Міністерство освіти України

Національний технічний університет "ХПІ"

кафедра "Комп’ютерних наук та інформаційних технологій"

**Звіт**

**Лабораторна робота 3**

з дисципліни "Проєктування серверних застосунків"

Виконав: студент групи КН-922В

Мельник Б. В.

Перевірив:

Івашко А. В.

Харків 2025

**Зміст**

[Завдання 1 4](#_Toc193287519)

[Текст програми 5](#_Toc193287520)

[Результати роботи програми 14](#_Toc193287521)

[Алгоритм побудови кодів 15](#_Toc193287522)

[Функціональність 16](#_Toc193287523)

[Висновок 16](#_Toc193287524)

[Завдання 2 18](#_Toc193287525)

[Текст програми 18](#_Toc193287526)

[Результат роботи програми 27](#_Toc193287527)

[Алгоритм побудови кодів 28](#_Toc193287528)

[Функціональність 29](#_Toc193287529)

[Висновок 29](#_Toc193287530)

[Завдання 3 30](#_Toc193287531)

[Текст програми 30](#_Toc193287532)

[Результати роботи програми 39](#_Toc193287533)

[Алгоритм побудови кодів 40](#_Toc193287534)

[Функціональність 41](#_Toc193287535)

[Висновок 41](#_Toc193287536)

[\*Завдання 4 42](#_Toc193287537)

[Текст програми 42](#_Toc193287538)

[Результати роботи програми 52](#_Toc193287539)

[Алгоритм побудови кодів 53](#_Toc193287540)

[Функціональність 54](#_Toc193287541)

[Висновок 54](#_Toc193287542)

[Завдання 5 55](#_Toc193287543)

[Текст програми 55](#_Toc193287544)

[Результати роботи програми 64](#_Toc193287545)

[Алгоритм побудови кодів 65](#_Toc193287546)

[Функціональність 66](#_Toc193287547)

[Висновок 66](#_Toc193287548)

[Загальний Висновок 67](#_Toc193287549)

**Тема:** Шляхи використання можливостей форм у засобах ASP.NET MVC WEB-застосунків. Частина 2.

**Мета роботи:** набути практику та досвіт у використанні різноманітних шля-хів отримання даних із форм на поданнях під час розробки та використання MVC WEB-застосунків.

**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**

Створити застосунок, стартове подання (Index) якого містить форму згідно до власного варіанту. За допомогою стандартного класу Request контролер, викликаний кнопкою Відіслати приймає дані з форми та пе-редає їх назад до подання для їхньої демонстрації під формою, кнопка Скинути має повертати форму у «дефолтний» (початковий) стан, тобто очистити текстові поля.

Варіант №12:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

## **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.

Изображение выглядит как текст, облако, небо, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

## **Алгоритм побудови кодів**

1. **Створення проєкту та налаштування Layout**: Спочатку в середовищі ASP.NET Core MVC створюється новий проєкт. У файлі \_Layout.cshtml задається основна структура сторінки (header, footer, підключення стилів Bootstrap та власних CSS).
2. **Підготовка сторінки з формою (Index.cshtml)**:
   * Додається HTML-форма, яка відправляє дані методом POST.
   * У формі використовуються стандартні елементи керування (input, select, textarea, radio), розділені на тематичні таблиці для зручності.
   * Кнопка «Відіслати» надсилає дані на метод контролера (через form method="post"), а кнопка «Скинути» оновлює сторінку, повертаючи поля до початкового стану.
3. **Налаштування контролера (HomeController.cs)**:
   * Метод Index() (з атрибутом [HttpGet]) просто відображає форму при першому завантаженні сторінки.
   * Метод IndexPost() (з атрибутом [HttpPost]) отримує надіслані дані через Request.Form[...], зберігає їх у ViewBag, і повертає те саме представлення, але вже з відображенням зібраної інформації.
4. **Підключення стилів (site.css)**:
   * Налаштовується загальний фон сторінки.
   * Стилізується контейнер форми, таблиці, кнопки та блок виведення даних.
   * Забезпечується привабливий та зручний для читання інтерфейс.

## **Функціональність**

* При завантаженні сторінки (HTTP GET) користувач бачить форму для введення контактних та персональних даних.
* Після натискання «Відіслати» (HTTP POST) дані зчитуються контролером і передаються назад на ту ж сторінку для візуального підтвердження під формою.
* Кнопка «Скинути» перезавантажує сторінку, видаляючи введену інформацію з усіх полів.
* Таким чином, реалізовано базову взаємодію «користувач → форма → контролер → представлення», що ілюструє принципи ASP.NET Core MVC (Model-View-Controller).

## **Висновок**

Даний застосунок демонструє основні механізми побудови вебформи з використанням ASP.NET Core MVC: структурування коду в окремих представленнях, передавання даних за допомогою POST-запитів і відображення введеної інформації. Завдяки простій архітектурі та використанню ViewBag можна легко розширювати функціонал, наприклад, додавати валідацію чи роботу з базою даних. Це базовий, але показовий приклад взаємодії користувача з формою, де всі дані обробляються в межах одного контролера та одного представлення.

# **Завдання 2**

Виконати функції минулого завдання, використавши для отримання да-них форми замість класу Request колекцію типу IFormCollection.

## **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>

<!-- 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" здійснюється перехід на GET-версію, що очищує форму та результати -->

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

</div>

</form>

</div>

<!-- Якщо форма відправлена, тут відображаються введені дані -->

@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;

using Microsoft.AspNetCore.Http;

namespace Task\_2.Controllers

{

public class HomeController : Controller

{

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

[HttpGet]

public IActionResult Index()

{

// Повертаємо вигляд для відображення форми

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;

// Повертаємо той самий вигляд з даними, що були введені

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;

}

/\* Стиль кнопки "Скинути" з червоним фоном \*/

.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;

}

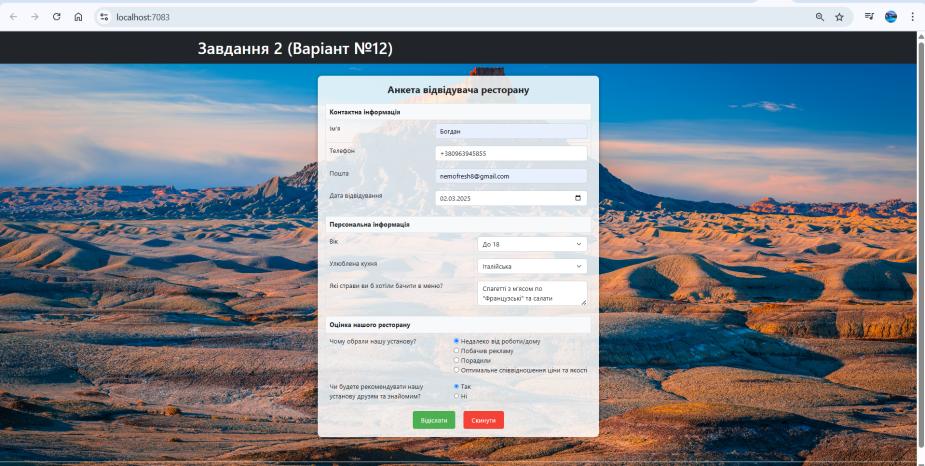
/\* Колір тексту у футері \*/

footer p {

color: gold;

}

## **Результат роботи програми:**



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

Контент, сгенерированный ИИ, может содержать ошибки.

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

Контент, сгенерированный ИИ, может содержать ошибки.

## **Алгоритм побудови кодів**

1. **Створення ASP.NET Core MVC проєкту**: Спочатку створюється базовий проєкт, у якому налаштовується структура контролерів, представлень та загальний Layout.
2. **Робота з формою (Index.cshtml)**:
   * На сторінці Index.cshtml формується HTML-форма з потрібними полями (текстові поля, дата, селекти, радіокнопки тощо).
   * Використовується метод POST, щоб надіслати дані до HomeController.
   * Кнопка «Скинути» викликає GET-версію методу Index, яка просто повертає порожню форму.
3. **Заміна Request на IFormCollection**:
   * У контролері (HomeController) оголошується метод Index(IFormCollection form) з атрибутом [HttpPost].
   * Замість Request.Form[...] тепер усі дані зчитуються з колекції form["Name"], form["Phone"] тощо.
4. **Відображення даних**:
   * Отримані значення зберігаються у ViewBag і повертаються назад у те саме представлення Index.cshtml, де під формою виводяться всі введені користувачем дані.

## **Функціональність**

* При завантаженні сторінки (HTTP GET) користувач бачить форму для заповнення.
* Після натискання «Відіслати» (HTTP POST) контролер зчитує дані з об’єкта IFormCollection, зберігає їх у ViewBag і повертає сторінку з відображенням введеної інформації.
* Якщо натиснути «Скинути», сторінка оновиться і поля форми буде очищено (GET-запит).
* Таким чином, реалізовано приклад обробки форм у MVC, але тепер без використання Request, а з використанням типу IFormCollection.

## **Висновок**

У цій задачі показано альтернативний спосіб доступу до даних форми в ASP.NET Core MVC. Заміна Request.Form[...] на IFormCollection робить код більш гнучким і зручним у подальшому розширенні (наприклад, при використанні модельних класів та механізмів валідації). Принцип роботи залишається аналогічним до Task\_1: заповнення форми, передавання даних до контролера та відображення результатів на тій самій сторінці.

# **Завдання 3**

Виконати функції першого завдання, використавши для отримання да-них форми замість класу Request слабку типізацію форми, додавши до кожного елементу форми відповідну змінну.

## **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>

<!-- Поле для введення імені -->

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

</tr>

<tr>

<td>Телефон</td>

<!-- Поле для введення телефону -->

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

</tr>

<tr>

<td>Пошта</td>

<!-- Поле для введення email -->

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

</tr>

<tr>

<td>Дата відвідування</td>

<!-- Поле для вибору дати -->

<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>

<!-- Селектор вікової категорії -->

<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>

<!-- Селектор для вибору кухні -->

<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>

<!-- Місце для введення побажань щодо меню -->

<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>

<!-- 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>

<!-- Радіокнопки для вибору відповіді -->

<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>

<!-- Кнопка скидання: переходить на GET-версію, щоб очистити форму -->

<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 вже заповнив 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\_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

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;

}

/\* Ефект при наведенні на кнопку \*/

.btn-custom:hover {

background-color: #45a049;

}

/\* Кнопка скидання з яскравим червоним кольором \*/

.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;

}

/\* Золотистий колір тексту для футера \*/

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, небо, снимок экрана

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 ASP.NET Core MVC застосунку**: У межах проєкту налаштовується стандартна структура (контролер, представлення, Layout, файли стилів).
2. **Налаштування форми (Index.cshtml)**:
   * Розміщується HTML-форма з полями (текстові поля, дата, селект, радіокнопки тощо).
   * Використовується метод POST для відправки даних у контролер.
   * Кнопка «Скинути» повертає сторінку у вихідний стан (HTTP GET).
3. **Слабка типізація (HomeController)**:
   * У методі Index з атрибутом [HttpPost] вказуються параметри, назви яких збігаються з атрибутами name у формі (Name, Phone, Email тощо).
   * ASP.NET Core автоматично заповнює ці параметри значеннями з полів форми.
   * Далі отримані дані зберігаються у ViewBag для відображення на тій самій сторінці.
4. **Відображення результатів**:
   * Якщо дані вже надіслані, у представленні під формою формується список введених користувачем значень (через ViewBag).

## **Функціональність**

* На початку (HTTP GET) користувач бачить чисту форму.
* При натисканні «Відіслати» (HTTP POST) контролер приймає дані як окремі параметри.
* Отримані значення повертаються в те саме представлення, де відображаються під формою.
* Кнопка «Скинути» знову викликає GET-версію сторінки, очищуючи всі поля форми та результати.

## **Висновок**

Даний підхід ілюструє спосіб зчитування даних форми за допомогою «слабкої типізації» — коли параметри методу контролера напряму відповідають іменам полів форми. Це зручно для простих проєктів і швидкої розробки, але в більших системах зазвичай застосовують модельну прив’язку з використанням strongly-typed моделей. У будь-якому разі принцип залишається тим самим: збір даних з форми, обробка та повернення для відображення.

# **\*Завдання 4**

Виконати функції першого завдання, використавши для отримання даних форми замість класу Request сувору типізацію форми, додавши до елементів форми відповідну модель.

## **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>

<!-- Поле для введення імені -->

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

</tr>

<tr>

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

<!-- Поле для введення телефону -->

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

</tr>

<tr>

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

<!-- Поле для введення email (тип - email) -->

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

</tr>

<tr>

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

<!-- Поле для введення дати (формат дати) -->

<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>

<!-- Селектор для вибору вікової категорії -->

@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>

<!-- Селектор для вибору улюбленої кухні -->

@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>

<!-- Текстове поле для побажань щодо страв -->

<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>

<!-- Радіокнопки для вибору причини -->

<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>

<!-- Радіокнопки для вибору, чи рекомендуватимуть ресторан -->

<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>

<!-- Кнопка "Скинути": переходить до GET-версії, щоб очистити форму -->

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

</div>

}

</div>

<!-- Якщо ім'я не порожнє, відображаємо введені дані -->

@if (Model.Name != null)

{

<div class="data-display">

<h4>Введені дані:</h4>

<ul>

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

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

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

<li><strong>Дата відвідування:</strong> @(Model.VisitDate.HasValue ? Model.VisitDate.Value.ToString("yyyy-MM-dd") : "")</li>

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

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

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

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

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

</ul>

</div>

}

**HomeController.cs:**

using Microsoft.AspNetCore.Mvc;

using Task\_4.Models;

namespace Task\_4.Controllers

{

public class HomeController : Controller

{

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

[HttpGet]

public IActionResult Index()

{

// Створюємо новий екземпляр моделі і передаємо її в подання

return View(new VisitorViewModel());

}

// POST: /Home/Index - обробляємо дані, надіслані з форми

[HttpPost]

public IActionResult Index(VisitorViewModel model)

{

// Model binding автоматично зв'язує дані форми з моделлю

// За бажанням можна перевірити ModelState.IsValid

return View(model);

}

}

}

**VisitorViewModel.cs:**

using System;

using System.ComponentModel.DataAnnotations;

namespace Task\_4.Models

{

public class VisitorViewModel

{

// Властивість для імені відвідувача

[Display(Name = "Ім'я")]

public string? Name { get; set; }

// Властивість для номеру телефону

[Display(Name = "Телефон")]

public string? Phone { get; set; }

// Властивість для електронної пошти з валідацією формату

[Display(Name = "Пошта")]

[EmailAddress]

public string? Email { get; set; }

// Властивість для дати відвідування (тип DateTime)

[Display(Name = "Дата відвідування")]

[DataType(DataType.Date)]

public DateTime? VisitDate { get; set; }

// Властивість для вибору віку

[Display(Name = "Вік")]

public string? Age { get; set; }

// Властивість для улюбленої кухні

[Display(Name = "Улюблена кухня")]

public string? Cuisine { get; set; }

// Властивість для побажань щодо страв, які має бачити меню

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

public string? Dishes { get; set; }

// Властивість для вибору причини відвідування

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

public string? Reason { get; set; }

// Властивість для відповіді, чи рекомендуватимуть ресторан

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

public string? Recommend { get; set; }

}

}

**\_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() <!-- Тут буде підставлено вміст конкретної сторінки -->

</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;

}

/\* Ефект при наведенні на кнопку \*/

.btn-custom:hover {

background-color: #45a049;

}

/\* Кнопка "Скинути" з червоним фоном \*/

.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;

}

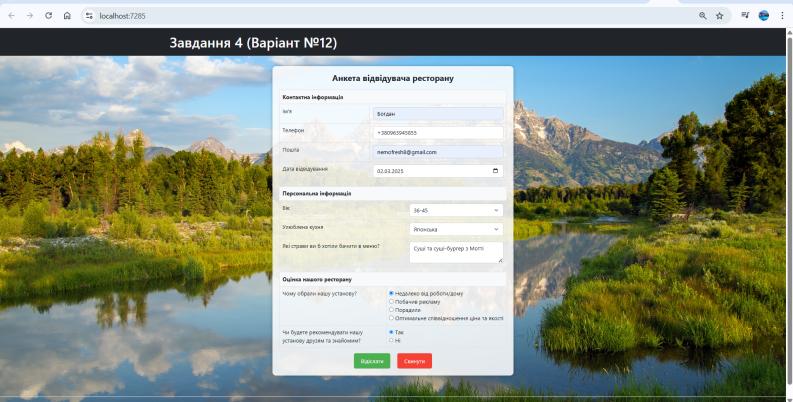
/\* Стиль для тексту у футері \*/

footer p {

color: gold;

}

## **Результати роботи програми:**



Изображение выглядит как текст, вода, облако, небо

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 моделі**:
   * Розробляється модель VisitorViewModel з властивостями (Name, Phone, Email, VisitDate, Age, Cuisine, Dishes, Reason, Recommend) та додатковими атрибутами для валідації та відображення.
2. **Розробка представлення (Index.cshtml)**:
   * Файл стає strongly typed через директиву @model Task\_4.Models.VisitorViewModel.
   * Використовуються HTML-хелпери (Html.LabelFor, Html.TextBoxFor, Html.DropDownListFor, Html.TextAreaFor) для створення полів форми, що автоматично пов’язуються з відповідними властивостями моделі.
   * Форма оформлюється у вигляді таблиць із контактною, персональною інформацією та оцінкою ресторану, а також містить кнопки «Відіслати» та «Скинути».
3. **Реалізація контролера (HomeController.cs)**:
   * Метод GET повертає новий екземпляр моделі для заповнення форми.
   * Метод POST приймає дані форми як об’єкт моделі (VisitorViewModel model) завдяки механізму Model Binding і повертає той самий View з відображенням введених даних.

## **Функціональність**

* **Запуск форми**: При GET-запиті користувач отримує чисту форму для заповнення.
* **Введення та обробка даних**:
  + Після введення інформації та натискання кнопки «Відіслати» дані автоматично прив’язуються до властивостей моделі завдяки strongly typed підходу.
  + Контролер отримує модель із заповненими даними і повертає її для відображення у вигляді результату під формою.
* **Очищення форми**: Кнопка «Скинути» викликає GET-запит, що повертає форму у початковий стан.

## **Висновок**

Застосування суворої типізації форми через модель забезпечує чистоту коду, кращу підтримку валідації та більш зрозумілий процес прив’язки даних. Такий підхід сприяє більшій масштабованості проєкту, спрощує подальше розширення функціоналу і сприяє зниженню кількості помилок при передачі даних між представленням і контролером.

# **Завдання 5**

Створити застосунок із поданням, що містить форму та розробити кон-тролер з вирішення геометричної задачі згідно свого варіанту. У випадку введення до полів форми некоректних даних вивести відповідне повідомлення.

|  |  |  |
| --- | --- | --- |
| **№ варіанту** | **Задача** | **Спосіб перемикання розрахункових параметрів** |
| 1 | 2 | 3 |
| 12 | Дани гіпотенуза рівнобічного пря-мокутного трикутника, розрахувати його периметр та площу | За допомогою списку що ро-зкривається |

## **Текст програми:**

**Index.cshtml:**

@model Task\_5.Models.TriangleViewModel

@{

// Використовуємо спільний layout та задаємо заголовок сторінки

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

ViewData["Title"] = "Геометричні розрахунки";

}

<!-- Відображення картинки рівнобічного прямокутного трикутника -->

<div class="triangle-image text-center mb-4">

<img src="https://upload.wikimedia.org/wikipedia/commons/thumb/d/d2/45-45-triangle.svg/300px-45-45-triangle.svg.png" alt="Рівнобічний прямокутний трикутник" class="img-fluid" />

</div>

<div class="form-container">

<h2 class="form-title">Розрахунок периметру та площі рівнобічного прямокутного трикутника</h2>

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

{

<!-- Поле для введення значення гіпотенузи -->

<div class="mb-3">

@Html.LabelFor(m => m.Hypotenuse)

@Html.TextBoxFor(m => m.Hypotenuse, new { @class = "form-control", type = "number", step = "any", placeholder = "Введіть значення гіпотенузи" })

@Html.ValidationMessageFor(m => m.Hypotenuse, "", new { @class = "text-danger" })

</div>

<!-- Селектор для вибору типу розрахунку -->

<div class="mb-3">

@Html.LabelFor(m => m.CalculationType, "Оберіть тип розрахунку")

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

{

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

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

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

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

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

@Html.ValidationMessageFor(m => m.CalculationType, "", new { @class = "text-danger" })

</div>

<!-- Кнопки для відправки форми або скидання даних -->

<div class="text-center">

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

<!-- Скидання форми через перехід на GET-версію -->

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

</div>

}

<!-- Якщо є повідомлення про помилку, виводимо його -->

@if (!string.IsNullOrEmpty(Model.ErrorMessage))

{

<div class="alert alert-danger mt-3">

@Model.ErrorMessage

</div>

}

else if (Model.Perimeter != null || Model.Area != null)

{

<!-- Вивід результатів розрахунку, якщо вони присутні -->

<div class="data-display mt-3">

<h4>Результати розрахунку:</h4>

<ul>

@if (Model.Perimeter != null)

{

<li><strong>Периметр:</strong> @(Model.Perimeter.Value.ToString("F2"))</li>

}

@if (Model.Area != null)

{

<li><strong>Площа:</strong> @(Model.Area.Value.ToString("F2"))</li>

}

</ul>

</div>

}

</div>

**HomeController.cs:**

using Microsoft.AspNetCore.Mvc;

using Task\_5.Models;

using System;

namespace Task\_5.Controllers

{

public class HomeController : Controller

{

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

[HttpGet]

public IActionResult Index()

{

// Створюємо нову модель для трикутника

var model = new TriangleViewModel();

return View(model);

}

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

[HttpPost]

public IActionResult Index(TriangleViewModel model)

{

// Якщо модель не пройшла валідацію, повертаємо форму з помилками

if (!ModelState.IsValid)

{

return View(model);

}

// Перевірка: значення гіпотенузи має бути більше нуля

if (model.Hypotenuse <= 0)

{

model.ErrorMessage = "Значення гіпотенузи повинно бути більше нуля.";

return View(model);

}

// Перевірка: обов'язково обрати тип розрахунку

if (string.IsNullOrEmpty(model.CalculationType))

{

model.ErrorMessage = "Будь ласка, оберіть тип розрахунку.";

return View(model);

}

// Розрахунок для рівнобічного прямокутного трикутника:

// Периметр = h + 2\*(h/√2) = h + h√2 = h\*(1 + √2)

// Площа = (h²)/4

double h = model.Hypotenuse.Value;

switch (model.CalculationType)

{

case "Периметр":

model.Perimeter = h \* (1 + Math.Sqrt(2));

model.Area = null;

break;

case "Площа":

model.Area = (h \* h) / 4.0;

model.Perimeter = null;

break;

case "Обидва":

model.Perimeter = h \* (1 + Math.Sqrt(2));

model.Area = (h \* h) / 4.0;

break;

default:

model.ErrorMessage = "Невідомий тип розрахунку.";

break;

}

return View(model);

}

}

}

**TriangleViewModel.cs:**

using System.ComponentModel.DataAnnotations;

namespace Task\_5.Models

{

public class TriangleViewModel

{

// Значення гіпотенузи (обов'язкове поле)

[Required(ErrorMessage = "Будь ласка, введіть значення гіпотенузи.")]

[Range(0.0001, double.MaxValue, ErrorMessage = "Гіпотенуза має бути додатнім числом.")]

[Display(Name = "Гіпотенуза")]

public double? Hypotenuse { get; set; }

// Тип розрахунку: "Периметр", "Площа" або "Обидва"

[Display(Name = "Розрахунок")]

public string? CalculationType { get; set; }

// Результат розрахунку периметру

[Display(Name = "Периметр")]

public double? Perimeter { get; set; }

// Результат розрахунку площі

[Display(Name = "Площа")]

public double? Area { get; set; }

// Повідомлення про помилку, якщо щось не так

public string? ErrorMessage { get; set; }

}

}

**\_Layout.cshtml:**

<!DOCTYPE html>

<html lang="uk">

<head>

<meta charset="utf-8" />

<!-- Динамічний заголовок сторінки -->

<title>@ViewData["Title"] - Task\_5</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 5 (Варіант №12)</h1>

</div>

</header>

<!-- Основний контейнер з контентом -->

<div class="container">

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

@RenderBody() <!-- Тут буде розміщено вміст конкретної сторінки -->

</main>

</div>

<!-- Футер сайту -->

<footer class="custom-footer">

<div class="container text-center">

<p>&copy; 2025 - Task\_5</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://a-static.besthdwallpaper.com/clear-waters-and-green-rocky-hills-in-america-s-wonderful-nature-wallpaper-2560x1600-107789\_7.jpg") no-repeat center center fixed;

background-size: cover;

}

/\* Контейнер для форми з розрахунками \*/

.form-container {

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

margin: 30px auto;

padding: 20px 30px;

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 \*/

.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;

}

/\* Кнопка скидання \*/

.btn-reset {

background-color: #f44336 !important;

}

.btn-reset:hover {

background-color: #e53935 !important;

}

/\* Стилізація блоку для виведення результатів \*/

.data-display {

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

margin: 20px auto;

padding: 15px 20px;

border-radius: 10px;

max-width: 700px;

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

}

.data-display h4 {

text-align: center;

}

/\* Футер сайту \*/

.custom-footer {

margin-top: 80px;

padding: 10px 0;

font-size: 0.85em;

background: rgba(0, 0, 0, 0.7);

}

.custom-footer p {

color: gold;

margin: 0;

}

/\* Стиль для картинки трикутника \*/

.triangle-image img {

max-width: 300px;

border: 2px solid #fff;

border-radius: 5px;

}

## **Результати роботи програми:**

Изображение выглядит как текст, снимок экрана, дерево, риф

Контент, сгенерированный ИИ, может содержать ошибки.

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

Контент, сгенерированный ИИ, может содержать ошибки.

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

Контент, сгенерированный ИИ, может содержать ошибки.

## **Алгоритм побудови кодів**

1. **Створення моделі (TriangleViewModel):**
   * Визначаються властивості для введення значення гіпотенузи, типу розрахунку, а також для збереження обчислених результатів (периметр, площа) та повідомлень про помилки.
2. **Розробка представлення (Index.cshtml):**
   * Представлення робиться strongly typed із зазначенням моделі через директиву @model.
   * Використовуються HTML-хелпери (Html.LabelFor, Html.TextBoxFor, Html.DropDownListFor, Html.ValidationMessageFor) для створення форми, де користувач вводить значення гіпотенузи та обирає тип розрахунку (периметр, площа або обидва).
   * Форма включає кнопки «Відіслати» для передачі даних та «Скинути» для повернення до початкового стану, а також область для виведення результатів або повідомлень про помилки.
3. **Реалізація контролера (HomeController):**
   * GET-метод повертає порожню форму, створюючи новий екземпляр моделі.
   * POST-метод приймає модель, перевіряє її на коректність (наприклад, значення гіпотенузи має бути більше нуля та обрано тип розрахунку) і в разі виявлення помилок повертає форму з відповідними повідомленнями.
   * Якщо дані коректні, обчислюються периметр (за формулою: *h*(1 + √2)) та/або площа (за формулою: *h²/4*) рівнобічного прямокутного трикутника, після чого модель з результатами повертається до представлення.
4. **Оформлення стилів та Layout:**
   * Використання Bootstrap та власних CSS-файлів для створення привабливого інтерфейсу, включаючи розміщення форми, відображення зображення трикутника та результатів розрахунків.

## **Функціональність**

* **Введення даних:** Користувач вводить значення гіпотенузи та обирає тип розрахунку через випадаючий список.
* **Валідація:** Система перевіряє, чи введене значення гіпотенузи є додатнім числом та чи обрано тип розрахунку. У випадку некоректного введення виводяться відповідні повідомлення про помилки.
* **Обчислення:** За коректних даних контролер виконує розрахунок периметра та/або площі за заданими формулами.
* **Відображення результатів:** Результати обчислень або повідомлення про помилки виводяться під формою, що дозволяє користувачу одразу бачити результат своєї взаємодії.

## **Висновок**

Реалізація завдання демонструє застосування суворої типізації форми за допомогою strongly typed моделі, що забезпечує чистоту коду та ефективну валідацію даних. Завдяки інтеграції серверної логіки та зручного інтерфейсу користувача, застосунок не лише виконує геометричні обчислення, але й гарантує коректність введених даних, що є критично важливим для створення надійних веб-застосунків.

# **Загальний Висновок**

Виконання лабораторної роботи дозволило отримати практичний досвід у використанні форм у веб-застосунках, створених на основі ASP.NET MVC. У процесі виконання завдань було досліджено різні підходи до отримання та обробки даних з форм, включаючи використання:

* **Request.Form** – для отримання даних у найпростішому вигляді.
* **IFormCollection** – для збереження та обробки введеної користувачем інформації у вигляді колекції.
* **Слабкої типізації** – коли дані передаються у методи контролера у вигляді окремих параметрів.
* **Сильної типізації** – використовуючи модельну прив’язку та strongly-typed моделі.

Досліджено валідацію введених даних, обробку помилок та їх коректне відображення для користувача. Також було продемонстровано використання ViewBag для збереження та передачі даних між контролером і представленням. Крім того, реалізовано інтеграцію з Bootstrap для покращення зовнішнього вигляду форми.

Завдяки застосуванню цих підходів розроблені веб-сторінки стали більш зручними, інтуїтивно зрозумілими та адаптивними до користувацьких потреб. Отримані знання та навички можуть бути корисними при розробці реальних веб-застосунків, що працюють з динамічними формами введення даних.

Таким чином, виконання лабораторної роботи дозволило засвоїти основи роботи з формами у ASP.NET MVC, механізми передачі та обробки даних, а також принципи побудови зручного інтерфейсу для користувача.