicHttp Binding " contract=" Simple
InterestWCFService . IService1 " / >
<host >
< base Addresses >
<add  base Address=" http: // localhost:8000 / Service1 "  / >
</ base Addresses >
</ host >
</ service >
```

```
</ services >
<behaviors >
< service Behaviors >
<behavior >
< service Metadata  http GetEnabled =" true "  / >
< service Debug  include Exception DetailIn Faults =" true "  / >
</ behavior >
</ service Behaviors >
</ behaviors >
< service Hosting Environment multiple Site BindingsEnabled =" true " / >
</ system . service Model >
<system . web Server >
<modules  run AllManaged ModulesForAllRequests =" true "  / >
<handlers >
<add name =" svc - Integrated " path ="*. svc" verb ="*" type =" System . Service Model.
Activation .
Service Http HandlerFactory "
resource Type =" Unspecified " pre Condition =" integrated Mode , runtime Versionv4 .0 "/
>
</ handlers >
</ system . web Server >
</ configuration >
```

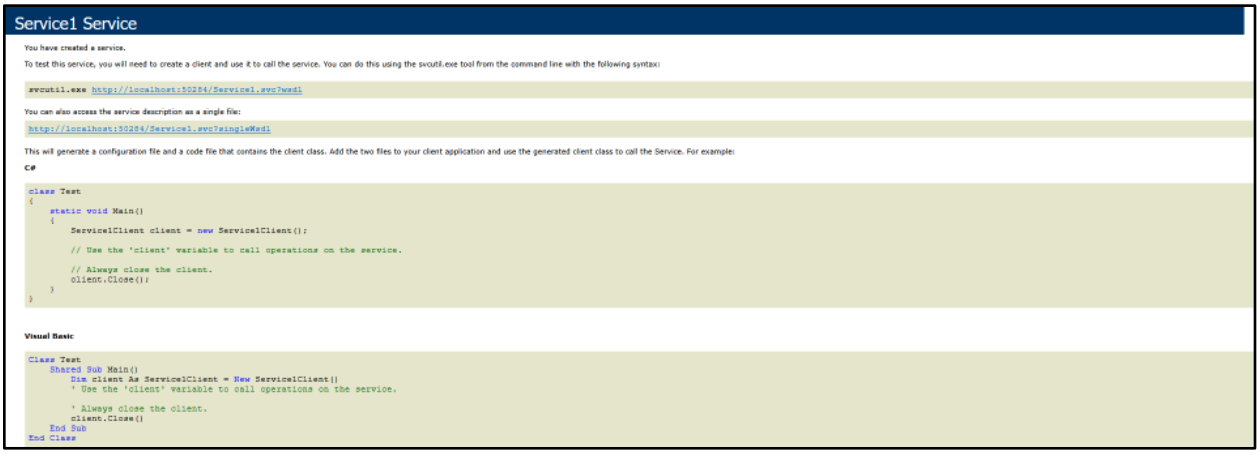
Run the WCF Service
PART 2: Create Console Client to Consume the Service Create Console App
Add Service Reference

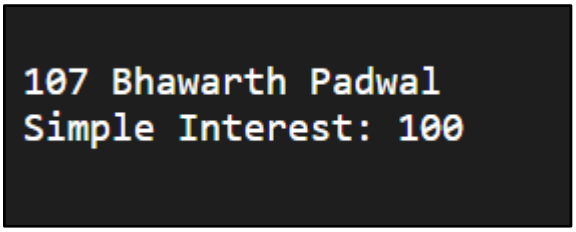
Program.cs:

```
using System ;
using  Simple InterestClient . Simple InterestWCFService ;

namespace  Simple InterestClient{
class  Program{
static void Main ( string [] args){
//  Create  proxy
Service 1 Client client = new Service 1 Client ();
//  Input
double principal = 1000.0; double rate = 5.0;
double time = 2.0;
//  Call  service
```

```
double interest = client. Calculate Simple Interest ( principal , rate , time );
//  Output
Console . Write Line ( $" Principal: { principal}, Rate : { rate }, Time : { time }");
Console . Write Line ( $" Simple Interest: { interest}");
client. Close ();
Console.Read Line ();
}}}
```

Output:





MODULE 6: ASP.NET CORE MVC FRAMEWORK

EXPERIMENT NO 1:

Aim: Create an simple ASP.NET Core MVC application with models, views, and controllers to display book info

Book.cs:

```
namespace  BookStore App . Models{
public class Book{
public int Id { get; set; }
public string Title { get; set; }
public string Author { get; set; }
public string Genre { get; set; }
public int Year { get; set; }
}}
```

BooksController.cs:

```
using Microsoft. Asp NetCore . Mvc; using BookStore App . Models;
using  System . Collections. Generic;
namespace   BookStore App . Controllers{
public class BooksController : Controller{
public static List <Book > Books = new List <Book >{
new Book { Id = 1 , Title = " The Catcher in the Rye", Author
= " J. D. Salinger", Genre = " Fiction ", Year = 1951 },
new Book { Id = 2 , Title = " To Kill a Mockingbird ", Author = " Harper Lee", Genre = "
Fiction ", Year = 1960 },
new Book { Id = 3 , Title = " 1984 ", Author = " George Orwell", Genre = " Dystopian ", Year =
1949 },
new Book { Id = 4 , Title = " Moby Dick", Author = " Herman Melville ", Genre = " Adventure
", Year = 1851 }
};
public  IAction Result  Index (){
return View ( Books);}
public IAction Result Details( int id){
var book = Books. Find ( b => b. Id == id); if ( book == null){
return  NotFound ();}
return View ( book);
}}}
```

Index.cshtml:

```
@ model List < BookStore App . Models. Book > @{
View Data [" Title "] = " Books List";}
<h1 >@ View Data [" Title "] </ h1 >
<table  class=" table " >
<thead >
<tr>
<th >Title </ th >
<th >Author </ th >
<th >Genre </ th >
<th >Year </ th >
<th >Actions </ th >
</ tr>
</ thead >
<tbody >
@ foreach ( var book in Model)
{

<tr>
<td >@ book. Title </ td >
<td >@ book. Author </ td >
<td >@ book. Genre </ td >
<td >@ book. Year </ td >
<td >
<a href="@Url. Action (" Details", " Books", new { id = book. Id })" class=" btn btn - info "
>Details </ a>
</ td >
</ tr>
}
</ body >
</ table >
```

Details.cshtml:

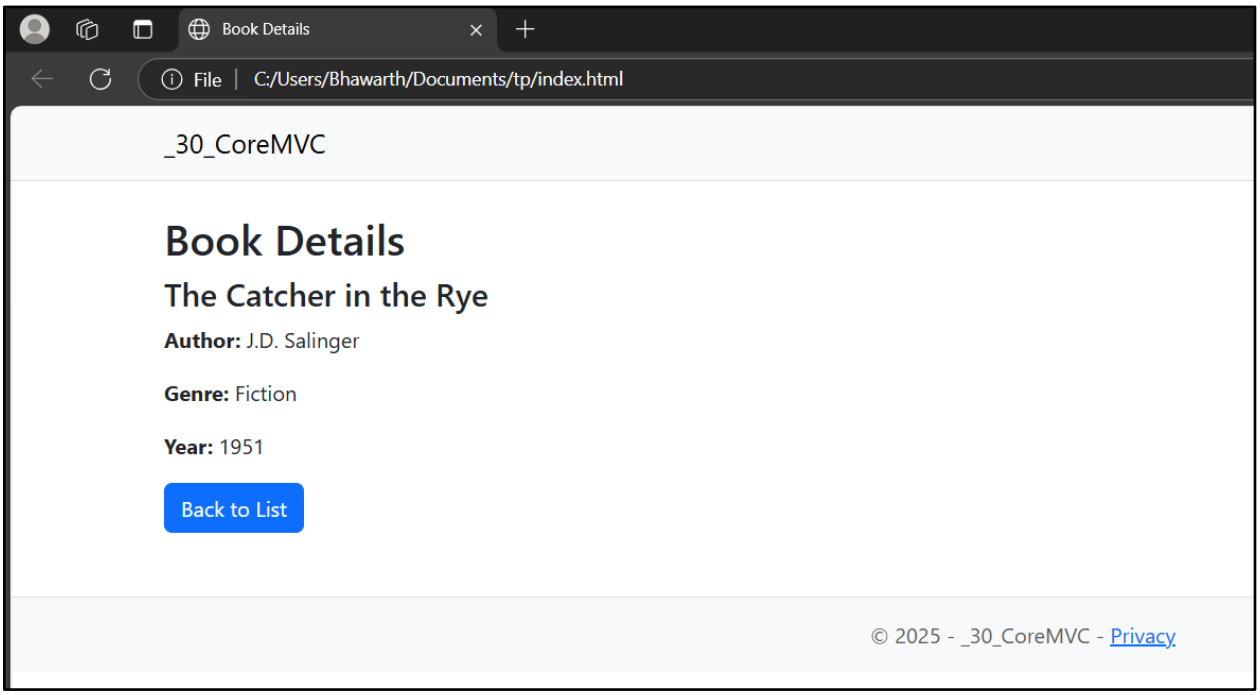
```
@ model BookStore App . Models. Book @{
View Data [" Title "] = " Book Details";}
<h1 >@ View Data [" Title "] </ h1 >
<div >
<h3 >@ Model. Title </ h3 >
<p><strong >Author: </ strong >  @ Model. Author </ p>
<p><strong >Genre : </ strong >  @ Model. Genre </ p>
```

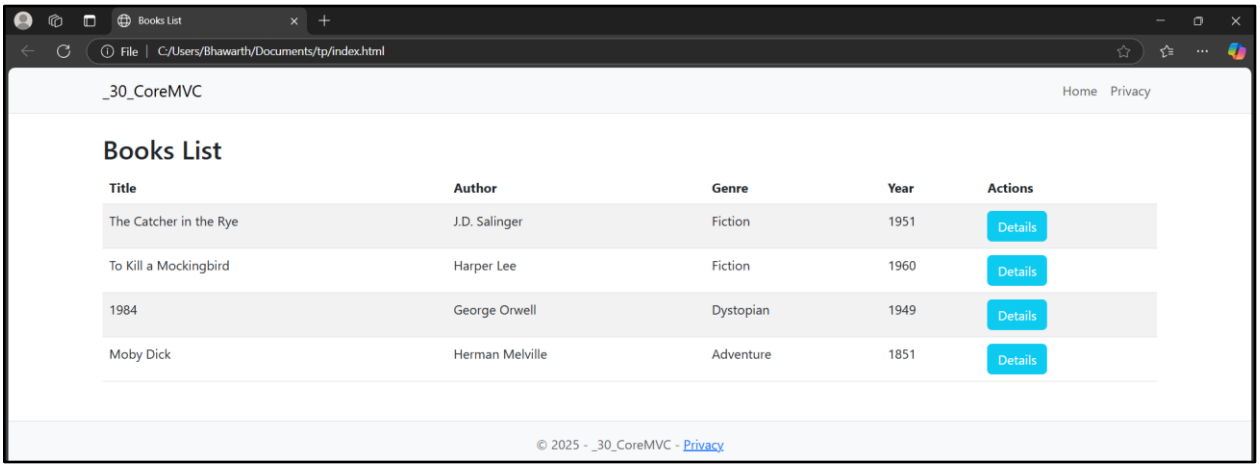
```
<p><strong >Year: </ strong >  @ Model. Year </ p>
<a href="@Url. Action (" Index", " Books")" class=" btn btn - primary" >Back to List </ a>
</ div >
```

Program.cs:

```
endpoints. Map ControllerRoute ( name : " default",
pattern :  "{ controller= Books }/{ action = Index }/{ id ?}");
```

Output:





EXPERIMENT NO 2:

Aim: Design a form with data annotations for validation and display appropriate error messages.

UserViewModel.cs:

```
using  System . ComponentModel . Data Annotations ;
public class UserView Model
{
[ Required ( ErrorMessage = " Name is required ")]
[ String Length (50 , ErrorMessage = " Name cannot be longer than 50 characters")]
public string Name { get; set;
[ Required ( ErrorMessage  =  " Email  is  required ")] [ EmailAddress( ErrorMessage = "
Invalid email address")] public string Email { get; set; }
[ Range (18 , 100 , ErrorMessage = " Age must be between 18 and 100 ")] public int? Age {
get; set; }
}
```

UserController.cs:

```
using Microsoft. Asp NetCore . Mvc;
public class UserController : Controller
{
[ Http Get]
public IAction Result Register (){
return  View ();
}
[ Http Post]
public  IAction Result  Register( UserView Model  model){
if  ( ModelState . IsValid )
{
return RedirectTo Action (" Success");
}
return View ( model);}
public  IAction Result  Success (){
return  View ();
}}
```

Register.cshtml:

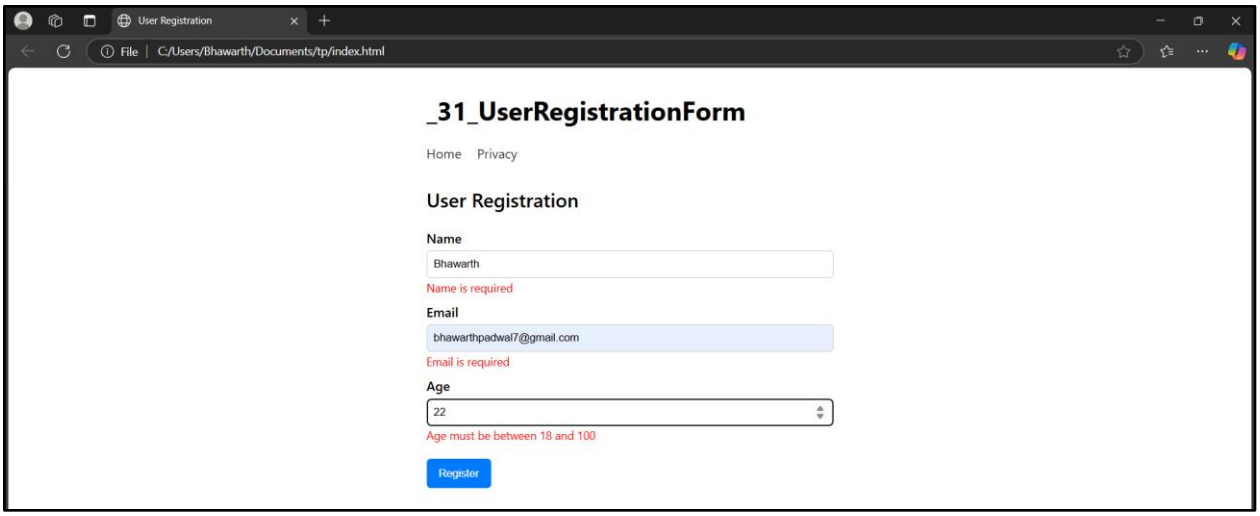
```
@ model  UserView Model
@{
```

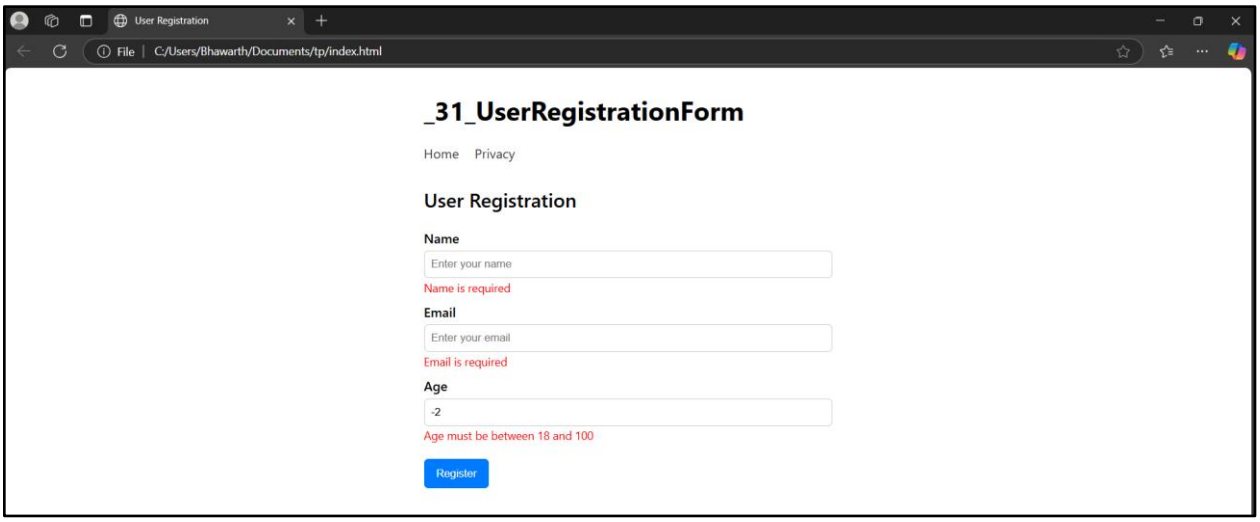
```
View Data [" Title "] = " User  Registration ";
}
<h2 >User  Registration </ h2 >
<form  asp - action =" Register"  method =" post" >
<div  class=" form - group " >
<label  asp - for=" Name " ></ label >
<input  asp - for=" Name "  class=" form - control"  / >
<span asp - validation - for=" Name "  class=" text - danger" ></ span >
</ div >
<div  class=" form - group " >
<label  asp - for=" Email" ></ label >
<input asp - for=" Email" class=" form - control" / >
<span  asp - validation - for=" Email"  class=" text - danger" ></ span >
</ div >
<div  class=" form - group " >
<label  asp - for=" Age " ></ label >
<input asp - for=" Age " class=" form - control" / >
<span  asp - validation - for=" Age"  class=" text - danger" ></ span >
</ div >
<button type =" submit" class=" btn btn - primary" >Register </ button >
</ form >
@ section Scripts {
@{ await  Html. RenderPartialAsync (" _Validation ScriptsPartial ");}
}
```

Success.cshtml:

```
@{
View Data [" Title "] = " Success";
}
<h2 > Registration  Successful! </ h2 >
<p>Thank you for registering . </ p>
```

Output:


User Registration
File | C:/Users/Bhawarth/Documents/tp/index.html
_31_UserRegistrationForm
Home Privacy
User Registration
Name
Bhawarth
Name is required
Email
bhawarthpadwal7@gmail.com
Email is required
Age
22
Age must be between 18 and 100
Register


User Registration
File | C:/Users/Bhawarth/Documents/tp/index.html
_31_UserRegistrationForm
Home Privacy
User Registration
Name
Enter your name
Name is required
Email
Enter your email
Email is required
Age
-2
Age must be between 18 and 100
Register

EXPERIMENT NO 3:

Aim: Implement CRUD operations using Entity Framework Code-First approach for emp data.

Models/Emp.cs:

```
using  Microsoft. Entity Framework Core ;
using  System . ComponentModel . Data Annotations ;
namespace Emp CRUD . Models{
public class Emp{
[ Key]
public int Id { get; set; }
[ Required ]
public string Name { get; set; } public string Department { get; set; }
[ Precision (18 , 2)]
public decimal Salary { get; set; }
}}
```

Data/EmpContext.cs:

```
using Microsoft. Entity Framework Core ; using Emp CRUD . Models;
namespace Emp CRUD . Data{
public class Emp Context : Db Context{
public Emp Context( Db ContextOptions < Emp Context > options) : base ( options) { }
public DbSet <Emp > Emps { get; set; }
}}
```

Program.cs:

```
using  Emp CRUD . Data;
using  Microsoft. Entity Framework Core ;
var builder = Web Application . Create Builder( args);
builder. Services. Add Controllers (); builder. Services. Add EndpointsApiExplorer ();
builder. Services. Add SwaggerGen ();
builder. Services. Add Db Context < Emp Context >( options    => options. Use SqlServer(
builder. Configuration . GetConnection String ("
DefaultConnection "))); var app = builder. Build ();
app  .  Use Swagger  (); app . Use SwaggerUI (); app . Use HttpsRedirection ();
app . Map Controllers (); app . Run ();
```

appsettings.json:

```
" Connection Strings ": {
```

```
" DefaultConnection ": " Server=LAPTOP - BPSVF1 HI \\ SQLEXPRESS ; Database = Emp Db
; Trusted_Connection = True ; TrustServerCertificate = True ;"
}
```

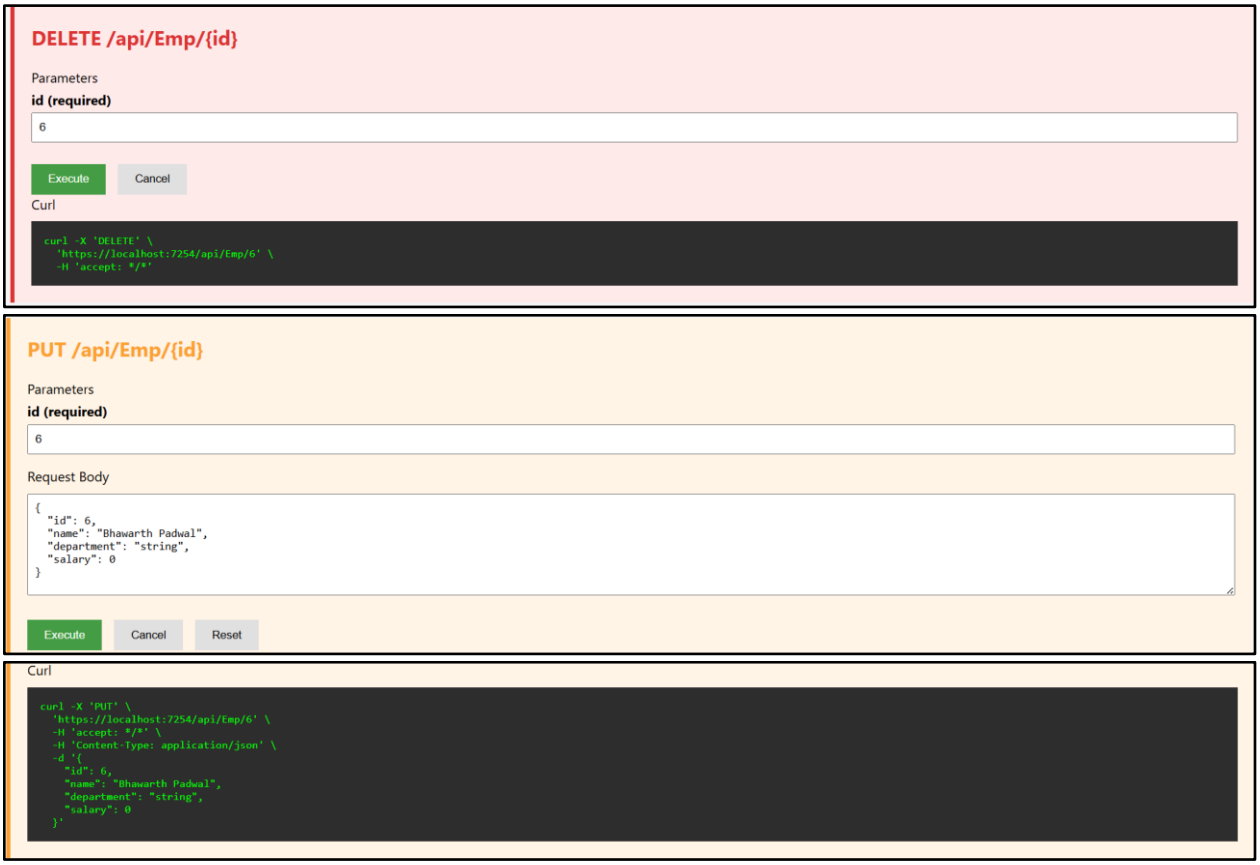
Controllers/EmpController.cs:

```
using Emp CRUD . Data; using Emp CRUD . Models;
using Microsoft. Asp NetCore . Mvc; using Microsoft. Entity Framework Core ;
namespace  Emp CRUD . Controllers{
[ Route (" api/[ controller]")] [ ApiController]
public class Emp Controller : ControllerBase{
private readonly Emp Context _context;
public  Emp Controller( Emp Context  context)
{
_context = context;
}
[ Http Get]
public  async  Task < Action Result < IEnumerable <Emp >>>  GetEmps (){
return await _context. Emps. To ListAsync ();
}
[ Http Get("{ id}")]
public async Task < Action Result <Emp >> GetEmp ( int id)
{
var emp = await _context. Emps. Find Async( id); if ( emp == null) return NotFound ();
return emp ;
}[ Http Post]
public async Task < Action Result <Emp >> PostEmp ([ From Body ] Emp emp ){
_context. Emps. Add ( emp );
await _context. Save ChangesAsync ();
return Created AtAction ( nameof( GetEmp ), new { id = emp . Id }, emp );}
[ Http Put("{ id}")]
public async Task < IAction Result > PutEmp ( int id , Emp emp )
{
if ( id != emp . Id) return Bad Request ();
_context. Entry( emp ). State = EntityState . Modified ; try
{
await _context. Save ChangesAsync ();
}
catch  ( Db Update Concurrency Exception )
{
if (! _context. Emps. Any( e => e. Id == id)) return NotFound ();
```

```
throw ;}
return No Content ();}
[ Http Delete ("{ id}")]
public async Task < IAction Result > Delete Emp ( int id)
{
var emp = await _context. Emps. Find Async( id); if ( emp == null) return NotFound ();
_context. Emps. Remove ( emp );
await _context. Save ChangesAsync (); return No Content ();
}}}
```

Run Migrations
Add - Migration InitialCreate Update - Database

Output:



GET /api/Emp/{id}

id (required)

8

Execute Cancel

Curl

```
curl -X 'GET' \
  'https://localhost:7254/api/Emp/8' \
  -H 'accept: text/plain'
```

POST /api/Emp

Request Body

```
{
  "id": 0,
  "name": "Bhawarth",
  "department": "Computer",
  "salary": 50000
}
```

Execute Cancel Reset

Curl

```
curl -X 'POST' \
  'https://localhost:7254/api/Emp' \
  -H 'accept: text/plain' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 0,
    "name": "Bhawarth",
    "department": "Computer",
    "salary": 50000
  }'
```

Server Response

Code: 201

Response body:

```
{
  "id": 8,
  "name": "Bhawarth",
  "department": "Computer",
  "salary": 50000
}
```

EXPERIMENT NO 4:

Aim: ASP.NET Core MVC with EF Core DB First Approach
https://dotnettutorials.net/lesson/asp-net-core-mvc-with-ef-core-db-first/

Employee.sql:

CREATE DATABASE Emp Crud DB ; GO

USE Emp Crud DB ; GO

CREATE TABLE Employee (
Employee Id INT PRIMARY KEY IDENTITY (1 ,1),
FirstName NVARCHAR (100) , LastName NVARCHAR (100) , Email NVARCHAR (100) ,
Phone NVARCHAR (15) , Salary DECIMAL (18 , 2)
);

Install EF Core Packages

Install - Package Microsoft. Entity Framework Core . SqlServer Install - Package Microsoft. Entity Framework Core . Tools

Scaffold Models from Database (DB-First)

- Run the following command in Package Manager Console:

Scaffold - Db Context " Server= localhost\ SQLEXPRESS ; Database = Emp Crud DB ; Trusted_Connection = True ; TrustServerCertificate = True ;" Microsoft. EntityFramework Core . SqlServer - OutputDir Models - Force

Program.cs:

using Emp MvcDb First. Models;
using Microsoft. Entity Framework Core ;
var builder = Web Application . Create Builder(args); builder. Services. Add ControllersWith Views ();
builder. Services. Add Db Context < Emp Crud Db Context >(options => options. Use SqlServer(builder. Configuration . GetConnection String ("
DefaultConnection "))); var app = builder. Build ();

```
if  (! app . Environment. IsDevelopment ()){
app . Use Exception Handler ("/ Home / Error"); app . Use Hsts ();
}
app . Use HttpsRedirection (); app . Use StaticFiles (); app .  Use Routing  (); app . Use
Authorization ();
app . Map ControllerRoute ( name : " default",
pattern : "{ controller= Employee }/{ action = Index }/{ id ?}");
app . Run ();
```

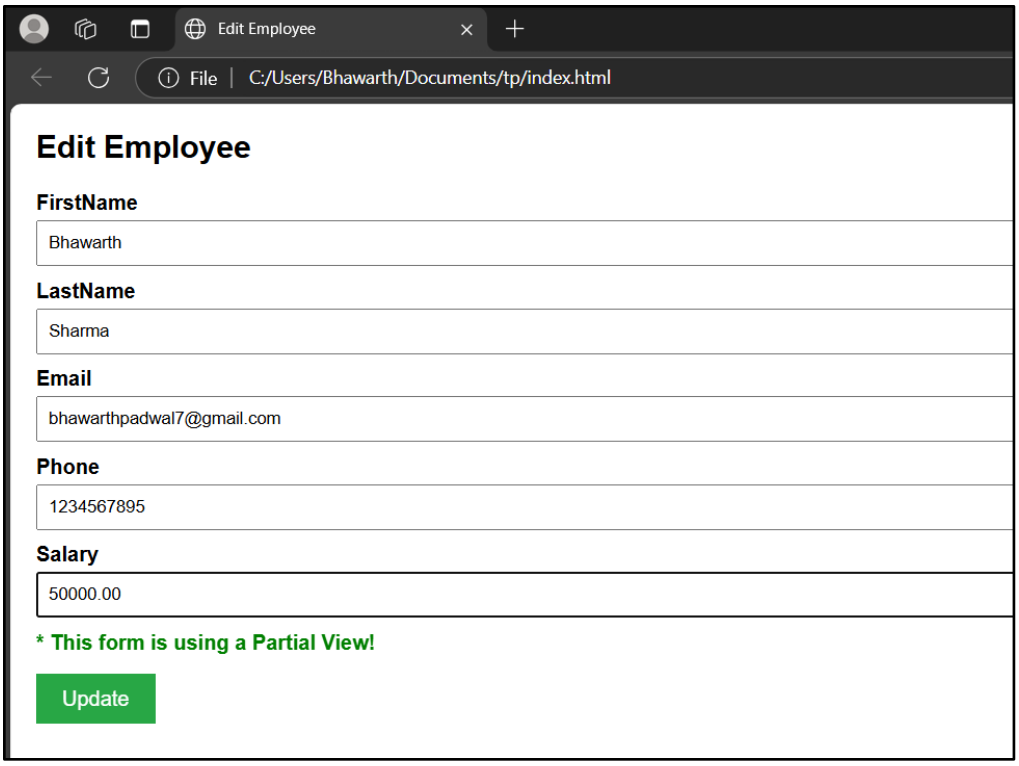
Set the Connection String:

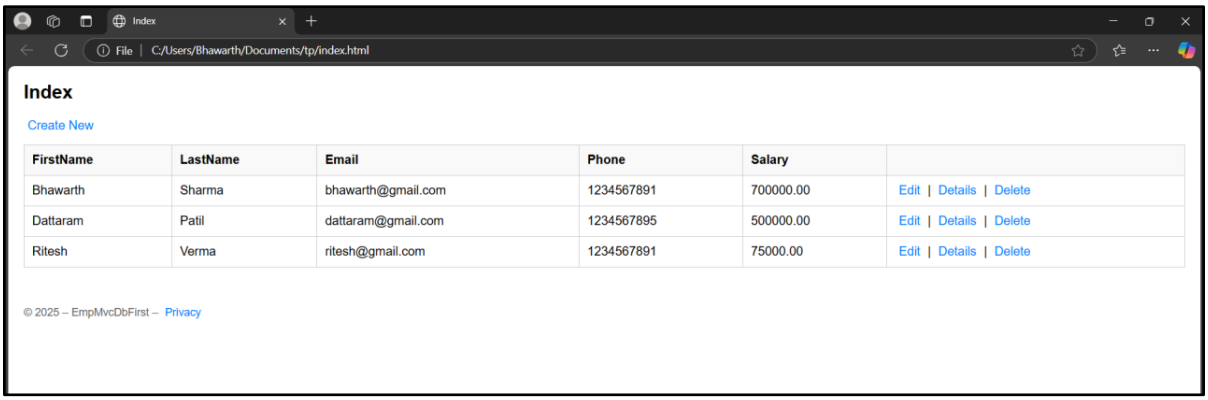
```
" Connection Strings ": {
" DefaultConnection ": " Server= localhost \\ SQLEXPRESS ; Database = Emp Crud DB ;
Trusted_Connection = True ; TrustServerCertificate = True ;"
}
```

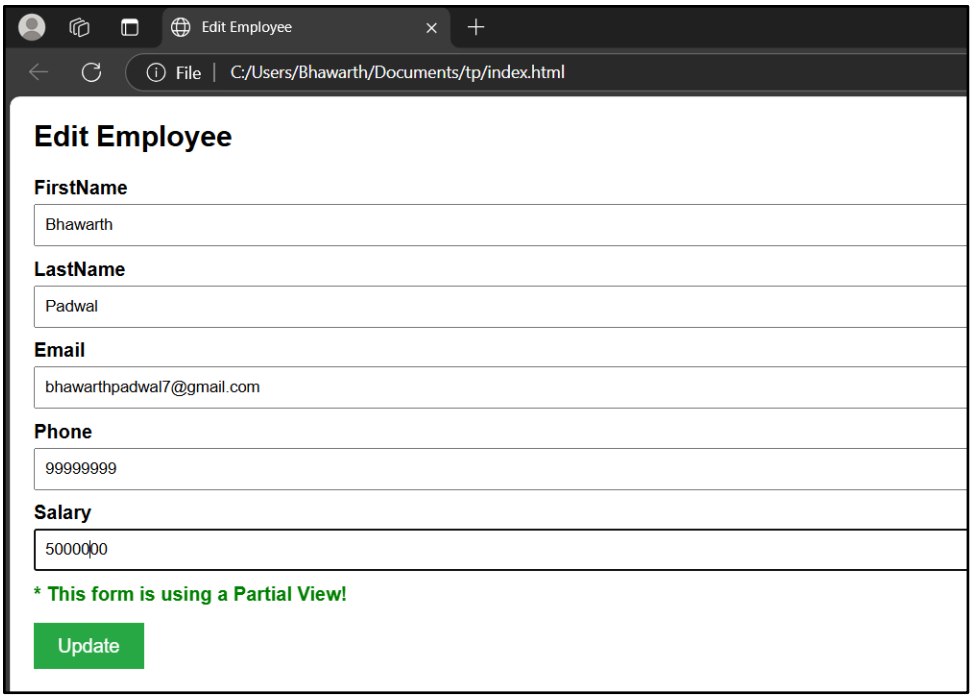
Create the Employee Controller with Views (CRUD)

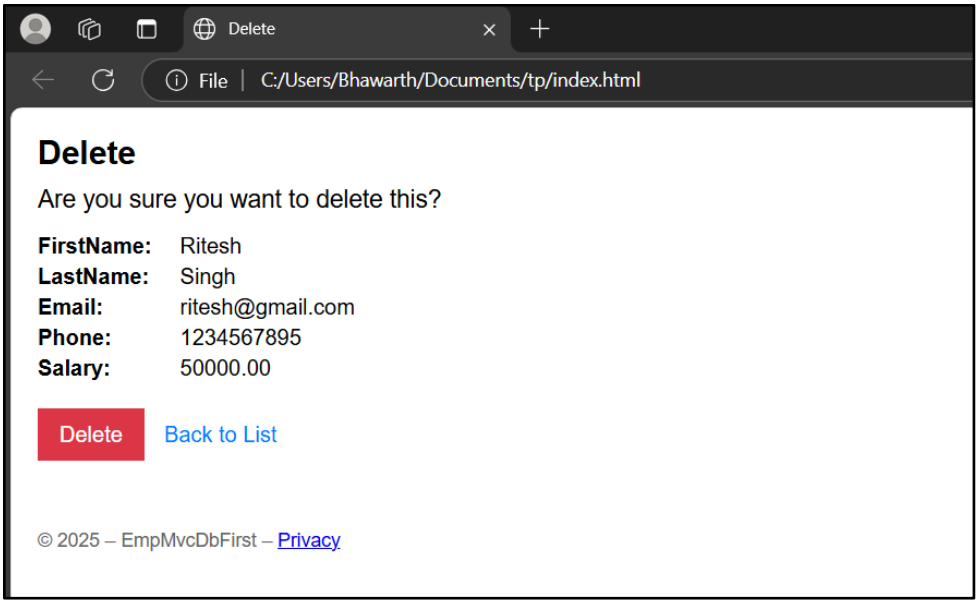
- Controllers → Add → Controller
- Select:
    - MVC Controller with views, using Entity Framework
    - Model class: Employee
    - Data Context class: EmpCrudDbContext
- Name: EmployeeController

Output:


Edit Employee
File | C:/Users/Bhawarth/Documents/tp/index.html
Edit Employee
FirstName
Bhawarth
LastName
Sharma
Email
bhawarthpadwal7@gmail.com
Phone
1234567895
Salary
50000.00
* This form is using a Partial View!
Update


Index
File | C:/Users/Bhawarth/Documents/tp/index.html
Index
Create New
FirstName | LastName | Email | Phone | Salary
Bhawarth | Sharma | bhawarth@gmail.com | 1234567891 | 700000.00 | Edit | Details | Delete
Dattaram | Patil | dattaram@gmail.com | 1234567895 | 500000.00 | Edit | Details | Delete
Ritesh | Verma | ritesh@gmail.com | 1234567891 | 75000.00 | Edit | Details | Delete
© 2025 – EmpMvcDbFirst – Privacy


Edit Employee
File | C:/Users/Bhawarth/Documents/tp/index.html
Edit Employee
FirstName
Bhawarth
LastName
Padwal
Email
bhawarthpadwal7@gmail.com
Phone
99999999
Salary
5000000
* This form is using a Partial View!
Update


Delete
File | C:/Users/Bhawarth/Documents/tp/index.html
Delete
Are you sure you want to delete this?
FirstName: Ritesh
LastName: Singh
Email: ritesh@gmail.com
Phone: 1234567895
Salary: 50000.00
Delete
Back to List
© 2025 – EmpMvcDbFirst – Privacy

EXPERIMENT NO 5:

Aim: Create partial views.

EmployeeFormPartial.cshtml:

```
@ model  Emp MvcDb First. Models. Employee
<div  class=" form - group " >
<label  asp - for=" FirstName "  class=" control - label" ></ label >
<input asp - for=" FirstName " class=" form - control" / >
<span  asp - validation - for=" FirstName "  class=" text - danger" ></ span >
</ div >
<div  class=" form - group " >
<label asp - for=" LastName " class=" control - label" ></ label >
<input  asp - for=" LastName "  class=" form - control" / >
<span  asp - validation - for=" LastName "  class=" text - danger" ></ span >
</ div >
<div  class=" form - group " >
<label  asp - for=" Email"  class=" control - label" ></ label >
<input asp - for=" Email" class=" form - control" / >
<span  asp - validation - for=" Email"  class=" text - danger" ></ span >
</ div >
<div  class=" form - group " >
<label  asp - for=" Phone "  class=" control - label" ></ label >
<input asp - for=" Phone " class=" form - control" / >
<span  asp - validation - for=" Phone "  class=" text - danger" ></ span >
</ div >
<div  class=" form - group " >
<label asp - for=" Salary" class=" control - label" ></ label >
<input  asp - for=" Salary"  class=" form - control" / >
<span  asp - validation - for=" Salary"  class=" text - danger" ></ span >
</ div >
<p style =" color: green ;" >  This form is using a Partial View ! </ p>
```

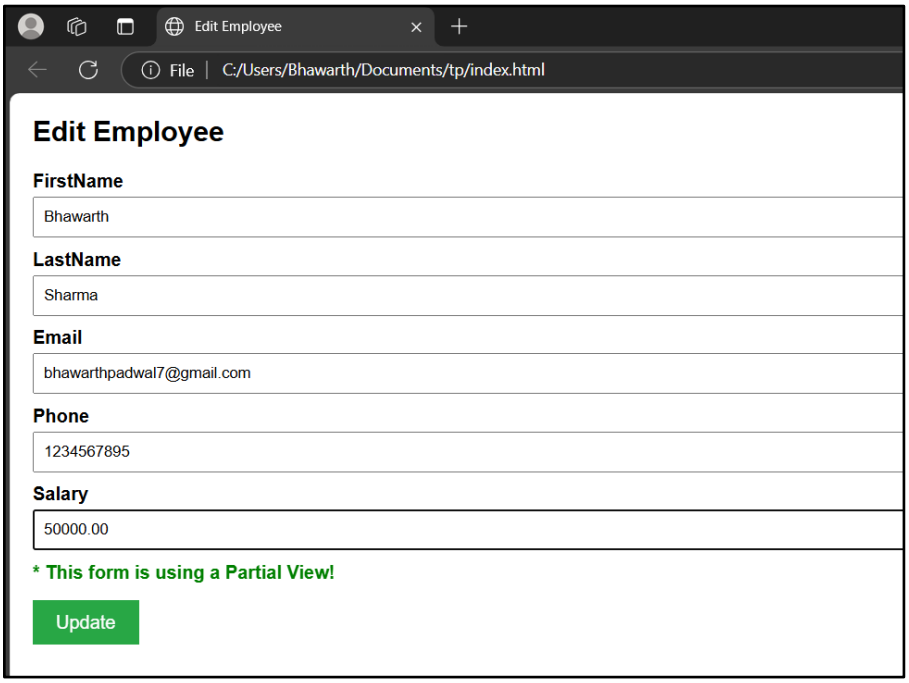
Create.cshtml:

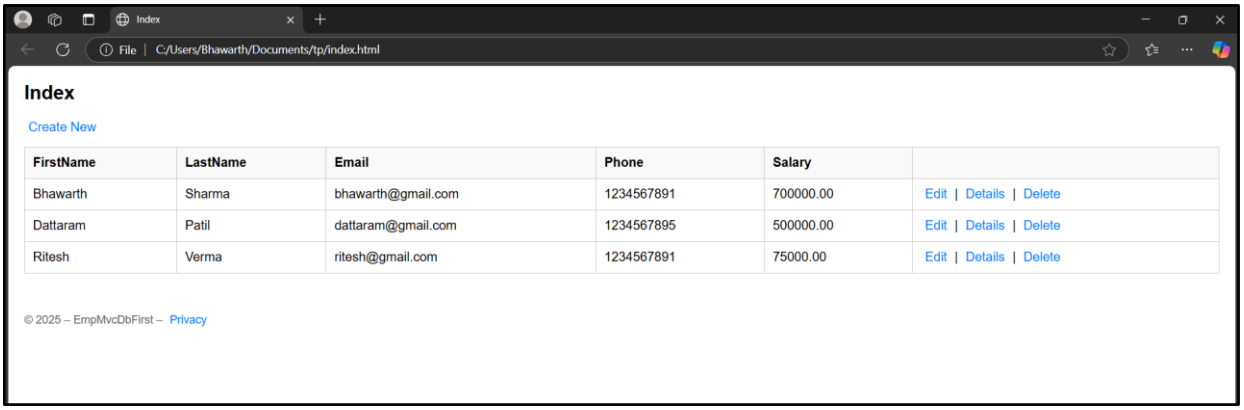
```
@ model  Emp MvcDb First. Models. Employee
<h2 >Create  Employee </ h2 >
<form  asp - action =" Create " >
<partial name =" _Employee Form Partial " model=" Model" / >
<button  type =" submit"  class=" btn  btn - primary" >Save </ button >
</ form >
```

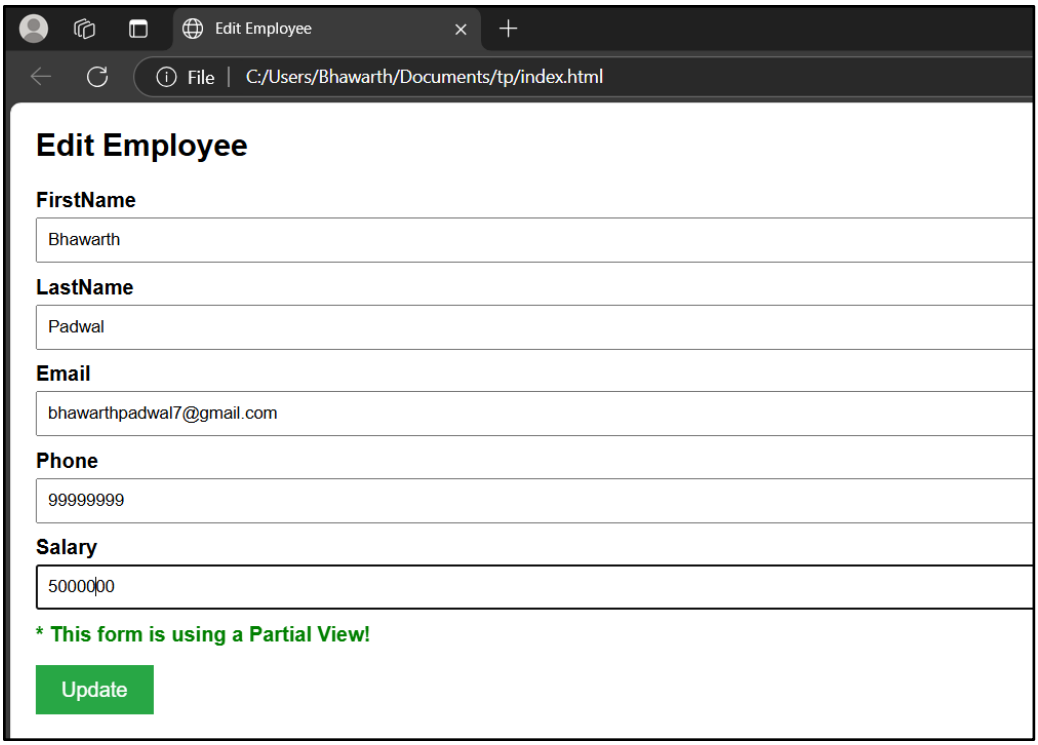
Edit.cshtml:

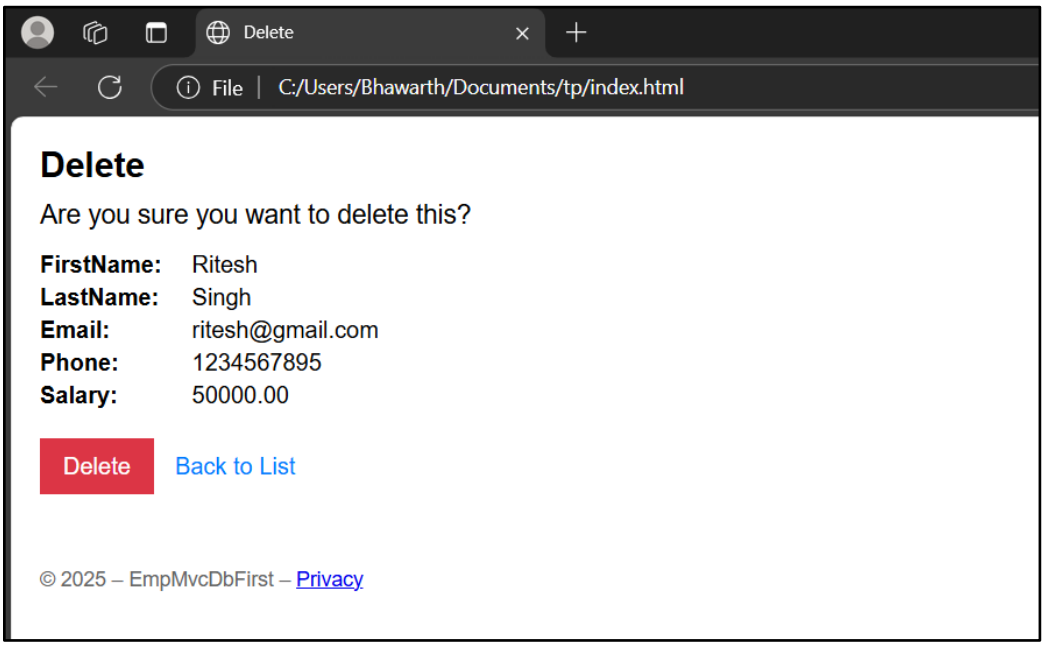
```
@ model  Emp MvcDb First. Models. Employee
<h2 >Edit  Employee </ h2 >
<form  asp - action =" Edit" >
<partial name =" _Employee Form Partial " model=" Model" / >
<input type =" hidden " asp - for=" Employee Id " / >
<button  type =" submit"  class=" btn  btn - success" >Update </ button >
</ form >
```

Output:






Edit Employee
File | C:/Users/Bhawarth/Documents/tp/index.html
Edit Employee
FirstName
Bhawarth
LastName
Padwal
Email
bhawarthpadwal7@gmail.com
Phone
99999999
Salary
5000000
* This form is using a Partial View!
Update


Delete
File | C:/Users/Bhawarth/Documents/tp/index.html
Delete
Are you sure you want to delete this?
FirstName: Ritesh
LastName: Singh
Email: ritesh@gmail.com
Phone: 1234567895
Salary: 50000.00
Delete
Back to List
© 2025 – EmpMvcDbFirst – Privacy