Міністерство освіти України

Національний технічний університет "ХПІ"

кафедра "Інформатики та інтелектуальної власності"

**Звіт**

**Лабораторна робота 8**

з дисципліни "Проєктування серверних застосунків"

Виконав: студент групи КН-922В

Шершньов М.О.

Перевірив:

Івашко А.В.

Харків 2025

Зміст

[Завдання 3](#_Toc195439234)

[Класи accountregistration, accountauth, registrationandauth 5](#_Toc195439235)

[Клас AccountRegistration 5](#_Toc195439236)

[Клас AccountAuth 8](#_Toc195439237)

[Клас RegistrationAndAuth 10](#_Toc195439238)

[Висновок 14](#_Toc195439239)

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

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

Завдання

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

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

Мій варіант:

Изображение выглядит как текст, чек, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

3. Додати до проекту функцію авторизації користувачів умовної WEB-системи із використанням дизайну реєстраційної форми з минулого за-вдання відповідного варіанту. Авторизаційна форма має складатися з двох полів: Username та Password, до яких користувач має ввести відповідну 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

Мій варіант:



Изображение выглядит как снимок экрана, линия, Прямоугольник, прямоугольный

Контент, сгенерированный ИИ, может содержать ошибки.

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

Класи для роботи:

1. «Реєстрація акаунту» - **AccountRegistration**
2. «Авторизація акаунту» – **AccountAuth**
3. «Реєстрація та авторизація» – **RegistrationAndAuth**

Класи accountregistration, accountauth, registrationandauth

Клас AccountRegistration

Клас **AccountRegistration** відповідає за обробку реєстрації нового облікового запису користувача у веб-додатку.

Основні функції класу:

1. Приймає дані користувача з форми реєстрації.
2. Перевіряє, чи ім’я користувача вже існує у базі даних.
3. Виконує перевірку коректності введених даних (валідація моделі).
4. У разі успішної перевірки — створює нового користувача та зберігає його у базу.
5. У разі помилок — повертає форму з повідомленням про помилку.
6. Після успішної реєстрації перенаправляє на сторінку підтвердження (Success).

using Microsoft.AspNetCore.Mvc;

using Task\_1.Models;

public class HomeController : Controller

{

private readonly ApplicationDbContext \_context;

public HomeController(ApplicationDbContext context)

{

\_context = context;

}

[HttpGet]

public IActionResult Register() => View();

[HttpPost]

public IActionResult Register(UserRegistrationModel model)

{

if (\_context.Users.Any(u => u.UserName == model.UserName))

{

ModelState.AddModelError("UserName", "Користувач вже існує.");

}

if (!ModelState.IsValid)

return View(model);

var user = new User

{

UserName = model.UserName,

Password = model.Password,

Email = model.Email,

BirthDate = new DateTime(model.Year, model.Month, model.Day)

};

\_context.Users.Add(user);

\_context.SaveChanges();

return RedirectToAction("Success");

}

public IActionResult Success() => View();

}

using System.ComponentModel.DataAnnotations;

namespace Task\_1.Models

{

public class UserRegistrationModel

{

[Required(ErrorMessage = "Поле 'Ім'я користувача' є обов'язковим.")]

[MaxLength(15)]

[RegularExpression(@"^[a-zA-Z0-9]+$", ErrorMessage = "Тільки латинські літери та цифри.")]

public string? UserName { get; set; }

[Required(ErrorMessage = "Поле 'Пароль' є обов'язковим.")]

[MinLength(12, ErrorMessage = "Пароль повинен містити щонайменше 12 символів.")]

[RegularExpression(@"^(?=.\*[!@#$%^&\*()\_**\-**+=]).+$", ErrorMessage = "Пароль має містити спецсимвол.")]

public string? Password { get; set; }

[Compare("Password", ErrorMessage = "Паролі не співпадають.")]

public string? ConfirmPassword { get; set; }

[Required(ErrorMessage = "Поле 'Електронна пошта' є обов'язковим.")]

[EmailAddress(ErrorMessage = "Некоректна пошта.")]

public string? Email { get; set; }

[Range(1900, 2025)]

public int Year { get; set; }

[Range(1, 12)]

public int Month { get; set; }

[Range(1, 31)]

public int Day { get; set; }

}

}

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

Контент, сгенерированный ИИ, может содержать ошибки.

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

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, Прямоугольник

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, Шрифт, линия, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунки 8.1-8.4 – Результат роботи класу AccountRegistration

Клас AccountAuth

Клас **AccountAuth** відповідає за авторизацію користувача у веб-додатку.

**Основні функції класу:**

1. Приймає дані для входу (логін і пароль) через форму.
2. Перевіряє, чи існує користувач з таким іменем у базі даних.
3. Перевіряє правильність введеного пароля.
4. У разі помилки — повертає користувача на сторінку входу з відповідними повідомленнями.
5. У разі успішної авторизації — зберігає дані користувача у TempData.
6. Перенаправляє користувача на сторінку Welcome з відображенням його облікових даних.

using Microsoft.AspNetCore.Mvc;

using Task\_2.Models;

public class HomeController : Controller

{

private readonly ApplicationDbContext \_context;

public HomeController(ApplicationDbContext context)

{

\_context = context;

}

[HttpGet]

public IActionResult Register() => View();

[HttpPost]

public IActionResult Register(UserRegistrationModel model)

{

if (\_context.Users.Any(u => u.UserName == model.UserName))

{

ModelState.AddModelError("UserName", "Користувач вже існує.");

}

if (!ModelState.IsValid)

return View(model);

var user = new User

{

UserName = model.UserName,

Password = model.Password,

Email = model.Email,

BirthDate = new DateTime(model.Year, model.Month, model.Day)

};

\_context.Users.Add(user);

\_context.SaveChanges();

return RedirectToAction("Success");

}

public IActionResult Success() => View();

[HttpGet]

public IActionResult Login() => View();

[HttpPost]

public IActionResult Login(LoginModel model)

{

if (!ModelState.IsValid)

return View(model);

var user = \_context.Users.FirstOrDefault(u => u.UserName == model.Username);

if (user == null)

{

ViewBag.ErrorUsername = "Користувача не знайдено.";

return View(model);

}

if (user.Password != model.Password)

{

ViewBag.ErrorPassword = "Невірний пароль.";

return View(model);

}

TempData["UserName"] = user.UserName;

TempData["Email"] = user.Email;

TempData["BirthDate"] = user.BirthDate.ToString("yyyy-MM-dd");

return RedirectToAction("Welcome");

}

public IActionResult Welcome()

{

ViewBag.UserName = TempData["UserName"];

ViewBag.Email = TempData["Email"];

ViewBag.BirthDate = TempData["BirthDate"];

return View();

}

}

using System.ComponentModel.DataAnnotations;

namespace Task\_2.Models

{

public class LoginModel

{

[Required]

[MaxLength(15)]

public string? Username { get; set; }

[Required]

[MinLength(12)]

public string? Password { get; set; }

}

}

Изображение выглядит как текст, снимок экрана, число, Прямоугольник

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунки 8.5-8.6 – Результат роботи класу AccountAuth

Клас RegistrationAndAuth

Клас **RegistrationAndAuth** реалізує повний функціонал реєстрації та авторизації користувача у веб-додатку.

**Основні функції класу:**

* **Реєстрація користувача:**

1. Отримує дані з форми реєстрації.
2. Перевіряє, чи ім’я користувача вже існує у базі даних.
3. Валідує введені дані.
4. Якщо валідація успішна — створює нового користувача та зберігає його у базу.
5. Зберігає повідомлення про успішну реєстрацію у TempData.
6. Переадресовує користувача на сторінку входу.

* **Авторизація користувача:**

1. Приймає дані для входу з форми (Username, Password).
2. Перевіряє, чи існує користувач з таким ім’ям.
3. Перевіряє правильність пароля.
4. У разі помилки — повертає користувача на сторінку входу з повідомленням.
5. При успішному вході — зберігає дані користувача у TempData.
6. Переадресовує на сторінку Welcome, де відображається профіль користувача.

using Microsoft.AspNetCore.Mvc;

using Task\_3.Models;

public class HomeController : Controller

{

private readonly ApplicationDbContext \_context;

public HomeController(ApplicationDbContext context)

{

\_context = context;

}

[HttpGet]

public IActionResult Register() => View();

[HttpPost]

public IActionResult Register(UserRegistrationModel model)

{

if (\_context.Users.Any(u => u.UserName == model.UserName))

{

ModelState.AddModelError("UserName", "Користувач вже існує.");

return View(model);

}

if (!ModelState.IsValid)

return View(model);

var user = new User

{

UserName = model.UserName!,

Password = model.Password!,

Email = model.Email!,

BirthDate = new DateTime(model.Year, model.Month, model.Day)

};

\_context.Users.Add(user);

\_context.SaveChanges();

TempData["Success"] = "Реєстрація успішна! Увійдіть до системи.";

return RedirectToAction("Login");

}

[HttpGet]

public IActionResult Login() => View();

[HttpPost]

public IActionResult Login(LoginModel model)

{

if (!ModelState.IsValid)

return View(model);

var user = \_context.Users.FirstOrDefault(u => u.UserName == model.Username);

if (user == null)

{

ViewBag.ErrorUsername = "Користувача не знайдено.";

return View(model);

}

if (user.Password != model.Password)

{

ViewBag.ErrorPassword = "Невірний пароль.";

return View(model);

}

TempData["UserName"] = user.UserName;

TempData["Email"] = user.Email;

TempData["BirthDate"] = user.BirthDate.ToString("yyyy-MM-dd");

return RedirectToAction("Welcome");

}

public IActionResult Welcome()

{

ViewBag.UserName = TempData["UserName"];

ViewBag.Email = TempData["Email"];

ViewBag.BirthDate = TempData["BirthDate"];

return View();

}

}

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, Прямоугольник

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунки 8.7-8.9 – Результат роботи класу RegistrationAndAuth

Висновок

У ході виконання цієї лабораторної роботи я отримав практичні навички розробки реєстраційного та авторизаційного функціоналу у MVC ASP.NET WEB-застосунку. Було реалізовано окремі класи для управління обліковими записами: **AccountRegistration** — для створення нового користувача, **AccountAuth** — для перевірки облікових даних при вході, а також **RegistrationAndAuth**, який поєднує обидва процеси в єдиному контролері.

У процесі розробки я ознайомився з механізмами обробки форм, валідації даних на стороні сервера за допомогою атрибутів DataAnnotations, перевірки унікальності користувачів у базі даних, а також із принципами безпечного перенаправлення та передачі даних між сторінками за допомогою TempData. Було реалізовано представлення для кожного етапу: реєстрації, входу, підтвердження успішної реєстрації та відображення профілю користувача.

Ця робота дозволила мені глибше зрозуміти архітектуру MVC, навчитися створювати зручний та безпечний інтерфейс для користувачів, а також зміцнити навички у реалізації базового функціоналу сучасного веб-застосунку. Отримані знання є основою для побудови масштабованих, зручних та безпечних систем аутентифікації в реальних проєктах.