**Table of Content**

**Practical 1:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sn** | **Title** | **Date** | **Signature** |
| 14 | Write a program to demonstrate interface |  |  |

**Practical 2**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sn** | **Title** | **Date** | **Signature** |
| 1 | Create a web form that contains two label that display Enter first number and enter second number, two text box for taking an input, third text box for output and three button add, subtract and find prime. Add proper validation like text box should not be empty, value of first field should be greater than value of second field. If add button is clicked display the addition of two number given in textboxes, if subtract button is clicked display the subtraction of two number given in textboxes and if findprime is clicked then display the prime number from first value to second value given in textboxes. |  |  |
| 2 | Write a console program (ADO.net) to create a table tbl\_registration that have fields (id int primary key, username, password, repassword, gender, course and country). After this perform the following operation   * Insert any 5 data into tbl\_registration. All the required input should be taken from user * Display all the record of database table * Update the name and course of a person to data given by user according to id given by user * Delete the record of person whose id is given by user * Display all the record of person who are male and also from country Nepal |  |  |
| 3 | For the table created in question no. 3, create a web form for registration which should contains username, password, repassword, gender (radio button), course (checkbox) and country (dropdown) and submit button. When submit is pressed insert the value given by user into database table. Use proper validation: username, password and repassword should not be empty, item of radio button, checkbox and dropdown menu should be selected. |  |  |

**Practical 3**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sn** | **Title** | **Date** | **Signature** |
| 1 | Demonstrate model, view and controller by showing different action method, views, model, accessing controller, model and view. |  |  |
| 2 | Demonstrate use of razor syntax |  |  |
| 3 | Demonstrate use of html tag helper |  |  |
| 4 | Using Entity framework create a table tbl\_officer having field (id, name, gender, phone, department and position) after this perform complete CRUDE operation (insert, update, display and delete). User proper validation. |  |  |
| 5 | Demonstrate different state management technique like SessionState, TempData, HttpContext |  |  |
| 6 | Demonstrate different client-side state management like cookies, Query string and hidden fields |  |  |
| 7 | Write a program to create complete form and validate using jquery and react |  |  |
| 8 | Write a program to demonstrate authentication and authorization (Role, claim and policies) by create a complete form in asp.net core |  |  |
| 9 | Write a program to prevent SQLInjectionAttack, Cross Site Request forgery (CSRF) and open redirect attack |  |  |

1. **Create a web form that contains two label that display Enter first number and enter second number, two text box for taking an input, third text box for output and three button add, subtract and find prime. Add proper validation like text box should not be empty, value of first field should be greater than value of second field. If add button is clicked display the addition of two number given in textboxes, if subtract button is clicked display the subtraction of two number given in textboxes and if findprime is clicked then display the prime number from first value to second value given in textboxes.**

Source Code:

**WebForm1.aspx:**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication2.WebForm1" %>

<!DOCTYPE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<!-- Labels and Textboxes for Input -->

<asp:Label ID="Label1" runat="server" Text="Enter first number:"></asp:Label>

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

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"

ControlToValidate="TextBox1" ErrorMessage="First number is required." ForeColor="Red"></asp:RequiredFieldValidator>

<br /><br />

<asp:Label ID="Label2" runat="server" Text="Enter second number:"></asp:Label>

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

<asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"

ControlToValidate="TextBox2" ErrorMessage="Second number is required." ForeColor="Red"></asp:RequiredFieldValidator>

<br /><br />

<asp:CustomValidator ID="CustomValidator1" runat="server"

ControlToValidate="TextBox1"

ErrorMessage="First number must be greater than second number."

ForeColor="Red" OnServerValidate="ValidateFirstGreater"></asp:CustomValidator>

<br /><br />

<!-- Result Textbox -->

<asp:TextBox ID="TextBoxResult" runat="server" ReadOnly="true"></asp:TextBox>

<br /><br />

<!-- Buttons -->

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

<asp:Button ID="ButtonSubtract" runat="server" Text="Subtract" OnClick="ButtonSubtract\_Click" />

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

</div>

</form>

</body>

</html>

**WebForm1.aspx.cs:**

using System;

using System.Text;

using System.Web.UI;

namespace WebApplication2

{

public partial class WebForm1 : Page

{

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

{

}

// Custom Validator to ensure first number is greater than second number

protected void ValidateFirstGreater(object source, ServerValidateEventArgs args)

{

int firstNumber, secondNumber;

if (int.TryParse(TextBox1.Text, out firstNumber) && int.TryParse(TextBox2.Text, out secondNumber))

{

args.IsValid = firstNumber > secondNumber;

}

else

{

args.IsValid = false;

}

}

// Add Button Click event

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

{

if (Page.IsValid)

{

int firstNumber = int.Parse(TextBox1.Text);

int secondNumber = int.Parse(TextBox2.Text);

int result = firstNumber + secondNumber;

TextBoxResult.Text = "Sum: " + result.ToString();

}

}

// Subtract Button Click event

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

{

if (Page.IsValid)

{

int firstNumber = int.Parse(TextBox1.Text);

int secondNumber = int.Parse(TextBox2.Text);

int result = firstNumber - secondNumber;

TextBoxResult.Text = "Difference: " + result.ToString();

}

}

// Find Prime Button Click event

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

{

if (Page.IsValid)

{

int firstNumber = int.Parse(TextBox1.Text);

int secondNumber = int.Parse(TextBox2.Text);

StringBuilder primeNumbers = new StringBuilder();

primeNumbers.Append("Prime Numbers: ");

for (int i = secondNumber; i <= firstNumber; i++)

{

if (IsPrime(i))

{

primeNumbers.Append(i + " ");

}

}

TextBoxResult.Text = primeNumbers.ToString();

}

}

// Helper method to check if a number is prime

private bool IsPrime(int num)

{

if (num <= 1)

{

return false;

}

for (int i = 2; i <= Math.Sqrt(num); i++)

{

if (num % i == 0)

{

return false;

}

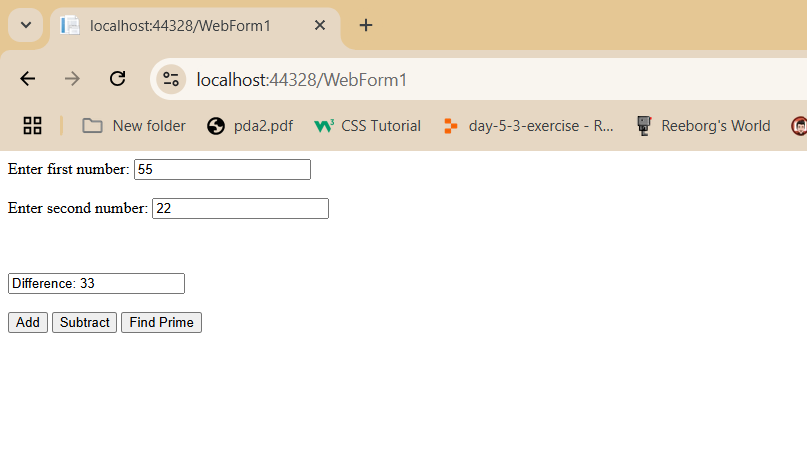
}

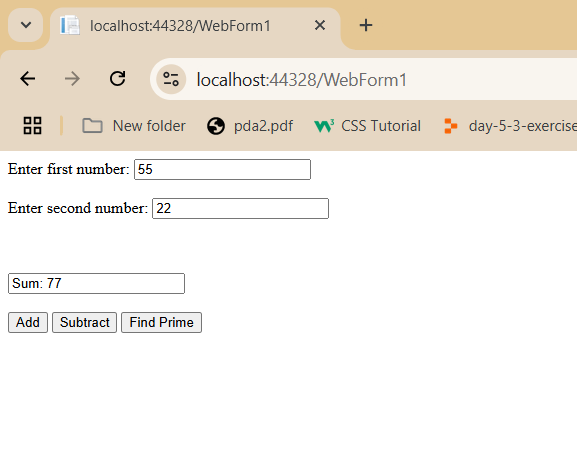
return true;

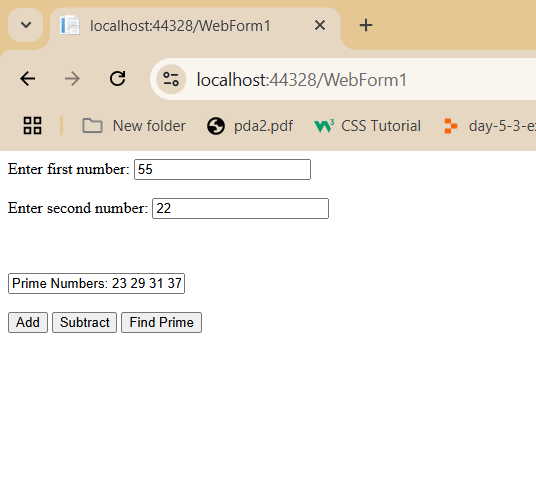
}

}}

**Output:**







1. **Write a console program (ADO.net) to create a table tbl\_registration that have fields (id int primary key, username, password, repassword, gender, course and country). After this perform the following operation**

* **Insert any 5 data into tbl\_registration. All the required input should be taken from user**
* **Display all the record of database table**
* **Update the name and course of a person to data given by user according to id given by user**
* **Delete the record of person whose id is given by user**
* **Display all the record of person who are male and also from country Nepal**

**CRUD.cs**

using System;

using System.Collections.Generic;

using System.Diagnostics.Metrics;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Data.SqlClient;

namespace ADO

{

class CRUD

{

string cs = "Data Source=DESKTOP-R5GKI2K\\SQLEXPRESS;Initial Catalog=db\_nccsa;Integrated Security=True;TrustServerCertificate=True";

public void CreateTable()

{

try

{

SqlConnection sc = new SqlConnection(cs);

sc.Open();

string createTableQuery = "CREATE TABLE tbl\_users (id INT PRIMARY KEY, username VARCHAR(50), password VARCHAR(50), repassword VARCHAR(50), gender VARCHAR(50), course VARCHAR(50), country VARCHAR(50))";

SqlCommand cmd = new SqlCommand(createTableQuery, sc);

int res = cmd.ExecuteNonQuery();

Console.WriteLine("Table 'tbl\_users' created successfully!!");

}

catch (SqlException ex)

{

Console.WriteLine("SQL Error: " + ex.Message);

}

catch (Exception ex)

{

Console.WriteLine("General Error: " + ex.Message);

}

}

public void InsertRecord()

{

try

{

Console.Write("ID: ");

int id = int.Parse(Console.ReadLine());

Console.Write("Username: ");

string username = Console.ReadLine();

Console.Write("Password: ");

string password = Console.ReadLine();

Console.Write("Re-enter Password: ");

string repassword = Console.ReadLine();

Console.Write("Gender: ");

string gender = Console.ReadLine();

Console.Write("Course: ");

string course = Console.ReadLine();

Console.Write("Country: ");

string country = Console.ReadLine();

SqlConnection sc = new SqlConnection(cs);

sc.Open();

string insertQuery = "INSERT INTO tbl\_users (id, username, password, repassword, gender, course, country) VALUES (@id, @username, @password, @repassword, @gender, @course, @country)";

SqlCommand cmd = new SqlCommand(insertQuery, sc);

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

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

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

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

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

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

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

cmd.ExecuteNonQuery();

Console.WriteLine("Record inserted successfully!!");

}

catch (SqlException ex)

{

Console.WriteLine("SQL Error: " + ex.Message);

}

catch (Exception ex)

{

Console.WriteLine("General Error: " + ex.Message);

}

}

public void DisplayAllRecords()

{

try

{

SqlConnection sc = new SqlConnection(cs);

sc.Open();

string disQuery = "SELECT \* FROM tbl\_users";

SqlCommand cmd = new SqlCommand(disQuery, sc);

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

Console.WriteLine("Id is " + reader["id"]);

Console.WriteLine("Username is " + reader["username"]);

Console.WriteLine("Course is " + reader["course"]);

Console.WriteLine("Country is " + reader["country"]);

}

}

catch (SqlException ex)

{

Console.WriteLine("SQL Error: " + ex.Message);

}

catch (Exception ex)

{

Console.WriteLine("General Error: " + ex.Message);

}

}

public void UpdateRecord()

{

try

{

Console.Write("Enter ID of the user to update: ");

int updateId = int.Parse(Console.ReadLine());

Console.Write("Enter new username: ");

string newUsername = Console.ReadLine();

Console.Write("Enter new course: ");

string newCourse = Console.ReadLine();

SqlConnection sc = new SqlConnection(cs);

sc.Open();

string updateQuery = "UPDATE tbl\_users SET username = @username, course = @course WHERE id = @id";

SqlCommand cmd = new SqlCommand(updateQuery, sc);

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

cmd.Parameters.AddWithValue("@username", newUsername);

cmd.Parameters.AddWithValue("@course", newCourse);

int rowsAffected = cmd.ExecuteNonQuery();

if (rowsAffected > 0)

{

Console.WriteLine("Record updated successfully!!");

}

else

{

Console.WriteLine("No record found with the given ID.");

}

}

catch (SqlException ex)

{

Console.WriteLine("SQL Error: " + ex.Message);

}

catch (Exception ex)

{

Console.WriteLine("General Error: " + ex.Message);

}

}

public void DeleteRecord()

{

try

{

Console.Write("Enter ID of the user to delete: ");

int deleteId = int.Parse(Console.ReadLine());

SqlConnection sc = new SqlConnection(cs);

sc.Open();

string deleteQuery = "DELETE FROM tbl\_users WHERE id = @id";

SqlCommand cmd = new SqlCommand(deleteQuery, sc);

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

int rowsAffected = cmd.ExecuteNonQuery();

if (rowsAffected > 0)

{

Console.WriteLine("Record deleted successfully!!");

}

else

{

Console.WriteLine("No record found with the given ID.");

}

}

catch (SqlException ex)

{

Console.WriteLine("SQL Error: " + ex.Message);

}

catch (Exception ex)

{

Console.WriteLine("General Error: " + ex.Message);

}

}

public void DisplayMaleUsersFromNepal()

{

try

{

SqlConnection sc = new SqlConnection(cs);

sc.Open();

string selectQuery = "SELECT \* FROM tbl\_users WHERE gender = 'Male' AND country = 'Nepal'";

SqlCommand cmd = new SqlCommand(selectQuery, sc);

SqlDataReader reader = cmd.ExecuteReader();

Console.WriteLine("Male Users from Nepal:");

while (reader.Read())

{

Console.WriteLine($"ID: {reader["id"]}, Username: {reader["username"]}, Course: {reader["course"]}");

}

}

catch (SqlException ex)

{

Console.WriteLine("SQL Error: " + ex.Message);

}

catch (Exception ex)

{

Console.WriteLine("General Error: " + ex.Message);

}

}

}

}

**Program.cs:**

using System.Data;

using System.Data.SqlClient;

using ADO;

class Program

{

static void Main(string[] args)

{

CRUD c1 = new CRUD();

c1.CreateTable();

c1.InsertRecord();

c1.UpdateRecord();

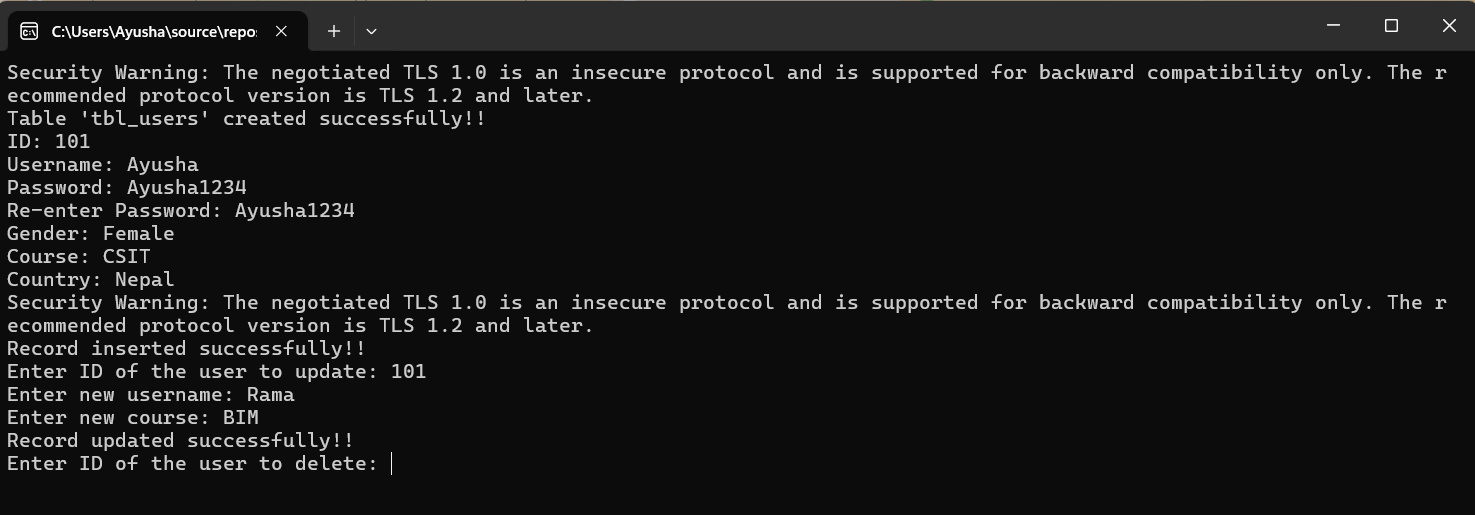
c1.DeleteRecord();

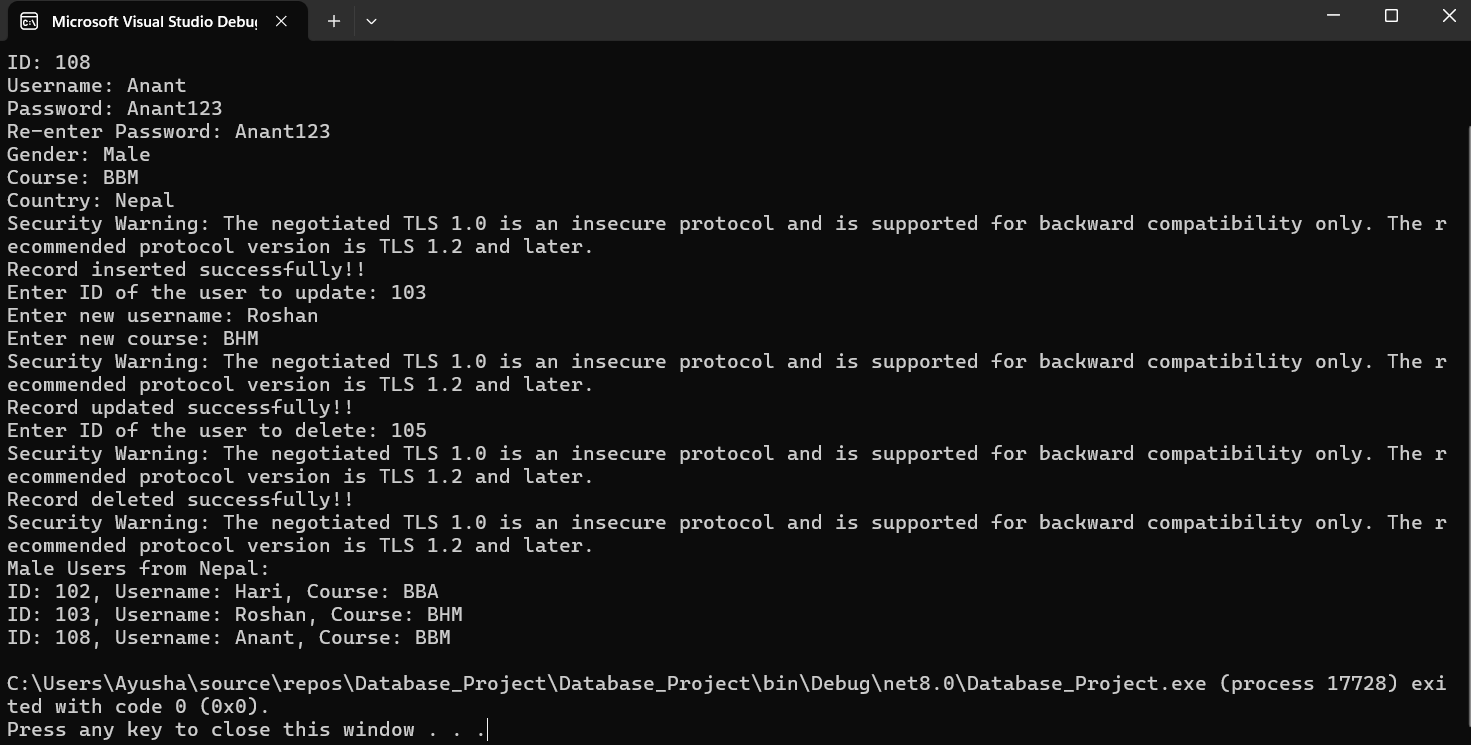
c1.DisplayMaleUsersFromNepal();

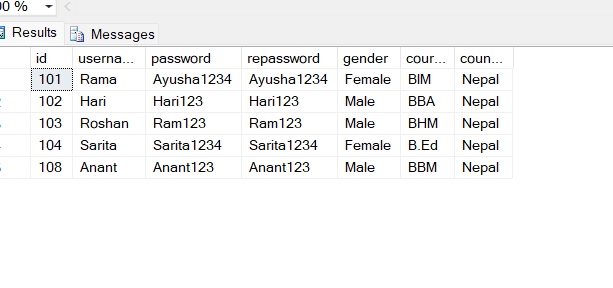
}

}

**Output:**





\

1. **For the table created in question no. 3, create a web form for registration which should contains username, password, repassword, gender (radio button), course (checkbox) and country (dropdown) and submit button. When submit is pressed insert the value given by user into database table. Use proper validation: username, password and repassword should not be empty, item of radio button, checkbox and dropdown menu should be selected.**

**Registration.aspx:**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Registration.aspx.cs" Inherits="WebApplication7.Registration" %>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Registration</title>

</head>

<body>

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

<h2>Registration Form</h2>

<div>

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

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

</div>

<div>

<label for="txtPassword">Password:</label>

<asp:TextBox ID="txtPassword" runat="server" TextMode="Password" required="true"></asp:TextBox>

</div>

<div>

<label for="txtRepassword">Re-enter Password:</label>

<asp:TextBox ID="txtRepassword" runat="server" TextMode="Password" required="true"></asp:TextBox>

</div>

<div>

<label>Gender:</label>

<asp:RadioButton ID="rbMale" runat="server" Text="Male" GroupName="Gender" />

<asp:RadioButton ID="rbFemale" runat="server" Text="Female" GroupName="Gender" />

</div>

<div>

<label>Courses:</label>

<asp:CheckBox ID="csit" runat="server" Text="CSIT" />

<asp:CheckBox ID="bbm" runat="server" Text="BBM" />

<asp:CheckBox ID="bim" runat="server" Text="BIM" />

</div>

<div>

<label for="ddlCountry">Country:</label>

<asp:DropDownList ID="ddlCountry" runat="server">

<asp:ListItem Text="Select Country" Value="" />

<asp:ListItem Text="Nepal" Value="Nepal" />

<asp:ListItem Text="USA" Value="USA" />

<asp:ListItem Text="India" Value="India" />

</asp:DropDownList>

</div>

<div>

<asp:Button ID="btnSubmit" runat="server" Text="Submit" OnClick="btnSubmit\_Click" />

</div>

</form>

</body>

</html>

**Registration.aspx.cs**

using System;

using System.Data.SqlClient;

using System.Web.UI;

namespace WebApplication7

{

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

{

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

{

}

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

{

// Retrieve form data

string username = txtUsername.Text.Trim();

string password = txtPassword.Text.Trim();

string repassword = txtRepassword.Text.Trim();

string gender = rbMale.Checked ? "Male" : rbFemale.Checked ? "Female" : "";

// Collect selected courses

string courses = "";

if (csit.Checked) courses += "CSIT, ";

if (bbm.Checked) courses += "BBM, ";

if (bim.Checked) courses += "BIM, ";

courses = courses.TrimEnd(',', ' ');

string country = ddlCountry.SelectedValue;

// Validate inputs

if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password) || string.IsNullOrEmpty(repassword))

{

Response.Write("<script>alert('All fields are required!');</script>");

return;

}

if (password != repassword)

{

Response.Write("<script>alert('Passwords do not match!');</script>");

return;

}

if (string.IsNullOrEmpty(gender))

{

Response.Write("<script>alert('Please select a gender!');</script>");

return;

}

if (string.IsNullOrEmpty(courses))

{

Response.Write("<script>alert('Please select at least one course!');</script>");

return;

}

if (string.IsNullOrEmpty(country))

{

Response.Write("<script>alert('Please select a country!');</script>");

return;

}

// Database connection

string connString = "Data Source=DESKTOP-R5GKI2K\\SQLEXPRESS;Initial Catalog=db\_nccsa;Integrated Security=True;TrustServerCertificate=True";

try

{

using (SqlConnection conn = new SqlConnection(connString))

{

conn.Open();

// Random user ID for simplicity

Random rand = new Random();

int randomId = rand.Next(10000, 99999);

string query = "INSERT INTO tbl\_users (id, username, password, gender, course, country) " +

"VALUES (@id, @username, @password, @gender, @course, @country)";

using (SqlCommand cmd = new SqlCommand(query, conn))

{

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

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

cmd.Parameters.AddWithValue("@password", password); // Only store the password

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

cmd.Parameters.AddWithValue("@course", courses);

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

int rows = cmd.ExecuteNonQuery();

if (rows > 0)

{

Response.Write("<script>alert('Registration Successful!');</script>");

}

else

{

Response.Write("<script>alert('Error in registration!');</script>");

}

}

}

}

catch (Exception ex)

{

Response.Write("<script>alert('Error: " + ex.Message + "');</script>");

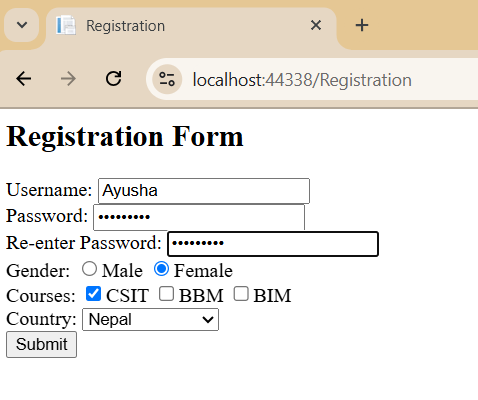
}

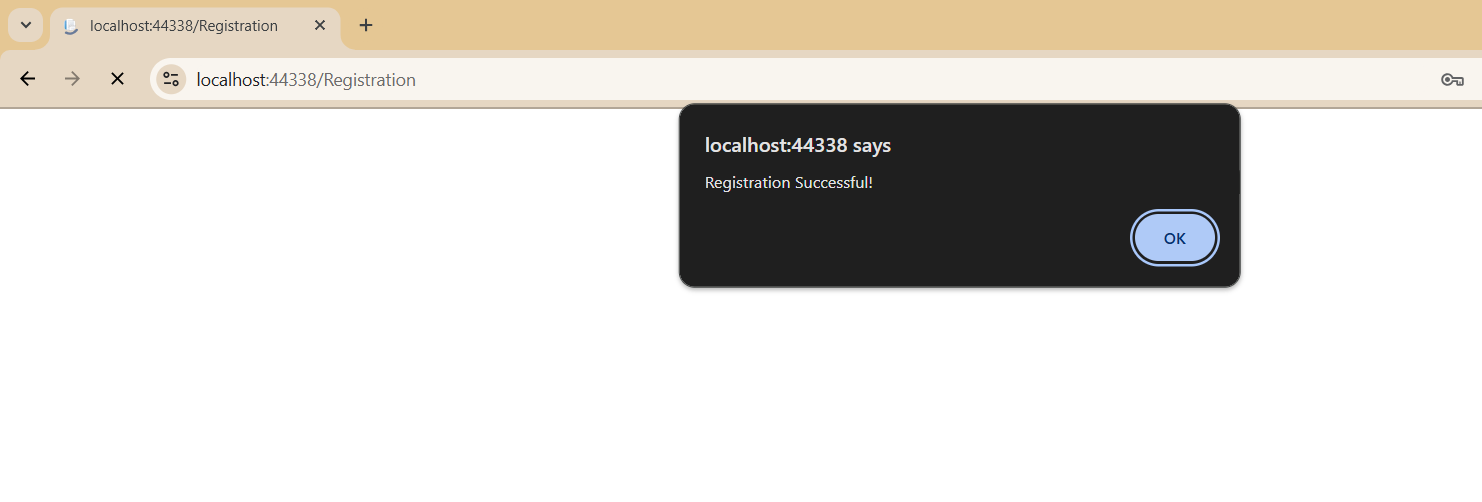
}

}

}

**Output:**

****

****

1. **Demonstrate model, view and controller by showing different action method, views, model, accessing controller, model and view**.

**Model:**

namespace Practical3.Models

{

public class Book

{

public int Id { get; set; }

public string Title { get; set; }

public string Author { get; set; }

public decimal Price { get; set; }

}

}

**Controller:**

using Microsoft.AspNetCore.Mvc;

using Practical3.Models;

namespace Practical3.Controllers

{

public class BooksController : Controller

{

// Action Method to return the list of books

public IActionResult Index()

{

List<Book> books = new List<Book>

{

new Book { Id = 1, Title = "C# Programming", Author = "Ram Sharma", Price = 29.99M },

new Book { Id = 2, Title = "Learning ASP.NET Core", Author = "Janeshia Shrestha", Price = 34.99M },

new Book { Id = 3, Title = "Mastering MVC", Author = "James Lamichane", Price = 39.99M }

};

return View(books); // Returning the View with the list of books

}

// Action Method to return details of a single book

public IActionResult Details(int id)

{

Book book = new Book

{

Id = id,

Title = "Sample Book",

Author = "Sample Author",

Price = 19.99M

};

return View(book); // Returning the View with a single book

}

}

}

**Views:**

**Index.cshtml**

<!-- Views/Books/Index.cshtml -->

@model List<Practical3.Models.Book>

<h1>Books List</h1>

<table>

<thead>

<tr>

<th>Title</th>

<th>Author</th>

<th>Price</th>

<th>Details</th>

</tr>

</thead>

<tbody>

@foreach (var book in Model)

{

<tr>

<td>@book.Title</td>

<td>@book.Author</td>

<td>@book.Price.ToString("C")</td>

<td>

<a href="@Url.Action("Details", new { id = book.Id })">View Details</a>

</td>

</tr>

}

</tbody>

</table>

**Details.cshtml:**

<!-- Views/Books/Details.cshtml -->

@model Practical3.Models.Book

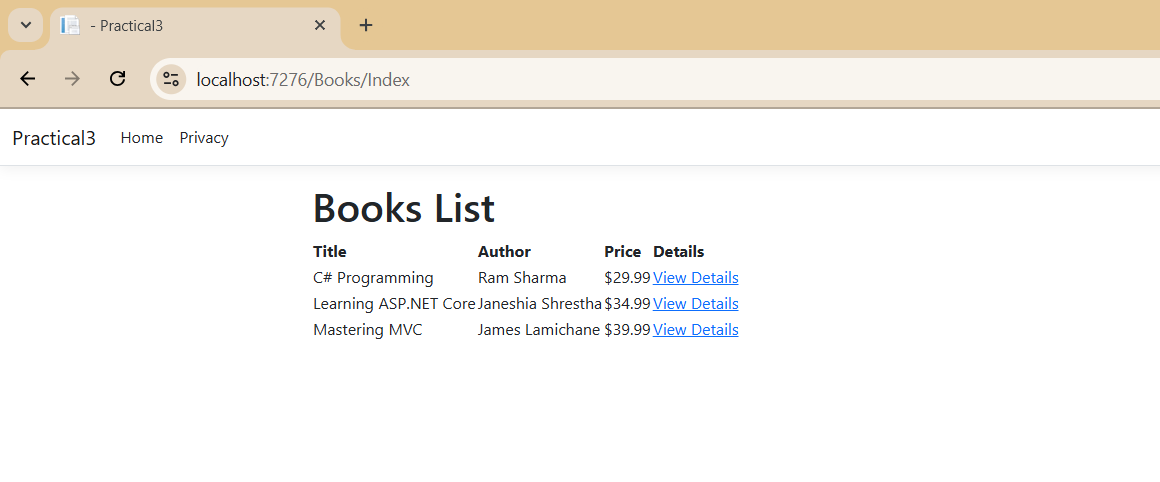
<h1>@Model.Title</h1>

<p><strong>Author:</strong> @Model.Author</p>

<p><strong>Price:</strong> @Model.Price.ToString("C")</p>

<a href="@Url.Action("Index")">Back to List</a>

**Output:**

****

1. **Demonstrate use of razor syntax**

**Model:**

namespace Practical3.Models

{

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

public bool InStock { get; set; }

}

}

**Controller:**

using Microsoft.AspNetCore.Mvc;

using Practical3.Models;

namespace Practical3.Controllers

{

public class ProductsController : Controller

{

public IActionResult Index()

{

var products = new List<Product>

{

new Product { Id = 1, Name = "Laptop", Price = 750.00M, InStock = true },

new Product { Id = 2, Name = "Smartphone", Price = 500.00M, InStock = false },

new Product { Id = 3, Name = "Headphones", Price = 150.00M, InStock = true }

};

return View(products);

}

}

}

**View:**

@model List<Practical3.Models.Product>

<h1>Product List</h1>

<p>Total Products: @Model.Count</p>

@if (Model.Count > 0)

{

<ul>

@foreach (var product in Model)

{

<li>

@product.Name - @product.Price.ToString("C")

@if (product.InStock)

{

<span>(In Stock)</span>

}

else

{

<span>(Out of Stock)</span>

}

</li>

}

</ul>

}

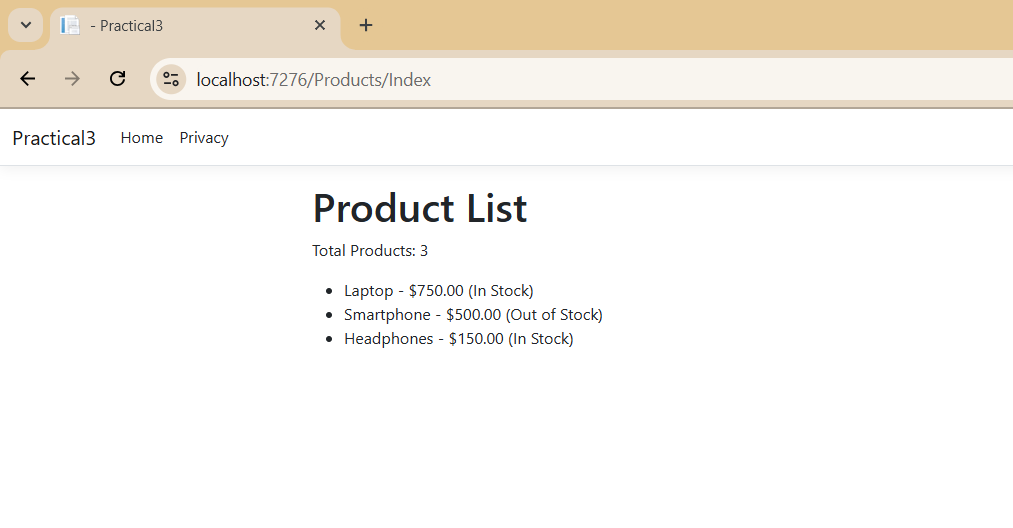
else

{

<p>No products available.</p>

}

**Output:**

****

1. **Demonstrate use of html tag helper**

**Model:**

namespace Lab3.Models

{

public class Service

{

public int Id { get; set; }

public string Name { get; set; }

public decimal Cost { get; set; }

}

}

**Controller:**

using Microsoft.AspNetCore.Mvc;

using Lab3.Models;

using System.Collections.Generic;

namespace Lab3.Controllers

{

public class ServiceController : Controller

{

private static List<Service> services = new List<Service>();

// GET: Service/Create

public IActionResult Create()

{

return View();

}

// POST: Service/Create

[HttpPost]

public IActionResult Create(Service service)

{

if (ModelState.IsValid)

{

service.Id = services.Count + 1;

services.Add(service);

return RedirectToAction("Index");

}

return View(service);

}

// GET: Service/Index

public IActionResult Index()

{

return View(services);

}

}

}

**Views:**

**Index.cshtml**

@model IEnumerable<Lab3.Models.Service>

<!DOCTYPE html>

<html>

<head>

<title>Service List</title>

</head>

<body>

<h1>Service List</h1>

<table border="1">

<thead>

<tr>

<th>ID</th>

<th>Name</th>

<th>Cost</th>

</tr>

</thead>

<tbody>

@foreach (var service in Model)

{

<tr>

<td>@service.Id</td>

<td>@service.Name</td>

<td>@service.Cost.ToString("C")</td>

</tr>

}

</tbody>

</table>

<**a** **asp-action**="Create">Add New Service</**a**>

</body>

</html>

**Create.cshtml:**

@model Lab3.Models.Service

<!DOCTYPE html>

<html>

<head>

<title>Create Service</title>

</head>

<body>

<h1>Create New Service</h1>

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

<div>

<**label** **asp-for**="Name">Service Name</**label**>

<**input** **asp-for**="Name" **type**="text" />

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

</div>

<div>

<**label** **asp-for**="Cost">Cost</**label**>

<**input** **asp-for**="Cost" **type**="number" step="0.01" />

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

</div>

<div>

<button type="submit">Add Service</button>

</div>

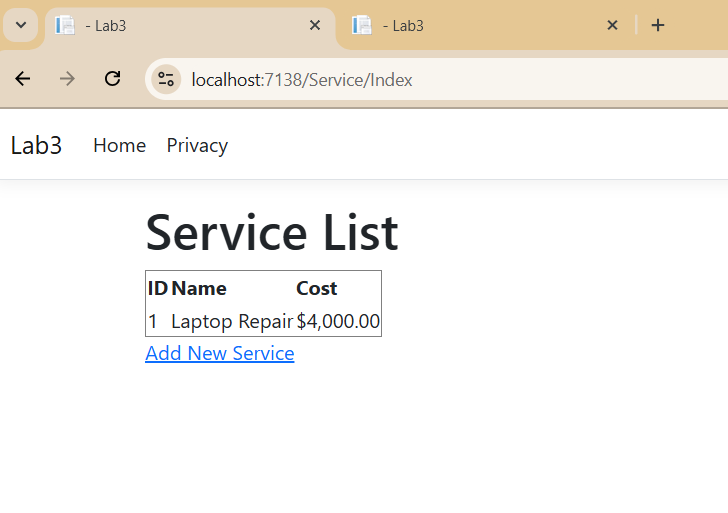
</**form**>

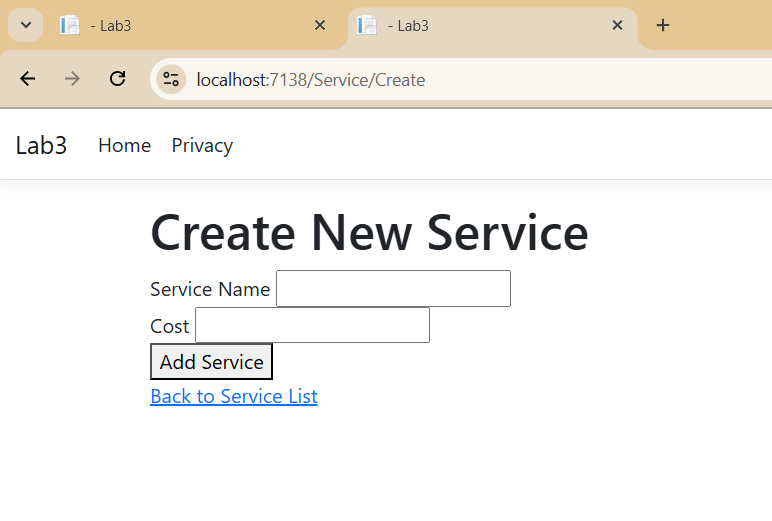
<**a** **asp-action**="Index">Back to Service List</**a**>

</body>

</html>

**Output:**

****

****

1. **Using Entity framework create a table tbl\_officer having field (id, name, gender, phone, department and position) after this perform complete CRUDE operation (insert, update, display and delete). User proper validation.**

**Models:**

**Officer.cs**

using System;

using System.ComponentModel.DataAnnotations;

namespace lab4.Models

{

public class Officer

{

[Key]

public int Id { get; set; } // Primary Key (Auto-increment)

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

[StringLength(100, ErrorMessage = "Name cannot exceed 100 characters")]

public string Name { get; set; }

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

[RegularExpression("^(Male|Female|Other)$", ErrorMessage = "Gender must be Male, Female, or Other")]

public string Gender { get; set; }

[Required(ErrorMessage = "Phone number is required")]

[Phone(ErrorMessage = "Invalid phone number")]

public string Phone { get; set; }

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

[StringLength(50, ErrorMessage = "Department name cannot exceed 50 characters")]

public string Department { get; set; }

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

[StringLength(50, ErrorMessage = "Position name cannot exceed 50 characters")]

public string Position { get; set; }

}

}

**ApplicationDbContext.cs**

using Microsoft.EntityFrameworkCore;

namespace lab4.Models

{

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

public DbSet<Officer> Officers { get; set; }

}

}

**Controller:**

**officeController.cs**

using Microsoft.AspNetCore.Mvc;

using lab4.Models;

using Microsoft.EntityFrameworkCore;

using System.Threading.Tasks;

namespace lab4.Controllers

{

public class OfficerController : Controller

{

private readonly ApplicationDbContext \_context;

public OfficerController(ApplicationDbContext context)

{

\_context = context;

}

public async Task<IActionResult> Index()

{

var officers = await \_context.Officers.ToListAsync();

return View(officers);

}

public IActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(Officer officer)

{

if (ModelState.IsValid)

{

\_context.Add(officer);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(officer);

}

public async Task<IActionResult> Edit(int id)

{

var officer = await \_context.Officers.FindAsync(id);

if (officer == null) return NotFound();

return View(officer);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, Officer officer)

{

if (id != officer.Id) return NotFound();

if (ModelState.IsValid)

{

\_context.Update(officer);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(officer);

}

public async Task<IActionResult> Delete(int id)

{

var officer = await \_context.Officers.FindAsync(id);

if (officer == null) return NotFound();

return View(officer);

}

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var officer = await \_context.Officers.FindAsync(id);

if (officer != null)

{

\_context.Officers.Remove(officer);

await \_context.SaveChangesAsync();

}

return RedirectToAction(nameof(Index));

}

}

}

**Views:  
Index.cshtml**

@model IEnumerable<lab4.Models.Officer>

<h2>Officers List</h2>

<a class="btn btn-success" asp-action="Create">Add Officer</a>

<table class="table">

<thead>

<tr>

<th>Name</th>

<th>Gender</th>

<th>Phone</th>

<th>Department</th>

<th>Position</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

@foreach (var officer in Model)

{

<tr>

<td>@officer.Name</td>

<td>@officer.Gender</td>

<td>@officer.Phone</td>

<td>@officer.Department</td>

<td>@officer.Position</td>

<td>

<a asp-action="Edit" asp-route-id="@officer.Id" class="btn btn-warning btn-sm">Edit</a>

<a asp-action="Delete" asp-route-id="@officer.Id" class="btn btn-danger btn-sm">Delete</a>

</td>

</tr>

}

</tbody>

</table>

**Create.cshtml:**

@model lab4.Models.Officer

<h2>Add Officer</h2>

<**form** **asp-action**="Create">

<label>Name</label>

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

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

<label>Gender</label>

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

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

<label>Phone</label>

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

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

<label>Department</label>

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

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

<label>Position</label>

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

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

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

</**form**>

**Edit.cshtml:**

@model lab4.Models.Officer

<h2>Edit Officer</h2>

<**form** **asp-action**="Edit">

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

<label>Name</label>

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

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

<label>Gender</label>

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

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

<label>Phone</label>

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

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

<label>Department</label>

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

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

<label>Position</label>

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

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

<button type="submit" class="btn btn-primary">Save</button>

</**form**>

**Delete.cshtml**

@model lab4.Models.Officer

<h2>Delete Officer</h2>

<h3>Are you sure you want to delete this officer?</h3>

<div>

<p><strong>Name:</strong> @Model.Name</p>

<p><strong>Gender:</strong> @Model.Gender</p>

<p><strong>Phone:</strong> @Model.Phone</p>

<p><strong>Department:</strong> @Model.Department</p>

<p><strong>Position:</strong> @Model.Position</p>

<**form** **asp-action**="Delete" method="post">

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

<button type="submit" class="btn btn-danger">Delete</button>

<**a** **asp-action**="Index" class="btn btn-secondary">Cancel</**a**>

</**form**>

</div>

**appsettings.json:**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

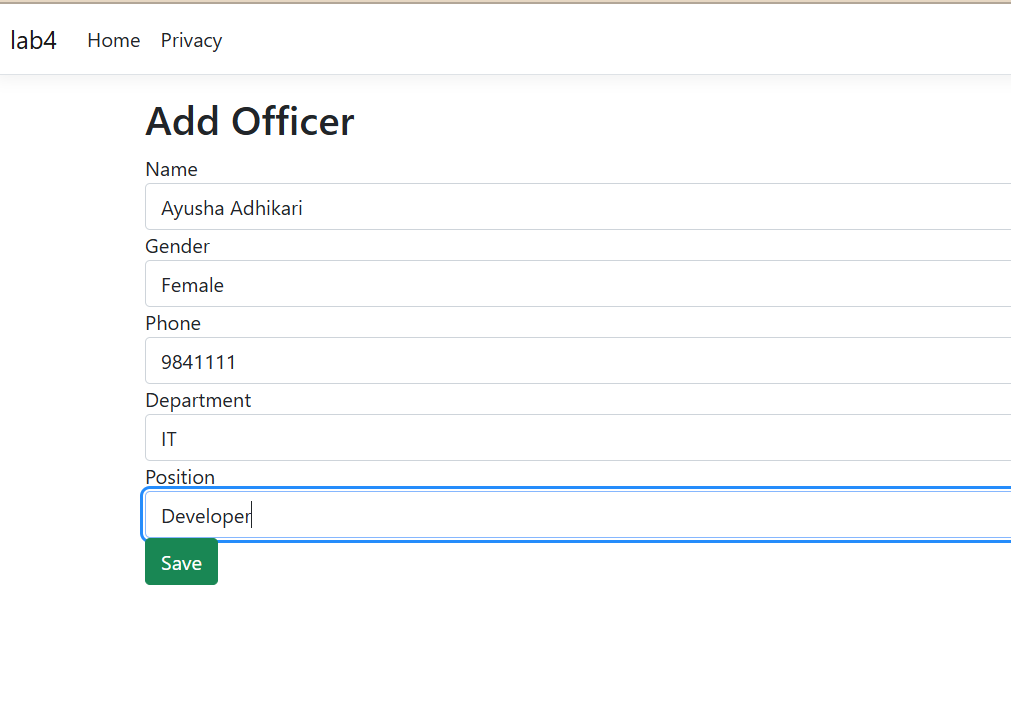
"AllowedHosts": "\*",

"ConnectionStrings": {

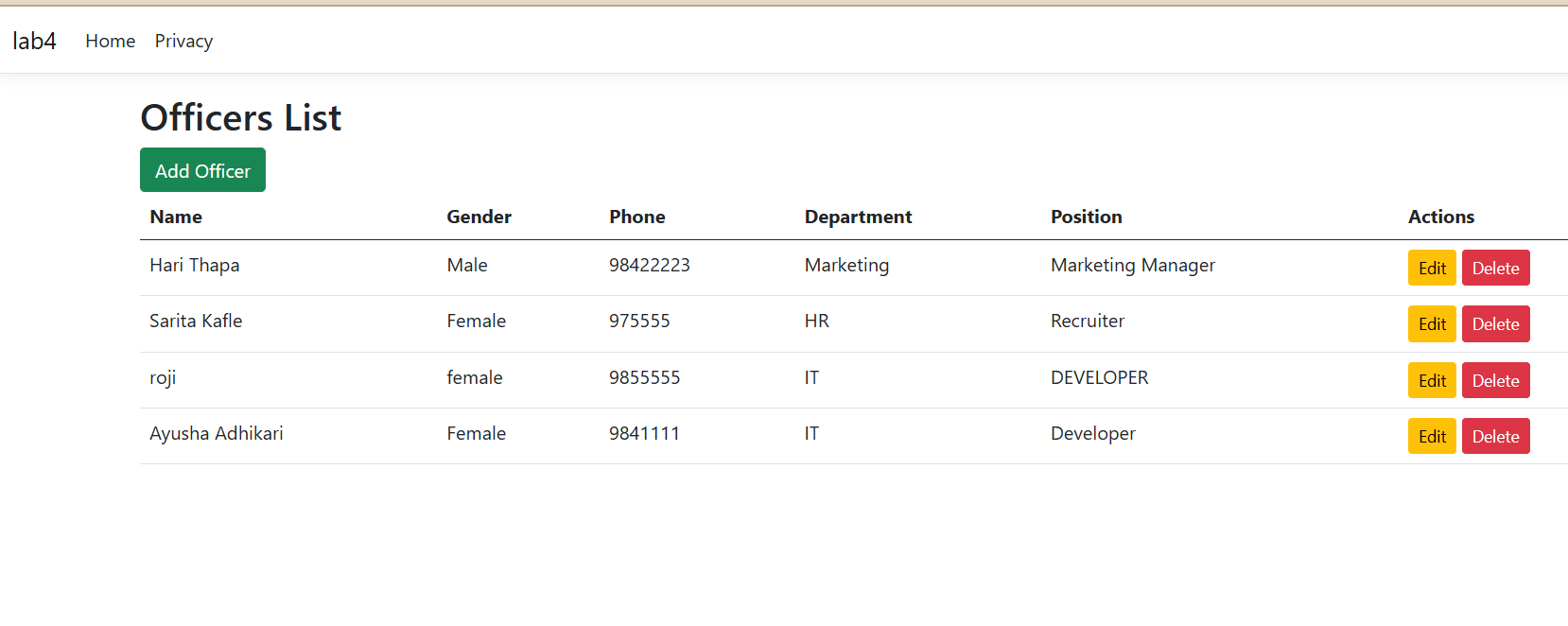
"DefaultConnection": "Server=DESKTOP-R5GKI2K\\SQLEXPRESS;Database=db\_nccsa;Trusted\_Connection=True;MultipleActiveResultSets=true;TrustServerCertificate=True"

}}

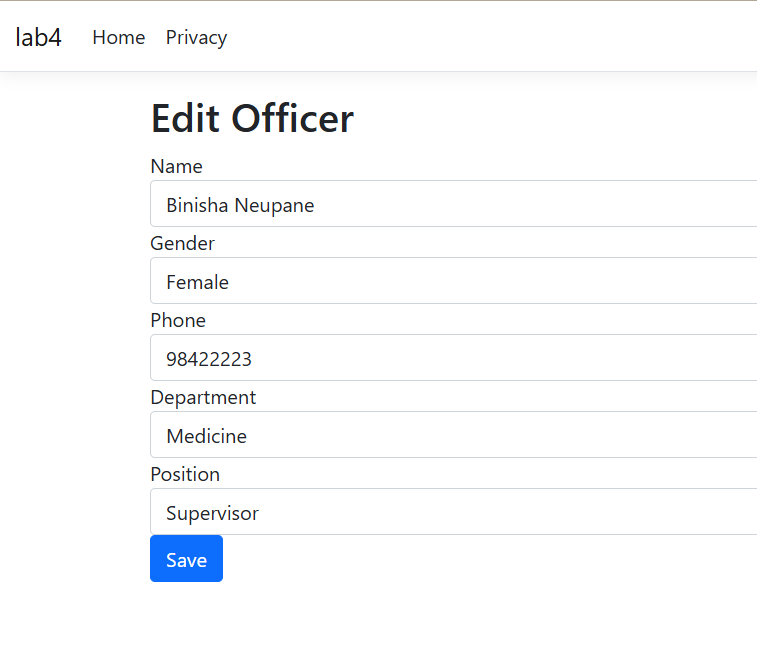
**Add Officer:**

****

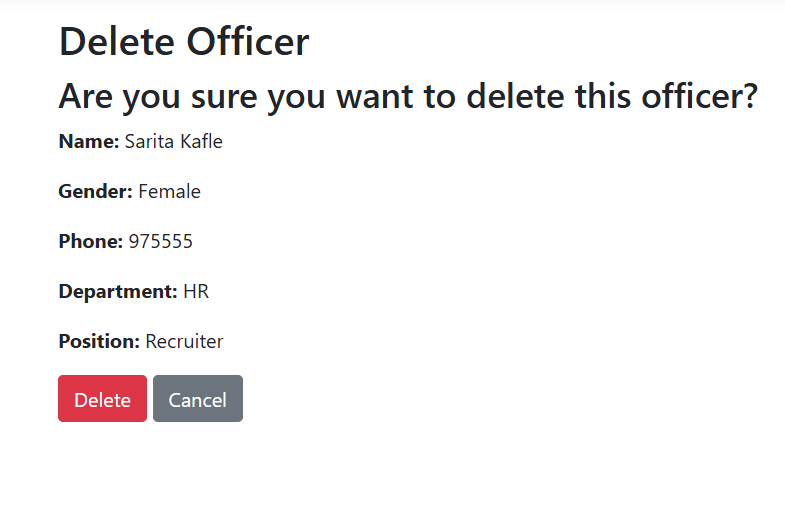
**Display officer**

****

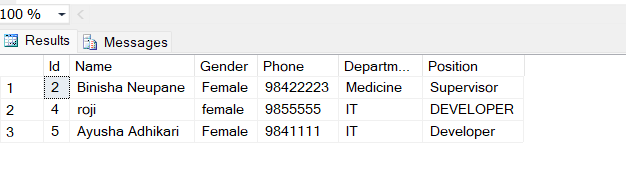
**Editing officer:**

****

**Deleting:**

****

**Database:**

****

**5. Demonstrate different state management technique like SessionState, TempData, HttpContext**

**Controller:**

**StateController.cs:**

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace lab4.Controllers

{

public class StateController : Controller

{

public IActionResult Index()

{

return View();

}

// Set session and TempData

public IActionResult SetSession()

{

HttpContext.Session.SetString("User", "Ayusha");

TempData["Message"] = "Session value has been set!";

return RedirectToAction("GetSession");

}

// Retrieve session and TempData

public IActionResult GetSession()

{

string user = HttpContext.Session.GetString("User");

ViewBag.User = user ?? "No session found!";

ViewBag.Message = TempData["Message"];

return View("SessionView");

}

// TempData demo (using ViewData, ViewBag, and TempData)

public IActionResult TempDataDemo()

{

ViewData["data1"] = "this is data from view data (Ayusha)";

ViewBag.data2 = "this is data from view bag (CSIT)";

TempData["data3"] = "data from temp data (Tokha, Kathmandu)";

return View();

}

}

}

**Views:**

**SessionView.cshtml**

@{

ViewData["Title"] = "Session Management";

}

<h2>Session Data</h2>

@if (ViewBag.Message != null)

{

<p><strong style="color: green;">@ViewBag.Message</strong></p>

}

<p>Stored Session Value: <strong>@ViewBag.User</strong></p>

<a asp-controller="State" asp-action="SetSession">Set Session</a> |

<a asp-controller="State" asp-action="GetSession">Get Session</a>

**TempDataDemo.cshtml:**

<h2>data are</h2>

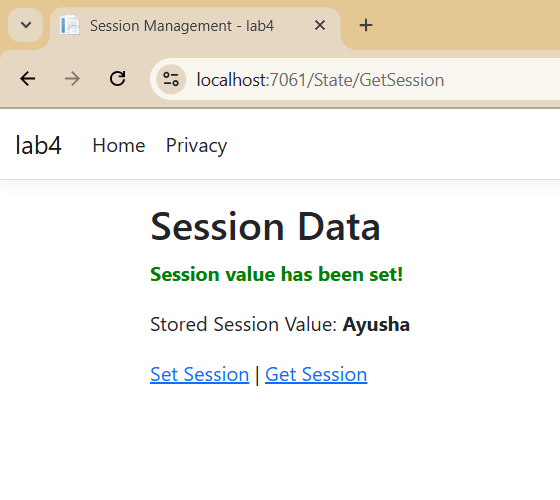
<h2>@ViewData["data1"]</h2>

<h2>@ViewBag.data2</h2>

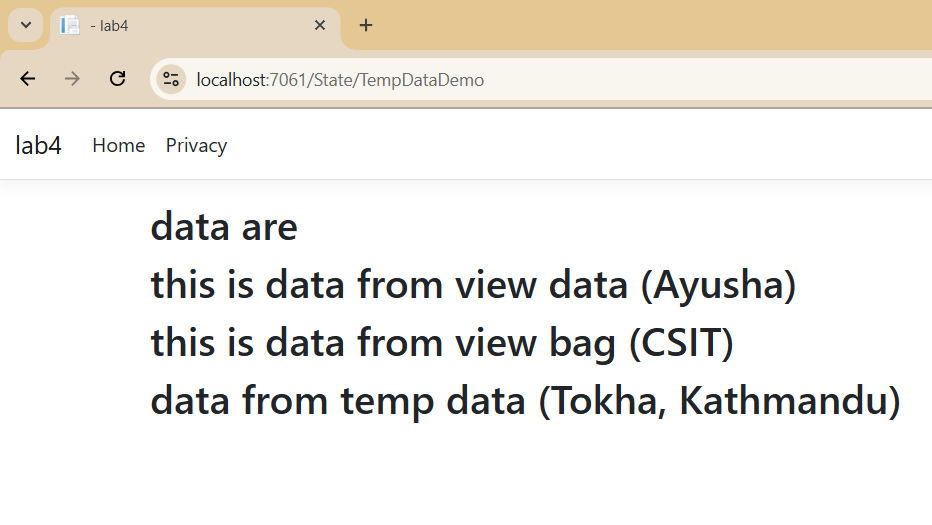
<h2>@TempData["data3"]</h2>

**Output:**

**Session**

****

**TempData:**

****

1. **Demonstrate different client-side state management like cookies, Query string and hidden fields**

**Session:**

**SessionControllerDemo.cs**

using Microsoft.AspNetCore.Mvc;

namespace StateManagement.Controllers

{

public class StateControllerDemo : Controller

{

public IActionResult SetCookie()

{

var options = new CookieOptions

{

Expires = DateTime.Now.AddMinutes(10),

Secure = true,

SameSite = SameSiteMode.Lax,

Path = "/"

};

Response.Cookies.Append("UserName", "Ayusha", options);

ViewBag.Message = "Cookie has been set!";

return View("ClientStateView");

}

public IActionResult GetCookie()

{

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

ViewBag.Message = userName ?? "No cookie found!";

return View("ClientStateView");

}

public IActionResult QueryStringExample(string name, int age)

{

ViewBag.Message = $"Name: {name}, Age: {age}";

return View("ClientStateView");

}

[HttpPost]

public IActionResult SubmitHidden(string HiddenData)

{

ViewBag.Message = "Hidden Field Value: " + HiddenData;

return View("ClientStateView");

}

public IActionResult Index()

{

return View();

}

}

}

**View:**

**ClientStateView.cshtml:**

@{

ViewData["Title"] = "Client-Side State Management";

}

<h2>Client-Side State Management</h2>

@if (ViewBag.Message != null)

{

<p><strong style="color: green;">@ViewBag.Message</strong></p>

}

<!-- Set & Get Cookies -->

<**a** **asp-controller**="StateControllerDemo" **asp-action**="SetCookie">Set Cookie</**a**>

<**a** **asp-controller**="StateControllerDemo" **asp-action**="GetCookie">Get Cookie</**a**>

<!-- Query String Example -->

<br>

<a href="@Url.Action("QueryStringExample", "StateControllerDemo", new { name = "Ayusha", age = 21 })">

Send Data Using Query String

</a>

<!-- Hidden Field Example -->

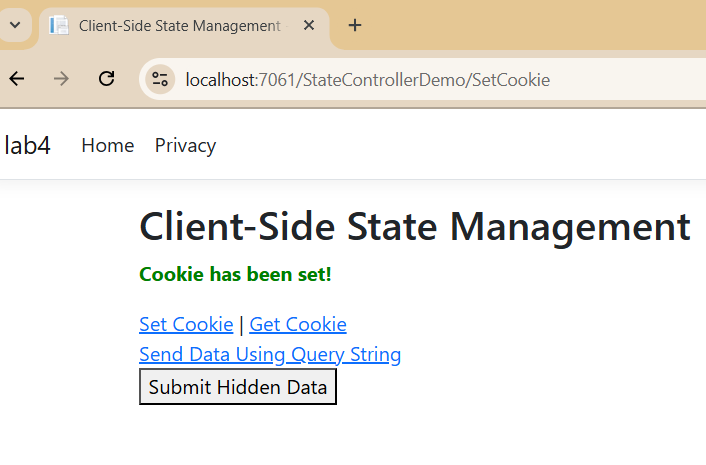
<**form** method="post" **asp-action**="SubmitHidden">

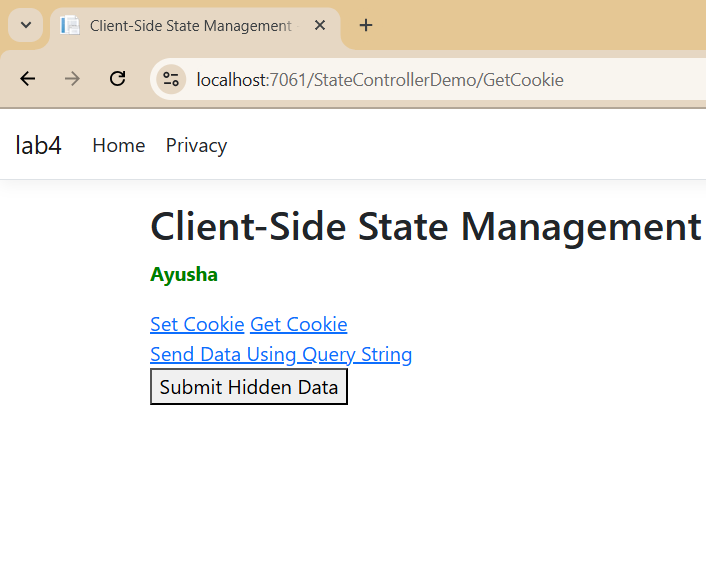
<input type="hidden" name="HiddenData" value="SecretValue123" />

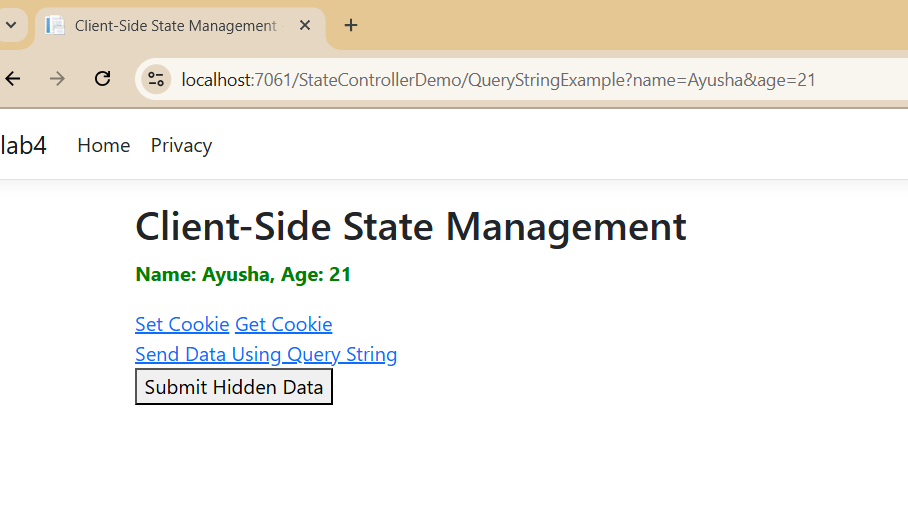
<button type="submit">Submit Hidden Data</button>

</**form**>

**Output:**

****

****

****

1. **Write a program to create complete form and validate using jquery and react.**

**FormComponent.js**

// src/FormComponent.js

import React, { useState } from "react";

import $ from "jquery";

const FormComponent = () => {

  const [formData, setFormData] = useState({

    name: "",

    email: "",

    password: "",

    rememberMe: false,

  });

  const handleChange = (e) => {

    const { name, value, type, checked } = e.target;

    setFormData({

      ...formData,

      [name]: type === "checkbox" ? checked : value,

    });

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    // Clear previous validation messages

    $(".error").remove();

    // Validate form fields

    let isValid = true;

    if (!formData.name) {

      isValid = false;

      $("#name").after('<span class="error">Name is required</span>');

    }

    if (!formData.email) {

      isValid = false;

      $("#email").after('<span class="error">Email is required</span>');

    } else if (!/\S+@\S+\.\S+/.test(formData.email)) {

      isValid = false;

      $("#email").after('<span class="error">Email is invalid</span>');

    }

    if (!formData.password) {

      isValid = false;

      $("#password").after('<span class="error">Password is required</span>');

    }

    if (formData.password.length < 6) {

      alert("Password must be of at least 6 character");

    }

    if (isValid) {

      alert("Form submitted successfully!");

      // Here you can handle form submission, e.g., send data to the server

    }

  };

  return (

    <form onSubmit={handleSubmit}>

      <div>

        <label htmlFor="name">Name:</label>

        <input

          type="text"

          id="name"

          name="name"

          value={formData.name}

          onChange={handleChange}

        />

      </div>

      <div>

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

        <input

          type="email"

          id="email"

          name="email"

          value={formData.email}

          onChange={handleChange}

        />

      </div>

  <div>

        <label htmlFor="password">Password:</label>

        <input

          type="password"

          id="password"

          name="password"

          value={formData.password}

          onChange={handleChange}

        />

</div>

      <div>

        <label>

          <input

            type="checkbox"

            name="rememberMe"

            checked={formData.rememberMe}

            onChange={handleChange}

          />

          Remember Me

        </label>

      </div>

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

    </form>

  );

};

export default FormComponent;

**App.js**

import React from "react";

import FormComponent from "./FormComponent";

const App = () => {

  return (

    <div>

      <h1>This is Practical 3.7</h1>

      <FormComponent />

    </div>

  );

};

export default App;

**OUTPUT**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **Write a program to demonstrate authentication and authorization (Role, claim and policies) by create a complete form in asp.net core.**

**Controller**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Authentication;

using Microsoft.AspNetCore.Authentication.Cookies;

using System.Security.Claims;

using Microsoft.AspNetCore.Authorization;

namespace WebApplication6.Controllers

{

public class AuthenticationController : Controller

{

public IActionResult Login()

{

return View();

}

[HttpPost]

public async Task<IActionResult> Login(string username, string password, string returnUrl)

{

if (username == "admin" && password == "password")

{

List<Claim> claims = new List<Claim>

{

new Claim(ClaimTypes.Name, username),

new Claim(ClaimTypes.Role, "Admin")

};

ClaimsIdentity claimsIdentity = new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme);

ClaimsPrincipal claimsPrincipal = new ClaimsPrincipal(claimsIdentity);

await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, claimsPrincipal);

if (string.IsNullOrEmpty(returnUrl))

{

returnUrl = Url.Action("Dashboard", "Authentication");

}

return Redirect(returnUrl);

}

ViewBag.ErrorMessage = "Invalid username or password.";

return View();

}

public async Task<IActionResult> Logout()

{

await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);

return RedirectToAction("Login");

}

[Authorize]

public IActionResult Dashboard()

{

return View();

}

}

}

**View (Login.cshtml)**

@{

ViewData["Title"] = "Login";

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

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

</head>

<body>

<div>

<h2>Login</h2>

<form method="post">

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

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

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

<input type="password" name="password" required /><br />

<input type="submit" value="Login" />

</form>

@if (ViewBag.ErrorMessage != null)

{

<div>@ViewBag.ErrorMessage</div>

}

</div>

</body>

</html>

**View (Dashboard)**

@{

ViewData["Title"] = "Dashboard";

}

<h2>Welcome to the Dashboard!</h2>

<p>You are logged in as @User.Identity.Name</p>

<a href="/Authentication/Logout">Logout</a>

**Program.cs**

**Adding the authentication and authorization methods in the builder options**

using Microsoft.AspNetCore.Authentication.Cookies;

using Microsoft.AspNetCore.Mvc;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)

.AddCookie(options =>

{

options.LoginPath = "/Authentication/Login";

options.LogoutPath = "/Authentication/Logout";

});

builder.Services.AddAuthorization(options =>

{

options.AddPolicy("Admin", policy => policy.RequireRole("Admin"));

});

builder.Services.AddControllersWithViews();

var app = builder.Build();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllerRoute(

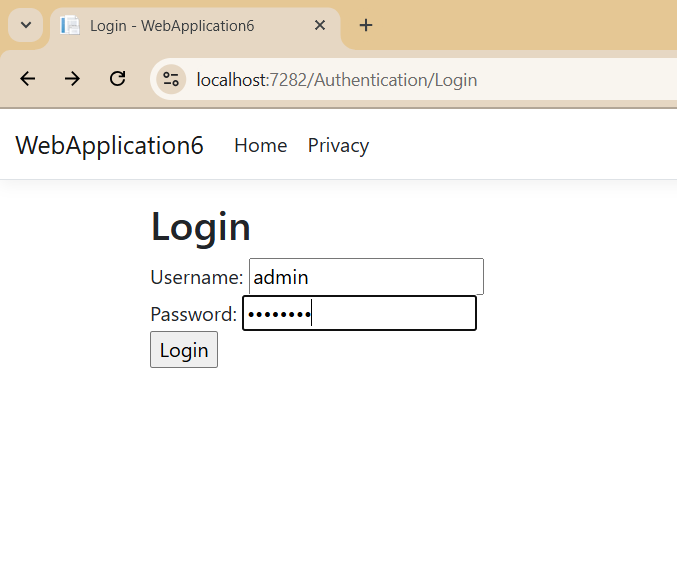
name: "default",

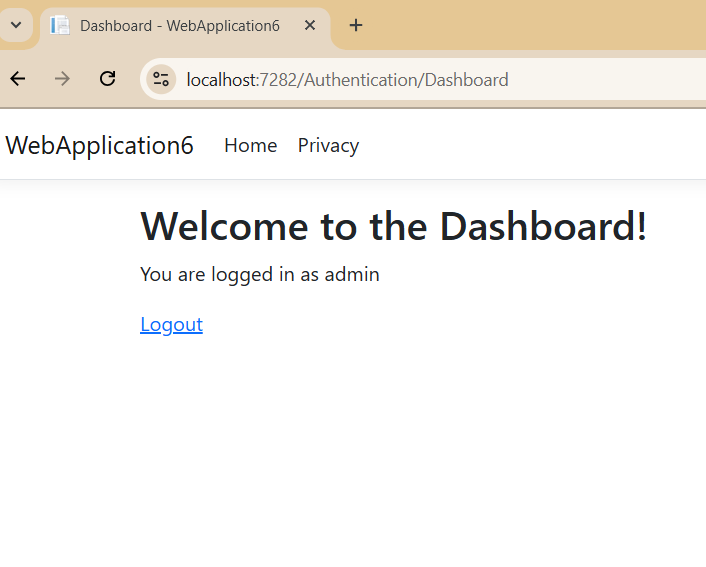
pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();

**OUTPUT**

**Login into Contact with correct credentials**





1. **Write a program to prevent SQLInjectionAttack, Cross Site Request forgery (CSRF) and open redirect attack**

**Controller:**

using Microsoft.AspNetCore.Mvc;

namespace SQLInjectionAttack.Controllers

{

public class SafeController : Controller

{

[HttpGet]

public IActionResult Index()

{

ViewData["Title"] = "Secure Form";

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public IActionResult Index(string userInput)

{

if (string.IsNullOrWhiteSpace(userInput))

{

ViewBag.Message = "Input cannot be empty.";

return View();

}

string safeInput = System.Web.HttpUtility.HtmlEncode(userInput);

ViewBag.Message = "Processed Input: " + safeInput;

return View();

}

public IActionResult SafeRedirect(string returnUrl)

{

Uri redirectUri;

if (Uri.TryCreate(returnUrl, UriKind.RelativeOrAbsolute, out redirectUri))

{

if (!redirectUri.IsAbsoluteUri || redirectUri.Host == Request.Host.Host)

{

return Redirect(returnUrl);

}

}

return RedirectToAction("Index");

}

}

}

**View:**

@{

// Optionally, ViewData["Title"] could be set here, but it's already set in the controller.

}

<h2>@ViewData["Title"]</h2> <!-- Access ViewData["Title"] set in the controller -->

<**form** method="post" action="/Safe/Index">

@Html.AntiForgeryToken() <!-- CSRF Token -->

<label>Enter Text:</label>

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

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

</**form**>

@if (ViewBag.Message != null)

{

<p style="color: green">@ViewBag.Message</p>

}

<hr>

<h3>Try Unsafe Redirect</h3>

<**form** action="/Safe/SafeRedirect" method="get">

<input type="text" name="returnUrl" placeholder="Enter redirect URL">

<button type="submit">Click to Try Unsafe Redirect</button>

</**form**>

**appsettings.json**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"DefaultConnection": "Server=DESKTOP-R5GKI2K\\SQLEXPRESS;Database=db\_nccsa;Trusted\_Connection=True;"

}

}

**Output:**

