МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра мережевих та інтернет технологій

СУЧАСНІ ІНТЕРНЕТ ТЕХНОЛОГІЇ

ГЛОБАЛІЗАЦІЯ ТА ЛОКАЛІЗАЦІЯ ЗАСТОСУНКУ ASP.NET CORE

Лабораторне заняття № 5

Заяць Діани

Хід виконання роботи:

Завдання 1. Ознайомитися з теоретичними основами глобалізації та локалізації: поняття Culture, UICulture, RFC 4646, ISO 639, ISO 3166 (сайт Microsoft).

Глобалізація та локалізація у програмуванні

- **Глобалізація (Globalization, G11n)** – це процес розробки програмного забезпечення таким чином, щоб його можна було без змін у коді адаптувати до різних мов та регіонів.
- **Локалізація (Localization, L10n)** – це подальший етап, коли додаток налаштовується під конкретну мову та культуру користувача: перекладаються текстові ресурси, змінюються формати дат, чисел, валют тощо.

Culture та UICulture (.NET)

- **Culture** – визначає культурні налаштування користувача (формати дат, чисел, грошових одиниць).
- **UICulture** – відповідає за мову інтерфейсу, тобто за те, які ресурси та текстові рядки будуть відображені в UI.

Стандарти мов та країн

Стандарти мови та країни

- RFC 4646 – стандарт тегів мови, наприклад:
  - en-US → англійська (США)
  - uk-UA → українська (Україна)
- ISO 639 – коди мов (en – англійська, uk – українська).
- ISO 3166 – коди країн (US – США, UA – Україна).

Взаємозв’язок

Мовні теги поєднують код мови з ISO 639 та код країни з ISO 3166:

- en-US → мова: en, країна: US
- uk-UA → мова: uk, країна: UA

У середовищі .NET ці теги використовуються при налаштуванні Culture та UICulture, що дає змогу адаптувати інтерфейс та формати даних під конкретного користувача.

Завдання 2. Додати до ASP.NET Core застосунку підтримку локалізації через AddLocalization() з вказанням папки ресурсів.
До конфігурації сервісів ASP.NET Core було додано виклик AddLocalization() із зазначенням каталогу, в якому зберігатимуться ресурсні файли (наприклад, папка *Resources*). Це увімкнуло підтримку локалізації в застосунку та дозволило працювати з .resx-файлами для різних культур.

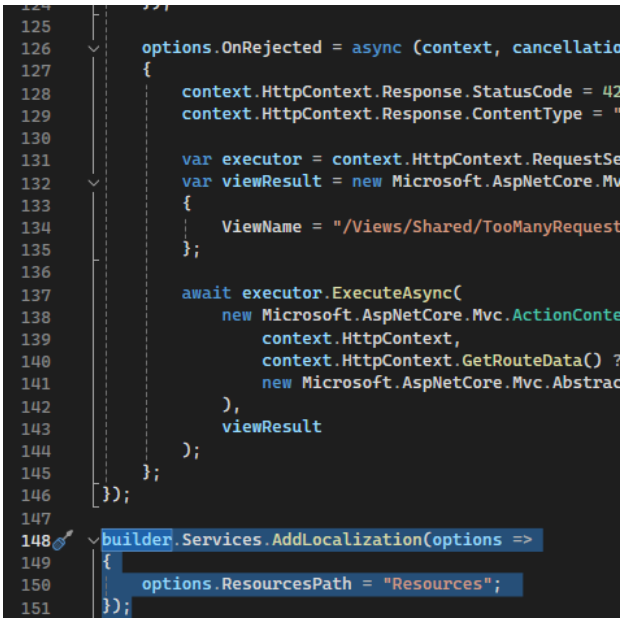


Рисунок 5.1 – Включення локалізації з папкою Resources до проекту

Завдання 3. Створити ресурсні файли .resx для базової культури та щонайменше двох додаткових.

Були створені ресурсні файли для трьох мов: чеської, англійської та української. Для кожної сторінки застосунку підготовлено окремі .resx-файли, в яких заповнюються переклади текстів інтерфейсу відповідними мовами.

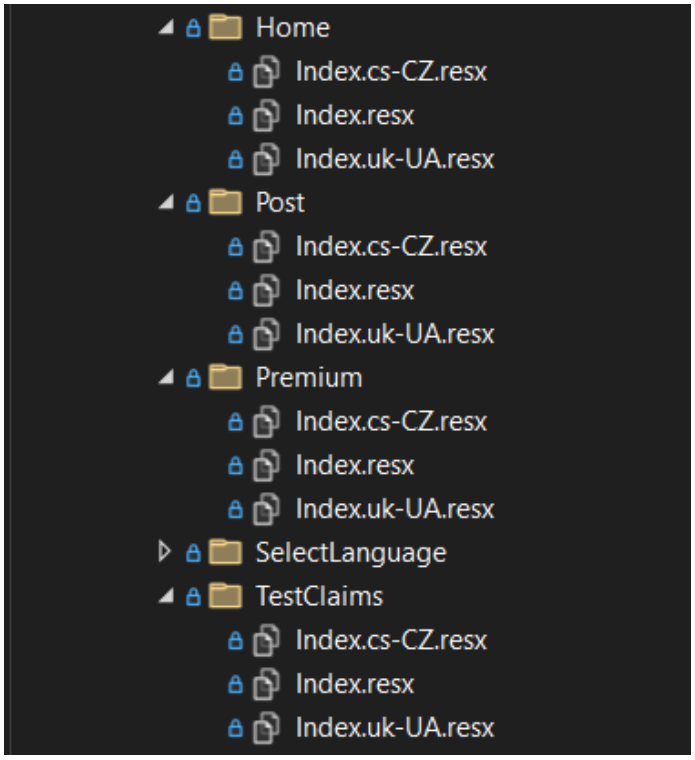


Рисунок 5.2 – Створені ресурсні файли

Name	Neutral Value	Neutral Comment	uk-UA
Available	This page is only available for user...		Ця сторінка доступна тільки кор...
Content	This is explicit content is only for p... \n You have access only be...		Це ексклюзивний контент для Pr... \n Ви маєте доступ до цієї...
ContentTitle	Premium content		Преміум контент
Title	Premium page		Преміум сторінка
Welcome	Welcome to the premium page!		Вітаємо на преміум сторінці!
YourHours	Your working hours:		Ваші робочі години:

Рисунок 5.3 – Приклад заповнення

Завдання 4. Налаштувати RequestLocalizationOptions з підтримкою щонайменше трьох культур. Встановити культуру за замовчуванням. Для цього додамо наступний код до конвеєра

У конфігурації застосунку було визначено RequestLocalizationOptions, у яких:

- перелічено підтримувані культури (наприклад, en-US, uk-UA, cs-CZ);
- встановлено культуру за замовчуванням, яка застосовується, якщо культура користувача не визначена або не підтримується.

Відповідні параметри були додані до конвеєра обробки запитів.

```
var supportedCultures = new[] { "en-US", "uk-UA", "cs-CZ" };

builder.Services.Configure<RequestLocalizationOptions>(options =>
{
    options.SetDefaultCulture("en-US")
    .AddSupportedCultures(supportedCultures)
    .AddSupportedUICultures(supportedCultures);
});
```

Рисунок 5.4 – Налаштування підтримки трьох культур

Завдання 5. Додати middleware UseRequestLocalization() у правильному місці конвеєра запиту. Пояснити, чому порядок має значення.

До конвеєра HTTP-запитів було додано проміжне ПЗ UseRequestLocalization(). Воно розміщене перед middleware, які відповідають за маршрутизацію, обробку ендпоїнтів та формування відповіді.

Порядок має значення, оскільки локалізація повинна бути застосована ще до того, як відпрацьовують контролери і представлення. Тільки в цьому випадку всі подальші компоненти (маршрути, валідація, відображення UI) будуть використовувати коректно обрану культуру.

```
var app = builder.Build();

app.UseRequestLocalization();

if (app.Environment.IsDevelopment())
{
    app.UseMigrationsEndPoint();
}
else
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRateLimiter();

app.UseRouting();
```

Рисунок 5.5 – Включення локалізації в конвеєр

Завдання 6. Реалізувати типізовану локалізацію у контролері через IStringLocalizer<T> та у Razor View через IViewLocalizer.

Було налаштовано використання типізованих локалізаторів:

- у контролерах застосовано IStringLocalizer<T> для отримання локалізованих рядків за ключами;
- у Razor-поданнях використано IViewLocalizer для заміни «захардкоджених» текстів на ті, що зчитуються з ресурсних файлів.

Таким чином, усі текстові елементи інтерфейсу поступово винесено до .resx-файлів, що спрощує підтримку кількох мов.

```razor
@{
    ViewData["Title"] = Localizer["Title"];
}
@inject Microsoft.AspNetCore.Mvc.Localization.IViewLocalizer Localizer

<h1>@ViewData["Title"]</h1>

<div class="alert alert-success">
    <h2>@Localizer["Welcome"]</h2>
    <p>@Localizer["Available"]</p>
    @if (ViewData["WorkingHours"] != null)
    {
        <p><strong>@Localizer["YourHours"]</strong> @ViewData["WorkingHours"]</p>
    }
</div>

<div class="card">
    <div class="card-body">
        <h3 class="card-title">@Localizer["ContentTitle"]</h3>
        <p class="card-text">
            @Localizer["Content"]
        </p>
    </div>
</div>
```

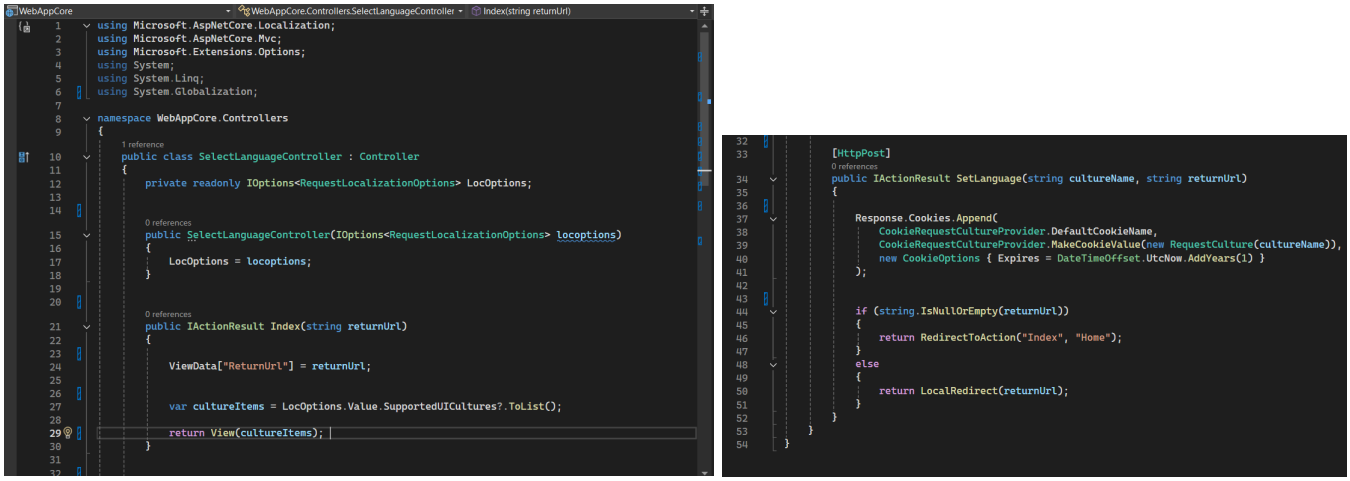
Рисунок 5.6 – Використання локалізації в шаблоні відображення

Завдання 7. Реалізувати перемикання мов, що встановлює культуру через cookies. Перевірити збереження обраної культури між запитами та сесіями.

Для перевірки локалізації інтерфейсу було використано розширення Chrome, за допомогою якого змінено значення заголовка Accept-Language на cs для локального домену. Це дозволило оцінити коректність роботи чеської локалізації, зокрема при відкритті сторінки *Premium* за умови, що ресурсні файли заповнені коректно.

Далі до RequestLocalizationOptions було додано Cookie-провайдер, що зберігає вибрану користувачем мову у cookie браузера. На рівні інтерфейсу реалізовано елементи керування для ручного перемикання мови.

Створено контролер та відповідне подання, які дозволяють користувачу обрати мову. Після вибору культура зберігається в cookie та використовується в наступних запитах.



```
WebAppCore  WebAppCore.Controllers.SelectLanguageController  Index(string returnUrl)
```

```razor
using Microsoft.AspNetCore.Localization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Options;
using System;
using System.Linq;
using System.Globalization;

namespace WebAppCore.Controllers
{
    1 reference
    public class SelectLanguageController : Controller
    {
        private readonly IOptions<RequestLocalizationOptions> LocOptions;


        0 references
        public SelectLanguageController(IOptions<RequestLocalizationOptions> locoptions)
        {
            LocOptions = locoptions;
        }


        0 references
        public IActionResult Index(string returnUrl)
        {

            ViewData["ReturnUrl"] = returnUrl;


            var cultureItems = LocOptions.Value.SupportedUICultures?.ToList();

            return View(cultureItems);
        }
```

```razor
        [HttpPost]
        0 references
        public IActionResult SetLanguage(string cultureName, string returnUrl)
        {

            Response.Cookies.Append(
                CookieRequestCultureProvider.DefaultCookieName,
                CookieRequestCultureProvider.MakeCookieValue(new RequestCulture(cultureName)),
                new CookieOptions { Expires = DateTimeOffset.UtcNow.AddYears(1) }
            );


            if (string.IsNullOrEmpty(returnUrl))
            {
                return RedirectToAction("Index", "Home");
            }
            else
            {
                return LocalRedirect(returnUrl);
            }
        }
    }
}
```

Рисунок 5.7 - Створення контролера SelectLanguageController

У каталозі Views/SelectLanguage створено подання Index.cshtml, яке відображає список підтримуваних культур у вигляді кнопок-форм. Кожна кнопка викликає дію SetLanguage контролера для зміни мови.

Щоб забезпечити доступ до перемикача мови на всіх сторінках, посилання на SelectLanguage/Index було додано до основного макету _Layout.cshtml (наприклад, до панелі навігації).

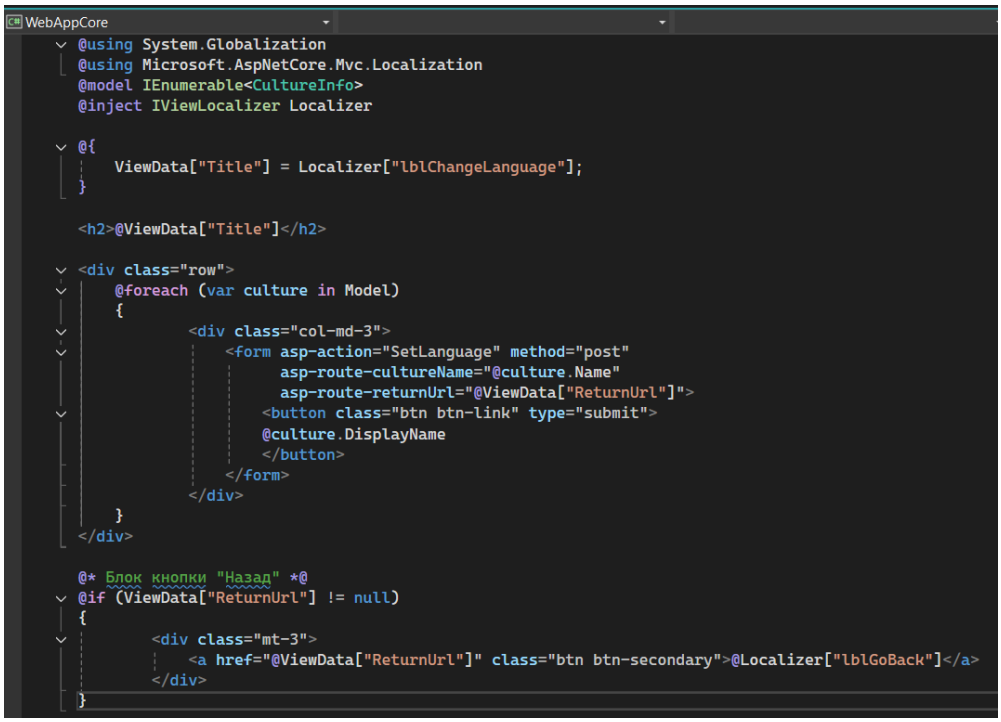


Рисунок 5.8 - Створення View для перемикання мови (Index.cshtml)

Завдання 8. Реалізувати підтримку параметрів URL (?culture=en-US) для вибору культури. Перевірити зміну мови.

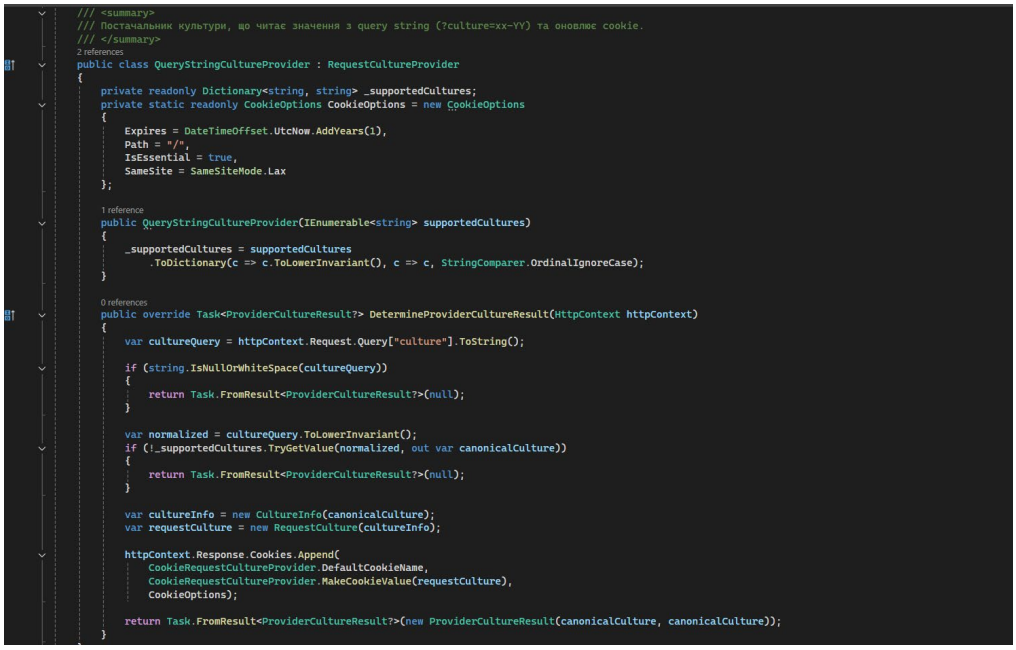
У межах цього завдання до вже налаштованої системи локалізації (на основі cookie) було додано підтримку вибору культури через параметр запиту ?culture=.... Логіка реалізована через власний провайдер культури QueryStringCultureProvider.

Реалізація власного QueryStringCultureProvider

У проєкті створено клас QueryStringCultureProvider у просторі імен WebAppCore.Localization. Даний провайдер:

- зчитує значення параметра culture з рядка запиту (наприклад, ?culture=en-US або ?culture=uk-UA);
- перевіряє, чи входить ця культура до списку підтримуваних (en-US, uk-UA, cs-CZ), визначеного в RequestLocalizationOptions;
- якщо культура підтримується, створює відповідний RequestCulture і:
    - одразу застосовує її до поточного запиту;
    - записує/оновлює cookie з ім’ям CookieRequestCultureProvider.DefaultCookieName зі строком життя 1 рік, Path = "/" і IsEssential = true.

Це забезпечує одночасно миттєву зміну культури для поточного запиту та її збереження для наступних запитів без параметра culture.



```
/// <summary>
/// Постачальник культури, що читає значення з query string (?culture=xx-YY) та оновлює cookie.
/// </summary>
2 references
public class QueryStringCultureProvider : RequestCultureProvider
{
    private readonly Dictionary<string, string> _supportedCultures;
    private static readonly CookieOptions CookieOptions = new CookieOptions
    {
        Expires = DateTimeOffset.UtcNow.AddYears(1),
        Path = "/",
        IsEssential = true,
        SameSite = SameSiteMode.Lax
    };

    1 reference
    public QueryStringCultureProvider(IEnumerable<string> supportedCultures)
    {
        _supportedCultures = supportedCultures
            .ToDictionary(c => c.ToLowerInvariant(), c => c, StringComparer.OrdinalIgnoreCase);
    }

    0 references
    public override Task<ProviderCultureResult?> DetermineProviderCultureResult(HttpContext httpContext)
    {
        var cultureQuery = httpContext.Request.Query["culture"].ToString();

        if (string.IsNullOrWhiteSpace(cultureQuery))
        {
            return Task.FromResult<ProviderCultureResult?>(null);
        }

        var normalized = cultureQuery.ToLowerInvariant();
        if (!_supportedCultures.TryGetValue(normalized, out var canonicalCulture))
        {
            return Task.FromResult<ProviderCultureResult?>(null);
        }

        var cultureInfo = new CultureInfo(canonicalCulture);
        var requestCulture = new RequestCulture(cultureInfo);

        httpContext.Response.Cookies.Append(
            CookieRequestCultureProvider.DefaultCookieName,
            CookieRequestCultureProvider.MakeCookieValue(requestCulture),
            CookieOptions);

        return Task.FromResult<ProviderCultureResult?>(new ProviderCultureResult(canonicalCulture, canonicalCulture));
    }
```

Рисунок 5.9 – Клас QueryStringCultureProvider у папці Localization проєкту WebAppCore

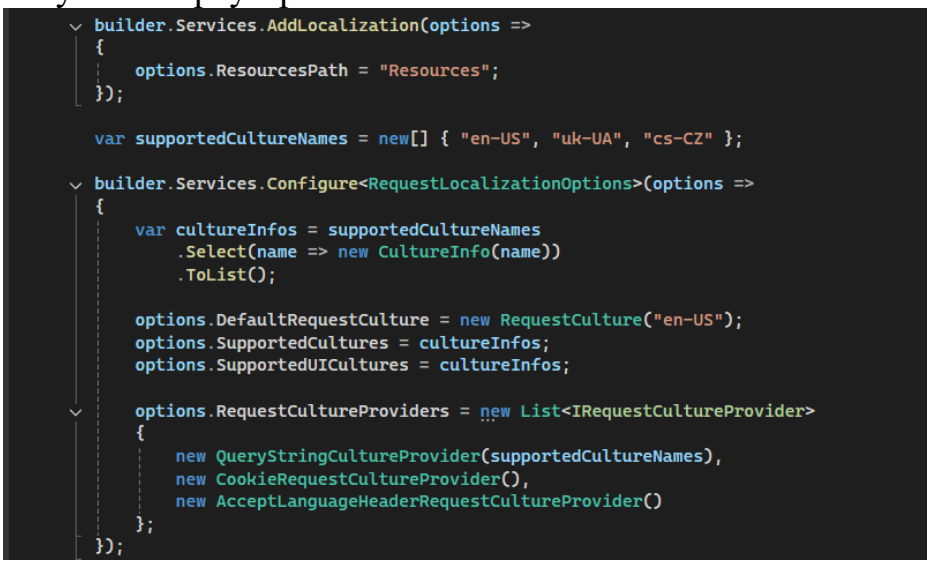
Налаштування RequestLocalizationOptions та порядок провайдерів

У файлі Program.cs зосереджено всю конфігурацію локалізації. В RequestLocalizationOptions:

- визначено перелік підтримуваних культур: en-US, uk-UA, cs-CZ;
- встановлено культуру за замовчуванням;
- задається порядок провайдерів культури:

1. власний QueryStringCultureProvider (пріоритет параметра ?culture=...);
2. CookieRequestCultureProvider (значення з cookie);
3. провайдер на основі заголовка Accept-Language (як запасний варіант).

Такий порядок гарантує, що в першу чергу враховується явний вибір користувача через URL, потім – раніше збережений вибір у cookie, і лише в кінці – налаштування браузера.



```
builder.Services.AddLocalization(options =>
{
    options.ResourcesPath = "Resources";
});

var supportedCultureNames = new[] { "en-US", "uk-UA", "cs-CZ" };

builder.Services.Configure<RequestLocalizationOptions>(options =>
{
    var cultureInfos = supportedCultureNames
        .Select(name => new CultureInfo(name))
        .ToList();

    options.DefaultRequestCulture = new RequestCulture("en-US");
    options.SupportedCultures = cultureInfos;
    options.SupportedUICultures = cultureInfos;

    options.RequestCultureProviders = new List<IRequestCultureProvider>
    {
        new QueryStringCultureProvider(supportedCultureNames),
        new CookieRequestCultureProvider(),
        new AcceptLanguageHeaderRequestCultureProvider()
    };
});
```

Рисунок 5.10 – Фрагмент Program.cs з підключенням QueryStringCultureProvider і налаштуванням порядку провайдерів у RequestLocalizationOptions

Взаємодія з UI-перемикачем мови

Раніше, у завданні 7, був реалізований контролер SelectLanguageController та подання Views/SelectLanguage/Index.cshtml, які змінювали мову через запис того самого cookie (за допомогою CookieRequestCultureProvider.MakeCookieValue). Після додавання QueryStringCultureProvider обидва механізми працюють узгоджено:

- UI-перемикач мови записує cookie напряму;
- параметр ?culture=...:
    - застосовує культуру до поточного запиту;
    - оновлює той самий cookie, синхронізуючи стан між URL та UI-перемикачем.

Таким чином, незалежно від способу перемикання (кнопка в інтерфейсі чи параметр в URL), використовується одна й та сама точка істини – cookie локалізації.

Перевірка роботи перемикання через URL

Для перевірки виконано такі кроки:

1. Перехід на головну сторінку без параметра, наприклад: https://localhost:7081/ – сторінка відображається мовою за замовчуванням (en-US).
2. Перехід за посиланням: https://localhost:7081/?culture=uk-UA – інтерфейс одразу відображається українською, у cookie фіксується uk-UA.
3. Перехід на будь-яку іншу сторінку без параметра culture (наприклад, /Premium або /Home/Index) – застосунок продовжує працювати з культурою uk-UA завдяки збереженому cookie.
4. Аналогічно, при переході на ?culture=cs-CZ інтерфейс перемикається на чеську, і ця культура зберігається для наступних запитів.

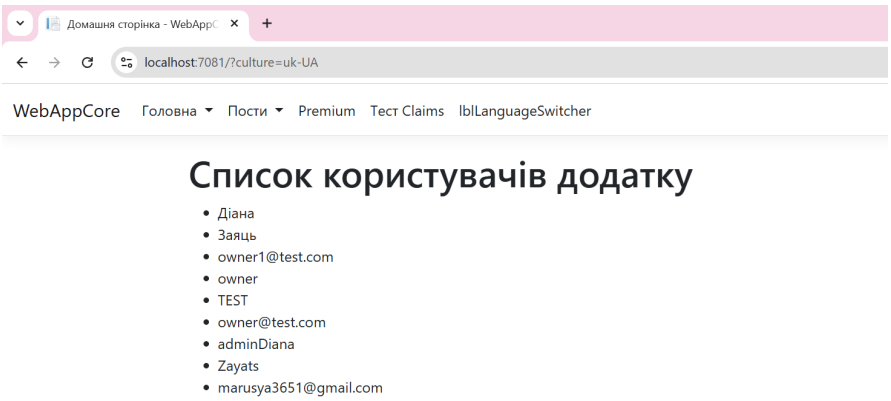


Рисунок 5.11 – Приклад роботи перемикання культури через URL: сторінка застосунку при виклику з параметром ?culture=uk-UA

Завдання 9. Реалізувати локалізацію форматів дат, чисел і валют у залежності від обраної культури.

У цьому завданні було додано демонстрацію того, як вибрана культура впливає не тільки на текстові ресурси, але й на форматування дат, чисел і грошових значень. Для цього використано сторінку *Premium*.

ViewModel для демонстрації локалізованих значень

У проєкті створено модель подання PremiumMetricsViewModel у папці WebAppCore/ViewModels. Модель містить три властивості:

- NextReviewDate – дата наступного оновлення/перегляду;
- ProductivityScore – показник продуктивності з дробовою частиною;
- SubscriptionFee – вартість підписки (тип decimal або double).

Дані властивості використовуються як джерело значень, які будуть по-різному відображатися залежно від поточної культури.

```
using System;

namespace WebAppCore.ViewModels
{
    4 references
    public class PremiumMetricsViewModel
    {
        2 references
        public DateTime NextReviewDate { get; set; }
        2 references
        public double ProductivityScore { get; set; }
        2 references
        public decimal SubscriptionFee { get; set; }
    }
}
```

Рисунок 5.12 – Клас PremiumMetricsViewModel з полями для дати, числового показника та вартості підписки

Формування моделі у PremiumController

У контролері PremiumController (файл WebAppCore/Controllers/PremiumController.cs) реалізовано дію, яка формує екземпляр PremiumMetricsViewModel з демонстраційними значеннями, наприклад:

- дата наступного оновлення (умовно 15.01.2026);
- індекс продуктивності (наприклад, 87.45);
- вартість підписки (наприклад, 249.99 у базовій валюті).

Створена модель передається в представлення Views/Premium/Index.cshtml.

```
var metrics = new PremiumMetricsViewModel
{
    NextReviewDate = DateTime.UtcNow.AddDays(14),
    ProductivityScore = 87.45,
    SubscriptionFee = 249.99m
};
```

Рисунок 5.13 – Метод контролера PremiumController, що створює PremiumMetricsViewModel і передає її до подання

Форматування в Razor-поданні з урахуванням CultureInfo.CurrentCulture

У поданні WebAppCore/Views/Premium/Index.cshtml значення з моделі виводяться з використанням поточної культури (CultureInfo.CurrentCulture) та стандартних форматних рядків .NET:

```
@{
    ViewData["Title"] = Localizer["Title"];
    var culture = CultureInfo.CurrentCulture;
}
```

Рисунок 5.14 – Фрагмент представлення Premium/Index.cshtml з форматуванням дати, числа та валюти через CultureInfo.CurrentCulture

Результат при зміні культури

Завдяки використанню CultureInfo.CurrentCulture одна й та сама сторінка *Premium* відображає різні формати без будь-яких умовних операторів у коді. Перевірка проводилася для культур en-US та cs-CZ:

- для en-US (англійська, США) приклад відображення:
    - дата: January 15, 2026;
    - індекс продуктивності: 87.45;
    - вартість підписки: $249.99.
- для cs-CZ (чеська, Чехія) при тих же вихідних значеннях:
    - дата: 15. ledna 2026;
    - індекс продуктивності: 87,45;
    - вартість підписки: 249,99 Kč.

Перемикання культури виконується або через UI-перемикач мови, або через параметр ?culture=... (див. завдання 8). У всіх випадках форматування автоматично підлаштовується під поточну культуру, без дублювання логіки.

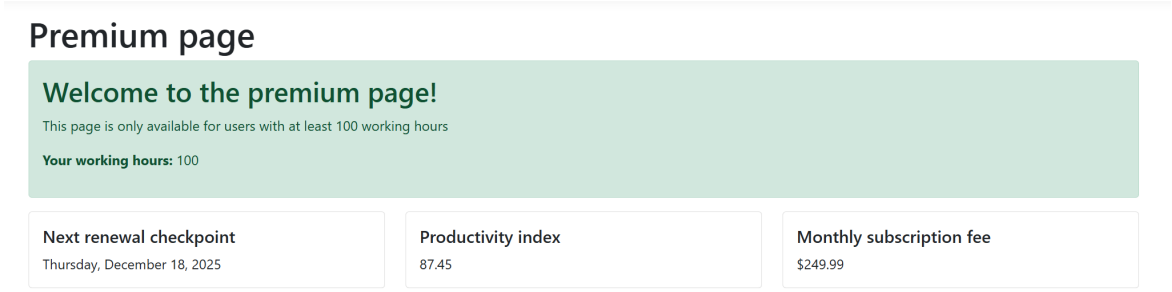


Рисунок 5.15 – Сторінка Premium при культурі en-US (формати дати, числа та валюти для США)

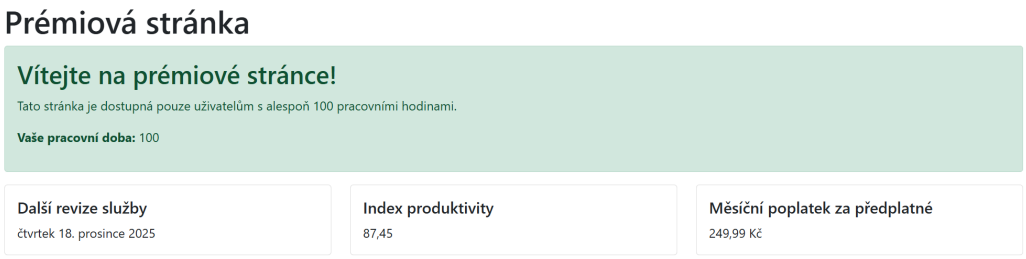


Рисунок 5.16 – Та сама сторінка Premium при культурі cs-CZ (інші роздільники та позначення валюти Kč)

Висновок:

У ході виконання лабораторної роботи було розглянуто та реалізовано механізми авторизації в ASP.NET Core на практиці. Налаштовано політики доступу на основі клеймів користувача, що дає змогу гнучко керувати доступом до окремих сторінок і ресурсів веб-застосунку. Додатково було опрацьовано імперативну авторизацію з перевіркою ресурсів, яка обмежує дії користувача його власними даними (зокрема, редагування лише власного допису на форумі). Отримані вміння забезпечують побудову безпечної системи доступу до різних частин застосунку та дозволяють реалізовувати складні правила контролю доступу.