Backend (.NET) Final Exam

Make a MVC application with project name `Task 1` and solution name `BackendFinal`.

Task 1

> Authentication mechanism
>
> create application in which user can authenticate by username and password

Create MVC application, then go to the `controller` folder add another controller (Empty) named: `SecurityController.cs`

In this controller define action:

```
//SecurityController.cs
using Microsoft.AspNetCore.Mvc;


namespace Task1.Controllers
{
    public class SecurityController : Controller
    {
        public IActionResult Login()
        {
            return View();
        }
    }
}
```

This action should load a form, which will contain two inputs for username and password, for that go to the `Views` folder and another subfolder named `Security` , in that subfolder add a view (Empty) named `Login` , in this view add a form, this form method must only generate post so add it as a parameter to `Html.BeginForm` :

```
//Login.cshtml
@*
    For more information on enabling MVC for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}


@using (Html.BeginForm(FormMethod.Post))
{
    <div>@ViewBag.Msg</div> //If viewBag contains nothing, nothing will be displayed.
    <div>
        @Html.TextBox("UID")
    </div>
    <div>
        @Html.Password("PWD")
    </div>
    <div>
        <input type="submit" value="OK" />
    </div>
}
```

go to this address https://localhost:7061/Security/Login and check

Now w need another action in `SecurityController.cs` to process the submit information, this method must be called only when the user submits (POST), this method must check if the username and password are correct:

Hold in memory dictionary of the usernames and passwords.

and add method which signs out the user.

```
//SecurityController.cs
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;

namespace Task1.Controllers
{
    public class SecurityController : Controller
    {
        public IActionResult Login()
        {
            return View();
        }

        //dictionary to store users usernames and passwords
        static Dictionary<string, string> users = new Dictionary<string, string>
        {
            { "Michael", "1234" },
            { "John", "5678" }
        };

        //method to authenticate the user
        [HttpPost]
        public IActionResult Login(String UID, String PWD)
        {
            if(users.ContainsKey(UID) && users[UID] == PWD)
            {
                // Create claims (store the name) for the authenticated user
                var claims = new List<Claim> { new Claim(ClaimTypes.Name, UID) };
                var claimsIdentity = new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme);
                //call sign in aync method to authenticate the user
                HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(claimsIden
                // Redirect to the home page or dashboard after successful login
                return RedirectToAction("Index", "Home");
            }
            else
            {
                // Return to the login view with an error message
                ViewBag.ErrorMessage = "Incorrect username or password.";
                return Login();
            }
        }

        //method to log out the user
        public IActionResult Logout()
        {
            // Sign out the user
            HttpContext.SignOutAsync().Wait();
            // Redirect to the login page after logout
            return RedirectToAction("Login");
        }
```

```
    }
}
```

If UID and PWD are correct, we need to authenticate the user, for it we need to enable authentication in the `Program.cs` file and add the following lines of code:

```
builder.Services.AddControllersWithViews();

//Add this line to enable cookie authentication
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme).AddCookie();
```

```
app.UseAuthorization();

// Add this line to enable authentication middleware
app.UseAuthentication();
```

call https://localhost:7061/Security/Logout to logout.

Task 2

Create ASP .NET Core Web API project named Task 2 and a Console App project named Task 2 client.

> We are asked to implement a method, for example squares and this method will calculate sum of the squares of two numbers, this two numbers must be submitted to the method using POST method.

in Task 2 Project:

Add new controller (API controller - Empty) named `MathController.cs` , in this controller add method to calculate sum of the squares.

```
//MathConrtroller.cs
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace Task_2.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class MathController : ControllerBase
    {
        //post must be used for this method
        [HttpPost]
        public double Square([FromForm] double x, [FromForm] double y)
        {
            return x * x + y * y;
        }
    }
}
```

now run the application and make sure that it works. now we move to the client side

in Task 2 client Project:

Go to the `Program.cs` file and write the following Code:

```
//Program.cs
var cnt = new HttpClient();
```

```
//base address uri is from task 2
cnt.BaseAddress = new Uri("https://localhost:7065");
var content = new StringContent("X=3&Y=4", System.Text.Encoding.UTF8, "application/x-www-form-urlencoded");
var msg = cnt.PostAsync("/api/Math", content).Result;

if(msg.StatusCode == System.Net.HttpStatusCode.OK)
{
   var result = msg.Content.ReadAsStringAsync().Result;
   Console.WriteLine(result);
}
```

Task 3

Add MVC project named Task 3 in the solutions folder.

AJAX methodology

call action on the server through POST method from console application, we are asked to call server method, server action using AJAX methodology (only GET method). server action must return JSON object, and this JSON object must be parsed using javascript, must display parts, not the whole object.

`HomeController.cs` is loading the index view, we want to call server action from index view using AJAX methodology, add action to `HomeController.cs`

```
//HomeController.cs
public JsonResult Sum(double a, double b)
{
   return Json(new { r = a + b });
}
```

now we need to go to the index view (Views → Home → `index.cshtml`) add two inputs to index view

```
//index.cshtml
@{
   ViewData["Title"] = "Home Page";
}

<div class="text-center">
   <h1 class="display-4">Welcome</h1>
   <p>Learn about <a href="https://learn.microsoft.com/aspnet/core">building Web apps with ASP.NET Core</a>.</p>
</div>

<script type="text/javascript">
   function okrun()
   {
      var http = new XMLHttpRequest();
      http.open("GET", "/Home/Sum?a=" + x.value + "&b=" + y.value, false);
      http.send();
      //parse the JSON object, to make it more user friendly
      var h = JSON.parse(http.responseText);
      //only show the value
      alert(h.r);
   }
</script>

<div>
```

```
    <input type="number" id="x" />
    <br/>
    <input type="number" id="y" />
    <br/>
    <input type="button" value="ok" onclick="okrun();" />
</div>
```

Task 4

Add MVC project named Task 4 in the solutions folder.

> our goal is to write an application that will display the text field, for example writing in this text field “abcd” and then pressing send, server should return the word in which the order of the symbols are reversed, so “dcba” should be returned and for exchanging the information between the server and browser it is required to use `WebSockets`.

In index view display one input:

```
//index.cshtml
@{
    ViewData["Title"] = "Home Page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://learn.microsoft.com/aspnet/core">building Web apps with ASP.NET Core</a>.</p>
</div>

<div>
    <input type="text" id="x" />
    <br/>
    <input type="button" value="send"/>
</div>
```

First we should write server version, for which we need to create an action, which is enabling `websocket`, and then in `javacript` we will write instructions, which is connecting to server using `websocket`.

Let’s go to `Program.cs`

```
//Program.cs
app.UseStaticFiles();
//use WebSockets
app.UseWebSockets();
```

Next, go to `HomeController.cs` and add action, which will be called from browser using `websocket`.

```
//HomeController.cs
using Microsoft.AspNetCore.Mvc;
using System.Diagnostics;
using System.Net.WebSockets;
using Task_4.Models;

namespace Task_4.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;
```

```
public HomeController(ILogger<HomeController> logger)
{
    _logger = logger;
}
public void Talk()
{
    if(HttpContext.WebSockets.IsWebSocketRequest)
    {
        //create websockt from the serber side
        var sock = HttpContext.WebSockets.AcceptWebSocketAsync().Result;
        DoTalking(sock);
    }
}
//respond to the clients websocket request
void DoTalking(WebSocket sock)
{
    var buffer = new byte[1024];
    //we need to know the actual value of the bytes received
    WebSocketReceiveResult n;
    //receive messages from the client
    n = sock.ReceiveAsync(new ArraySegment<byte>(buffer), CancellationToken.None).Result;
    //must be converted to string to be readable
    var info = System.Text.Encoding.UTF8.GetString(buffer, 0, n.Count);
    //reverse the string
    var result = new String(info.Reverse().ToArray());
    //send it back to the client
    sock.SendAsync(new ArraySegment<byte>(System.Text.Encoding.UTF8.GetBytes(result)),
        WebSocketMessageType.Text, true, CancellationToken.None).Wait();
}

public IActionResult Index()
{
    return View();
}

public IActionResult Privacy()
{
    return View();
}


        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
        }
    }
}
```

and in index view add onclick logic:

```
//index.html
@{
    ViewData["Title"] = "Home Page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
```

```
    <p>Learn about <a href="https://learn.microsoft.com/aspnet/core">building Web apps with ASP.NET Core</a>.</p>
</div>

<script type="text/javascript">
    //run the program and see the address in the console
    ws = new WebSocket("wss://localhost:7160/Home/Talk");
    ws.onmessage = function(evt) { alert(evt.data.toString()); };

    function sendclick()
    {
        ws.send(x.value);
    }
</script>

<div>
    <input type="text" id="x" />
    <br/>
    <input type="button" value="send" onclick="sendclick();"/>
</div>
```