Міністерство освіти України

Національний технічний університет "ХПІ"

кафедра "Комп’ютерних наук та інформаційних технологій"

**Звіт**

**Лабораторна робота 4**

з дисципліни "Проєктування серверних застосунків"

Виконав: студент групи КН-922В

Мельник Б. В.

Перевірив:

Івашко А. В.

Харків 2025

**Зміст**

[Завдання 1 4](#_Toc194010076)

[Текст програми 4](#_Toc194010077)

[Результати роботи програми 14](#_Toc194010078)

[Алгоритм побудови кодів 15](#_Toc194010079)

[Функціональність 15](#_Toc194010080)

[Висновок 16](#_Toc194010081)

[Завдання 2 17](#_Toc194010082)

[Текст програми 17](#_Toc194010083)

[Результати роботи програми 27](#_Toc194010084)

[Алгоритм побудови кодів 28](#_Toc194010085)

[Функціональність 29](#_Toc194010086)

[Висновок 29](#_Toc194010087)

[Завдання 3 31](#_Toc194010088)

[Текст програми 31](#_Toc194010089)

[Результати роботи програми 41](#_Toc194010090)

[Алгоритм побудови кодів 42](#_Toc194010091)

[Функціональність 43](#_Toc194010092)

[Висновок 43](#_Toc194010093)

[Завдання 4 44](#_Toc194010094)

[Текст програми 44](#_Toc194010095)

[Результати роботи програми 55](#_Toc194010096)

[Алгоритм побудови кодів 56](#_Toc194010097)

[Функціональність 57](#_Toc194010098)

[Висновок 57](#_Toc194010099)

[Завдання 5 58](#_Toc194010100)

[Текст програми 58](#_Toc194010101)

[Результати роботи програми 68](#_Toc194010102)

[Алгоритм побудови кодів 69](#_Toc194010103)

[Функціональність 70](#_Toc194010104)

[Висновок 70](#_Toc194010105)

[Завдання 6 72](#_Toc194010106)

[Текст програми 72](#_Toc194010107)

[Результати роботи програми 83](#_Toc194010108)

[Алгоритм побудови кодів 84](#_Toc194010109)

[Функціональність 85](#_Toc194010110)

[Висновок 85](#_Toc194010111)

[Завдання 7 86](#_Toc194010112)

[Текст програми 86](#_Toc194010113)

[Результати роботи програми 92](#_Toc194010114)

[Алгоритм побудови кодів 93](#_Toc194010115)

[Функціональність 94](#_Toc194010116)

[Висновок: 94](#_Toc194010117)

[Завдання 8 95](#_Toc194010118)

[Текст програми 95](#_Toc194010119)

[Результати роботи програми 104](#_Toc194010120)

[Алгоритм побудови кодів 105](#_Toc194010121)

[Функціональність 106](#_Toc194010122)

[Висновок 106](#_Toc194010123)

[Загальний Висновок 108](#_Toc194010124)

**Тема:** Застосування технології Razor для реалізації інтерфейсної логіки по-дання (View) MVC-застосунків.

**Мета роботи:** ознайомитися із можливостями технології Razor та придбати практику у використанні засобів Razor для оптимального формування та об-робки інформації на view.

**Програмні засоби розробки:** середовище програмування (IDE-Integrated De-

velopment Environment) MS Visual Studio, програмний фреймворк ASP.NET версії не нижче 6.0 та дуже бажано використання дизайнерського фреймво-рку BOOTSTRAP, або якогось іншого відповідного аналогу, MySQL Server, або якась аналогічна СУБД.

**Індивідуальні завдання**

**Варіант №12**

# **Завдання 1**

Виконати перше завдання минулої роботи згідно свого варіанту (Варіант №12), створюючи текстові блоки на той самій сторінці (поданні), де знаходиться початкова форма із відповідними параметрами, тобто у застосунку має бути одне подання (Index) та один контролер.

## **Текст програми:**

**Index.cshtml:**

@{

// Встановлюємо layout та заголовок сторінки

Layout = "~/Views/Shared/\_Layout.cshtml";

ViewData["Title"] = "Анкета відвідувача ресторану";

}

<div class="form-container">

<h2 class="form-title">Анкета відвідувача ресторану</h2>

<!-- Форма надсилається методом POST на Home/Index -->

<form method="post">

<!-- Контактна інформація -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Контактна інформація</th>

</tr>

</thead>

<tbody>

<tr>

<td>Ім’я</td>

<td><input type="text" name="Name" class="form-control" /></td>

</tr>

<tr>

<td>Телефон</td>

<td><input type="text" name="Phone" class="form-control" /></td>

</tr>

<tr>

<td>Пошта</td>

<td><input type="email" name="Email" class="form-control" /></td>

</tr>

<tr>

<td>Дата відвідування</td>

<td><input type="date" name="VisitDate" class="form-control" /></td>

</tr>

</tbody>

</table>

<!-- Персональна інформація -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Персональна інформація</th>

</tr>

</thead>

<tbody>

<tr>

<td>Вік</td>

<td>

<select name="Age" class="form-select">

<option value="">-- Оберіть категорію --</option>

<option>До 18</option>

<option>18-25</option>

<option>26-35</option>

<option>36-45</option>

<option>46 і старше</option>

</select>

</td>

</tr>

<tr>

<td>Улюблена кухня</td>

<td>

<select name="Cuisine" class="form-select">

<option value="">-- Оберіть кухню --</option>

<option>Грецька</option>

<option>Італійська</option>

<option>Французька</option>

<option>Японська</option>

<option>Українська</option>

</select>

</td>

</tr>

<tr>

<td>Які страви ви б хотіли бачити в меню?</td>

<td>

<textarea name="Dishes" class="form-control"></textarea>

</td>

</tr>

</tbody>

</table>

<!-- Оцінка ресторану -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Оцінка нашого ресторану</th>

</tr>

</thead>

<tbody>

<tr>

<td>Чому обрали нашу установу?</td>

<td>

<!-- Радіокнопки для вибору причини -->

<div>

<input type="radio" name="Reason" value="Недалеко від роботи/дому" id="reason1" />

<label for="reason1">Недалеко від роботи/дому</label>

</div>

<div>

<input type="radio" name="Reason" value="Побачив рекламу" id="reason2" />

<label for="reason2">Побачив рекламу</label>

</div>

<div>

<input type="radio" name="Reason" value="Порадили" id="reason3" />

<label for="reason3">Порадили</label>

</div>

<div style="white-space: nowrap;">

<input type="radio" name="Reason" value="Оптимальне співвідношення ціни та якості" id="reason4" />

<label for="reason4">Оптимальне співвідношення ціни та якості</label>

</div>

</td>

</tr>

<tr>

<td>Чи будете рекомендувати нашу установу друзям та знайомим?</td>

<td>

<input type="radio" name="Recommend" value="Так" id="recYes" />

<label for="recYes">Так</label>

<br />

<input type="radio" name="Recommend" value="Ні" id="recNo" />

<label for="recNo">Ні</label>

</td>

</tr>

</tbody>

</table>

<!-- Кнопки для відправки та скидання форми -->

<div class="text-center mt-3">

<button type="submit" class="btn-custom me-3">Відіслати</button>

<!-- При натисканні "Скинути" форма перезавантажується -->

<button type="button" onclick="window.location.href='@Url.Action("Index", "Home")'" class="btn-custom btn-reset">Скинути</button>

</div>

</form>

</div>

<!-- Вивід даних, якщо користувач уже надіслав форму -->

@if (ViewBag.Name != null)

{

<div class="data-display">

<h4>Введені дані:</h4>

<ul>

<li><strong>Ім’я:</strong> @ViewBag.Name</li>

<li><strong>Телефон:</strong> @ViewBag.Phone</li>

<li><strong>Пошта:</strong> @ViewBag.Email</li>

<li><strong>Дата відвідування:</strong> @ViewBag.VisitDate</li>

<li><strong>Вік:</strong> @ViewBag.Age</li>

<li><strong>Улюблена кухня:</strong> @ViewBag.Cuisine</li>

<li><strong>Бажані страви:</strong> @ViewBag.Dishes</li>

<li><strong>Причина вибору:</strong> @ViewBag.Reason</li>

<li><strong>Чи рекомендуватимете?</strong> @ViewBag.Recommend</li>

</ul>

</div>

}

**HomeController.cs:**

using Microsoft.AspNetCore.Mvc;

namespace Task\_1.Controllers

{

public class HomeController : Controller

{

// GET: /Home/Index - показуємо форму

[HttpGet]

public IActionResult Index()

{

// Просто повертаємо представлення (View)

return View();

}

// POST: /Home/Index - обробка даних форми

[HttpPost]

[ActionName("Index")]

public IActionResult IndexPost()

{

// Отримуємо дані з форми через Request.Form

string name = Request.Form["Name"];

string phone = Request.Form["Phone"];

string email = Request.Form["Email"];

string visitDate = Request.Form["VisitDate"];

string age = Request.Form["Age"];

string cuisine = Request.Form["Cuisine"];

string dishes = Request.Form["Dishes"];

string reason = Request.Form["Reason"];

string recommend = Request.Form["Recommend"];

// Записуємо дані у ViewBag для виводу у представленні

ViewBag.Name = name;

ViewBag.Phone = phone;

ViewBag.Email = email;

ViewBag.VisitDate = visitDate;

ViewBag.Age = age;

ViewBag.Cuisine = cuisine;

ViewBag.Dishes = dishes;

ViewBag.Reason = reason;

ViewBag.Recommend = recommend;

// Повертаємо ту саму сторінку з отриманими даними

return View();

}

}

}

**\_Layout.cshtml:**

<!DOCTYPE html>

<html lang="uk">

<head>

<meta charset="utf-8" />

<!-- Заголовок, який комбінує назву сторінки та назву завдання -->

<title>@ViewData["Title"] - Task\_1</title>

<!-- Підключення Bootstrap для стандартних стилів -->

<link rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" />

<!-- Підключення власних стилів -->

<**link** rel="stylesheet" href="~/css/site.css" />

</head>

<body>

<!-- Хедер сайту -->

<header class="p-3 bg-dark text-white">

<div class="container">

<h1 class="mb-0">Завдання 1 (Варіант №12)</h1>

</div>

</header>

<!-- Основний блок контенту -->

<div class="container">

<main role="main" class="pb-3">

@RenderBody() <!-- Вставка контенту сторінки -->

</main>

</div>

<!-- Футер сайту -->

<footer class="border-top pt-3 mt-3">

<div class="container">

<p>&copy; 2025 - Task\_1</p>

</div>

</footer>

<!-- Підключення скриптів Bootstrap -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>

@RenderSection("Scripts", required: false)

</body>

</html>

**site.css:**

/\* Основні стилі для сторінки \*/

/\* Фонове зображення для body \*/

body {

margin: 0;

padding: 0;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background: url("https://images7.alphacoders.com/116/1162515.jpg") no-repeat center center fixed;

background-size: cover;

}

/\* Стилізація контейнера для форми \*/

.form-container {

background-color: rgba(255, 255, 255, 0.85);

margin: 30px auto;

padding: 20px;

border-radius: 10px;

max-width: 700px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

}

/\* Стиль заголовку форми \*/

.form-title {

text-align: center;

margin-bottom: 20px;

font-weight: bold;

font-size: 24px;

}

/\* Стилізація кнопок \*/

.btn-custom {

background-color: #4CAF50;

color: #fff;

border: none;

padding: 10px 20px;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.2s;

}

/\* Ефект при наведенні на кнопку \*/

.btn-custom:hover {

background-color: #45a049;

}

/\* Стиль кнопки скидання (червона кнопка) \*/

.btn-reset {

background-color: #f44336 !important;

}

/\* Ефект при наведенні на кнопку скидання \*/

.btn-reset:hover {

background-color: #e53935 !important;

}

/\* Стилізація блоку для відображення введених даних \*/

.data-display {

background-color: rgba(255, 255, 255, 0.85);

margin: 20px auto;

padding: 15px;

border-radius: 10px;

max-width: 700px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

}

/\* Вирівнювання заголовку даних \*/

.data-display h4 {

text-align: center;

}

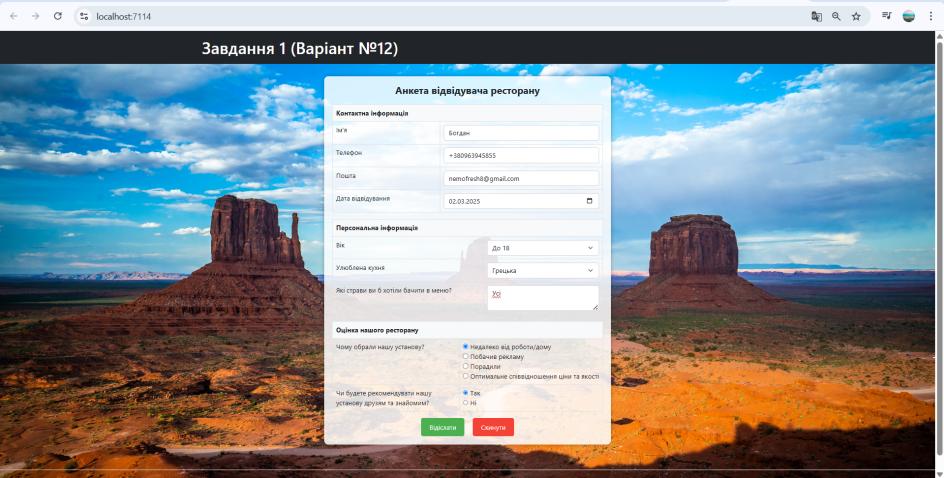
/\* Стиль тексту у футері \*/

footer p {

color: gold;

}

## **Результати роботи програми:**



Изображение выглядит как текст, облако, небо, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, облако, небо, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

## **Алгоритм побудови кодів**

1. **Розробка шаблону (\_Layout.cshtml):**
   * Створено основний шаблон сторінки, який містить заголовок, хедер, футер, а також підключення стилів (Bootstrap та власного CSS) і скриптів.
2. **Створення подання (Index.cshtml):**
   * Розроблено HTML-форму для введення даних анкети (контактна інформація, персональні дані, оцінка ресторану).
   * Використано різні елементи управління (input, select, textarea, radio) для збору даних.
   * Додано логіку виводу даних: за умови, що дані вже введено (перевірка ViewBag), дані відображаються під формою.
3. **Реалізація контролера (HomeController.cs):**
   * Визначено метод GET для відображення форми.
   * Визначено метод POST для обробки даних форми:
     + Дані отримуються через Request.Form.
     + Дані записуються у ViewBag.
     + Повертається те саме подання з відображенням введених даних.
4. **Стилізація (site.css):**
   * Створено файл стилів для оформлення фону, контейнера форми, кнопок та інших елементів, що забезпечує приємний вигляд сторінки.

## **Функціональність**

* **Ввід даних:**

Користувач заповнює анкету із контактною та персональною інформацією, а також зазначає свої побажання та оцінку ресторану.

* **Обробка та відображення:**

Після натискання кнопки «Відіслати» введені дані надсилаються на сервер методом POST. Контролер отримує ці дані, зберігає їх у ViewBag, і ті самі дані відображаються на тій же сторінці.

* **Реалізація логіки:**

Уся логіка обробки даних здійснюється безпосередньо у поданні за допомогою технології Razor, а контролер виконує роль посередника.

## **Висновок**

* Завдання виконано відповідно до вимог варіанту №12 з використанням єдиного подання (Index) та одного контролера (HomeController).
* Забезпечено ефективну інтеграцію форми з логікою обробки даних завдяки використанню Razor-синтаксису, що дозволяє виконувати всі необхідні вирази, умови та цикли у самому поданні.
* Реалізований підхід сприяє простоті підтримки та подальшому розвитку застосунку, забезпечуючи при цьому зручний інтерфейс для користувача.

# **Завдання 2**

Виконати друге завдання минулої роботи згідно свого варіанту, створюючи відповідні топології зображень на той самій сторінці (поданні), де знаходиться початкова форма із відповідними параметрами, тобто у застосунку має бути одне подання (Index) та один контролер.

## **Текст програми:**

**Index.cshtml:**

@{

// Використовуємо спільний layout та задаємо заголовок сторінки

Layout = "~/Views/Shared/\_Layout.cshtml";

ViewData["Title"] = "Анкета відвідувача ресторану";

}

<div class="form-container">

<h2 class="form-title">Анкета відвідувача ресторану</h2>

<!-- Форма, що надсилає дані методом POST на Home/Index -->

<form method="post">

<!-- Блок з контактною інформацією -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Контактна інформація</th>

</tr>

</thead>

<tbody>

<tr>

<td>Ім’я</td>

<td><input type="text" name="Name" class="form-control" /></td>

</tr>

<tr>

<td>Телефон</td>

<td><input type="text" name="Phone" class="form-control" /></td>

</tr>

<tr>

<td>Пошта</td>

<td><input type="email" name="Email" class="form-control" /></td>

</tr>

<tr>

<td>Дата відвідування</td>

<td><input type="date" name="VisitDate" class="form-control" /></td>

</tr>

</tbody>

</table>

<!-- Блок з персональною інформацією -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Персональна інформація</th>

</tr>

</thead>

<tbody>

<tr>

<td>Вік</td>

<td>

<select name="Age" class="form-select">

<option value="">-- Оберіть категорію --</option>

<option>До 18</option>

<option>18-25</option>

<option>26-35</option>

<option>36-45</option>

<option>46 і старше</option>

</select>

</td>

</tr>

<tr>

<td>Улюблена кухня</td>

<td>

<select name="Cuisine" class="form-select">

<option value="">-- Оберіть кухню --</option>

<option>Грецька</option>

<option>Італійська</option>

<option>Французька</option>

<option>Японська</option>

<option>Українська</option>

</select>

</td>

</tr>

<tr>

<td>Які страви ви б хотіли бачити в меню?</td>

<td>

<textarea name="Dishes" class="form-control"></textarea>

</td>

</tr>

</tbody>

</table>

<!-- Блок з оцінкою ресторану -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Оцінка нашого ресторану</th>

</tr>

</thead>

<tbody>

<tr>

<td>Чому обрали нашу установу?</td>

<td>

<div>

<input type="radio" name="Reason" value="Недалеко від роботи/дому" id="reason1" />

<label for="reason1">Недалеко від роботи/дому</label>

</div>

<div>

<input type="radio" name="Reason" value="Побачив рекламу" id="reason2" />

<label for="reason2">Побачив рекламу</label>

</div>

<div>

<input type="radio" name="Reason" value="Порадили" id="reason3" />

<label for="reason3">Порадили</label>

</div>

<div style="white-space: nowrap;">

<input type="radio" name="Reason" value="Оптимальне співвідношення ціни та якості" id="reason4" />

<label for="reason4">Оптимальне співвідношення ціни та якості</label>

</div>

</td>

</tr>

<tr>

<td>Чи будете рекомендувати нашу установу друзям та знайомим?</td>

<td>

<input type="radio" name="Recommend" value="Так" id="recYes" />

<label for="recYes">Так</label>

<br />

<input type="radio" name="Recommend" value="Ні" id="recNo" />

<label for="recNo">Ні</label>

</td>

</tr>

</tbody>

</table>

<!-- Кнопки для відправки та скидання форми -->

<div class="text-center mt-3">

<button type="submit" class="btn-custom me-3">Відіслати</button>

<button type="button" onclick="window.location.href='@Url.Action("Index", "Home")'" class="btn-custom btn-reset">Скинути</button>

</div>

</form>

</div>

<!-- Блок з топологією зображень, реалізований засобами Razor -->

<div class="image-topology mt-4">

<h3 class="text-center">Галерея зображень</h3>

<div class="row">

@{

// Створюємо список URL зображень (логіка виконана в Razor)

var imageUrls = new List<string>

{

"https://images.unsplash.com/photo-1504674900247-0877df9cc836",

"https://images.unsplash.com/photo-1517248135467-4c7edcad34c4",

"https://images.unsplash.com/photo-1528605248644-14dd04022da1",

"https://images.unsplash.com/photo-1506784365847-bbad939e9335"

};

}

@foreach (var url in imageUrls)

{

<div class="col-md-3 col-sm-6 mb-3">

<img src="@url" alt="Image" class="img-fluid rounded" />

</div>

}

</div>

</div>

<!-- Відображення даних, якщо форма відправлена -->

@if (ViewBag.Name != null)

{

<div class="data-display mt-4">

<h4>Введені дані:</h4>

<ul>

<li><strong>Ім’я:</strong> @ViewBag.Name</li>

<li><strong>Телефон:</strong> @ViewBag.Phone</li>

<li><strong>Пошта:</strong> @ViewBag.Email</li>

<li><strong>Дата відвідування:</strong> @ViewBag.VisitDate</li>

<li><strong>Вік:</strong> @ViewBag.Age</li>

<li><strong>Улюблена кухня:</strong> @ViewBag.Cuisine</li>

<li><strong>Бажані страви:</strong> @ViewBag.Dishes</li>

<li><strong>Причина вибору:</strong> @ViewBag.Reason</li>

<li><strong>Чи рекомендуватимете?</strong> @ViewBag.Recommend</li>

</ul>

</div>

}

**HomeController.cs:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Http;

namespace Task\_2.Controllers

{

public class HomeController : Controller

{

// GET: /Home/Index - показуємо порожню форму

[HttpGet]

public IActionResult Index()

{

return View();

}

// POST: /Home/Index - отримуємо дані форми

[HttpPost]

public IActionResult Index(IFormCollection form)

{

// Отримуємо значення з форми через IFormCollection

string name = form["Name"];

string phone = form["Phone"];

string email = form["Email"];

string visitDate = form["VisitDate"];

string age = form["Age"];

string cuisine = form["Cuisine"];

string dishes = form["Dishes"];

string reason = form["Reason"];

string recommend = form["Recommend"];

// Передаємо дані у View через ViewBag для відображення

ViewBag.Name = name;

ViewBag.Phone = phone;

ViewBag.Email = email;

ViewBag.VisitDate = visitDate;

ViewBag.Age = age;

ViewBag.Cuisine = cuisine;

ViewBag.Dishes = dishes;

ViewBag.Reason = reason;

ViewBag.Recommend = recommend;

return View();

}

}

}

**\_Layout.cshtml:**

<!DOCTYPE html>

<html lang="uk">

<head>

<meta charset="utf-8" />

<!-- Динамічний заголовок сторінки -->

<title>@ViewData["Title"] - Task\_2</title>

<!-- Підключення Bootstrap для базових стилів -->

<link rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" />

<!-- Підключення власних стилів -->

<**link** rel="stylesheet" href="~/css/site.css" />

</head>

<body>

<!-- Шапка сайту -->

<header class="p-3 bg-dark text-white">

<div class="container">

<h1 class="mb-0">Завдання 2 (Варіант №12)</h1>

</div>

</header>

<!-- Основний контент сайту -->

<div class="container">

<main role="main" class="pb-3">

@RenderBody() <!-- Тут буде підставлено контент конкретної сторінки -->

</main>

</div>

<!-- Футер сайту -->

<footer class="border-top pt-3 mt-3">

<div class="container">

<p>&copy; 2025 - Task\_2</p>

</div>

</footer>

<!-- Підключення скриптів Bootstrap -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>

@RenderSection("Scripts", required: false)

</body>

</html>

**site.css:**

/\* Основні стилі для сайту \*/

/\* Фонове зображення для сторінки \*/

body {

margin: 0;

padding: 0;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background: url("https://images4.alphacoders.com/122/1222302.jpg") no-repeat center center fixed;

background-size: cover;

}

/\* Стилізація контейнера для форми \*/

.form-container {

background-color: rgba(255, 255, 255, 0.85);

margin: 30px auto;

padding: 20px;

border-radius: 10px;

max-width: 700px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

}

/\* Стиль заголовку форми \*/

.form-title {

text-align: center;

margin-bottom: 20px;

font-weight: bold;

font-size: 24px;

}

/\* Стилізація кнопок \*/

.btn-custom {

background-color: #4CAF50;

color: #fff;

border: none;

padding: 10px 20px;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.2s;

}

.btn-custom:hover {

background-color: #45a049;

}

.btn-reset {

background-color: #f44336 !important;

}

.btn-reset:hover {

background-color: #e53935 !important;

}

/\* Блок для відображення даних, які ввів користувач \*/

.data-display {

background-color: rgba(255, 255, 255, 0.85);

margin: 20px auto;

padding: 15px;

border-radius: 10px;

max-width: 700px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

}

.data-display h4 {

text-align: center;

}

/\* Стилі для блоку з зображеннями \*/

.image-topology {

background-color: rgba(255, 255, 255, 0.85);

padding: 20px;

border-radius: 10px;

margin-bottom: 30px;

}

footer p {

color: gold;

}

## **Результати роботи програми:**

Изображение выглядит как текст, снимок экрана, программное обеспечение, облако

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, небо, облако

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, облако, небо

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, гора

Контент, сгенерированный ИИ, может содержать ошибки.

## **Алгоритм побудови кодів**

1. **Створення спільного шаблону (\_Layout.cshtml):**
   * Налаштовано заголовок, хедер, футер, підключено Bootstrap та власні стилі (site.css).
2. **Розробка подання (Index.cshtml):**
   * Розроблено HTML-форму для введення даних анкети, що включає контактну та персональну інформацію, а також оцінку ресторану.
   * Додано блок з топологією зображень, де за допомогою Razor створено список URL зображень, який перебирається циклом для відображення в сітці.
3. **Реалізація контролера (HomeController.cs):**
   * Створено метод GET для відображення пустої форми.
   * Створено метод POST, що отримує дані через IFormCollection, записує їх у ViewBag та повертає подання з відображенням введених даних.
4. **Стилізація (site.css):**
   * Визначено стилі для фону, контейнера форми, блоку з даними та зображень, що забезпечують сучасний та зручний інтерфейс.

## **Функціональність**

* **Ввід даних:**

Користувач заповнює анкету із контактною, персональною інформацією та оцінкою ресторану.

* **Обробка форми:**

Після натискання кнопки «Відіслати» дані відправляються методом POST, контролер отримує їх через IFormCollection і передає у ViewBag для виведення.

* **Динамічна топологія зображень:**

На сторінці розміщено галерею, яка формуються засобами Razor:

* + Створюється список URL зображень.
  + За допомогою циклу foreach зображення розташовуються в сітці, що забезпечує адаптивне відображення.
* **Цілісність логіки:**

Уся логіка (вирази, умови, цикли) реалізована в поданні за допомогою Razor, тоді як контролер виступає як посередник для передачі даних.

## **Висновок**

* Завдання реалізовано згідно вимог Варіанту №12, використовуючи одне подання (Index) та один контролер (HomeController).
* Додатковий блок з топологією зображень підвищує візуальну привабливість застосунку та демонструє можливості динамічного формування контенту через Razor.
* Підхід сприяє простоті підтримки коду та зручності подальшого розвитку застосунку, забезпечуючи інтегроване відображення даних та динамічних елементів на одній сторінці.

# **Завдання 3**

Виконати третє завдання минулої роботи згідно свого варіанту, створюючи відповідну таблицю на той самій сторінці (поданні), де знаходиться початкова форма із відповідними параметрами, тобто у застосунку має бути одне подання (Index) та один контролер.

## **Текст програми:**

**Index.cshtml:**

@{

// Встановлюємо загальний layout та задаємо заголовок сторінки

Layout = "~/Views/Shared/\_Layout.cshtml";

ViewData["Title"] = "Анкета відвідувача ресторану";

}

<div class="form-container">

<h2 class="form-title">Анкета відвідувача ресторану</h2>

<!-- Форма надсилається методом POST на Home/Index -->

<form method="post">

<!-- Таблиця з контактною інформацією -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Контактна інформація</th>

</tr>

</thead>

<tbody>

<tr>

<td>Ім’я</td>

<td><input type="text" name="Name" class="form-control" /></td>

</tr>

<tr>

<td>Телефон</td>

<td><input type="text" name="Phone" class="form-control" /></td>

</tr>

<tr>

<td>Пошта</td>

<td><input type="email" name="Email" class="form-control" /></td>

</tr>

<tr>

<td>Дата відвідування</td>

<td><input type="date" name="VisitDate" class="form-control" /></td>

</tr>

</tbody>

</table>

<!-- Таблиця з персональною інформацією -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Персональна інформація</th>

</tr>

</thead>

<tbody>

<tr>

<td>Вік</td>

<td>

<select name="Age" class="form-select">

<option value="">-- Оберіть категорію --</option>

<option>До 18</option>

<option>18-25</option>

<option>26-35</option>

<option>36-45</option>

<option>46 і старше</option>

</select>

</td>

</tr>

<tr>

<td>Улюблена кухня</td>

<td>

<select name="Cuisine" class="form-select">

<option value="">-- Оберіть кухню --</option>

<option>Грецька</option>

<option>Італійська</option>

<option>Французька</option>

<option>Японська</option>

<option>Українська</option>

</select>

</td>

</tr>

<tr>

<td>Які страви ви б хотіли бачити в меню?</td>

<td>

<textarea name="Dishes" class="form-control"></textarea>

</td>

</tr>

</tbody>

</table>

<!-- Таблиця з оцінкою ресторану -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Оцінка нашого ресторану</th>

</tr>

</thead>

<tbody>

<tr>

<td>Чому обрали нашу установу?</td>

<td>

<div>

<input type="radio" name="Reason" value="Недалеко від роботи/дому" id="reason1" />

<label for="reason1">Недалеко від роботи/дому</label>

</div>

<div>

<input type="radio" name="Reason" value="Побачив рекламу" id="reason2" />

<label for="reason2">Побачив рекламу</label>

</div>

<div>

<input type="radio" name="Reason" value="Порадили" id="reason3" />

<label for="reason3">Порадили</label>

</div>

<div style="white-space: nowrap;">

<input type="radio" name="Reason" value="Оптимальне співвідношення ціни та якості" id="reason4" />

<label for="reason4">Оптимальне співвідношення ціни та якості</label>

</div>

</td>

</tr>

<tr>

<td>Чи будете рекомендувати нашу установу друзям та знайомим?</td>

<td>

<input type="radio" name="Recommend" value="Так" id="recYes" />

<label for="recYes">Так</label>

<br />

<input type="radio" name="Recommend" value="Ні" id="recNo" />

<label for="recNo">Ні</label>

</td>

</tr>

</tbody>

</table>

<!-- Кнопки для відправки форми та її скидання -->

<div class="text-center mt-3">

<button type="submit" class="btn-custom me-3">Відіслати</button>

<button type="button" onclick="window.location.href='@Url.Action("Index", "Home")'" class="btn-custom btn-reset">Скинути</button>

</div>

</form>

</div>

<!-- Вивід даних у вигляді таблиці з чіткими бордюрами, якщо користувач заповнив форму -->

@if (ViewBag.Name != null)

{

<div class="data-display">

<h4>Введені дані:</h4>

<table class="table table-bordered table-striped">

<thead>

<tr>

<th>Поле</th>

<th>Значення</th>

</tr>

</thead>

<tbody>

<tr>

<td>Ім’я</td>

<td>@ViewBag.Name</td>

</tr>

<tr>

<td>Телефон</td>

<td>@ViewBag.Phone</td>

</tr>

<tr>

<td>Пошта</td>

<td>@ViewBag.Email</td>

</tr>

<tr>

<td>Дата відвідування</td>

<td>@ViewBag.VisitDate</td>

</tr>

<tr>

<td>Вік</td>

<td>@ViewBag.Age</td>

</tr>

<tr>

<td>Улюблена кухня</td>

<td>@ViewBag.Cuisine</td>

</tr>

<tr>

<td>Бажані страви</td>

<td>@ViewBag.Dishes</td>

</tr>

<tr>

<td>Причина вибору</td>

<td>@ViewBag.Reason</td>

</tr>

<tr>

<td>Чи рекомендуватимете?</td>

<td>@ViewBag.Recommend</td>

</tr>

</tbody>

</table>

</div>

}

**HomeController.cs:**

using Microsoft.AspNetCore.Mvc;

namespace Task\_3.Controllers

{

public class HomeController : Controller

{

// GET: /Home/Index - показуємо форму для введення даних

[HttpGet]

public IActionResult Index()

{

return View();

}

// POST: /Home/Index - отримуємо дані з форми

[HttpPost]

public IActionResult Index(string Name, string Phone, string Email, string VisitDate,

string Age, string Cuisine, string Dishes,

string Reason, string Recommend)

{

// Заносимо отримані дані у ViewBag для відображення у вигляді таблиці

ViewBag.Name = Name;

ViewBag.Phone = Phone;

ViewBag.Email = Email;

ViewBag.VisitDate = VisitDate;

ViewBag.Age = Age;

ViewBag.Cuisine = Cuisine;

ViewBag.Dishes = Dishes;

ViewBag.Reason = Reason;

ViewBag.Recommend = Recommend;

// Повертаємо той самий View, щоб показати дані

return View();

}

}

}

**\_Layout.cshtml:**

<!DOCTYPE html>

<html lang="uk">

<head>

<meta charset="utf-8" />

<!-- Динамічний заголовок, який формується із ViewData -->

<title>@ViewData["Title"] - Task\_3</title>

<!-- Підключення Bootstrap для стилізації -->

<link rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" />

<!-- Підключення власних стилів -->

<link rel="stylesheet" href="~/css/site.css" />

</head>

<body>

<!-- Шапка сайту -->

<header class="p-3 bg-dark text-white">

<div class="container">

<h1 class="mb-0">Завдання 3 (Варіант №12)</h1>

</div>

</header>

<!-- Основний блок, де розміщується контент конкретної сторінки -->

<div class="container">

<main role="main" class="pb-3">

@RenderBody() <!-- Тут підставляється вміст сторінки -->

</main>

</div>

<!-- Футер сайту -->

<footer class="border-top pt-3 mt-3">

<div class="container">

<p>&copy; 2025 - Task\_3</p>

</div>

</footer>

<!-- Підключення скриптів Bootstrap -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>

@RenderSection("Scripts", required: false)

</body>

</html>

**site.css:**

/\* Основні стилі для сайту \*/

/\* Фонове зображення, яке покриває весь екран \*/

body {

margin: 0;

padding: 0;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background: url("https://images8.alphacoders.com/112/1120753.jpg") no-repeat center center fixed;

background-size: cover;

}

/\* Стилізація контейнера, в якому знаходиться форма \*/

.form-container {

background-color: rgba(255, 255, 255, 0.85);

margin: 30px auto;

padding: 20px;

border-radius: 10px;

max-width: 700px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

}

/\* Заголовок форми \*/

.form-title {

text-align: center;

margin-bottom: 20px;

font-weight: bold;

font-size: 24px;

}

/\* Стилізація кнопок для взаємодії \*/

.btn-custom {

background-color: #4CAF50;

color: #fff;

border: none;

padding: 10px 20px;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.2s;

}

.btn-custom:hover {

background-color: #45a049;

}

.btn-reset {

background-color: #f44336 !important;

}

.btn-reset:hover {

background-color: #e53935 !important;

}

/\* Стилізація блоку для відображення введених даних \*/

.data-display {

background-color: rgba(255, 255, 255, 0.85);

margin: 20px auto;

padding: 15px;

border-radius: 10px;

max-width: 700px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

}

.data-display table {

border-collapse: collapse;

width: 100%;

}

.data-display table th,

.data-display table td {

border: 2px solid #000 !important; /\* чорний колір, товщина 2px \*/

padding: 8px;

}

.data-display h4 {

text-align: center;

}

footer p {

color: gold;

}

## **Результати роботи програми:**

Изображение выглядит как текст, облако, снимок экрана, небо

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как облако, текст, снимок экрана, небо

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, облако, снимок экрана, небо

Контент, сгенерированный ИИ, может содержать ошибки.

## **Алгоритм побудови кодів**

1. **Створення спільного шаблону (\_Layout.cshtml):**
   * Налаштовано заголовок, хедер та футер сторінки.
   * Підключено Bootstrap та власні стилі (site.css) для забезпечення відповідного вигляду.
2. **Розробка подання (Index.cshtml):**
   * Реалізовано форму, що складається з трьох таблиць для введення контактної, персональної інформації та оцінки ресторану.
   * Дані форми надсилаються методом POST на один і той же контролер.
3. **Обробка даних контролером (HomeController.cs):**
   * Метод GET відображає порожню форму для введення даних.
   * Метод POST отримує дані (через параметри або Model Binding), зберігає їх у ViewBag та повертає подання для відображення введених даних.
4. **Відображення даних:**
   * За допомогою Razor здійснюється умовне відображення блоку, де дані користувача виводяться у вигляді таблиці з чіткими бордюрами.
5. **Стилізація (site.css):**
   * Оформлено зовнішній вигляд контейнера, форми, кнопок та таблиці, що дозволяє досягти сучасного та читабельного інтерфейсу.

## **Функціональність**

* **Ввід даних:**

Користувач заповнює анкету з контактною, персональною інформацією та оцінкою ресторану за допомогою зручної форми.

* **Обробка та відображення:**

Після натискання кнопки «Відіслати» введені дані надсилаються на сервер, де контролер отримує інформацію та зберігає її у ViewBag, після чого дані відображаються на тій самій сторінці у вигляді таблиці.

* **Використання технології Razor:**

Уся необхідна логіка (вирази, умови, цикли) реалізована безпосередньо у поданні, що дозволяє ефективно обробляти та відображати дані без залучення додаткових контролерів.

## **Висновок**

* Завдання виконано відповідно до вимог варіанту №12, з використанням одного подання (Index) та одного контролера (HomeController).
* Реалізовано інтегроване відображення форми для введення даних та таблиці з результатами, що підвищує зручність користувача.
* Використання Razor для реалізації всієї логіки сприяє простоті підтримки та подальшому розвитку застосунку.

# **Завдання 4**

Виконати четверте завдання минулої роботи згідно свого варіанту (Варіант №12), реалізуючи переходи між поданнями за допомогою одного контролеру, а формування чергового вмісту на одному поданні, тобто у застосунку має бути одне подання (Index) та один контролер.

## **Текст програми:**

**Index.cshtml:**

@model Task\_4.Models.VisitorViewModel

@{

// Встановлюємо загальний layout та задаємо заголовок сторінки

Layout = "~/Views/Shared/\_Layout.cshtml";

ViewData["Title"] = "Анкета відвідувача ресторану";

}

<div class="form-container">

<h2 class="form-title">Анкета відвідувача ресторану</h2>

@using (Html.BeginForm("Index", "Home", FormMethod.Post))

{

<!-- Таблиця для контактної інформації -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Контактна інформація</th>

</tr>

</thead>

<tbody>

<tr>

<td>@Html.LabelFor(m => m.Name)</td>

<td>@Html.TextBoxFor(m => m.Name, new { @class = "form-control" })</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.Phone)</td>

<td>@Html.TextBoxFor(m => m.Phone, new { @class = "form-control" })</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.Email)</td>

<td>@Html.TextBoxFor(m => m.Email, new { @class = "form-control", type = "email" })</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.VisitDate)</td>

<td>@Html.TextBoxFor(m => m.VisitDate, "{0:yyyy-MM-dd}", new { @class = "form-control", type = "date" })</td>

</tr>

</tbody>

</table>

<!-- Таблиця для персональної інформації -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Персональна інформація</th>

</tr>

</thead>

<tbody>

<tr>

<td>@Html.LabelFor(m => m.Age)</td>

<td>

@Html.DropDownListFor(m => m.Age, new SelectList(new[] {

new { Value = "", Text = "-- Оберіть категорію --" },

new { Value = "До 18", Text = "До 18" },

new { Value = "18-25", Text = "18-25" },

new { Value = "26-35", Text = "26-35" },

new { Value = "36-45", Text = "36-45" },

new { Value = "46 і старше", Text = "46 і старше" }

}, "Value", "Text"), new { @class = "form-select" })

</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.Cuisine)</td>

<td>

@Html.DropDownListFor(m => m.Cuisine, new SelectList(new[] {

new { Value = "", Text = "-- Оберіть кухню --" },

new { Value = "Грецька", Text = "Грецька" },

new { Value = "Італійська", Text = "Італійська" },

new { Value = "Французька", Text = "Французька" },

new { Value = "Японська", Text = "Японська" },

new { Value = "Українська", Text = "Українська" }

}, "Value", "Text"), new { @class = "form-select" })

</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.Dishes)</td>

<td>@Html.TextAreaFor(m => m.Dishes, new { @class = "form-control" })</td>

</tr>

</tbody>

</table>

<!-- Таблиця для оцінки ресторану -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Оцінка нашого ресторану</th>

</tr>

</thead>

<tbody>

<tr>

<td>@Html.LabelFor(m => m.Reason)</td>

<td style="white-space: nowrap;">

<span>

<input type="radio" name="Reason" value="Недалеко від роботи/дому" id="reason1"

@(Model.Reason == "Недалеко від роботи/дому" ? "checked" : "") />

<label for="reason1">Недалеко від роботи/дому</label>

</span>

<br />

<span>

<input type="radio" name="Reason" value="Побачив рекламу" id="reason2"

@(Model.Reason == "Побачив рекламу" ? "checked" : "") />

<label for="reason2">Побачив рекламу</label>

</span>

<br />

<span>

<input type="radio" name="Reason" value="Порадили" id="reason3"

@(Model.Reason == "Порадили" ? "checked" : "") />

<label for="reason3">Порадили</label>

</span>

<br />

<span>

<input type="radio" name="Reason" value="Оптимальне співвідношення ціни та якості" id="reason4"

@(Model.Reason == "Оптимальне співвідношення ціни та якості" ? "checked" : "") />

<label for="reason4">Оптимальне співвідношення ціни та якості</label>

</span>

</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.Recommend)</td>

<td>

<input type="radio" name="Recommend" value="Так" id="recYes" @(Model.Recommend == "Так" ? "checked" : "") />

<label for="recYes">Так</label>

<br />

<input type="radio" name="Recommend" value="Ні" id="recNo" @(Model.Recommend == "Ні" ? "checked" : "") />

<label for="recNo">Ні</label>

</td>

</tr>

</tbody>

</table>

<!-- Кнопки для відправки форми та її скидання -->

<div class="text-center mt-3">

<button type="submit" class="btn-custom me-3">Відіслати</button>

<button type="button" onclick="window.location.href='@Url.Action("Index", "Home")'" class="btn-custom btn-reset">Скинути</button>

</div>

}

</div>

<!-- Вивід даних у вигляді таблиці з чіткими чорними бордюрами -->

@if (Model.Name != null)

{

<div class="data-display">

<h4>Введені дані:</h4>

<table class="table custom-bordered-table table-striped">

<thead>

<tr>

<th>Поле</th>

<th>Значення</th>

</tr>

</thead>

<tbody>

<tr>

<td>Ім’я</td>

<td>@Model.Name</td>

</tr>

<tr>

<td>Телефон</td>

<td>@Model.Phone</td>

</tr>

<tr>

<td>Пошта</td>

<td>@Model.Email</td>

</tr>

<tr>

<td>Дата відвідування</td>

<td>@(Model.VisitDate.HasValue ? Model.VisitDate.Value.ToString("yyyy-MM-dd") : "")</td>

</tr>

<tr>

<td>Вік</td>

<td>@Model.Age</td>

</tr>

<tr>

<td>Улюблена кухня</td>

<td>@Model.Cuisine</td>

</tr>

<tr>

<td>Бажані страви</td>

<td>@Model.Dishes</td>

</tr>

<tr>

<td>Причина вибору</td>

<td>@Model.Reason</td>

</tr>

<tr>

<td>Чи рекомендуватимете?</td>

<td>@Model.Recommend</td>

</tr>

</tbody>

</table>

</div>

}

**VisitorViewModel.cs:**

using System;

using System.ComponentModel.DataAnnotations;

namespace Task\_4.Models

{

public class VisitorViewModel

{

[Display(Name = "Ім'я")]

public string? Name { get; set; }

[Display(Name = "Телефон")]

public string? Phone { get; set; }

[Display(Name = "Пошта")]

[EmailAddress]

public string? Email { get; set; }

[Display(Name = "Дата відвідування")]

[DataType(DataType.Date)]

public DateTime? VisitDate { get; set; }

[Display(Name = "Вік")]

public string? Age { get; set; }

[Display(Name = "Улюблена кухня")]

public string? Cuisine { get; set; }

[Display(Name = "Які страви ви б хотіли бачити в меню?")]

public string? Dishes { get; set; }

[Display(Name = "Чому обрали нашу установу?")]

public string? Reason { get; set; }

[Display(Name = "Чи будете рекомендувати нашу установу друзям та знайомим?")]

public string? Recommend { get; set; }

}

}

**HomeController.cs:**

using Microsoft.AspNetCore.Mvc;

using Task\_4.Models;

namespace Task\_4.Controllers

{

public class HomeController : Controller

{

// GET: /Home/Index - відображаємо порожню форму

[HttpGet]

public IActionResult Index()

{

return View(new VisitorViewModel());

}

// POST: /Home/Index - отримуємо дані з форми та повертаємо їх у вигляді моделі

[HttpPost]

public IActionResult Index(VisitorViewModel model)

{

// За потреби тут можна додати перевірку ModelState.IsValid

return View(model);

}

}

}

**\_Layout.cshtml:**

<!DOCTYPE html>

<html lang="uk">

<head>

<meta charset="utf-8" />

<!-- Заголовок сторінки, що формується через ViewData -->

<title>@ViewData["Title"] - Task\_4</title>

<!-- Підключення Bootstrap для базових стилів -->

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" />

<!-- Підключення власних стилів -->

<link rel="stylesheet" href="~/css/site.css" />

</head>

<body>

<!-- Шапка сайту -->

<header class="p-3 bg-dark text-white">

<div class="container">

<h1 class="mb-0">Завдання 4 (Варіант №12)</h1>

</div>

</header>

<!-- Основний контейнер для контенту -->

<div class="container">

<main role="main" class="pb-3">

@RenderBody()

</main>

</div>

<!-- Футер сайту -->

<footer class="border-top pt-3 mt-3">

<div class="container">

<p>&copy; 2025 - Task\_4</p>

</div>

</footer>

<!-- Підключення скриптів Bootstrap -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>

@RenderSection("Scripts", required: false)

</body>

</html>

**site.css:**

/\* Фонове зображення для всієї сторінки \*/

body {

margin: 0;

padding: 0;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background: url("https://4kwallpapers.com/images/wallpapers/teton-range-rocky-mountains-wyoming-usa-mirror-lake-2560x1440-4274.jpg") no-repeat center center fixed;

background-size: cover;

}

/\* Стилізація контейнера для форми \*/

.form-container {

background-color: rgba(255, 255, 255, 0.85);

margin: 30px auto;

padding: 20px;

border-radius: 10px;

max-width: 700px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

}

/\* Стилізація заголовку форми \*/

.form-title {

text-align: center;

margin-bottom: 20px;

font-weight: bold;

font-size: 24px;

}

/\* Стилізація кнопок \*/

.btn-custom {

background-color: #4CAF50;

color: #fff;

border: none;

padding: 10px 20px;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.2s;

}

.btn-custom:hover {

background-color: #45a049;

}

.btn-reset {

background-color: #f44336 !important;

}

.btn-reset:hover {

background-color: #e53935 !important;

}

/\* Стилізація блоку для відображення введених даних \*/

.data-display {

background-color: rgba(255, 255, 255, 0.85);

margin: 20px auto;

padding: 15px;

border-radius: 10px;

max-width: 700px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

}

.data-display h4 {

text-align: center;

}

footer p {

color: gold;

}

/\* Користувацькі стилі для таблиці з чіткими чорними бордюрами \*/

.custom-bordered-table {

border-collapse: collapse;

width: 100%;

}

.custom-bordered-table th,

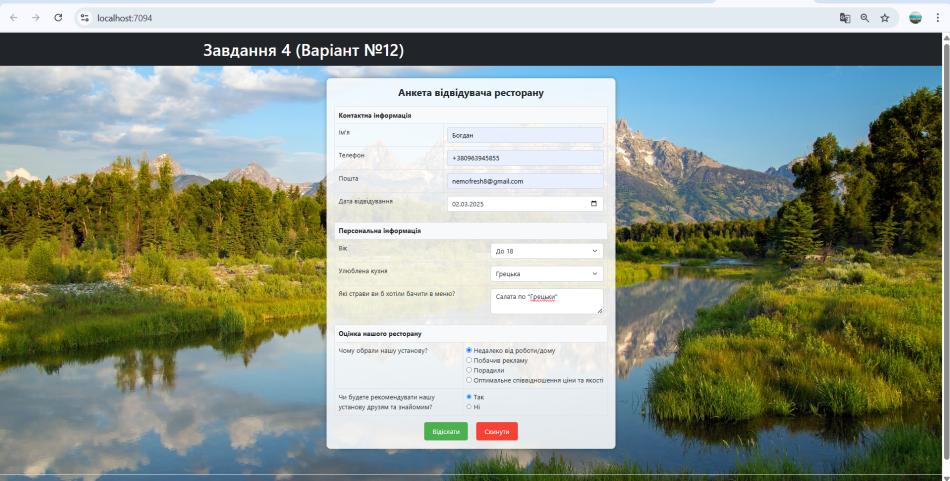
.custom-bordered-table td {

border: 2px solid #000 !important;

padding: 8px;

}

## **Результати роботи програми:**



Изображение выглядит как текст, вода, облако, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, вода, снимок экрана, дерево

Контент, сгенерированный ИИ, может содержать ошибки.

## **Алгоритм побудови кодів**

1. **Створення спільного шаблону (\_Layout.cshtml):**
   * Налаштовано заголовок, хедер та футер сторінки.
   * Підключено Bootstrap та власні стилі (site.css) для забезпечення сучасного дизайну.
2. **Розробка подання (Index.cshtml):**
   * Подання базується на сильно типізованій моделі (VisitorViewModel).
   * За допомогою Html-хелперів (Html.BeginForm, Html.TextBoxFor, Html.DropDownListFor, тощо) створено форму для введення контактної, персональної інформації та оцінки ресторану.
   * Використано Razor для умовного відображення даних (таблиця з введеними значеннями) після відправки форми.
3. **Реалізація контролера (HomeController.cs):**
   * Метод GET відображає порожню форму, створюючи екземпляр VisitorViewModel.
   * Метод POST приймає модель із заповненими даними та повертає її ж для відображення результатів на тому ж поданні.
4. **Модель (VisitorViewModel.cs):**
   * Визначено властивості для всіх полів форми з використанням атрибутів для відображення і валідації.

## **Функціональність**

* **Ввід даних:**

Користувач заповнює анкету з контактною та персональною інформацією, а також зазначає оцінку ресторану.

* **Обробка та відображення:**

Після натискання кнопки «Відіслати» дані передаються методом POST до одного контролера, який повертає подання з актуальною моделлю.  
Введені дані відображаються у вигляді таблиці з чіткими чорними бордюрами.

* **Перехід між станами:**

Всі переходи та формування контенту виконуються на одному поданні за допомогою Razor, що спрощує логіку застосунку та забезпечує єдність роботи.

* **Валідація та підтримка:**

Використання сильно типізованої моделі сприяє кращій організації коду та можливості подальшої розширюваності.

## **Висновок**

* Завдання реалізовано згідно вимог Варіанту №12, використовуючи єдине подання (Index) та один контролер (HomeController).
* Підхід із застосуванням сильно типізованої моделі та Razor-хелперів забезпечує ефективну обробку даних, чітке розділення логіки та відображення, а також спрощує підтримку коду.
* Реалізація переходів між різними станами форми на одному поданні дозволяє створити зручний та інтуїтивно зрозумілий інтерфейс для користувача.

# **Завдання 5**

Модифікувати минуле завдання додавши відповідні лічильники відвідувань до кожного подання.

## **Текст програми:**

**Index.cshtml:**

@model Task\_5.Models.VisitorViewModel

@{

// Встановлюємо загальний layout та задаємо заголовок сторінки

Layout = "~/Views/Shared/\_Layout.cshtml";

ViewData["Title"] = "Анкета відвідувача ресторану";

}

<!-- Лічильник відвідувань (реалізовано за допомогою Razor) -->

@{

// Збільшуємо лічильник відвідувань кожного разу при завантаженні сторінки

VisitCounter++;

}

<p class="text-center fw-bold">Кількість відвідувань цієї сторінки: @VisitCounter</p>

<div class="form-container">

<h2 class="form-title">Анкета відвідувача ресторану</h2>

@using (Html.BeginForm("Index", "Home", FormMethod.Post))

{

<!-- Таблиця для контактної інформації -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Контактна інформація</th>

</tr>

</thead>

<tbody>

<tr>

<td>@Html.LabelFor(m => m.Name)</td>

<td>@Html.TextBoxFor(m => m.Name, new { @class = "form-control" })</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.Phone)</td>

<td>@Html.TextBoxFor(m => m.Phone, new { @class = "form-control" })</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.Email)</td>

<td>@Html.TextBoxFor(m => m.Email, new { @class = "form-control", type = "email" })</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.VisitDate)</td>

<td>@Html.TextBoxFor(m => m.VisitDate, "{0:yyyy-MM-dd}", new { @class = "form-control", type = "date" })</td>

</tr>

</tbody>

</table>

<!-- Таблиця для персональної інформації -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Персональна інформація</th>

</tr>

</thead>

<tbody>

<tr>

<td>@Html.LabelFor(m => m.Age)</td>

<td>

@Html.DropDownListFor(m => m.Age, new SelectList(new[] {

new { Value = "", Text = "-- Оберіть категорію --" },

new { Value = "До 18", Text = "До 18" },

new { Value = "18-25", Text = "18-25" },

new { Value = "26-35", Text = "26-35" },

new { Value = "36-45", Text = "36-45" },

new { Value = "46 і старше", Text = "46 і старше" }

}, "Value", "Text"), new { @class = "form-select" })

</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.Cuisine)</td>

<td>

@Html.DropDownListFor(m => m.Cuisine, new SelectList(new[] {

new { Value = "", Text = "-- Оберіть кухню --" },

new { Value = "Грецька", Text = "Грецька" },

new { Value = "Італійська", Text = "Італійська" },

new { Value = "Французька", Text = "Французька" },

new { Value = "Японська", Text = "Японська" },

new { Value = "Українська", Text = "Українська" }

}, "Value", "Text"), new { @class = "form-select" })

</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.Dishes)</td>

<td>@Html.TextAreaFor(m => m.Dishes, new { @class = "form-control" })</td>

</tr>

</tbody>

</table>

<!-- Таблиця для оцінки ресторану -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Оцінка нашого ресторану</th>

</tr>

</thead>

<tbody>

<tr>

<td>@Html.LabelFor(m => m.Reason)</td>

<td style="white-space: nowrap;">

<span>

<input type="radio" name="Reason" value="Недалеко від роботи/дому" id="reason1"

@(Model.Reason == "Недалеко від роботи/дому" ? "checked" : "") />

<label for="reason1">Недалеко від роботи/дому</label>

</span>

<br />

<span>

<input type="radio" name="Reason" value="Побачив рекламу" id="reason2"

@(Model.Reason == "Побачив рекламу" ? "checked" : "") />

<label for="reason2">Побачив рекламу</label>

</span>

<br />

<span>

<input type="radio" name="Reason" value="Порадили" id="reason3"

@(Model.Reason == "Порадили" ? "checked" : "") />

<label for="reason3">Порадили</label>

</span>

<br />

<span>

<input type="radio" name="Reason" value="Оптимальне співвідношення ціни та якості" id="reason4"

@(Model.Reason == "Оптимальне співвідношення ціни та якості" ? "checked" : "") />

<label for="reason4">Оптимальне співвідношення ціни та якості</label>

</span>

</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.Recommend)</td>

<td>

<input type="radio" name="Recommend" value="Так" id="recYes" @(Model.Recommend == "Так" ? "checked" : "") />

<label for="recYes">Так</label>

<br />

<input type="radio" name="Recommend" value="Ні" id="recNo" @(Model.Recommend == "Ні" ? "checked" : "") />

<label for="recNo">Ні</label>

</td>

</tr>

</tbody>

</table>

<!-- Кнопки для відправки форми та її скидання -->

<div class="text-center mt-3">

<button type="submit" class="btn-custom me-3">Відіслати</button>

<button type="button" onclick="window.location.href='@Url.Action("Index", "Home")'" class="btn-custom btn-reset">Скинути</button>

</div>

}

</div>

<!-- Вивід даних у вигляді таблиці з чіткими чорними бордюрами -->

@if (Model.Name != null)

{

<div class="data-display">

<h4>Введені дані:</h4>

<table class="table custom-bordered-table table-striped">

<thead>

<tr>

<th>Поле</th>

<th>Значення</th>

</tr>

</thead>

<tbody>

<tr>

<td>Ім’я</td>

<td>@Model.Name</td>

</tr>

<tr>

<td>Телефон</td>

<td>@Model.Phone</td>

</tr>

<tr>

<td>Пошта</td>

<td>@Model.Email</td>

</tr>

<tr>

<td>Дата відвідування</td>

<td>@(Model.VisitDate.HasValue ? Model.VisitDate.Value.ToString("yyyy-MM-dd") : "")</td>

</tr>

<tr>

<td>Вік</td>

<td>@Model.Age</td>

</tr>

<tr>

<td>Улюблена кухня</td>

<td>@Model.Cuisine</td>

</tr>

<tr>

<td>Бажані страви</td>

<td>@Model.Dishes</td>

</tr>

<tr>

<td>Причина вибору</td>

<td>@Model.Reason</td>

</tr>

<tr>

<td>Чи рекомендуватимете?</td>

<td>@Model.Recommend</td>

</tr>

</tbody>

</table>

</div>

}

@functions {

// Статична змінна для лічильника відвідувань цього подання

public static int VisitCounter = 0;

}

**VisitorViewModel.cs:**

using System;

using System.ComponentModel.DataAnnotations;

namespace Task\_5.Models

{

public class VisitorViewModel

{

[Display(Name = "Ім'я")]

public string? Name { get; set; }

[Display(Name = "Телефон")]

public string? Phone { get; set; }

[Display(Name = "Пошта")]

[EmailAddress]

public string? Email { get; set; }

[Display(Name = "Дата відвідування")]

[DataType(DataType.Date)]

public DateTime? VisitDate { get; set; }

[Display(Name = "Вік")]

public string? Age { get; set; }

[Display(Name = "Улюблена кухня")]

public string? Cuisine { get; set; }

[Display(Name = "Які страви ви б хотіли бачити в меню?")]

public string? Dishes { get; set; }

[Display(Name = "Чому обрали нашу установу?")]

public string? Reason { get; set; }

[Display(Name = "Чи будете рекомендувати нашу установу друзям та знайомим?")]

public string? Recommend { get; set; }

}

}

**HomeController.cs:**

using Microsoft.AspNetCore.Mvc;

using Task\_5.Models;

namespace Task\_5.Controllers

{

public class HomeController : Controller

{

// GET: /Home/Index - відображаємо порожню форму для заповнення

[HttpGet]

public IActionResult Index()

{

return View(new VisitorViewModel());

}

// POST: /Home/Index - обробляємо дані з форми і повертаємо їх у вигляді моделі

[HttpPost]

public IActionResult Index(VisitorViewModel model)

{

// Можна додати перевірку ModelState.IsValid, якщо потрібно

return View(model);

}

}

}

**site.css:**

/\* Фонове зображення для всієї сторінки \*/

body {

margin: 0;

padding: 0;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background: url("https://4kwallpapers.com/images/wallpapers/teton-range-rocky-mountains-wyoming-usa-mirror-lake-2560x1440-4274.jpg") no-repeat center center fixed;

background-size: cover;

}

/\* Стилізація контейнера для форми \*/

.form-container {

background-color: rgba(255, 255, 255, 0.85);

margin: 30px auto;

padding: 20px;

border-radius: 10px;

max-width: 700px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

}

/\* Стилізація заголовку форми \*/

.form-title {

text-align: center;

margin-bottom: 20px;

font-weight: bold;

font-size: 24px;

}

/\* Стилізація кнопок \*/

.btn-custom {

background-color: #4CAF50;

color: #fff;

border: none;

padding: 10px 20px;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.2s;

}

.btn-custom:hover {

background-color: #45a049;

}

.btn-reset {

background-color: #f44336 !important;

}

.btn-reset:hover {

background-color: #e53935 !important;

}

/\* Стилізація блоку для відображення введених даних \*/

.data-display {

background-color: rgba(255, 255, 255, 0.85);

margin: 20px auto;

padding: 15px;

border-radius: 10px;

max-width: 700px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

}

.data-display h4 {

text-align: center;

}

footer p {

color: gold;

}

/\* Користувацькі стилі для таблиці з чіткими чорними бордюрами \*/

.custom-bordered-table {

border-collapse: collapse;

width: 100%;

}

.custom-bordered-table th,

.custom-bordered-table td {

border: 2px solid #000 !important;

padding: 8px;

}

## **Результати роботи програми:**

Изображение выглядит как текст, вода, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, вода, снимок экрана, небо

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, вода, дерево

Контент, сгенерированный ИИ, может содержать ошибки.

## **Алгоритм побудови кодів**

1. **Створення основної структури:**
   * Використовується спільний шаблон (\_Layout.cshtml) для формування загального вигляду сторінки, включаючи хедер, футер та підключення стилів (Bootstrap, site.css).
2. **Розробка подання з формою:**
   * Подання (Index.cshtml) базується на сильно типізованій моделі (VisitorViewModel) для зручної роботи з даними.
   * За допомогою Razor-хелперів створено форму для введення контактної, персональної інформації та оцінки ресторану.
3. **Додавання лічильника відвідувань:**
   * Усередині подання оголошено статичну змінну (VisitCounter) за допомогою секції @functions.
   * Кожного разу при завантаженні сторінки лічильник збільшується через виклик VisitCounter++ та виводиться на сторінці.
4. **Обробка даних контролером:**
   * Контролер (HomeController.cs) містить метод GET, який повертає новий екземпляр моделі, та метод POST для обробки заповненої форми.
   * Логіка обробки даних здійснюється виключно у поданні за допомогою технології Razor.

## **Функціональність**

* **Відображення лічильника відвідувань:**

При кожному завантаженні сторінки статична змінна VisitCounter збільшується, що дозволяє відслідковувати кількість переглядів.

* **Заповнення та обробка форми:**

Користувач заповнює анкету з контактною, персональною інформацією та оцінкою ресторану. Після відправки даних, контролер повертає модель з введеними даними, яка відображається у вигляді таблиці.

* **Інтеграція логіки у поданні:**

Вся необхідна логіка (вирази, умови, цикли) реалізована безпосередньо у Razor, що спрощує підтримку та розвиток застосунку, а контролер використовується лише як посередник.

## **Висновок**

* Завдання виконано відповідно до вимог, модифікуючи попереднє рішення (Task\_4) додаванням лічильника відвідувань, що демонструє можливість використання логіки Razor у поданні.
* Використання статичної змінної у @functions дозволяє відслідковувати перегляди сторінки без залучення додаткових серверних ресурсів чи бази даних.
* Підхід із єдиним поданням і контролером сприяє простоті підтримки, підвищує зручність використання та демонструє гнучкість технології Razor для реалізації додаткової логіки без зміни основної архітектури застосунку.

# **Завдання 6**

Модифікувати минуле завдання , використовуючи лише один метод одного контролеру.

## **Текст програми:**

**Index.cshtml:**

@model Task\_6.Models.VisitorViewModel

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

ViewData["Title"] = "Анкета відвідувача ресторану";

// Збільшуємо лічильник відвідувань при кожному завантаженні сторінки

VisitCounter++;

}

<p class="text-center fw-bold">Кількість відвідувань цієї сторінки: @VisitCounter</p>

<div class="form-container">

<h2 class="form-title">Анкета відвідувача ресторану</h2>

@using (Html.BeginForm("Index", "Home", FormMethod.Post))

{

<!-- Таблиця для контактної інформації -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Контактна інформація</th>

</tr>

</thead>

<tbody>

<tr>

<td>@Html.LabelFor(m => m.Name)</td>

<td>@Html.TextBoxFor(m => m.Name, new { @class = "form-control" })</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.Phone)</td>

<td>@Html.TextBoxFor(m => m.Phone, new { @class = "form-control" })</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.Email)</td>

<td>@Html.TextBoxFor(m => m.Email, new { @class = "form-control", type = "email" })</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.VisitDate)</td>

<td>@Html.TextBoxFor(m => m.VisitDate, "{0:yyyy-MM-dd}", new { @class = "form-control", type = "date" })</td>

</tr>

</tbody>

</table>

<!-- Таблиця для персональної інформації -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Персональна інформація</th>

</tr>

</thead>

<tbody>

<tr>

<td>@Html.LabelFor(m => m.Age)</td>

<td>

@Html.DropDownListFor(m => m.Age, new SelectList(new[] {

new { Value = "", Text = "-- Оберіть категорію --" },

new { Value = "До 18", Text = "До 18" },

new { Value = "18-25", Text = "18-25" },

new { Value = "26-35", Text = "26-35" },

new { Value = "36-45", Text = "36-45" },

new { Value = "46 і старше", Text = "46 і старше" }

}, "Value", "Text"), new { @class = "form-select" })

</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.Cuisine)</td>

<td>

@Html.DropDownListFor(m => m.Cuisine, new SelectList(new[] {

new { Value = "", Text = "-- Оберіть кухню --" },

new { Value = "Грецька", Text = "Грецька" },

new { Value = "Італійська", Text = "Італійська" },

new { Value = "Французька", Text = "Французька" },

new { Value = "Японська", Text = "Японська" },

new { Value = "Українська", Text = "Українська" }

}, "Value", "Text"), new { @class = "form-select" })

</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.Dishes)</td>

<td>@Html.TextAreaFor(m => m.Dishes, new { @class = "form-control" })</td>

</tr>

</tbody>

</table>

<!-- Таблиця для оцінки ресторану -->

<table class="table table-bordered">

<thead>

<tr>

<th colspan="2" class="bg-light">Оцінка нашого ресторану</th>

</tr>

</thead>

<tbody>

<tr>

<td>@Html.LabelFor(m => m.Reason)</td>

<td style="white-space: nowrap;">

<span>

<input type="radio" name="Reason" value="Недалеко від роботи/дому" id="reason1"

@(Model.Reason == "Недалеко від роботи/дому" ? "checked" : "") />

<label for="reason1">Недалеко від роботи/дому</label>

</span>

<br />

<span>

<input type="radio" name="Reason" value="Побачив рекламу" id="reason2"

@(Model.Reason == "Побачив рекламу" ? "checked" : "") />

<label for="reason2">Побачив рекламу</label>

</span>

<br />

<span>

<input type="radio" name="Reason" value="Порадили" id="reason3"

@(Model.Reason == "Порадили" ? "checked" : "") />

<label for="reason3">Порадили</label>

</span>

<br />

<span>

<input type="radio" name="Reason" value="Оптимальне співвідношення ціни та якості" id="reason4"

@(Model.Reason == "Оптимальне співвідношення ціни та якості" ? "checked" : "") />

<label for="reason4">Оптимальне співвідношення ціни та якості</label>

</span>

</td>

</tr>

<tr>

<td>@Html.LabelFor(m => m.Recommend)</td>

<td>

<input type="radio" name="Recommend" value="Так" id="recYes" @(Model.Recommend == "Так" ? "checked" : "") />

<label for="recYes">Так</label>

<br />

<input type="radio" name="Recommend" value="Ні" id="recNo" @(Model.Recommend == "Ні" ? "checked" : "") />

<label for="recNo">Ні</label>

</td>

</tr>

</tbody>

</table>

<!-- Кнопки для відправки форми та її скидання -->

<div class="text-center mt-3">

<button type="submit" class="btn-custom me-3">Відіслати</button>

<button type="button" onclick="window.location.href='@Url.Action("Index", "Home")'" class="btn-custom btn-reset">Скинути</button>

</div>

}

</div>

@if (Model.Name != null)

{

<div class="data-display">

<h4>Введені дані:</h4>

<table class="table custom-bordered-table table-striped">

<thead>

<tr>

<th>Поле</th>

<th>Значення</th>

</tr>

</thead>

<tbody>

<tr>

<td>Ім’я</td>

<td>@Model.Name</td>

</tr>

<tr>

<td>Телефон</td>

<td>@Model.Phone</td>

</tr>

<tr>

<td>Пошта</td>

<td>@Model.Email</td>

</tr>

<tr>

<td>Дата відвідування</td>

<td>@(Model.VisitDate.HasValue ? Model.VisitDate.Value.ToString("yyyy-MM-dd") : "")</td>

</tr>

<tr>

<td>Вік</td>

<td>@Model.Age</td>

</tr>

<tr>

<td>Улюблена кухня</td>

<td>@Model.Cuisine</td>

</tr>

<tr>

<td>Бажані страви</td>

<td>@Model.Dishes</td>

</tr>

<tr>

<td>Причина вибору</td>

<td>@Model.Reason</td>

</tr>

<tr>

<td>Чи рекомендуватимете?</td>

<td>@Model.Recommend</td>

</tr>

</tbody>

</table>

</div>

}

@functions {

// Статична змінна для лічильника відвідувань цього подання

public static int VisitCounter = 0;

}

**VisitorViewModel.cs:**

using System;

using System.ComponentModel.DataAnnotations;

namespace Task\_6.Models

{

public class VisitorViewModel

{

[Display(Name = "Ім'я")]

public string? Name { get; set; }

[Display(Name = "Телефон")]

public string? Phone { get; set; }

[Display(Name = "Пошта")]

[EmailAddress]

public string? Email { get; set; }

[Display(Name = "Дата відвідування")]

[DataType(DataType.Date)]

public DateTime? VisitDate { get; set; }

[Display(Name = "Вік")]

public string? Age { get; set; }

[Display(Name = "Улюблена кухня")]

public string? Cuisine { get; set; }

[Display(Name = "Які страви ви б хотіли бачити в меню?")]

public string? Dishes { get; set; }

[Display(Name = "Чому обрали нашу установу?")]

public string? Reason { get; set; }

[Display(Name = "Чи будете рекомендувати нашу установу друзям та знайомим?")]

public string? Recommend { get; set; }

}

}

**HomeController.cs:**

using Microsoft.AspNetCore.Mvc;

using Task\_6.Models;

namespace Task\_6.Controllers

{

public class HomeController : Controller

{

// Один метод, що обробляє як GET, так і POST запити

[HttpGet, HttpPost]

public IActionResult Index(VisitorViewModel? model)

{

if (model == null)

{

model = new VisitorViewModel();

}

return View(model);

}

}

}

**\_Layout.cshtml:**

<!DOCTYPE html>

<html lang="uk">

<head>

<meta charset="utf-8" />

<title>@ViewData["Title"] - Task\_6</title>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" />

<link rel="stylesheet" href="~/css/site.css" />

</head>

<body>

<header class="p-3 bg-dark text-white">

<div class="container">

<h1 class="mb-0">Завдання 6 (Варіант №12)</h1>

</div>

</header>

<div class="container">

<main role="main" class="pb-3">

@RenderBody()

</main>

</div>

<footer class="border-top pt-3 mt-3">

<div class="container">

<p>&copy; 2025 - Task\_6</p>

</div>

</footer>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>

@RenderSection("Scripts", required: false)

</body>

</html>

**site.css:**

/\* Фонове зображення для всієї сторінки \*/

body {

margin: 0;

padding: 0;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background: url("https://4kwallpapers.com/images/wallpapers/teton-range-rocky-mountains-wyoming-usa-mirror-lake-2560x1440-4274.jpg") no-repeat center center fixed;

background-size: cover;

}

/\* Стилізація контейнера для форми \*/

.form-container {

background-color: rgba(255, 255, 255, 0.85);

margin: 30px auto;

padding: 20px;

border-radius: 10px;

max-width: 700px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

}

/\* Стилізація заголовку форми \*/

.form-title {

text-align: center;

margin-bottom: 20px;

font-weight: bold;

font-size: 24px;

}

/\* Стилізація кнопок \*/

.btn-custom {

background-color: #4CAF50;

color: #fff;

border: none;

padding: 10px 20px;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.2s;

}

.btn-custom:hover {

background-color: #45a049;

}

.btn-reset {

background-color: #f44336 !important;

}

.btn-reset:hover {

background-color: #e53935 !important;

}

/\* Стилізація блоку для відображення введених даних \*/

.data-display {

background-color: rgba(255, 255, 255, 0.85);

margin: 20px auto;

padding: 15px;

border-radius: 10px;

max-width: 700px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

}

.data-display h4 {

text-align: center;

}

footer p {

color: gold;

}

/\* Користувацькі стилі для таблиці з чіткими чорними бордюрами \*/

.custom-bordered-table {

border-collapse: collapse;

width: 100%;

}

.custom-bordered-table th,

.custom-bordered-table td {

border: 2px solid #000 !important;

padding: 8px;

}

## **Результати роботи програми:**

Изображение выглядит как текст, вода, снимок экрана, дерево

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, вода, облако, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, вода, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

## **Алгоритм побудови кодів**

1. **Створення базової структури:**
   * Використовується спільний шаблон (\_Layout.cshtml) для формування загального вигляду сторінки (хедер, футер, підключення стилів).
2. **Розробка подання (Index.cshtml):**
   * Подання базується на сильно типізованій моделі (VisitorViewModel) для роботи з даними.
   * За допомогою Razor-хелперів створено форму для введення контактної та персональної інформації, а також оцінки ресторану.
   * Реалізовано лічильник відвідувань за допомогою статичної змінної, оголошеної в секції @functions, який збільшується щоразу при завантаженні сторінки.
3. **Реалізація єдиного методу контролера:**
   * Контролер (HomeController.cs) містить один метод, який обробляє як GET, так і POST запити (за допомогою атрибутів [HttpGet, HttpPost]).
   * Метод перевіряє, чи передана модель є null, і у разі потреби створює новий екземпляр, після чого повертає подання з актуальними даними.

## **Функціональність**

* **Заповнення форми:**

Користувач вводить дані анкети (контактна інформація, персональна інформація, оцінка ресторану) через зручну форму, реалізовану з використанням Razor-хелперів.

* **Обробка даних:**

Єдиний метод контролера обробляє як GET, так і POST запити, що спрощує логіку обробки форми.Після відправки форми дані повертаються і відображаються у поданні.

* **Лічильник відвідувань:**

Статична змінна, оголошена в поданні, збільшується щоразу при завантаженні сторінки, відображаючи кількість відвідувань.

## **Висновок**

* Модифікація попереднього завдання (Task\_5) з використанням єдиного методу контролера дозволяє спростити код і зменшити кількість методів, що відповідають за обробку запитів.
* Вся логіка (включаючи лічильник відвідувань) реалізована безпосередньо в поданні за допомогою Razor, що відповідає вимогам і сприяє легкості підтримки застосунку.
* Такий підхід забезпечує ефективну інтеграцію форми, обробки даних і додаткової логіки, що підвищує зручність використання та розвиток застосунку в майбутньому.

# **Завдання 7**

Зробити простий (одне подання та один контролер) MVC-застосунок, який вирішує таку задачу: на поданні (Index) в окремому текстовому полі ввести рядок, що містить не менше п'яти слів, а в інше окреме поле кількість разів для повторення обробленого за власним варіантом (Варіант №12) рядка; нижче на цьому ж поданні має з'явитися вже оброблений рядок, повторений задану кількість разів.

|  |  |
| --- | --- |
| **№ варіанту** | **Функція обробки рядка** |
| 1 | 2 |
| 12 | Перевести другу половину рядка до верхнього регістру |

## **Текст програми:**

**Index.cshtml:**

<!-- Підключаємо модель даних -->

@model Task\_7.Models.StringProcessingViewModel

@{

Layout = "~/Views/Shared/\_Layout.cshtml"; // Використовуємо стандартний шаблон

ViewData["Title"] = "Обробка рядка"; // Встановлюємо заголовок сторінки

}

<div class="container mt-4">

<div class="form-container p-4 shadow-sm rounded">

<h2 class="text-center mb-4">Обробка рядка</h2>

@using (Html.BeginForm("Index", "Home", FormMethod.Post))

{

<div class="mb-3">

<!-- Лейбл для текстового поля -->

@Html.LabelFor(m => m.InputText, new { @class = "form-label" })

@Html.TextBoxFor(

m => m.InputText,

new

{

@class = "form-control",

placeholder = "Введіть рядок (не менше 5 слів)",

value = Model.InputText ?? "" // Якщо InputText == null, показуємо порожній рядок

}

)

</div>

<div class="mb-3">

<!-- Лейбл для числа повторень -->

@Html.LabelFor(m => m.RepeatCount, new { @class = "form-label" })

@Html.TextBoxFor(

m => m.RepeatCount,

new

{

@class = "form-control",

type = "number",

placeholder = "Введіть кількість повторень",

value = Model.RepeatCount.HasValue ? Model.RepeatCount.Value.ToString() : ""

// Якщо RepeatCount == null, робимо поле порожнім

}

)

</div>

<div class="text-center">

<!-- Кнопка для надсилання даних -->

<button type="submit" class="btn btn-primary" name="action" value="Process">Обробити</button>

<!-- Кнопка для скидання форми -->

<button type="submit" class="btn btn-secondary ms-2" name="action" value="Reset">Скинути</button>

</div>

}

</div>

@\* Якщо користувач щось ввів, починаємо обробку \*@

@if (!string.IsNullOrEmpty(Model.InputText) && Model.RepeatCount.HasValue)

{

// Розбиваємо рядок на слова, щоб перевірити кількість

var words = Model.InputText.Split(new char[] { ' ' }, System.StringSplitOptions.RemoveEmptyEntries);

if (words.Length < 5)

{

<div class="alert alert-danger mt-4" role="alert">

Рядок має містити не менше 5 слів. Будь ласка, введіть коректний рядок.

</div>

}

else

{

int len = Model.InputText.Length; // Отримуємо довжину всього рядка

int mid = len / 2; // Знаходимо середину рядка

// Формуємо новий рядок: перша половина як є, друга - у верхньому регістрі

string processed = Model.InputText.Substring(0, mid) + Model.InputText.Substring(mid).ToUpper();

<div class="result mt-4 p-3 border rounded bg-light">

<h4 class="text-center">Оброблений рядок (друга половина у верхньому регістрі)</h4>

<!-- Виводимо оброблений рядок задану кількість разів -->

@for (int i = 0; i < Model.RepeatCount; i++)

{

<p>@processed</p>

// Повторюємо рядок

}

</div>

}

}

</div>

**StringProcessingViewModel.cs:**

using System.ComponentModel.DataAnnotations;

namespace Task\_7.Models

{

public class StringProcessingViewModel

{

[Display(Name = "Введіть рядок (не менше 5 слів)")]

public string? InputText { get; set; } // Текст, який вводить користувач

[Display(Name = "Кількість повторень")]

public int? RepeatCount { get; set; } // Кількість повторень виводу рядка

}

}

**HomeController.cs:**

using Microsoft.AspNetCore.Mvc;

using Task\_7.Models;

namespace Task\_7.Controllers

{

public class HomeController : Controller

{

// Метод обробляє як GET, так і POST запити

[HttpGet, HttpPost]

public IActionResult Index(StringProcessingViewModel? model)

{

// Якщо модель пуста, створюємо новий екземпляр

if (model == null)

{

model = new StringProcessingViewModel();

}

// Якщо запит POST і натиснута кнопка "Скинути"

if (Request.Method == "POST" && Request.Form["action"] == "Reset")

{

// Очищаємо ModelState, щоб скинути всі дані

ModelState.Clear();

// Повертаємо нову порожню модель

model = new StringProcessingViewModel();

return View(model);

}

// Відправляємо модель у View

return View(model);

}

}

}

**\_Layout.cshtml:**

<!DOCTYPE html>

<html lang="uk">

<head>

<meta charset="utf-8" /> <!-- Встановлюємо кодування -->

<title>@ViewData["Title"] - Task\_7</title> <!-- Динамічний заголовок сторінки -->

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" /> <!-- Підключення Bootstrap -->

<link rel="stylesheet" href="~/css/site.css" /> <!-- Наш кастомний CSS -->

</head>

<body class="d-flex flex-column min-vh-100">

<header class="py-3 mb-4 border-bottom">

<div class="container d-flex flex-wrap justify-content-center">

<h1 class="fw-bold">Task 7 - Обробка рядка</h1> <!-- Назва сайту -->

</div>

</header>

<div class="container flex-fill">

@RenderBody() <!-- Тут буде контент з інших сторінок -->

</div>

<footer class="py-3 mt-auto border-top">

<div class="container text-center">

<p>&copy; 2025 - Task\_7</p> <!-- Футер сайту -->

</div>

</footer>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script> <!-- Підключення Bootstrap JS -->

@RenderSection("Scripts", required: false) <!-- Додаткові скрипти, якщо є -->

</body>

</html>

**site.css:**

/\* Загальні стилі для сайту \*/

body {

margin: 0;

padding: 0;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background: url("https://i.imgur.com/WzY59S2.jpg") no-repeat center center fixed;

background-size: cover;

}

.container {

margin-top: 20px;

}

/\* Стилізація форми \*/

.form-container {

background-color: rgba(255, 255, 255, 0.85);

padding: 20px;

border-radius: 8px;

box-shadow: 0 2px 8px rgba(0, 0, 0, 0.3);

}

/\* Стилізація блоку результату \*/

.result {

background-color: rgba(255, 255, 255, 0.9);

padding: 15px;

border-radius: 5px;

box-shadow: 0 2px 8px rgba(0, 0, 0, 0.2);

}

## **Результати роботи програми:**

Изображение выглядит как текст, вода, снимок экрана, озеро

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, вода, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, вода, снимок экрана, озеро

Контент, сгенерированный ИИ, может содержать ошибки.

## **Алгоритм побудови кодів**

1. **Створення моделі даних:**
   * Визначено клас **StringProcessingViewModel** із двома властивостями:
     + **InputText** – рядок, який має містити не менше 5 слів.
     + **RepeatCount** – кількість повторень для виводу обробленого рядка.
2. **Розробка подання (Index.cshtml):**
   * Подання використовує сильну типізацію для роботи з моделлю.
   * За допомогою Razor-хелперів створено форму з двома текстовими полями:
     + Поле для введення рядка.
     + Поле для введення кількості повторень.
   * Додано логіку перевірки – розбиття рядка на слова і виведення повідомлення про помилку, якщо їх менше 5.
   * Реалізовано функцію обробки рядка: друга половина рядка переводиться до верхнього регістру, а отриманий результат повторюється задану кількість разів.
3. **Реалізація контролера:**
   * Використано один контролер (**HomeController**) з одним методом, який обробляє як GET, так і POST запити.
   * Метод приймає модель через Model Binding. Якщо запит POST і натиснуто кнопку "Reset", дані скидаються.
   * Контролер виступає лише як посередник, передаючи модель у подання для подальшої обробки за допомогою Razor.

## **Функціональність:**

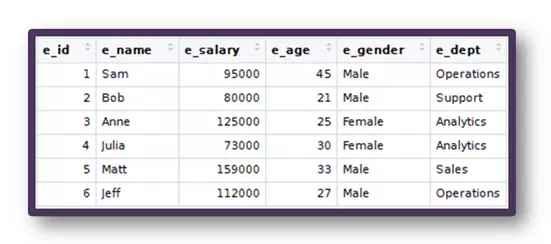
* Користувач вводить рядок (мінімум 5 слів) та кількість повторень у відповідні поля форми.
* Після відправки форми застосунок перевіряє коректність введених даних.
* Якщо дані валідні, у поданні за допомогою Razor виконується обробка: друга половина рядка переводиться у верхній регістр, і оброблений рядок повторюється задану кількість разів.
* Функціонал скидання форми дозволяє очистити введені дані.

## **Висновок:**

* Завдання демонструє ефективне використання технології Razor для реалізації всієї логіки без додаткових методів контролера.
* Єдиний метод контролера спрощує обробку запитів і відповідає принципам MVC, де контролер виступає лише посередником.
* Такий підхід забезпечує інтегроване відображення форми, валідацію даних та динамічну обробку рядка, що підвищує зручність використання застосунку та полегшує його підтримку.

# **Завдання 8**

В робочій базі даних (в БД в Visual Studio) створити таблицю за наступним зразком, вказаним на прикріпленій картинці.



Кількість записів можна збільшувати, дані в полях змінювати за потребою. Зробити МѴС-застосунок, який вирішує таку задачу: на основному поданні (Index) вміст усієї таблиці та форма з умовою згідно до власного варіанту до відображеної таблиці; друге (або це ж але трохи нижче) подання має відобразити результат виконання умови щодо таблиці.

|  |  |
| --- | --- |
| **№ варіанту** | **Приблизна форма запиту** |
| 12 | Парні або непарні ідентифікаційні номери |

Зауваження: контролер завжди вибирає увесь вміст таблиці, а Razor poбить фільтрацію даних таблиці на поданні.

## **Текст програми:**

**SQL-запит:**

CREATE TABLE [dbo].[Employees]

(

[e\_id] INT NOT NULL PRIMARY KEY IDENTITY(1,1),

[e\_name] NVARCHAR(100) NOT NULL,

[e\_salary] DECIMAL(18, 2) NOT NULL,

[e\_age] INT NOT NULL,

[e\_gender] NVARCHAR(10) NOT NULL,

[e\_dept] NVARCHAR(50) NOT NULL

);

-- Приклад вставки кількох записів

INSERT INTO [dbo].[Employees] (e\_name, e\_salary, e\_age, e\_gender, e\_dept)

VALUES

('Sam', 95000, 45, 'Male', 'Operations'),

('Bob', 80000, 21, 'Male', 'Support'),

('Anne', 125000, 25, 'Female', 'Analytics'),

('Julia', 73000, 30, 'Female', 'Analytics'),

('Matt', 159000, 33, 'Male', 'Sales'),

('Jeff', 112000, 27, 'Male', 'Operations');

**Index.cshtml:**

@model List<Task\_8.Models.Employee>

@inject Microsoft.AspNetCore.Http.IHttpContextAccessor HttpContextAccessor

@{

// Вказуємо розташування макета

Layout = "~/Views/Shared/\_Layout.cshtml";

// Заголовок сторінки

ViewData["Title"] = "Employees Table";

// Значення за замовчуванням для фільтра

string evenOddChoice = "All";

// Якщо форма відправлена методом POST, беремо значення фільтра з форми

if (HttpContextAccessor.HttpContext?.Request.Method == "POST" &&

HttpContextAccessor.HttpContext?.Request.HasFormContentType == true)

{

evenOddChoice = HttpContextAccessor.HttpContext.Request.Form["EvenOdd"];

if (string.IsNullOrEmpty(evenOddChoice))

{

evenOddChoice = "All";

}

}

}

<div class="container my-4">

<!-- Заголовок списку працівників -->

<h2 class="text-center mb-4 text-white">Список працівників</h2>

<!-- Форма для вибору фільтра по ідентифікаторах -->

<form method="post" class="mb-4">

<div class="mb-3">

<!-- Лейбл для вибору -->

<label class="form-label text-white">Виберіть ідентифікатори:</label>

<div>

<!-- Радіо кнопка для всіх працівників -->

<input type="radio" name="EvenOdd" value="All" id="all"

@(evenOddChoice == "All" ? "checked" : "") />

<label for="all" class="text-white">Всі</label>

<!-- Радіо кнопка для парних ID -->

<input type="radio" name="EvenOdd" value="Even" id="even"

@(evenOddChoice == "Even" ? "checked" : "") />

<label for="even" class="text-white">Парні</label>

<!-- Радіо кнопка для непарних ID -->

<input type="radio" name="EvenOdd" value="Odd" id="odd"

@(evenOddChoice == "Odd" ? "checked" : "") />

<label for="odd" class="text-white">Непарні</label>

</div>

</div>

<!-- Кнопка для застосування фільтра -->

<button type="submit" class="btn btn-primary">Застосувати</button>

</form>

<!-- Таблиця для виводу списку працівників -->

<table class="table table-bordered table-striped custom-table">

<thead class="table-light">

<tr>

<th>e\_id</th>

<th>e\_name</th>

<th>e\_salary</th>

<th>e\_age</th>

<th>e\_gender</th>

<th>e\_dept</th>

</tr>

</thead>

<tbody>

@foreach (var emp in Model)

{

// За замовчуванням показуємо рядок

bool showRow = true;

// Якщо обрано "Парні", показуємо лише ті рядки, де e\_id парний

if (evenOddChoice == "Even")

{

showRow = (emp.e\_id % 2 == 0);

}

// Якщо обрано "Непарні", показуємо лише ті рядки, де e\_id непарний

else if (evenOddChoice == "Odd")

{

showRow = (emp.e\_id % 2 != 0);

}

// Виводимо рядок, якщо умова виконується

if (showRow)

{

<tr>

<td>@emp.e\_id</td>

<td>@emp.e\_name</td>

<td>@emp.e\_salary</td>

<td>@emp.e\_age</td>

<td>@emp.e\_gender</td>

<td>@emp.e\_dept</td>

</tr>

}

}

</tbody>

</table>

</div>

**Models/Employee.cs:**

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Task\_8.Models

{

[Table("Employees")] // Вказуємо, що ця модель відповідає таблиці "Employees" в БД

public class Employee

{

[Key] // Це первинний ключ

public int e\_id { get; set; }

[Required] // Обов'язкове поле, не може бути пустим

public string e\_name { get; set; } = string.Empty;

// Зарплата співробітника

public decimal e\_salary { get; set; }

// Вік співробітника

public int e\_age { get; set; }

// Стать співробітника

public string e\_gender { get; set; } = string.Empty;

// Назва відділу, де працює співробітник

public string e\_dept { get; set; } = string.Empty;

}

}

**Data/AppDbContext.cs:**

using Microsoft.EntityFrameworkCore;

using Task\_8.Models;

namespace Task\_8.Data

{

public class AppDbContext : DbContext

{

// Конструктор, що приймає параметри для налаштування DbContext

public AppDbContext(DbContextOptions<AppDbContext> options)

: base(options)

{

}

// DbSet для таблиці Employees

public DbSet<Employee> Employees { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

// Вказуємо точний SQL-тип для зарплати, щоб значення не обрізались

modelBuilder.Entity<Employee>()

.Property(e => e.e\_salary)

.HasColumnType("decimal(18,2)");

}

}

}

**HomeController.cs:**

using Microsoft.AspNetCore.Mvc;

using Task\_8.Data;

using Task\_8.Models;

namespace Task\_8.Controllers

{

public class HomeController : Controller

{

// DbContext для доступу до бази даних

private readonly AppDbContext \_db;

// Конструктор, який отримує DbContext через dependency injection

public HomeController(AppDbContext db)

{

\_db = db; // зберігаємо контекст для використання в методах контролера

}

// GET: /Home/Index

[HttpGet]

public IActionResult Index()

{

// Завантажуємо всіх співробітників з бази даних

var employees = \_db.Employees.ToList();

return View(employees);

}

// POST: /Home/Index (фільтрація відбувається у поданні)

[HttpPost]

public IActionResult Index(IFormCollection form)

{

// Знову завантажуємо всіх співробітників (фільтрація вже робиться у View)

var employees = \_db.Employees.ToList();

return View(employees);

}

}

}

**\_Layout.cshtml:**

<!DOCTYPE html>

<html lang="uk">

<head>

<meta charset="utf-8" />

<title>@ViewData["Title"] - Task\_8</title>

<!-- Підключаємо Bootstrap для стилів -->

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" />

<!-- Підключаємо наш кастомний CSS -->

<link rel="stylesheet" href="~/css/site.css" />

</head>

<body class="d-flex flex-column min-vh-100">

<!-- Шапка сайту -->

<header class="p-3 bg-dark text-white">

<div class="container">

<h1 class="mb-0">Завдання 8 (Варіант №12)</h1>

</div>

</header>

<!-- Основна частина сторінки -->

<div class="container flex-fill">

<main role="main" class="pb-3">

@RenderBody() <!-- Тут буде основний контент, який підставляється -->

</main>

</div>

<!-- Нижня частина (футер) сайту -->

<footer class="border-top pt-3 mt-auto text-white">

<div class="container text-center">

<p>&copy; 2025 - Task\_8</p>

</div>

</footer>

<!-- Підключаємо JS для Bootstrap -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>

@RenderSection("Scripts", required: false) <!-- Рендер додаткових скриптів, якщо є -->

</body>

</html>

**site.css:**

/\* Фон для всієї сторінки \*/

body {

margin: 0;

padding: 0;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background: url("https://c.wallhere.com/photos/5d/9a/landscape\_nature-285967.jpg!d") no-repeat center center fixed;

background-size: cover;

}

.container {

margin-top: 20px;

}

/\* Стилізація таблиці \*/

.custom-table {

background-color: rgba(255, 255, 255, 0.85);

}

/\* Забезпечуємо, що футер завжди залишається внизу \*/

html, body {

height: 100%;

}

.flex-fill {

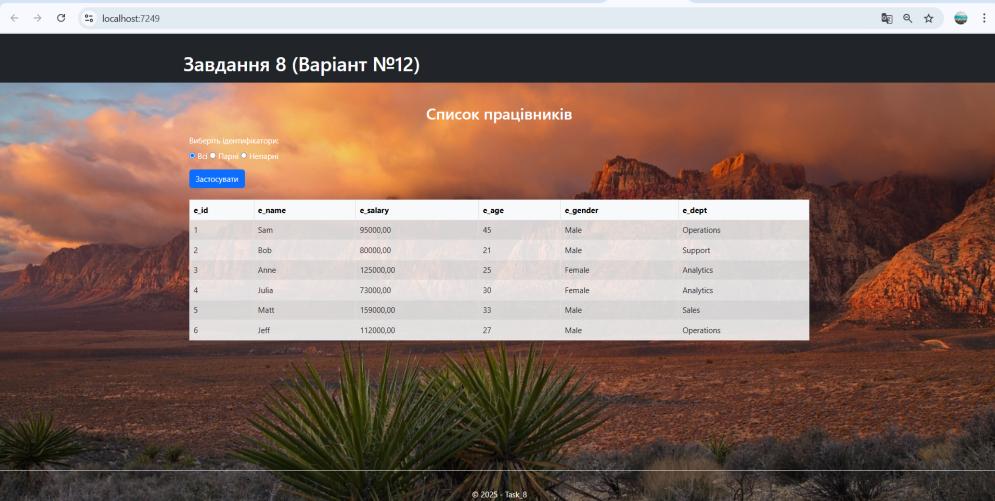
flex: 1 1 auto;

}

## **Результати роботи програми:**

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.



Изображение выглядит как текст, снимок экрана, растение, облако

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, растение, облако

Контент, сгенерированный ИИ, может содержать ошибки.

## **Алгоритм побудови кодів**

1. **Створення структури бази даних:**
   * У **SQL-запиті** створено таблицю Employees із полями e\_id, e\_name, e\_salary, e\_age, e\_gender, e\_dept.
   * Здійснено початкове наповнення таблиці демонстраційними записами.
2. **Налаштування моделі та DbContext:**
   * Створено клас **Employee** (у папці *Models*), який відображає поля таблиці Employees.
   * У **AppDbContext** (папка *Data*) додано DbSet<Employee> для доступу до записів у таблиці.
3. **Реалізація контролера (HomeController.cs):**
   * Є два методи Index (GET і POST), що завантажують усі дані з таблиці (через DbContext) та повертають їх у подання.
   * Фільтрація (парні/непарні ідентифікатори) відбувається безпосередньо у поданні за допомогою Razor.
4. **Розробка подання (Index.cshtml):**
   * Показує форму з радіокнопками («Всі», «Парні», «Непарні» ідентифікатори).
   * Після відправки форми методом POST обробляється логіка відбору (парні/непарні) у самому поданні:
     + Якщо обрано «Парні», відображаються записи з e\_id % 2 == 0.
     + Якщо «Непарні», відображаються записи з e\_id % 2 != 0.
     + Якщо «Всі», відображаються усі записи.
   * Таблиця виводить результати відповідно до фільтра.
5. **Стилізація та макет (site.css та \_Layout.cshtml):**
   * Підключено Bootstrap та власні CSS-стилі (фон, таблиці, контейнери).
   * Макет забезпечує єдину структуру сторінки з хедером, футером та основним контейнером.

## **Функціональність**

* **Відображення даних:**

Користувач бачить усі записи з таблиці Employees у вигляді таблиці.

* **Фільтрація за ідентифікаторами:**

Можна обрати відображення лише працівників із парними, непарними або всіма e\_id. Логіка фільтрації реалізована Razor-умовами безпосередньо у поданні.

* **Гнучке розширення:**

Код легко розширити, додаючи нові поля або фільтри, оскільки фільтрація здійснюється на рівні представлення, а контролер просто передає повний набір даних.

## **Висновок**

* Завдання реалізовано з використанням MVC-підходу: контролер отримує дані з бази, а вся логіка відбору (парні/непарні ідентифікатори) виконується у поданні за допомогою Razor.
* Такий підхід відповідає вимозі мінімізувати логіку у контролері, зберігаючи простоту розширення застосунку.
* Успішно продемонстровано, як можна інтегрувати базу даних, відобразити усі записи таблиці, а також застосувати фільтри, не змінюючи контролер.

# **Загальний Висновок**

Лабораторна робота №4 стала чудовою практичною реалізацією можливостей технології Razor для створення сучасних MVC-застосунків. У процесі виконання завдань було опрацьовано весь спектр елементів – від побудови простих форм із збору даних до впровадження динамічних елементів, таких як топологія зображень, табличне відображення результатів, перехід між різними станами подання та інтеграція лічильників відвідувань.

Використання Razor-хелперів, сильно типізованих моделей та засобів Bootstrap забезпечило:

* чітке розділення логіки обробки даних і їх відображення,
* спрощення підтримки та подальшого розширення застосунку,
* створення привабливого та зручного інтерфейсу для користувача.

Отриманий досвід дозволяє краще зрозуміти принципи роботи MVC-архітектури та переваги використання Razor у формуванні динамічного вмісту.

Загалом, виконання даної лабораторної роботи підтвердило ефективність застосування технології Razor для розробки складних і масштабованих веб-застосунків, сприяючи підвищенню якості програмного коду та зручності користувацького інтерфейсу.