

**Виконав: Мельник Б. В.; КН-922в**

### **Лабораторна робота №1**

**Тема:** Створення простих MVC WEB-застосунків засобами ASP.NET.

**Мета роботи:** придбати практику у створенні простих MVC WEB-додатків, заснованих на зв'язках Model - View Controller з невеликою кількістю конт-ролерів та елементів подання і без використання моделей.

**Програмні засоби розробки:** середовище програмування (IDE-Integrated Development Environment) MS Visual Studio, програмний фреймворк ASP.NET версії не нижче 6.0 та дуже бажано використання дизайнерського фреймворку BOOTSTRAP, або якогось іншого відповідного аналогу, MySQL Server, або якась аналогічна СУБД.

### **Індивідуальні завдання**

#### **Варіант №12**

#### **Завдання 1**

Створити MVC WEB-застосунок, головне подання (Index) якого містить нетипізовану (непов'язану із моделлю) форму (текстові поля, області, списки, радібатони, чекбокси тощо.) для задання групи параметрів, які через контролер, що викликається з цієї ж форми, передаються до іншого подання, на якому створюються три блоки (абзаци) тексту, котрі відрізняються один від одного за розміром, кольором, вмістом, типом вирівнювання та формою тексту; кольором, зображенням (із його параметрами) фону; розмірами та відступами між блоками. Під час виконання завдання треба врахувати те, що за номером власного варіанту частина інформації (вміст та css-параметри абзаців) має розташовуватися в діалогових об'єктах форми головного подання застосунку, а інша частина css-параметрів має бути зчитана контролером із відповідної таблиці чи таблиць бази даних (БД). База даних та таблиця (ї) мають бути створені заздалегідь і мати будь-яку структуру та вміст у будь-якій системі управління базами даних.

<b>№ варіанту</b>	<b>Вміст абзаців</b>	<b>Текстові параметри</b>	<b>Фонові параметри</b>	<b>Розміри та відступи</b>
12	FORM	FORM	FORM	FORM

## Текст програми:

### 1. Модель

**ParagraphStyle.cs** – файл, який описує модель (шрифти, відступи тощо).

```
using System.ComponentModel.DataAnnotations;

namespace Task_1.Models
{
    public class ParagraphStyle
    {
        [Key]
        public int Id { get; set; }

        // Розмір шрифту
        public string FontSize { get; set; } = string.Empty;

        // Відступи ззовні
        public string Margin { get; set; } = string.Empty;

        // Відступи зсередини
        public string Padding { get; set; } = string.Empty;
    }
}
```

### 2. Контекст бази даних

**ApplicationDbContext.cs** – файл контексту, де визначено DbSet для таблиці з параметрами абзаців.

```
using Microsoft.EntityFrameworkCore;
using Task_1.Models; // Модель ParagraphStyle

namespace Task_1.Data
{
    // Контекст БД для Task_1
    public class ApplicationDbContext : DbContext
    {
        // Конструктор який приймає опції контексту
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options)
        {
        }

        // Власт для доступу до табл ParagraphStyles (розміри та відступи)
    }
}
```

```

        public DbSet<ParagraphStyle> ParagraphStyles { get; set; }
    }
}

```

### 3. Контролер

**HomeController.cs** – файл контролера, який зчитує дані з форми, отримує параметри з бази даних і передає їх у ViewBag для наступного подання.

```

using Microsoft.AspNetCore.Mvc;
using Task_1.Data;          // Контекст БД
using Task_1.Models;        // Модель ParagraphStyle

namespace Task_1.Controllers
{
    public class HomeController : Controller
    {
        // Контекст БД для роботи з табл ParagraphStyles
        private readonly ApplicationDbContext _context;

        // Констр отримує контекст через dependency injection
        public HomeController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: Home/Index
        [HttpGet]
        public IActionResult Index()
        {
            // Повертаємо форму для введення даних (нетипізована форма)
            return View();
        }

        // POST: Home/IndexPost
        [HttpPost]
        public IActionResult IndexPost()
        {
            // Зчитуємо дані з форми через Request.Form (нетипізовано)
            string paragraph1Text = Request.Form["paragraph1Text"];
            string paragraph1Color = Request.Form["paragraph1Color"];
            string paragraph1Align = Request.Form["paragraph1Align"];
            string paragraph1BackgroundColor = Request.Form["paragraph1BgColor"];
            string paragraph1BackgroundImage = Request.Form["paragraph1BgImage"];

```

```
string paragraph2Text = Request.Form["paragraph2Text"];
string paragraph2Color = Request.Form["paragraph2Color"];
string paragraph2Align = Request.Form["paragraph2Align"];
string paragraph2BackgroundColor = Request.Form["paragraph2BgColor"];
string paragraph2BackgroundImage = Request.Form["paragraph2BgImage"];

string paragraph3Text = Request.Form["paragraph3Text"];
string paragraph3Color = Request.Form["paragraph3Color"];
string paragraph3Align = Request.Form["paragraph3Align"];
string paragraph3BackgroundColor = Request.Form["paragraph3BgColor"];
string paragraph3BackgroundImage = Request.Form["paragraph3BgImage"];

// Отримуємо з БД дані стилю для кожного абзацу
ParagraphStyle style1 = _context.ParagraphStyles.Find(1);
ParagraphStyle style2 = _context.ParagraphStyles.Find(2);
ParagraphStyle style3 = _context.ParagraphStyles.Find(3);

// Передаємо дані для 1 абзацу через ViewBag
ViewBag.Paragraph1Text = paragraph1Text;
ViewBag.Paragraph1Color = paragraph1Color;
ViewBag.Paragraph1Align = paragraph1Align;
ViewBag.Paragraph1BgColor = paragraph1BackgroundColor;
ViewBag.Paragraph1BgImage = paragraph1BackgroundImage;
// Явне приведення літералів до string
ViewBag.Paragraph1FontSize = style1?.FontSize ?? (string)"16px";
ViewBag.Paragraph1Margin = style1?.Margin ?? (string)"0px";
ViewBag.Paragraph1Padding = style1?.Padding ?? (string)"0px";

// Передаємо дані для 2 абзацу через ViewBag
ViewBag.Paragraph2Text = paragraph2Text;
ViewBag.Paragraph2Color = paragraph2Color;
ViewBag.Paragraph2Align = paragraph2Align;
ViewBag.Paragraph2BgColor = paragraph2BackgroundColor;
ViewBag.Paragraph2BgImage = paragraph2BackgroundImage;
ViewBag.Paragraph2FontSize = style2?.FontSize ?? (string)"16px";
ViewBag.Paragraph2Margin = style2?.Margin ?? (string)"0px";
ViewBag.Paragraph2Padding = style2?.Padding ?? (string)"0px";

// Передаємо дані для 3 абзацу через ViewBag
ViewBag.Paragraph3Text = paragraph3Text;
ViewBag.Paragraph3Color = paragraph3Color;
ViewBag.Paragraph3Align = paragraph3Align;
ViewBag.Paragraph3BgColor = paragraph3BackgroundColor;
```

```

        ViewBag.Paragraph3BgImage = paragraph3BackgroundImage;
        ViewBag.Paragraph3FontSize = style3?.FontSize ?? (string)"16px";
        ViewBag.Paragraph3Margin = style3?.Margin ?? (string)"0px";
        ViewBag.Paragraph3Padding = style3?.Padding ?? (string)"0px";

        // Перенаправляємо дані до подання Display
        return View("Display");
    }

    // Дод екшн для відображення подання
    public IActionResult Display()
    {
        return View();
    }
}
}

```

#### 4. Подання (Views)

**Index.cshtml** – головне подання з формою введення даних (нетипізована форма).

**Display.cshtml** – подання, яке отримує дані з ViewBag і відображає абзаци з параметрами (шрифти, відступи, фони тощо).

**Layout.cshtml** – шаблон, що забезпечує загальне оформлення застосунку (Navbar, контейнер, футер, фон).

**\_Layout.cshtml.css** – файл, який містить кастомні CSS-стилі для загального оформлення веб-застосунку (Navbar, контейнер, футер, фон тощо).

#### Index.cshtml:

```

@{
    // Заголовок сторінки
    ViewData["Title"] = "Головна (Index)";
}

<!-- Заголовок форми -->
<h2 class="mb-4">Форма (нетипізована) для введення параметрів абзаців</h2>

<!-- Форма для введення параметрів (POST-запит до HomeController, метод IndexPost) -->
<form method="post" asp-action="IndexPost" asp-controller="Home">
    <div class="row">
        <!-- Параграф 1 -->
        <div class="col-md-4">
            <h4>Параграф 1</h4>

```

```

<!-- Поле введення тексту -->
<div class="mb-3">
    <label class="form-label">Текст:</label>
    <input type="text" name="paragraph1Text" value="FORM" class="form-control" />
</div>
<!-- Поле введення кольору тексту -->
<div class="mb-3">
    <label class="form-label">Колір тексту (color):</label>
    <input type="text" name="paragraph1Color" placeholder="red, #FF0000"
class="form-control" />
</div>
<!-- Випадаючий список вирівнювання тексту -->
<div class="mb-3">
    <label class="form-label">Вирівнювання (text-align):</label>
    <select name="paragraph1Align" class="form-select">
        <option value="left">left</option>
        <option value="center">center</option>
        <option value="right">right</option>
        <option value="justify">justify</option>
    </select>
</div>
<!-- Поле введення кольору фону -->
<div class="mb-3">
    <label class="form-label">Фон (colір) background-color:</label>
    <input type="text" name="paragraph1BgColor" placeholder="white / #FFFFFF / ..."
class="form-control" />
</div>
<!-- Поле введення URL фонового зображення -->
<div class="mb-3">
    <label class="form-label">Фон (зображення) background-image (URL):</label>
    <input type="text" name="paragraph1BgImage" placeholder="https://..."
class="form-control" />
</div>
</div>

<!-- Параграф 2 -->
<div class="col-md-4">
    <h4>Параграф 2</h4>
    <!-- Поле введення тексту -->
    <div class="mb-3">
        <label class="form-label">Текст:</label>
        <input type="text" name="paragraph2Text" value="FORM" class="form-control" />
    </div>
    <!-- Поле введення кольору тексту -->

```

```

        <div class="mb-3">
            <label class="form-label">Колір тексту:</label>
            <input type="text" name="paragraph2Color" placeholder="red, #FF0000"
class="form-control" />
        </div>
        <!-- Випадаючий список вирівнювання -->
        <div class="mb-3">
            <label class="form-label">Вирівнювання:</label>
            <select name="paragraph2Align" class="form-select">
                <option value="left">left</option>
                <option value="center">center</option>
                <option value="right">right</option>
                <option value="justify">justify</option>
            </select>
        </div>
        <!-- Поле введення кольору фону -->
        <div class="mb-3">
            <label class="form-label">Фон (колір):</label>
            <input type="text" name="paragraph2BgColor" placeholder="white" class="form-
control" />
        </div>
        <!-- Поле введення URL фонового зображення -->
        <div class="mb-3">
            <label class="form-label">Фон (зображення) (URL):</label>
            <input type="text" name="paragraph2BgImage" placeholder="https://..."
class="form-control" />
        </div>
    </div>

    <!-- Параграф 3 -->
    <div class="col-md-4">
        <h4>Параграф 3</h4>
        <!-- Поле введення тексту -->
        <div class="mb-3">
            <label class="form-label">Текст:</label>
            <input type="text" name="paragraph3Text" value="FORM" class="form-control" />
        </div>
        <!-- Поле введення кольору тексту -->
        <div class="mb-3">
            <label class="form-label">Колір тексту:</label>
            <input type="text" name="paragraph3Color" placeholder="red, #FF0000"
class="form-control" />
        </div>
        <!-- Випадаючий список вирівнювання -->

```

```

<div class="mb-3">
    <label class="form-label">Вирівнювання:</label>
    <select name="paragraph3Align" class="form-select">
        <option value="left">left</option>
        <option value="center">center</option>
        <option value="right">right</option>
        <option value="justify">justify</option>
    </select>
</div>
<!-- Поле введення кольору фону -->
<div class="mb-3">
    <label class="form-label">Фон (колір):</label>
    <input type="text" name="paragraph3BgColor" placeholder="white" class="form-
control" />
</div>
<!-- Поле введення URL фонового зображення -->
<div class="mb-3">
    <label class="form-label">Фон (зображення) (URL):</label>
    <input type="text" name="paragraph3BgImage" placeholder="https://..."
class="form-control" />
</div>
</div>
</div>

<!-- Горизонтальна лінія для розділення -->
<hr />
<!-- Кнопка відправки форми -->
<button type="submit" class="btn btn-primary">Надіслати</button>
</form>

```

## Display.cshtml:

```

@{
    // Встановлюємо заголовок сторінки
    ViewData["Title"] = "Результат (Display)";

    // Дані для 1 абзацу з ViewBag
    var p1Text = ViewBag.Paragraph1Text;
    var p1Color = ViewBag.Paragraph1Color;
    var p1Align = ViewBag.Paragraph1Align;
    var p1BgColor = ViewBag.Paragraph1BgColor;
    var p1BgImage = ViewBag.Paragraph1BgImage;
    var p1FontSize = ViewBag.Paragraph1FontSize;
    var p1Margin = ViewBag.Paragraph1Margin;
}

```



```

var p1Padding = ViewBag.Paragraph1Padding;

// Дані для 2 абзацу
var p2Text = ViewBag.Paragraph2Text;
var p2Color = ViewBag.Paragraph2Color;
var p2Align = ViewBag.Paragraph2Align;
var p2BgColor = ViewBag.Paragraph2BgColor;
var p2BgImage = ViewBag.Paragraph2BgImage;
var p2FontSize = ViewBag.Paragraph2FontSize;
var p2Margin = ViewBag.Paragraph2Margin;
var p2Padding = ViewBag.Paragraph2Padding;

// Дані для 3 абзацу
var p3Text = ViewBag.Paragraph3Text;
var p3Color = ViewBag.Paragraph3Color;
var p3Align = ViewBag.Paragraph3Align;
var p3BgColor = ViewBag.Paragraph3BgColor;
var p3BgImage = ViewBag.Paragraph3BgImage;
var p3FontSize = ViewBag.Paragraph3FontSize;
var p3Margin = ViewBag.Paragraph3Margin;
var p3Padding = ViewBag.Paragraph3Padding;
}

<!-- Заголовок вмісту -->
<h2 class="mb-3">3генеровані абзаци</h2>
<!-- Звідки беруться параметри -->
<p class="mb-4">
    Параметри кольору, тексту, фону тощо взято з форми.<br />
    Параметри розміру та відступів (FontSize, Margin, Padding) взято з БД (таблиця
ParagraphStyles).
</p>

<div class="mb-4">
    <!-- Блок для 1 абзацу -->
    <div class="border p-3 mb-3">
        <p style="
            color:@p1Color;
            text-align:@p1Align;
            background-color:@p1BgColor;
            background-image: @(string.IsNullOrEmpty(p1BgImage) ? "none" : "url('" + p1BgImage +
            "')");
            font-size:@p1FontSize;
            margin:@p1Margin;
            padding:@p1Padding;

```

```

        ">
        @p1Text
    </p>
</div>

<!-- Блок для 2 абзацу -->
<div class="border p-3 mb-3">
    <p style="
        color:@p2Color;
        text-align:@p2Align;
        background-color:@p2BgColor;
        background-image: @(string.IsNullOrEmpty(p2BgImage) ? "none" : "url('" + p2BgImage
+ "')");
        font-size:@p2FontSize;
        margin:@p2Margin;
        padding:@p2Padding;
    ">
        @p2Text
    </p>
</div>

<!-- Блок для 3 абзацу -->
<div class="border p-3 mb-3">
    <p style="
        color:@p3Color;
        text-align:@p3Align;
        background-color:@p3BgColor;
        background-image: @(string.IsNullOrEmpty(p3BgImage) ? "none" : "url('" + p3BgImage
+ "')");
        font-size:@p3FontSize;
        margin:@p3Margin;
        padding:@p3Padding;
    ">
        @p3Text
    </p>
</div>
</div>

<br />
<!-- Кнопка для повернення на головну сторінку -->
<a asp-action="Index" asp-controller="Home" class="btn btn-secondary">Назад</a>

```

## \_Layout.cshtml:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Task_1</title>
    <!-- Підключення Bootstrap CSS -->
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <!-- Підключення власних стилів -->
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link rel="stylesheet" href="~/Task_1.styles.css" asp-append-version="true" />
</head>
<body style="background: url('https://coolbackgrounds.io/images/unsplash/josh-bean-medium-9501ba9f.jpg') no-repeat center center fixed; background-size: cover;">
    <header>
        <!-- Navbar з мін відступом знизу -->
        <nav class="navbar navbar-expand-sm navbar-dark bg-dark border-bottom box-shadow mb-1">
            <div class="container-fluid">
                <!-- Логотип/назва сайту -->
                <a class="navbar-brand text-white" asp-area="" asp-controller="Home" asp-
action="Index">Task_1</a>
                <!-- Кнопка для адапт меню -->
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target=".navbar-collapse"
                    aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <!-- Основне меню -->
                <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
                    <ul class="navbar-nav flex-grow-1">
                        <li class="nav-item">
                            <a class="nav-link text-white" asp-area="" asp-controller="Home"
asp-action="Index">Home</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-white" asp-area="" asp-controller="Home"
asp-action="Privacy">Privacy</a>
                        </li>
                    </ul>
            </div>
        </nav>
    </header>
```

```

        </div>
    </div>
</nav>
</header>

<!-- Основний контейнер для вмісту -->
<div class="container my-5" style="background-color: rgba(255, 255, 255, 0.9); border-radius:
10px; padding: 20px;">
    <main role="main" class="pb-3">
        @RenderBody() <!-- Рендер вміст кожної сторінки -->
    </main>
</div>

<!-- Футер з відступом зверху -->
<footer class="border-top footer text-muted mt-4">
    <div class="container">
        &copy; 2025 - Task_1 -
        <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
    </div>
</footer>

<!-- Підключення jQuery, Bootstrap JS та скриптів -->
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>
@await RenderSectionAsync("Scripts", required: false)
</body>
</html>

```

## \_Layout.cshtml.css:

/\* Please see documentation at <https://learn.microsoft.com/aspnet/core/client-side/bundling-and-minification>

for details on configuring this project to bundle and minify static web assets. \*/

/\* Стиль для логотипу в Navbar \*/

```

a.navbar-brand {
    white-space: normal;
    text-align: center;
    word-break: break-all;
}

```

/\* Загальний стиль для всіх посилань \*/

```

a {

```

```
        color: #0077cc;
    }

/* Стиль для кнопок primary */
.btn-primary {
    color: #fff;
    background-color: #1b6ec2;
    border-color: #1861ac;
}

/* Стиль для активних елементів в nav-pills */
.nav-pills .nav-link.active,
.nav-pills .show > .nav-link {
    color: #fff;
    background-color: #1b6ec2;
    border-color: #1861ac;
}

/* Стиль для верхньої межі */
.border-top {
    border-top: 1px solid #e5e5e5;
}

/* Стиль для нижньої межі */
.border-bottom {
    border-bottom: 1px solid #e5e5e5;
}

/* Легка тінь для об'ємного ефекту */
.box-shadow {
    box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);
}

/* Стиль для кнопки прийняття політики */
button.accept-policy {
    font-size: 1rem;
    line-height: inherit;
}

/* Стиль для футера */
.footer {
    position: absolute;
    width: 100%;
    white-space: nowrap;
```

```
    line-height: 60px;  
}
```

## 5. SQL-скрипт:

```
CREATE TABLE [dbo].[ParagraphStyles](  
    [Id] INT NOT NULL IDENTITY(1,1) PRIMARY KEY,  
    [FontSize] NVARCHAR(50) NOT NULL,  
    [Margin] NVARCHAR(50) NOT NULL,  
    [Padding] NVARCHAR(50) NOT NULL  
);  
  
INSERT INTO [dbo].[ParagraphStyles] (FontSize, Margin, Padding)  
VALUES  
('14px', '10px', '5px'),  
('18px', '15px', '10px'),  
('22px', '20px', '15px');
```

## Результат роботи програми:

Task\_1 Home Privacy

localhost:7249

Приват 24/7 Teams

### Форма (нетипізована) для введення параметрів абзаців

Параграф 1	Параграф 2	Параграф 3
Текст: <input type="text" value="FORM"/>	Текст: <input type="text" value="FORM"/>	Текст: <input type="text" value="FORM"/>
Колір тексту (color): <input type="text" value="red, #FF0000"/>	Колір тексту: <input type="text" value="red, #FF0000"/>	Колір тексту: <input type="text" value="red, #FF0000"/>
Вирівнювання (text-align): <input type="text" value="left"/>	Вирівнювання: <input type="text" value="left"/>	Вирівнювання: <input type="text" value="left"/>
Фон (колір) background-color: <input type="text" value="white / #FFFFFF / ..."/>	Фон (колір): <input type="text" value="white"/>	Фон (колір): <input type="text" value="white"/>
Фон (зображення) background-image (URL): <input type="text" value="https://..."/>	Фон (зображення) (URL): <input type="text" value="https://..."/>	Фон (зображення) (URL): <input type="text" value="https://..."/>

localhost:7249

Приват 24/7 Teams

Task\_1 Home Privacy

### Форма (нетипізована) для введення параметрів абзаців

#### Параграф 1

Текст:

Привіт, це перший абзац

Колір тексту (color):

red

Вирівнювання (text-align):

left

Фон (колір) background-color:

#FFFFFF

Фон (зображення) background-image (URL):

https://cs10.pikabu.ru/post\_img/big/2020/05/12/9/15E

#### Параграф 2

Текст:

Це другий параграф

Колір тексту:

blue

Вирівнювання:

center

Фон (колір):

white

Фон (зображення) (URL):

https://cs12.pikabu.ru/post\_img/big/2020/05/12/9/15E

#### Параграф 3

Текст:

І нарешті третій блок тексту

Колір тексту:

red, #FF0000

Вирівнювання:

left

Фон (колір):

#333333

Фон (зображення) (URL):

https://cs11.pikabu.ru/post\_img/big/2020/05/12/9/15E

Надіслати

localhost:7249/Home/IndexPost

Приват 24/7 Teams

Task\_1 Home Privacy

### Згенеровані абзаци

Параметри кольору, тексту, фону тощо взято з форми.  
Параметри розміру та відступів (FontSize, Margin, Padding) взято з БД (таблиця ParagraphStyles).

Привіт, це перший абзац

Це другий параграф

І нарешті третій блок тексту

Назад

Файл Правка Вид Git Проект Сборка Отладка Тест Анализ Средства Расширения Окно Справка

Debug

Any CPU

Task\_1

https

Обозреватель объектов SQL Server

SQL Server

(localdb)\MSSQLLocalDB (SQL Server 15.0.4)

Базы данных

Системные базы данных

C:\USERS\NEMOF\ONEDRIVE\ИЗОБ

Task1Db

Таблицы

Системные таблицы

Внешние таблицы

Удаленные таблицы реестра

dbo.ParagraphStyles

.Paragrap...les [Данные]

HomeController.cs

\_Layout.cshtml

Максимальное количество строк: 1000

Id	FontSize	Margin	Padding
1	14px	10px	5px
2	18px	15px	10px
3	22px	20px	15px
NULL	NULL	NULL	NULL

## Алгоритм побудови проєкту і кодів

### 1. Створення рішення і проєкту:

- Створив рішення (*Lab\_1*) у Visual Studio.
- Додав новий проєкт ASP.NET Core MVC із назвою *Task\_1*.

### 2. Налаштування бази даних:

- Створив базу даних LocalDB (внутрішню базу Visual Studio).
- Створив SQL-скрипт для таблиці **ParagraphStyles** з полями:
  - Id (ціле число, автоінкремент, первинний ключ)
  - FontSize (NVARCHAR(50))
  - Margin (NVARCHAR(50))
  - Padding (NVARCHAR(50))
- Додав цей SQL-скрипт до проєкту для подальшого використання.

### 3. Розробка моделі і контексту:

- Створив клас **ParagraphStyle.cs** у папці *Models*, який описує властивості таблиці.
- Створив клас **ApplicationDbContext.cs** у папці *Data*, де визначив DbSet для таблиці ParagraphStyles.

### 4. Розробка контролера:

- Створив **HomeController.cs** у папці *Controllers*, який містить:
  - **GET метод Index()** – повертає головну сторінку із формою для введення даних (нетипізована форма).
  - **POST метод IndexPost()** – зчитує дані з форми за допомогою Request.Form, отримує дані про розміри (FontSize, Margin, Padding) з бази даних (через ApplicationDbContext) і передає всі параметри через ViewBag до подання Display.
  - **Метод Display()** – повертає подання Display, де відображаються згенеровані абзаци.

### 5. Розробка подань (Views):

- **Index.cshtml** – містить форму для введення даних:
  - Текст для кожного абзацу
  - Параметри тексту (колір, вирівнювання)
  - Фонові параметри (колір, URL зображення)



- Ці дані вводяться користувачем через діалогові вікна форми.
- **Display.cshtml** – відображає абзаци:
  - Дані з форми (вміст, текстові і фонографічні параметри) передаються через ViewBag.
  - Розміри та відступи (FontSize, Margin, Padding) беруться з бази даних.
  - Абзаци оформлюються через inline CSS атрибути.
- **\_Layout.cshtml** – загальний шаблон, що включає навігаційну панель (Navbar), основний контейнер для вмісту та футер. Оформлення здійснене із використанням Bootstrap, фон задається через інлайн-стиль.

## 6. Тестування:

- Запустив застосунок, заповнив форму параметрами.
- Переконався, що дані з форми та параметри з бази даних передаються через контролер і відображаються у поданні Display.
- Всі стилі застосовані через Bootstrap та власні CSS, що забезпечує адаптивне оформлення.

## Функціональність

- **Введення даних:** Користувач заповнює форму з параметрами для трьох абзців (текст, кольори, вирівнювання, фон тощо).
- **Обробка даних:** Контролер зчитує дані з форми, отримує розміри та відступи з бази даних і передає усі значення у ViewBag.
- **Відображення:** Подання Display відображає три абзаци з унікальними CSS-параметрами, що залежать від введених даних і даних з БД.

## Висновок

Завдання виконано відповідно до варіанту №12. Проєкт демонструє використання ASP.NET Core MVC, EF Core і Bootstrap для динамічного формування вмісту. Структура проєкту чітка, і застосунок ефективно взаємодіє з БД для отримання параметрів стилю.

## Завдання 2 1

Створити MVC WEB-застосунок, головне подання якого містить нетипізовану форму для задання групи параметрів, які через контролер, що викликається з цієї ж форми, передаються до іншого подання, на якому формуються зображення згідно із топологією власного варіанту.

№ Варіанту	Форма топології
1	2
12	

Посилання на зображення задаються на формі, розміри зображень з таблиці відповідної БД.

### Текст програми:

#### 1. Модель

**ImageDimension.cs** – файл, який описує модель розмірів зображення (Width, Height). Цей клас використовується для зберігання параметрів зображень у БД.

```
using System.ComponentModel.DataAnnotations;
```

```
namespace Task_2_1.Models
{
    public class ImageDimension
    {
        [Key] // Первинний ключ
        public int Id { get; set; }

        // Ширина зображення
        public string Width { get; set; } = string.Empty;

        // Висота зображення
        public string Height { get; set; } = string.Empty;
    }
}
```

## 2. Контекст БД

**ApplicationDbContext.cs** – файл контексту, де визначено DbSet для таблиці з розмірами зображень (ImageDimensions). Він забезпечує зв'язок між застосунком і БД (LocalDB).

```
using Microsoft.EntityFrameworkCore;
using Task_2_1.Models;

namespace Task_2_1.Data
{
    // Контекст БД для Task_2_1
    public class ApplicationDbContext : DbContext
    {
        // Конст отримує опції через DI і --> їх у базовий констр
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options)
        {
        }

        // Властивість для роботи з табл ImageDimensions у БД
        public DbSet<ImageDimension> ImageDimensions { get; set; }
    }
}
```

## 3. Контролер

**HomeController.cs** – файл контролера, який зчитує дані з форми (URL зображень) та отримує розміри зображень із БД. Дані передаються через ViewBag до подання Display для відображення зображень за заданою топологією.

```
using Microsoft.AspNetCore.Mvc;
using Task_2_1.Data;
using Task_2_1.Models;

namespace Task_2_1.Controllers
{
    public class HomeController : Controller
    {
        // Контекст БД який ми отримуємо через DI
        private readonly ApplicationDbContext _context;

        // Констр який приймає контекст БД
        public HomeController(ApplicationDbContext context)
```

```

{
    _context = context;
}

// GET: Home/Index
[HttpGet]
public IActionResult Index()
{
    // Повертаємо головну сторінку з формою
    return View();
}

// POST: Home/IndexPost Відправка даних форми
[HttpPost]
public IActionResult IndexPost()
{
    // Зчитуємо URL-адреси з форми
    string image1Url = Request.Form["image1Url"];
    string image2Url = Request.Form["image2Url"];
    string image3Url = Request.Form["image3Url"];
    string image4Url = Request.Form["image4Url"];
    string image5Url = Request.Form["image5Url"];

    // Отримуємо розміри зображень з БД
    var dim1 = _context.ImageDimensions.Find(1);
    var dim2 = _context.ImageDimensions.Find(2);
    var dim3 = _context.ImageDimensions.Find(3);
    var dim4 = _context.ImageDimensions.Find(4);
    var dim5 = _context.ImageDimensions.Find(5);

    // Передаємо URL-адреси зображень у ViewBag
    ViewBag.Image1Url = image1Url;
    ViewBag.Image2Url = image2Url;
    ViewBag.Image3Url = image3Url;
    ViewBag.Image4Url = image4Url;
    ViewBag.Image5Url = image5Url;

    // Передаємо розміри зображень з БД у ViewBag
    ViewBag.Image1Width = dim1?.Width ?? "80px";
    ViewBag.Image1Height = dim1?.Height ?? "60px";

    ViewBag.Image2Width = dim2?.Width ?? "100px";
    ViewBag.Image2Height = dim2?.Height ?? "80px";

```

```

        ViewBag.Image3Width = dim3?.Width ?? "120px";
        ViewBag.Image3Height = dim3?.Height ?? "100px";

        ViewBag.Image4Width = dim4?.Width ?? "140px";
        ViewBag.Image4Height = dim4?.Height ?? "120px";

        ViewBag.Image5Width = dim5?.Width ?? "160px";
        ViewBag.Image5Height = dim5?.Height ?? "140px";

        // Повертаємо подання Display з усіма переданими даними
        return View("Display");
    }

    // Додатковий екшн для відображення
    public IActionResult Display()
    {
        return View();
    }
}
}

```

#### 4. Подання (Views)

**Index.cshtml** – головне подання з формою введення даних, де користувач вводить URL зображень (форма нетипізована).

**Display.cshtml** – подання, яке отримує дані з ViewBag (URL та розміри зображень) і відображає зображення з використанням заданих параметрів (ширина, висота) та CSS-стилів.

**\_Layout.cshtml** – загальний шаблон застосунку, що забезпечує оформлення (Navbar, основний контейнер, футер, фон). Шаблон використовує Bootstrap для адаптивного оформлення.

**\_Layout.cshtml.css** – файл, який містить кастомні CSS-стилі для загального оформлення веб-застосунку (Navbar, контейнер, футер, фон тощо).

##### **Index.cshtml:**

```

@{
    ViewData["Title"] = "Головна";
}

<!-- Обгортка для форми із відступами -->
<div class="container my-4">

```

```
<h2 class="mb-4">Форма для посилань на п'ять зображень</h2>
```

```
<!-- Форма що відправляє дані методом POST на IndexPost у HomeController -->
```

```
<form method="post" asp-action="IndexPost" asp-controller="Home">
```

```
  <!-- Використовується відступ gutter -->
```

```
  <div class="row g-3">
```

```
    <!-- Поле URL 1 -->
```

```
    <div class="col-md-2">
```

```
      <label class="form-label">Зобп1 (URL):</label>
```

```
      <input type="text" name="image1Url" class="form-control"
```

```
placeholder="https://site.com/img1.jpg" />
```

```
    </div>
```

```
    <!-- Поле URL 2 -->
```

```
    <div class="col-md-2">
```

```
      <label class="form-label">Зобп2 (URL):</label>
```

```
      <input type="text" name="image2Url" class="form-control"
```

```
placeholder="https://site.com/img2.jpg" />
```

```
    </div>
```

```
    <!-- Поле URL 3 -->
```

```
    <div class="col-md-2">
```

```
      <label class="form-label">Зобп3 (URL):</label>
```

```
      <input type="text" name="image3Url" class="form-control"
```

```
placeholder="https://site.com/img3.jpg" />
```

```
    </div>
```

```
    <!-- Поле URL 4 -->
```

```
    <div class="col-md-2">
```

```
      <label class="form-label">Зобп4 (URL):</label>
```

```
      <input type="text" name="image4Url" class="form-control"
```

```
placeholder="https://site.com/img4.jpg" />
```

```
    </div>
```

```
    <!-- Поле URL 5 -->
```

```
    <div class="col-md-2">
```

```
      <label class="form-label">Зобп5 (URL):</label>
```

```
      <input type="text" name="image5Url" class="form-control"
```

```
placeholder="https://site.com/img5.jpg" />
```

```
    </div>
```

```
  </div>
```

```
  <!-- Блок з кнопкою відправки форми з відступом зверху -->
```

```
  <div class="mt-4">
```

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

```
  </div>
```

```
</form>
```

```
</div>
```

## Display.cshtml:

```
@{
    ViewData["Title"] = "Відображення п'яти зображень (Топологія варіант №12)";

    // Отримуємо URL зображень з ViewBag
    var url1 = ViewBag.Image1Url;
    var url2 = ViewBag.Image2Url;
    var url3 = ViewBag.Image3Url;
    var url4 = ViewBag.Image4Url;
    var url5 = ViewBag.Image5Url;

    // Отримуємо розміри з БД
    var w1 = ViewBag.Image1Width;
    var h1 = ViewBag.Image1Height;
    var w2 = ViewBag.Image2Width;
    var h2 = ViewBag.Image2Height;
    var w3 = ViewBag.Image3Width;
    var h3 = ViewBag.Image3Height;
    var w4 = ViewBag.Image4Width;
    var h4 = ViewBag.Image4Height;
    var w5 = ViewBag.Image5Width;
    var h5 = ViewBag.Image5Height;
}

<!-- Обгортка контейнера з відступами -->
<div class="container my-4">
    <h2 class="mb-3">Відображення п'яти зображень (Топологія варіант №12)</h2>
    <!-- Опис -->
    <p class="mb-4">
        Посилання (URL) отримуємо з форми,<br />
        розміри (width, height) зчитуються з БД (таблиця ImageDimensions).
    </p>

    <!-- Контейнер для зображень з nowrap для відображення в одному рядку -->
    <div class="border p-3" style="white-space: nowrap; text-align: left;">
        <!-- 1 зображення -->
        
        <!-- 2 зображення -->
        
        <!-- 3 зображення -->
```

```

        
        <!-- 4 зображення -->
        
        <!-- 5 зображення -->
        
    </div>

    <br />
    <!-- Кнопка повернення на головну сторінку -->
    <a asp-action="Index" asp-controller="Home" class="btn btn-secondary">Назад</a>
</div>

```

## \_Layout.cshtml:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Task_2_1</title>
    <!-- Підключення Bootstrap CSS -->
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <!-- Підключення власних стилів -->
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link rel="stylesheet" href="~/Task_2_1.styles.css" asp-append-version="true" />
</head>
<body style="background:
url('https://wallpaper.forfun.com/fetch/98/98cee4756de1ddb07d58638a36e45c22.jpeg') no-repeat
center center fixed; background-size: cover;">
    <header>
        <!-- Навігаційна панель (Navbar) -->
        <nav class="navbar navbar-expand-sm navbar-light bg-white border-bottom box-shadow mb-
3">
            <div class="container-fluid">
                <!-- Назва сайту -->
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-
action="Index">Task_2_1</a>
                <!-- Кнопка для мобільного меню -->
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target=".navbar-collapse"

```



```

        aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <!-- Меню навігації -->
    <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
        <ul class="navbar-nav flex-grow-1">
            <li class="nav-item">

                <a class="nav-link text-dark" asp-area="" asp-controller="Home"
asp-action="Index">Home</a>
            </li>
            <li class="nav-item">

                <a class="nav-link text-dark" asp-area="" asp-controller="Home"
asp-action="Privacy">Privacy</a>
            </li>
        </ul>
    </div>
</nav>
</header>

<!-- Основний контейнер для вмісту -->
<div class="container my-5" style="background-color: rgba(255,255,255,0.9); border-radius:
10px; padding: 20px;">
    <main role="main" class="pb-3">
        @RenderBody() <!-- Рендер основний вміст сторінки -->
    </main>
</div>

<!-- Футер -->
<footer class="border-top footer text-white mt-4" style="background-color: rgba(0, 0, 0,
0.5);">
    <div class="container">
        &copy; 2025 - Task_2_1 -

        <a asp-area="" asp-controller="Home" asp-action="Privacy" class="text-
white">Privacy</a>
    </div>
</footer>

<!-- Підключення скриптів -->
<script src="~/lib/jquery/dist/jquery.min.js"></script>

```

```
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>
@await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

## **\_Layout.cshtml.css:**

```
/* Please see documentation at https://learn.microsoft.com/aspnet/core/client-side/bundling-
and-minification
for details on configuring this project to bundle and minify static web assets. */
```

```
a.navbar-brand {
    white-space: normal;
    text-align: center;
    word-break: break-all;
}

/* Загальний стиль для посилань */
a {
    color: #0077cc;
    transition: color 0.3s ease; /* Плавна зміна */
}

/* Стиль для кнопок primary */
.btn-primary {
    color: #fff;
    background-color: #1b6ec2;
    border-color: #1861ac;
    transition: background-color 0.3s, border-color 0.3s;
}

/* Стиль для активних посилань у nav-pills */
.nav-pills .nav-link.active, .nav-pills .show > .nav-link {
    color: #fff;
    background-color: #1b6ec2;
    border-color: #1861ac;
}

/* Стиль для верхньої межі елементів */
.border-top {
    border-top: 1px solid #e5e5e5;
}
```

```

/* Стиль для нижньої межі елементів */
.border-bottom {
    border-bottom: 1px solid #e5e5e5;
}

/* Легка тінь для створення об'ємного ефекту */
.box-shadow {
    box-shadow: 0 .25rem .75rem rgba(0, 0, 0, 0.05);
}

/* Стиль для кнопки прийняття політики */
button.accept-policy {
    font-size: 1rem;
    line-height: inherit;
    cursor: pointer;
}

/* Стиль для футера */
.footer {
    position: absolute;
    bottom: 0;
    width: 100%;
    white-space: nowrap;
    line-height: 60px;
}

```

## 5. SQL-скрипт:

```

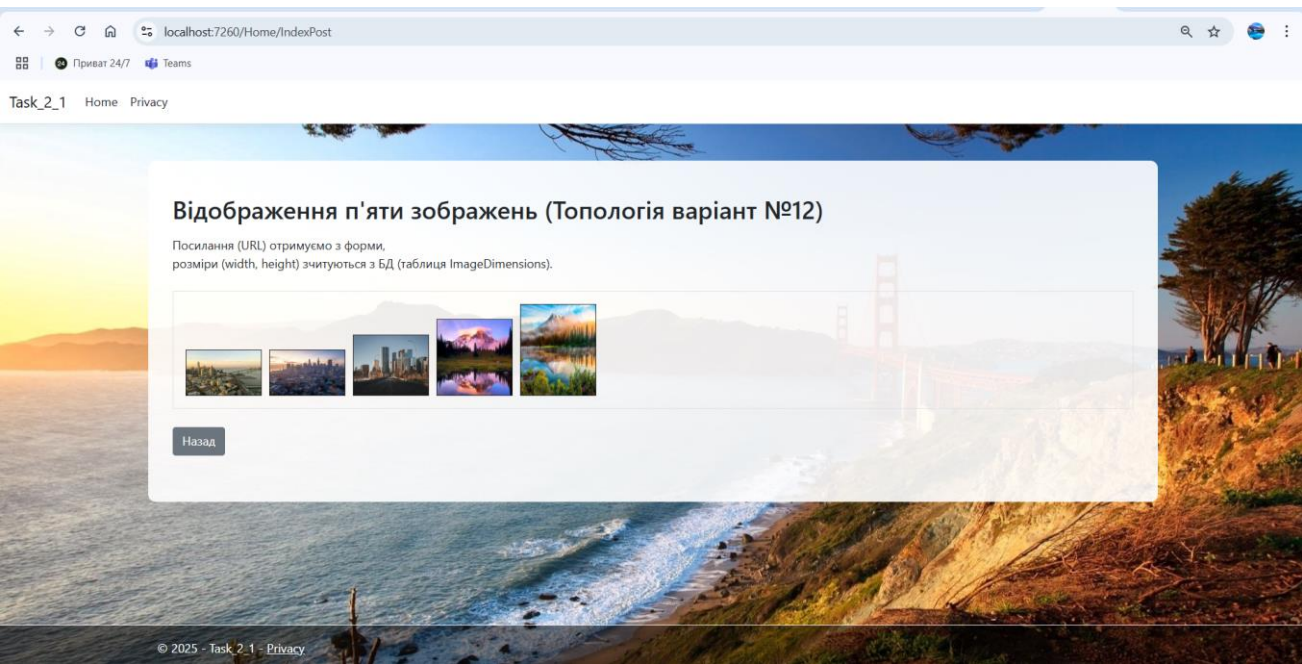
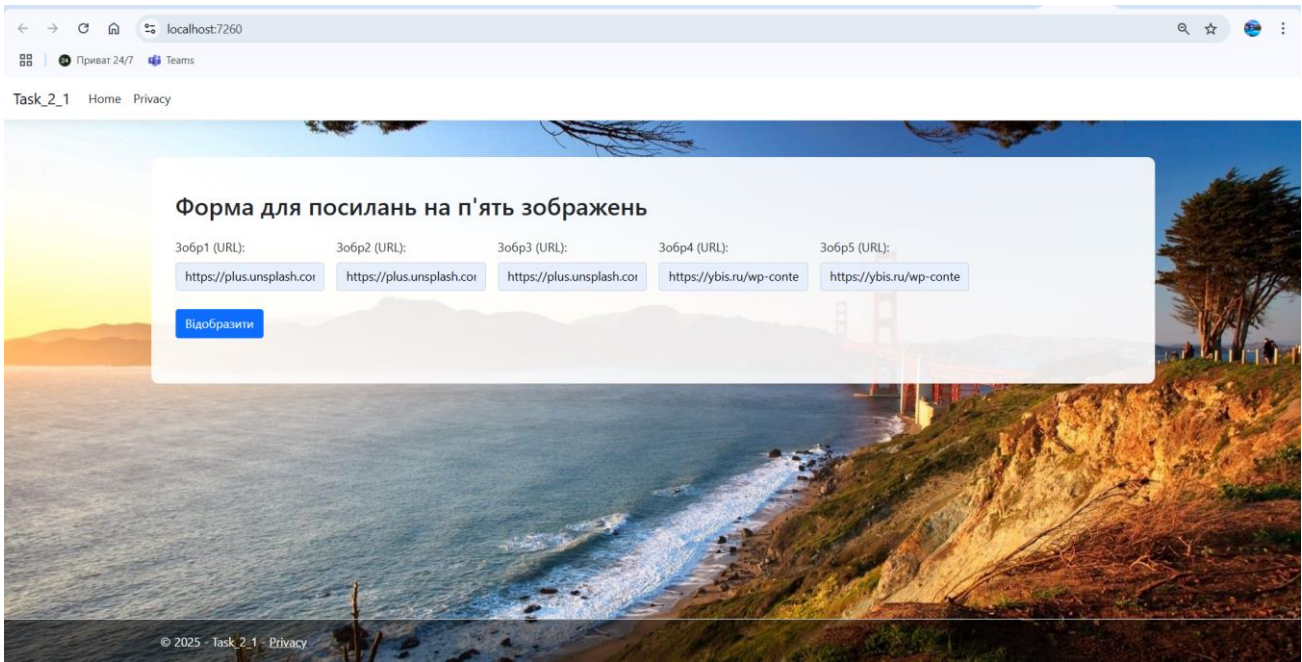
-- Створення таблиці:
IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE [name] = 'ImageDimensions')
BEGIN
    CREATE TABLE [dbo].[ImageDimensions](
        [Id] INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
        [Width] NVARCHAR(50) NOT NULL,
        [Height] NVARCHAR(50) NOT NULL
    );
END

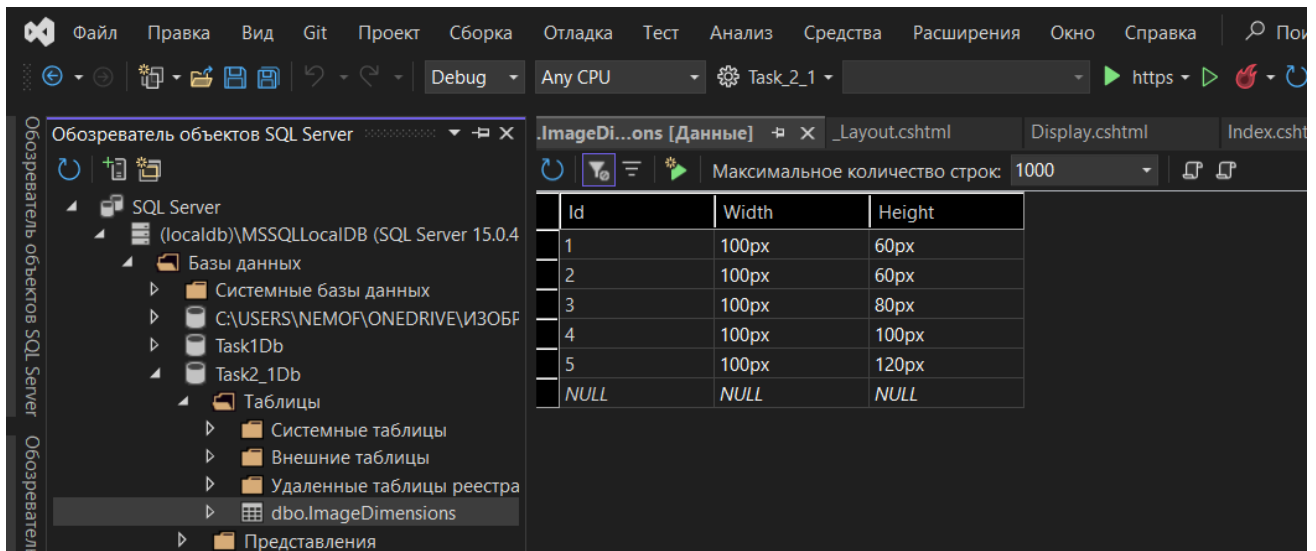
-- Вставлення кількох записів:
INSERT INTO [dbo].[ImageDimensions] (Width, Height)
VALUES
    ('80px', '60px'),
    ('100px', '70px'),
    ('120px', '120px'),

```

('140px', '150px'),  
('160px', '220px');

Результат роботи програми:





## Алгоритм побудови проєкту і кодів

### 1. Створення проєкту:

- Додав новий проєкт типу ASP.NET Core MVC з назвою *Task\_2\_1*.

### 2. Налаштування бази даних (LocalDB):

- Створив БД LocalDB (внутрішню базу Visual Studio).
- Створив таблицю **ImageDimensions** для зберігання розмірів зображень (поля: Id, Width, Height).
- Додав SQL-скрипт для створення таблиці.

### 3. Розробка моделі і контексту:

- **ImageDimension.cs** (у папці *Models*): описує властивості зображень (Width, Height).
- **ApplicationDbContext.cs** (у папці *Data*): містить `DbSet<ImageDimension>`, що відповідає таблиці ImageDimensions.

### 4. Розробка контролера:

- **HomeController.cs** (у папці *Controllers*):
  - **Index() [GET]** – повертає головну сторінку з формою (нетипізована форма) для введення URL 5 зображень.
  - **IndexPost() [POST]** – зчитує URL-адреси з форми, отримує з БД розміри (Width, Height) для кожного зображення, передає дані через ViewBag у подання Display.
  - **Display()** – повертає подання, яке відображає зображення за заданою топологією.

## 5. Розробка подань (Views):

- **Index.cshtml** – головне подання з формою, де користувач вводить URL п'яти зображень. Форма надсилає дані методом POST у метод IndexPost контролера Home.
- **Display.cshtml** – подання, яке приймає з ViewBag URL зображень і їхні розміри, відображає зображення в один рядок за топологією «сходинок». Використовуються стилі (inline CSS) для white-space: nowrap;, vertical-align: bottom; тощо.
- **\_Layout.cshtml** – шаблон із підключенням Bootstrap для оформлення, Navbar, футер, фон тощо.

## 6. Тестування:

- Запустив застосунок, ввів URL зображень у формі.
- Перевішив, чи коректно отримуються розміри зображень з таблиці ImageDimensions і відображаються в поданні Display за заданою топологією (сходинок).
- Змінив ширину/висоту в таблиці ImageDimensions, аби перевірити, чи застосунок правильно відображає нові розміри.

## Функціональність

- **Введення даних:** На головній сторінці (Index.cshtml) користувач задає URL для п'яти зображень.
- **Обробка даних:** Контролер HomeController зчитує URL з форми, отримує розміри (Width, Height) з бази даних (ImageDimensions) і формує набір даних для кожного зображення.
- **Відображення:** Подання Display.cshtml відображає п'ять зображень в одному рядку, застосовуючи CSS (white-space: nowrap, margin-right тощо) для створення ефекту «сходинок».


## Висновок

Завдання Task\_2\_1 виконано згідно з варіантом №12. Реалізовано механізм зчитування URL зображень із форми та отримання їхніх розмірів із бази даних, а потім динамічне відображення зображень за топологією «сходинок». Оформлення здійснено за допомогою Bootstrap і власних CSS, що забезпечує сучасний та адаптивний вигляд застосунку. Структура проєкту логічна: моделі, контекст БД, контролер із двома

методами (для GET/POST), а також подання для введення даних та відображення результату.

## Завдання 2\_2

Модифікувати застосунок Завдання 2\_1 для реалізації ще однієї топології зображень.

№ Варіанту	Форма топології
1	2
12	

Посилання на зображення задаються на формі, розміри зображень з таблиці відповідної БД.

### Текст програми:

#### 1. Модель

**ImageDimension.cs** – Файл, що описує модель для зберігання розмірів зображень (Width, Height). Саме з цієї таблиці зчитуються розміри для відображення зображень у потрібній топології.

```
using System.ComponentModel.DataAnnotations;
```

```
namespace Task_2_2.Models
```

```
{
```

```
    public class ImageDimension
```

```
    {
```

```
        [Key]
```

```
        public int Id { get; set; }
```

```
        // Ширина зображення
```

```
        public string Width { get; set; } = string.Empty;
```

```
        // Висота зображення
```

```
        public string Height { get; set; } = string.Empty;
```

```
    }
```

```
}
```



## 2. Контекст бази даних

**ApplicationDbContext.cs** – Файл контексту, де визначено `DbSet<ImageDimension>` (таблиця `ImageDimensions`). Він відповідає за підключення до БД (`LocalDB`) та роботу з моделлю `ImageDimension`.

```
using Microsoft.EntityFrameworkCore;
using Task_2_2.Models;

namespace Task_2_2.Data
{
    // Контекст БД для Task_2_2
    public class ApplicationDbContext : DbContext
    {
        // Констр який приймає опції БД через DI
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options)
        {
        }

        // DbSet для таблиці ImageDimensions
        // Таблиця зберігає розміри зображень (Width, Height)
        public DbSet<ImageDimension> ImageDimensions { get; set; }
    }
}
```

## 3. Контролер

**HomeController.cs** – Файл контролера, який:

- Зчитує URL трьох зображень із форми (нетипізована форма).
- Звертається до бази даних (`ImageDimensions`) для отримання розмірів кожного зображення.
- Передає всі дані (URL, розміри) через `ViewBag` у подання `Display` для відображення.

```
using Microsoft.AspNetCore.Mvc;
using Task_2_2.Data;
using Task_2_2.Models;

namespace Task_2_2.Controllers
{
    public class HomeController : Controller
    {

```

```
// Змінна для роботи з БД через EF Core
private readonly ApplicationDbContext _context;

// Конструктор що отримує контекст через dependency injection
public HomeController(ApplicationDbContext context)
{
    _context = context;
}

// GET: /Home/Index
[HttpGet]
public IActionResult Index()
{
    return View();
}

// POST: /Home/IndexPost
[HttpPost]
public IActionResult IndexPost()
{
    // Зчитуємо URL зображень з форми (нетипізовано)
    string image1Url = Request.Form["image1Url"];
    string image2Url = Request.Form["image2Url"];
    string image3Url = Request.Form["image3Url"];

    // Отримуємо дані розмірів з БД для зображень з Id 1, 2 та 3
    var dim1 = _context.ImageDimensions.Find(1);
    var dim2 = _context.ImageDimensions.Find(2);
    var dim3 = _context.ImageDimensions.Find(3);

    // Передаємо URL зображень у ViewBag
    ViewBag.Image1Url = image1Url;
    ViewBag.Image2Url = image2Url;
    ViewBag.Image3Url = image3Url;

    // Передаємо розміри зображень у ViewBag
    ViewBag.Image1Width = dim1?.Width ?? "60px";
    ViewBag.Image1Height = dim1?.Height ?? "120px";

    ViewBag.Image2Width = dim2?.Width ?? "300px";
    ViewBag.Image2Height = dim2?.Height ?? "100px";

    ViewBag.Image3Width = dim3?.Width ?? "150px";
    ViewBag.Image3Height = dim3?.Height ?? "50px";
}
```

```

        // Перенаправляємо дані до подання Display
        return View("Display");
    }

    // Метод для відображення подання Display
    public IActionResult Display()
    {
        return View();
    }
}
}

```

#### 4. Подання (Views)

**Index.cshtml** – Головне подання з формою, де користувач вводить URL трьох зображень. Дані надсилаються методом POST у метод IndexPost контролера Home.

**Display.cshtml** – Подання, яке приймає з ViewBag URL та розміри зображень і відображає їх за новою топологією (наприклад, «сходинок»). Використовуються inline-стили (white-space, vertical-align тощо) для розміщення зображень у заданому порядку.

**\_Layout.cshtml** – Загальний шаблон застосунку, що забезпечує оформлення (Navbar, контейнер для вмісту, футер, фон). Підключає Bootstrap та власні стилі для гарного відображення сторінок.

**\_Layout.cshtml.css** – файл, який містить кастомні CSS-стилі для загального оформлення веб-застосунку (Navbar, контейнер, футер, фон тощо).

##### Index.cshtml:

```

@{
    ViewData["Title"] = "Головна (Task_2_2)";
}

<!-- Контейнер для форми із відступами -->
<div class="container my-4">
    <h2 class="mb-4">Форма для 3 зображень (Варіант із іншою топологією)</h2>
    <p class="mb-4">URL зображень задаються у формі, а розміри - з БД.</p>

    <!-- Форма що відправляє дані методом POST на IndexPost у HomeController -->
    <form method="post" asp-action="IndexPost" asp-controller="Home">
        <!-- Ряд із полями форми де використовується gutter -->
        <div class="row g-3">
            <!-- Зображення 1 -->

```

```

        <div class="col-md-4">
            <label class="form-label">Зобп1 (URL):</label>
            <input type="text" name="image1Url" class="form-control"
placeholder="https://site.com/img1.jpg" />
        </div>
        <!-- Зображення 2 -->
        <div class="col-md-4">
            <label class="form-label">Зобп2 (URL):</label>
            <input type="text" name="image2Url" class="form-control"
placeholder="https://site.com/img2.jpg" />
        </div>
        <!-- Зображення 3 -->
        <div class="col-md-4">
            <label class="form-label">Зобп3 (URL):</label>
            <input type="text" name="image3Url" class="form-control"
placeholder="https://site.com/img3.jpg" />
        </div>
    </div>

    <!-- Блок з кнопкою відправки форми з відступом зверху -->
    <div class="mt-4">
        <button type="submit" class="btn btn-primary">Показати</button>
    </div>
</form>
</div>

```

## Display.cshtml:

```

@{
    ViewData["Title"] = "Зображення за новою топологією (Сходинки)";

    // Зчитуємо URL зображень ViewBag
    var url1 = ViewBag.Image1Url;
    var url2 = ViewBag.Image2Url;
    var url3 = ViewBag.Image3Url;

    // Зчитуємо розміри з БД (ширина та висота)
    var w1 = ViewBag.Image1Width;
    var h1 = ViewBag.Image1Height;
    var w2 = ViewBag.Image2Width;
    var h2 = ViewBag.Image2Height;
    var w3 = ViewBag.Image3Width;
    var h3 = ViewBag.Image3Height;
}

```

```

<!-- Обгортка контейнера з відступами -->
<div class="container my-4">
    <h2 class="mb-3">Зображення “сходишками” (Варіант 12)</h2>
    <p class="mb-4">
        Зобp1, Зобp2, Зобp3 із різними розмірами. “Сходишки” виходять завдяки
        <code>white-space: nowrap;</code> та <code>vertical-align: bottom;</code>.
    </p>

    <!-- Контейнер для зображень із nowrap (не переносилися) -->
    <div class="border p-3" style="white-space: nowrap; text-align: left;">
        <!-- Відображення 1 Зображення -->
        
        <!-- Відображення 2 Зображення -->
        
        <!-- Відображення 3 Зображення -->
        
    </div>

    <!-- Кнопка повернення на головну сторінку -->
    <div class="mt-3">
        <a asp-action="Index" asp-controller="Home" class="btn btn-secondary">Назад</a>
    </div>
</div>

```

## \_Layout.cshtml:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>@ViewData["Title"] - Task_2_2</title>
    <!-- Підключення Bootstrap CSS для стандартного оформлення -->
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <!-- Підключення власних стилів -->
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link rel="stylesheet" href="~/Task_2_2.styles.css" asp-append-version="true" />
</head>

```

```

<body style="background: url('https://eskipaper.com/images/cityscape-wallpaper-6.jpg') no-repeat center center fixed; background-size: cover;">
  <header>
    <!-- Навігаційна панель з використанням Bootstrap -->
    <nav class="navbar navbar-expand-sm navbar-light bg-white border-bottom box-shadow mb-3">
      <div class="container-fluid">
        <!-- Назва сайту де посилання веде на головну сторінку -->
        <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">Task_2_2</a>
        <!-- Кнопка для мобільного меню -->
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse"
          aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <!-- Меню навігації -->
        <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
          <ul class="navbar-nav flex-grow-1">
            <li class="nav-item">
              <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
            </li>
            <li class="nav-item">
              <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
            </li>
          </ul>
        </div>
      </div>
    </nav>
  </header>

  <!-- Основний контейнер для відображення вмісту -->
  <div class="container my-5" style="background-color: rgba(255,255,255,0.9); border-radius: 10px; padding: 20px;">
    <main role="main" class="pb-3">
      @RenderBody() <!-- Вміст який задається у поданнях -->
    </main>
  </div>

```

```

<!-- Футер -->
<footer class="border-top footer text-white mt-4" style="background-color: rgba(0, 0, 0,
0.5);">
    <div class="container">
        &copy; 2025 - Task_2_2 -

        <a asp-area="" asp-controller="Home" asp-action="Privacy" class="text-
white">Privacy</a>
    </div>
</footer>

<!-- Підключення jQuery та Bootstrap JS -->
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>
@await RenderSectionAsync("Scripts", required: false)
</body>
</html>

```

## Layout.cshtml.css:

```

/* Please see documentation at https://learn.microsoft.com/aspnet/core/client-side/bundling-
and-minification
for details on configuring this project to bundle and minify static web assets. */

```

```

/* Стиль для логотипу в navbar */

```

```

a.navbar-brand {
    white-space: normal;
    text-align: center;
    word-break: break-all; /
}

```

```

/* Загальний стиль для посилань */

```

```

a {
    color: #0077cc;
    transition: color 0.3s ease;
}

```

```

/* Стилiзацiя кнопок типу primary */

```

```

.btn-primary {
    color: #fff;
    background-color: #1b6ec2;
    border-color: #1861ac;
    transition: background-color 0.3s, border-color 0.3s;
}

```

```
}
```

```
/* Активні елементи навігації (nav-pills) */
```

```
.nav-pills .nav-link.active, .nav-pills .show > .nav-link {  
    color: #fff;  
    background-color: #1b6ec2;  
    border-color: #1861ac;  
}
```

```
/* Стиль для верхньої межі */
```

```
.border-top {  
    border-top: 1px solid #e5e5e5;  
}
```

```
/* Стиль для нижньої межі */
```

```
.border-bottom {  
    border-bottom: 1px solid #e5e5e5;  
}
```

```
/* Додаємо легку тінь для об'ємного ефекту */
```

```
.box-shadow {  
    box-shadow: 0 .25rem .75rem rgba(0, 0, 0, 0.05);  
}
```

```
/* Стиль для кнопки прийняття політики */
```

```
button.accept-policy {  
    font-size: 1rem;  
    line-height: inherit;  
    cursor: pointer;  
}
```

```
/* Стиль для футера */
```

```
.footer {  
    position: absolute;  
    bottom: 0;  
    width: 100%;  
    white-space: nowrap;  
    line-height: 60px;  
}
```

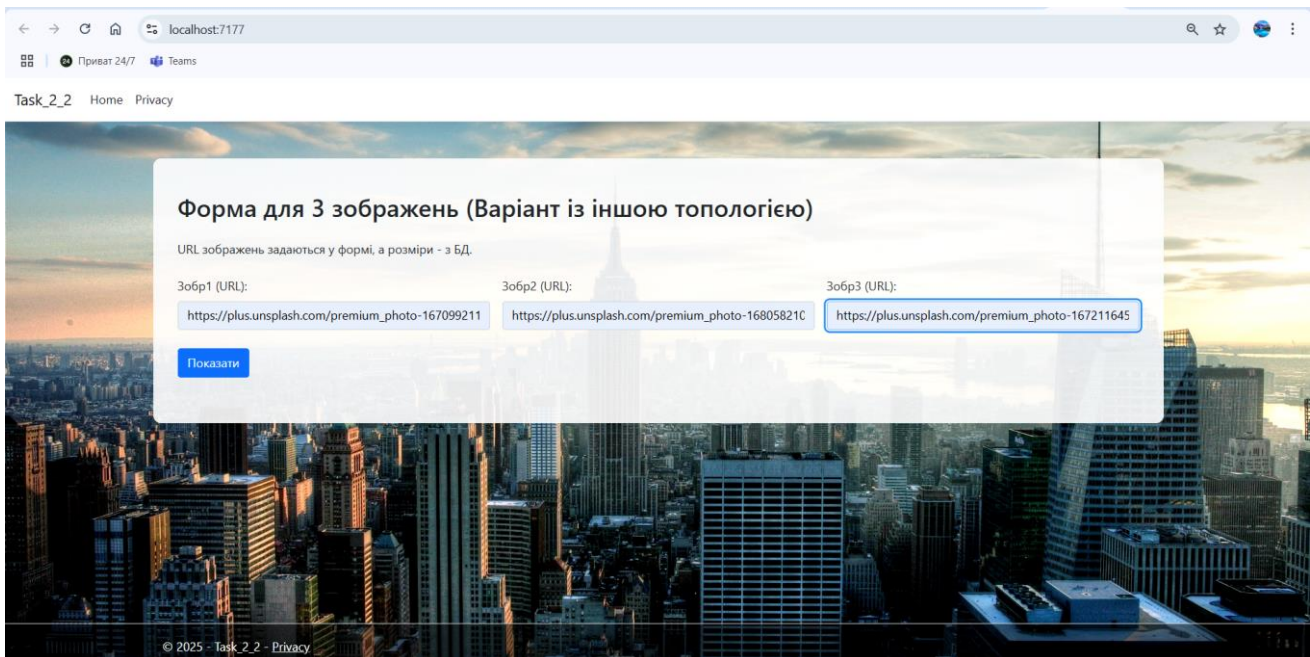


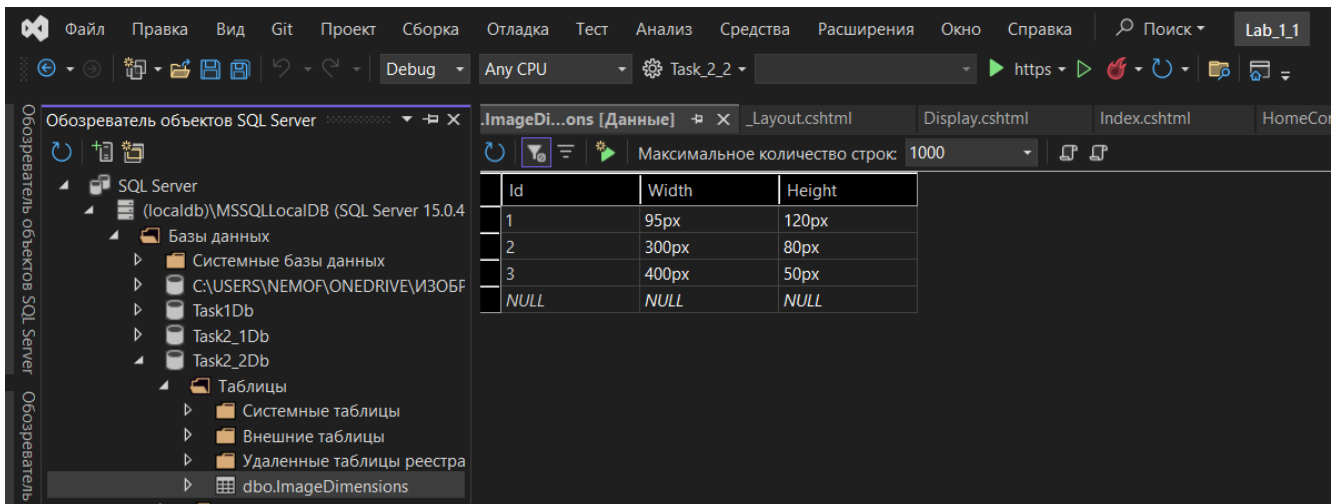
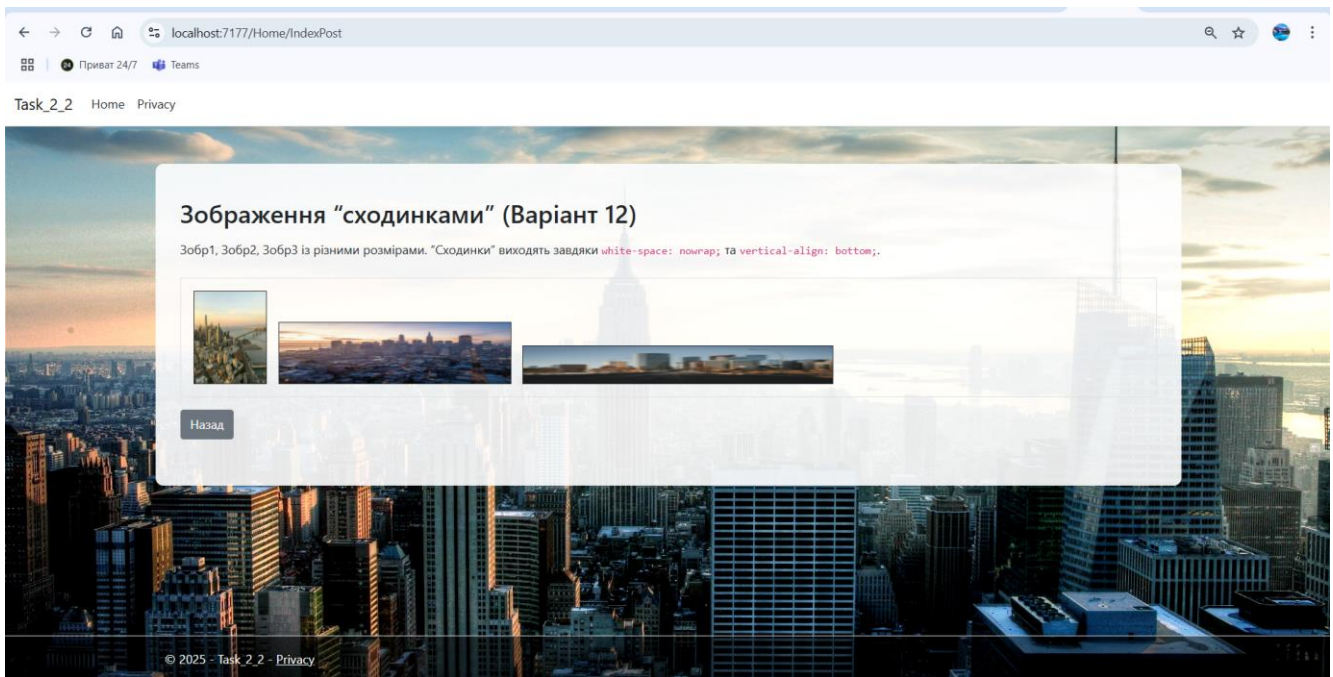
## 5. SQL-скрипт:

```
-- Створення таблиці ImageDimensions
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'ImageDimensions')
BEGIN
    CREATE TABLE [dbo].[ImageDimensions](
        [Id] INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
        [Width] NVARCHAR(50) NOT NULL,
        [Height] NVARCHAR(50) NOT NULL
    );
END

-- Вставлення 3 записів для нашої топології (зобр1, зобр2, зобр3):
INSERT INTO [dbo].[ImageDimensions] (Width, Height)
VALUES
('60px', '120px'),    -- Зобр1: вертикально-втягнутий
('300px', '100px'),   -- Зобр2: широкий
('150px', '50px');    -- Зобр3: горизонтальний прямокутник
```

## Результат роботи програми:





## Алгоритм побудови проєкту і кодів

### 1. Створення проєкту (на основі Task\_2\_1):

- Використав існуюче рішення (*Lab\_1*).
- Зробив копію проєкту Task\_2\_1 або створив новий проєкт *Task\_2\_2* для модифікації попереднього коду.

### 2. Налаштування БД (LocalDB):

- Використовується та ж таблиця **ImageDimensions** (поля: Id, Width, Height).
- Зберігаємо розміри зображень (ширину, висоту) для нової топології.

### 3. Модифікація контролера:

- **HomeController.cs:**
  - Залишив методи Index() (GET) та IndexPost() (POST), але тепер логіка відображення/розмірів зображень змінена для нової топології.

- При зчитуванні з форми (URL зображень) звертаємося до таблиці ImageDimensions, отримуємо розміри і передаємо через ViewBag до подання Display.
- Топологія зображень формується у поданні Display через inline-стилі.

#### 4. Модифікація подань (Views):

##### ○ **Index.cshtml:**

- Залишився файл із формою для введення URL зображень, але, наприклад, замість 5 зображень тепер 3 (або навпаки) залежно від вимог нової топології.
- Надсилає дані методом POST до IndexPost() контролера.

##### ○ **Display.cshtml:**

- Змінено CSS-властивості (inline-стилі) для відображення зображень згідно з новою топологією (наприклад, зображення розташовані «по-сходинках», але в іншому порядку, або вертикально, тощо).
- Використовується white-space: nowrap; та vertical-align: bottom;, відступи (margin), чи інші стилі, щоб створити потрібну геометрію.

##### ○ **\_Layout.cshtml:**

- Залишається спільним шаблоном із використанням Bootstrap. Змінено фон, Navbar та футер, але загальна структура така ж, як і в Task\_2\_1.

#### 5. Тестування:

- Запустив застосунок, ввів нові URL зображень.
- Перевірив, що розміри зображень (з таблиці ImageDimensions) правильно застосовуються.
- Переконався, що топологія зображень відображається за новою схемою.

#### Функціональність

- **Введення даних:** На сторінці Index.cshtml користувач задає URL кількох зображень (наприклад, трьох).
- **Отримання розмірів:** Контролер HomeController зчитує ці URL, звертається до бази даних (ImageDimensions) для отримання потрібних розмірів (Width, Height) і передає все у ViewBag.
- **Відображення:** У поданні Display.cshtml зображення відображаються за зміненою топологією (інший порядок, розташування «по-сходинках»).

Застосовуються inline-стилі для позиціювання, `white-space: nowrap;`, `vertical-align: bottom;`, відступи (`margin`) і т. д.

- **Оформлення:** Загальний шаблон (`_Layout.cshtml`) із Bootstrap забезпечує адаптивне та привабливе оформлення сторінок.

## **Висновок**

Завдання Task\_2\_2 успішно модифікує попередній застосунок (Task\_2\_1) для реалізації нової топології зображень. Логіка зчитування URL з форми і розмірів з бази даних залишилася подібною, однак змінилися CSS-властивості в поданні `Display.cshtml`. Оформлення залишилося на Bootstrap, що забезпечує гарний вигляд і адаптивність застосунку.

### Завдання 3

Створити MVC WEB-застосунок, головне подання якого містить нетипізовану форму для задання групи параметрів, які через контролер, що викликається з цієї ж форми, передаються до іншого подання, на якому формуються шахова (без об'єднаних комірок) таблиця із вказаними розмірами. Парні номери варіантів кількість рядків задає на формі, а кількість стовпчиків з БД, непарні номери - навпаки. Комірки таблиці послідовно заповнюються цілими числами, починаючи з 1, тобто 1, 2, 3 і т. д. Під час формування таблиці використовуються рядкові змінні в контролері, а не цикли технології Razor.

#### **Текст програми:**

##### **1. Модель**

**ColumnCount.cs** – Файл, що описує модель для зберігання кількості стовпчиків (Columns). Ця таблиця зберігає інформацію про кількість стовпчиків для шахової таблиці.

```
using System.ComponentModel.DataAnnotations;

namespace Task_3.Models
{
    // Модель для зберігання кількості стовпчиків у таблиці
    public class ColumnCount
    {
        [Key]
        public int Id { get; set; }

        // Зберігання кількості стовпчиків
        public int Columns { get; set; }
    }
}
```

##### **2. Контекст бази даних**

**ApplicationDbContext.cs** – Файл, у якому визначено `DbSet<ColumnCount>`, що відповідає таблиці `ColumnCounts`. Він відповідає за підключення до БД (LocalDB) та роботу з моделлю `ColumnCount` (для зчитування кількості стовпчиків).

```
using Microsoft.EntityFrameworkCore;
using Task_3.Models;

namespace Task_3.Data
```

```

{
    // Контекст БД для Task_3
    public class ApplicationDbContext : DbContext
    {
        // Конструктор який приймає опції через DI та передає їх у базовий клас
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options)
        {
        }

        // DbSet для зберігання кількості стовпчиків (табл ColumnCounts)
        public DbSet<ColumnCount> ColumnCounts { get; set; }
    }
}

```

### 3. Контролер

**HomeController.cs** – Файл контролера, що:

- Зчитує кількість рядків із форми (нетипізована форма, Index.cshtml).
- Звертається до БД (ColumnCounts) для отримання кількості стовпчиків.
- Генерує HTML-код шахової таблиці (без Razor-циклів) за допомогою рядкових змінних (StringBuilder) і послідовно заповнює комірки числами від 1, 2, 3 і т.д.
- Передає згенерований HTML-код таблиці у ViewBag, а також кількість рядків і стовпчиків у подання Display.

```

using Microsoft.AspNetCore.Mvc;
using Task_3.Data;
using Task_3.Models;
using System.Text;

namespace Task_3.Controllers
{
    public class HomeController : Controller
    {
        // Контекст БД, отриманий через dependency injection
        private readonly ApplicationDbContext _context;

        // Конструктор отримує контекст і зберігає його в приватній змінній
        public HomeController(ApplicationDbContext context)
        {
            _context = context;
        }
    }
}

```

```

}

// GET: Home/Index
[HttpGet]
public IActionResult Index()
{
    return View();
}

// POST: Home/IndexPost
// Метод для обробки даних введених у формі
[HttpPost]
public IActionResult IndexPost()
{
    // 1. Зчитуємо кількість рядків із форми
    string rowStr = Request.Form["rowCount"];
    int rowCount = 0;
    int.TryParse(rowStr, out rowCount);

    // 2. Отримуємо кіл стовпчиків з БД (запис з Id=1)
    ColumnCount colData = _context.ColumnCounts.Find(1);
    int colCount = colData?.Columns ?? 5; // Якщо немає даних, встановлюємо 5
    стовпчиків за замовчуванням

    // 3. Генер HTML-код шахової таблиці без Razor-циклів
    StringBuilder sb = new StringBuilder();
    sb.Append("<table style='border-collapse: collapse;'>");

    // Нумерація комірок
    int cellNumber = 1;

    // Проходимо по кожному рядку
    for (int r = 0; r < rowCount; r++)
    {
        sb.Append("<tr>");
        // Для кожного рядка проходимо по стовпчиках
        for (int c = 0; c < colCount; c++)
        {
            // Визначаємо колір комірки (парне - біла; непарне - сіре)
            string bgColor = ((r + c) % 2 == 0) ? "#ffffff" : "#cccccc";

            // HTML для комірки із заданими стилями і нумерацією
            sb.Append($"<td style='width:50px; height:50px; border:1px solid #000;
background-color:{bgColor}; text-align:center;'>");

```

```

        sb.Append(cellNumber);
        sb.Append("</td>");
        cellNumber++;
    }
    sb.Append("</tr>");
}
sb.Append("</table>");

// Передаємо згенерований HTML-код таблиці у ViewBag
ViewBag.TableHtml = sb.ToString();
// Передаємо кількість рядків та стовпчиків
ViewBag.Rows = rowCount;
ViewBag.Cols = colCount;

// Повертаємо подання Display
return View("Display");
}

// Метод для відображення подання з табл
public IActionResult Display()
{
    return View();
}
}
}

```

#### 4. Подання (Views)

**Index.cshtml** – Головне подання з формою, де користувач вводить кількість рядків (для парного варіанту). Дані надсилаються методом POST до IndexPost() контролера Home.

**Display.cshtml** – Подання, що отримує згенерований HTML-код шахової таблиці (через ViewBag) і виводить його за допомогою @Html.Raw(...). Також відображає інформацію про кількість рядків і стовпчиків.

**\_Layout.cshtml** – Загальний шаблон застосунку, що забезпечує оформлення (Navbar, контейнер для основного вмісту, футер, фон). Застосовує Bootstrap для адаптивного дизайну.

**\_Layout.cshtml.css** – файл, який містить кастомні CSS-стилі для загального оформлення веб-застосунку (Navbar, контейнер, футер, фон тощо).



## Index.cshtml:

```
@{
    ViewData["Title"] = "Головна (Завдання 3)";
}

<!-- Основний контейнер -->
<div class="container my-4">
    <!-- Заголовок форми -->
    <h2 class="mb-3">Форма: Кількість рядків (парний варіант)</h2>

    <p class="mb-4">Кількість стовпчиків буде взята з БД (таблиця ColumnCounts).</p>

    <!-- Форма для введення кількості рядків де дані відправляються методом POST -->
    <form method="post" asp-action="IndexPost" asp-controller="Home">
        <!-- Поле введення кількості рядків -->
        <div class="mb-3">
            <label class="form-label">Введіть кількість рядків:</label>
            <input type="number" name="rowCount" class="form-control" placeholder="наприклад 5"
        />
        </div>
        <!-- Кнопка для побудови таблиці -->
        <button type="submit" class="btn btn-primary">Побудувати таблицю</button>
    </form>
</div>
```

## Display.cshtml:

```
@{
    ViewData["Title"] = "Результат (Display)";
}

<!-- Контейнер для вмісту -->
<div class="container my-4">
    <h2 class="mb-3">Шахова таблиця</h2>
    <p class="mb-4">
        Кількість рядків (з форми): @ViewBag.Rows<br />
        Кількість стовпчиків (з БД): @ViewBag.Cols
    </p>

    <!-- Виводимо HTML-код таблиці згенерований у контролері -->
    <div class="mb-4">
        @Html.Raw(ViewBag.TableHtml)
    </div>
```

```
<!-- Кнопка повернення на головну сторінку -->
<a asp-action="Index" asp-controller="Home" class="btn btn-secondary">Назад</a>
</div>
```

## **\_Layout.cshtml:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Task_3</title>
    <!-- Підключення Bootstrap CSS -->
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <!-- Підключення власних стилів -->
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link rel="stylesheet" href="~/Task_3.styles.css" asp-append-version="true" />
</head>
<body style="background: url('https://images6.alphacoders.com/126/1261894.jpg') no-repeat
center center fixed; background-size: cover;">
    <header>
        <!-- Навігаційна панель (Navbar) з використанням Bootstrap -->
        <nav class="navbar navbar-expand-sm navbar-light bg-white border-bottom box-shadow mb-
3">
            <div class="container-fluid">
                <!-- Назва сайту де посилання веде на головну сторінку -->
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-
action="Index">Task_3</a>

                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target=".navbar-collapse"
                    aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <!-- Меню навігації -->
                <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
                    <ul class="navbar-nav flex-grow-1">
                        <li class="nav-item">

                            <a class="nav-link text-dark" asp-area="" asp-controller="Home"
asp-action="Index">Home</a>
                        </li>
                        <li class="nav-item">
```

```

        <a class="nav-link text-dark" asp-area="" asp-controller="Home"
asp-action="Privacy">Privacy</a>
    </li>
</ul>
</div>
</div>
</nav>
</header>

<!-- Основний контейнер для вмісту -->
<div class="container my-5" style="background-color: rgba(255,255,255,0.9); border-radius:
10px; padding: 20px;">
    <main role="main" class="pb-3">
        @RenderBody() <!-- Тут рендер основного контенту сторінки -->
    </main>
</div>

<!-- Футер -->
<footer class="border-top footer mt-4" style="background-color: rgba(0,0,0,0.5); color:
#FFD700;">
    <div class="container">
        &copy; 2025 - Task_3 -

        <a asp-area="" asp-controller="Home" asp-action="Privacy" style="color:
#FFD700;">Privacy</a>
    </div>
</footer>

<!-- Підключення jQuery, Bootstrap JS та власних скриптів -->
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>
@await RenderSectionAsync("Scripts", required: false)
</body>
</html>

```

## Layout.cshtml.css:

```

/* Please see documentation at https://learn.microsoft.com/aspnet/core/client-side/bundling-
and-minification
for details on configuring this project to bundle and minify static web assets. */

/* Стиль для логотипу в Navbar */

```

```
a.navbar-brand {
    white-space: normal;
    text-align: center;
    word-break: break-all;
}

/* Загальний стиль для всіх посилань */
a {
    color: #0077cc;
}

/* Стилізація кнопок типу primary */
.btn-primary {
    color: #fff;
    background-color: #1b6ec2;
    border-color: #1861ac;
}

/* Активні елементи в nav-pills */
.nav-pills .nav-link.active, .nav-pills .show > .nav-link {
    color: #fff;
    background-color: #1b6ec2;
    border-color: #1861ac;
}

/* Стиль для верхньої межі */
.border-top {
    border-top: 1px solid #e5e5e5;
}

/* Стиль для нижньої межі */
.border-bottom {
    border-bottom: 1px solid #e5e5e5;
}

/* Додаємо легку тінь для створення об'ємного ефекту */
.box-shadow {
    box-shadow: 0 .25rem .75rem rgba(0, 0, 0, 0.05);
}

/* Стиль для кнопки прийняття політики */
button.accept-policy {
    font-size: 1rem;
    line-height: inherit;
```

```

        cursor: pointer;
    }

/* Стиль для футера */
.footer {
    position: absolute;
    bottom: 0;
    width: 100%;
    white-space: nowrap;
    line-height: 40px;
}

```

## 5. SQL-скрипт:

```

IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'ColumnCounts')
BEGIN
    CREATE TABLE [dbo].[ColumnCounts](
        [Id] INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
        [Columns] INT NOT NULL
    );
END

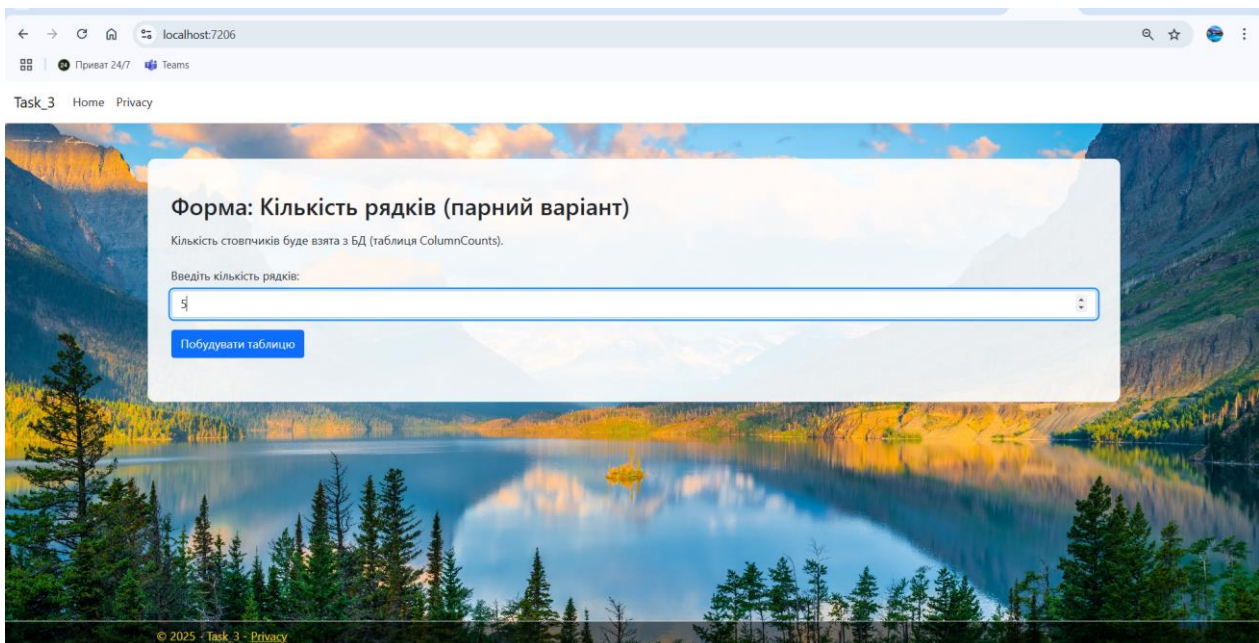
```

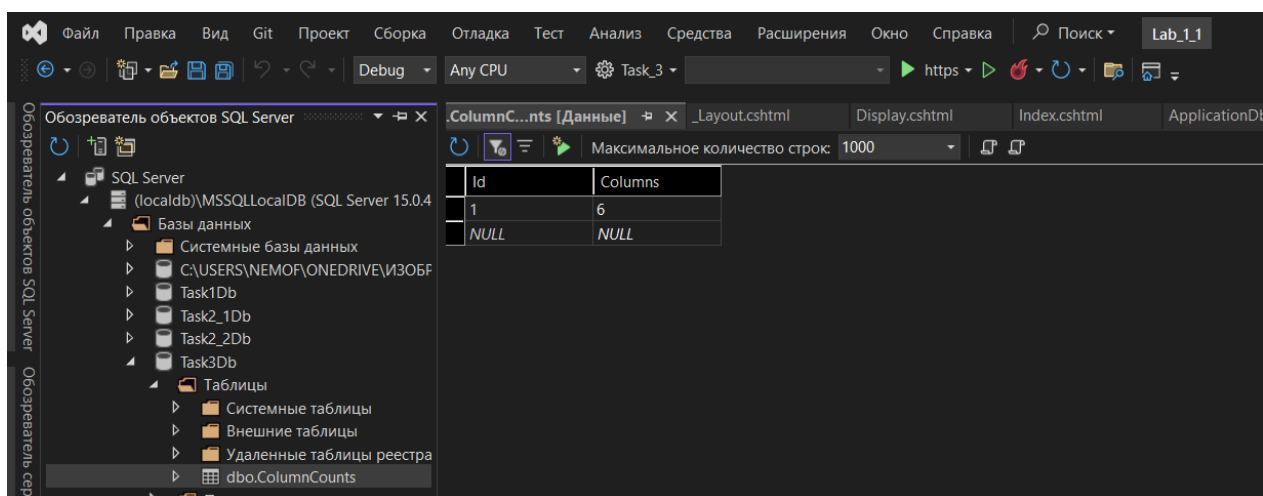
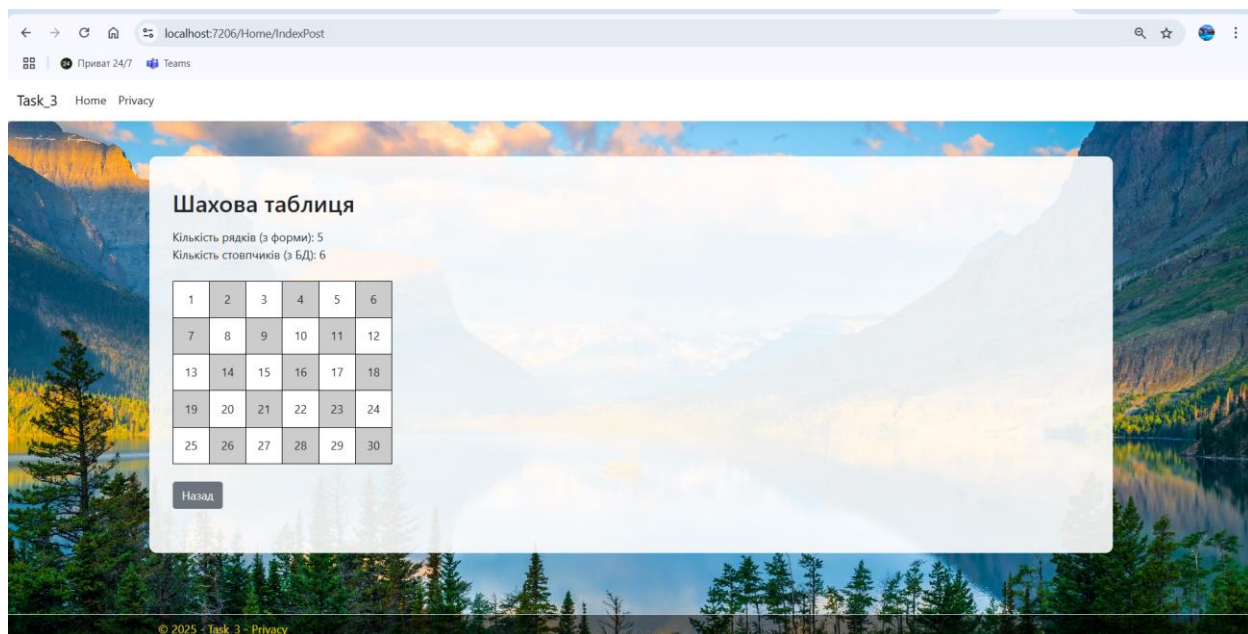
```

-- Вставляємо 1 запис, наприклад: Columns=6
INSERT INTO [dbo].[ColumnCounts] ([Columns]) VALUES (6);

```

## Результат роботи програми:





## Алгоритм побудови проєкту і кодів

### 1. Створення проєкту:

- У Visual Studio додав до існуючого рішення новий проєкт типу ASP.NET Core MVC з назвою *Task\_3*.

### 2. Налаштування БД (LocalDB):

- Створив таблицю **ColumnCounts**, яка містить поля:
  - Id (ціле число, автоінкремент, первинний ключ),
  - Columns (int) – кількість стовпчиків.
- Додав SQL-скрипт для створення таблиці до проєкту.

### 3. Розробка моделі і контексту:

- ColumnCount.cs** (у папці *Models*): описує властивість Columns для зберігання кількості стовпчиків.
- ApplicationDbContext.cs** (у папці *Data*): містить DbSet<ColumnCount> для таблиці ColumnCounts, що забезпечує доступ до бази даних через EF Core.

#### 4. Розробка контролера (HomeController.cs):

- **Index() [GET]:** повертає форму (нетипізована), де користувач вводить кількість рядків (оскільки в парному варіанті кількість рядків задається на формі).
- **IndexPost() [POST]:**
  1. Зчитує кількість рядків із форми.
  2. Звертається до таблиці ColumnCounts (наприклад, бере запис із Id=1), щоб отримати кількість стовпчиків.
  3. Генерує HTML-код шахової таблиці за допомогою рядкових змінних (StringBuilder) без Razor-циклів. Заповнює комірки числами, починаючи з 1, 2, 3 тощо.
  4. Передає згенерований HTML (через ViewBag) у подання Display, а також передає кількість рядків і стовпчиків для відображення.
- **Display():** повертає подання Display.cshtml, де виводиться згенерований HTML-код.

#### 5. Розробка подань (Views):

- **Index.cshtml:**
  - Містить форму з одним полем rowCount (для введення кількості рядків).
  - Надсилає дані методом POST до методу IndexPost() контролера.
- **Display.cshtml:**
  - Приймає HTML-код таблиці з ViewBag.TableHtml і виводить його за допомогою @Html.Raw(...).
  - Відображає також кількість рядків і стовпчиків для наочності.
- **\_Layout.cshtml:**
  - Використовується як загальний шаблон. Містить підключення Bootstrap для оформлення, заголовков, навігацію, контейнер для основного вмісту та футер.

#### 6. Тестування:

- Запустив застосунок, ввів кількість рядків у формі.

- Перевішив, що застосунок правильно отримує кількість стовпчиків із БД, генерує HTML-код шахової таблиці, і відображає її з послідовним заповненням комірок (1, 2, 3, ...).
- Переконався, що шахова таблиця не має об'єднаних комірок, а колір фону може чергуватися (за потреби) або бути однаковим, залежно від реалізації.

### Функціональність

- **Введення даних:** Користувач вводить кількість рядків у формі (Index.cshtml).
- **Обробка даних:** Контролер (HomeController) зчитує цю кількість, звертається до бази даних для отримання кількості стовпчиків і генерує HTML-таблицю без Razor-циклів.
- **Відображення:** Подання Display.cshtml виводить готовий HTML-код шахової таблиці, а також показує кількість рядків і стовпчиків.

### Висновок

Завдання Task\_3 виконано відповідно до парного варіанту №12. Реалізовано форму для введення кількості рядків, зчитування кількості стовпчиків із БД, а також побудову шахової таблиці без об'єднаних комірок. Під час формування таблиці використано рядкові змінні в контролері (StringBuilder), а не Razor-цикли. Це відповідає вимогам завдання. Оформлення застосунку здійснено за допомогою Bootstrap.



## Завдання 4

Створити подання із переходами на інші подання через відповідні контролери, відповідно до схеми схемою власного варіанту.

№ Варіанту	Схема переходів
1	2
12	<pre>graph TD; S1[Стор1] --&gt; S3[Стор3]; S3 --&gt; S4[Стор4]; S4 --&gt; S5[Стор5]; S5 --&gt; S2[Стор2]; S2 --&gt; S1;</pre>

**Текст програми:**

### **1. Контролери:**

- **Page1Controller.cs** – контролер для Сторінки 1, який повертає подання, що містить кнопку переходу до Стор3.
- **Page2Controller.cs** – контролер для Сторінки 2, який відповідає за відображення Сторінки 2 та має кнопку для повернення на Стор1 (або інший напрямок, згідно з вашою схемою).
- **Page3Controller.cs** – контролер для Сторінки 3, що повертає подання з кнопкою переходу до Стор4.
- **Page4Controller.cs** – контролер для Сторінки 4, що повертає подання з кнопкою переходу до Стор5.
- **Page5Controller.cs** – контролер для Сторінки 5, що повертає подання з кнопкою переходу на Стор2.

### **Page1Controller.cs:**

```
using Microsoft.AspNetCore.Mvc;
```

```
namespace Task_4.Controllers
```

```
{
```

```

public class Page1Controller : Controller
{
    // GET: Page1/Index
    // Метод повертає подання для Сторінки 1
    public IActionResult Index()
    {
        // Повертаємо View для відображ Стор 1
        return View();
    }
}
}

```

### **Page2Controller.cs:**

```

using Microsoft.AspNetCore.Mvc;

namespace Task_4.Controllers
{
    public class Page2Controller : Controller
    {
        // GET: Page2/Index
        // Повертаємо подання для Стор 2
        public IActionResult Index()
        {
            // Повертаємо View для Стор 2
            return View();
        }
    }
}

```

### **Page3Controller.cs:**

```

using Microsoft.AspNetCore.Mvc;

namespace Task_4.Controllers
{
    public class Page3Controller : Controller
    {
        // GET: Page3/Index
        // Повертаємо подання для Стор 3
        public IActionResult Index()

```

```
{  
    // Повертаємо подання View для Стор 3  
    return View();  
}  
}  
}
```

### **Page4Controller.cs:**

```
using Microsoft.AspNetCore.Mvc;  
  
namespace Task_4.Controllers  
{  
    public class Page4Controller : Controller  
    {  
        // GET: Page4/Index  
        // Повертаємо View для Сторінки 4  
        public IActionResult Index()  
        {  
            // Повертаємо View Стор 4)  
            return View();  
        }  
    }  
}
```

### **Page5Controller.cs:**

```
using Microsoft.AspNetCore.Mvc;  
  
namespace Task_4.Controllers  
{  
    public class Page5Controller : Controller  
    {  
        // GET: Page5/Index
```

```

// Метод повертає подання для Стор 5
public IActionResult Index()
{
    // Повертаємо подання для Стор 5
    return View();
}
}
}

```

## 2. Подання (Views):

- **Index.cshtml** для кожної сторінки (Page1, Page2, Page3, Page4, Page5) – ці файли демонструють оформлення кожної окремої сторінки (задання фонового зображення через ViewBag, відображення тексту та кнопок для навігації згідно з заданою схемою переходів).
- **\_Layout.cshtml** – загальний шаблон застосунку, який забезпечує єдине оформлення (Navbar, основний контейнер, футер, фон). Він показує використання Bootstrap та власних стилів.
- **\_Layout.cshtml.css** – файл, який містить кастомні CSS-стилі для загального оформлення веб-застосунку (Navbar, контейнер, футер, фон тощо).

### Page1/Index.cshtml:

```

@{
    ViewData["Title"] = "Стор1";

    // Задаємо фон для Стор1 через ViewBag
    ViewBag.BackgroundColor =
        "https://travel.home.sndimg.com/content/dam/images/travel/stock/2016/9/9/0/GettyImages-SapnaReddyPhotography-459319033_Yosemite.jpg.rend.hgtvcom.1280.853.suffix/1491594411873.jpeg";
}

<!-- Контейнер для вмісту сторінки -->
<div class="container my-5">
    <h2 class="mb-3">Сторінка 1</h2>
    <p>Це Стор1. Перейдіть далі, натиснувши кнопку нижче.</p>
    <!-- Кнопка для переходу на сторінку 3 -->
    <a asp-controller="Page3" asp-action="Index" class="btn btn-primary">Перейти на Стор3</a>
</div>

```

## Page2/Index.cshtml:

```
@{
    ViewData["Title"] = "Стор2";
    // Задаємо фон для сторінки 2 через ViewBag
    ViewBag.BackgroundColor = "https://images.unsplash.com/photo-1498429089284-41f8cf3fffd39?fm=jpg&q=60&w=3000&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWZyY2h8OHx8d2FsbHBhcGVyJTlIwbmF0dXJlGVufDB8fDB8fHww";
}

<!-- Контейнер з відступами для розміщення вмісту -->
<div class="container my-5">
    <h2 class="mb-3">Сторінка 2</h2>
    <p>Це Стор2. Перейдіть далі, натиснувши кнопку нижче.</p>
    <!-- Кнопка для повернення на Стор1 -->
    <a asp-controller="Page1" asp-action="Index" class="btn btn-primary">Повернутися на Стор1</a>
</div>
```

## Page3/Index.cshtml:

```
@{
    ViewData["Title"] = "Стор3";
    // Задаємо фон для Стор3 через ViewBag
    ViewBag.BackgroundColor = "https://www.baltana.com/files/wallpapers-17/Yosemite-Valley-California-USA-Wallpaper-44410.jpg";
}

<!-- Основний контейнер для вмісту сторінки -->
<div class="container my-5">
    <h2 class="mb-3">Сторінка 3</h2>
    <p>Це Стор3. Перейдіть до наступної сторінки.</p>
    <!-- Кнопка для переходу на Стор4 -->
    <a asp-controller="Page4" asp-action="Index" class="btn btn-primary">Перейти на Стор4</a>
</div>
```

## Page4/Index.cshtml:

```
@{
    ViewData["Title"] = "Стор4";
    // Задаємо фон для Стор4 через ViewBag
    ViewBag.BackgroundColor = "https://wallpapercave.com/wp/wp8975834.jpg";
}
```

```

<!-- Основний контейнер для розміщення вмісту -->
<div class="container my-5">
    <h2 class="mb-3">Сторінка 4</h2>
    <p>Це Стор4. Перейдіть до наступної сторінки.</p>
    <!-- Кнопка для переходу на Стор5 -->
    <a asp-controller="Page5" asp-action="Index" class="btn btn-primary">Перейти на Стор5</a>
</div>

```

## Page5/Index.cshtml:

```

@{
    ViewData["Title"] = "Стор5";
    // Задаємо фон для Стор5 через ViewBag
    ViewBag.BackgroundColor = "https://assets.tommackie.com/wp-
content/uploads/2021/05/25145056/180540-1-1.jpg";
}

```

```

<!-- Основний контейнер для вмісту сторінки -->
<div class="container my-5">
    <h2 class="mb-3">Сторінка 5</h2>
    <p>Це Стор5. Перейдіть до наступної сторінки.</p>
    <!-- Кнопка для переходу на Стор2 -->
    <a asp-controller="Page2" asp-action="Index" class="btn btn-primary">Перейти на Стор2</a>
</div>

```

## \_Layout.cshtml:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Task_4</title>
    <!-- Підключення Bootstrap CSS -->
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <!-- Підключення загальних стилів сайту -->
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <!-- Підключення кастомних стилів для Task_4 -->
    <link rel="stylesheet" href="~/Task_4.styles.css" asp-append-version="true" />
</head>
<body style="background: url('@(ViewBag.BackgroundColor ??
"https://images4.alphacoders.com/122/1222302.jpg")') no-repeat center center fixed; background-
size: cover;">

```

```

<header>
    <!-- Навігаційна панель з Bootstrap -->
    <nav class="navbar navbar-expand-sm navbar-light bg-white border-bottom box-shadow mb-
3">
        <div class="container-fluid">
            <!-- Назва сайту посилання на Home -->
            <a class="navbar-brand" asp-area="" asp-controller="Home" asp-
action="Index">Task_4</a>

            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target=".navbar-collapse"
                aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <!-- Меню навігації -->
            <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
                <ul class="navbar-nav flex-grow-1">

                    <li class="nav-item">
                        <a class="nav-link text-dark" asp-area="" asp-controller="Home"
asp-action="Index">Home</a>
                    </li>

                    <li class="nav-item">
                        <a class="nav-link text-dark" asp-area="" asp-controller="Home"
asp-action="Privacy">Privacy</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>
</header>

<!-- Основний контейнер -->
<div class="container my-5" style="background-color: rgba(255,255,255,0.9); border-radius:
10px; padding: 20px;">
    <main role="main" class="pb-3">
        @RenderBody() <!-- Тут рендер вмісту конкретної сторінки -->
    </main>
</div>

<!-- Футер -->

```

```

        <footer class="border-top footer mt-4" style="background-color: rgba(0,0,0,0.5); color:
#FFD700;">
            <div class="container">
                &copy; 2025 - Task_4 -

                <a asp-area="" asp-controller="Home" asp-action="Privacy" style="color:
#FFD700;">Privacy</a>
            </div>
        </footer>

        <!-- Підключення jQuery, Bootstrap JS та власних скриптів -->
        <script src="~/lib/jquery/dist/jquery.min.js"></script>
        <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
        <script src="~/js/site.js" asp-append-version="true"></script>
        @await RenderSectionAsync("Scripts", required: false)
    </body>
</html>

```

## \_Layout.cshtml.css:

```

/* Please see documentation at https://learn.microsoft.com/aspnet/core/client-side/bundling-
and-minification
for details on configuring this project to bundle and minify static web assets. */

/* Стиль для логотипу у navbar */
a.navbar-brand {
    white-space: normal;
    text-align: center;
    word-break: break-all;
}

/* Загальний стиль для посилань */
a {
    color: #0077cc;
}

/* Стиль для кнопок primary */
.btn-primary {
    color: #fff;
    background-color: #1b6ec2;
    border-color: #1861ac;
}

/* Стиль для активних елементів у nav-pills */

```



```
.nav-pills .nav-link.active, .nav-pills .show > .nav-link {  
    color: #fff;  
    background-color: #1b6ec2;  
    border-color: #1861ac;  
}
```

```
/* Стил ь для верхньої межі */  
.border-top {  
    border-top: 1px solid #e5e5e5;  
}
```

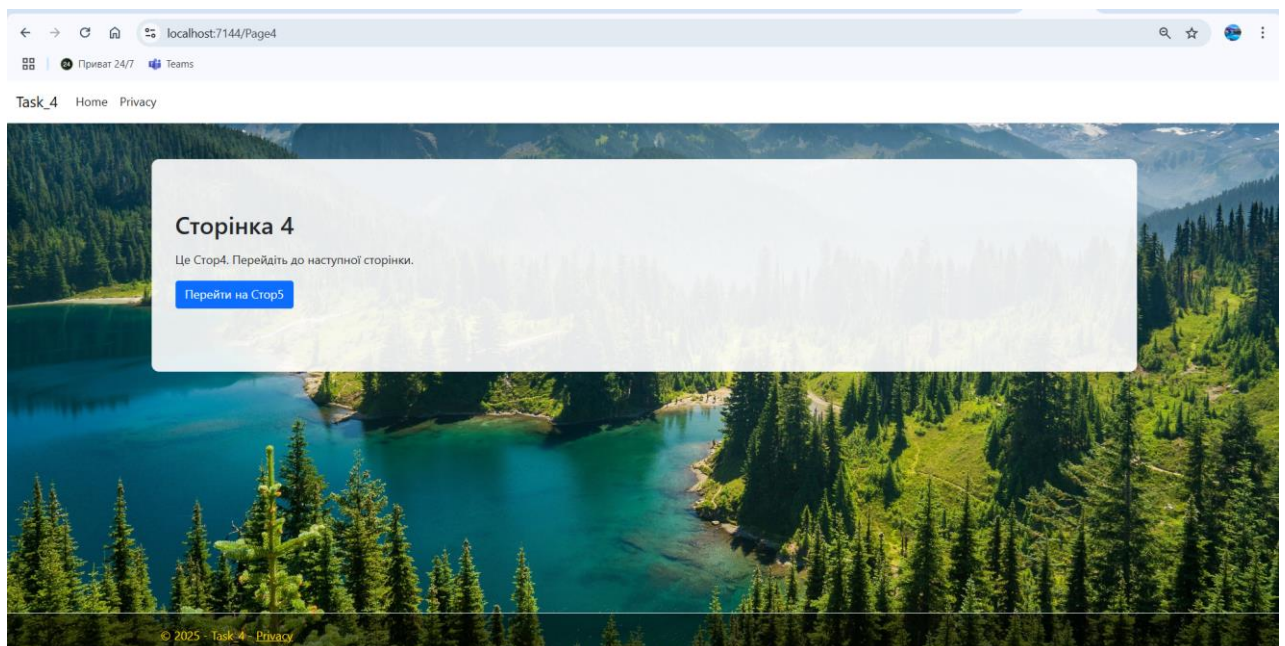
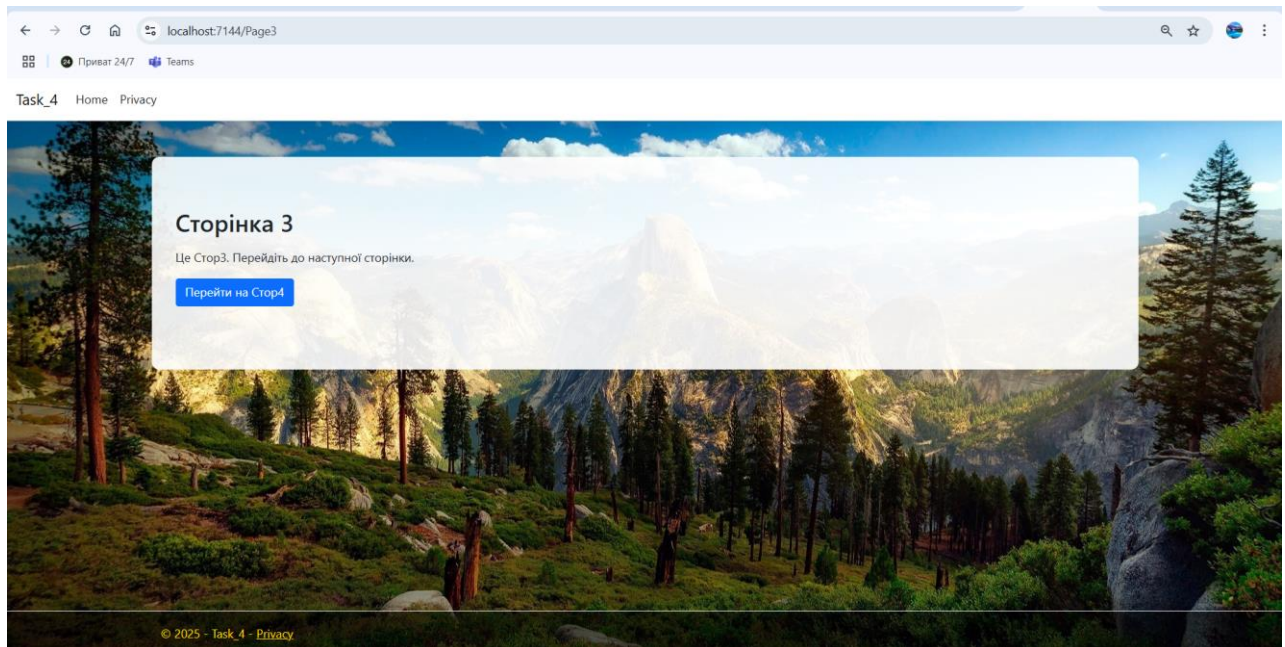
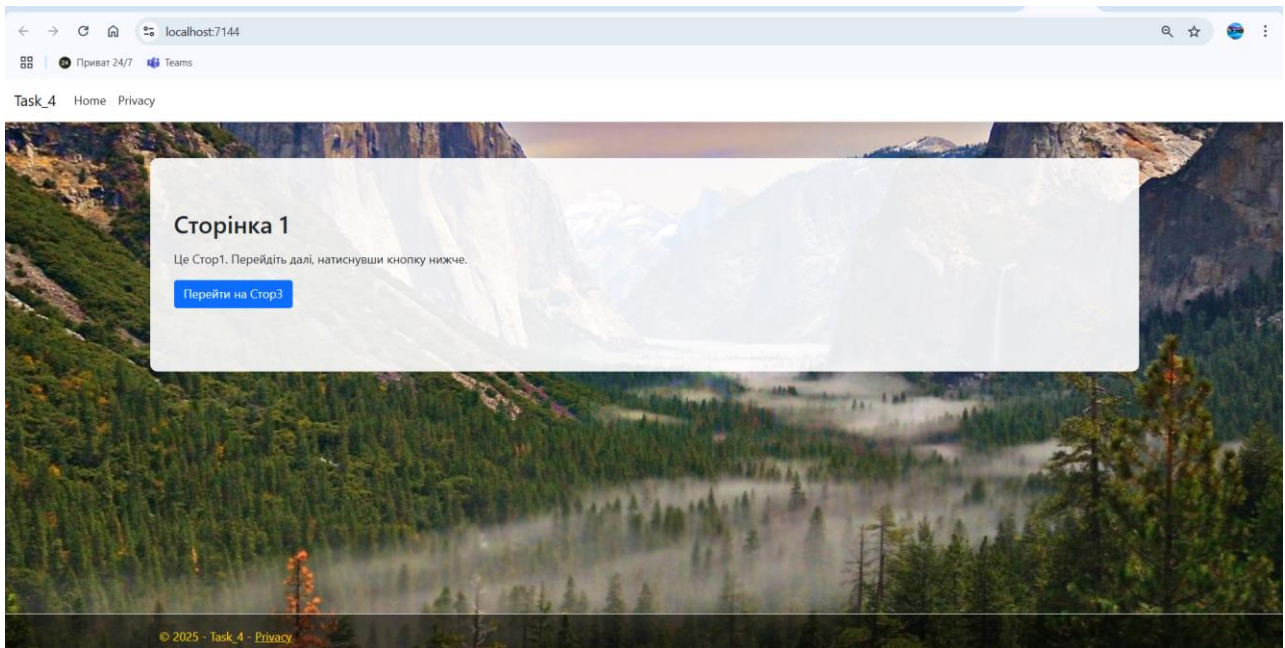
```
/* Стил ь для нижньої межі */  
.border-bottom {  
    border-bottom: 1px solid #e5e5e5;  
}
```

```
/* Додаємо легку тінь для ефекту об'єму */  
.box-shadow {  
    box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);  
}
```

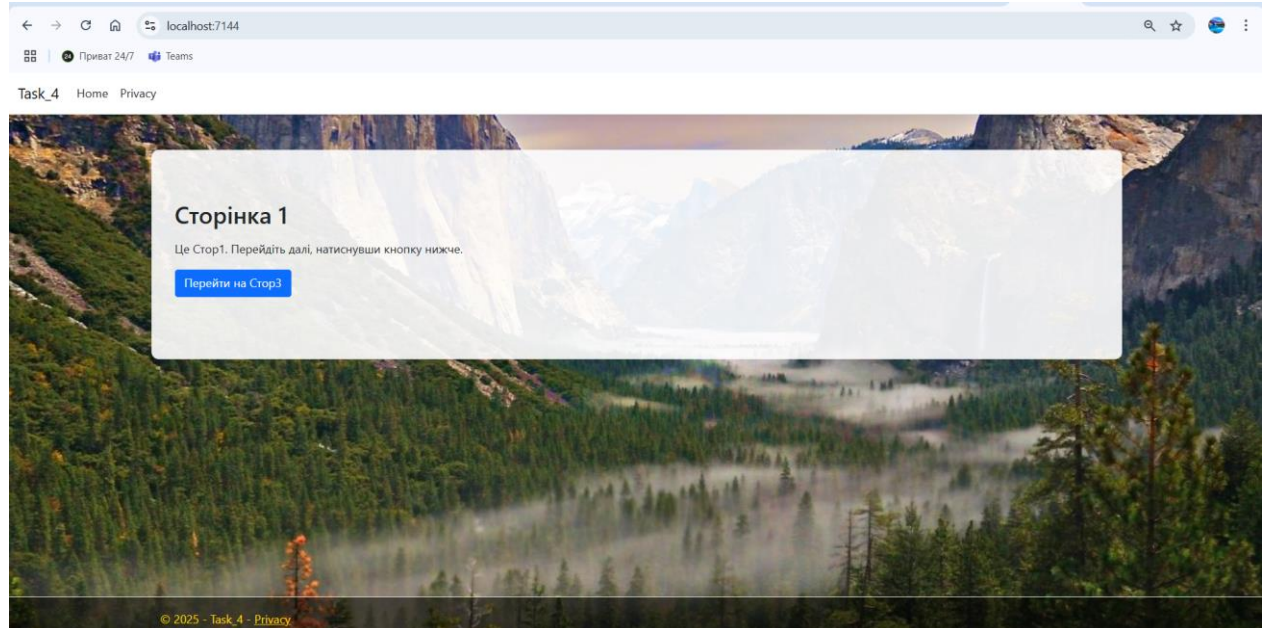
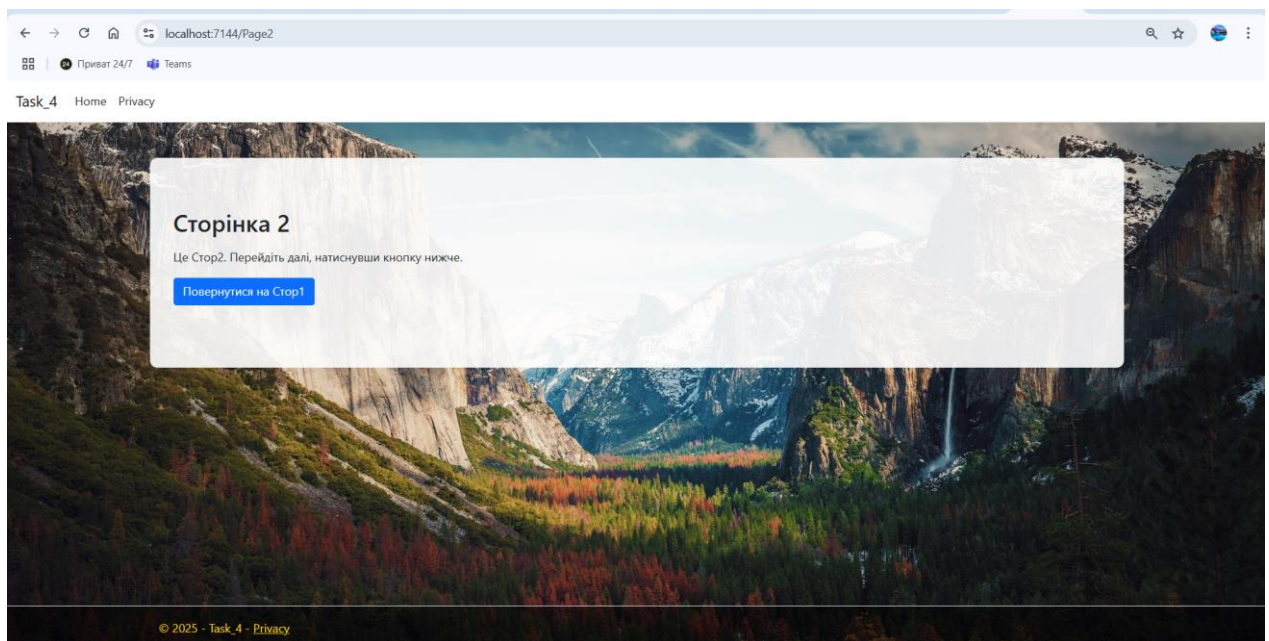
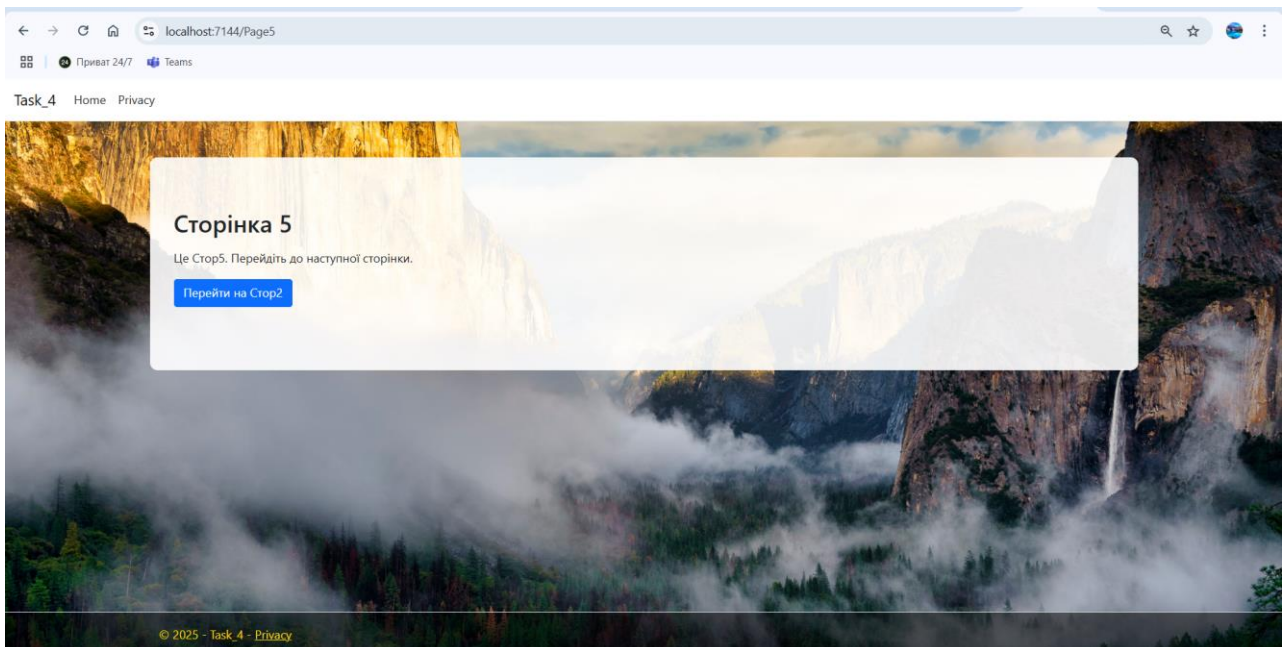
```
/* Стил ь для кнопки прийняття політики */  
button.accept-policy {  
    font-size: 1rem;  
    line-height: inherit;  
}
```

```
/* Стил ь для футера */  
.footer {  
    position: absolute;  
    bottom: 0;  
    width: 100%;  
    white-space: nowrap;  
    line-height: 60px;  
}
```

# Результат роботи програми:







## Алгоритм побудови проєкту і кодів

### 1. Створення проєкту:

- У Visual Studio додав до існуючого рішення новий проєкт типу ASP.NET Core MVC з назвою *Task\_4*.

### 2. Налаштування маршрутів:

- У файлі **Program.cs** вказав маршрут за замовчуванням так, щоб відкривалася перша сторінка.

### 3. Розробка контролерів (Page1Controller, Page2Controller, Page3Controller, Page4Controller, Page5Controller):

- У кожному контролері є метод **Index()**, який повертає відповідне подання (Index.cshtml).
- Кожен контролер відповідає за свою сторінку (Стор1, Стор2, Стор3, Стор4, Стор5).
- Схема переходів реалізується через кнопки (або посилання) у відповідному поданні (наприклад, Стор1 має кнопку, що веде до Стор3, Стор3 – до Стор4 тощо).

### 4. Розробка подань для кожної сторінки (Page1/Index.cshtml, Page2/Index.cshtml, Page3/Index.cshtml, Page4/Index.cshtml, Page5/Index.cshtml):

- Кожне подання відображає свою сторінку з потрібним вмістом (заголовки, текст) і має кнопку для переходу на іншу сторінку за схемою.
- Для передачі даних про фонове зображення або інші параметри використовуємо ViewBag або прописуємо прямо в коді подання.

### 5. Використання \_Layout.cshtml:

- Створив загальний шаблон **\_Layout.cshtml** із підключенням Bootstrap для оформлення (Navbar, контейнер, футер).
- У кожній сторінці (Page1, Page2 тощо) можна задати власне фонове зображення через ViewBag.BackgroundColor або прямо в поданні.

### 6. Тестування:

- Запустив застосунок, відкрив Стор1.
- Перевірив, що кнопка на Стор1 веде до Стор3, Стор3 веде до Стор4, Стор4 – до Стор5, Стор5 – до Стор2, а Стор2 – назад на Стор1.

- Переконався, що переходи працюють безпомилково та відображаються вірні подання.

## **Функціональність**

- **Переходи між сторінками:** Кожна сторінка має власний контролер і подання, де передбачено кнопку для переходу на іншу сторінку.
- **Схема навігації:** Стор1 → Стор3, Стор3 → Стор4, Стор4 → Стор5, Стор5 → Стор2, Стор2 → Стор1 (згідно з варіантом №12).
- **Оформлення:** Використано Bootstrap для сучасного адаптивного дизайну (\_Layout.cshtml).

## **Висновок**

Завдання **Task\_4** виконано згідно з вимогами варіанта №12. У проєкті створено п'ять сторінок (Стор1...Стор5) із контролерами та поданнями. Кожна сторінка має кнопку для переходу до наступної сторінки за заданою схемою, забезпечуючи циклічну навігацію. Оформлення сторінок реалізовано за допомогою Bootstrap у загальному шаблоні \_Layout.cshtml.

## Завдання 5

Використовуючи сесійні атрибути (Session) модифікувати минуле завдання таким чином, щоб на кожному поданні відображався лічильник, який рахував би кількість відвідувань відповідного подання.

**Текст програми:**

### **1. Контролери**

**Page1Controller.cs, Page2Controller.cs, Page3Controller.cs, Page4Controller.cs, Page5Controller.cs**

Кожен контролер містить метод Index(), де:

1. Зчитується поточне значення лічильника зі сесії (наприклад, Page1Count),
2. Інкрементується (додається 1),
3. Записується назад у сесію,
4. Передається у ViewBag.VisitCount для відображення у відповідному поданні.

### **Page1Controller.cs:**

```
using Microsoft.AspNetCore.Http;  
using Microsoft.AspNetCore.Mvc;
```

```
namespace Task_5.Controllers
```

```
{  
    public class Page1Controller : Controller  
    {  
        // GET: Page1/Index  
        public IActionResult Index()  
        {  
            // Ключ для зберігання ліч Стор 1  
            const string sessionKey = "Page1Count";  
            // Зчитуємо поточ значення ліч  
            int count = (HttpContext.Session.GetInt32(sessionKey) ?? 0) + 1;  
            // Записуємо оновлене  
            HttpContext.Session.SetInt32(sessionKey, count);  
            // Передаємо значення лічильника у ViewBag  
            ViewBag.VisitCount = count;  
  
            return View();  
        }  
    }  
}
```

## Page2Controller.cs:

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace Task_5.Controllers
{
    public class Page2Controller : Controller
    {
        // GET: Page2/Index
        public IActionResult Index()
        {
            // Визначаємо ключ для збереження ліч відв Стор 2
            const string sessionKey = "Page2Count";
            // Зчитуємо поточне знач з сесії
            int count = (HttpContext.Session.GetInt32(sessionKey) ?? 0) + 1;
            // Записуємо оновлене значення
            HttpContext.Session.SetInt32(sessionKey, count);
            // Передаємо оновлений ліч у ViewBag
            ViewBag.VisitCount = count;

            return View();
        }
    }
}
```

## Page3Controller.cs:

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace Task_5.Controllers
{
    public class Page3Controller : Controller
    {
        // GET: Page3/Index
        public IActionResult Index()
        {
            // Визначаємо ключ сесії для збереження ліч відв Стор 3
            const string sessionKey = "Page3Count";
            // Зчитуємо поточне знач з сесії
            int count = (HttpContext.Session.GetInt32(sessionKey) ?? 0) + 1;
            // Записуємо оновлене значення
            HttpContext.Session.SetInt32(sessionKey, count);
            // Передаємо оновлений ліч у ViewBag
        }
    }
}
```

```

        ViewBag.VisitCount = count;

        return View();
    }
}

```

### Page4Controller.cs:

```

using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace Task_5.Controllers
{
    public class Page4Controller : Controller
    {
        // GET: Page4/Index
        public IActionResult Index()
        {
            // Визначаємо ключ сесії для збереження ліч відв Стор 4
            const string sessionKey = "Page4Count";
            // Зчитуємо поточне знач з сесії
            int count = (HttpContext.Session.GetInt32(sessionKey) ?? 0) + 1;
            // Записуємо оновлене значення ліч
            HttpContext.Session.SetInt32(sessionKey, count);
            // Передаємо значення ліч у ViewBag
            ViewBag.VisitCount = count;

            return View();
        }
    }
}

```

### Page5Controller.cs:

```

using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace Task_5.Controllers
{
    public class Page5Controller : Controller
    {
        // GET: Page5/Index
        public IActionResult Index()

```



```

{
    // Визначаємо ключ для ліч відв Стор 5
    const string sessionKey = "Page5Count";
    // Зчитуємо поточне знач з сесії
    int count = (HttpContext.Session.GetInt32(sessionKey) ?? 0) + 1;
    // Записуємо оновлене знач
    HttpContext.Session.SetInt32(sessionKey, count);
    // Передаємо ліч у ViewBag
    ViewBag.VisitCount = count;

    return View();
}
}
}

```

## Подання (Views)

**Index.cshtml** у кожній папці (Page1, Page2, Page3, Page4, Page5).

- У цих поданнях відображається лічильник відвідувань: `@ViewBag.VisitCount`.
- Міститься кнопка для переходу до наступної сторінки (згідно з заданою схемою).
- Це найкраще показує, як відображається результат підрахунку відвідувань та організовано навігацію.

### Page1/Index.cshtml:

```

@{
    ViewData["Title"] = "Сторінка 1";
    // Задаємо фон для Стор 1 через ViewBag
    ViewBag.BackgroundImage =
"https://static.vecteezy.com/system/resources/previews/049/547/631/non_2x/stunning-high-
resolution-nature-and-landscape-backgrounds-breathtaking-scenery-in-hd-free-photo.jpg";
}

<!-- Контейнер для основного вмісту сторінки -->
<div class="container my-5">
    <h2 class="mb-3">Сторінка 1</h2>
    <!-- Відображення ліч відвідувань -->
    <p class="mb-3">Відвідувань: @ViewBag.VisitCount</p>
    <!-- Опис сторінки -->
    <p>Це Сторінка 1.</p>
    <!-- Кнопка для переходу на Стор 3 -->
    <a asp-controller="Page3" asp-action="Index" class="btn btn-primary">Далі</a>

```

</div>

## Page2/Index.cshtml:

```
@{
    ViewData["Title"] = "Сторінка 2";
    // Задаємо фон для Стор 2 через ViewBag
    ViewBag.BackgroundColor = "https://www.bsr.org/images/heroes/bsr-focus-nature-hero.jpg";
}

<!-- Контейнер для основного вмісту сторінки -->
<div class="container my-5">
    <h2 class="mb-3">Сторінка 2</h2>
    <!-- Відображення ліч відвідувань -->
    <p class="mb-3">Відвідувань: @ViewBag.VisitCount</p>
    <p>Це Сторінка 2.</p>
    <!-- Кнопка для переходу на Стор 1 -->
    <a asp-controller="Page1" asp-action="Index" class="btn btn-primary">Далі</a>
</div>
```

## Page3/Index.cshtml:

```
@{
    ViewData["Title"] = "Сторінка 3";
    // Задаємо фон для Стор 3 через ViewBag
    ViewBag.BackgroundColor = "https://images.pexels.com/photos/158063/bellingrath-gardens-alabama-landscape-scenic-158063.jpeg?cs=srgb&dl=pexels-pixabay-158063.jpg&fm=jpg";
}

<!-- Контейнер для основного вмісту сторінки -->
<div class="container my-5">
    <h2 class="mb-3">Сторінка 3</h2>
    <!-- Виводимо ліч відвідувань -->
    <p class="mb-3">Відвідувань: @ViewBag.VisitCount</p>
    <p>Це Сторінка 3.</p>
    <!-- Кнопка для переходу до Стор 4 -->
    <a asp-controller="Page4" asp-action="Index" class="btn btn-primary">Далі</a>
</div>
```

## Page4/Index.cshtml:

```
@{
    ViewData["Title"] = "Сторінка 4";
    // Задаємо фон для Стор 4 через ViewBag
```

```

        ViewBag.BackgroundColor = "https://images.squarespace-
cdn.com/content/v1/5d777de8109c315fd22faf3a/1693407136044-
G90XQURX1GABMYGAS8K1/shutterstock_1288634614.jpg?format=2500w";
    }

<!-- Контейнер для основного вмісту сторінки -->
<div class="container my-5">
    <h2 class="mb-3">Сторінка 4</h2>
    <!-- Вивід ліч відвідувань -->
    <p class="mb-3">Відвідувань: @ViewBag.VisitCount</p>
    <p>Це Сторінка 4.</p>
    <!-- Кнопка для переходу на Стр 5-->
    <a asp-controller="Page5" asp-action="Index" class="btn btn-primary">Далі</a>
</div>

```

## Page5/Index.cshtml:

```

@{
    ViewData["Title"] = "Сторінка 5";
    // Задаємо фон для Стр 5 через ViewBag
    ViewBag.BackgroundColor = "https://assets.tommackie.com/wp-
content/uploads/2021/05/25145056/180540-1-1.jpg";
}

<!-- Контейнер для основного вмісту з відступами -->
<div class="container my-5">
    <h2 class="mb-3">Сторінка 5</h2>
    <!-- Відображення ліч відвідувань -->
    <p class="mb-3">Відвідувань: @ViewBag.VisitCount</p>
    <p>Це Сторінка 5.</p>
    <!-- Кнопка "Далі" для переходу на Стр 2 -->
    <a asp-controller="Page2" asp-action="Index" class="btn btn-primary">Далі</a>
</div>

```

## 2. Подання (Views):

- **Index.cshtml** для кожної сторінки (Page1, Page2, Page3, Page4, Page5) – ці файли демонструють оформлення кожної окремої сторінки (задання фонового зображення через ViewBag, відображення тексту та кнопок для навігації згідно з заданою схемою переходів).

- **\_Layout.cshtml** – загальний шаблон застосунку, який забезпечує єдине оформлення (Navbar, основний контейнер, футер, фон). Він показує використання Bootstrap та власних стилів.
- **\_Layout.cshtml.css** – файл, який містить кастомні CSS-стилі для загального оформлення веб-застосунку (Navbar, контейнер, футер, фон тощо).

### **Index.cshtml:**

```
@{
    ViewData["Title"] = "Home Page";
}

<!-- Контейнер для центрування вмісту на сторінці -->
<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>
        Learn about
        <a href="https://learn.microsoft.com/aspnet/core">building Web apps with ASP.NET
Core</a>.
    </p>
</div>
```

### **\_Layout.cshtml:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Task_5</title>
    <!-- Підключення Bootstrap CSS -->
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <!-- Підключення основних стилів сайту -->
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <!-- Підключення кастомних стилів для Task_5 -->
    <link rel="stylesheet" href="~/Task_5.styles.css" asp-append-version="true" />
</head>
<body style="background: url('@(ViewBag.BackgroundColor ??
"https://static.vecteezy.com/system/resources/previews/049/547/631/non_2x/stunning-high-
resolution-nature-and-landscape-backgrounds-breathtaking-scenery-in-hd-free-photo.jpg")') no-
repeat center center fixed; background-size: cover;">
    <header>
        <!-- Navbar з Bootstrap -->
```

```

<nav class="navbar navbar-expand-sm navbar-light bg-white border-bottom box-shadow mb-
3">

    <div class="container-fluid">
        <!-- Назва сайту де посилання веде на Home -->
        <a class="navbar-brand" asp-area="" asp-controller="Home" asp-
action="Index">Task_5</a>

        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target=".navbar-collapse"
            aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <!-- Меню навігації -->
        <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
            <ul class="navbar-nav flex-grow-1">
                <li class="nav-item">
                    <a class="nav-link text-dark" asp-area="" asp-controller="Home"
asp-action="Index">Home</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link text-dark" asp-area="" asp-controller="Home"
asp-action="Privacy">Privacy</a>
                </li>
            </ul>
        </div>
    </div>
</nav>
</header>

<!-- Основний контейнер для контенту -->
<div class="container my-5" style="background-color: rgba(255,255,255,0.9); border-radius:
10px; padding: 20px;">
    <main role="main" class="pb-3">
        @RenderBody() <!-- Рендер контенту конкретної стор -->
    </main>
</div>

<!-- Футер -->
<footer class="border-top footer mt-4" style="background-color: rgba(0,0,0,0.5); color:
#FFD700;">
    <div class="container">
        &copy; 2025 - Task_5 -

```

```

        <a asp-area="" asp-controller="Home" asp-action="Privacy" style="color:
#FFD700;">Privacy</a>
    </div>
</footer>

<!-- Підключення jQuery та Bootstrap JS -->
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
<!-- Підключення основних скриптів сайту -->
<script src="~/js/site.js" asp-append-version="true"></script>
@await RenderSectionAsync("Scripts", required: false)
</body>
</html>

```

## \_Layout.cshtml.css:

```

/* Please see documentation at https://learn.microsoft.com/aspnet/core/client-side/bundling-
and-minification
for details on configuring this project to bundle and minify static web assets. */

```

```

/* Стиль для логотипу в Navbar */

```

```

a.navbar-brand {
    white-space: normal;
    text-align: center;
    word-break: break-all;
}

```

```

/* Загальний стиль для посилань */

```

```

a {
    color: #0077cc;
}

```

```

/* Стиль для кнопок з класом btn-primary */

```

```

.btn-primary {
    color: #fff;
    background-color: #1b6ec2;
    border-color: #1861ac;
}

```

```

/* Стиль для активних елементів у nav-pills */

```

```

.nav-pills .nav-link.active,
.nav-pills .show > .nav-link {
    color: #fff;
    background-color: #1b6ec2;
}

```

```
border-color: #1861ac;
}

/* Стиль для верхньої межі елементів */
.border-top {
border-top: 1px solid #e5e5e5;
}

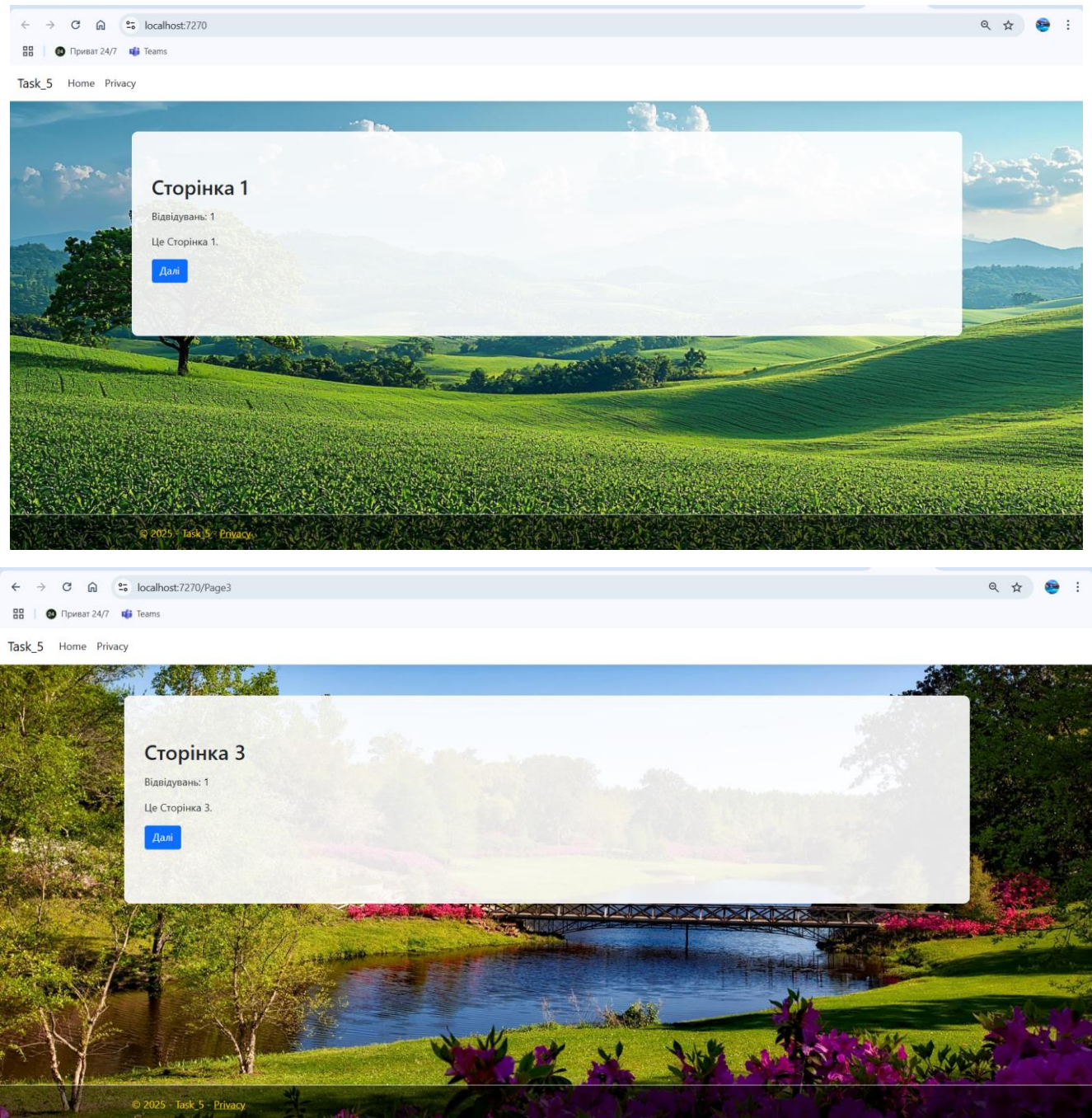
/* Стиль для нижньої межі елементів */
.border-bottom {
border-bottom: 1px solid #e5e5e5;
}

/* Стиль для додання легкого об'ємного ефекту */
.box-shadow {
box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);
}

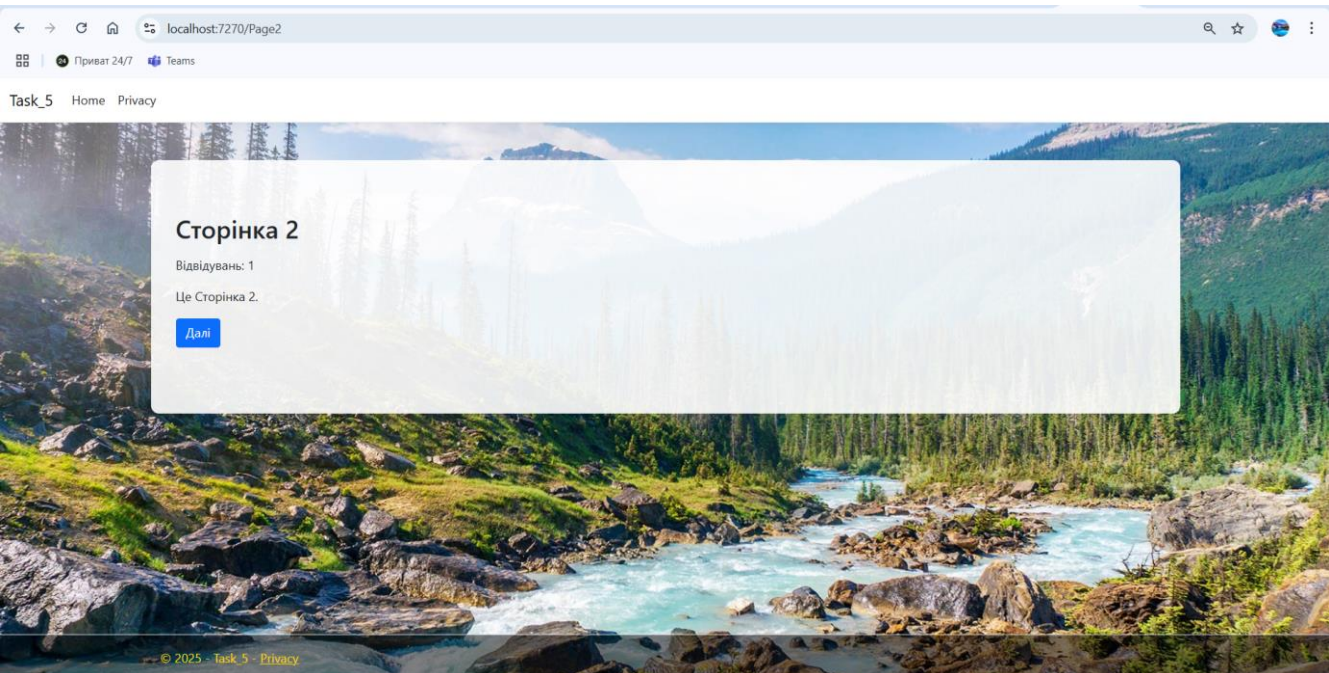
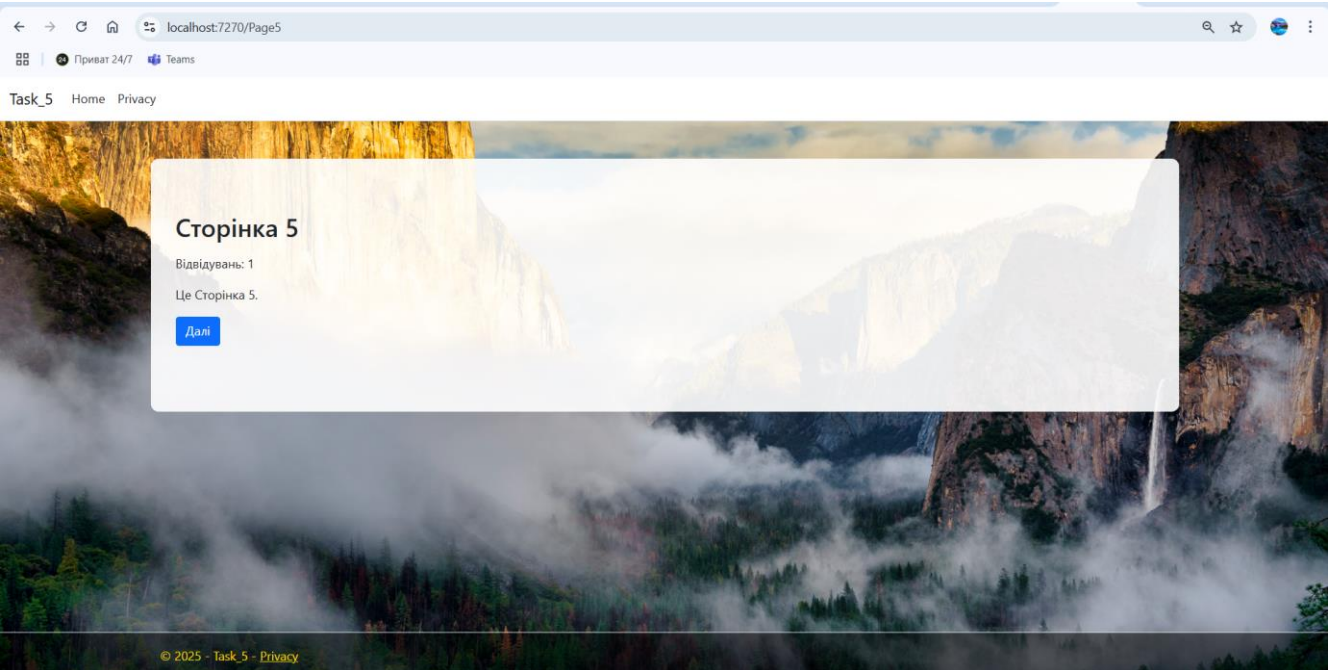
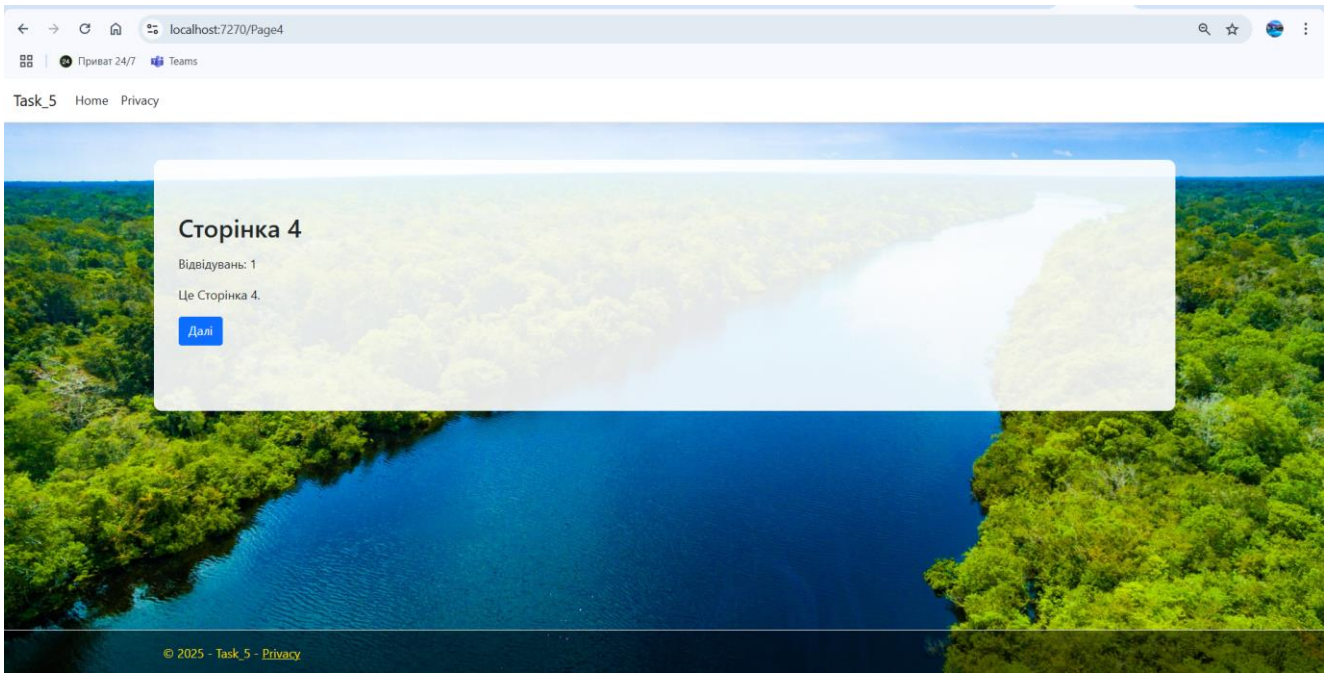
/* Стиль для кнопки прийняття політики */
button.accept-policy {
font-size: 1rem;
line-height: inherit;
}

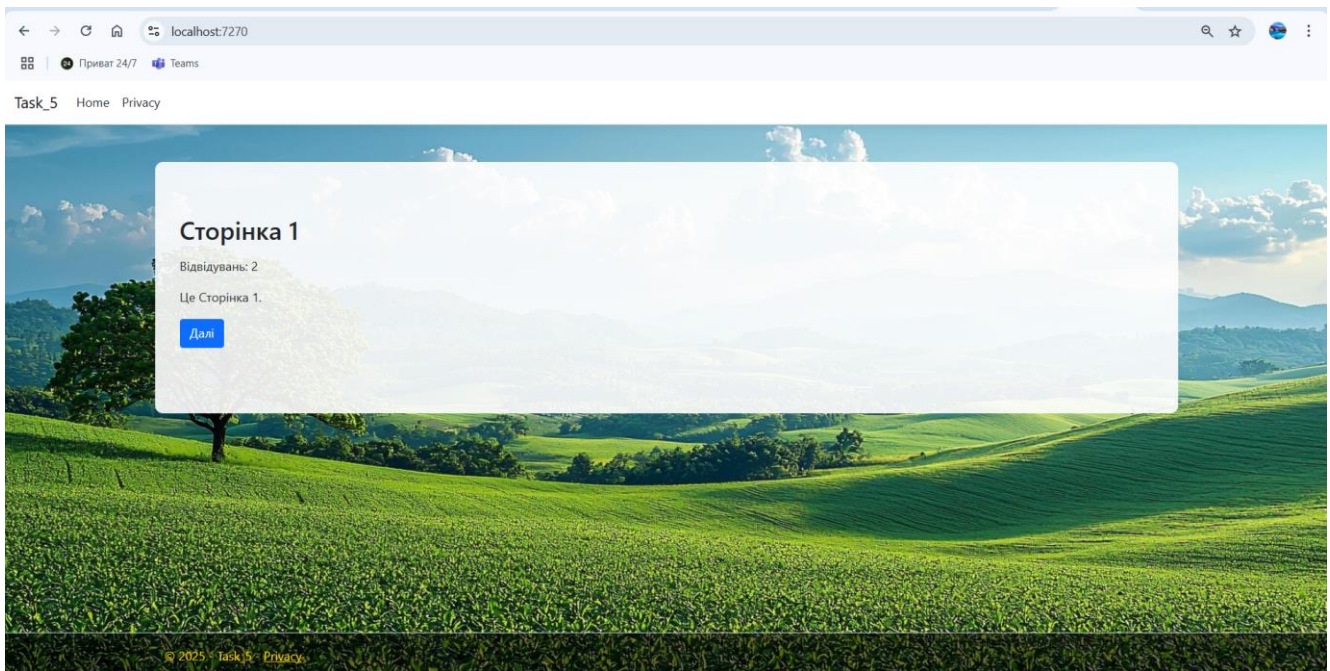
/* Стиль для футера */
.footer {
position: absolute;
bottom: 0;
width: 100%;
white-space: nowrap;
line-height: 60px;
}
```

# Результати роботи програми:









## Алгоритм побудови проєкту і кодів

### 1. Створення проєкту:

- У Visual Studio додав до існуючого рішення новий проєкт типу ASP.NET Core MVC з назвою *Task\_5*.

### 2. Налаштування підтримки сесій:

- У файлі **Program.cs** додано служби:
  - `AddDistributedMemoryCache()` – для зберігання сесій у пам'яті.
  - `AddSession()` – для використання сесійних атрибутів.
- У конвеєрі HTTP-запитів увімкнув використання сесій за допомогою `app.UseSession()` перед визначенням маршруту.

### 3. Розробка контролерів:

- Створив 5 контролерів (`Page1Controller`, `Page2Controller`, `Page3Controller`, `Page4Controller`, `Page5Controller`), кожен з яких містить метод **Index()**.
- У кожному методі **Index()** реалізовано:
  - Зчитування поточного значення лічильника із сесії за допомогою `HttpContext.Session.GetInt32(...)` (використовуючи унікальний ключ для кожної сторінки, наприклад, "Page1Count").
  - Інкрементування значення (додавання 1).
  - Запис оновленого значення назад у сесію за допомогою `HttpContext.Session.SetInt32(...)`.



- Передача значення лічильника у ViewBag (наприклад, ViewBag.VisitCount) для подальшого відображення у поданні.

#### 4. Розробка подань (Views):

- Створено окремі подання для кожної сторінки (Page1/Index.cshtml, Page2/Index.cshtml, ..., Page5/Index.cshtml), у яких:
  - Відображається лічильник відвідувань (наприклад, Відвідувань: @ViewBag.VisitCount).
  - Реалізовано навігацію між сторінками відповідно до схеми (наприклад, Стор1 → Стор3, Стор3 → Стор4, Стор4 → Стор5, Стор5 → Стор2, Стор2 → Стор1).
- Загальний шаблон **\_Layout.cshtml** забезпечує єдиний вигляд застосунку, підключає Bootstrap та власні стилі.

#### 5. Тестування:

- Запустив застосунок.
- Перевілив, що при кожному переході на конкретну сторінку лічильник відвідувань оновлюється і відображається в поданні.
- Переконався, що навігація між сторінками працює відповідно до заданої схеми.

### Функціональність

- **Підрахунок відвідувань:**

За допомогою сесійних атрибутів кожна сторінка має свій власний лічильник відвідувань, який збільшується при кожному перегляді.

- **Навігація:**

Кожна сторінка містить кнопку для переходу на наступну сторінку згідно з наступною схемою:

- Стор1 → Стор3
- Стор3 → Стор4
- Стор4 → Стор5
- Стор5 → Стор2
- Стор2 → Стор1

- **Оформлення:**

Загальне оформлення застосунку реалізовано за допомогою Bootstrap та власних CSS, що забезпечує адаптивний і сучасний вигляд.

## **Висновок**

Завдання Task\_5 виконано відповідно до вимог варіанта №12. Реалізовано механізм підрахунку відвідувань кожної сторінки за допомогою сесійних атрибутів, що демонструється через оновлення лічильників у кожному контролері і їх відображення у відповідних поданнях. Також реалізовано навігацію між 5 сторінками відповідно до заданої схеми. Оформлення застосунку виконано за допомогою Bootstrap, що забезпечує сучасний та адаптивний інтерфейс.

## **Загальний висновок**

### **1. Завдання 1:**

У ході виконання першого завдання було створено MVC-застосунок, який дозволяє передавати параметри через контролер у подання. Використано механізм ViewBag для передачі даних між компонентами. Реалізовано загальний шаблон оформлення (\_Layout.cshtml), що забезпечує сучасний та адаптивний дизайн застосунку.

### **2. Завдання 2\_1:**

Реалізовано веб-застосунок із можливістю завантаження зображень, розміри яких отримуються з БД. Використано механізм ViewBag для динамічного відображення зображень відповідно до топології «сходинок». Оформлення виконано з використанням Bootstrap для адаптивного інтерфейсу.

### **3. Завдання 2\_2:**

Додано ще одну топологію відображення зображень, що відрізняється від попереднього варіанту. Основна логіка збережена, але змінено CSS-стилі та порядок розташування елементів у поданні Display.cshtml. Завдання виконано з урахуванням вимог варіанта №12.

### **4. Завдання 3:**

Реалізовано динамічну генерацію шахової таблиці з введеною користувачем кількістю рядків. Дані про кількість стовпців отримуються з БД. Згенерований HTML-код таблиці передається у ViewBag та виводиться у поданні.

### **5. Завдання 4:**

Створено веб-застосунок із 5 сторінками та реалізованою схемою переходів між ними. Навігація між сторінками відбувається за допомогою кнопок, що відповідають заданій у варіанті схемі. Загальний вигляд оформлений за допомогою Bootstrap у шаблоні \_Layout.cshtml.

### **6. Завдання 5:**

Додано механізм підрахунку відвідувань кожної сторінки за допомогою сесійних змінних. Підрахунок оновлюється під час кожного перегляду відповідної сторінки. Це дозволяє відстежувати активність користувачів у межах застосунку.

## **Загальний висновок**

Лабораторна робота №1 успішно продемонструвала основи використання ASP.NET Core MVC, баз даних та механізмів передачі параметрів між компонентами застосунку. Реалізовані завдання охоплюють ключові концепції, такі як робота з поданнями, контролерами, ViewBag та сесійними змінними. Особливу увагу приділено адаптивному дизайну та зручності використання, що забезпечено завдяки Bootstrap. Усі завдання виконані відповідно до вимог варіанта №12, що дозволяє підтвердити правильність реалізації та ефективність використаних підходів.