МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра мережевих та інтернет технологій

СУЧАСНІ ІНТЕРНЕТ ТЕХНОЛОГІЇ

МОДЕЛІ ВІДОБРАЖЕННЯ. ВАЛІДАЦІЯ ДАНИХ У ЗАСТОСУНКУ ASP.NET CORE

Лабораторне заняття № 6

Черкун Єви Сергіївни

Хід виконання роботи:

Завдання 1. Ознайомитися з теоретичними основами валідації (сайт Microsoft).

Валідація — це процес перевірки правильності та відповідності даних певним правилам перед їх обробкою або збереженням. В ASP.NET Core вона поділяється на клієнтську та серверну:

1. Серверна валідація – перевірка даних на сервері після надсилання форми, гарантує безпеку та правильність обробки.
2. Клієнтська валідація – перевірка даних у браузері до відправки на сервер, підвищує зручність для користувача та зменшує навантаження на сервер.
3. Атрибути валідації – вбудовані правила, які можна застосовувати до властивостей моделей (наприклад, [Required], [StringLength], [Range]).
4. Валідаційний pipeline – послідовність кроків, через які проходять дані моделі: перевірка атрибутів, повідомлення про помилки, відображення у вигляді ValidationSummary або поруч з полем форми (asp-validation-for).

Завдання 2. Створити модель з щонайменше 5 властивостями, використовуючи стандартні атрибути валідації

1. [Required]
2. [StringLength]
3. [EmailAddress]
4. [Range]
5. [Compare]

Приклад наведений нижче на рисунку 6.1:

```
{
    [Required(ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName = "FullName_Required")]
    [StringLength(50, MinimumLength = 2, ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName
    [Display(Name = "Повне ім'я")]
    3 references
    public string FullName { get; set; } = string.Empty;

    [Required(ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName = "Email_Required")]
    [EmailAddress(ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName = "Email_EmailAddress")
    [Remote(action: "VerifyEmail", controller: "Validation", ErrorMessage = "Цей email вже зареєстрований")]
    [Display(Name = "Email")]
    4 references
    public string Email { get; set; } = string.Empty;

    [Required(ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName = "Age_Required")]
    [Range(18, 120, ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName = "Age_Range")]
    [Display(Name = "Вік")]
    3 references
    public int? Age { get; set; }

    [Required(ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName = "Password_Required")]
    [StringLength(100, MinimumLength = 8, ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName
    [DataType(DataType.Password)]
    [Display(Name = "Пароль")]
    3 references
    public string Password { get; set; } = string.Empty;

    [Required(ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName = "ConfirmPassword_Required
    [DataType(DataType.Password)]
    [Compare("Password", ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName = "ConfirmPasswo
    [Display(Name = "Підтвердження пароля")]
    3 references
    public string ConfirmPassword { get; set; } = string.Empty;
}
```

Рисунок 6.1 – Створена ContactViewModel із атрибутами валідації

Завдання 3. Реалізувати серверну валідацію у контролері через ModelState.IsValid. Забезпечити повернення помилок у View.

```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public IActionResult Index(ContactViewModel model)
{

    if (!ModelState.IsValid)
    {

        _logger.LogWarning("Форма зворотного зв'язку містить помилки валідації");
        return View(model);
    }


    _logger.LogInformation("Форма зворотного зв'язку успішно оброблена для користувача:


    return RedirectToAction(nameof(Success));
}
```

Рисунок 6.2 - Обробка відправки форми зворотного зв'язку з серверною валідацією

```
@model WebAppCore.ViewModels.ContactViewModel

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

<h4>Заповніть форму для зворотного зв'язку</h4>
<hr />

<div class="row">
    <div class="col-md-6">
        <form asp-action="Index" method="post">
            @Html.AntiForgeryToken()


            <div asp-validation-summary="All" class="text-danger mb-3"></div>
```

Рисунок 6.3 - Загальний summary помилок валідації у View

Завдання 4. Налаштувати клієнтську валідацію у Razor View: asp-validation-for, asp validation-summary, підключення _ValidationScriptsPartial.

ASP.NET Core читає атрибути валідації з ContactViewModel та генерує HTML з data-атрибутами (`data-val`, `data-val-required`, `data-val-length` тощо). jQuery Validation автоматично читає ці атрибути та налаштовує клієнтську валідацію без додаткового JavaScript коду. Валідація виконується при введенні даних та при відправці форми.

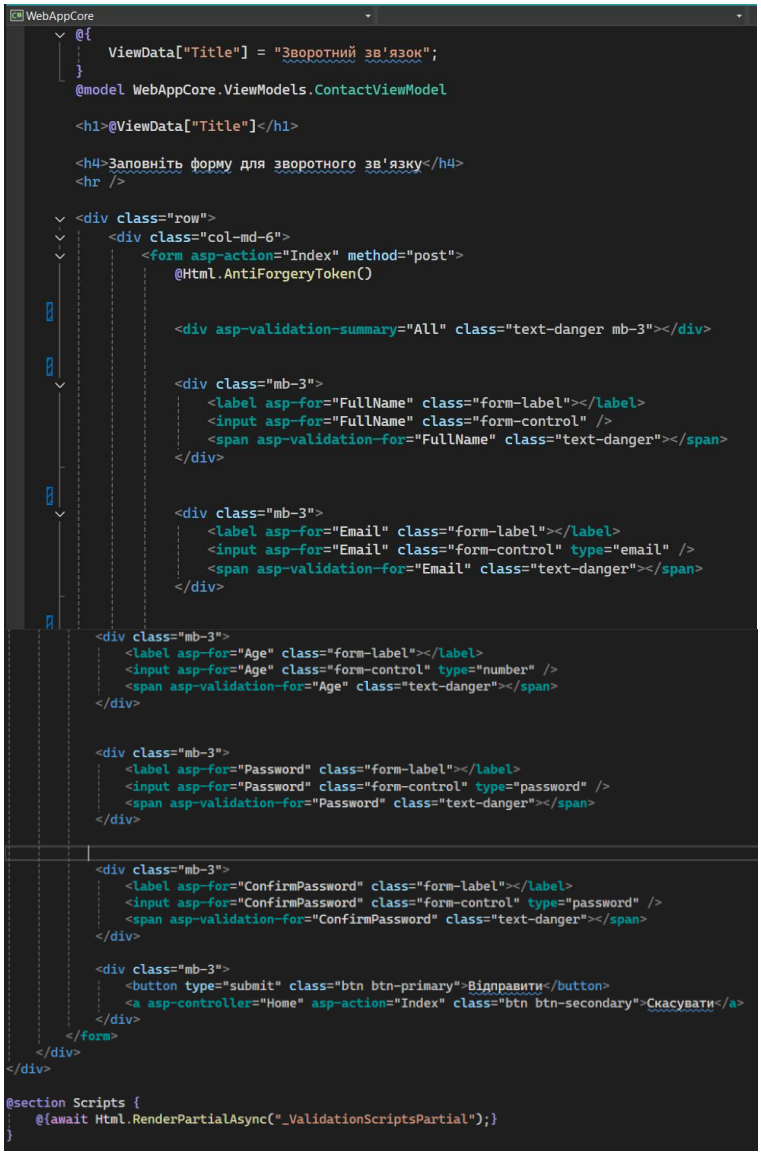


Рисунок 6.4 - Налаштування клієнтської валідації у Razor View

Завдання 5. Реалізувати Remote Validation для перевірки унікальності email або логіна. Створити відповідний метод у контролері.

Remote Validation - це тип валідації, при якому перевірка виконується на сервері через асинхронний HTTP-запит, але інтегрується з клієнтською валідацією.

Метод контролера для Remote Validation:

```
[AcceptVerbs("GET", "POST")]
0 references
public async Task<IActionResult> VerifyEmail(string email)
{
    if (string.IsNullOrEmpty(email))
    {
        return Json("Email є обов'язковим полем");
    }

    var user = await _userManager.FindByEmailAsync(email);
    if (user != null)
    {
        return Json("Цей email вже зареєстрований");
    }

    return Json(true);
}
```

Рисунок 6.5 - Remote Validation для перевірки унікальності

Валідація при введенні:

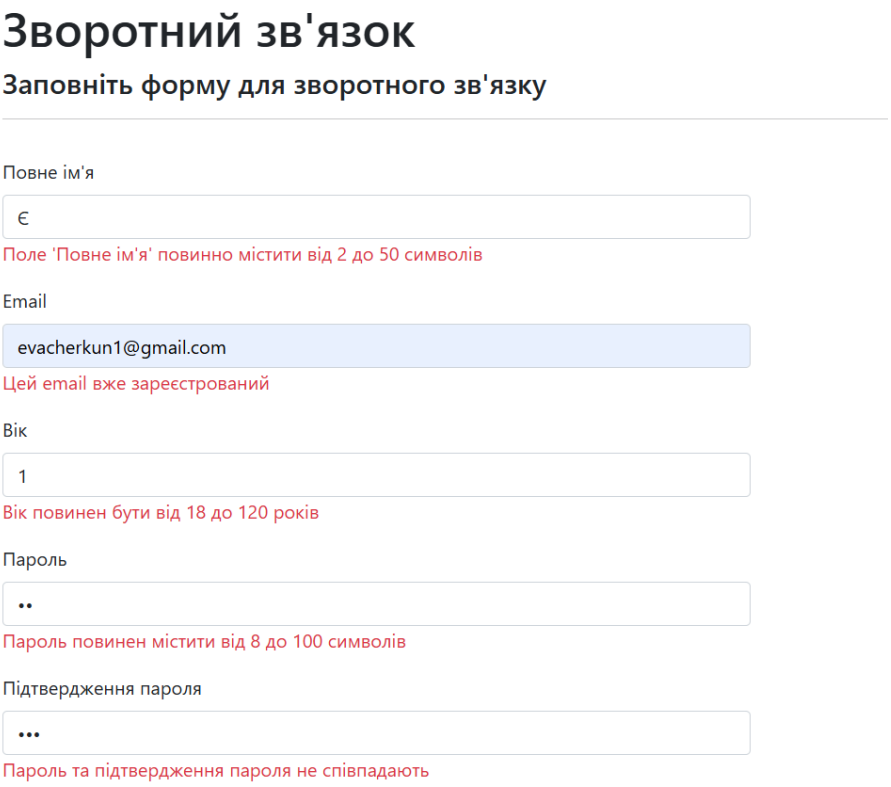


Рисунок 6.6 – Виконання валідації при заповненні форми (asp-validation-for)

Валідація при відправленні форми:

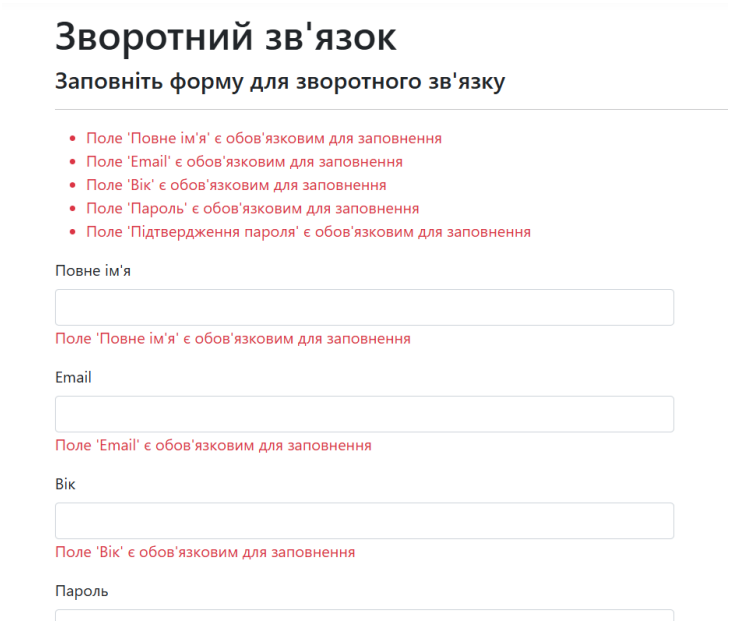


Рисунок 6.7 – Виконання валідації при відправці форми (asp-validation-summary)

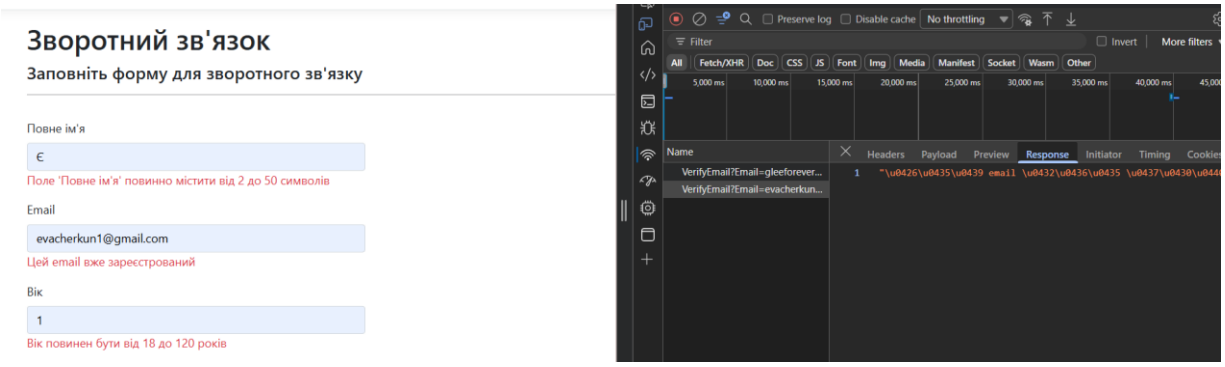


Рисунок 6.8 – Відповідь сервера якщо користувач зареєстрований

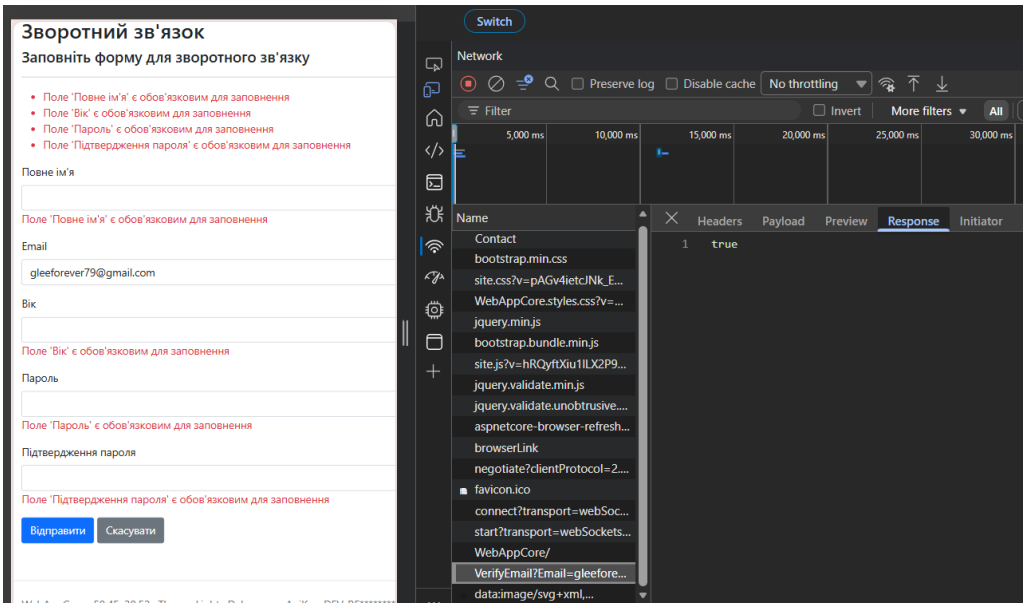


Рисунок 6.9 – Відповідь сервера якщо email унікальний

Завдання 6. Реалізувати залежну Remote Validation з використанням AdditionalFields. (адаптувати завдання для своєї предметної області).

```
        [AcceptVerbs("GET", "POST")]
        public IActionResult VerifyCookingTime(int? cookingTime, string
difficulty)
        {

            if (!cookingTime.HasValue)
            {
                return Json(true);
            }


            if (string.IsNullOrWhiteSpace(difficulty))
            {
                return Json(true);
            }

            var difficultyRules = new Dictionary<string, (int min, int
max)>
            {
                { "Easy", (5, 30) },
                { "Medium", (15, 60) },
                { "Hard", (30, 180) }
            };


            var normalizedDifficulty = difficulty.Trim();

            if (difficultyRules.TryGetValue(normalizedDifficulty, out var
rule))
            {
                if (cookingTime.Value >= rule.min && cookingTime.Value <=
rule.max)
                {
                    return Json(true);
                }
                else if (cookingTime.Value < rule.min)
                {
                    return Json($"Для
{GetDifficultyName(normalizedDifficulty)} рецепту час приготування повинен
бути не менше {rule.min} хвилин");
                }
                else
                {
                    return Json($"Для
{GetDifficultyName(normalizedDifficulty)} рецепту час приготування повинен
бути не більше {rule.max} хвилин");
                }
            }

            return Json("Невідома складність рецепту");
```

```
        }


        private string GetDifficultyName(string difficulty)
        {
            return difficulty switch
            {
                "Easy" => "простого",
                "Medium" => "середнього",
                "Hard" => "складного",
                _ => difficulty
            };
        }
    }
}
```

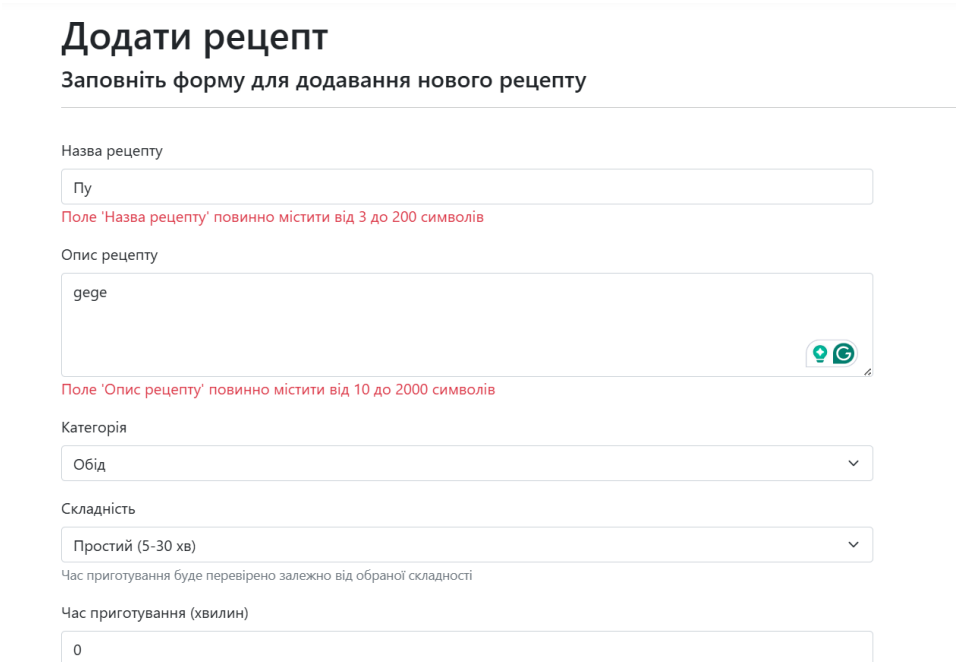


Рисунок 6.10 – Приклад валідації при додаванні рецепту

Завдання 7. Продемонструвати локалізацію повідомлень про помилки валідації через ресурсні файли.

Клас ValidationMessages використовується для централізованого зберігання та отримання локалізованих повідомлень валідації з ресурсних файлів. Він забезпечує підтримку багатомовності для атрибутів валідації в ASP.NET Core.

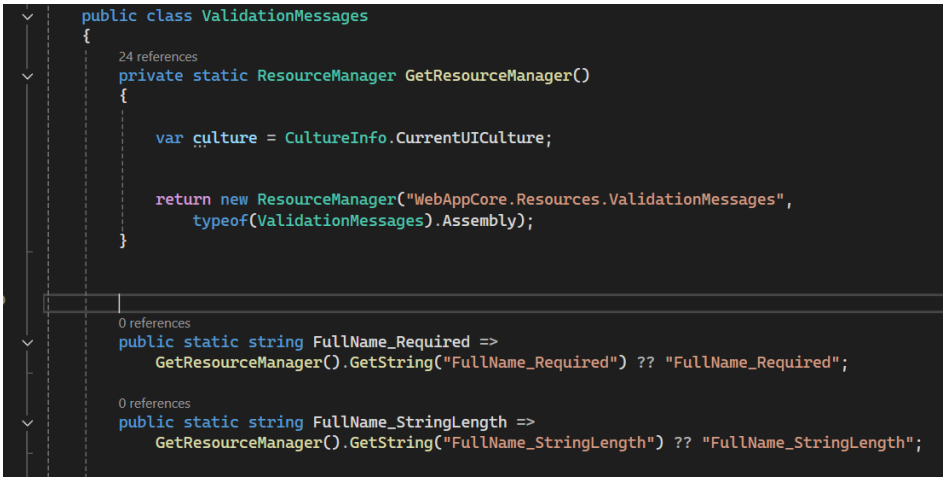
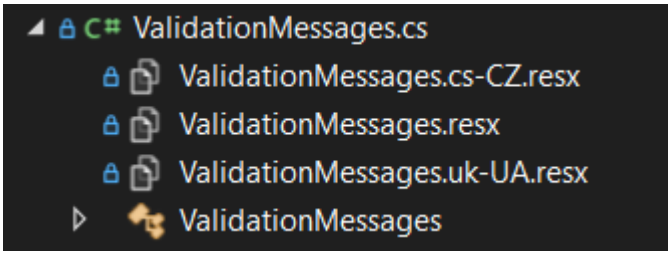


Рисунок 6.11 – ValidationMessages клас

У класі використовується ResourceManager, який завантажує ресурсні файли:



6.12 – Ресурсні файли для трьох мов

Приклад локалізації:

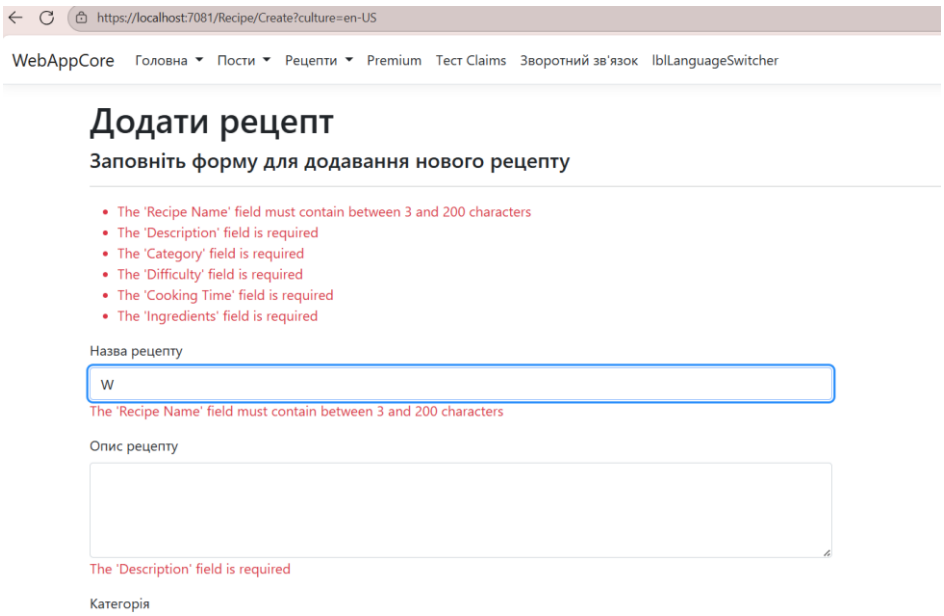


Рисунок 6.13 - Локалізація про помилки валідації англійською

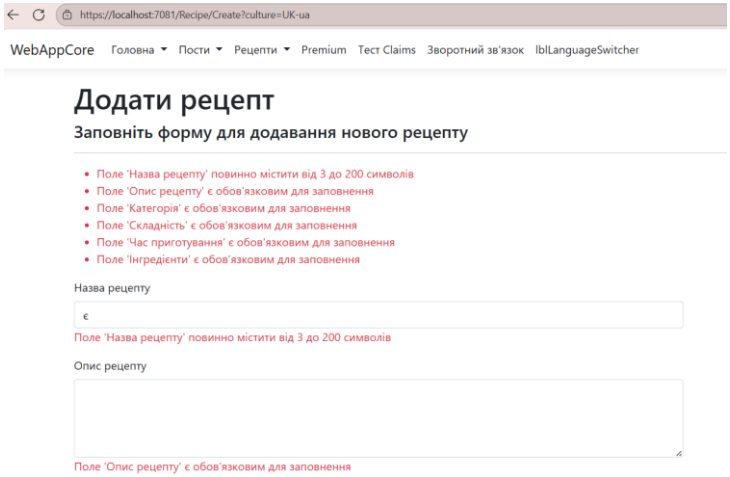


Рисунок 6.14 - Локалізація про помилки валідації українською

Висновок:
У ході роботи було вивчено теоретичні основи валідації в ASP.NET Core та реалізовано всі основні її механізми: серверну й клієнтську валідацію, перевірку через стандартні атрибути моделі, Remote Validation та залежну Remote Validation. Також було налаштовано повернення помилок у View і впроваджено локалізовані повідомлення через ресурсні файли. Це дозволило комплексно засвоїти принципи перевірки даних у веб-застосунках.