17.2-Building a Simple Web Application using ASP.NET

Program

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Login.aspx.cs"
Inherits="YourNamespace.Login" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Login Page</title>
    <link href="Styles.css" rel="stylesheet" />
</head>
<style>
.form-group {
    margin-bottom: 20px;
}

.error-message {
    color: red;
    font-size: 14px;
}

</style>
<body>
    <form id="loginForm" runat="server">
        <div>
```

```
        <h2>Login</h2>
        <div class="form-group">
            <label for="txtUsername">Username/Email:</label>
            <asp:TextBox ID="txtUsername" runat="server" CssClass="form-control"></asp:TextBox>
                                    <asp:RequiredFieldValidator ID="rfvUsername" runat="server"
ControlToValidate="txtUsername"
                                                ErrorMessage="Username/Email is required."
CssClass="error-message"></asp:RequiredFieldValidator>
        </div>
        <div class="form-group">
            <label for="txtPassword">Password:</label>
                            <asp:TextBox ID="txtPassword" runat="server" TextMode="Password"
CssClass="form-control"></asp:TextBox>
                                    <asp:RequiredFieldValidator ID="rfvPassword" runat="server"
ControlToValidate="txtPassword"
                                                ErrorMessage="Password is required."
CssClass="error-message"></asp:RequiredFieldValidator>
        </div>
        <div class="form-group">
                <asp:Button ID="btnLogin" runat="server" Text="Login" CssClass="btn btn-primary"
OnClick="btnLogin_Click" />
        </div>
        <div class="error-message" id="lblErrorMessage" runat="server" visible="false"></div>
      </div>
   </form>
</body>
</html>
```

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

namespace namespace
{
    public partial class Login : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            // Check if user is already authenticated, redirect to home page if true
            if (User.Identity.IsAuthenticated)
            {
                Response.Redirect("Home.aspx");
            }
        }

        protected void btnLogin_Click(object sender, EventArgs e)
        {
            string username = txtUsername.Text;
            string password = txtPassword.Text;

            // Authenticate user (you'll replace this with your authentication logic)
            bool isAuthenticated = AuthenticateUser(username, password);

            if (isAuthenticated)
```

```
        {
            // Redirect authenticated user to home page
            Response.Redirect("Home.aspx");
        }
        else
        {
            lblErrorMessage.InnerText = "Invalid username/email or password.";
            lblErrorMessage.Visible = true;
        }
    }

    // Example authentication logic (replace with your actual logic)
    private bool AuthenticateUser(string username, string password)
    {
        // Your authentication logic goes here (e.g., check against database, external provider, etc.)
            // For demonstration purposes, return true if username is "admin" and password is "password"
        return username == "admin" && password == "password";
    }
  }
}
```

Exercise

Design and implement a login page using ASP.NET for a web application. The login page should include the following functionalities:

Create a visually appealing and user-friendly login form.

Include input fields for username/email and password.

Add validation for input fields to ensure the required format and length constraints are met.

Implement client-side validation using JavaScript for a seamless user experience.

Implement server-side authentication to verify user credentials.

Authenticate users against a database or another authentication provider.

Handle authentication failures and display appropriate error messages to users.

Provide options for users to reset their passwords or recover their accounts if they forget their credentials.

Hint

Utilize modern CSS frameworks like Bootstrap or Tailwind CSS to create a visually appealing login form with responsive design for seamless usability across different devices.

Use HTML <input> elements with appropriate attributes such as type="text" for username/email and type="password" for password input fields.

Implement client-side validation using JavaScript to check for empty fields, validate email format, and enforce password length constraints. Display error messages dynamically to guide users.

Explanation

For the UI, a visually appealing login form is created using HTML and CSS, possibly leveraging frontend frameworks like Bootstrap for responsive design. The form includes input fields for username/email and password, with validation rules enforced to ensure the required format and length constraints are met. Client-side validation using JavaScript enhances the user experience by providing instant feedback on input errors without the need for server requests.

On the authentication side, server-side authentication logic is implemented within the ASP.NET

backend. This involves verifying user credentials against a database, ensuring secure storage and retrieval of user data. Authentication failures, such as incorrect username/email or password, are handled gracefully, with appropriate error messages displayed to users for clarity. Additionally, options for password reset or account recovery are provided to users who forget their credentials, directing them to the necessary steps to regain access to their accounts.