Q1. Create and display welcome message using MVC and create custom url by modifying in route.config file

Program :

FirstController.cs :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace Question1.Controllers
{
    public class FirstController : Controller
    {
        // GET: First
        public ActionResult Index()
        {
            return View();
        }
        public ActionResult First()
        {
            return View("FirstView");
        }
    }
}
```

FirstView.cshtml:

```
@{
    Layout = null;
}
```

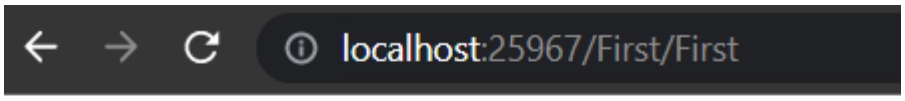
```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>FirstView</title>
</head>
<body>
  <div>
    <h1>Welcome</h1>
  </div>
</body>
</html>
```

RouteConfig.cs :

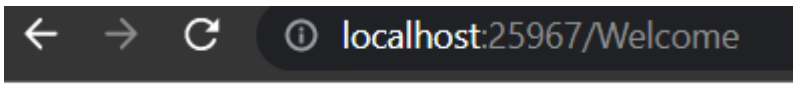
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace Question1
{
  public class RouteConfig
  {
    public static void RegisterRoutes(RouteCollection routes)
    {
      routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
      routes.MapRoute(
        name: "Home",
        url: "Welcome",
        defaults: new { controller = "First", action = "First", id = UrlParameter.Optional }
```

```
        );
         routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
         );
      }
   }
}
```

Output :



Welcome



Welcome

Q2. Create an employee model using strongly typed views

Program :

EmployeeController.cs :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Question2.Models;
```

```
namespace Question2.Controllers
{
    public class EmployeeController : Controller
    {
        // GET: Employee
        public ActionResult Index()
        {
            return View();
        }
        public ActionResult Load()
        {
            Employee obj = new Employee {
                EId = 1 ,
                EName ="abc",
                Salary = 5000
            };
            return View("Emp",obj);
        }
    }
}
```

Employee.cs :

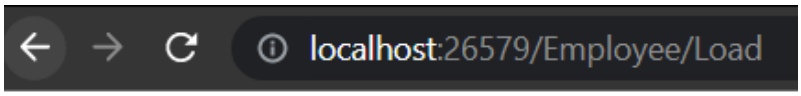
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
namespace Question2.Models
{
    public class Employee
    {
        public int EId { get; set; }
```

```
        public String EName { get; set; }
        public int Salary { get; set; }
    }
}
```

Emp.cshtml :

```
@model Question2.Models.Employee
@{
    Layout = null;
}
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Emp</title>
</head>
<body>
    <div>
        Employee id : @Model.EId<br />
        Employee Name : @Model.EName<br />
        Employee Salary : @Model.Salary
    </div>
</body>
</html>
```

Output:



Employee id : 1
Employee Name : abc
Employee Salary : 5000

Q3. Write a program demonstrating ViewData,ViewBag, Temp data and session variables

Program :

HomeController.cs :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace Question3.Controllers
{
   public class HomeController : Controller
   {
      public ActionResult Index()
      {
         // ViewData
         ViewData["Message"] = "Hello from ViewData";
         // ViewBag
         ViewBag.Title = "ViewBag Title";
         // TempData
         TempData["StatusMessage"] = "Temporary message";
         // Session
         // HttpContext.Session.SetString("Username", "JohnDoe");
         Session["UserName"] = "abc";
         return View();
      }
      public ActionResult Display()
      {
         // ViewData
```

```
        var message = ViewData["Message"];
        // ViewBag
        var title = ViewBag.Title;
        // TempData
        var tempMessage = TempData["StatusMessage"];
        TempData.Keep(); // Retains TempData values for the next request
        // Session
        //var username = HttpContext.Session.GetString("Username");
        var username = Session["UserName"];
        ViewData["Message"] = message;
        ViewBag.Title = title;
        TempData["StatusMessage"] = tempMessage;
        TempData.Keep();
        ViewBag.Username = username;
        return View();
    }
  }
  }
```

Index.cshtml :

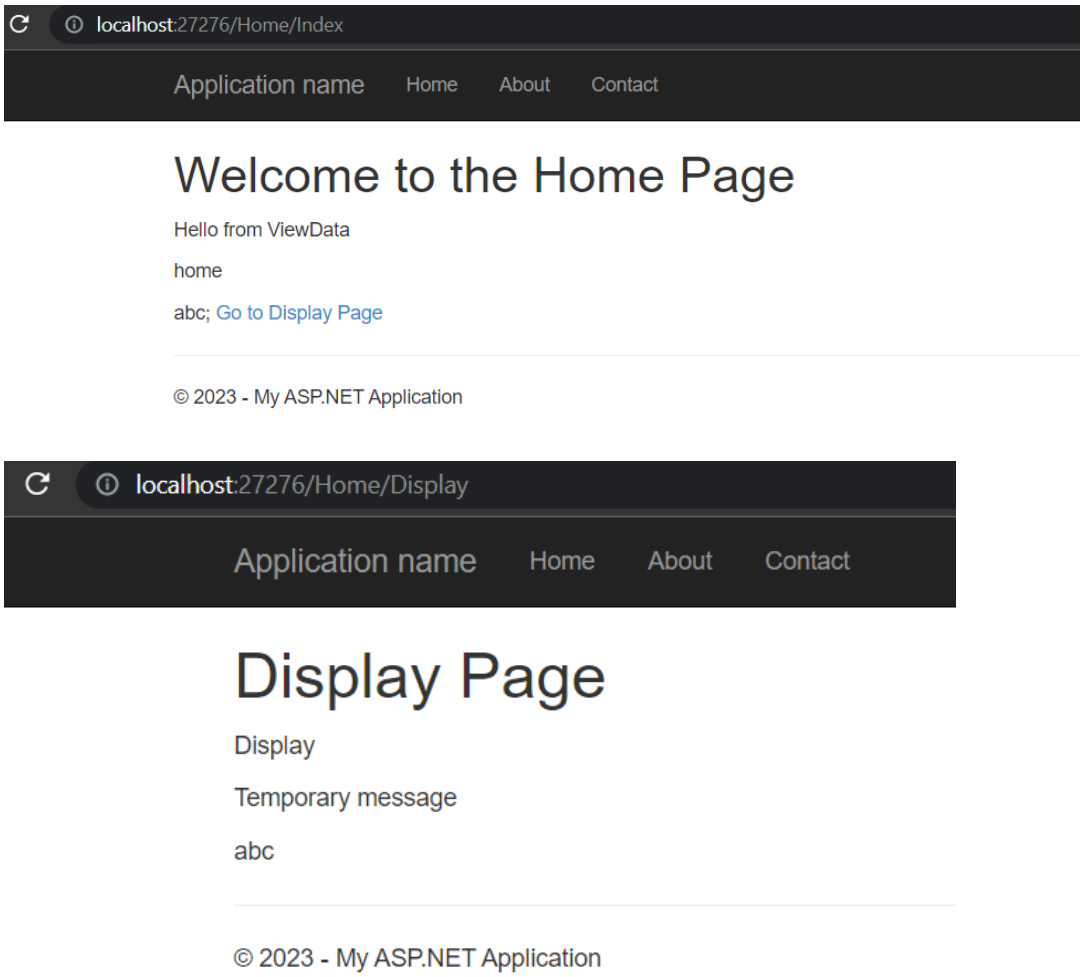
```
<!-- Index.cshtml -->
@{
  ViewData["Title"] = "home";
}
<h1>Welcome to the Home Page</h1>
<p>@ViewData["Message"]</p>
<p>@ViewBag.Title</p>
@{
  TempData["StatusMessage"] = "Temporary message";
}
@Session["Username"].ToString();
```

```
<a href="Display">Go to Display Page</a>
```

Display.cshtml :

```
<!-- Display.cshtml -->

@{

   ViewData["Title"] = "Display";

}

<h1>Display Page</h1>

<p>@ViewData["Message"]</p>

<p>@ViewBag.Title</p>

<p>@TempData["StatusMessage"]</p>

<p>@ViewBag.Username</p>
```

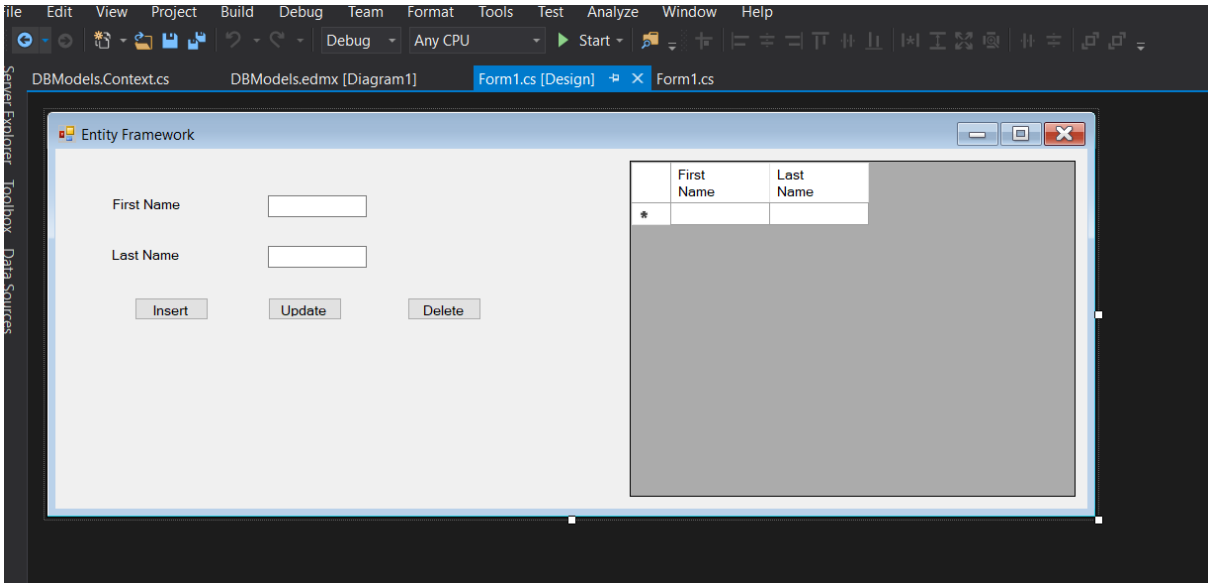
Output :



Q4. Write a program using Entity framework to provide facilty to insert, edit and delete

Program :

Form1.cs[Design]:



Form1.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Entity;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Question4
{
public partial class Form1 : Form
   {
      Customer model = new Customer();
```

```
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            Clear();
            PopulateDataGridView();
        }
        private void BtnDelete_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Are you sure to delete this record ?", "EF CRUD Operation",
MessageBoxButtons.YesNo) == DialogResult.Yes) ;
            using (DBEntities db = new DBEntities())
            {
                var entry = db.Entry(model);
                if (entry.State == EntityState.Detached)
                    db.Customers.Attach(model);
                db.Customers.Remove(model);
                db.SaveChanges();
                PopulateDataGridView();
                Clear();
                MessageBox.Show("Deleted Successfully");
            }
        }
        void Clear()
        {
            txtFirstName.Text = txtLastName.Text = "";
            BtnInsert.Text = "Insert";
            BtnDelete.Enabled = false;
            model.CustomerID = 0;
```

```
        }
        private void BtnInsert_Click(object sender, EventArgs e)
        {
            model.FirstName = txtFirstName.Text.Trim();
            model.LastName = txtLastName.Text.Trim();
            using (DBEntities db = new DBEntities())
            {
                db.Customers.Add(model);
                db.SaveChanges();
            }
            Clear();
            PopulateDataGridView();
            MessageBox.Show("Inserted Successfully");
        }
        void PopulateDataGridView()
        {
            DGVCustomer.AutoGenerateColumns = false;
            using (DBEntities db = new DBEntities()) {
                DGVCustomer.DataSource = db.Customers.ToList<Customer>();
            }
        }
        private void DGVCustomer_DoubleClick(object sender, EventArgs e)
        {
            if(DGVCustomer.CurrentRow.Index != -1){
                model.CustomerID =
Convert.ToInt32(DGVCustomer.CurrentRow.Cells["CustomerID"].Value);
                using (DBEntities db = new DBEntities())
                {
                    model = db.Customers.Where(x => x.CustomerID == model.CustomerID).FirstOrDefault();
                    txtFirstName.Text = model.FirstName;
                    txtLastName.Text = model.LastName;
```

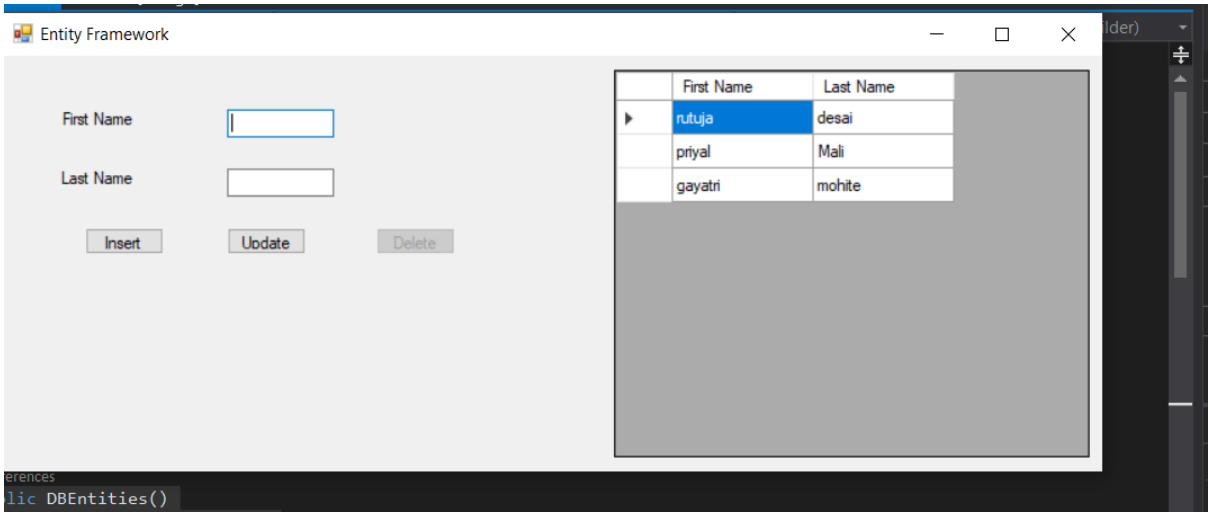
```
            }
            //BtnInsert.Text = "Update";
            BtnDelete.Enabled=true;
        }
    }
    private void BtnUpdate_Click(object sender, EventArgs e)
    {
        model.FirstName = txtFirstName.Text.Trim();
        model.LastName = txtLastName.Text.Trim();
        using (DBEntities db = new DBEntities())
        {
            db.Entry(model).State = EntityState.Modified;
            db.SaveChanges();
        }
        Clear();
        PopulateDataGridView();
        MessageBox.Show("Updated Successfully");
    }
  }
}
```

DBModels.context.cs :

```
namespace Question4
{
  using System;
  using System.Data.Entity;
  using System.Data.Entity.Infrastructure;
  public partial class DBEntities : DbContext
  {
    public DBEntities()
      : base("name=DBEntities")
    {
```

```
        }
        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            throw new UnintentionalCodeFirstException();
        }
        public virtual DbSet<Customer> Customers { get; set; }
    }
    public class Customer
    {
        public int CustomerID { get; set; }
        public String FirstName { get; set; }
        public String LastName { get; set; }
    }
}
```

Output:



Q5. Write a program to send student information form controller to view

Program :

StudentController.cs :

```
using Question5.Models;
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace Question5.Controllers
{
   public class StudentController : Controller
   {
      // GET: Student
      public ActionResult Index()
      {
         return View();
      }
      public ActionResult Load()
      {
        Student obj= new Student
         {
          Id = 1,
          Name = "John Doe",
          Age = 20,
          Grade = "A"
         };
         return View("student",obj);
      }
   }
}
```

Student.cs :

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

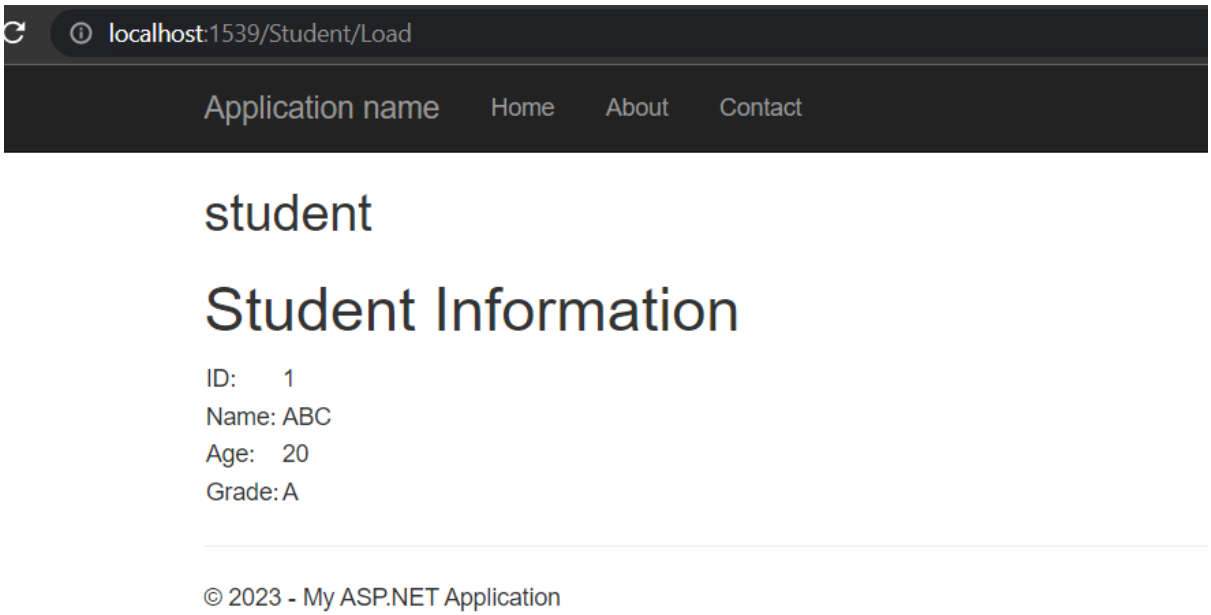
```
namespace Question5.Models
{
    public class Student
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public int Age { get; set; }
        public string Grade { get; set; }
    }
}
```

Student.cshtml :

```
@model Question5.Models.Student
@{
    ViewBag.Title = "student";
}
<h2>student</h2>
<!DOCTYPE html>
<html>
<head>
    <title>Student Information</title>
</head>
<body>
    <h1>Student Information</h1>
    <table>
        <tr>
            <td>ID:</td>
            <td>@Model.Id</td>
        </tr>
        <tr>
            <td>Name:</td>
```

```
        <td>@Model.Name</td>
      </tr>
      <tr>
        <td>Age:</td>
        <td>@Model.Age</td>
      </tr>
      <tr>
        <td>Grade:</td>
        <td>@Model.Grade</td>
      </tr>
    </table>
</body>
</html>
```

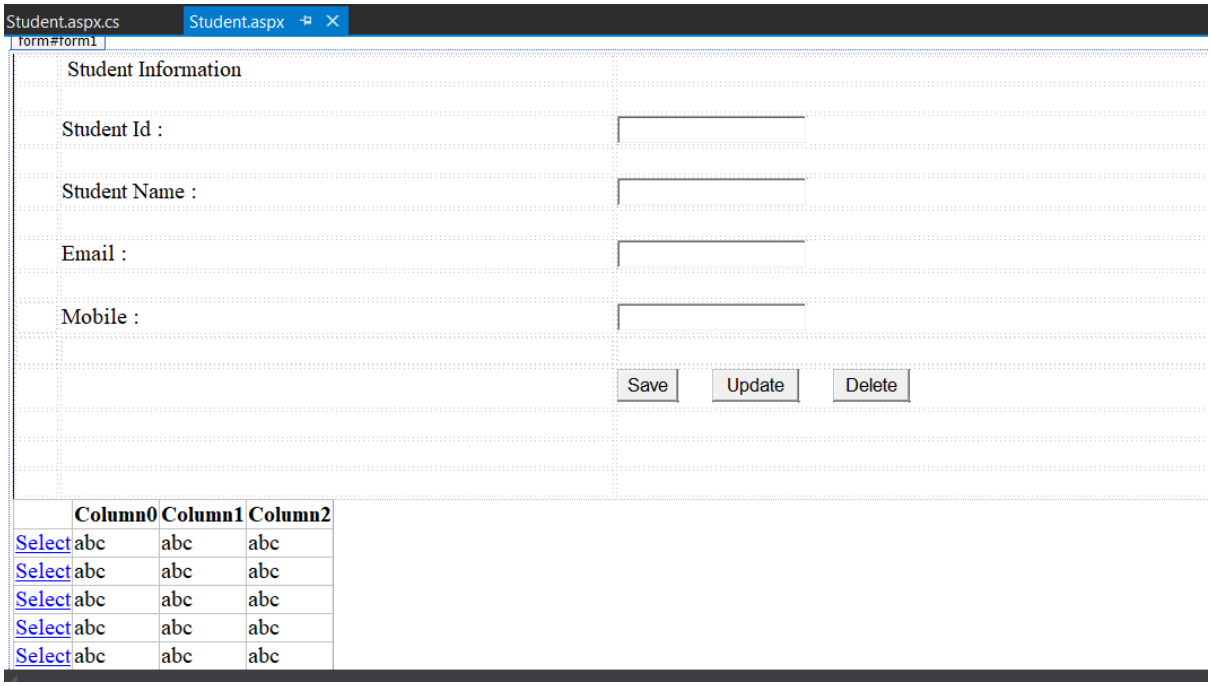
Output :



Q6. Write a program for student registration with Crud Operations

Program:

Student.aspx :



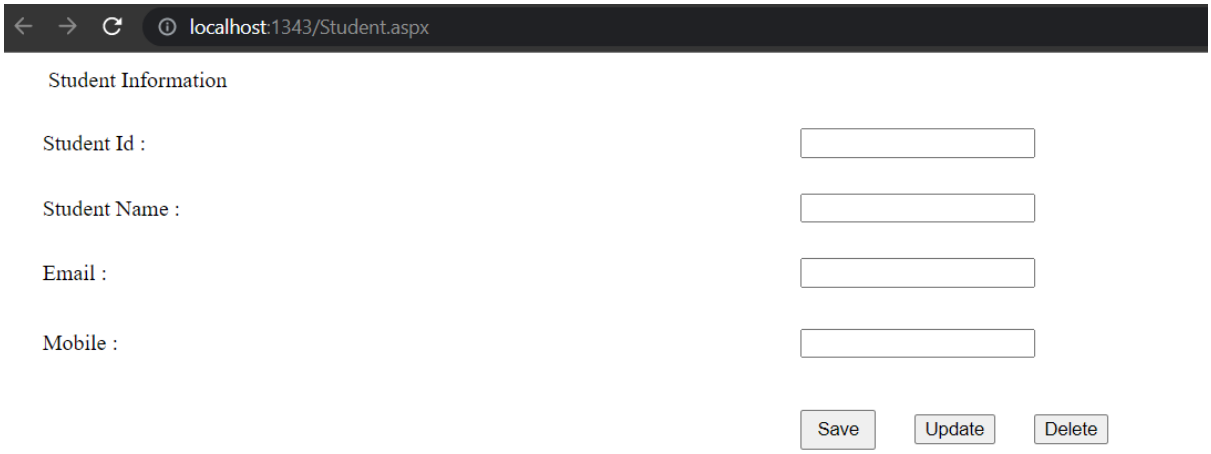
Student.aspx.cs :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
public partial class Student : System.Web.UI.Page
{
   SqlConnection cn = new SqlConnection(@"Data Source = LAPTOP-SLTOQ4HI\RUTU;Initial catalog = MCA;Integrated security=True");
   protected void Page_Load(object sender, EventArgs e)
   {
      LoadData();
   }
   protected void btnSave_Click(object sender, EventArgs e)
   {
```

```
        cn.Open();
        SqlCommand cmd = new SqlCommand("insert into Student values(" +
Convert.ToInt16(txtId.Text) + ",'" + txtName.Text + "','" + txtEmail.Text + "','" + txtMobile.Text + "')",
cn);
        cmd.ExecuteNonQuery();
        Response.Write("<script language='javascript'>alert('Record Saved')</script>");
        cn.Close();
        Clear();
        LoadData();
    }
    void Clear()
    {
        txtId.Text = txtName.Text = txtMobile.Text = txtEmail.Text = "";
    }
    void LoadData()
    {
        cn.Open();
        SqlCommand cmd1 = new SqlCommand("select * from Student", cn);
        SqlDataReader dr = cmd1.ExecuteReader();
        DataTable dt = new DataTable();
        dt.Load(dr);
        GridView1.DataSource = dt;
        GridView1.DataBind();
        cn.Close();
    }
    protected void btnUpdate_Click(object sender, EventArgs e)
    {
        cn.Open();
        SqlCommand cmd = new SqlCommand("update Student set SName='"+txtName.Text+"',
Email='"+txtEmail.Text+"',Mobile='"+txtMobile.Text+"' where SID="+Convert.ToInt16(txtId.Text)+"",
cn);
        cmd.ExecuteNonQuery();
```

```
        Response.Write("<script language='javascript'>alert('Record Updated')</script>");
        cn.Close();
        Clear();
        LoadData();
    }
    protected void btnDelete_Click(object sender, EventArgs e)
    {
        cn.Open();
        SqlCommand cmd = new SqlCommand("delete from Student where SId =
"+Convert.ToInt16(txtId.Text)+"", cn);
        cmd.ExecuteNonQuery();
        Response.Write("<script language='javascript'>alert('Record Deleted')</script>");
        cn.Close();
        Clear();
        LoadData();
    }
    protected void GridView1_SelectedIndexChanged(object sender, EventArgs e)
    {
        cn.Open();
        SqlCommand cmd2 = new SqlCommand("select * from Student where SId ='" +
GridView1.SelectedRow.Cells[1].Text + "' ", cn);
        SqlDataReader dr = cmd2.ExecuteReader();
        dr.Read();
        txtId.Text = dr[0].ToString();
        txtName.Text = dr[1].ToString();
        txtEmail.Text = dr[2].ToString();
        txtMobile.Text = dr[3].ToString();
        cn.Close();
    }
}
```

Output :



	SId	SName	Email	Mobile
Select	1	rutuja	desai	124785345
Select	2	Priyal	jhagioau	12456789
Select	3	jkeyt	jahfioq	1230456789
Select	4	hgfuyty	4gjgui	456889122
Select	3	jkeyt	jahfioq	4578963210
Select	6	ajyruY	AHFOIAU	456789123

Q7. Create a program for Button Group, and write a class for a basic Button Group

Program :

HtmlPage.html :

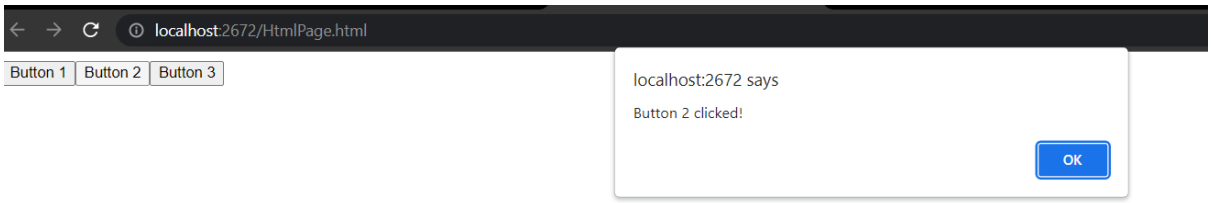
```
<!DOCTYPE html>

<html>

<head>

  <title>Button Group Example</title>

  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">

</head>

<body>

  <div id="buttonGroupContainer"></div>

  <script src="script.js"></script>

</body>

</html>
```

Script.js :

```
class ButtonGroup {
  constructor(containerId, buttons) {
    this.containerId = containerId;
    this.buttons = buttons;
  }
  render() {
    const container = document.getElementById(this.containerId);
    // Create a button group div
    const buttonGroupDiv = document.createElement('div');
    buttonGroupDiv.classList.add('btn-group');
    // Create buttons and add them to the button group
    this.buttons.forEach(button => {
      const buttonElement = document.createElement('button');
      buttonElement.classList.add('btn');
      buttonElement.classList.add('btn-primary');
      buttonElement.textContent = button.label;
      // Add event listener to the button
      buttonElement.addEventListener('click', button.onClick);
      buttonGroupDiv.appendChild(buttonElement);
    });
    container.appendChild(buttonGroupDiv);
  }
}
// Usage example
const buttons = [
 {
   label: 'Button 1',
   onClick: () => {
     alert('Button 1 clicked!');
```

```
    }
  },
  {
    label: 'Button 2',
    onClick: () => {
      alert('Button 2 clicked!');
    }
  },
  {
    label: 'Button 3',
    onClick: () => {
      alert('Button 3 clicked!');
    }
  }
];
const buttonGroup = new ButtonGroup('buttonGroupContainer', buttons);
buttonGroup.render();
```

Output :



Q8. Design a Simple one-page template for photo galleries, portfolios, and more.

Program :

HtmlPage.html :

```
<!DOCTYPE html>
<html>
```

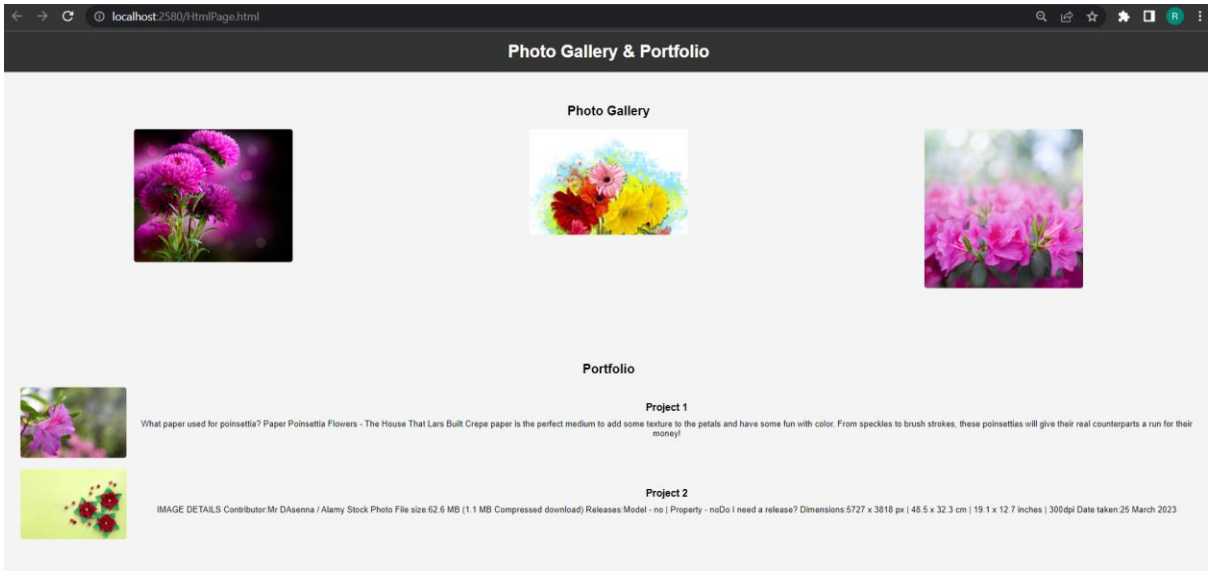
```
<head>
  <title>Photo Gallery Template</title>
  <style>
    /* Add your custom CSS styles here */
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      background-color: #f4f4f4;
    }
    header {
      background-color: #333;
      color: #fff;
      padding: 20px;
      text-align: center;
    }
    h1 {
      margin: 0;
    }
    section {
      padding: 40px;
      text-align: center;
    }
    .gallery {
      display: grid;
      grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
      grid-gap: 20px;
      justify-items: center;
    }
      .gallery img {
        width: 100%;
```

```
            max-width: 300px;
            height: auto;
            border-radius: 5px;
        }
    .portfolio {
        margin-top: 40px;
    }
        .portfolio h2 {
            margin-bottom: 20px;
        }
                .portfolio .item {
                    display: flex;
                    justify-content: space-between;
                    align-items: center;
                    margin-bottom: 20px;
                }
                    .portfolio .item img {
                        width: 200px;
                        height: auto;
                        border-radius: 5px;
                    }
                    .portfolio .item .item-details {
                        flex-grow: 1;
                        margin-left: 20px;
                    }
                       .portfolio .item .item-details h3 {
                            margin: 0;
                        }
                        .portfolio .item .item-details p {
                            margin: 10px 0;
                        }
    </style>
</head>
<body>
```

```
    <header>
        <h1>Photo Gallery & Portfolio</h1>
    </header>
    <section>
        <h2>Photo Gallery</h2>
        <div class="gallery">
            <img src="Flower/Flower1.jpg" alt="Image 1">
            <img src="Flower/Flower2.jpg" alt="Image 2">
            <img src="Flower/Flower3.jpg" alt="Image 3">
            <!-- Add more images here -->
        </div>
    </section>
    <section class="portfolio">
        <h2>Portfolio</h2>
        <div class="item">
            <img src="Flower/Flower4.jpg" alt="Project 1">
            <div class="item-details">
                <h3>Project 1</h3>
                <p>What paper used for poinsettia?
Paper Poinsettia Flowers - The House That Lars Built
Crepe paper is the perfect medium to add some texture to the petals and have some fun with color. From speckles to brush strokes, these poinsettias will give their real counterparts a run for their money!</p>
            </div>
        </div>
        <div class="item">
            <img src="Flower/Flower5.jpg" alt="Project 2" >
            <div class="item-details">
                <h3>Project 2</h3>
                <p>IMAGE DETAILS
Contributor:Mr DAsenna / Alamy Stock Photo
File size:62.6 MB (1.1 MB Compressed download)
Releases:Model - no | Property - noDo I need a release?
Dimensions:5727 x 3818 px | 48.5 x 32.3 cm | 19.1 x 12.7 inches | 300dpi
Date taken:25 March 2023</p>
            </div>
```

```
        </div>
        <!-- Add more portfolio items here -->
    </section>
</body>
</html>
```

Output :



Q9. Design a webpage to demonstrate an example of pricing page built with Cards and featuring a custom header and footer

Program :

HtmlPage.html :

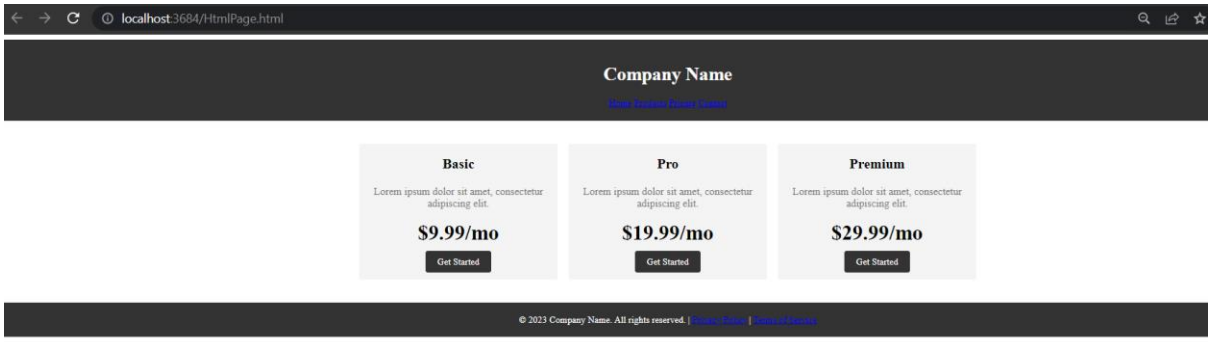
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Pricing Page</title>
  <style>
    /* Styles for header and footer */
    header {
      background-color: #333;
```

```
    color: #fff;
    padding: 20px;
    text-align: center;
}
footer {
    background-color: #333;
    color: #fff;
    padding: 20px;
    text-align: center;
}
/* Styles for pricing cards */
.pricing-cards {
    display: flex;
    justify-content: center;
    align-items: center;
    flex-wrap: wrap;
    gap: 20px;
    max-width: 1200px;
    margin: 0 auto;
    padding: 40px;
}
.card {
    background-color: #f4f4f4;
    border-radius: 4px;
    padding: 20px;
    width: 300px;
    text-align: center;
}
    .card h3 {
        font-size: 24px;
        margin-top: 0;
```

```
        }
        .card p {
            font-size: 18px;
            color: #777;
        }
        .card .price {
            font-size: 36px;
            font-weight: bold;
            margin: 20px 0;
        }
        .card .button {
            background-color: #333;
            color: #fff;
            padding: 10px 20px;
            border: none;
            border-radius: 4px;
            text-decoration: none;
            transition: background-color 0.3s ease;
        }
            .card .button:hover {
                background-color: #555;
            }
    </style>
</head>
<body>
    <header>
        <h1>Company Name</h1>
        <nav>
            <a href="#">Home</a>
            <a href="#">Products</a>
            <a href="#">Pricing</a>
```

```
        <a href="#">Contact</a>
      </nav>
    </header>
    <div class="pricing-cards">
      <div class="card">
        <h3>Basic</h3>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
        <div class="price">$9.99/mo</div>
        <a href="#" class="button">Get Started</a>
      </div>
      <div class="card">
        <h3>Pro</h3>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
        <div class="price">$19.99/mo</div>
        <a href="#" class="button">Get Started</a>
      </div>
      <div class="card">
        <h3>Premium</h3>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
        <div class="price">$29.99/mo</div>
        <a href="#" class="button">Get Started</a>
      </div>
    </div>
    <footer>
      &copy; 2023 Company Name. All rights reserved. | <a href="#">Privacy Policy</a> | <a href="#">Terms of Service</a>
    </footer>
</body>
</html>
```

Output :



Q10. Design a basic admin dashboard shell with fixed sidebar and navbar.

Program:

HtmlPage.html :

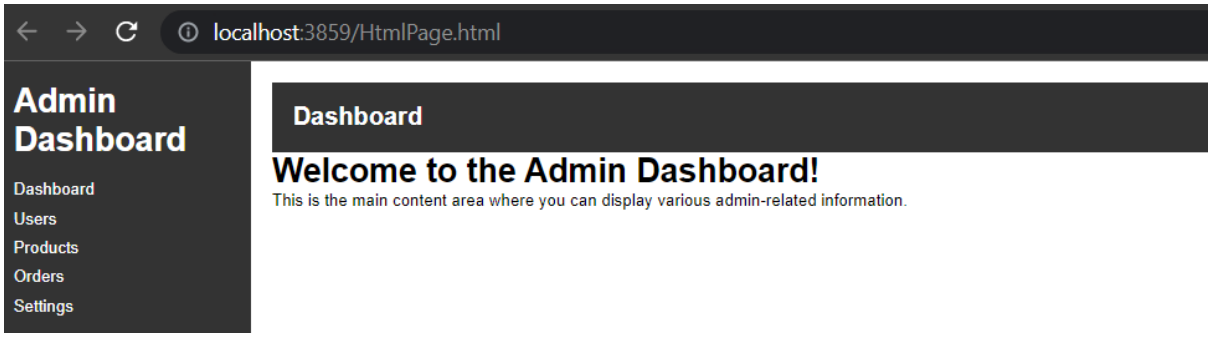
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Admin Dashboard</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }
    body {
      font-family: Arial, sans-serif;
    }
    .container {
      display: flex;
      height: 100vh;
```

```
}
.sidebar {
    width: 250px;
    background-color: #333;
    color: #fff;
    padding: 20px;
}
    .sidebar h1 {
        margin-bottom: 20px;
    }
    .sidebar ul {
        list-style-type: none;
        padding: 0;
        margin: 0;
    }
    .sidebar li {
        margin-bottom: 10px;
    }
    .sidebar a {
        color: #fff;
        text-decoration: none;
    }
        .sidebar a:hover {
            text-decoration: underline;
        }
.content {
    flex: 1;
    padding: 20px;
}
.navbar {
    background-color: #333;
```

```
            color: #fff;
            padding: 20px;
            display: flex;
            justify-content: space-between;
        }
            .navbar h2 {
                margin: 0;
            }
    </style>
</head>
<body>
    <div class="container">
        <div class="sidebar">
            <h1>Admin Dashboard</h1>
            <ul>
                <li><a href="#">Dashboard</a></li>
                <li><a href="#">Users</a></li>
                <li><a href="#">Products</a></li>
                <li><a href="#">Orders</a></li>
                <li><a href="#">Settings</a></li>
            </ul>
        </div>
        <div class="content">
            <div class="navbar">
                <h2>Dashboard</h2>
                <a href="#">Logout</a>
            </div>
            <!-- Your content goes here -->
            <h1>Welcome to the Admin Dashboard!</h1>
            <p>This is the main content area where you can display various admin-related
information.</p>
```

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

Output :



Q11. Design a webpage for demonstration of all responsive and container options for the navbar

Program :

HtmlPage.html :

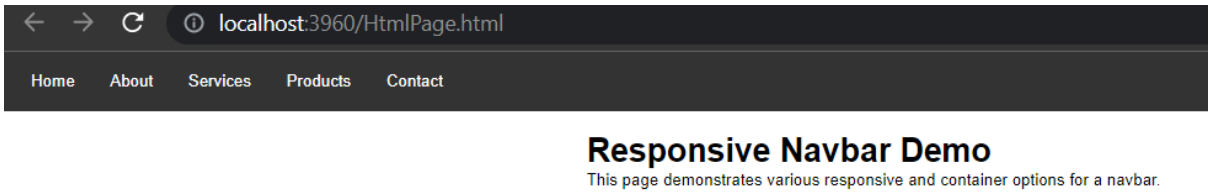
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Responsive Navbar Demo</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }
    body {
      font-family: Arial, sans-serif;
```

```
}
/* Navbar styles */
.navbar {
  background-color: #333;
  color: #fff;
  padding: 20px;
}
  .navbar ul {
    list-style-type: none;
    padding: 0;
    margin: 0;
  }
  .navbar li {
    display: inline-block;
    margin-right: 10px;
  }
  .navbar a {
    color: #fff;
    text-decoration: none;
    padding: 10px;
  }
    .navbar a:hover {
      background-color: #555;
    }
/* Container styles */
.container {
  max-width: 1200px;
  margin: 0 auto;
  padding: 20px;
}
/* Responsive styles */
```

```
        @media (max-width: 768px) {
            .navbar {
                flex-direction: column;
                align-items: flex-start;
            }
                .navbar li {
                    display: block;
                    margin-bottom: 10px;
                }
        }
        @media (max-width: 576px) {
            .navbar {
                padding: 10px;
            }
                .navbar a {
                    padding: 5px;
                }
        }
    </style>
</head>
<body>
    <div class="navbar">
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">About</a></li>
            <li><a href="#">Services</a></li>
            <li><a href="#">Products</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
    </div>
```

```
    <div class="container">
      <h1>Responsive Navbar Demo</h1>
      <p>This page demonstrates various responsive and container options for a navbar.</p>
    </div>
</body>
</html>
```

Output :



Responsive Navbar Demo

This page demonstrates various responsive and container options for a navbar.