Міністерство освіти України

Національний технічний університет "ХПІ"

кафедра "Комп’ютерних наук та інформаційних технологій"

**Звіт**

**Лабораторна робота 10**

з дисципліни "Проєктування серверних застосунків"

Виконав: студент групи КН-922В

Мельник Б. В.

Перевірив:

Івашко А. В.

Харків 2025

**Зміст**

[Завдання 1 3](#_Toc197630468)

[Текст програми: 3](#_Toc197630469)

[Результати роботи програми: 40](#_Toc197630470)

[Алгоритм побудови кодів 42](#_Toc197630471)

[Функціональність 43](#_Toc197630472)

[Висновок 44](#_Toc197630473)

[Завдання 2 45](#_Toc197630474)

[Результати роботи програми: 54](#_Toc197630475)

[Алгоритм побудови кодів 55](#_Toc197630476)

[Функціональність 57](#_Toc197630477)

[Висновок 57](#_Toc197630478)

[Завдання 3 59](#_Toc197630479)

[Текст програми: 59](#_Toc197630480)

[Результати роботи програми: 62](#_Toc197630481)

[Алгоритм побудови кодів 64](#_Toc197630482)

[Функціональність 65](#_Toc197630483)

[Висновок 65](#_Toc197630484)

[Завдання 4 66](#_Toc197630485)

[Текст програми: 66](#_Toc197630486)

[Результати роботи програми: 70](#_Toc197630487)

[Алгоритм побудови кодів 72](#_Toc197630488)

[Функціональність 73](#_Toc197630489)

[Висновок 73](#_Toc197630490)

[Завдання 5 74](#_Toc197630491)

[Текст програми: 74](#_Toc197630492)

[Результати роботи програми: 80](#_Toc197630493)

[Алгоритм побудови кодів 83](#_Toc197630494)

[Функціональність 84](#_Toc197630495)

[Висновок 84](#_Toc197630496)

[Завдання 6 85](#_Toc197630497)

[Текст програми: 85](#_Toc197630498)

[Результати роботи програми: 111](#_Toc197630499)

[Алгоритм побудови кодів 115](#_Toc197630500)

[Функціональність 116](#_Toc197630501)

[Висновок 116](#_Toc197630502)

[Загальний висновок 117](#_Toc197630503)

**Тема:** Створення завершених MVC WEB-застосунків із обов'язковим стандарт-ним функціоналом, використовуючи можливості ASP.NET. Частина 2.

**Мета роботи:** придбати практику та досвід у створенні WEB-застосунків із стандартним набором сучасних функцій та можливостей за допомогою засобів ASP.NET.

**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**

Модифікувати проект завдання минулої роботи, додавши до нього комплекс контролерів, подання та необхідних таблиць БД, які дозволяли б проходити тестування зареєстрованих та авторизованих користувачів за напрямком згідно до власного варіанту.

|  |  |
| --- | --- |
| **№ варіанту** | **Напрямок тестування** |
| 12 | Історія України |

## **Текст програми:**

**SQL-запит:**

GO

USE TestetDB;

GO

CREATE TABLE Questions (

Id INT IDENTITY(1,1) PRIMARY KEY,

Text NVARCHAR(500) NOT NULL,

OptionA NVARCHAR(200) NOT NULL,

OptionB NVARCHAR(200) NOT NULL,

OptionC NVARCHAR(200) NOT NULL,

OptionD NVARCHAR(200) NOT NULL,

CorrectOption CHAR(1) NOT NULL

);

GO

CREATE TABLE TestResults (

Id INT IDENTITY(1,1) PRIMARY KEY,

UserName NVARCHAR(100) NOT NULL,

Score INT NOT NULL,

TakenAt DATETIME NOT NULL

);

GO

INSERT INTO Questions (Text, OptionA, OptionB, OptionC, OptionD, CorrectOption) VALUES

(N'Яка дата Першого Збору Української Центральної Ради?',

N'17 березня 1917',

N'7 березня 1917',

N'17 квітня 1917',

N'7 квітня 1917',

'A'),

(N'Хто очолив УНР у 1918 році як гетьман?',

N'Володимир Винниченко',

N'Олександр Шульгин',

N'Павло Скоропадський',

N'Симон Петлюра',

'C'),

(N'Коли відбулося проголошення Акта Злуки УНР та ЗУНР?',

N'22 січня 1919',

N'22 лютого 1919',

N'1 січня 1919',

N'1 лютого 1919',

'A'),

(N'Хто був автором Конституції Пилипа Орлика?',

N'Пилип Орлик',

N'Іван Мазепа',

N'Іван Мазепинець',

N'Григорій Сковорода',

'A'),

(N'У якому році відбулася Чорнобильська катастрофа?',

N'1984',

N'1986',

N'1988',

N'1990',

'B');

GO

**Program.cs:**

using Task\_1.Services;

using Task\_1.Models;

var builder = WebApplication.CreateBuilder(args);

// File-based stores для користувачів та повідомлень

builder.Services.AddSingleton<FileStore<User>>(sp =>

new FileStore<User>(sp.GetRequiredService<IWebHostEnvironment>(), "users.json"));

builder.Services.AddSingleton<FileStore<Message>>(sp =>

new FileStore<Message>(sp.GetRequiredService<IWebHostEnvironment>(), "messages.json"));

builder.Services.AddControllersWithViews();

builder.Services.AddSession();

builder.Services.AddHttpContextAccessor();

var app = builder.Build();

app.UseStaticFiles();

app.UseSession();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Splash}/{action=Index}/{id?}"

);

app.Run();

**appsettings.json:**

{

"ConnectionStrings": {

"TestDb": "Server=(localdb)\\MSSQLLocalDB;Database=TestetDB;Trusted\_Connection=True;MultipleActiveResultSets=true"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

**Views/Account/EditEmail.cshtml:**

@model Task\_1.Models.EditEmailViewModel

@{

ViewData["Title"] = "Edit Email";

}

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

<h2>Редагувати Email</h2>

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

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

<span asp-validation-for="Email" class="error-message"></span>

<label asp-for="Email">New Email</label>

<input asp-for="Email" class="form-control" />

</div>

<button type="submit" class="btn-register">Save</button>

</form>

</div>

@section Scripts {

@await Html.PartialAsync("\_ValidationScriptsPartial")

<script>

$(function(){

var f=$('form');

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

$.validator.setDefaults({

showErrors(eMap,eList){

this.defaultShowErrors();

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

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

if(txt)$(this).html(txt.split(' ').join('<br/>'));

});

}

});

$.validator.unobtrusive.parse(f);

});

</script>

}

**Views/Account/EditName.cshtml:**

@model Task\_1.Models.EditNameViewModel

@{

ViewData["Title"] = "Edit Name";

}

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

<h2>Редагувати ім’я</h2>

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

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

<span asp-validation-for="Name" class="error-message"></span>

<label asp-for="Name">New Name</label>

<input asp-for="Name" class="form-control" />

</div>

<button type="submit" class="btn-register">Save</button>

</form>

</div>

@section Scripts {

@await Html.PartialAsync("\_ValidationScriptsPartial")

<script>

$(function(){

var f=$('form');

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

$.validator.setDefaults({

showErrors(eMap,eList){

this.defaultShowErrors();

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

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

if(txt)$(this).html(txt.split(' ').join('<br/>'));

});

}

});

$.validator.unobtrusive.parse(f);

});

</script>

}

**Views/Account/EditPassword.cshtml:**

@model Task\_1.Models.EditPasswordViewModel

@{

ViewData["Title"] = "Edit Password";

}

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

<h2>Редагувати пароль</h2>

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

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

<span asp-validation-for="NewPassword" class="error-message"></span>

<label asp-for="NewPassword">New Password</label>

<input asp-for="NewPassword" type="password" class="form-control" />

</div>

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

<span asp-validation-for="ConfirmPassword" class="error-message"></span>

<label asp-for="ConfirmPassword">Confirm Password</label>

<input asp-for="ConfirmPassword" type="password" class="form-control" />

</div>

<button type="submit" class="btn-register">Save</button>

</form>

</div>

@section Scripts {

@await Html.PartialAsync("\_ValidationScriptsPartial")

<script>

$(function(){

var f=$('form');

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

$.validator.setDefaults({

showErrors(eMap,eList){

this.defaultShowErrors();

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

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

if(txt)$(this).html(txt.split(' ').join('<br/>'));

});

}

});

$.validator.unobtrusive.parse(f);

});

</script>

}

**Views/Account/EditPhone.cshtml:**

@model Task\_1.Models.EditPhoneViewModel

@{

ViewData["Title"] = "Edit Phone";

}

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

<h2>Редагувати телефон</h2>

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

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

<span asp-validation-for="Phone" class="error-message"></span>

<label asp-for="Phone">New Phone</label>

<input asp-for="Phone" class="form-control" />

</div>

<button type="submit" class="btn-register">Save</button>

</form>

</div>

@section Scripts {

@await Html.PartialAsync("\_ValidationScriptsPartial")

<script>

$(function(){

var f=$('form');

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

$.validator.setDefaults({

showErrors(eMap,eList){

this.defaultShowErrors();

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

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

if(txt)$(this).html(txt.split(' ').join('<br/>'));

});

}

});

$.validator.unobtrusive.parse(f);

});

</script>

}

**Views/Account/Login.cshtml:**

@model Task\_1.Models.LoginViewModel

@{

ViewData["Title"] = "Login";

}

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

<h2>Login</h2>

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

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

<span asp-validation-for="Name" class="error-message"></span>

<label asp-for="Name">Name</label>

<input asp-for="Name" class="form-control" />

</div>

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

<span asp-validation-for="Password" class="error-message"></span>

<label asp-for="Password">Password</label>

<input asp-for="Password" type="password" class="form-control" />

</div>

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

<button type="submit" class="btn-register">LOGIN</button>

</form>

</div>

@section Scripts {

@await Html.PartialAsync("\_ValidationScriptsPartial")

<script>

$(function(){

var f=$('form');

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

$.validator.setDefaults({

showErrors(eMap,eList){

this.defaultShowErrors();

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

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

if(txt)$(this).html(txt.split(' ').join('<br/>'));

});

}

});

$.validator.unobtrusive.parse(f);

});

</script>

}

**Views/Account/Profile.cshtml:**

@model Task\_1.Models.User

@{

ViewData["Title"] = "Profile";

}

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

<h2>Мій профіль</h2>

<ul class="list-unstyled">

<li><strong>Name:</strong> @Model.Name <a asp-action="EditName" class="ms-2">Edit</a></li>

<li><strong>Email:</strong> @Model.Email <a asp-action="EditEmail" class="ms-2">Edit</a></li>

<li><strong>Phone:</strong> @Model.Phone <a asp-action="EditPhone" class="ms-2">Edit</a></li>

<li><strong>Password:</strong> \*\*\*\*\*\*\*\* <a asp-action="EditPassword" class="ms-2">Edit</a></li>

</ul>

</div>

**Views/Account/Register.cshtml:**

@model Task\_1.Models.RegisterViewModel

@{

ViewData["Title"] = "Register";

}

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

<h2>Register</h2>

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

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

<span asp-validation-for="Name" class="error-message"></span>

<label asp-for="Name">Name</label>

<input asp-for="Name" class="form-control" />

</div>

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

<span asp-validation-for="Email" class="error-message"></span>

<label asp-for="Email">Email</label>

<input asp-for="Email" class="form-control" />

</div>

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

<span asp-validation-for="Phone" class="error-message"></span>

<label asp-for="Phone">Phone</label>

<input asp-for="Phone" class="form-control" />

</div>

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

<span asp-validation-for="Password" class="error-message"></span>

<label asp-for="Password">Password</label>

<input asp-for="Password" type="password" class="form-control" />

</div>

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

<span asp-validation-for="ConfirmPassword" class="error-message"></span>

<label asp-for="ConfirmPassword">Confirm Password</label>

<input asp-for="ConfirmPassword" type="password" class="form-control" />

</div>

<button type="submit" class="btn-register">REGISTER</button>

</form>

</div>

@section Scripts {

@await Html.PartialAsync("\_ValidationScriptsPartial")

<script>

$(function(){

var f=$('form');

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

$.validator.setDefaults({

showErrors(eMap,eList){

this.defaultShowErrors();

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

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

if(txt)$(this).html(txt.split(' ').join('<br/>'));

});

}

});

$.validator.unobtrusive.parse(f);

});

</script>

}

**Views/Messages/Details.cshtml:**

@model Task\_1.Models.Message

@{

ViewData["Title"] = "Message Details";

}

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

<h2>Повідомлення</h2>

<dl class="row">

<dt class="col-sm-3">From</dt>

<dd class="col-sm-9">@Model.From</dd>

<dt class="col-sm-3">To</dt>

<dd class="col-sm-9">@Model.To</dd>

<dt class="col-sm-3">Theme</dt>

<dd class="col-sm-9">@Model.Theme</dd>

<dt class="col-sm-3">Sent At</dt>

<dd class="col-sm-9">@Model.SentAt.ToString("f")</dd>

<dt class="col-sm-3">Text</dt>

<dd class="col-sm-9">@Model.Text</dd>

</dl>

<a asp-action="Inbox" class="btn btn-secondary">Back to Inbox</a>

</div>

**Views/Messages/Inbox.cshtml:**

@model List<Task\_1.Models.Message>

@{

ViewData["Title"] = "Inbox";

}

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

<h2>Вхідні повідомлення</h2>

@if (!Model.Any())

{

<p>У вас немає нових повідомлень.</p>

}

else

{

<ul class="list-group">

@foreach (var m in Model)

{

<li class="list-group-item">

<a asp-action="Details" asp-route-id="@m.Id">

<strong>@m.SentAt.ToString("g")</strong> від <strong>@m.From</strong> — @m.Theme

</a>

</li>

}

</ul>

}

</div>

**Views/Messages/Send.cshtml:**

@model Task\_1.Models.SendMessageViewModel

@{

ViewData["Title"] = "Send Message";

var users = ViewBag.Users as List<string> ?? new List<string>();

}

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

<h2>Відправити повідомлення</h2>

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

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

<span asp-validation-for="To" class="error-message"></span>

<label asp-for="To">To</label>

<select asp-for="To" class="form-control">

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

@foreach (var u in users)

{

<option value="@u">@u</option>

}

</select>

</div>

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

<span asp-validation-for="Theme" class="error-message"></span>

<label asp-for="Theme">Theme</label>

<input asp-for="Theme" class="form-control" />

</div>

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

<span asp-validation-for="Text" class="error-message"></span>

<label asp-for="Text">Text</label>

<textarea asp-for="Text" class="form-control" rows="5"></textarea>

</div>

<button type="submit" class="btn-register">Send</button>

</form>

</div>

@section Scripts {

@await Html.PartialAsync("\_ValidationScriptsPartial")

<script>

$(function(){

var f=$('form');

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

$.validator.setDefaults({

showErrors(eMap,eList){

this.defaultShowErrors();

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

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

if(txt)$(this).html(txt.split(' ').join('<br/>'));

});

}

});

$.validator.unobtrusive.parse(f);

});

</script>

}

**Views/Messages/Sent.cshtml:**

@model List<Task\_1.Models.Message>

@{

ViewData["Title"] = "Sent Messages";

}

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

<h2>Надіслані повідомлення</h2>

@if (!Model.Any())

{

<p>Ви ще не надсилали жодного повідомлення.</p>

}

else

{

<ul class="list-group">

@foreach (var m in Model)

{

<li class="list-group-item">

<strong>@m.SentAt.ToString("g")</strong> до <strong>@m.To</strong> — @m.Theme

</li>

}

</ul>

}

</div>

**Views/Shared/\_Layout.cshtml:**

@{

var ctrl = ViewContext.RouteData.Values["controller"]?.ToString();

var act = ViewContext.RouteData.Values["action"]?.ToString();

var user = Context.Session.GetString("UserName");

}

<!DOCTYPE html>

<html lang="uk">

<head>

<meta charset="utf-8" />

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

<title>@ViewData["Title"] – NovaChat</title>

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

<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" asp-controller="Splash" asp-action="Index">NovaChat</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#nav">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="nav">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

@\* При заході на Splash нема жодних лінків \*@

@if (ctrl == "Splash")

{

}

@\* На сторінці логіну — показати Register \*@

else if (ctrl == "Account" && act == "Login")

{

<li class="nav-item">

<a class="nav-link" asp-action="Register">Register</a>

</li>

}

@\* На сторінці реєстрації — показати Login \*@

else if (ctrl == "Account" && act == "Register")

{

<li class="nav-item">

<a class="nav-link" asp-action="Login">Login</a>

</li>

}

@\* Коли в профілі — показати Logout + Messages + Тест \*@

else if (ctrl == "Account" && act == "Profile")

{

<li class="nav-item">

<a class="nav-link" asp-action="Logout">Logout</a>

</li>

<li class="nav-item">

<a class="nav-link" asp-controller="Messages" asp-action="Send">Send Msg</a>

</li>

<li class="nav-item">

<a class="nav-link" asp-controller="Messages" asp-action="Inbox">Inbox</a>

</li>

<li class="nav-item">

<a class="nav-link" asp-controller="Messages" asp-action="Sent">Sent</a>

</li>

<li class="nav-item">

<a class="nav-link" asp-controller="Test" asp-action="Index">Тест</a>

</li>

}

@\* У Messages — лише «Back to Profile» \*@

else if (ctrl == "Messages" && (act == "Send" || act == "Inbox" || act == "Sent" || act == "Details"))

{

<li class="nav-item">

<a class="nav-link" asp-controller="Account" asp-action="Profile">Back to Profile</a>

</li>

}

@\* В режимах редагування профілю теж «Back to Profile» \*@

else if (ctrl == "Account" && act.StartsWith("Edit"))

{

<li class="nav-item">

<a class="nav-link" asp-action="Profile">Back to Profile</a>

</li>

}

@\* Наш новий пункт: коли в TestController показуємо «Профіль» \*@

else if (ctrl == "Test")

{

<li class="nav-item">

<a class="nav-link" asp-controller="Account" asp-action="Profile">Профіль</a>

</li>

}

</ul>

<span class="navbar-text text-muted">@user</span>

</div>

</div>

</nav>

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

@RenderBody()

</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/dist/jquery.validate.unobtrusive.min.js"></script>

@RenderSection("Scripts", required: false)

</body>

</html>

**Views/Splash/Index.cshtml:**

@{

Layout = null;

}

<!DOCTYPE html>

<html lang="uk">

<head>

<meta charset="utf-8" />

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

<title>NovaChat</title>

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

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

<style>

body {

background: url('https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS7\_zlBPsa\_kgNmEOgmIlH3ExsVIVdLrV99DQ&s')

no-repeat center center fixed;

background-size: cover;

}

.splash {

position:absolute; top:50%; left:50%;

transform:translate(-50%,-50%); text-align:center;

}

.logo { width:150px; margin-bottom:20px; }

.title {

color:#fff; font-size:3rem; font-weight:700;

text-shadow:2px 2px 4px rgba(0,0,0,0.5);

}

</style>

</head>

<body>

<div class="splash animate\_\_animated animate\_\_fadeIn">

<img src="https://upload.wikimedia.org/wikipedia/commons/thumb/b/be/Facebook\_Messenger\_logo\_2020.svg/2048px-Facebook\_Messenger\_logo\_2020.svg.png"

alt="Logo" class="logo animate\_\_zoomIn" />

<div class="title animate\_\_fadeInUp">NovaChat</div>

</div>

<script>

setTimeout(()=>location.href='@Url.Action("Login", "Account")',3000);

</script>

</body>

</html>

**Views/Test/Index.cshtml:**

@{

ViewData["Title"] = "Тест: Історія України";

}

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

<h2>@ViewData["Title"]</h2>

<p>Натисніть «Почати», щоб розпочати тестування.</p>

<div class="d-flex justify-content-between">

<a asp-controller="Account" asp-action="Profile" class="btn btn-outline-secondary">

← Назад до профілю

</a>

<a asp-action="Take" class="btn btn-primary">

Почати

</a>

</div>

</div>

**Views/Test/Results.cshtml:**

@{

ViewData["Title"] = "Результат тесту";

var sc = ViewBag.Score as int? ?? 0;

}

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

<h2>Ваш результат: @sc</h2>

<a asp-action="Index" class="btn btn-primary">Повернутися</a>

</div>

**Views/Test/Take.cshtml:**

@model List<Task\_1.Models.TestAnswerViewModel>

@{

ViewData["Title"] = "Проходження тесту";

}

<form asp-action="Take" method="post">

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

<h2>Питання</h2>

@\* Валідаційна підказка, якщо раптом знадобиться \*@

<span asp-validation-summary="ModelOnly" class="text-danger mb-3"></span>

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

{

<div class="mb-4">

@\* приховані поля для Id та CorrectOption \*@

<input asp-for="@Model[i].Id" type="hidden" />

<input asp-for="@Model[i].CorrectOption" type="hidden" />

<p><strong>@(i + 1). @Model[i].Text</strong></p>

<div>

<label>

<input asp-for="@Model[i].SelectedOption" value="A" type="radio" />

@Model[i].OptionA

</label><br />

<label>

<input asp-for="@Model[i].SelectedOption" value="B" type="radio" />

@Model[i].OptionB

</label><br />

<label>

<input asp-for="@Model[i].SelectedOption" value="C" type="radio" />

@Model[i].OptionC

</label><br />

<label>

<input asp-for="@Model[i].SelectedOption" value="D" type="radio" />

@Model[i].OptionD

</label>

</div>

</div>

}

<button type="submit" class="btn btn-success">Завершити</button>

</div>

</form>

@section Scripts {

@await Html.PartialAsync("\_ValidationScriptsPartial")

}

**Services/FileStore.cs:**

using System.Text.Json;

namespace Task\_1.Services

{

public class FileStore<T>

{

private readonly string \_path;

private readonly JsonSerializerOptions \_opts = new() { WriteIndented = true };

public FileStore(IWebHostEnvironment env, string fileName)

{

var dir = Path.Combine(env.ContentRootPath, "App\_Data");

if (!Directory.Exists(dir)) Directory.CreateDirectory(dir);

\_path = Path.Combine(dir, fileName);

if (!File.Exists(\_path)) File.WriteAllText(\_path, "[]");

}

public List<T> Load()

=> JsonSerializer.Deserialize<List<T>>(File.ReadAllText(\_path)) ?? new List<T>();

public void Save(List<T> items)

=> File.WriteAllText(\_path, JsonSerializer.Serialize(items, \_opts));

}

}

**Models/EditEmailViewModel.cs:**

using System.ComponentModel.DataAnnotations;

namespace Task\_1.Models

{

public class EditEmailViewModel

{

[Required, EmailAddress]

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

}

}

**Models/EditNameViewModel.cs:**

using System.ComponentModel.DataAnnotations;

namespace Task\_1.Models

{

public class EditNameViewModel

{

[Required, StringLength(15), RegularExpression("^[A-Za-z0-9]+$")]

public string Name { get; set; } = null!;

}

}

**Models/EditPasswordViewModel.cs:**

using System.ComponentModel.DataAnnotations;

namespace Task\_1.Models

{

public class EditPasswordViewModel

{

[Required, MinLength(12), DataType(DataType.Password)]

[RegularExpression(@"^(?=.\*[!@#$%^&\*()]).+$")]

public string NewPassword { get; set; } = null!;

[Required, Compare("NewPassword"), DataType(DataType.Password)]

public string ConfirmPassword { get; set; } = null!;

}

}

**Models/EditPhoneViewModel.cs:**

using System.ComponentModel.DataAnnotations;

namespace Task\_1.Models

{

public class EditPhoneViewModel

{

[Required, Phone]

public string Phone { get; set; } = null!;

}

}

**Models/ErrorViewModel.cs:**

namespace Task\_1.Models

{

public class ErrorViewModel

{

public string? RequestId { get; set; }

public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);

}

}

**Models/LoginViewModel.cs:**

using System.ComponentModel.DataAnnotations;

namespace Task\_1.Models

{

public class LoginViewModel

{

[Required]

public string Name { get; set; } = null!;

[Required, DataType(DataType.Password)]

public string Password { get; set; } = null!;

}

}

**Models/Message.cs:**

using System;

namespace Task\_1.Models

{

public class Message

{

public Guid Id { get; set; }

public string From { get; set; } = null!;

public string To { get; set; } = null!;

public string Theme { get; set; } = null!;

public string Text { get; set; } = null!;

public DateTime SentAt { get; set; }

}

}

**Models/RegisterViewModel.cs:**

using System.ComponentModel.DataAnnotations;

namespace Task\_1.Models

{

public class RegisterViewModel

{

[Required, StringLength(15), RegularExpression("^[A-Za-z0-9]+$")]

public string Name { get; set; } = null!;

[Required, EmailAddress]

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

[Required, Phone]

public string Phone { get; set; } = null!;

[Required, MinLength(12), DataType(DataType.Password)]

[RegularExpression(@"^(?=.\*[!@#$%^&\*()]).+$")]

public string Password { get; set; } = null!;

[Required, Compare("Password"), DataType(DataType.Password)]

public string ConfirmPassword { get; set; } = null!;

}

}

**Models/SendMessageViewModel.cs:**

using System.ComponentModel.DataAnnotations;

namespace Task\_1.Models

{

public class SendMessageViewModel

{

[Required]

public string To { get; set; } = null!;

[Required]

public string Theme { get; set; } = null!;

[Required]

public string Text { get; set; } = null!;

}

}

**Models/TestAnswerViewModel.cs:**

using System.ComponentModel.DataAnnotations;

namespace Task\_1.Models

{

public class TestAnswerViewModel

{

public int Id { get; set; }

public string Text { get; set; } = null!;

public string OptionA { get; set; } = null!;

public string OptionB { get; set; } = null!;

public string OptionC { get; set; } = null!;

public string OptionD { get; set; } = null!;

// Приберемо [Required], щоб не було model‐state помилки

public char? SelectedOption { get; set; }

// цільова правильна відповідь з БД

public char CorrectOption { get; set; }

}

}

**Models/TestQuestion.cs:**

namespace Task\_1.Models

{

public class TestQuestion

{

public int Id { get; set; }

public string Text { get; set; } = null!;

public string OptionA { get; set; } = null!;

public string OptionB { get; set; } = null!;

public string OptionC { get; set; } = null!;

public string OptionD { get; set; } = null!;

public char CorrectOption { get; set; }

}

}

**Models/TestResult.cs:**

namespace Task\_1.Models

{

public class TestResult

{

public int Id { get; set; }

public string UserName { get; set; } = null!;

public int Score { get; set; }

public DateTime TakenAt { get; set; }

}

}

**Models/User.cs:**

namespace Task\_1.Models

{

public class User

{

public string Name { get; set; } = null!;

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

public string Phone { get; set; } = null!;

public string Password { get; set; } = null!;

}

}

**Controllers/AccountController.cs:**

using Microsoft.AspNetCore.Mvc;

using Task\_1.Models;

using Task\_1.Services;

namespace Task\_1.Controllers

{

public class AccountController : Controller

{

private readonly FileStore<User> \_users;

public AccountController(FileStore<User> users) => \_users = users;

[HttpGet]

public IActionResult Register() => View();

[HttpPost]

public IActionResult Register(RegisterViewModel vm)

{

var all = \_users.Load();

if (all.Any(u => u.Name == vm.Name))

ModelState.AddModelError(nameof(vm.Name), "Name taken");

if (!ModelState.IsValid)

return View(vm);

all.Add(new User

{

Name = vm.Name,

Email = vm.Email,

Phone = vm.Phone,

Password = vm.Password

});

\_users.Save(all);

return RedirectToAction("Login");

}

[HttpGet]

public IActionResult Login() => View();

[HttpPost]

public IActionResult Login(LoginViewModel vm)

{

if (!ModelState.IsValid)

return View(vm);

var user = \_users.Load()

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

if (user == null)

{

ModelState.AddModelError(string.Empty, "Invalid credentials");

return View(vm);

}

HttpContext.Session.SetString("UserName", user.Name);

return RedirectToAction("Profile");

}

public IActionResult Logout()

{

HttpContext.Session.Clear();

return RedirectToAction("Login");

}

[HttpGet]

public IActionResult Profile()

{

var name = HttpContext.Session.GetString("UserName");

if (name == null)

return RedirectToAction("Login");

var user = \_users.Load().First(u => u.Name == name);

return View(user);

}

// EditName, EditEmail, EditPhone, EditPassword — аналогічно Task\_6, з єдиною зміною namespace

[HttpGet]

public IActionResult EditName()

{

var name = HttpContext.Session.GetString("UserName");

if (name == null) return RedirectToAction("Login");

var user = \_users.Load().First(u => u.Name == name);

return View(new EditNameViewModel { Name = user.Name });

}

[HttpPost]

public IActionResult EditName(EditNameViewModel vm)

{

var old = HttpContext.Session.GetString("UserName");

if (old == null) return RedirectToAction("Login");

var all = \_users.Load();

if (all.Any(u => u.Name == vm.Name && u.Name != old))

ModelState.AddModelError(nameof(vm.Name), "Name taken");

if (!ModelState.IsValid)

return View(vm);

var user = all.First(u => u.Name == old);

user.Name = vm.Name;

\_users.Save(all);

HttpContext.Session.SetString("UserName", vm.Name);

return RedirectToAction("Profile");

}

[HttpGet]

public IActionResult EditEmail()

{

var name = HttpContext.Session.GetString("UserName");

if (name == null) return RedirectToAction("Login");

var user = \_users.Load().First(u => u.Name == name);

return View(new EditEmailViewModel { Email = user.Email });

}

[HttpPost]

public IActionResult EditEmail(EditEmailViewModel vm)

{

var name = HttpContext.Session.GetString("UserName");

if (name == null) return RedirectToAction("Login");

if (!ModelState.IsValid) return View(vm);

var all = \_users.Load();

var user = all.First(u => u.Name == name);

user.Email = vm.Email;

\_users.Save(all);

return RedirectToAction("Profile");

}

[HttpGet]

public IActionResult EditPhone()

{

var name = HttpContext.Session.GetString("UserName");

if (name == null) return RedirectToAction("Login");

var user = \_users.Load().First(u => u.Name == name);

return View(new EditPhoneViewModel { Phone = user.Phone });

}

[HttpPost]

public IActionResult EditPhone(EditPhoneViewModel vm)

{

var name = HttpContext.Session.GetString("UserName");

if (name == null) return RedirectToAction("Login");

if (!ModelState.IsValid) return View(vm);

var all = \_users.Load();

var user = all.First(u => u.Name == name);

user.Phone = vm.Phone;

\_users.Save(all);

return RedirectToAction("Profile");

}

[HttpGet]

public IActionResult EditPassword()

{

if (HttpContext.Session.GetString("UserName") == null)

return RedirectToAction("Login");

return View();

}

[HttpPost]

public IActionResult EditPassword(EditPasswordViewModel vm)

{

var name = HttpContext.Session.GetString("UserName");

if (name == null) return RedirectToAction("Login");

if (!ModelState.IsValid) return View(vm);

var all = \_users.Load();

var user = all.First(u => u.Name == name);

user.Password = vm.NewPassword;

\_users.Save(all);

return RedirectToAction("Profile");

}

}

}

**Controllers/HomeController.cs:**

using System.Diagnostics;

using Microsoft.AspNetCore.Mvc;

using Task\_1.Models;

namespace Task\_1.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

return View();

}

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

**Controllers/MessagesController.cs:**

using Microsoft.AspNetCore.Mvc;

using Task\_1.Models;

using Task\_1.Services;

namespace Task\_1.Controllers

{

public class MessagesController : Controller

{

private readonly FileStore<Message> \_msgs;

private readonly FileStore<User> \_users;

public MessagesController(FileStore<Message> msgs, FileStore<User> users)

{

\_msgs = msgs;

\_users = users;

}

[HttpGet]

public IActionResult Send()

{

var from = HttpContext.Session.GetString("UserName");

if (from == null) return RedirectToAction("Login", "Account");

ViewBag.Users = \_users.Load().Select(u => u.Name).ToList();

return View(new SendMessageViewModel());

}

[HttpPost]

public IActionResult Send(SendMessageViewModel vm)

{

var from = HttpContext.Session.GetString("UserName");

if (from == null) return RedirectToAction("Login", "Account");

var list = \_msgs.Load();

list.Add(new Message

{

Id = Guid.NewGuid(),

From = from,

To = vm.To,

Theme = vm.Theme,

Text = vm.Text,

SentAt = DateTime.Now

});

\_msgs.Save(list);

return RedirectToAction("Sent");

}

[HttpGet]

public IActionResult Inbox()

{

var me = HttpContext.Session.GetString("UserName");

if (me == null) return RedirectToAction("Login", "Account");

var inbox = \_msgs.Load()

.Where(m => m.To == me)

.OrderByDescending(m => m.SentAt)

.ToList();

return View(inbox);

}

[HttpGet]

public IActionResult Sent()

{

var me = HttpContext.Session.GetString("UserName");

if (me == null) return RedirectToAction("Login", "Account");

var sent = \_msgs.Load()

.Where(m => m.From == me)

.OrderByDescending(m => m.SentAt)

.ToList();

return View(sent);

}

[HttpGet]

public IActionResult Details(Guid id)

{

var me = HttpContext.Session.GetString("UserName");

if (me == null) return RedirectToAction("Login", "Account");

var msg = \_msgs.Load().FirstOrDefault(m => m.Id == id);

if (msg == null || (msg.From != me && msg.To != me))

return NotFound();

return View(msg);

}

}

}

**Controllers/SplashController.cs:**

using Microsoft.AspNetCore.Mvc;

namespace Task\_1.Controllers

{

public class SplashController : Controller

{

[HttpGet]

public IActionResult Index() => View();

}

}

**Controllers/TestController.cs:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.Data.SqlClient;

using Task\_1.Models;

namespace Task\_1.Controllers

{

public class TestController : Controller

{

private readonly string \_conn;

public TestController(IConfiguration cfg)

{

\_conn = cfg.GetConnectionString("TestDb")!;

}

[HttpGet]

public IActionResult Index()

=> View();

[HttpGet]

public IActionResult Take()

{

var list = new List<TestAnswerViewModel>();

using var cn = new SqlConnection(\_conn);

cn.Open();

using var cmd = new SqlCommand("SELECT \* FROM Questions", cn);

using var r = cmd.ExecuteReader();

while (r.Read())

{

list.Add(new TestAnswerViewModel

{

Id = (int)r["Id"],

Text = (string)r["Text"],

OptionA = (string)r["OptionA"],

OptionB = (string)r["OptionB"],

OptionC = (string)r["OptionC"],

OptionD = (string)r["OptionD"],

CorrectOption = ((string)r["CorrectOption"])[0]

});

}

return View(list);

}

[HttpPost]

public IActionResult Take(List<TestAnswerViewModel> answers)

{

// Порахувати бали, порівнюючи заповнене з прихованим CorrectOption

int score = 0;

foreach (var ans in answers)

{

if (ans.SelectedOption.HasValue

&& ans.SelectedOption.Value == ans.CorrectOption)

{

score++;

}

}

var user = HttpContext.Session.GetString("UserName")!;

using var cn = new SqlConnection(\_conn);

cn.Open();

using var cmd = new SqlCommand(

"INSERT INTO TestResults(UserName,Score,TakenAt) VALUES(@u,@s,@t)", cn);

cmd.Parameters.AddWithValue("@u", user);

cmd.Parameters.AddWithValue("@s", score);

cmd.Parameters.AddWithValue("@t", DateTime.Now);

cmd.ExecuteNonQuery();

return RedirectToAction("Results", new { score });

}

[HttpGet]

public IActionResult Results(int score)

{

ViewBag.Score = score;

return View();

}

}

}

**App\_Data/messages.json:**

[ ]

**App\_Data/users.json:**

[ ]

**site.css:**

body {

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

background-size: cover;

}

.form-card {

background: #fff;

padding: 2rem;

border-radius: 8px;

max-width: 480px;

width: 100%;

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

}

.form-card h2 {

margin-bottom: 1.5rem;

font-weight: 500;

}

.form-material .form-control {

background: transparent;

border: none;

border-bottom: 1px solid #ccc;

border-radius: 0;

box-shadow: none;

}

.form-material .form-control:focus {

outline: none;

border-bottom-color: #1abc9c;

}

.form-material label {

display: block;

color: #888;

font-size: 0.9rem;

margin: 0.25rem 0;

}

.error-message {

display: block;

margin-bottom: 0.25rem;

color: #e74c3c;

font-size: 0.8rem;

animation: fade-in 0.4s;

}

.btn-register {

width: 30%;

background-color: #1abc9c;

border: none;

padding: 0.75rem;

text-transform: uppercase;

letter-spacing: 1px;

}

.btn-register:hover {

background-color: #16a085;

}

.list-group-item {

margin-bottom: 0.5rem;

border-radius: 0.5rem;

}

@keyframes fade-in {

from {

opacity: 0;

}

to {

opacity: 1;

}

}

## **Результати роботи програми:**

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

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.1. Таблиця питань в БД

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

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.2. Таблиця результатів в БД

Изображение выглядит как облако, текст, небо, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.3. Загальний вигляд сайту

Изображение выглядит как облако, небо, текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.4. Вигляд форми старту тестування

Изображение выглядит как облако, текст, небо, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.5. Вигляд форми питань

Изображение выглядит как небо, текст, облако, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.6. Вигляд форми питань

Изображение выглядит как облако, небо, текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1.7. Вигляд форми результатів

## **Алгоритм побудови кодів**

1. **Підготовка бази даних**
   * Виконати SQL-скрипт створення таблиць Questions та TestResults у базі TestetDB.
   * Заповнити таблицю Questions питаннями з варіанту №12 (Історія України).
2. **Налаштування проєкту**
   * У appsettings.json додати рядок з’єднання "TestDb" : "Server=(localdb)\\MSSQLLocalDB;Database=TestetDB;...".
   * У Program.cs зареєструвати сервіси:

builder.Services.AddControllersWithViews();

builder.Services.AddSession();

builder.Services.AddHttpContextAccessor();

1. **Створення моделей**
   * **TestQuestion** та **TestAnswerViewModel** для передачі даних запитань і відповідей.
   * **TestResult** для збереження результату (ім’я, бал, час).
2. **Реалізація TestController**
   * **Index()** – відображення сторінки із кнопкою “Почати”.
   * **Take() [GET]** – зчитати всі питання з БД через SqlConnection/SqlCommand і передати у View(list).
   * **Take(List<TestAnswerViewModel>) [POST]** – порівняти SelectedOption та CorrectOption, порахувати score.
   * Записати результат у таблицю TestResults за допомогою SQL-команди.
   * **Results(int score)** – показати користувачу підсумковий бал.
3. **Створення подань (Views/Test/…)**
   * **Index.cshtml** – інформація про тест, кнопка “Почати”.
   * **Take.cshtml** – відображення списку питань з радіокнопками для вибору варіанту.
   * **Results.cshtml** – відображення набраних балів.
4. **Додавання навігації**
   * У \_Layout.cshtml додати пункт меню «Тест» для зареєстрованих користувачів (TestController → Index).
5. **Тестування та перевірка**
   * Зареєструвати й увійти під новим користувачем.
   * Перейти в розділ «Тест», пройти тестування, переконатися в коректному відображенні питань.
   * Завершити тест, перевірити відображення балу та рядок запису в таблиці TestResults.

## **Функціональність**

* **Доступ до тесту** мають тільки авторизовані користувачі зі своїм іменем у сесії.
* **Відображення питань** напряму із таблиці Questions за варіантом №12 (Історія України).
* **Вибір відповіді** здійснюється через радіокнопки – POST-запит повертає список відповідей із прихованим полем CorrectOption.
* **Підрахунок балів**: кожна правильна відповідь додає +1.
* **Збереження результату**: ім’я користувача, отриманий бал та час тестування заносяться в таблицю TestResults.
* **Відображення результату** одразу після завершення тесту (сторінка Results).

## **Висновок**

У Task\_1 реалізовано повністю функціональний механізм тестування зареєстрованих користувачів за напрямком «Історія України» з використанням ASP.NET Core MVC та ADO.NET. Система дозволяє безпечно зберігати дані про питання у SQL-базі, відображати їх у вигляді інтерактивної форми, коректно підраховувати й зберігати результати. Завдяки чіткій архітектурі (моделі, контролери, подання) та шаблону MVC код легко підтримувати та розширювати: можна додавати нові напрямки тестування, статистику або розгортати проєкт на інших СУБД.

# **Завдання 2**

Система тестування має складатися з 10-ти питань відповідної галузі. При чому чотири питання радіо-вигляду (з декількох варіантів відповідей вибирається один варіант), які формуються за допомогою радіо-кнопок, три питання селективного вигляду (з декількох варіан-тів відповідей вибираються декілька варіантів), які формуються за допомогою «чекбоксів» та три питання з безпосередньою відпо-віддю, яка друкується у відповідному текстовому полі чи полях.

**Текст програми:**

**SQL-запит:**

-- Перевірка та створення бази даних (якщо ще не існує)

IF DB\_ID(N'Testet01DB') IS NULL

BEGIN

CREATE DATABASE Testet01DB;

END

GO

USE Testet01DB;

GO

-- Видалити старі таблиці (якщо вони були)

IF OBJECT\_ID(N'dbo.TestResults', N'U') IS NOT NULL

DROP TABLE dbo.TestResults;

IF OBJECT\_ID(N'dbo.Questions', N'U') IS NOT NULL

DROP TABLE dbo.Questions;

GO

-- Створити таблицю запитань із новими полями QuestionType та CorrectAnswers

CREATE TABLE dbo.Questions

(

Id INT IDENTITY(1,1) PRIMARY KEY,

Text NVARCHAR(500) NOT NULL,

OptionA NVARCHAR(200) NOT NULL,

OptionB NVARCHAR(200) NOT NULL,

OptionC NVARCHAR(200) NOT NULL,

OptionD NVARCHAR(200) NOT NULL,

QuestionType CHAR(1) NOT NULL

CONSTRAINT DF\_Questions\_QuestionType DEFAULT ('R'),

CorrectAnswers NVARCHAR(200) NOT NULL

CONSTRAINT DF\_Questions\_CorrectAnswers DEFAULT ('')

);

GO

-- Створити таблицю результатів

CREATE TABLE dbo.TestResults

(

Id INT IDENTITY(1,1) PRIMARY KEY,

UserName NVARCHAR(100) NOT NULL,

Score INT NOT NULL,

TakenAt DATETIME NOT NULL

);

GO

-- Заповнити таблицю Questions 10 тестовими питаннями

-- 4 питання типу R (radio), 3 питання типу C (checkbox), 3 питання типу T (text)

INSERT INTO dbo.Questions

(Text, OptionA, OptionB, OptionC, OptionD, QuestionType, CorrectAnswers)

VALUES

-- 1–4: радіокнопки (лише одна правильна відповідь)

(N'Яка дата Першого Збору Української Центральної Ради?',

N'17 березня 1917', N'7 березня 1917', N'17 квітня 1917', N'7 квітня 1917',

'R', 'A'),

(N'Хто очолив УНР у 1918 році як гетьман?',

N'Володимир Винниченко', N'Олександр Шульгин', N'Павло Скоропадський', N'Симон Петлюра',

'R', 'C'),

(N'Коли відбулося проголошення Акта Злуки УНР та ЗУНР?',

N'22 січня 1919', N'22 лютого 1919', N'1 січня 1919', N'1 лютого 1919',

'R', 'A'),

(N'У якому році сталася Чорнобильська катастрофа?',

N'1984', N'1986', N'1988', N'1990',

'R', 'B'),

-- 5–7: чекбокси (декілька правильних відповідей, зберігаємо безпосередньо символи варіантів)

(N'Які з наведених міст входять до Золотого кільця України?',

N'Київ', N'Львів', N'Чернігів', N'Чернівці',

'C', 'ABC'),

(N'Які символи є державними символами України?',

N'Тризуб', N'Прапор', N'Гімн', N'Калина',

'C', 'ABC'),

(N'Які з перелічених мов є офіційними мовами ООН?',

N'Англійська', N'Французька', N'Українська', N'Іспанська',

'C', 'ABD'),

-- 8–10: текстовий ввід (правильна відповідь — точний текст)

(N'Напишіть столицю України',

N'', N'', N'', N'',

'T', 'Київ'),

(N'У якому році проголошено незалежність України?',

N'', N'', N'', N'',

'T', '1991'),

(N'Прізвище автора слів Гімну України?',

N'', N'', N'', N'',

'T', 'Чубинський')

;

GO

**Models/TestQuestion.cs:**

namespace Task\_2.Models

{

public class TestQuestion

{

public int Id { get; set; }

public string Text { get; set; } = null!;

public string OptionA { get; set; } = null!;

public string OptionB { get; set; } = null!;

public string OptionC { get; set; } = null!;

public string OptionD { get; set; } = null!;

/// <summary>

/// 'R' = radio (1 варіант), 'C' = checkbox (декілька), 'T' = text (вільний ввід)

/// </summary>

public char QuestionType { get; set; }

/// <summary>

/// Для R: "A"; C: "AC"; T: "правильний текст"

/// </summary>

public string CorrectAnswers { get; set; } = null!;

}

}

**Models/TestAnswerViewModel.cs:**

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace Task\_2.Models

{

public class TestAnswerViewModel

{

// приховані поля

public int Id { get; set; }

public char QuestionType { get; set; }

public string CorrectAnswers { get; set; } = null!;

// загальні для всіх

public string Text { get; set; } = null!;

// варіанти для R та C

public string OptionA { get; set; } = null!;

public string OptionB { get; set; } = null!;

public string OptionC { get; set; } = null!;

public string OptionD { get; set; } = null!;

// Ввід користувача:

// для R

public char? SelectedOption { get; set; }

// для C

public List<string> SelectedOptions { get; set; } = new List<string>();

// для T

public string? TextAnswer { get; set; }

}

}

**Controllers/TestController.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Data.SqlClient;

using Microsoft.Extensions.Configuration;

using Task\_2.Models;

namespace Task\_2.Controllers

{

public class TestController : Controller

{

private readonly string \_conn;

public TestController(IConfiguration cfg)

{

\_conn = cfg.GetConnectionString("TestDb")!;

}

[HttpGet]

public IActionResult Index()

{

return View();

}

[HttpGet]

public IActionResult Take()

{

var list = new List<TestAnswerViewModel>();

using var cn = new SqlConnection(\_conn);

cn.Open();

using var cmd = new SqlCommand("SELECT \* FROM Questions", cn);

using var r = cmd.ExecuteReader();

while (r.Read())

{

list.Add(new TestAnswerViewModel

{

Id = (int)r["Id"],

Text = (string)r["Text"],

OptionA = (string)r["OptionA"],

OptionB = (string)r["OptionB"],

OptionC = (string)r["OptionC"],

OptionD = (string)r["OptionD"],

QuestionType = ((string)r["QuestionType"])[0],

CorrectAnswers = (string)r["CorrectAnswers"]

});

}

return View(list);

}

[HttpPost]

public IActionResult Take(List<TestAnswerViewModel> answers)

{

// 1) Порахувати бали

int score = 0;

foreach (var q in answers)

{

switch (q.QuestionType)

{

case 'R': // одиночний вибір

if (q.SelectedOption.HasValue &&

q.CorrectAnswers.Equals(q.SelectedOption.Value.ToString(), StringComparison.OrdinalIgnoreCase))

{

score++;

}

break;

case 'C': // множинний вибір

var correctSet = q.CorrectAnswers

.ToUpper()

.ToCharArray()

.Select(c => c.ToString())

.OrderBy(s => s);

var userSet = q.SelectedOptions

.Select(s => s.ToUpper())

.OrderBy(s => s);

if (correctSet.SequenceEqual(userSet))

{

score++;

}

break;

case 'T': // текстова відповідь

if (!string.IsNullOrWhiteSpace(q.TextAnswer) &&

string.Equals(q.TextAnswer.Trim(), q.CorrectAnswers.Trim(), StringComparison.OrdinalIgnoreCase))

{

score++;

}

break;

}

}

// 2) Отримати ім'я користувача

var userName = HttpContext.Session.GetString("UserName");

if (string.IsNullOrEmpty(userName))

return RedirectToAction("Login", "Account");

// 3) Зберегти результат у БД

using var cn = new SqlConnection(\_conn);

cn.Open();

using var cmd = new SqlCommand(

@"INSERT INTO TestResults (UserName, Score, TakenAt)

VALUES (@user, @score, @time)", cn);

cmd.Parameters.AddWithValue("@user", userName);

cmd.Parameters.AddWithValue("@score", score);

cmd.Parameters.AddWithValue("@time", DateTime.Now);

cmd.ExecuteNonQuery();

// 4) Показати сторінку з результатом

return RedirectToAction("Results", new { score });

}

[HttpGet]

public IActionResult Results(int score)

{

ViewBag.Score = score;

return View();

}

}

}

**Views/Test/Take.cshtml:**

@model List<Task\_2.Models.TestAnswerViewModel>

@{

ViewData["Title"] = "Проходження тесту";

}

<form asp-action="Take" method="post">

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

<h2>Питання</h2>

<span asp-validation-summary="ModelOnly" class="text-danger mb-3"></span>

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

{

var q = Model[i];

<div class="mb-4">

<input asp-for="@q.Id" type="hidden" name="[@i].Id" />

<input asp-for="@q.QuestionType" type="hidden" name="[@i].QuestionType" />

<input asp-for="@q.CorrectAnswers" type="hidden" name="[@i].CorrectAnswers" />

<p><strong>@(i + 1). @q.Text</strong></p>

@if (q.QuestionType == 'R')

{

<div>

<label>

<input asp-for="@q.SelectedOption" name="[@i].SelectedOption" type="radio" value="A" />

@q.OptionA

</label><br />

<label>

<input asp-for="@q.SelectedOption" name="[@i].SelectedOption" type="radio" value="B" />

@q.OptionB

</label><br />

<label>

<input asp-for="@q.SelectedOption" name="[@i].SelectedOption" type="radio" value="C" />

@q.OptionC

</label><br />

<label>

<input asp-for="@q.SelectedOption" name="[@i].SelectedOption" type="radio" value="D" />

@q.OptionD

</label>

</div>

}

else if (q.QuestionType == 'C')

{

<div>

<label>

<input type="checkbox" name="[@i].SelectedOptions" value="A" />

@q.OptionA

</label><br />

<label>

<input type="checkbox" name="[@i].SelectedOptions" value="B" />

@q.OptionB

</label><br />

<label>

<input type="checkbox" name="[@i].SelectedOptions" value="C" />

@q.OptionC

</label><br />

<label>

<input type="checkbox" name="[@i].SelectedOptions" value="D" />

@q.OptionD

</label>

</div>

}

else if (q.QuestionType == 'T')

{

<div class="form-material mb-2">

<input asp-for="@q.TextAnswer" name="[@i].TextAnswer" class="form-control" />

</div>

}

</div>

}

<button type="submit" class="btn btn-success">Завершити</button>

</div>

</form>

@section Scripts {

@await Html.PartialAsync("\_ValidationScriptsPartial")

}

## **Результати роботи програми:**

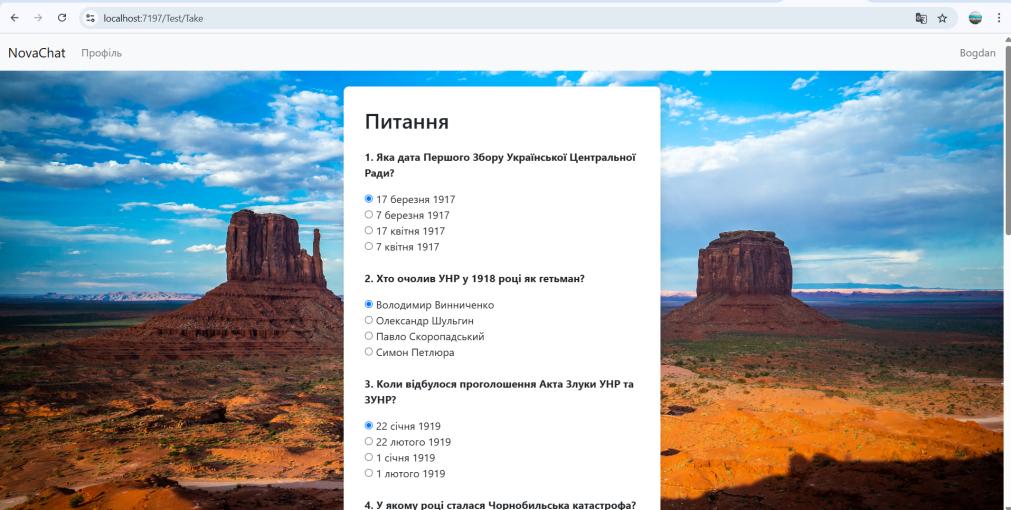


Рисунок 2.1. Форма питань згідно поставленого завдання

Изображение выглядит как небо, облако, текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.2. Форма питань згідно поставленого завдання

Изображение выглядит как небо, облако, текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.3. Форма питань згідно поставленого завдання

## **Алгоритм побудови кодів**

1. **Підготовка БД**
   * Виконати SQL-скрипт: перевірити наявність бази Testet01DB, створити її за потреби.
   * Видалити старі таблиці Questions і TestResults (якщо існують) та створити їх заново з новими полями QuestionType і CorrectAnswers.
   * Наповнити Questions рівно 10 записами:
     + 4 записи з QuestionType='R' (radio), де в полі CorrectAnswers лежить один символ (наприклад, 'A').
     + 3 записи з QuestionType='C' (checkbox), де CorrectAnswers — строка з кількох символів (наприклад, 'ABC').
     + 3 записи з QuestionType='T' (text), де CorrectAnswers — правильний текст відповіді (наприклад, 'Київ').
2. **Моделі**
   * **TestQuestion** – дзеркальна модель рядка з таблиці Questions (поля Id, Text, OptionA–D, QuestionType, CorrectAnswers).
   * **TestAnswerViewModel** – для передачі на View і назад:
     + Приховані поля Id, QuestionType, CorrectAnswers
     + Спільні Text, OptionA–D
     + Для R: char? SelectedOption
     + Для C: List<string> SelectedOptions
     + Для T: string? TextAnswer
3. **Налаштування проєкту**
   * У appsettings.json прописати рядок підключення "TestDb".
   * В Program.cs зареєструвати MVC та сесію:

builder.Services.AddControllersWithViews();

builder.Services.AddSession();

1. **Контролер TestController**
   * **Index()** – просто показ стартової View із кнопкою «Почати».
   * **Take() [GET]** – відкриває з’єднання з БД, читає всі 10 питань у список TestAnswerViewModel і передає у View.
   * **Take(List<TestAnswerViewModel>) [POST]** – для кожного запитання:
     + Якщо QuestionType='R', порівняти SelectedOption з CorrectAnswers → +1 бал.
     + Якщо QuestionType='C', впорядкувати та порівняти множини символів із SelectedOptions → +1 бал.
     + Якщо QuestionType='T', порівняти введений текст із CorrectAnswers (ігноруючи регістр і зайві пробіли) → +1 бал.
   * Перевірити, що користувач залогінений (ім’я в сесії), відкрити з’єднання з БД і вставити новий рядок у TestResults із UserName, Score і TakenAt=DateTime.Now.
   * **Results(int score)** – відобразити кінцевий результат користувачу.
2. **Подання Take.cshtml**
   * Ітеруватися по моделі списку TestAnswerViewModel.
   * Для кожного запитання, залежно від QuestionType, виводити:
     + **R** – чотири input type="radio".
     + **C** – чотири input type="checkbox".
     + **T** – одне текстове поле input type="text".
   * Приховані поля передаються через type="hidden".
   * Кнопка «Завершити» відправляє POST.

## **Функціональність**

* **10 питань**: 4 одиночного вибору (radio), 3 множинного (checkbox), 3 текстових (input).
* **Динамічне відображення**: один шаблон Take.cshtml обробляє всі три типи.
* **Валідація**: якщо жодної відповіді не вибрано/не введено, з’являється Model-state помилка.
* **Підрахунок балів** за чіткою логікою для кожного типу запитань.
* **Збереження результатів** у таблицю TestResults разом з ім’ям користувача та часом проходження.
* **Перехід між екранами**: Index → Take → Results із можливістю повернутися в профіль чи пройти тест знову.

## **Висновок**

У Task\_2 реалізовано розширену систему тестування з трьома різними форматами запитань, що забезпечує більшу гнучкість і точність оцінювання знань. Архітектура на базі ASP.NET Core MVC та ADO.NET гарантує просте розширення (наприклад, додавання інших типів питань) і надійне зберігання даних у SQL Server. Такий підхід дозволяє швидко адаптувати тест під будь-яку тематику та масштабувати функціонал без значного переписування коду.

# **Завдання 3**

Розробити можливість того щоб під час тестування питання подавалися б у випадковому вигляді.

## **Текст програми:**

**Controllers/TestController.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Data.SqlClient;

using Microsoft.Extensions.Configuration;

using Task\_3.Models;

namespace Task\_3.Controllers

{

public class TestController : Controller

{

private readonly string \_conn;

public TestController(IConfiguration cfg)

{

\_conn = cfg.GetConnectionString("TestDb")!;

}

[HttpGet]

public IActionResult Index()

{

return View();

}

[HttpGet]

public IActionResult Take()

{

var list = new List<TestAnswerViewModel>();

using var cn = new SqlConnection(\_conn);

cn.Open();

using (var cmd = new SqlCommand("SELECT \* FROM Questions", cn))

using (var r = cmd.ExecuteReader())

{

while (r.Read())

{

list.Add(new TestAnswerViewModel

{

Id = (int)r["Id"],

Text = (string)r["Text"],

OptionA = (string)r["OptionA"],

OptionB = (string)r["OptionB"],

OptionC = (string)r["OptionC"],

OptionD = (string)r["OptionD"],

QuestionType = ((string)r["QuestionType"])[0],

CorrectAnswers = (string)r["CorrectAnswers"]

});

}

}

// --- Ось тут рандомізуємо порядок питань ---

var rnd = new Random();

list = list

.OrderBy(\_ => rnd.Next()) // або OrderBy(\_ => Guid.NewGuid())

.ToList();

return View(list);

}

[HttpPost]

public IActionResult Take(List<TestAnswerViewModel> answers)

{

int score = 0;

foreach (var q in answers)

{

switch (q.QuestionType)

{

case 'R':

if (q.SelectedOption.HasValue &&

q.CorrectAnswers.Equals(q.SelectedOption.Value.ToString(), StringComparison.OrdinalIgnoreCase))

{

score++;

}

break;

case 'C':

var correctSet = q.CorrectAnswers

.ToUpper().ToCharArray()

.Select(c => c.ToString())

.OrderBy(s => s);

var userSet = q.SelectedOptions

.Select(s => s.ToUpper())

.OrderBy(s => s);

if (correctSet.SequenceEqual(userSet))

score++;

break;

case 'T':

if (!string.IsNullOrWhiteSpace(q.TextAnswer) &&

string.Equals(q.TextAnswer.Trim(), q.CorrectAnswers.Trim(), StringComparison.OrdinalIgnoreCase))

{

score++;

}

break;

}

}

var userName = HttpContext.Session.GetString("UserName");

if (string.IsNullOrEmpty(userName))

return RedirectToAction("Login", "Account");

using var cn = new SqlConnection(\_conn);

cn.Open();

using var cmd = new SqlCommand(

@"INSERT INTO TestResults (UserName, Score, TakenAt)

VALUES (@user, @score, @time)", cn);

cmd.Parameters.AddWithValue("@user", userName);

cmd.Parameters.AddWithValue("@score", score);

cmd.Parameters.AddWithValue("@time", DateTime.Now);

cmd.ExecuteNonQuery();

return RedirectToAction("Results", new { score });

}

[HttpGet]

public IActionResult Results(int score)

{

ViewBag.Score = score;

return View();

}

}

}

## **Результати роботи програми:**

Изображение выглядит как облако, небо, текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.1. Подання питань у випадковому порядку

Изображение выглядит как небо, облако, текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.2. Подання питань у випадковому порядку

Изображение выглядит как небо, облако, текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.3. Подання питань у випадковому порядку

Изображение выглядит как небо, облако, текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.4. Подання питань у випадковому порядку

Изображение выглядит как облако, небо, текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.5. Подання питань у випадковому порядку

Изображение выглядит как небо, облако, текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.6. Подання питань у випадковому порядку

## **Алгоритм побудови кодів**

1. **Завантаження питань з БД**
   * Відкрити з’єднання з TestDb і виконати SELECT \* FROM Questions.
   * Заповнити List<TestAnswerViewModel> об’єктами із полями Id, Text, OptionA–D, QuestionType, CorrectAnswers.
2. **Рандомізація списку**
   * Створити екземпляр Random rnd = new Random();
   * Переструктурувати порядок елементів:

list = list.OrderBy(\_ => rnd.Next()).ToList();

1. **Відображення у View**
   * Передати перемішаний list у View(list) і вивести питання у формі (radio, checkbox, input).
2. **Обробка відповіді (POST)**
   * Отримати назад список answers з вибором користувача.
   * Для кожного питання порахувати бал:
     + 'R' – зіставити SelectedOption з CorrectAnswers.
     + 'C' – порівняти впорядковані множини символів.
     + 'T' – порівняти текстову відповідь, ігноруючи регістр.
3. **Збереження результату**
   * Перевірити, що користувач залогінений (Session "UserName").
   * Виконати INSERT INTO TestResults (UserName,Score,TakenAt) VALUES (...).
   * Перенаправити на Results, передавши score.

## **Функціональність**

* При кожному завантаженні сторінки Take порядок запитань генерується заново, тож користувач не бачить їх у сталому порядку.
* Решта механізму (відображення типів питань, підрахунок балів, запис у таблицю TestResults) працює за вже відпрацьованою логікою.

## **Висновок**

Додавши випадкову послідовність питань, ми ускладнюємо можливість зубріння фіксованої черги і підвищуємо об’єктивність оцінювання. Така рандомізація робить тест більш адаптивним і гарантує, що кожне проходження є унікальним, зменшуючи ризик повторюваних шаблонних відповідей.

# **Завдання 4**

Після отримання відповідей на усі питання система має виставити оцінку від 0 до 100 балів, яка зберігається в профілі користувача і не може бути ним змінена. Критерії оцінювання вибираються за бажанням розробника, наприклад кожна вірна відповідь додає 10 балів до підсумкової оцінки, або використовувати критерій зважених питань тощо.

## **Текст програми:**

**Models/User.cs:**

// Models/User.cs

namespace Task\_4.Models

{

public class User

{

public string Name { get; set; } = null!;

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

public string Phone { get; set; } = null!;

public string Password { get; set; } = null!;

// ← НОВЕ поле для збереження останньої оцінки 0–100

public int Score { get; set; }

}

}

**Controllers/TestController.cs:**

// Controllers/TestController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Data.SqlClient;

using Microsoft.Extensions.Configuration;

using Task\_4.Models;

using Task\_4.Services; // для FileStore<User>

namespace Task\_4.Controllers

{

public class TestController : Controller

{

private readonly string \_conn;

private readonly FileStore<User> \_users;

public TestController(IConfiguration cfg, FileStore<User> users)

{

\_conn = cfg.GetConnectionString("TestDb")!;

\_users = users;

}

[HttpGet]

public IActionResult Index() => View();

[HttpGet]

public IActionResult Take()

{

var list = new List<TestAnswerViewModel>();

using var cn = new SqlConnection(\_conn);

cn.Open();

using var cmd = new SqlCommand("SELECT \* FROM Questions", cn);

using var r = cmd.ExecuteReader();

while (r.Read())

{

list.Add(new TestAnswerViewModel

{

Id = (int)r["Id"],

Text = (string)r["Text"],

OptionA = (string)r["OptionA"],

OptionB = (string)r["OptionB"],

OptionC = (string)r["OptionC"],

OptionD = (string)r["OptionD"],

QuestionType = ((string)r["QuestionType"])[0],

CorrectAnswers = (string)r["CorrectAnswers"]

});

}

// Випадковий порядок питань

var rnd = new Random();

list = list.OrderBy(\_ => rnd.Next()).ToList();

return View(list);

}

[HttpPost]

public IActionResult Take(List<TestAnswerViewModel> answers)

{

// 1) Порахувати кількість правильних відповідей

int correctCount = 0;

foreach (var q in answers)

{

switch (q.QuestionType)

{

case 'R':

if (q.SelectedOption.HasValue &&

q.CorrectAnswers.Equals(

q.SelectedOption.Value.ToString(),

StringComparison.OrdinalIgnoreCase))

correctCount++;

break;

case 'C':

var correctSet = q.CorrectAnswers

.ToUpper().ToCharArray()

.Select(c => c.ToString())

.OrderBy(s => s);

var userSet = q.SelectedOptions

.Select(s => s.ToUpper())

.OrderBy(s => s);

if (correctSet.SequenceEqual(userSet))

correctCount++;

break;

case 'T':

if (!string.IsNullOrWhiteSpace(q.TextAnswer) &&

string.Equals(

q.TextAnswer.Trim(),

q.CorrectAnswers.Trim(),

StringComparison.OrdinalIgnoreCase))

correctCount++;

break;

}

}

// 2) Перевести в оцінку 0–100

int total = answers.Count;

int percentage = total > 0

? (int)Math.Round(correctCount \* 100.0 / total)

: 0;

// 3) Отримати ім'я з сесії

var userName = HttpContext.Session.GetString("UserName");

if (string.IsNullOrEmpty(userName))

return RedirectToAction("Login", "Account");

// 4) Зберегти в TestResults (DB)

using (var cn = new SqlConnection(\_conn))

{

cn.Open();

using var cmd = new SqlCommand(

@"INSERT INTO TestResults (UserName, Score, TakenAt)

VALUES (@user, @score, @time)", cn);

cmd.Parameters.AddWithValue("@user", userName);

cmd.Parameters.AddWithValue("@score", percentage);

cmd.Parameters.AddWithValue("@time", DateTime.Now);

cmd.ExecuteNonQuery();

}

// 5) Оновити поле Score у профілі (JSON через FileStore<User>)

var allUsers = \_users.Load();

var me = allUsers.First(u => u.Name == userName);

me.Score = percentage;

\_users.Save(allUsers);

// 6) Показати результат

return RedirectToAction("Results", new { score = percentage });

}

[HttpGet]

public IActionResult Results(int score)

{

ViewBag.Score = score;

return View();

}

}

}

**Views/Account/Profile.cshtml:**

@model Task\_4.Models.User

@{

ViewData["Title"] = "Profile";

}

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

<h2>Мій профіль</h2>

<ul class="list-unstyled">

<li><strong>Name:</strong> @Model.Name <a asp-action="EditName" class="ms-2">Edit</a></li>

<li><strong>Email:</strong> @Model.Email <a asp-action="EditEmail" class="ms-2">Edit</a></li>

<li><strong>Phone:</strong> @Model.Phone <a asp-action="EditPhone" class="ms-2">Edit</a></li>

<li><strong>Password:</strong> \*\*\*\*\*\*\*\* <a asp-action="EditPassword" class="ms-2">Edit</a></li>

<li><strong>Test Score:</strong> @Model.Score / 100</li>

</ul>

</div>

## **Результати роботи програми:**

Изображение выглядит как текст, облако, небо, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.1. Виконання заповнення форми питань

Изображение выглядит как небо, облако, текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.2. Виконання заповнення форми питань

Изображение выглядит как небо, облако, текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.3. Виконання заповнення форми питань

Изображение выглядит как облако, небо, снимок экрана, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.4. Результат у 100-бальній системі оцінювання

Изображение выглядит как облако, текст, небо, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4.5. Результат у 100-бальній системі оцінювання у формі Profile

## **Алгоритм побудови кодів**

1. **Оновити модель користувача**
   * У Models/User.cs додати властивість public int Score { get; set; }, яка зберігатиме останню оцінку 0–100.
2. **Зчитування та відображення питань**
   * У дії Take() контролера TestController завантажити всі питання з БД (SELECT \* FROM Questions), перемішати їх через OrderBy(\_ => rnd.Next()) і передати у View.
3. **Обробка відповідей (POST)**
   * Після надходження списку answers підрахувати кількість правильних відповідей (correctCount) за звичною логікою для радіо-, чекбокс- та текстових питань.
4. **Перетворення у шкалу 0–100**
   * Визначити загальну кількість питань total = answers.Count.
   * Обчислити percentage = (int)Math.Round(correctCount \* 100.0 / total).
5. **Збереження результату**
   * Перевірити, що користувач автентифікований (UserName у сесії).
   * Через ADO.NET вставити новий запис у таблицю TestResults з полями UserName, Score = percentage та TakenAt = DateTime.Now.
   * Разом з цим, завантажити через FileStore<User> JSON-файл користувачів, знайти поточного, присвоїти me.Score = percentage і зберегти оновлений список.
6. **Відображення підсумку**
   * Перенаправити на дію Results(int score), де в ViewBag.Score передати percentage для відображення.

## **Функціональність**

* Кожна вірна відповідь рахується як 1 бал.
* Після завершення тесту кількість правильних відповідей переводиться у відсотки від 0 до 100.
* Оцінка одночасно зберігається в реляційній БД (TestResults) та в профілі користувача (поле Score у JSON через FileStore<User>), що унеможливлює її ручну зміну.
* Відображення результату відбувається двома шляхами: на сторінці Results та в розділі «Мій профіль» поруч із іншими даними користувача.

## **Висновок**

Task\_4 дозволяє не лише перевіряти знання у вигляді правильних відповідей, а й відразу конвертувати їх у зрозумілу шкалу 0–100, зберігаючи цю оцінку у профілі користувача. Поєднання запису в БД для історії тестувань та оновлення поля Score у JSON гарантує прозорість і захищеність даних, що підвищує надійність системи оцінювання та зручність моніторингу успішності кожного користувача.

# **Завдання 5**

Розробити можливість обмеження кількості спроб проходження те-стування, наприклад не більше одного (двох чи трьох) разу. Після вичерпання усіх спроб тестування має бути недосяжним.

## **Текст програми:**

**Controllers/TestController.cs:**

// Controllers/TestController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Data.SqlClient;

using Microsoft.Extensions.Configuration;

using Task\_5.Models;

using Task\_5.Services;

namespace Task\_5.Controllers

{

public class TestController : Controller

{

private readonly string \_conn;

private readonly FileStore<User> \_users;

private const int AllowedAttempts = 3;

public TestController(IConfiguration cfg, FileStore<User> users)

{

\_conn = cfg.GetConnectionString("TestDb")!;

\_users = users;

}

[HttpGet]

public IActionResult Index()

{

// 1) Перевірка сесії

var userName = HttpContext.Session.GetString("UserName");

if (string.IsNullOrEmpty(userName))

return RedirectToAction("Login", "Account");

// 2) Порахувати вже використані спроби

int used;

using (var cn = new SqlConnection(\_conn))

{

cn.Open();

using var cmd = new SqlCommand(

"SELECT COUNT(\*) FROM TestResults WHERE UserName = @user", cn);

cmd.Parameters.AddWithValue("@user", userName);

used = (int)cmd.ExecuteScalar()!;

}

// 3) Передати у View

ViewBag.AttemptsLeft = Math.Max(0, AllowedAttempts - used);

ViewBag.AllowedAttempts = AllowedAttempts;

return View();

}

[HttpGet]

public IActionResult Take()

{

var userName = HttpContext.Session.GetString("UserName");

if (string.IsNullOrEmpty(userName))

return RedirectToAction("Login", "Account");

// Ліміт спроб

int used;

using (var cn = new SqlConnection(\_conn))

{

cn.Open();

using var cmd = new SqlCommand(

"SELECT COUNT(\*) FROM TestResults WHERE UserName = @user", cn);

cmd.Parameters.AddWithValue("@user", userName);

used = (int)cmd.ExecuteScalar()!;

}

if (used >= AllowedAttempts)

return RedirectToAction("Index");

// Зчитати й перемішати питання

var list = new List<TestAnswerViewModel>();

using (var cn = new SqlConnection(\_conn))

{

cn.Open();

using var cmd = new SqlCommand("SELECT \* FROM Questions", cn);

using var r = cmd.ExecuteReader();

while (r.Read())

{

list.Add(new TestAnswerViewModel

{

Id = (int)r["Id"],

Text = (string)r["Text"],

OptionA = (string)r["OptionA"],

OptionB = (string)r["OptionB"],

OptionC = (string)r["OptionC"],

OptionD = (string)r["OptionD"],

QuestionType = ((string)r["QuestionType"])[0],

CorrectAnswers = (string)r["CorrectAnswers"]

});

}

}

list = list.OrderBy(\_ => Guid.NewGuid()).ToList();

return View(list);

}

[HttpPost]

public IActionResult Take(List<TestAnswerViewModel> answers)

{

var userName = HttpContext.Session.GetString("UserName")!;

// Перевірити ліміт знову

int used;

using (var cn = new SqlConnection(\_conn))

{

cn.Open();

using var cmd = new SqlCommand(

"SELECT COUNT(\*) FROM TestResults WHERE UserName = @user", cn);

cmd.Parameters.AddWithValue("@user", userName);

used = (int)cmd.ExecuteScalar()!;

}

if (used >= AllowedAttempts)

return RedirectToAction("Index");

// Порахувати правильні

int correctCount = 0;

foreach (var q in answers)

{

switch (q.QuestionType)

{

case 'R':

if (q.SelectedOption.HasValue &&

q.CorrectAnswers.Equals(

q.SelectedOption.Value.ToString(),

StringComparison.OrdinalIgnoreCase))

correctCount++;

break;

case 'C':

var correctSet = q.CorrectAnswers

.ToUpper().ToCharArray()

.Select(c => c.ToString())

.OrderBy(s => s);

var userSet = q.SelectedOptions

.Select(s => s.ToUpper())

.OrderBy(s => s);

if (correctSet.SequenceEqual(userSet))

correctCount++;

break;

case 'T':

if (!string.IsNullOrWhiteSpace(q.TextAnswer) &&

string.Equals(

q.TextAnswer.Trim(),

q.CorrectAnswers.Trim(),

StringComparison.OrdinalIgnoreCase))

correctCount++;

break;

}

}

// Конвертувати в 0–100

int total = answers.Count;

int percentage = total > 0

? (int)Math.Round(correctCount \* 100.0 / total)

: 0;

// Зберегти в БД

using (var cn = new SqlConnection(\_conn))

{

cn.Open();

using var cmd = new SqlCommand(

@"INSERT INTO TestResults (UserName, Score, TakenAt)

VALUES (@user, @score, @time)", cn);

cmd.Parameters.AddWithValue("@user", userName);

cmd.Parameters.AddWithValue("@score", percentage);

cmd.Parameters.AddWithValue("@time", DateTime.Now);

cmd.ExecuteNonQuery();

}

// Оновити JSON-профіль

var allUsers = \_users.Load();

var me = allUsers.First(u => u.Name == userName);

me.Score = percentage;

\_users.Save(allUsers);

return RedirectToAction("Results", new { score = percentage });

}

[HttpGet]

public IActionResult Results(int score)

{

ViewBag.Score = score;

return View();

}

}

}

**Views/Test/Index.cshtml:**

@{

ViewData["Title"] = "Тест: Історія України";

int attemptsLeft = (int)ViewBag.AttemptsLeft;

int maxAttempts = (int)ViewBag.AllowedAttempts;

}

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

<h2>@ViewData["Title"]</h2>

@if (attemptsLeft > 0)

{

<p>

У вас залишилося спроб:

<strong>@attemptsLeft</strong> з

<strong>@maxAttempts</strong>.

</p>

<div class="d-flex justify-content-between">

<a asp-controller="Account" asp-action="Profile" class="btn btn-outline-secondary">

← Назад до профілю

</a>

<a asp-action="Take" class="btn btn-primary">Почати</a>

</div>

}

else

{

<p class="text-danger">

Ви використали всі дозволені спроби проходження тесту.

</p>

<a asp-controller="Account" asp-action="Profile" class="btn btn-outline-secondary">

← Назад до профілю

</a>

}

</div>

## **Результати роботи програми:**

Изображение выглядит как облако, текст, небо, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5.1. Обмеження кількості спроб проходження тестування на вхідній тестувальній формі

Изображение выглядит как облако, небо, снимок экрана, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5.2. Виконання 1-ої спроби

Изображение выглядит как облако, текст, небо, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5.3. Обмеження кількості спроб проходження тестування на вхідній тестувальній формі

Изображение выглядит как облако, небо, текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5.4. Виконання 2-ої спроби

Изображение выглядит как облако, текст, небо, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5.5. Обмеження кількості спроб проходження тестування на вхідній тестувальній формі

Изображение выглядит как облако, небо, снимок экрана, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5.6. Виконання 3-ої спроби

Изображение выглядит как облако, небо, текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5.7. Обмеження кількості спроб проходження тестування на вхідній тестувальній формі

Изображение выглядит как облако, текст, небо, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5.7. Вигляд оцінки проходження фінальної спроби тестування

## **Алгоритм побудови кодів**

1. **Конфігурація ліміту спроб**
   * У контролері TestController оголосити константу, наприклад

private const int AllowedAttempts = 3;

1. **Перевірка на старті (Index)**
   * Зчитати ім’я з сесії; якщо відсутнє — переадресувати на логін.
   * Запитати з БД кількість уже використаних спроб

SELECT COUNT(\*) FROM TestResults WHERE UserName = @user

* + Обчислити attemptsLeft = AllowedAttempts − used (не менше 0) і передати у ViewBag.

1. **GET-дія Take()**
   * Знову перевірити сесію та підрахувати used.
   * Якщо used >= AllowedAttempts, перенаправити на Index (тест недоступний).
   * Інакше: зчитати із таблиці Questions всі запитання, перемішати їх (OrderBy(Guid.NewGuid())) і віддати у View.
2. **POST-дія Take(List<TestAnswerViewModel>)**
   * Повторити перевірку сесії та ліміту used.
   * Порахувати число правильних відповідей (correctCount) за звичною логікою для R/C/T.
   * Обчислити відсоткову оцінку:

int percentage = (int)Math.Round(correctCount \* 100.0 / answers.Count);

* + Вставити новий запис у TestResults(UserName, Score, TakenAt).
  + Завантажити список користувачів з JSON через FileStore<User>, знайти поточного, оновити me.Score = percentage і зберегти файл.
  + Переадресувати на Results, передавши score = percentage.

1. **Дія Results(int score)**
   * Передати score у ViewBag й відобразити користувачу.

## **Функціональність**

* **Жорсткий ліміт спроб**: кожен користувач може пройти тест лише AllowedAttempts разів (за замовчуванням — 3).
* **Динамічне відображення залишку**: на сторінці Index показується, скільки спроб залишилось із максимально дозволених.
* **Блокування доступу**: при вичерпанні спроб кнопка «Почати» недоступна — замість неї виводиться повідомлення про закінчення ліміту.
* **Збереження результатів**: кожна спроба записується в реляційну БД (історія результатів) і одразу оновлює поле Score у профілі (JSON-файл), яке користувач не може змінити вручну.

## **Висновок**

Task\_5 реалізує механізм обмеження кількості проходжень тесту, що підвищує дисципліну та запобігає “машинному” підбору правильних відповідей. Опція відображення залишку спроб дозволяє прозоро інформувати користувача про його статус. Поєднання запису в БД та оновлення профілю гарантує коректний облік і збереження результатів кожної спроби.

# **Завдання 6**

Розробити роль модифікатора (майстра) тестів, який мав би можливість змінювати питання тестування, кількість спроб тестування та передивлятися оцінки тестування користувачів.

## **Текст програми:**

**Program.cs:**

// Program.cs

using Task\_6.Services;

using Task\_6.Models;

var builder = WebApplication.CreateBuilder(args);

// FileStore для користувачів

builder.Services.AddSingleton<FileStore<User>>(sp =>

new FileStore<User>(sp.GetRequiredService<IWebHostEnvironment>(), "users.json"));

// Сервіс налаштувань (ліміт спроб)

builder.Services.AddSingleton<SettingsService>();

builder.Services.AddControllersWithViews();

builder.Services.AddSession();

builder.Services.AddHttpContextAccessor();

var app = builder.Build();

app.UseStaticFiles();

app.UseSession();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Splash}/{action=Index}/{id?}"

);

app.Run();

**Models/User.cs:**

// Models/User.cs

namespace Task\_6.Models

{

public class User

{

public string Name { get; set; } = null!;

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

public string Phone { get; set; } = null!;

public string Password { get; set; } = null!;

// Остання тестова оцінка

public int Score { get; set; }

// Роль: "User" або "Master"

public string Role { get; set; } = "User";

}

}

**Models/TestSettings.cs:**

// Models/TestSettings.cs

namespace Task\_6.Models

{

public class TestSettings

{

// Ліміт спроб тесту

public int AllowedAttempts { get; set; }

}

}

**Services/SettingsService.cs:**

// Services/SettingsService.cs

using System.Text.Json;

using Task\_6.Models;

namespace Task\_6.Services

{

public class SettingsService

{

private readonly string \_path;

private readonly JsonSerializerOptions \_opts = new() { WriteIndented = true };

public SettingsService(IWebHostEnvironment env)

{

var dir = Path.Combine(env.ContentRootPath, "App\_Data");

if (!Directory.Exists(dir)) Directory.CreateDirectory(dir);

\_path = Path.Combine(dir, "settings.json");

// Якщо немає — створити з дефолтом 3 спроби

if (!File.Exists(\_path))

{

var def = new TestSettings { AllowedAttempts = 3 };

File.WriteAllText(\_path, JsonSerializer.Serialize(def, \_opts));

}

}

public TestSettings Load()

=> JsonSerializer.Deserialize<TestSettings>(File.ReadAllText(\_path))!;

public void Save(TestSettings s)

=> File.WriteAllText(\_path, JsonSerializer.Serialize(s, \_opts));

}

}

**Controllers/TestController.cs:**

// Controllers/TestController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Data.SqlClient;

using Microsoft.Extensions.Configuration;

using Task\_6.Models;

using Task\_6.Services;

namespace Task\_6.Controllers

{

public class TestController : Controller

{

private readonly string \_conn;

private readonly FileStore<User> \_users;

private readonly SettingsService \_settings;

public TestController(

IConfiguration cfg,

FileStore<User> users,

SettingsService settings)

{

\_conn = cfg.GetConnectionString("TestDb")!;

\_users = users;

\_settings = settings;

}

[HttpGet]

public IActionResult Index()

{

// 1) Перевірка, чи залогінений

var userName = HttpContext.Session.GetString("UserName");

if (string.IsNullOrEmpty(userName))

return RedirectToAction("Login", "Account");

// 2) Порахувати використані спроби з БД

int used;

using (var cn = new SqlConnection(\_conn))

{

cn.Open();

using var cmd = new SqlCommand(

"SELECT COUNT(\*) FROM TestResults WHERE UserName = @user", cn);

cmd.Parameters.AddWithValue("@user", userName);

used = (int)cmd.ExecuteScalar()!;

}

// 3) Отримати динамічний ліміт із налаштувань

int allowed = \_settings.Load().AllowedAttempts;

// 4) Передати до View

ViewBag.AttemptsLeft = Math.Max(0, allowed - used);

ViewBag.AllowedAttempts = allowed;

return View();

}

[HttpGet]

public IActionResult Take()

{

// 1) Авторизація

var userName = HttpContext.Session.GetString("UserName");

if (string.IsNullOrEmpty(userName))

return RedirectToAction("Login", "Account");

// 2) Порахувати вже використані спроби

int used;

using (var cn = new SqlConnection(\_conn))

{

cn.Open();

using var cmd = new SqlCommand(

"SELECT COUNT(\*) FROM TestResults WHERE UserName = @user", cn);

cmd.Parameters.AddWithValue("@user", userName);

used = (int)cmd.ExecuteScalar()!;

}

// 3) Динамічний ліміт

int allowed = \_settings.Load().AllowedAttempts;

if (used >= allowed)

return RedirectToAction("Index");

// 4) Завантажити та перемішати питання

var list = new List<TestAnswerViewModel>();

using (var cn = new SqlConnection(\_conn))

{

cn.Open();

using var cmd = new SqlCommand("SELECT \* FROM Questions", cn);

using var r = cmd.ExecuteReader();

while (r.Read())

{

list.Add(new TestAnswerViewModel

{

Id = (int)r["Id"],

Text = (string)r["Text"],

OptionA = (string)r["OptionA"],

OptionB = (string)r["OptionB"],

OptionC = (string)r["OptionC"],

OptionD = (string)r["OptionD"],

QuestionType = ((string)r["QuestionType"])[0],

CorrectAnswers = (string)r["CorrectAnswers"]

});

}

}

list = list.OrderBy(\_ => Guid.NewGuid()).ToList();

return View(list);

}

[HttpPost]

public IActionResult Take(List<TestAnswerViewModel> answers)

{

// 1) Авторизація

var userName = HttpContext.Session.GetString("UserName");

if (string.IsNullOrEmpty(userName))

return RedirectToAction("Login", "Account");

// 2) Перевірити ліміт знову

int used;

using (var cn = new SqlConnection(\_conn))

{

cn.Open();

using var cmd = new SqlCommand(

"SELECT COUNT(\*) FROM TestResults WHERE UserName = @user", cn);

cmd.Parameters.AddWithValue("@user", userName);

used = (int)cmd.ExecuteScalar()!;

}

int allowed = \_settings.Load().AllowedAttempts;

if (used >= allowed)

return RedirectToAction("Index");

// 3) Порахувати правильні відповіді

int correctCount = 0;

foreach (var q in answers)

{

switch (q.QuestionType)

{

case 'R':

if (q.SelectedOption.HasValue &&

string.Equals(

q.SelectedOption.Value.ToString(),

q.CorrectAnswers,

StringComparison.OrdinalIgnoreCase))

{

correctCount++;

}

break;

case 'C':

var correctSet = q.CorrectAnswers

.ToUpper().ToCharArray()

.Select(c => c.ToString())

.OrderBy(s => s);

var userSet = q.SelectedOptions

.Select(s => s.ToUpper())

.OrderBy(s => s);

if (correctSet.SequenceEqual(userSet))

correctCount++;

break;

case 'T':

if (!string.IsNullOrWhiteSpace(q.TextAnswer) &&

string.Equals(

q.TextAnswer.Trim(),

q.CorrectAnswers.Trim(),

StringComparison.OrdinalIgnoreCase))

{

correctCount++;

}

break;

}

}

// 4) Розрахувати оцінку 0–100

int total = answers.Count;

int percentage = total > 0

? (int)Math.Round(correctCount \* 100.0 / total)

: 0;

// 5) Зберегти в TestResults

using (var cn = new SqlConnection(\_conn))

{

cn.Open();

using var cmd = new SqlCommand(

@"INSERT INTO TestResults (UserName, Score, TakenAt)

VALUES (@user, @score, @time)", cn);

cmd.Parameters.AddWithValue("@user", userName);

cmd.Parameters.AddWithValue("@score", percentage);

cmd.Parameters.AddWithValue("@time", DateTime.Now);

cmd.ExecuteNonQuery();

}

// 6) Оновити JSON-профіль

var allUsers = \_users.Load();

var me = allUsers.First(u => u.Name == userName);

me.Score = percentage;

\_users.Save(allUsers);

return RedirectToAction("Results", new { score = percentage });

}

[HttpGet]

public IActionResult Results(int score)

{

ViewBag.Score = score;

return View();

}

}

}

**Controllers/AccountController.cs:**

// Controllers/AccountController.cs

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Http;

using Task\_6.Models;

using Task\_6.Services;

namespace Task\_6.Controllers

{

public class AccountController : Controller

{

private readonly FileStore<User> \_users;

public AccountController(FileStore<User> users)

{

\_users = users;

}

[HttpGet]

public IActionResult Register() => View();

[HttpPost]

public IActionResult Register(RegisterViewModel vm)

{

var all = \_users.Load();

if (all.Any(u => u.Name == vm.Name))

ModelState.AddModelError(nameof(vm.Name), "Name taken");

if (!ModelState.IsValid)

return View(vm);

all.Add(new User

{

Name = vm.Name,

Email = vm.Email,

Phone = vm.Phone,

Password = vm.Password,

Score = 0,

Role = "User"

});

\_users.Save(all);

return RedirectToAction("Login");

}

[HttpGet]

public IActionResult Login() => View();

[HttpPost]

public IActionResult Login(LoginViewModel vm)

{

if (!ModelState.IsValid)

return View(vm);

var user = \_users

.Load()

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

if (user == null)

{

ModelState.AddModelError(string.Empty, "Invalid credentials");

return View(vm);

}

HttpContext.Session.SetString("UserName", user.Name);

HttpContext.Session.SetString("UserRole", user.Role);

return RedirectToAction("Profile");

}

public IActionResult Logout()

{

HttpContext.Session.Clear();

return RedirectToAction("Login");

}

[HttpGet]

public IActionResult Profile()

{

var name = HttpContext.Session.GetString("UserName");

if (string.IsNullOrEmpty(name))

return RedirectToAction("Login");

var user = \_users.Load().First(u => u.Name == name);

return View(user);

}

[HttpGet]

public IActionResult EditName()

{

var name = HttpContext.Session.GetString("UserName");

if (string.IsNullOrEmpty(name))

return RedirectToAction("Login");

var user = \_users.Load().First(u => u.Name == name);

return View(new EditNameViewModel { Name = user.Name });

}

[HttpPost]

public IActionResult EditName(EditNameViewModel vm)

{

var old = HttpContext.Session.GetString("UserName");

if (string.IsNullOrEmpty(old))

return RedirectToAction("Login");

var all = \_users.Load();

if (all.Any(u => u.Name == vm.Name && u.Name != old))

ModelState.AddModelError(nameof(vm.Name), "Name taken");

if (!ModelState.IsValid)

return View(vm);

var user = all.First(u => u.Name == old);

user.Name = vm.Name;

\_users.Save(all);

HttpContext.Session.SetString("UserName", vm.Name);

return RedirectToAction("Profile");

}

[HttpGet]

public IActionResult EditEmail()

{

var name = HttpContext.Session.GetString("UserName");

if (string.IsNullOrEmpty(name))

return RedirectToAction("Login");

var user = \_users.Load().First(u => u.Name == name);

return View(new EditEmailViewModel { Email = user.Email });

}

[HttpPost]

public IActionResult EditEmail(EditEmailViewModel vm)

{

var name = HttpContext.Session.GetString("UserName");

if (string.IsNullOrEmpty(name))

return RedirectToAction("Login");

if (!ModelState.IsValid)

return View(vm);

var all = \_users.Load();

var user = all.First(u => u.Name == name);

user.Email = vm.Email;

\_users.Save(all);

return RedirectToAction("Profile");

}

[HttpGet]

public IActionResult EditPhone()

{

var name = HttpContext.Session.GetString("UserName");

if (string.IsNullOrEmpty(name))

return RedirectToAction("Login");

var user = \_users.Load().First(u => u.Name == name);

return View(new EditPhoneViewModel { Phone = user.Phone });

}

[HttpPost]

public IActionResult EditPhone(EditPhoneViewModel vm)

{

var name = HttpContext.Session.GetString("UserName");

if (string.IsNullOrEmpty(name))

return RedirectToAction("Login");

if (!ModelState.IsValid)

return View(vm);

var all = \_users.Load();

var user = all.First(u => u.Name == name);

user.Phone = vm.Phone;

\_users.Save(all);

return RedirectToAction("Profile");

}

[HttpGet]

public IActionResult EditPassword()

{

if (string.IsNullOrEmpty(HttpContext.Session.GetString("UserName")))

return RedirectToAction("Login");

return View();

}

[HttpPost]

public IActionResult EditPassword(EditPasswordViewModel vm)

{

var name = HttpContext.Session.GetString("UserName");

if (string.IsNullOrEmpty(name))

return RedirectToAction("Login");

if (!ModelState.IsValid)

return View(vm);

var all = \_users.Load();

var user = all.First(u => u.Name == name);

user.Password = vm.NewPassword;

\_users.Save(all);

return RedirectToAction("Profile");

}

}

}

**Controllers/MasterController.cs:**

// Controllers/MasterController.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.Data.SqlClient;

using Task\_6.Models;

using Task\_6.Services;

namespace Task\_6.Controllers

{

public class MasterController : Controller

{

private readonly FileStore<User> \_users;

private readonly SettingsService \_settings;

private readonly string \_conn;

public MasterController(

FileStore<User> users,

SettingsService settings,

IConfiguration cfg)

{

\_users = users;

\_settings = settings;

\_conn = cfg.GetConnectionString("TestDb")!;

}

// Захист: лише Master

private bool IsMaster() =>

HttpContext.Session.GetString("UserRole") == "Master";

private IActionResult Denied() =>

RedirectToAction("Index", "Account");

// 1) Список питань

[HttpGet]

public IActionResult Questions()

{

if (!IsMaster()) return Denied();

var list = new List<TestQuestion>();

using var cn = new SqlConnection(\_conn);

cn.Open();

using var cmd = new SqlCommand("SELECT \* FROM Questions", cn);

using var r = cmd.ExecuteReader();

while (r.Read())

{

list.Add(new TestQuestion

{

Id = (int)r["Id"],

Text = (string)r["Text"],

OptionA = (string)r["OptionA"],

OptionB = (string)r["OptionB"],

OptionC = (string)r["OptionC"],

OptionD = (string)r["OptionD"],

QuestionType = ((string)r["QuestionType"])[0],

CorrectAnswers = (string)r["CorrectAnswers"]

});

}

return View(list);

}

// 2) Створити нове питання

[HttpGet]

public IActionResult CreateQuestion()

{

if (!IsMaster()) return Denied();

return View();

}

[HttpPost]

public IActionResult CreateQuestion(TestQuestion q)

{

if (!IsMaster()) return Denied();

using var cn = new SqlConnection(\_conn);

cn.Open();

using var cmd = new SqlCommand(

@"INSERT INTO Questions

(Text,OptionA,OptionB,OptionC,OptionD,QuestionType,CorrectAnswers)

VALUES(@t,@a,@b,@c,@d,@type,@corr)", cn);

cmd.Parameters.AddWithValue("@t", q.Text);

cmd.Parameters.AddWithValue("@a", q.OptionA);

cmd.Parameters.AddWithValue("@b", q.OptionB);

cmd.Parameters.AddWithValue("@c", q.OptionC);

cmd.Parameters.AddWithValue("@d", q.OptionD);

cmd.Parameters.AddWithValue("@type", q.QuestionType.ToString());

cmd.Parameters.AddWithValue("@corr", q.CorrectAnswers);

cmd.ExecuteNonQuery();

return RedirectToAction("Questions");

}

// 3) Редагувати питання

[HttpGet]

public IActionResult EditQuestion(int id)

{

if (!IsMaster()) return Denied();

TestQuestion q = null!;

using var cn = new SqlConnection(\_conn);

cn.Open();

using var cmd = new SqlCommand(

"SELECT \* FROM Questions WHERE Id=@id", cn);

cmd.Parameters.AddWithValue("@id", id);

using var r = cmd.ExecuteReader();

if (r.Read()) q = new TestQuestion

{

Id = id,

Text = (string)r["Text"],

OptionA = (string)r["OptionA"],

OptionB = (string)r["OptionB"],

OptionC = (string)r["OptionC"],

OptionD = (string)r["OptionD"],

QuestionType = ((string)r["QuestionType"])[0],

CorrectAnswers = (string)r["CorrectAnswers"]

};

return View(q);

}

[HttpPost]

public IActionResult EditQuestion(TestQuestion q)

{

if (!IsMaster()) return Denied();

using var cn = new SqlConnection(\_conn);

cn.Open();

using var cmd = new SqlCommand(

@"UPDATE Questions SET

Text=@t, OptionA=@a, OptionB=@b, OptionC=@c, OptionD=@d,

QuestionType=@type, CorrectAnswers=@corr

WHERE Id=@id", cn);

cmd.Parameters.AddWithValue("@t", q.Text);

cmd.Parameters.AddWithValue("@a", q.OptionA);

cmd.Parameters.AddWithValue("@b", q.OptionB);

cmd.Parameters.AddWithValue("@c", q.OptionC);

cmd.Parameters.AddWithValue("@d", q.OptionD);

cmd.Parameters.AddWithValue("@type", q.QuestionType.ToString());

cmd.Parameters.AddWithValue("@corr", q.CorrectAnswers);

cmd.Parameters.AddWithValue("@id", q.Id);

cmd.ExecuteNonQuery();

return RedirectToAction("Questions");

}

// 4) Видалити питання

[HttpPost]

public IActionResult DeleteQuestion(int id)

{

if (!IsMaster()) return Denied();

using var cn = new SqlConnection(\_conn);

cn.Open();

using var cmd = new SqlCommand(

"DELETE FROM Questions WHERE Id=@id", cn);

cmd.Parameters.AddWithValue("@id", id);

cmd.ExecuteNonQuery();

return RedirectToAction("Questions");

}

// 5) Налаштування ліміту спроб

[HttpGet]

public IActionResult Settings()

{

if (!IsMaster()) return Denied();

var m = \_settings.Load();

return View(m);

}

[HttpPost]

public IActionResult Settings(TestSettings m)

{

if (!IsMaster()) return Denied();

\_settings.Save(m);

return RedirectToAction("Settings");

}

// 6) Перегляд оцінок користувачів

[HttpGet]

public IActionResult Scores()

{

if (!IsMaster()) return Denied();

var users = \_users.Load()

.OrderByDescending(u => u.Score)

.ToList();

return View(users);

}

}

}

**Views/Shared/\_Layout.cshtml:**

@{

var ctrl = ViewContext.RouteData.Values["controller"]?.ToString();

var act = ViewContext.RouteData.Values["action"]?.ToString();

var user = Context.Session.GetString("UserName");

var role = Context.Session.GetString("UserRole");

}

<!DOCTYPE html>

<html lang="uk">

<head>

<meta charset="utf-8" />

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

<title>@ViewData["Title"] – NovaChat</title>

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

<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" asp-controller="Splash" asp-action="Index">NovaChat</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#nav">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="nav">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

@\* При заході на Splash нема жодних лінків \*@

@if (ctrl == "Splash")

{

}

@\* На сторінці логіну — показати Register \*@

else if (ctrl == "Account" && act == "Login")

{

<li class="nav-item">

<a class="nav-link" asp-action="Register">Register</a>

</li>

}

@\* На сторінці реєстрації — показати Login \*@

else if (ctrl == "Account" && act == "Register")

{

<li class="nav-item">

<a class="nav-link" asp-action="Login">Login</a>

</li>

}

@\* Коли в профілі — показати Logout + Messages + Тест \*@

else if (ctrl == "Account" && act == "Profile")

{

<li class="nav-item">

<a class="nav-link" asp-action="Logout">Logout</a>

</li>

<li class="nav-item">

<a class="nav-link" asp-controller="Messages" asp-action="Send">Send Msg</a>

</li>

<li class="nav-item">

<a class="nav-link" asp-controller="Messages" asp-action="Inbox">Inbox</a>

</li>

<li class="nav-item">

<a class="nav-link" asp-controller="Messages" asp-action="Sent">Sent</a>

</li>

<li class="nav-item">

<a class="nav-link" asp-controller="Test" asp-action="Index">Тест</a>

</li>

}

@\* У Messages — лише «Back to Profile» \*@

else if (ctrl == "Messages" &&

(act == "Send" || act == "Inbox" || act == "Sent" || act == "Details"))

{

<li class="nav-item">

<a class="nav-link" asp-controller="Account" asp-action="Profile">Back to Profile</a>

</li>

}

@\* В режимах редагування профілю теж «Back to Profile» \*@

else if (ctrl == "Account" && act?.StartsWith("Edit") == true)

{

<li class="nav-item">

<a class="nav-link" asp-action="Profile">Back to Profile</a>

</li>

}

@\* Коли в TestController показуємо «Профіль» \*@

else if (ctrl == "Test")

{

<li class="nav-item">

<a class="nav-link" asp-controller="Account" asp-action="Profile">Профіль</a>

</li>

}

@\* Меню майстра для ролі Master \*@

@if (role == "Master")

{

<li class="nav-item">

<a class="nav-link" asp-controller="Master" asp-action="Questions">Manage Questions</a>

</li>

<li class="nav-item">

<a class="nav-link" asp-controller="Master" asp-action="Settings">Settings</a>

</li>

<li class="nav-item">

<a class="nav-link" asp-controller="Master" asp-action="Scores">User Scores</a>

</li>

}

</ul>

<span class="navbar-text text-muted">

@if (!string.IsNullOrEmpty(user))

{

@($"{user}{(string.IsNullOrEmpty(role) ? "" : $" ({role})")}")

}

</span>

</div>

</div>

</nav>

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

@RenderBody()

</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/dist/jquery.validate.unobtrusive.min.js"></script>

@RenderSection("Scripts", required: false)

</body>

</html>

**Views/Master/Questions.cshtml:**

@model List<Task\_6.Models.TestQuestion>

@{

Layout = "\_Layout";

ViewData["Title"] = "Manage Questions";

}

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

<h2>Manage Questions</h2>

<div class="mb-3">

<a asp-controller="Account" asp-action="Profile" class="btn btn-outline-secondary">

← Back to Profile

</a>

<a asp-action="CreateQuestion" class="btn btn-primary">

Create New

</a>

</div>

<table class="table table-striped">

<thead>

<tr>

<th style="width:5%">Id</th>

<th style="width:55%">Text</th>

<th style="width:5%">Type</th>

<th style="width:15%">Correct</th>

<th style="width:20%"></th>

</tr>

</thead>

<tbody>

@foreach (var q in Model)

{

<tr>

<td>@q.Id</td>

<td>@q.Text</td>

<td>@q.QuestionType</td>

<td>@q.CorrectAnswers</td>

<td>

<a asp-action="EditQuestion" asp-route-id="@q.Id" class="btn btn-sm btn-secondary">Edit</a>

<form asp-action="DeleteQuestion" method="post" class="d-inline">

<input type="hidden" name="id" value="@q.Id" />

<button type="submit" class="btn btn-sm btn-danger">Delete</button>

</form>

</td>

</tr>

}

</tbody>

</table>

</div>

**Views/Master/CreateQuestion.cshtml:**

@model Task\_6.Models.TestQuestion

@{

Layout = "\_Layout";

ViewData["Title"] = "Create Question";

}

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

<h2>Create Question</h2>

<div class="mb-3">

<a asp-controller="Account" asp-action="Profile" class="btn btn-outline-secondary">

← Back to Profile

</a>

</div>

<form asp-action="CreateQuestion" method="post">

<div class="mb-3">

<label asp-for="Text" class="form-label">Question Text</label>

<input asp-for="Text" class="form-control" />

</div>

<div class="row mb-3">

<div class="col">

<label asp-for="OptionA" class="form-label">Option A</label>

<input asp-for="OptionA" class="form-control" />

</div>

<div class="col">

<label asp-for="OptionB" class="form-label">Option B</label>

<input asp-for="OptionB" class="form-control" />

</div>

</div>

<div class="row mb-3">

<div class="col">

<label asp-for="OptionC" class="form-label">Option C</label>

<input asp-for="OptionC" class="form-control" />

</div>

<div class="col">

<label asp-for="OptionD" class="form-label">Option D</label>

<input asp-for="OptionD" class="form-control" />

</div>

</div>

<div class="mb-3">

<label asp-for="QuestionType" class="form-label">Type (R/C/T)</label>

<input asp-for="QuestionType" class="form-control" maxlength="1" />

</div>

<div class="mb-3">

<label asp-for="CorrectAnswers" class="form-label">Correct Answers</label>

<input asp-for="CorrectAnswers" class="form-control" />

</div>

<button type="submit" class="btn btn-success">Save</button>

</form>

</div>

**Views/Master/EditQuestion.cshtml:**

@model Task\_6.Models.TestQuestion

@{

Layout = "\_Layout";

ViewData["Title"] = "Edit Question";

}

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

<h2>Edit Question</h2>

<div class="mb-3">

<a asp-controller="Account" asp-action="Profile" class="btn btn-outline-secondary">

← Back to Profile

</a>

</div>

<form asp-action="EditQuestion" method="post">

<input asp-for="Id" type="hidden" />

<div class="mb-3">

<label asp-for="Text" class="form-label">Question Text</label>

<input asp-for="Text" class="form-control" />

</div>

<div class="row mb-3">

<div class="col">

<label asp-for="OptionA" class="form-label">Option A</label>

<input asp-for="OptionA" class="form-control" />

</div>

<div class="col">

<label asp-for="OptionB" class="form-label">Option B</label>

<input asp-for="OptionB" class="form-control" />

</div>

</div>

<div class="row mb-3">

<div class="col">

<label asp-for="OptionC" class="form-label">Option C</label>

<input asp-for="OptionC" class="form-control" />

</div>

<div class="col">

<label asp-for="OptionD" class="form-label">Option D</label>

<input asp-for="OptionD" class="form-control" />

</div>

</div>

<div class="mb-3">

<label asp-for="QuestionType" class="form-label">Type (R/C/T)</label>

<input asp-for="QuestionType" class="form-control" maxlength="1" />

</div>

<div class="mb-3">

<label asp-for="CorrectAnswers" class="form-label">Correct Answers</label>

<input asp-for="CorrectAnswers" class="form-control" />

</div>

<button type="submit" class="btn btn-success">Update</button>

</form>

</div>

**Views/Master/Settings.cshtml:**

@model Task\_6.Models.TestSettings

@{

Layout = "\_Layout";

ViewData["Title"] = "Test Settings";

}

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

<h2>Test Settings</h2>

<div class="mb-3">

<a asp-controller="Account" asp-action="Profile" class="btn btn-outline-secondary">

← Back to Profile

</a>

</div>

<form asp-action="Settings" method="post">

<div class="mb-3">

<label asp-for="AllowedAttempts" class="form-label">Allowed Attempts</label>

<input asp-for="AllowedAttempts" class="form-control" />

</div>

<button type="submit" class="btn btn-success">Save</button>

</form>

</div>

**Views/Master/Scores.cshtml:**

@model List<Task\_6.Models.User>

@{

Layout = "\_Layout";

ViewData["Title"] = "User Scores";

}

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

<h2>User Scores</h2>

<div class="mb-3">

<a asp-controller="Account" asp-action="Profile" class="btn btn-outline-secondary">

← Back to Profile

</a>

</div>

<table class="table table-striped">

<thead>

<tr>

<th>Name</th>

<th>Score</th>

</tr>

</thead>

<tbody>

@foreach (var u in Model)

{

<tr>

<td>@u.Name</td>

<td>@u.Score</td>

</tr>

}

</tbody>

</table>

</div>

## **Результати роботи програми:**

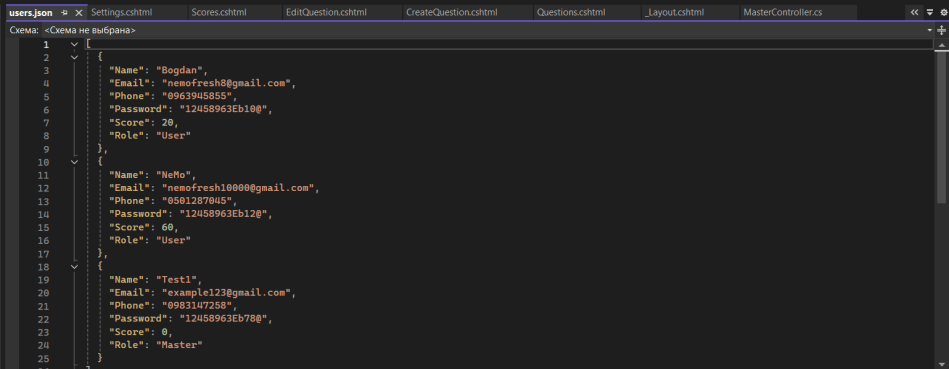


Рисунок 6.1. Надавання прав Master користувачу Test1

Изображение выглядит как текст, облако, небо, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.2. Вигляд оновленого меню з діями Master

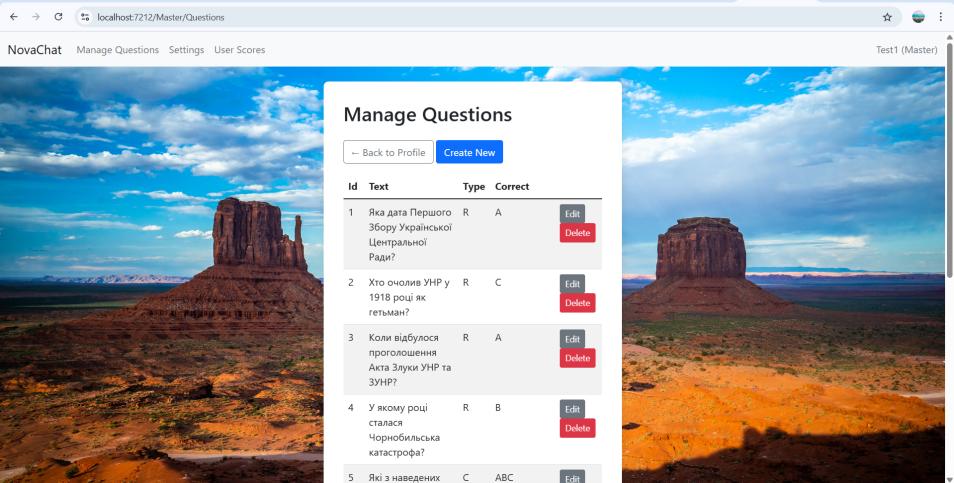


Рисунок 6.3. Форма редагування питань

Изображение выглядит как текст, небо, облако, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.4. Виконання редагування питання

Изображение выглядит как текст, небо, облако, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.5. Виконання редагування питання

Изображение выглядит как текст, небо, облако, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.6. Виконання редагування питання

Изображение выглядит как текст, небо, облако, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.7. Виконання створення питання

Изображение выглядит как небо, текст, облако, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.8. Виконання створення питання

Изображение выглядит как облако, текст, небо, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.9. Виконання змінення кількості спроб проходження тестування

Изображение выглядит как облако, текст, небо, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.10. Виконання змінення кількості спроб проходження тестування

Изображение выглядит как облако, текст, небо, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.11. Виконання змінення кількості спроб проходження тестування

Изображение выглядит как облако, небо, снимок экрана, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6.12. Перегляд оцінок тестування користувачів

## **Алгоритм побудови кодів**

1. **Оновлення моделей й сервісів**
   * У Models/User додати властивість Role («User» або «Master»).
   * Створити модель TestSettings та сервіс SettingsService для зчитування/запису AllowedAttempts у JSON (settings.json).
2. **Налаштування DI у Program.cs**
   * Зареєструвати FileStore<User> для роботи з users.json.
   * Зареєструвати SettingsService.
   * Додати сесію і MVC (AddSession(), AddControllersWithViews()).
3. **Реєстрація та логін**
   * При реєстрації (AccountController.Register) присвоювати новому користувачу Role = "User".
   * При логіні зберігати в сесії UserName і UserRole.
4. **Розмітка меню**
   * У \_Layout.cshtml перевіряти Session["UserRole"] == "Master", щоб показувати адміністраторські пункти:  
     – Управління запитаннями,  
     – Налаштування ліміту спроб,  
     – Перегляд оцінок користувачів.
5. **Контролер модифікатора (MasterController)**
   * **Authorize**: метод IsMaster() редіректить, якщо роль не “Master”.
   * **Questions**: CRUD (List, Create, Edit, Delete) запитань через ADO.NET (SqlCommand на Questions).
   * **Settings**: GET/POST для TestSettings через SettingsService.Load()/Save().
   * **Scores**: зчитати всіх користувачів із users.json та відобразити їхні поля Name і Score.
6. **Динамічний ліміт у TestController**
   * Зчитувати AllowedAttempts із SettingsService замість жорсткого константного значення.
   * Перевіряти і блокувати доступ до тесту за аналогією з попереднім завданням.

## **Функціональність**

* Користувачі з роллю **“User”** проходять тест у межах динамічного ліміту, який зберігається в settings.json.
* Користувачі з роллю **“Master”** бачать у меню розділи для:
  1. **Управління запитаннями** (CRUD у таблиці Questions),
  2. **Налаштування ліміту спроб**,
  3. **Перегляду оцінок** усіх користувачів із JSON-файлу.
* Всі зміни налаштувань і питань одним кліком записуються у відповідні сховища (SQL або JSON).

## **Висновок**

Task\_6 вводить модель ролей, що розділяє звичайних користувачів і «майстрів» тестів. Адміністратору надається інтерфейс для гнучкого керування контентом та налаштуваннями без залучення коду, а користувачам — прозорий механізм проходження тесту з динамічною кількістю спроб. Це забезпечує розширюваність, безпеку та зручність підтримки системи.

# **Загальний висновок**

У ході виконання лабораторної роботи № 10 було розроблено повноцінну веб-систему тестування користувачів на базі ASP NET Core MVC. Спершу реалізовано загальний функціонал реєстрації та авторизації, а потім – проходження тестів із різними типами питань (радіокнопки, чекбокси, текстовий ввід) з випадковим порядком відображення. Для збереження даних використано як SQL Server (через ADO .NET), так і файлове сховище JSON (через сервіс FileStore), що забезпечує гнучке налаштування середовища зберігання.

Додатково було впроваджено роль “майстра” (адміністратора) тестів, котрий може:

* змінювати питання тесту,
* задавати кількість дозволених спроб,
* переглядати та аналізувати результати користувачів.

У процесі роботи застосовано принципи MVC (розділення моделей, контролерів і подань), ін’єкцію залежностей, сесійний стан для зберігання даних користувача, а також адаптивний інтерфейс на базі Bootstrap. Особлива увага приділялась валідації даних на стороні сервера та клієнта й обробці можливих помилок.

**Ключові досягнення лабораторної роботи № 10:**

* Побудова комплексного ASP NET Core MVC-застосунку з чітким поділом на шари.
* Реалізація механізмів реєстрації, авторизації та управління сесією.
* Інтеграція двох способів зберігання даних: реляційної бази та JSON-файлів.
* Розробка ролі майстра тестів із розширеним функціоналом модифікації та перегляду результатів.
* Використання Bootstrap для створення адаптивного та зручного інтерфейсу.
* Опрацювання серверної та клієнтської валідації форм, обробки помилок і навігації користувача.

Загалом, виконана робота закріпила навички побудови масштабованих веб-систем на ASP NET Core та показала, як легко розширювати функціонал додатку відповідно до нових вимог.