Міністерство освіти України

Національний технічний університет "ХПІ"

кафедра "Комп’ютерних наук та інформаційних технологій"

**Звіт**

**Лабораторна робота 8**

з дисципліни "Проєктування серверних застосунків"

Виконав: студент групи КН-922В

Мельник Б. В.

Перевірив:

Івашко А. В.

Харків 2025

**Зміст**

[Завдання 1 3](#_Toc196603483)

[Завдання 2 5](#_Toc196603484)

[Текст програми 6](#_Toc196603485)

[Результати роботи програми 15](#_Toc196603486)

[Алгоритм побудови коду 16](#_Toc196603487)

[Функціональність 17](#_Toc196603488)

[Висновок 18](#_Toc196603489)

[Завдання 3 19](#_Toc196603490)

[Результати роботи програми 29](#_Toc196603491)

[Алгоритм побудови кодів 31](#_Toc196603492)

[Функціональність 32](#_Toc196603493)

[Висновок 32](#_Toc196603494)

[Завдання 4 34](#_Toc196603495)

[Текст програми 34](#_Toc196603496)

[Результати роботи програми 45](#_Toc196603497)

[Алгоритм побудови кодів 48](#_Toc196603498)

[Функціональність 48](#_Toc196603499)

[Висновок 49](#_Toc196603500)

[Загальний висновок 50](#_Toc196603501)

**Тема:** Створення реєстраційних та авторизаційних функцій в MVC ASP.NET WEB-застосунку.

**Мета роботи:** придбати практику та досвід у створенні і використанні реєст-раційного та авторизаційного функціоналу під час розробки MVC ASP.NET застосунків.

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

**Індивідуальні завдання**

**Варіант №12**

# **Завдання 1**

Створити початковий MVC WEB-проект та перевірити його роботу на одному (бажано на усіх) з найбільш популярних серверів.

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

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.1. Створення початкового MVC WEB-проекту

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.2. Перевірити роботи застосунку на одному з найбільш популярних серверів

# **Завдання 2**

Додати до проекту функцію реєстрації користувачів умовної WEB-системи із використанням реєстраційної форми згідно до власного варіанту. Перед занесенням реєстраційних даних до відповідної таблиці, виконати перевірку (валідацію) реєстраційних полів:

* UserName: лише числа та символи латиниці кількістю не більше 15 і щоб ім'я було унікальним (раніше не було в таблиці БД);
* Password: числа, символи латиниці та обов'язково спеціальні символи довжиною не менше 12, та його коректне повторення у відповідному полі;
* Email: коректне значення електронної поштової адреси

В разі наявності помилок повертатися до форми і відображати їх навпроти відповідних полів.

|  |  |
| --- | --- |
| **№ варіанту** | **Шаблон реєстраційної форми** |
| 1 | 1 |
| 12 |  |

Валідацію за бажанням можна робити або через перевірки в контролері,

або через нотації властивостей моделі.

## **Текст програми:**

**SQL-запит:**

USE User1DB; -- Вибір бази даних User1DB

-- Створення таблиці Users

CREATE TABLE Users (

Id INT IDENTITY(1,1) PRIMARY KEY, -- Створення поля Id як автоінкрементного (починається з 1)

Name NVARCHAR(15) NOT NULL UNIQUE, -- Створення поля Name (макс. 15 символів), обов'язкове та унікальне

Email NVARCHAR(255) NOT NULL, -- Створення поля Email (макс. 255 символів), обов'язкове

Phone NVARCHAR(20) NOT NULL, -- Створення поля Phone (макс. 20 символів), обов'язкове

Password NVARCHAR(255) NOT NULL -- Створення поля Password (макс. 255 символів), обов'язкове

);

**Account/Register.cshtml:**

@model Task\_2.Models.RegisterViewModel

@{

ViewData["Title"] = "Register"; // Встановлюємо заголовок сторінки

}

<div class="form-card animate\_\_animated animate\_\_fadeIn">

<h2>Register</h2>

<form asp-action="Register" method="post" novalidate>

<!-- Форма реєстрації -->

<!-- Name -->

<div class="form-material mb-4 position-relative">

<span asp-validation-for="Name" class="text-danger error-message"></span> <!-- Повідомлення про помилку -->

<label asp-for="Name" class="form-label">Name</label> <!-- Текст лейбла -->

<input asp-for="Name" class="form-control" /> <!-- Поле для введення імені -->

</div>

<!-- Email -->

<div class="form-material mb-4 position-relative">

<span asp-validation-for="Email" class="text-danger error-message"></span> <!-- Повідомлення про помилку -->

<label asp-for="Email" class="form-label">Email</label> <!-- Текст лейбла -->

<input asp-for="Email" class="form-control" /> <!-- Поле для введення email -->

</div>

<!-- Phone -->

<div class="form-material mb-4 position-relative">

<span asp-validation-for="Phone" class="text-danger error-message"></span> <!-- Повідомлення про помилку -->

<label asp-for="Phone" class="form-label">Phone</label> <!-- Текст лейбла -->

<input asp-for="Phone" class="form-control" /> <!-- Поле для введення номера телефону -->

</div>

<!-- Password -->

<div class="form-material mb-4 position-relative">

<span asp-validation-for="Password" class="text-danger error-message"></span> <!-- Повідомлення про помилку -->

<label asp-for="Password" class="form-label">Password</label> <!-- Текст лейбла -->

<input asp-for="Password" type="password" class="form-control" /> <!-- Поле для введення пароля -->

</div>

<!-- Confirm Password -->

<div class="form-material mb-4 position-relative">

<span asp-validation-for="ConfirmPassword" class="text-danger error-message"></span> <!-- Повідомлення про помилку -->

<label asp-for="ConfirmPassword" class="form-label">Confirm Password</label> <!-- Текст лейбла -->

<input asp-for="ConfirmPassword" type="password" class="form-control" /> <!-- Поле для підтвердження пароля -->

</div>

<button type="submit" class="btn btn-register">REGISTER</button> <!-- Кнопка для відправлення форми -->

</form>

</div>

@section Scripts {

@{

await Html.RenderPartialAsync("\_ValidationScriptsPartial"); // Підключення скриптів для валідації

}

<script>

$(function() {

var form = $('form');

// Перезапускаємо парсинг unobtrusive, щоб застосувати наші налаштування

form.removeData('validator').removeData('unobtrusiveValidation');

// Налаштовуємо спосіб показу помилок

$.validator.setDefaults({

showErrors: function(errorMap, errorList) {

// Спочатку показуємо усі звичайно

this.defaultShowErrors();

// Далі перетворюємо текст кожної помилки — замінюємо пробіли на <br/>

$('.error-message').each(function() {

var $el = $(this);

var txt = $el.text().trim();

if (txt) {

var parts = txt.split(' ');

$el.html(parts.join('<br/>'));

}

});

}

});

// Заново ініціалізуємо валідацію форми

$.validator.unobtrusive.parse(form);

});

</script>

}

**Models/User.cs:**

namespace Task\_2.Models

{

// Сутність користувача (таблиця Users у БД)

public class User

{

public int Id { get; set; } // Унікальний ідентифікатор

public string Name { get; set; } = null!; // Ім’я користувача

public string Email { get; set; } = null!; // Email адреса

public string Phone { get; set; } = null!; // Номер телефону

public string Password { get; set; } = null!; // Пароль (зберігається в чистому вигляді для прикладу)

}

}

**Models/RegisterViewModel.cs:**

using System.ComponentModel.DataAnnotations;

namespace Task\_2.Models

{

// Модель представлення для форми реєстрації з валідацією

public class RegisterViewModel

{

[Required(ErrorMessage = "Enter Name")]

[StringLength(15, ErrorMessage = "Max 15 chars")]

[RegularExpression("^[A-Za-z0-9]+$", ErrorMessage = "Only Latin & digits")]

public string Name { get; set; } = null!; // Поле Ім’я

[Required(ErrorMessage = "Enter Email")]

[EmailAddress(ErrorMessage = "Invalid Email")]

public string Email { get; set; } = null!; // Поле Email

[Required(ErrorMessage = "Enter Phone")]

[Phone(ErrorMessage = "Invalid Phone")]

public string Phone { get; set; } = null!; // Поле Телефон

[Required(ErrorMessage = "Enter Password")]

[MinLength(12, ErrorMessage = "At least 12 chars")]

[RegularExpression(@"^(?=.\*[!@#$%^&\*()]).+$", ErrorMessage = "Need special char")]

[DataType(DataType.Password)]

public string Password { get; set; } = null!; // Поле Пароль

[Required(ErrorMessage = "Confirm Password")]

[Compare("Password", ErrorMessage = "Passwords mismatch")]

[DataType(DataType.Password)]

public string ConfirmPassword { get; set; } = null!; // Поле Підтвердження пароля

}

}

**Data/ApplicationDbContext.cs:**

using Microsoft.EntityFrameworkCore;

using Task\_2.Models;

namespace Task\_2.Data

{

// Контекст бази даних EF Core

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> opts)

: base(opts) { } // Конструктор із параметрами конфігурації

public DbSet<User> Users { get; set; } = null!; // Набір користувачів у БД

}

}

**Controllers/AccountController.cs:**

using Microsoft.AspNetCore.Mvc;

using Task\_2.Data;

using Task\_2.Models;

using System.Linq;

namespace Task\_2.Controllers

{

// Контролер для реєстрації та обліку користувачів

public class AccountController : Controller

{

private readonly ApplicationDbContext \_db; // Контекст БД

public AccountController(ApplicationDbContext db) => \_db = db; // Впровадження контексту

[HttpGet]

public IActionResult Register() => View(); // Показати форму реєстрації

[HttpPost]

public IActionResult Register(RegisterViewModel model)

{

// Перевірка унікальності імені користувача

if (\_db.Users.Any(u => u.Name == model.Name))

ModelState.AddModelError(nameof(model.Name), "This Name is taken");

if (!ModelState.IsValid)

return View(model); // Повернути форму з повідомленнями про помилки

// Додавання нового користувача в БД

\_db.Users.Add(new User

{

Name = model.Name,

Email = model.Email,

Phone = model.Phone,

Password = model.Password

});

\_db.SaveChanges(); // Збереження змін

return RedirectToAction("Register"); // Перенаправлення після успіху

}

}

}

**\_Layout.cshtml:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width,initial-scale=1" />

<title>@ViewData["Title"] – Task\_2</title> <!-- Динамічний заголовок -->

<!-- Bootstrap 5 -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"

rel="stylesheet" />

<!-- Animate.css для анімацій -->

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css" />

<link rel="stylesheet" href="~/css/site.css" /> <!-- Кастомні стилі -->

</head>

<body>

<div class="d-flex justify-content-center align-items-center min-vh-100">

@RenderBody() <!-- Основний вміст сторінки -->

</div>

<!-- Скрипти Bootstrap та jQuery Validation -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>

<script src="~/lib/jquery/dist/jquery.min.js"></script>

<script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>

<script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.js"></script>

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

</body>

</html>

**site.css:**

/\* фон сторінки \*/

body {

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

background-size: cover;

}

/\* білий блок форми \*/

.form-card {

background: #fff;

padding: 2rem;

border-radius: 8px;

width: 100%;

max-width: 480px;

box-shadow: 0 4px 16px rgba(0,0,0,0.1);

}

/\* заголовок форми \*/

.form-card h2 {

margin-bottom: 1.5rem;

font-weight: 500;

}

/\* лише нижня лінія для input \*/

.form-material .form-control {

border: none;

border-bottom: 1px solid #ccc;

border-radius: 0;

background: transparent;

box-shadow: none;

padding-left: 0;

padding-right: 0;

}

/\* підсвічування фокусу \*/

.form-material .form-control:focus {

outline: none;

border-bottom-color: #1abc9c;

box-shadow: none;

}

/\* стиль лейбла \*/

.form-material label {

color: #888;

font-size: 0.9rem;

margin-bottom: 0.25rem;

}

/\* стиль повідомлення про помилку \*/

.text-danger {

font-size: 0.8rem;

margin-top: 0.25rem;

}

/\* кнопка реєстрації \*/

.btn-register {

width: 30%;

background-color: #1abc9c;

border: none;

padding: 0.75rem;

text-transform: uppercase;

letter-spacing: 1px;

}

/\* ховер ефект кнопки \*/

.btn-register:hover {

background-color: #16a085;

}

/\* позиціонування повідомлення про помилку \*/

.error-message {

display: block;

position: static;

margin-bottom: 0.25rem;

color: #e74c3c;

font-size: 0.8rem;

line-height: 1.2;

}

/\* трохи додаткового відступу між помилкою і лейблом \*/

.form-material .form-label {

margin-top: 0.25rem;

margin-bottom: 0.25rem;

}

## **Результати роботи програми:**

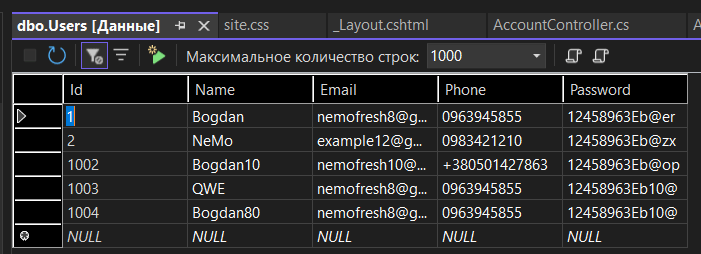


Рисунок 2.1. Створена БД

Изображение выглядит как облако, небо, снимок экрана, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.2. Загальний вигляд форми реєстрації

Изображение выглядит как облако, небо, снимок экрана, пейзаж

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.3. Вигляд помилок

Изображение выглядит как облако, небо, снимок экрана, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.4. Введення даних для реєстрації

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

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.5. Результат запису введених даних

## **Алгоритм побудови коду**

1. **Налаштування бази даних**
   * Виконати SQL-скрипт для створення БД User1DB та таблиці Users з полями Id, Name, Email, Phone, Password.
2. **Опис моделі даних**
   * Створити клас User (відповідність таблиці Users).
   * Створити RegisterViewModel з атрибутами DataAnnotations для валідації (StringLength, RegularExpression, EmailAddress, MinLength, Compare).
3. **Конфігурація EF Core**
   * Додати ApplicationDbContext з DbSet<User> та налаштувати в Startup/Program.
4. **Реалізація контролера**
   * У AccountController додати GET-дію Register() → View().
   * У POST-дії Register(RegisterViewModel model):
     1. Перевірити унікальність Name через \_db.Users.Any(...).
     2. Якщо ModelState не валідний → повернути View(model).
     3. Інакше – створити новий User, додати в \_db, зберегти зміни і редірект.
5. **Створення представлення**
   * У Register.cshtml описати форму з полями Name, Email, Phone, Password, ConfirmPassword.
   * Підключити asp-validation-for і \_ValidationScriptsPartial для клієнтської перевірки.
6. **Кастомізація UI та поведінки**
   * Додати стилі у site.css для форми, полів та помилок.
   * За потреби прописати невеликий jQuery-скрипт для переформатування тексту помилок.

## **Функціональність**

* Користувач переходить на сторінку реєстрації і заповнює п’ять полів.
* На клієнті (через Unobtrusive Validation) і на сервері (DataAnnotations + додаткова перевірка унікальності) перевіряються:
  + Ім’я (до 15 символів, лише латиниця і цифри, унікальне).
  + Email (формат адреси).
  + Телефон (коректний формат).
  + Пароль (мінімум 12 символів, обов’язковий спеціальний символ) та його підтвердження.
* Якщо є помилки → вони відображаються під відповідними полями, користувач виправляє і знову надсилає форму.
* При успішній валідації дані зберігаються в таблицю Users, і користувач перенаправляється (наприклад, на сторінку підтвердження або назад до форми з повідомленням про успіх).

## **Висновок**

У рамках завдання Task\_2 реалізовано повноцінну функцію реєстрації користувачів із двома рівнями валідації (клієнтська та серверна) й перевіркою унікальності імені. Інтерфейс надає зручний і зрозумілий зворотний зв’язок у випадку помилок, а використання Entity Framework Core спрощує роботу з БД. Така реалізація підвищує надійність та юзабіліті реєстрації в умовній веб-системі.

# **Завдання 3**

Додати до проекту функцію авторизації користувачів умовної WEB-си-стеми із використанням дизайну реєстраційної форми з минулого завдання відповідного варіанту. Авторизаційна форма має складатися з двох полів: Username та Password, до яких користувач має ввести відпо-відну інформацію. У випадку коректного вводу система має підтвердити авторизацію користувача, у протилежному випадку повернутися до назад форми та видати повідомлення про помилку в позиції згідно до свого варіанта.

|  |  |
| --- | --- |
| **№ варіанту** | **Відносна локація (сірий відтінок) повідомлень про помилки** |
| 1 | 2 |
| 12 |  |

**Текст програми:**

**Account/Register.cshtml:**

@model Task\_3.Models.RegisterViewModel

@{

ViewData["Title"] = "Register"; // Встановлюємо заголовок для сторінки

}

<div class="form-card animate\_\_animated animate\_\_fadeIn">

<h2>Register</h2> <!-- Заголовок форми -->

<form asp-action="Register" method="post" novalidate> <!-- Форма реєстрації -->

<div class="form-material position-relative mb-4">

<span asp-validation-for="Name" class="error-message"></span> <!-- Показуємо повідомлення про помилку для поля Name -->

<label asp-for="Name">Name</label> <!-- Лейбл для поля Name -->

<input asp-for="Name" class="form-control" /> <!-- Поле для введення імені -->

</div>

<div class="form-material position-relative mb-4">

<span asp-validation-for="Email" class="error-message"></span> <!-- Показуємо повідомлення про помилку для поля Email -->

<label asp-for="Email">Email</label> <!-- Лейбл для поля Email -->

<input asp-for="Email" class="form-control" /> <!-- Поле для введення Email -->

</div>

<div class="form-material position-relative mb-4">

<span asp-validation-for="Phone" class="error-message"></span> <!-- Показуємо повідомлення про помилку для поля Phone -->

<label asp-for="Phone">Phone</label> <!-- Лейбл для поля Phone -->

<input asp-for="Phone" class="form-control" /> <!-- Поле для введення Phone -->

</div>

<div class="form-material position-relative mb-4">

<span asp-validation-for="Password" class="error-message"></span> <!-- Показуємо повідомлення про помилку для поля Password -->

<label asp-for="Password">Password</label> <!-- Лейбл для поля Password -->

<input asp-for="Password" type="password" class="form-control" /> <!-- Поле для введення пароля -->

</div>

<div class="form-material position-relative mb-4">

<span asp-validation-for="ConfirmPassword" class="error-message"></span> <!-- Показуємо повідомлення про помилку для поля ConfirmPassword -->

<label asp-for="ConfirmPassword">Confirm Password</label> <!-- Лейбл для підтвердження пароля -->

<input asp-for="ConfirmPassword" type="password" class="form-control" /> <!-- Поле для підтвердження пароля -->

</div>

<button type="submit" class="btn-register">REGISTER</button> <!-- Кнопка для реєстрації -->

</form>

</div>

@section Scripts {

@{

await Html.RenderPartialAsync("\_ValidationScriptsPartial"); // Підключення скриптів для валідації

}

<script>

$(function() {

var form = $('form');

form.removeData('validator').removeData('unobtrusiveValidation'); // Перезавантаження налаштувань валідації

$.validator.setDefaults({

showErrors: function(errorMap, errorList) {

this.defaultShowErrors(); // Показуємо помилки за замовчуванням

$('.error-message').each(function() {

var txt = $(this).text().trim(); // Обробляємо текст помилки

if (txt) {

var parts = txt.split(' '); // Розділяємо текст на частини

$(this).html(parts.join('<br/>')); // Виводимо помилки через <br/>

}

});

}

});

$.validator.unobtrusive.parse(form); // Активуємо валідацію

});

</script>

}

**Account/Login.cshtml:**

@model Task\_3.Models.LoginViewModel

@{

ViewData["Title"] = "Login"; // Встановлюємо заголовок для сторінки

}

<div class="form-card animate\_\_animated animate\_\_fadeIn">

<h2>Login</h2> <!-- Заголовок форми -->

<form asp-action="Login" method="post" novalidate> <!-- Форма для входу -->

<div class="form-material position-relative mb-4">

<span asp-validation-for="Name" class="error-message"></span> <!-- Показуємо повідомлення про помилку для поля Name -->

<label asp-for="Name">Name</label> <!-- Лейбл для поля Name -->

<input asp-for="Name" class="form-control" /> <!-- Поле для введення імені -->

</div>

<div class="form-material position-relative mb-4">

<span asp-validation-for="Password" class="error-message"></span> <!-- Показуємо повідомлення про помилку для поля Password -->

<label asp-for="Password">Password</label> <!-- Лейбл для поля Password -->

<input asp-for="Password" type="password" class="form-control" /> <!-- Поле для введення пароля -->

</div>

<span asp-validation-summary="ModelOnly" class="text-danger mb-3 d-block"></span> <!-- Підсумок помилок -->

<button type="submit" class="btn-register">LOGIN</button> <!-- Кнопка для входу -->

</form>

</div>

@section Scripts {

@{

await Html.RenderPartialAsync("\_ValidationScriptsPartial"); // Підключення скриптів для валідації

}

<script>

$(function() {

var form = $('form');

form.removeData('validator').removeData('unobtrusiveValidation'); // Перезавантаження налаштувань валідації

$.validator.setDefaults({

showErrors: function(errorMap, errorList) {

this.defaultShowErrors(); // Показуємо помилки за замовчуванням

$('.error-message').each(function() {

var txt = $(this).text().trim(); // Обробляємо текст помилки

if (txt) {

var parts = txt.split(' '); // Розділяємо текст на частини

$(this).html(parts.join('<br/>')); // Виводимо помилки через <br/>

}

});

}

});

$.validator.unobtrusive.parse(form); // Активуємо валідацію

});

</script>

}

**Models/User.cs:**

namespace Task\_3.Models

{

public class User

{

public int Id { get; set; } // Унікальний ідентифікатор

public string Name { get; set; } = null!; // Ім’я користувача

public string Email { get; set; } = null!; // Email користувача

public string Phone { get; set; } = null!; // Номер телефону користувача

public string Password { get; set; } = null!; // Пароль користувача

}

}

**Models/RegisterViewModel.cs:**

using System.ComponentModel.DataAnnotations;

namespace Task\_3.Models

{

public class RegisterViewModel

{

[Required(ErrorMessage = "Enter Name")]

[StringLength(15, ErrorMessage = "Max 15 chars")]

[RegularExpression("^[A-Za-z0-9]+$", ErrorMessage = "Only Latin & digits")]

public string Name { get; set; } = null!; // Валідація для поля Name

[Required(ErrorMessage = "Enter Email")]

[EmailAddress(ErrorMessage = "Invalid Email")]

public string Email { get; set; } = null!; // Валідація для поля Email

[Required(ErrorMessage = "Enter Phone")]

[Phone(ErrorMessage = "Invalid Phone")]

public string Phone { get; set; } = null!; // Валідація для поля Phone

[Required(ErrorMessage = "Enter Password")]

[MinLength(12, ErrorMessage = "At least 12 chars")]

[RegularExpression(@"^(?=.\*[!@#$%^&\*()]).+$", ErrorMessage = "Need special char")]

[DataType(DataType.Password)]

public string Password { get; set; } = null!; // Валідація для поля Password

[Required(ErrorMessage = "Confirm Password")]

[Compare("Password", ErrorMessage = "Passwords mismatch")]

[DataType(DataType.Password)]

public string ConfirmPassword { get; set; } = null!; // Валідація для поля ConfirmPassword

}

}

**Models/LoginViewModel.cs:**

using System.ComponentModel.DataAnnotations;

namespace Task\_3.Models

{

public class LoginViewModel

{

[Required(ErrorMessage = "Enter Name")]

public string Name { get; set; } = null!; // Валідація для поля Name

[Required(ErrorMessage = "Enter Password")]

[DataType(DataType.Password)]

public string Password { get; set; } = null!; // Валідація для поля Password

}

}

**Data/ApplicationDbContext.cs:**

using Microsoft.EntityFrameworkCore;

using Task\_3.Models;

namespace Task\_3.Data

{

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> opts)

: base(opts) { }

public DbSet<User> Users { get; set; } = null!; // Набір користувачів у БД

}

}

**Controllers/AccountController.cs:**

using Microsoft.AspNetCore.Mvc;

using Task\_3.Data;

using Task\_3.Models;

using System.Linq;

namespace Task\_3.Controllers

{

public class AccountController : Controller

{

private readonly ApplicationDbContext \_db;

public AccountController(ApplicationDbContext db) => \_db = db; // Впровадження контексту

// ==== REGISTER ====

[HttpGet]

public IActionResult Register() => View(); // Показати форму реєстрації

[HttpPost]

public IActionResult Register(RegisterViewModel vm)

{

if (\_db.Users.Any(u => u.Name == vm.Name))

ModelState.AddModelError(nameof(vm.Name), "This Name is taken"); // Перевірка на унікальність імені

if (!ModelState.IsValid)

return View(vm); // Якщо форма не валідна, повертаємо її

\_db.Users.Add(new User

{

Name = vm.Name,

Email = vm.Email,

Phone = vm.Phone,

Password = vm.Password

});

\_db.SaveChanges(); // Зберігаємо новий користувач у базі даних

return RedirectToAction("Login"); // Перенаправляємо на сторінку входу

}

// ==== LOGIN ====

[HttpGet]

public IActionResult Login() => View(); // Показати форму входу

[HttpPost]

public IActionResult Login(LoginViewModel vm)

{

if (!ModelState.IsValid)

return View(vm); // Якщо форма не валідна, повертаємо її

var user = \_db.Users

.FirstOrDefault(u => u.Name == vm.Name && u.Password == vm.Password); // Перевірка на існування користувача

if (user == null)

{

ModelState.AddModelError(string.Empty, "Invalid credentials"); // Якщо користувача немає

return View(vm);

}

HttpContext.Session.SetString("UserName", user.Name); // Зберігаємо ім’я користувача в сесії

return RedirectToAction("Index", "Home"); // Перенаправляємо на домашню сторінку

}

}

}

**\_Layout.cshtml:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width,initial-scale=1" />

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

<!-- Bootstrap 5 -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" />

<!-- Animate.css -->

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css" />

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

</head>

<body>

<div class="d-flex justify-content-center align-items-center min-vh-100">

@RenderBody() <!-- Відображення основного вмісту сторінки -->

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>

<script src="~/lib/jquery/dist/jquery.min.js"></script>

<script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>

<script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.js"></script>

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

</body>

</html>

**site.css:**

/\* фон \*/

body {

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

background-size: cover;

}

/\* білий блок форми \*/

.form-card {

background: #fff;

padding: 2rem;

border-radius: 8px;

width: 100%;

max-width: 480px;

box-shadow: 0 4px 16px rgba(0,0,0,0.1);

}

/\* заголовок форми \*/

.form-card h2 {

margin-bottom: 1.5rem;

font-weight: 500;

}

/\* лише нижня лінія \*/

.form-material .form-control {

border: none;

border-bottom: 1px solid #ccc;

border-radius: 0;

background: transparent;

box-shadow: none;

}

/\* підсвічування фокусу \*/

.form-material .form-control:focus {

outline: none;

border-bottom-color: #1abc9c;

}

/\* стиль лейбла \*/

.form-material label {

color: #888;

font-size: 0.9rem;

margin-top: 0.25rem;

margin-bottom: 0.25rem;

}

/\* стиль повідомлення про помилку \*/

.error-message {

display: block;

position: static;

margin-bottom: 0.25rem;

color: #e74c3c;

font-size: 0.8rem;

line-height: 1.2;

animation: fade-in 0.4s;

}

/\* кнопка \*/

.btn-register {

width: 30%;

background-color: #1abc9c;

border: none;

padding: .75rem;

text-transform: uppercase;

letter-spacing: 1px;

}

/\* ховер ефект кнопки \*/

.btn-register:hover {

background-color: #16a085;

}

@keyframes fade-in {

from {

opacity: 0;

}

to {

opacity: 1;

}

}

## **Результати роботи програми:**

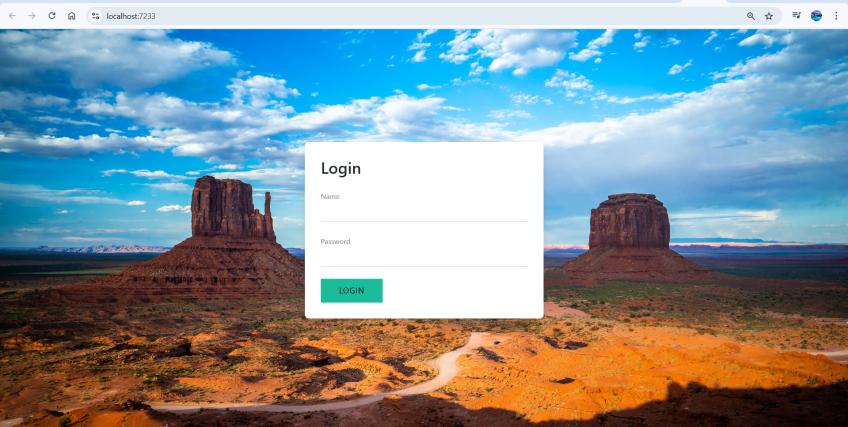


Рисунок 3.1. Загальний вигляд форми авторизації

Изображение выглядит как облако, небо, снимок экрана, пейзаж

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.2. Вигляд помилок

Изображение выглядит как облако, небо, снимок экрана, пейзаж

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.3. Введення даних, які вже є зареєстрованимим в БД

Изображение выглядит как облако, небо, на открытом воздухе, одинокая возвышенность

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.4. Результат успішної авторизації користувача

## **Алгоритм побудови кодів**

1. **Модель даних і контекст EF Core**
   * Створити клас User (Id, Name, Email, Phone, Password).
   * Налаштувати ApplicationDbContext з DbSet<User>.
2. **ViewModel-и з валідацією**
   * RegisterViewModel із DataAnnotations для полів Name, Email, Phone, Password, ConfirmPassword.
   * LoginViewModel лише з полями Name і Password та [Required].
3. **Контролер AccountController**
   * **Register**
     + **GET**: показати форму.
     + **POST**:
       1. Перевірити унікальність Name в БД.
       2. Якщо є помилки в ModelState → повернути форму з повідомленнями.
       3. Інакше – створити User, додати в контекст, зберегти зміни і перенаправити на вхід.
   * **Login**
     + **GET**: показати форму входу.
     + **POST**:
       1. Якщо ModelState не валідний → повернути форму.
       2. Спробувати знайти в БД користувача з таким Name і Password.
       3. Якщо не знайдено → додати ModelError і повернути форму.
       4. Інакше – зберегти ім’я в сесії і перенаправити на головну.
4. **Представлення (Views)**
   * **Register.cshtml** і **Login.cshtml**:
     + Поля з asp-for, спани з asp-validation-for, кнопка Submit.
     + Підключити \_ValidationScriptsPartial для клієнтської валідації.
     + Невеликий jQuery-скрипт для форматування повідомлень помилок (розрив рядків через <br/>).
5. **Інтерфейс і стилі**
   * Використати site.css зі стилями для .form-card, .form-material, .error-message, .btn-register.
   * Підключити Bootstrap та Animate.css у \_Layout.cshtml.

## **Функціональність**

* **Реєстрація**
  1. Користувач заповнює усі п’ять полів та натискає “REGISTER”.
  2. Система перевіряє:
     + Унікальність імені.
     + Формат Email та Phone.
     + Довжину та склад пароля і його підтвердження.
  3. У разі помилок – відобразити під відповідними полями.
  4. Якщо всі дані коректні – зберегти в БД і переадресувати на форму входу.
* **Авторизація**
  1. Користувач вводить Name та Password і натискає “LOGIN”.
  2. Система перевіряє наявність запису в таблиці Users із такими обліковими даними.
  3. Якщо невірно – показати загальне повідомлення “Invalid credentials” у вказаному місці.
  4. Якщо вірно – зберегти ім’я в сесії і перенаправити на головну сторінку.

## **Висновок**

Реалізовано двоетапну систему автентифікації: спочатку реєстрація з повною валідацією, потім вхід із перевіркою облікових даних. Використання EF Core спрощує роботу з БД, а DataAnnotations та Unobtrusive Validation забезпечують зрозумілий зворотній зв’язок для користувача. Збереження імені в сесії дає змогу підтримувати стан авторизованого користувача на всіх сторінках. Це рішення відповідає завданню Task\_3 та готове для подальшого розширення (ролі, захист маршрутів тощо).

# **Завдання 4**

Об'єднати функції реєстрації та авторизації та зробити загальний вхідний функціонал умовної WEВ-системи. Перевірити коректність його роботи.

## **Текст програми:**

**Account/Register.cshtml:**

@model Task\_4.Models.RegisterViewModel

@{

ViewData["Title"] = "Register"; // Встановлюємо заголовок сторінки

}

<div class="form-card animate\_\_animated animate\_\_fadeIn">

<h2>Register</h2> <!-- Заголовок для форми реєстрації -->

<form asp-action="Register" method="post" novalidate> <!-- Форма для реєстрації -->

<div class="form-material mb-4">

<span asp-validation-for="Name" class="error-message"></span> <!-- Помилка для поля Name -->

<label asp-for="Name">Name</label> <!-- Лейбл для поля Name -->

<input asp-for="Name" class="form-control" /> <!-- Поле для введення імені -->

</div>

<div class="form-material mb-4">

<span asp-validation-for="Email" class="error-message"></span> <!-- Помилка для поля Email -->

<label asp-for="Email">Email</label> <!-- Лейбл для поля Email -->

<input asp-for="Email" class="form-control" /> <!-- Поле для введення Email -->

</div>

<div class="form-material mb-4">

<span asp-validation-for="Phone" class="error-message"></span> <!-- Помилка для поля Phone -->

<label asp-for="Phone">Phone</label> <!-- Лейбл для поля Phone -->

<input asp-for="Phone" class="form-control" /> <!-- Поле для введення телефону -->

</div>

<div class="form-material mb-4">

<span asp-validation-for="Password" class="error-message"></span> <!-- Помилка для поля Password -->

<label asp-for="Password">Password</label> <!-- Лейбл для поля Password -->

<input asp-for="Password" type="password" class="form-control" /> <!-- Поле для введення пароля -->

</div>

<div class="form-material mb-4">

<span asp-validation-for="ConfirmPassword" class="error-message"></span> <!-- Помилка для поля ConfirmPassword -->

<label asp-for="ConfirmPassword">Confirm Password</label> <!-- Лейбл для підтвердження пароля -->

<input asp-for="ConfirmPassword" type="password" class="form-control" /> <!-- Поле для підтвердження пароля -->

</div>

<button type="submit" class="btn-register">REGISTER</button> <!-- Кнопка для реєстрації -->

</form>

</div>

@section Scripts {

@{

await Html.RenderPartialAsync("\_ValidationScriptsPartial"); // Підключаємо частковий скрипт для валідації

}

<script>

$(function() {

var form = $('form');

form.removeData('validator').removeData('unobtrusiveValidation'); // Перезавантажуємо валідацію

$.validator.setDefaults({

showErrors: function(errorMap, errorList) {

this.defaultShowErrors(); // Викликаємо стандартний метод для показу помилок

$('.error-message').each(function() {

var txt = $(this).text().trim(); // Очищаємо пробіли в тексті помилки

if (txt) {

var parts = txt.split(' '); // Розділяємо текст на частини

$(this).html(parts.join('<br/>')); // Виводимо помилки через <br/>

}

});

}

});

$.validator.unobtrusive.parse(form); // Запускаємо валідацію

});

</script>

}

**Account/Login.cshtml:**

@model Task\_4.Models.LoginViewModel

@{

ViewData["Title"] = "Login"; // Встановлюємо заголовок сторінки

}

<div class="form-card animate\_\_animated animate\_\_fadeIn">

<h2>Login</h2> <!-- Заголовок форми входу -->

<form asp-action="Login" method="post" novalidate> <!-- Форма для входу -->

<div class="form-material mb-4">

<span asp-validation-for="Name" class="error-message"></span> <!-- Помилка для поля Name -->

<label asp-for="Name">Name</label> <!-- Лейбл для поля Name -->

<input asp-for="Name" class="form-control" /> <!-- Поле для введення імені -->

</div>

<div class="form-material mb-4">

<span asp-validation-for="Password" class="error-message"></span> <!-- Помилка для поля Password -->

<label asp-for="Password">Password</label> <!-- Лейбл для поля Password -->

<input asp-for="Password" type="password" class="form-control" /> <!-- Поле для введення пароля -->

</div>

<span asp-validation-summary="ModelOnly" class="text-danger mb-3 d-block"></span> <!-- Підсумок помилок -->

<button type="submit" class="btn-register">LOGIN</button> <!-- Кнопка для входу -->

</form>

</div>

@section Scripts {

@{

await Html.RenderPartialAsync("\_ValidationScriptsPartial"); // Підключаємо частковий скрипт для валідації

}

<script>

$(function() {

var form = $('form');

form.removeData('validator').removeData('unobtrusiveValidation'); // Перезавантажуємо валідацію

$.validator.setDefaults({

showErrors: function(errorMap, errorList) {

this.defaultShowErrors(); // Викликаємо стандартний метод для показу помилок

$('.error-message').each(function() {

var txt = $(this).text().trim(); // Очищаємо пробіли в тексті помилки

if (txt) {

var parts = txt.split(' '); // Розділяємо текст на частини

$(this).html(parts.join('<br/>')); // Виводимо помилки через <br/>

}

});

}

});

$.validator.unobtrusive.parse(form); // Запускаємо валідацію

});

</script>

}

**Models/User.cs:**

namespace Task\_4.Models

{

public class User

{

public int Id { get; set; } // Ідентифікатор користувача

public string Name { get; set; } = null!; // Ім’я користувача

public string Email { get; set; } = null!; // Email користувача

public string Phone { get; set; } = null!; // Телефон користувача

public string Password { get; set; } = null!; // Пароль користувача

}

}

**Models/RegisterViewModel.cs:**

using System.ComponentModel.DataAnnotations;

namespace Task\_4.Models

{

public class RegisterViewModel

{

[Required(ErrorMessage = "Enter Name")]

[StringLength(15, ErrorMessage = "Max 15 chars")]

[RegularExpression("^[A-Za-z0-9]+$", ErrorMessage = "Only Latin & digits")]

public string Name { get; set; } = null!; // Валідація для поля Name

[Required(ErrorMessage = "Enter Email")]

[EmailAddress(ErrorMessage = "Invalid Email")]

public string Email { get; set; } = null!; // Валідація для поля Email

[Required(ErrorMessage = "Enter Phone")]

[Phone(ErrorMessage = "Invalid Phone")]

public string Phone { get; set; } = null!; // Валідація для поля Phone

[Required(ErrorMessage = "Enter Password")]

[MinLength(12, ErrorMessage = "At least 12 chars")]

[RegularExpression(@"^(?=.\*[!@#$%^&\*()]).+$", ErrorMessage = "Need special char")]

[DataType(DataType.Password)]

public string Password { get; set; } = null!; // Валідація для поля Password

[Required(ErrorMessage = "Confirm Password")]

[Compare("Password", ErrorMessage = "Passwords mismatch")]

[DataType(DataType.Password)]

public string ConfirmPassword { get; set; } = null!; // Валідація для поля ConfirmPassword

}

}

**Models/LoginViewModel.cs:**

using System.ComponentModel.DataAnnotations;

namespace Task\_4.Models

{

public class LoginViewModel

{

[Required(ErrorMessage = "Enter Name")]

public string Name { get; set; } = null!; // Валідація для поля Name

[Required(ErrorMessage = "Enter Password")]

[DataType(DataType.Password)]

public string Password { get; set; } = null!; // Валідація для поля Password

}

}

**Data/ApplicationDbContext.cs:**

using Microsoft.EntityFrameworkCore;

using Task\_4.Models;

namespace Task\_4.Data

{

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> opts)

: base(opts) { } // Конструктор з параметрами

public DbSet<User> Users { get; set; } = null!; // Таблиця користувачів у БД

}

}

**Controllers/AccountController.cs:**

using Microsoft.AspNetCore.Mvc;

using Task\_4.Data;

using Task\_4.Models;

using System.Linq;

namespace Task\_4.Controllers

{

public class AccountController : Controller

{

private readonly ApplicationDbContext \_db;

public AccountController(ApplicationDbContext db) => \_db = db; // Впровадження контексту БД

// REGISTER

[HttpGet]

public IActionResult Register() => View(); // Відображення форми реєстрації

[HttpPost]

public IActionResult Register(RegisterViewModel vm)

{

if (\_db.Users.Any(u => u.Name == vm.Name))

ModelState.AddModelError(nameof(vm.Name), "This Name is taken"); // Перевірка унікальності імені

if (!ModelState.IsValid)

return View(vm); // Повернення форми з помилками

\_db.Users.Add(new User

{

Name = vm.Name,

Email = vm.Email,

Phone = vm.Phone,

Password = vm.Password

});

\_db.SaveChanges(); // Збереження даних користувача в БД

return RedirectToAction("Login"); // Перенаправлення на форму входу

}

// LOGIN

[HttpGet]

public IActionResult Login() => View(); // Відображення форми входу

[HttpPost]

public IActionResult Login(LoginViewModel vm)

{

if (!ModelState.IsValid)

return View(vm); // Якщо форма не валідна, повертаємо її

var user = \_db.Users

.FirstOrDefault(u => u.Name == vm.Name && u.Password == vm.Password); // Перевірка користувача

if (user == null)

{

ModelState.AddModelError(string.Empty, "Invalid credentials"); // Якщо не знайдено користувача

return View(vm);

}

HttpContext.Session.SetString("UserName", user.Name); // Збереження імені користувача в сесії

return RedirectToAction("Index", "Home"); // Перенаправлення на головну сторінку

}

// LOGOUT

public IActionResult Logout()

{

HttpContext.Session.Clear(); // Очистка сесії

return RedirectToAction("Login"); // Перенаправлення на форму входу

}

}

}

**\_Layout.cshtml:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width,initial-scale=1" />

<title>@ViewData["Title"] – Task\_4</title> <!-- Динамічний заголовок -->

<!-- Bootstrap 5 -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"

rel="stylesheet" />

<!-- Animate.css -->

<link rel="stylesheet"

href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css" />

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

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-light bg-light mb-4">

<div class="container-fluid">

<a class="navbar-brand" href="#">Lab\_12</a> <!-- Назва проекту -->

<ul class="navbar-nav me-auto">

<li class="nav-item"><a class="nav-link" asp-action="Register" asp-controller="Account">Register</a></li> <!-- Посилання на реєстрацію -->

<li class="nav-item"><a class="nav-link" asp-action="Login" asp-controller="Account">Login</a></li> <!-- Посилання на вхід -->

</ul>

<span class="navbar-text">

@Context.Session.GetString("UserName") <!-- Виведення імені користувача, якщо в сесії -->

</span>

</div>

</nav>

<div class="d-flex justify-content-center align-items-center min-vh-100">

@RenderBody() <!-- Основний вміст сторінки -->

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>

<script src="~/lib/jquery/dist/jquery.min.js"></script>

<script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>

<script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.js"></script>

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

</body>

</html>

**site.css:**

/\* фон сторінки \*/

body {

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

background-size: cover;

}

/\* стилі для картки форми \*/

.form-card {

background: #fff;

padding: 2rem;

border-radius: 8px;

width: 100%;

max-width: 480px;

box-shadow: 0 4px 16px rgba(0,0,0,0.1);

}

.form-card h2 {

margin-bottom: 1.5rem;

font-weight: 500;

}

/\* стилі для полів вводу \*/

.form-material .form-control {

border: none;

border-bottom: 1px solid #ccc;

border-radius: 0;

background: transparent;

box-shadow: none;

}

/\* стилі для полів вводу при фокусі \*/

.form-material .form-control:focus {

outline: none;

border-bottom-color: #1abc9c;

}

/\* стилі для лейблів полів \*/

.form-material label {

display: block;

color: #888;

font-size: 0.9rem;

margin-top: 0.25rem;

margin-bottom: 0.25rem;

}

/\* стилі для повідомлень про помилки \*/

.error-message {

display: block;

position: static;

margin-bottom: 0.25rem;

color: #e74c3c;

font-size: 0.8rem;

line-height: 1.2;

animation: fade-in 0.4s;

}

/\* стилі для кнопки реєстрації \*/

.btn-register {

width: 30%;

background-color: #1abc9c;

border: none;

padding: .75rem;

text-transform: uppercase;

letter-spacing: 1px;

}

/\* стилі для кнопки при наведенні \*/

.btn-register:hover {

background-color: #16a085;

}

/\* анімація для плавної появи \*/

@keyframes fade-in {

from {

opacity: 0;

}

to {

opacity: 1;

}

}

## **Результати роботи програми:**

Изображение выглядит как облако, небо, снимок экрана, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.1. Загальний вигляд форми реєстрації

Изображение выглядит как облако, небо, снимок экрана, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.2. Вигляд помилок у формі реєстрації

Изображение выглядит как облако, небо, снимок экрана, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.3. Введення даних у формі реєстрації

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.4. Перевірка введених даних у таблиці БД

Изображение выглядит как облако, небо, снимок экрана, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.5. Загальний вигляд форми авторизації

Изображение выглядит как облако, небо, снимок экрана, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.6. Вигляд помилок у формі авторизації

Изображение выглядит как облако, небо, снимок экрана, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.7. Введення зареєстрованих даних у форму авторизації

Изображение выглядит как облако, текст, небо, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.8. Успішна авторизації користувача

## **Алгоритм побудови кодів**

1. **Модель і контекст**
   * Описати клас User (Id, Name, Email, Phone, Password).
   * Налаштувати ApplicationDbContext з DbSet<User>.
2. **ViewModel-и з валідацією**
   * RegisterViewModel (Name, Email, Phone, Password, ConfirmPassword) із DataAnnotations.
   * LoginViewModel (Name, Password) із [Required].
3. **Контролер AccountController**
   * **Register (GET/POST)**: перевірка унікальності імені, валідація, збереження в БД, редірект на Login.
   * **Login (GET/POST)**: перевірка ModelState, пошук користувача за ім’ям і паролем, у випадку успіху запис у сесію, інакше – повідомлення про помилку.
   * **Logout**: очистка сесії та перенаправлення на Login.
4. **Подання (Views)**
   * **Register.cshtml** та **Login.cshtml**: форми з asp-for, asp-validation-for та \_ValidationScriptsPartial.
   * JQuery-скрипт для форматування рядків помилок через <br/>.
5. **Універсальний макет і навігація**
   * У \_Layout.cshtml додати навбар із посиланнями на Register, Login, а також відображенням поточного UserName із сесії.
6. **Стилі**
   * Використати site.css для .form-card, .form-material, .error-message, .btn-register.
   * Підключити Bootstrap та Animate.css у заголовку.

## **Функціональність**

* **Реєстрація**: користувач вводить усі поля → серверна й клієнтська валідація → збереження в БД → перенаправлення на вхід.
* **Авторизація**: користувач вводить Name і Password → пошук у БД → при невдачі – повідомлення “Invalid credentials”, інакше – збереження імені в сесії → редірект на головну.
* **Вихід**: за кліком “Logout” система очищає сесію та повертає користувача на форму входу.
* **Навігація**: у шапці завжди видно, чи авторизований користувач, і його ім’я, а також посилання на Register/Login.

## **Висновок**

Об’єднаний функціонал покриває і реєстрацію, і вхід/вихід користувача в єдиному інтерфейсі. За рахунок поєднання клієнтської та серверної валідації, перевірки унікальності та сесійного зберігання стану користувача система працює коректно та зручно для кінцевого користувача. Це рішення закладає основу для подальшого розширення (ролі, авторизація на рівні контролерів тощо).

# **Загальний висновок**

У результаті виконання лабораторної роботи №8 були досягнуті такі результати:

1. **Створено MVC-застосунок** з підтримкою реєстрації та авторизації користувачів на базі ASP.NET Core MVC.
2. **Реалізовано реєстрацію користувачів** з перевіркою правильності введення даних (на клієнті та сервері).
3. **Реалізовано авторизацію користувачів** із збереженням сесії після успішного входу.
4. **Створено базу даних та налаштовано таблицю користувачів** для збереження реєстраційної інформації.
5. **Виконано об'єднання функцій реєстрації та авторизації** в єдиний вхідний функціонал веб-системи.
6. **Налаштовано перевірку унікальності імені** користувача під час реєстрації.
7. **Оформлено інтерфейс користувача** з використанням Bootstrap та кастомних стилів.
8. **Перевірено коректність роботи застосунку** шляхом тестування на реальних сценаріях введення правильних і неправильних даних.

Таким чином, у ході лабораторної роботи була створена повноцінна система реєстрації та авторизації користувачів із зручним інтерфейсом та коректною обробкою помилок. Отримано практичні навички роботи з ASP.NET Core MVC, налаштування Entity Framework Core для взаємодії з базою даних, а також створення адаптивного та інтерактивного користувацького інтерфейсу. Виконане завдання повністю відповідає вимогам варіанта та готове для подальшого розвитку, наприклад, додавання ролей користувачів, захисту маршрутів або реалізації складніших механізмів автентифікації.