**Common Process :**

**Install Sql server- > Make connection with Visual Studio -> Insert cde in files.**

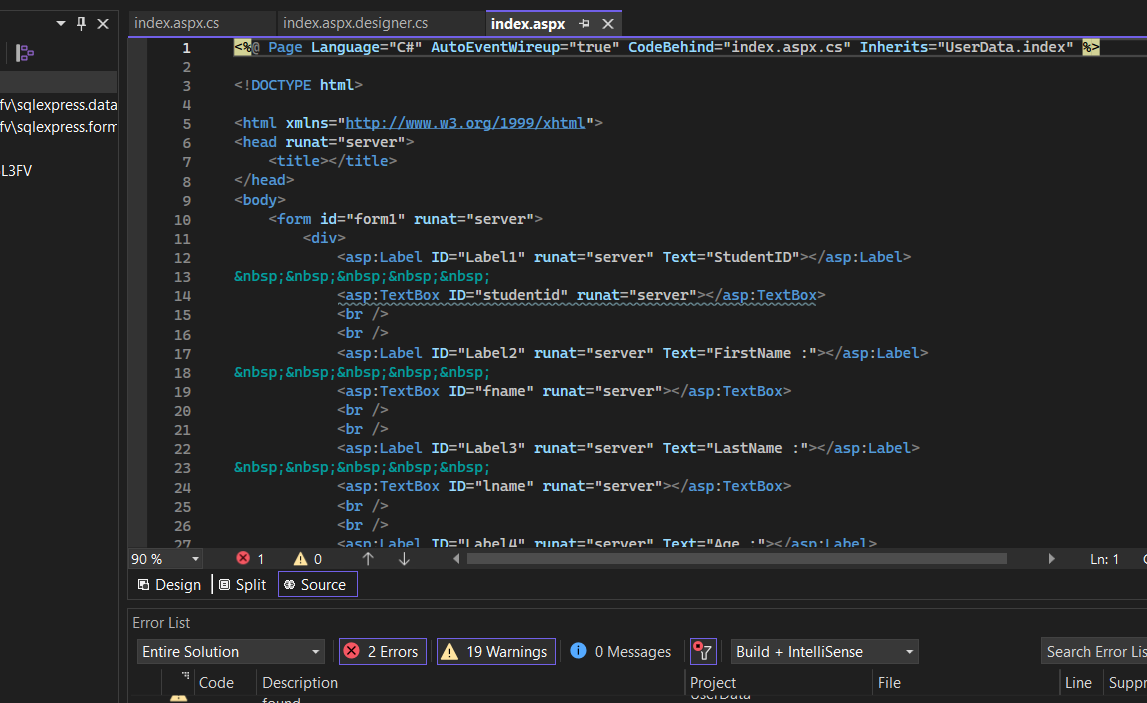
**Template : ASP.Net WebApplication(.Net Framework) -> Next -> Create ->Empty -> Next/Create/Finish.**

**After opening the window, in the “Solution Explorer” right click on the name of project -> Add -> WebForms -> (Rename as index).**

**In file\_name.aspx include HTML code.**

**In file\_name.aspx.cs include C# code**

**To open the file of C# code, Go on Design option**





**Take the tools like label, textbox from toolbox (Drag and drop). Double click on the component makes its C# code visible.**

Change the name of your files and projects accordingly in the code in order to avoid errors.

Statements.

**Q1. Create a database named "StudentDB" with a table named "Students" containing fields: StudentID (int, primary key), FirstName (varchar), LastName (varchar), and Age (int). Populate the table with sample data.**

**Q2. Design a web page (e.g., Default.aspx) with the following elements: A GridView to display the list of students from the database. TextBox controls for entering a new student's FirstName, LastName, and Age. Buttons for "Add Student" and "Refresh List."**

**Q3. Develop an ASP.NET web page that allows users to submit a form with their details (name, email, etc.). Store this information in a database using ADO.NET.**

***(All three Statements are covered in single code)***

**index.aspx.cs :**

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace UserData

{

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

{

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

{

}

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

{

try

{

// Retrieve values from the form controls

int studentID = Convert.ToInt32(studentid.Text);

string firstName = fname.Text;

string lastName = lname.Text;

int age = Convert.ToInt32(age1.Text);

// Assuming you have a database connection string in the configuration

string connectionString = "Data Source=localhost\\SQLEXPRESS;Initial Catalog=StudentDB;Integrated Security=True";

using (SqlConnection connection = new SqlConnection(connectionString))

{

// Open the database connection

connection.Open();

// Example: Inserting data into the Students table

string insertQuery = "INSERT INTO Students (StudentID, FirstName, LastName, Age) " +

"VALUES (@studentID, @firstName, @lastName, @age)";

using (SqlCommand command = new SqlCommand(insertQuery, connection))

{

// Add parameters to the SQL command

command.Parameters.AddWithValue("@studentID", studentID);

command.Parameters.AddWithValue("@firstName", firstName);

command.Parameters.AddWithValue("@lastName", lastName);

command.Parameters.AddWithValue("@age", age);

// Execute the SQL command

command.ExecuteNonQuery();

}

// Close the database connection

connection.Close();

// You can add any additional logic or messages here

success1.Text = "Inserted Succesful";

// For example, display success message

// Response.Write("Data added successfully!");

}

}

catch (Exception ex)

{

// Handle the exception, you can display an error message or log it

// For example, display an error message

Response.Write("Error: " + ex.Message);

}

}

protected void Button2\_Click(object sender, EventArgs e)

{

try { GridView1.DataBind();

}

catch (Exception c)

{

Response.Write(c.Message);

}

}

}

}

index.aspx

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

<!DOCTYPE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<asp:Label ID="Label1" runat="server" Text="StudentID"></asp:Label>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

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

<br />

<br />

<asp:Label ID="Label2" runat="server" Text="FirstName :"></asp:Label>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

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

<br />

<br />

<asp:Label ID="Label3" runat="server" Text="LastName :"></asp:Label>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

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

<br />

<br />

<asp:Label ID="Label4" runat="server" Text="Age :"></asp:Label>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<asp:TextBox ID="age1" runat="server"></asp:TextBox>

<br />

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<br />

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

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

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<asp:Button ID="Button2" runat="server" OnClick="Button2\_Click" Text="Refresh List" />

<br />

&nbsp;&nbsp;&nbsp;

<asp:Label ID="success1" runat="server" Text="" Style="color:palegreen"></asp:Label>

<br />

&nbsp;&nbsp;&nbsp;

<asp:Label ID="error" runat="server" Text="" Style="color:red"></asp:Label>

<br />

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

</div>

<div>

<h2>Student Database</h2>

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataKeyNames="StudentID" DataSourceID="SqlDataSource1">

<Columns>

<asp:BoundField DataField="StudentID" HeaderText="StudentID" SortExpression="StudentID" ReadOnly="True" />

<asp:BoundField DataField="FirstName" HeaderText="FirstName" SortExpression="FirstName" />

<asp:BoundField DataField="LastName" HeaderText="LastName" SortExpression="LastName" />

<asp:BoundField DataField="Age" HeaderText="Age" SortExpression="Age" />

</Columns>

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="Data Source=localhost\SQLEXPRESS;Initial Catalog=StudentDB;Integrated Security=True" SelectCommand="SELECT \* FROM [Students]"></asp:SqlDataSource>

</div>

</form>

</body>

</html>

**Q4. Write a code snippet in C# to validate user input in an ASP.NET web form. Include validation for required fields, email format, and password strength.**

**Chashvalidation.aspx**

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

<!DOCTYPE html>

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

<head runat="server">

<title>Validation Demo</title>

<script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>

</head>

<body>

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

<div>

<h2>User Registration</h2>

<asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>

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

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

<asp:RequiredFieldValidator ID="rfvUsername" runat="server" ControlToValidate="txtUsername"

ErrorMessage="Username is required." Display="Dynamic" ForeColor="Red"></asp:RequiredFieldValidator>

<br />

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

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

<asp:RequiredFieldValidator ID="rfvEmail" runat="server" ControlToValidate="txtEmail"

ErrorMessage="Email is required." Display="Dynamic" ForeColor="Red"></asp:RequiredFieldValidator>

<asp:RegularExpressionValidator ID="revEmail" runat="server" ControlToValidate="txtEmail"

ErrorMessage="Invalid email format." Display="Dynamic" ForeColor="Red"

ValidationExpression="\b[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b"></asp:RegularExpressionValidator>

<br />

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

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

<asp:RequiredFieldValidator ID="rfvPassword" runat="server" ControlToValidate="txtPassword"

ErrorMessage="Password is required." Display="Dynamic" ForeColor="Red"></asp:RequiredFieldValidator>

<asp:RegularExpressionValidator ID="revPassword" runat="server" ControlToValidate="txtPassword"

ErrorMessage="Password must be at least 8 characters long and contain at least one uppercase letter, one lowercase letter, and one digit."

Display="Dynamic" ForeColor="Red"

ValidationExpression="^(?=.[a-z])(?=.[A-Z])(?=.\*\d).{8,}$"></asp:RegularExpressionValidator>

<br />

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

<asp:Label ID="lblValidationResult" runat="server" Text="" ForeColor="Green"></asp:Label>

</div>

</form>

</body>

</html>

**Chashvalidation.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Chashvalidation

{

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

{

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

{

}

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

{

if (Page.IsValid)

{

// Your code to process the form data

lblValidationResult.Text = "Form submitted successfully!";

}

else

{

lblValidationResult.Text = "Please fix the validation errors.";

}

}

}

}

**Webconfig file**

<appSettings>

<add key="ValidationSettings:UnobtrusiveValidationMode" value="None" />

</appSettings>

**Q5. Develop a web application using ADO.NET in ASP.NET. Utilize a stored procedure to insert data into a database table. Implement a form in your ASP.NET application to take user inputs and invoke the stored procedure to insert the data.**

**Query :**

-- Create a sample table

CREATE TABLE Users (

UserId INT PRIMARY KEY IDENTITY(1,1),

UserName NVARCHAR(50),

Email NVARCHAR(50)

);

-- Create a stored procedure for inserting data

CREATE PROCEDURE InsertUser

@UserName NVARCHAR(50),

@Email NVARCHAR(50)

AS

BEGIN

INSERT INTO Users (UserName, Email)

VALUES (@UserName, @Email);

END;

**Question5\_WebForm1.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Question\_5.WebForm1" %>

<!DOCTYPE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<label for="txtUserName">User Name:</label>

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

<br />

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

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

<br />

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

<br />

<span id="lblMessage" runat="server"></span>

</div>

</form>

</body>

</html>

**Question5\_WebForm1.aspx.cs**

using System;

using System.Data;

using System.Data.SqlClient;

namespace Question\_5

{

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

{

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

{

}

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

{

string connectionString = "Data Source=localhost\\SQLEXPRESS;Initial Catalog=abc;Integrated Security=True;";

using (SqlConnection connection = new SqlConnection(connectionString))

{

using (SqlCommand cmd = new SqlCommand("InsertUser", connection))

{

cmd.CommandType = CommandType.StoredProcedure;

// Add parameters

cmd.Parameters.AddWithValue("@UserName", txtUserName.Text);

cmd.Parameters.AddWithValue("@Email", txtEmail.Text);

try

{

connection.Open();

cmd.ExecuteNonQuery();

lblMessage.InnerText = "User added successfully!";

ClearForm();

}

catch (Exception ex)

{

lblMessage.InnerText = "Error: " + ex.Message;

}

}

}

}

private void ClearForm()

{

txtUserName.Text = "";

txtEmail.Text = "";

}

}

}

**Q6. Build a responsive ASP.NET web application for blood bank that incorporates Bootstrap. Create a page with a navigation bar, a responsive grid layout, and an image carousel. Ensure that the layout adjusts gracefully to different screen sizes**

**index.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="file4.aspx.cs" Inherits="sandhya1.file4" %>

<!DOCTYPE html>

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

<head runat="server">

<title>Blood Bank</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

</head>

<body>

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

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<a class="navbar-brand" href="#">Blood Bank</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNav">

<ul class="navbar-nav">

<li class="nav-item active">

<a class="nav-link" href="#">Home</a>

</li>

<li class="nav-item">

<a class="nav-link" href="#">Donors</a>

</li>

<li class="nav-item">

<a class="nav-link" href="#">Requests</a>

</li>

<li class="nav-item">

<a class="nav-link" href="#">Contact</a>

</li>

</ul>

</div>

</nav>

<div class="container mt-4">

<div class="row">

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

<h3>Available Blood Types</h3>

<ul>

<li>A+</li>

<li>B+</li>

<li>O+</li>

<li>AB+</li>

</ul>

</div>

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

<h3>Upcoming Events</h3>

<p>Blood Donation Camp - January 15, 2023</p>

<p>Community Outreach Program - February 1, 2023</p>

</div>

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

<h3>Donate Now</h3>

<p>Help save lives by donating blood today. Visit our nearest blood donation center.</p>

</div>

</div>

</div>

<div id="carouselExampleIndicators" class="carousel slide mt-4" data-ride="carousel">

<ol class="carousel-indicators">

<li data-target="#carouselExampleIndicators" data-slide-to="0" class="active"></li>

<li data-target="#carouselExampleIndicators" data-slide-to="1"></li>

<li data-target="#carouselExampleIndicators" data-slide-to="2"></li>

</ol>

<div class="carousel-inner">

<div class="carousel-item active">

<img class="d-block w-100" src="your\_url" alt="Third slide">

</div>

<div class="carousel-item">

<img class="d-block w-100" src="your\_url" alt="Third slide">

</div>

<div class="carousel-item">

<img class="d-block w-100" src="your\_url" alt="Third slide"> </div>

</div>

<a class="carousel-control-prev" href="#carouselExampleIndicators" role="button" data-slide="prev">

<span class="carousel-control-prev-icon" aria-hidden="true"></span>

<span class="sr-only">Previous</span>

</a>

<a class="carousel-control-next" href="#carouselExampleIndicators" role="button" data-slide="next">

<span class="carousel-control-next-icon" aria-hidden="true"></span>

<span class="sr-only">Next</span>

</a>

</div>

</form>

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.2/dist/umd/popper.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.2/dist/js/bootstrap.min.js"></script>

</body>

<head>

**Q7. Develop an ASP.NET form that uses Bootstrap styles. Implement a form with various input elements (textboxes, checkboxes, radio buttons) and apply Bootstrap classes to enhance the visual appeal. Validate user inputs using Bootstrap's built-in validation styles**

**.aspx file**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="file4.aspx.cs" Inherits="sandhya1.file41" %>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<title>Bootstrap Form</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8sh+Wy6nP8eNVhCFmbo2KZ&" crossorigin="anonymous">

</head>

<body>

<div class="container mt-5">

<form runat="server" class="needs-validation" novalidate>

<div class="form-group">

<label for="txtName">Name</label>

<input type="text" class="form-control" id="txtName" runat="server" required />

<div class="invalid-feedback">

Please enter your name.

</div>

</div>

<div class="form-group">

<label>Gender</label>

<div class="custom-control custom-radio">

<input type="radio" id="radMale" name="gender" class="custom-control-input" runat="server" required />

<label class="custom-control-label" for="radMale">Male</label>

</div>

<div class="custom-control custom-radio">

<input type="radio" id="radFemale" name="gender" class="custom-control-input" runat="server" required />

<label class="custom-control-label" for="radFemale">Female</label>

</div>

<div class="invalid-feedback">

Please select your gender.

</div>

</div>

<div class="form-group">

<label for="chkAgree" class="custom-control custom-checkbox">

<input type="checkbox" class="custom-control-input" id="chkAgree" runat="server" required />

<span class="custom-control-indicator"></span>

<span class="custom-control-description">I agree to the terms and conditions</span>

<div class="invalid-feedback">

You must agree to the terms and conditions.

</div>

</label>

</div>

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

</form>

</div>

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.3/dist/umd/popper.min.js" integrity="sha384-X4G4FVCa7WAXVRvKzIAyoY82pbr5uF+lu8zOcg6+4PlrWA+bQhY7t2td1pDN5kPd" crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js" integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8sh+Wy6nP8eNVhCFmbo2KZ&" crossorigin="anonymous"></script>

<script>

// Bootstrap validation

(function () {

'use strict';

window.addEventListener('load', function () {

var forms = document.getElementsByClassName('needs-validation');

var validation = Array.prototype.filter.call(forms, function (form) {

form.addEventListener('submit', function (event) {

if (form.checkValidity() === false) {

event.preventDefault();

event.stopPropagation();

}

form.classList.add('was-validated');

}, false);

});

}, false);

})();

</script>

</body>

</html>

**Q8. Integrate a Bootstrap modal in an ASP.NET application. Create a button that, when clicked, triggers the display of a modal with relevant information. Include dynamic content in the modal and demonstrate the ability to close it programmatically**

**index.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="index.aspx.cs" Inherits="f\_proj.index" %>

<!DOCTYPE html>

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

<head runat="server">

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>

<title></title>

</head>

<body>

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

<div>

<!-- Your ASP.NET button -->

<asp:Button ID="btnShowModal" runat="server" Text="Show Modal" CssClass="btn btn-primary" OnClientClick="showModal(); return false;" OnClick="btnShowModal\_Click" />

<!-- Bootstrap Modal -->

<div class="modal" tabindex="-1" role="dialog" id="myModal">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title">Modal Title</h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<!-- Dynamic content goes here -->

<p>Dynamic content in the modal.</p>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>

</div>

</div>

</div>

</div>

<script>

// JavaScript function to show the Bootstrap modal

function showModal() {

$('#myModal').modal('show');

}

// JavaScript function to close the Bootstrap modal programmatically

function closeModal() {

$('#myModal').modal('hide');

}

</script>

</div>

</form>

</body>

</html>

**index.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace f\_proj

{

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

{

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

{

// Your server-side code

// Call JavaScript function to close the modal

ScriptManager.RegisterStartupScript(this, this.GetType(), "CloseModalScript", "closeModal();", true);

}

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

{

}

}

}

**Q9. Enhance an ASP.NET application by incorporating Bootstrap modals for user interactions. Implement a feature that loads data from the server using AJAX when a button inside a modal is clicked. Show how Bootstrap and AJAX work together to improve the user experience.**

**index.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="index.aspx.cs" Inherits="f\_proj.index" %>

<!DOCTYPE html>

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

<head runat="server">

<!-- Bootstrap CSS -->

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

<!-- jQuery -->

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>

<!-- Bootstrap JS -->

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>

<title></title>

</head>

<body>

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

<div>

<!-- Your ASP.NET button -->

<asp:Button ID="btnShowModal" runat="server" Text="Show Modal" CssClass="btn btn-primary" data-toggle="modal" data-target="#myModal" />

<!-- Bootstrap Modal -->

<div class="modal" tabindex="-1" role="dialog" id="myModal">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title">Modal Title</h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<p>Click the button below to load data using AJAX:</p>

<button type="button" class="btn btn-success" id="btnLoadData">Load Data</button>

<!-- Container to display loaded data -->

<div id="dataContainer"></div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>

</div>

</div>

</div>

</div>

<!-- JavaScript to handle AJAX request -->

<script>

$(document).ready(function () {

// AJAX request when the "Load Data" button is clicked

$('#btnLoadData').click(function () {

$.ajax({

url: 'www.google.com', // Replace with your server-side endpoint

method: 'GET',

success: function (data) {

// Display the loaded data in the modal

$('#dataContainer').html(data);

},

error: function (error) {

console.error('Error loading data:', error);

}

});

});

});

</script>

</div>

</form>

</body>

</html>

**index.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace f\_proj

{

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

{

// Example of code-behind in your .aspx.cs file

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

{

// Your server-side code

// Endpoint to handle AJAX request

if (Request.QueryString["action"] == "loadData")

{

// Simulate loading data from the server (replace with your logic)

string loadedData = "Data loaded from the server.";

// Return the loaded data

Response.Write(loadedData);

Response.End();

}

}

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

{

}

}

}

**Q 10. Develop a web application in ASP.NET that utilizes different state management techniques. Create a simple webpage with a form containing user input fields. Implement the following state management techniques and demonstrate their usage: - View State: a) Store a piece of information in the view state, such as a user's input on the form. b) Retrieve and display this information after a postback. - Session State: a) Store user-specific data in session state, like user preferences or settings. b) Implement a functionality to clear the session data after a certain event or timeout.**

**Q11. Develop a web application in ASP.NET that showcases the use of different state management techniques. Implement the following functionalities: - View State: a) Create a web form with a set of controls (e.g., TextBox, DropDownList). b) Use View State to maintain the state of these controls across postbacks. c) Demonstrate how to store and retrieve values in/from the View State. - Session State: a) Design a multi-page application with at least two pages. b) Store user-specific information (e.g., username) in the Session State when the user logs in on the first page. c) Retrieve and display this information on the second page.**

**(Q10 & Q11 are included in single code)**

**Question4validation.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="StateManagementDemo.aspx.cs" Inherits="Question4validation.StateManagementDemo" %>

<!DOCTYPE html>

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

<head runat="server">

<title>State Management Demo</title>

</head>

<body>

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

<div>

<h2>User Preferences</h2>

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

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

<br />

<label for="ddlTheme">Preferred Theme:</label>

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

<asp:ListItem Text="Light" Value="light" />

<asp:ListItem Text="Dark" Value="dark" />

</asp:DropDownList>

<br />

<asp:Button ID="btnSavePreferences" runat="server" Text="Save Preferences" OnClick="btnSavePreferences\_Click" />

<asp:Button ID="btnClearSession" runat="server" Text="Clear Session" OnClick="btnClearSession\_Click" />

</div>

<div>

<h2>View State Information</h2>

<asp:Label ID="lblViewStateInfo" runat="server"></asp:Label>

</div>

</form>

</body>

</html>

**Question4validation.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Question4validation

{

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

{

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

{

if (!IsPostBack)

{

// Check if there is data in the ViewState

if (ViewState["UserInfo"] != null)

{

string userInfo = ViewState["UserInfo"].ToString();

lblViewStateInfo.Text = $"View State Info: {userInfo}";

}

}

}

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

{

// Store user preferences in Session State

Session["Username"] = txtUsername.Text;

Session["Theme"] = ddlTheme.SelectedValue;

// Store a piece of information in View State

ViewState["UserInfo"] = $"Username: {txtUsername.Text}, Theme: {ddlTheme.SelectedValue}";

// Display user preferences

lblViewStateInfo.Text = $"View State Info: {ViewState["UserInfo"]}";

}

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

{

// Clear Session State data

Session.Clear();

lblViewStateInfo.Text = "View State Info: (Cleared)";

}

}

}

**Q12. Develop a web application in ASP.NET that showcases the use of different state management techniques. The application should include the following functionalities: - Cookies: a) Develop a page with a counter that increments each time the page is loaded. b) Use cookies to persistently store and retrieve the counter value. Display the counter on the page. - Application State: a) Implement a feature that maintains a count of the total number of visits to the application. b) Use Application State to store and update the visit count. Display the count on the web page**

**cookies.aspx**

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

<!DOCTYPE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<h2>Cookie Counter</h2>

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

<h2>Application State</h2>

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

</div>

</form>

</body>

</html>

**cookies.aspx.cs**

using System;

namespace Cookies

{

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

{

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

{

if (!IsPostBack)

{

// Cookies: Increment the counter using cookies

if (Request.Cookies["CounterCookie"] != null)

{

int counter = Convert.ToInt32(Request.Cookies["CounterCookie"].Value);

counter++;

Response.Cookies["CounterCookie"].Value = counter.ToString();

}

else

{

int counter = 1;

Response.Cookies["CounterCookie"].Value = counter.ToString();

}

lblCookieCounter.Text = "Cookie Counter: " + Request.Cookies["CounterCookie"].Value;

// Application State: Maintain a count of the total number of visits

if (Application["VisitCount"] == null)

{

Application["VisitCount"] = 1;

}

else

{

int visitCount = Convert.ToInt32(Application["VisitCount"]);

visitCount++;

Application["VisitCount"] = visitCount;

}

lblApplicationState.Text = "Total Visits (Application State): " + Application["VisitCount"];

}

}

}

}

**Q13**. **Develop a feature-rich ASP.NET web application that includes various validation techniques and utilizes rich controls. The application should have the following functionalities: Form Validation: a) Create a web page with a form that collects user registration information (e.g., username, email, password). b) Implement client-side validation using JavaScript/jQuery to ensure that required fields are filled and validate the email format. c) Use ASP.NET validation controls to enforce server-side validation for password complexity**

**index.aspx**

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

<!DOCTYPE html>

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

<head runat="server">

<title>User Registration</title>

<script type="text/javascript">

function validateForm() {

// Client-side validation

var isValid = true;

// Required field validation

var requiredFields = ["<%=txtName1.ClientID %>", "<%=txtEmail1.ClientID %>", "<%=txtPassword1.ClientID %>"];

for (var i = 0; i < requiredFields.length; i++) {

var field = document.getElementById(requiredFields[i]);

if (field.value.trim() === "") {

alert("Please fill in all required fields.");

isValid = false;

break;

}

}

// Email format validation

var emailField = document.getElementById("<%=txtEmail1.ClientID %>");

var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

if (!emailRegex.test(emailField.value)) {

alert("Please enter a valid email address.");

isValid = false;

}

// Password strength validation

var passwordField = document.getElementById("<%=txtPassword1.ClientID %>");

var passwordRegex = /^(?=.[A-Za-z])(?=.\d)[A-Za-z\d]{8,}$/;

if (!passwordRegex.test(passwordField.value)) {

alert("Password must be at least 8 characters long and contain at least one letter and one number.");

isValid = false;

}

return isValid;

}

</script>

</head>

<body>

<form id="form1" runat="server" onsubmit="return validateForm()">

<div>

<label for="txtName">Name:</label>

<asp:TextBox ID="txtName1" runat="server"></asp:TextBox>

<br />

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

<asp:TextBox ID="txtEmail1" runat="server"></asp:TextBox>

<br />

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

<asp:TextBox ID="txtPassword1" runat="server" TextMode="Password"></asp:TextBox>

<br />

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

</div>

</form>

</body>

</html>

**index.aspx.cs**

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data.SqlClient;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Xml.Linq;

namespace Validation

{

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

{

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

{

}

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

{

try {

// Server-side validation

if (IsValid)

{

// Process the form submission

string name = txtName1.Text;

string email = txtEmail1.Text;

string password = txtPassword1.Text;

// Perform further processing or database operations

// ...

// Display success message or redirect to another page

Response.Write("Form submitted successfully!");

}

}

catch(Exception ex)

{

}

}

}

}

**Q14. Develop a feature-rich ASP.NET web application that includes various validation techniques and utilizes rich controls. The application should have the following functionalities: Form Validation: a) Create a web page with a form that collects user registration information (e.g., username, email, password). b) Extend the registration form to include a custom validation rule (e.g., mobile numbers must be unique and 10 digits). Implement a custom validator to enforce this rule on the server-side.**

WebForm1.aspx

﻿<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Ques14.WebForm1" %>

<!DOCTYPE html>

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

<head runat="server">

<title></title>

<style type="text/css">

.auto-style1 {

margin-left: 0px;

}

.auto-style2 {

margin-left: 160px;

}

</style>

</head>

<body>

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

<div class="auto-style2">

<label for="txtUsername">Username:&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; </label>

&nbsp;<asp:TextBox ID="txtUsername" runat="server" required CssClass="auto-style1"></asp:TextBox>

<asp:CustomValidator ID="cvName" runat="server" ErrorMessage="Name should not contain any number." OnServerValidate="ValidateName"></asp:CustomValidator>

<br />

<label for="txtEmail">Email:</label>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

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

<asp:CustomValidator ID="cvEmail" runat="server" ErrorMessage="Invalid email format." OnServerValidate="ValidateEmail"></asp:CustomValidator>

<br />

<label for="txtPassword">Password:&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; </label>&nbsp;<asp:TextBox ID="txtPassword" runat="server" TextMode="Password" required></asp:TextBox>

<br />

<label for="txtMobileNumber">Mobile Number:&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; </label>&nbsp;<asp:TextBox ID="txtMobileNumber" runat="server" required></asp:TextBox>

<asp:CustomValidator ID="cvMobileNumber" runat="server" ErrorMessage="Mobile number must be unique and 10 digits." OnServerValidate="ValidateMobileNumber"></asp:CustomValidator>

<br />

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<asp:Button ID="btnRegister" runat="server" Text="Register" OnClick="btnRegister\_Click" />

</div>

</form>

</body>

</html>

**WebForm1.aspx.cs**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text.RegularExpressions;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Ques14

{

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

{

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

{

}

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

{

// Validate the form

if (Page.IsValid)

{

// Insert user data into the database

string connectionString = "Data Source=localhost\\SQLEXPRESS;Initial Catalog=abc;Integrated Security=True";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "INSERT INTO Users (Username, Email, Password, MobileNumber) VALUES (@Username, @Email, @Password, @MobileNumber)";

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

{

cmd.Parameters.AddWithValue("@Username", txtUsername.Text);

cmd.Parameters.AddWithValue("@Email", txtEmail.Text);

cmd.Parameters.AddWithValue("@Password", txtPassword.Text);

cmd.Parameters.AddWithValue("@MobileNumber", txtMobileNumber.Text);

cmd.ExecuteNonQuery();

}

}

// Registration successful, redirect or display a success message

Response.Redirect("WebForm1.aspx"); // Redirect to a success page

}

}

protected void ValidateMobileNumber(object source, System.Web.UI.WebControls.ServerValidateEventArgs args)

{

// Custom validation for mobile number (e.g., unique and 10 digits)

args.IsValid = IsMobileNumberValid(txtMobileNumber.Text);

}

protected void ValidateName(object source, System.Web.UI.WebControls.ServerValidateEventArgs args)

{

// Custom validation for name (e.g., should not contain any number)

args.IsValid = IsNameValid(txtUsername.Text);

}

protected void ValidateEmail(object source, System.Web.UI.WebControls.ServerValidateEventArgs args)

{

// Custom validation for email

args.IsValid = IsEmailValid(txtEmail.Text);

}

private bool IsMobileNumberValid(string mobileNumber)

{

// Implement your validation logic here

// Example: Check if the mobile number is unique and has 10 digits

// You may need to query the database to check for uniqueness

// For simplicity, let's assume the mobile number is valid

return mobileNumber.Length == 10;

}

private bool IsNameValid(string name)

{

// Implement your validation logic here

// Example: Check if the name contains any number

return !Regex.IsMatch(name, @"\d");

}

private bool IsEmailValid(string email)

{

// Implement your validation logic here

// Example: Check if the email is valid using a regular expression

return Regex.IsMatch(email, @"^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$");

}

}

}

(Process- Template - WindowsFormsApp) -> When Project is created, Drag and drop the components from toolbox, then Double click on the component to open related C# file and add validation code.

**Q15. Develop an ASP.NET web application that incorporates validation controls and rich user interface elements. The application should have the following features: Form with Validation Controls: a) Create a web form with various input fields (e.g., textboxes, dropdowns, checkboxes). b) Implement validation controls (e.g., RequiredFieldValidator, RegularExpressionValidator) to ensure data integrity. c) Include a custom validator for a specific validation scenario.**

**Validation.aspx**

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

<!DOCTYPE html>

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

<head runat="server">

<title>User Registration</title>

<script type="text/javascript">

function validateForm() {

// Client-side validation

var isValid = true;

// Required field validation

var requiredFields = ["<%=txtName1.ClientID %>", "<%=txtEmail1.ClientID %>", "<%=txtPassword1.ClientID %>"];

for (var i = 0; i < requiredFields.length; i++) {

var field = document.getElementById(requiredFields[i]);

if (field.value.trim() === "") {

alert("Please fill in all required fields.");

isValid = false;

break;

}

}

// Email format validation

var emailField = document.getElementById("<%=txtEmail1.ClientID %>");

var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

if (!emailRegex.test(emailField.value)) {

alert("Please enter a valid email address.");

isValid = false;

}

// Password strength validation

var passwordField = document.getElementById("<%=txtPassword1.ClientID %>");

var passwordRegex = /^(?=.[A-Za-z])(?=.\d)[A-Za-z\d]{8,}$/;

if (!passwordRegex.test(passwordField.value)) {

alert("Password must be at least 8 characters long and contain at least one letter and one number.");

isValid = false;

}

return isValid;

}

</script>

</head>

<body>

<form id="form1" runat="server" onsubmit="return validateForm()">

<div>

<label for="txtName">Name:</label>

<asp:TextBox ID="txtName1" runat="server"></asp:TextBox>

<br />

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

<asp:TextBox ID="txtEmail1" runat="server"></asp:TextBox>

<br />

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

<asp:TextBox ID="txtPassword1" runat="server" TextMode="Password"></asp:TextBox>

<br />

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

</div>

</form>

</body>

</html>

**Validation.aspx.cs**

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data.SqlClient;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Xml.Linq;

namespace Validation

{

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

{

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

{

}

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

{

try {

// Server-side validation

if (IsValid)

{

// Process the form submission

string name = txtName1.Text;

string email = txtEmail1.Text;

string password = txtPassword1.Text;

// Perform further processing or database operations

// ...

// Display success message or redirect to another page

Response.Write("Form submitted successfully!");

}

}

catch(Exception ex)

{

}

}

} }