Київський національний університет ім. Тараса Шевченка
Кафедра мережевих та інтернет технологій

Лабораторна робота № 4

Дисципліна: Сучасні Інтернет Технології

Тема: Автентифікація та авторизація користувачів у застосунку ASP.NET CORE

Підготувала студентка МІТ-41
Пась Олександра

Хід виконання

1. Забезпечте аутентифікацію користувачів у застосунку. Реалізуйте сторінку входу, реєстрації та виходу, використовуючи ASP.NET Core Identity. Перевірте, що неавтентифіковані користувачі не мають доступу до жодної сторінки, крім головної сторінки та сторінки реєстрації/автентифікації.

   Для виконання цього завдання було додано файли Indentity. У файлі HomeController ми додали [Authorize] над класом, це означає що всі сторінки будуть доступні тільки авторизований користучавам. Щоб головна сторінка була доступна і для анонімних користувачів, ми додали [AllowAnonymous].

```
[Authorize]
3 references
public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;
    private readonly AppConfiguration _appSettings; // Додаємо поле для збереження конфігурації
    //Додаємо AppSettings у конструктор через Dependency Injection
    0 references
    public HomeController(ILogger<HomeController> logger, AppConfiguration appSettings)
    {
        _logger = logger;
        _appSettings = appSettings;
    }

    [AllowAnonymous]
    //Використовуємо AppSettings у дії Index
    0 references
    public IActionResult Index()
    {
        // Передаємо дані у View через ViewBag
        ViewBag.AppName = _appSettings.ApplicationName;
        ViewBag.Theme = _appSettings.Theme;
        ViewBag.ApiKey = _appSettings.ApiSettings?.ApiKey;
        ViewBag.DefaultRole = _appSettings.DefaultRole;
```

рис. 4.1 - HomeController

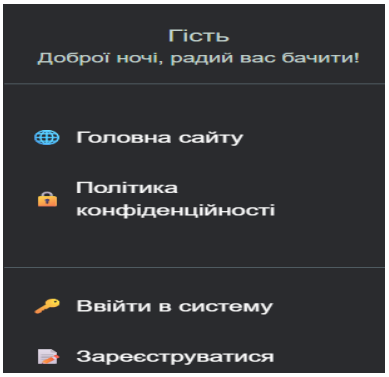


рис. 4.2 - доступні сторінки для анонімного користувача

2. Створіть політику авторизації, яка дозволяє доступ до сторінки «Архів матеріалів» лише тим користувачам, які мають твердження IsVerifiedClient. Додайте твердження вручну під час реєстрації.

Для цього завдання додамо політику авторизації у program.cs.

```
builder.Services.AddAuthorization(options =>
{
    //  Політика для перевірених клієнтів
    options.AddPolicy("IsVerifiedClient", policy =>
        policy.RequireClaim("IsVerifiedClient", "true"));
});
```

рис. 4.3 - політика авторизації

```
// Додаємо claim IsVerifiedClient при реєстрації
await _userManager.AddClaimAsync(user,
    new System.Security.Claims.Claim("IsVerifiedClient", "true"));
```

рис. 4.4 - claim в register.cshtml.cs

Було створено сторінку “Архів матеріалів”.

```
namespace WebApplication1.Controllers
{
    [Authorize(Policy = "IsVerifiedClient")] //  тільки для перевірених клієнтів
    0 references
    public class ArchiveController : Controller
    {
        0 references
        public IActionResult Index()
        {
            return View();
        }

        0 references
        public IActionResult Documents()
        {
            return View();
        }
    }
}
```

рис. 4.5 - сторінка Архів матеріалів

```
@{
    ViewData["Title"] = "Архів матеріалів";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<link rel="stylesheet" href="~/css/site.css" />

<div class="container">
    <h1 class="title">🗃 Архів матеріалів</h1>

    <div class="input-section">
        <h2 style="color: var(--secondary); margin-bottom: 1.5rem;">Доступні матеріали</h2>

        <div class="results-section">
            <div class="entity-card">
                <div class="entity-name">📄 Навчальні матеріали</div>
                <div style="padding: 1rem;">
                    <ul style="text-align: left; color: var(--secondary);">
                        <li>Конспекти лекцій</li>
                        <li>Презентації</li>
                        <li>Методичні вказівки</li>
                        <li>Лабораторні роботи</li>
                    </ul>
                </div>
            </div>

            <div class="entity-card">
                <div class="entity-name">📹 Відеоархів</div>
                <div style="padding: 1rem;">
                    <ul style="text-align: left; color: var(--secondary);">
                        <li>Записи лекцій</li>
                        <li>Вебінари</li>
                        <li>Інструкції</li>
                        <li>Демонстрації</li>
                    </ul>
                </div>
            </div>

            <div class="entity-card">
                <div class="entity-name">📚 Бібліотека</div>
                <div style="padding: 1rem;">
                    <ul style="text-align: left; color: var(--secondary);">
                        <li>Підручники</li>
                        <li>Статті</li>
```

рис. 4.6 - index

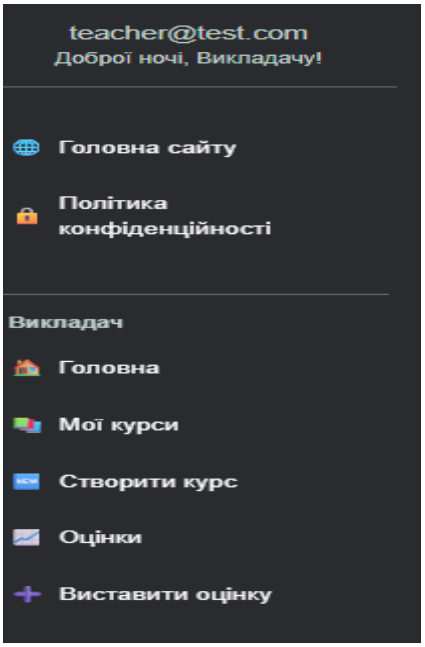


рис. 4.7 - сторінки користувача без IsVerifiedClient

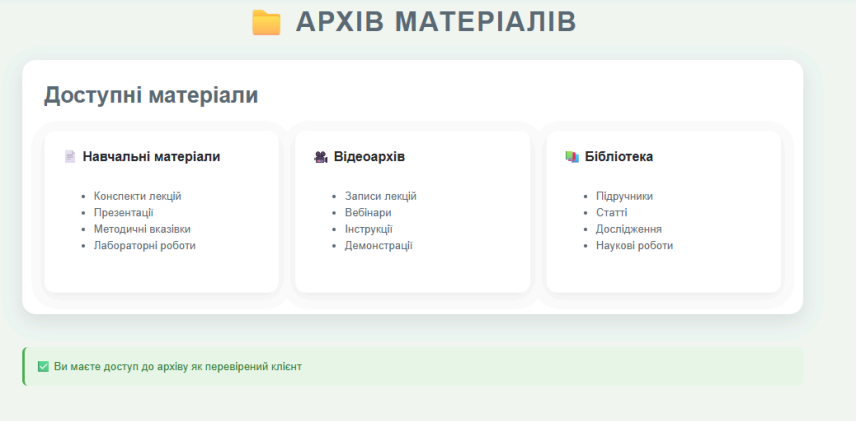


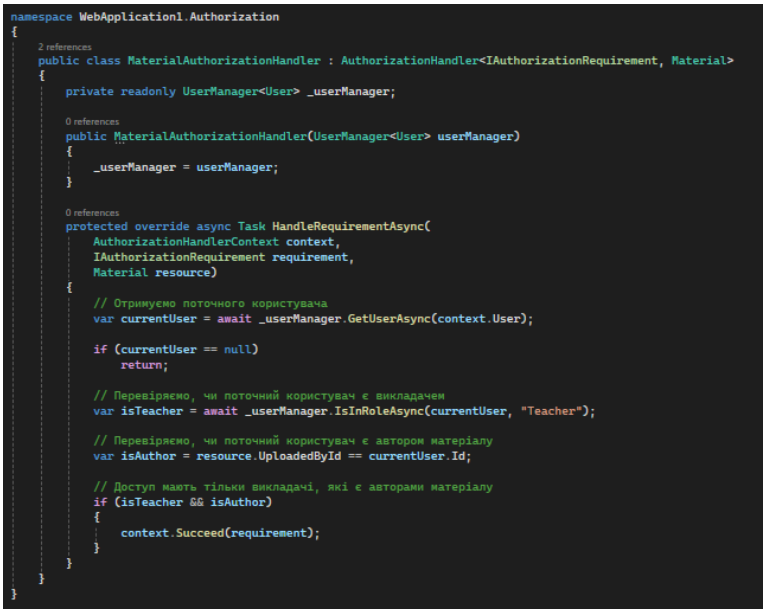
рис. 4.8 - користувач з IsVerifiedClient бачить сторінку

3. Реалізуйте ресурсну авторизацію для сторінки редагування ресурсу. Кожен ресурс має автора, і лише автор може редагувати його. Використайте IAuthorizationService та створіть обробник, який перевіряє, чи поточний користувач є автором ресурсу.

   Створимо папку Authorization у якій файли: MaterialAuthorizationHandler.cs (обробник), MaterialEditRequirement.cs (вимога).

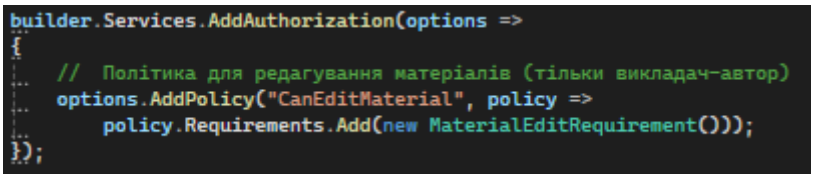
```
using Microsoft.AspNetCore.Authorization;

namespace WebApplication1.Authorization
{
    1 reference
    public class MaterialEditRequirement : IAuthorizationRequirement
    {

    }
}
```

рис. 4.9 - вимога



```
namespace WebApplication1.Authorization
{
    2 references
    public class MaterialAuthorizationHandler : AuthorizationHandler<IAuthorizationRequirement, Material>
    {
        private readonly UserManager<User> _userManager;

        0 references
        public MaterialAuthorizationHandler(UserManager<User> userManager)
        {
            _userManager = userManager;
        }

        0 references
        protected override async Task HandleRequirementAsync(
            AuthorizationHandlerContext context,
            IAuthorizationRequirement requirement,
            Material resource)
        {
            // Отримуємо поточного користувача
            var currentUser = await _userManager.GetUserAsync(context.User);

            if (currentUser == null)
                return;

            // Перевіряємо, чи поточний користувач є викладачем
            var isTeacher = await _userManager.IsInRoleAsync(currentUser, "Teacher");

            // Перевіряємо, чи поточний користувач є автором матеріалу
            var isAuthor = resource.UploadedById == currentUser.Id;

            // Доступ мають тільки викладачі, які є авторами матеріалу
            if (isTeacher && isAuthor)
            {
                context.Succeed(requirement);
            }
        }
    }
}
```

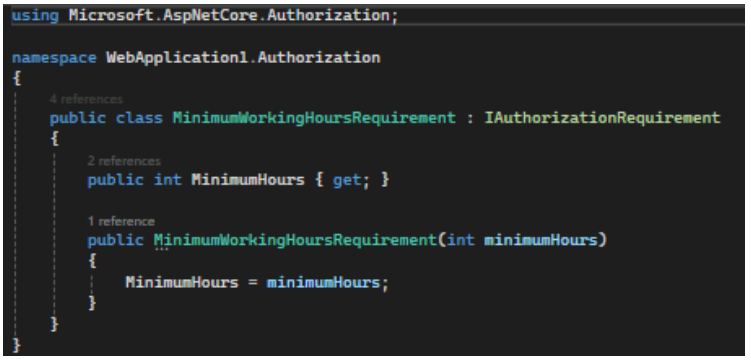
рис. 4.10 - обробник



```
builder.Services.AddAuthorization(options =>
{
    //  Політика для редагування матеріалів (тільки викладач-автор)
    options.AddPolicy("CanEditMaterial", policy =>
        policy.Requirements.Add(new MaterialEditRequirement()));
});
```

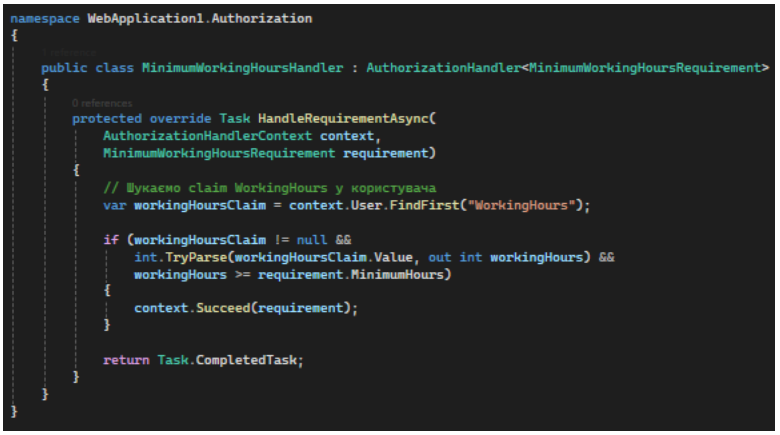
рис. 4.11 - політика авторизації

4. Створіть кастомну вимогу авторизації MinimumWorkingHoursRequirement, яка дозволяє доступ до сторінки «Преміум» лише тим користувачам, які мають твердження WorkingHours з числовим значенням не менше 100. Реалізуйте обробник, який виконує перевірку.



```
using Microsoft.AspNetCore.Authorization;

namespace WebApplication1.Authorization
{
    4 references
    public class MinimumWorkingHoursRequirement : IAuthorizationRequirement
    {
        2 references
        public int MinimumHours { get; }

        1 reference
        public MinimumWorkingHoursRequirement(int minimumHours)
        {
            MinimumHours = minimumHours;
        }
    }
}
```

рис. 4.12 - вимога MinimumWorkingHoursRequirement.cs



```
namespace WebApplication1.Authorization
{
    public class MinimumWorkingHoursHandler : AuthorizationHandler<MinimumWorkingHoursRequirement>
    {
        0 references
        protected override Task HandleRequirementAsync(
            AuthorizationHandlerContext context,
            MinimumWorkingHoursRequirement requirement)
        {
            // Шукаємо claim WorkingHours у користувача
            var workingHoursClaim = context.User.FindFirst("WorkingHours");

            if (workingHoursClaim != null &&
                int.TryParse(workingHoursClaim.Value, out int workingHours) &&
                workingHours >= requirement.MinimumHours)
            {
                context.Succeed(requirement);
            }

            return Task.CompletedTask;
        }
    }
}
```

рис. 4.13 - обробник MinimumWorkingHoursHandler.cs

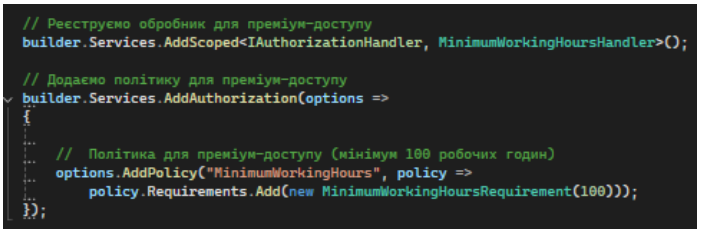


рис. 4.14 - реєструємо в program.cs

5. Створіть політику, яка дозволяє доступ до сторінки «Форум» лише тим користувачам, які мають хоча б одне з тверджень: IsMentor, IsVerifiedUser, або HasForumAccess. Реалізуйте обробник, який перевіряє хоча б одне з цих тверджень.

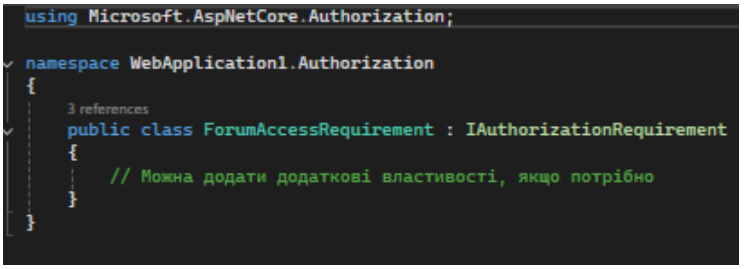


рис. 4.15 - вимога ForumAccessRequirement.cs

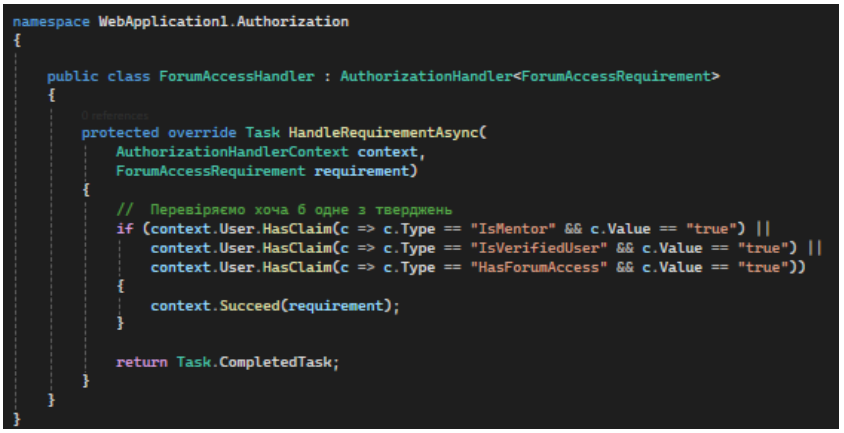


рис. 4.16 - обробник ForumAccessHandler.cs

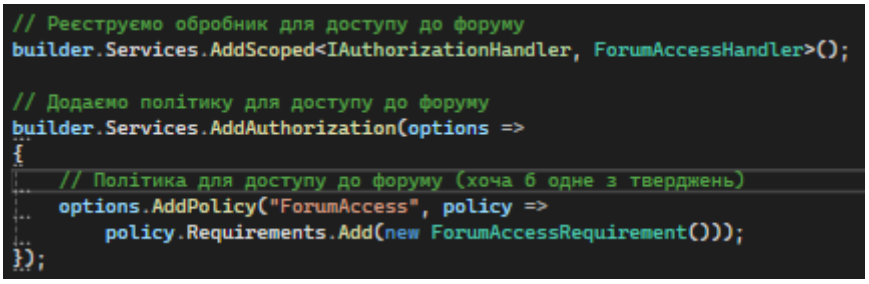


рис. 4.17 - реєструємо в program.cs

Висновок: Під час виконання лабораторної роботи було створено та налаштовано розширену систему автентифікації й авторизації в ASP.NET Core застосунку. Вона успішно забезпечує різні рівні доступу для користувачів залежно від їхніх ролей, claims та індивідуальних параметрів, наочно демонструючи гнучкі й потужні можливості безпеки цієї платформи.