Міністерство освіти України

Національний технічний університет "ХПІ"

кафедра "Інформатики та інтелектуальної власності"

**Звіт**

**Лабораторна робота 10**

з дисципліни "Проєктування серверних застосунків"

Виконав: студент групи КН-922В

Шершньов М.О.

Перевірив:

Івашко А.В.

Харків 2025

Зміст

[Завдання 3](#_Toc196218254)

[Класи geometricaltest, extendedtest, randomquestionorder, 100pointsystem, limitedattemps, moderatotmode 5](#_Toc196218255)

[Клас GeometricalTest 5](#_Toc196218256)

[Клас ExtendedTest 9](#_Toc196218257)

[Клас RandomQuestionOrder 13](#_Toc196218258)

[Клас 100PointSystem 15](#_Toc196218259)

[Клас LimitedAttemps 19](#_Toc196218260)

[Клас ModeratorMode 22](#_Toc196218261)

[Висновок 29](#_Toc196218262)

**Створення завершених MVC WEB-застосунків із обов'язковим стандартним 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

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.

Мій варіант:





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 варіантів відповідей вибираються декілька варіантів), які формуються за допомогою «чекбоксів» та три питання з безпосередньою відповіддю, яка друкується у відповідному текстовому полі чи полях.

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

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.

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

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

Класи для роботи:

1. «Геометричне Тестування» - **GeometricalTest**
2. «Розширений тест» – **ExtendedTest**
3. «Випадковий порядок питань» – **RandomQuestionOrder**
4. «100бальна система» – **100PointSystem**
5. «Обмеження кількості спроб» – **LimitedAttemps**
6. «Режим модератора» – **ModeratorMode**

Класи geometricaltest, extendedtest, randomquestionorder, 100pointsystem, limitedattemps, moderatotmode

Клас GeometricalTest

**Клас** GeometricalTest відповідає за реалізацію процесу проходження тесту з аналітичної геометрії у веб-додатку.

Основні функції класу:

1. **Ініціалізація тестування** — метод Start зчитує питання з JSON-файлу та зберігає їх у сесію.
2. **Виведення питань по одному** — метод Question динамічно показує поточне питання користувачу.
3. **Обробка відповідей** — перевіряє правильність вибраного варіанту, оновлює рахунок та зберігає у сесії.
4. **Підрахунок результату** — після завершення тесту показує кількість правильних відповідей у методі Result.
5. **Збереження питань** — окремий приватний метод LoadQuestions() читає файл math\_questions.json та десеріалізує його у список питань.
6. **Сесійне збереження стану** — усі дані зберігаються в HttpContext.Session, що дозволяє зберігати прогрес тесту між запитами.
7. **Перенаправлення** — у разі завершення тесту або неавторизованого доступу користувач перенаправляється на відповідні сторінки.

using Microsoft.AspNetCore.Mvc;

using System.Text.Json;

using Task\_1.Models;

public class TestController : Controller

{

private readonly string QuestionsPath = Path.Combine(Directory.GetCurrentDirectory(), "App\_Data", "math\_questions.json");

[HttpGet]

public IActionResult Start()

{

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

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

var questions = LoadQuestions();

HttpContext.Session.SetString("Questions", JsonSerializer.Serialize(questions));

HttpContext.Session.SetInt32("Score", 0);

HttpContext.Session.SetInt32("Current", 0);

return RedirectToAction("Question");

}

[HttpGet]

public IActionResult Question()

{

var json = HttpContext.Session.GetString("Questions");

var questions = JsonSerializer.Deserialize<List<TestQuestion>>(json!)!;

int current = HttpContext.Session.GetInt32("Current") ?? 0;

if (current >= questions.Count)

return RedirectToAction("Result");

ViewBag.Question = questions[current];

ViewBag.Index = current;

return View();

}

[HttpPost]

public IActionResult Question(int selectedOption)

{

var json = HttpContext.Session.GetString("Questions");

var questions = JsonSerializer.Deserialize<List<TestQuestion>>(json!)!;

int current = HttpContext.Session.GetInt32("Current") ?? 0;

int score = HttpContext.Session.GetInt32("Score") ?? 0;

if (questions[current].CorrectIndex == selectedOption)

score++;

HttpContext.Session.SetInt32("Current", current + 1);

HttpContext.Session.SetInt32("Score", score);

return RedirectToAction("Question");

}

[HttpGet]

public IActionResult Result()

{

int score = HttpContext.Session.GetInt32("Score") ?? 0;

ViewBag.Score = score;

return View();

}

private List<TestQuestion> LoadQuestions()

{

if (!System.IO.File.Exists(QuestionsPath))

return new List<TestQuestion>();

var json = System.IO.File.ReadAllText(QuestionsPath);

return JsonSerializer.Deserialize<List<TestQuestion>>(json)!;

}

}

@{

var q = ViewBag.Question as Task\_1.Models.TestQuestion;

int index = (int)ViewBag.Index;

ViewData["Title"] = $"Питання {index + 1}";

}

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" />

@if (q == null)

{

<div class="alert alert-danger mt-5 text-center">

Виникла помилка: питання не знайдене.

</div>

}

else

{

<div class="container text-center mt-5 bg-light bg-opacity-75 p-5 rounded shadow" style="max-width: 700px;">

<h4 class="mb-4">Питання @(@index + 1):</h4>

<h5 class="mb-4">@q.Question</h5>

<form asp-action="Question" method="post" class="text-start">

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

{

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

<input class="form-check-input" type="radio" name="selectedOption" value="@i" required />

<label class="form-check-label">@q.Options[i]</label>

</div>

}

<div class="mt-4 d-flex justify-content-between">

<a href="/Home/Welcome" class="btn btn-outline-secondary">Назад</a>

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

</div>

</form>

</div>

}

@{

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

}

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" />

<div class="container text-center mt-5 bg-light bg-opacity-75 p-5 rounded shadow" style="max-width: 600px;">

<h3 class="text-success mb-4">Результат тесту</h3>

<p class="fs-5">Ви набрали: <strong>@ViewBag.Score</strong> балів</p>

<a href="/Home/Welcome" class="btn btn-secondary mt-4">На головну</a>

</div>

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт, Прямоугольник

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунки 10.1-10.4 – Результат роботи класу GeometricalTest

Клас ExtendedTest

**Клас** ExtendedTest реалізує розширене тестування в ASP.NET MVC застосунку з підтримкою трьох типів запитань: радіокнопки, чекбокси та текстові відповіді.

Основні функції класу:

1. **Початок тестування (**Start**):**  
   Завантажує питання з JSON-файлу через DataStorage.LoadQuestions() і зберігає список питань у TempData разом із поточним індексом питання та кількістю балів.
2. **Виведення питання (**Question**):**  
   Показує одне питання за раз. Після кожного перегляду зберігає TempData через .Keep() для наступного запиту.
3. **Обробка відповіді (**Question [HttpPost]**)**. Перевіряє:
   * Одну правильну відповідь (radio) — за індексом
   * Кілька правильних відповідей (checkbox) — за множиною індексів
   * Текстову відповідь (text) — порівнює значення без урахування регістру. У разі правильної відповіді додає 1 бал до рахунку.
4. **Збереження проміжних результатів:**  
   Усі дані (Score, Current, Questions) оновлюються в TempData, що дозволяє зберегти стан тесту між HTTP-запитами.
5. **Підсумки тесту (**Result**):**  
   Метод Result отримує рахунок з TempData та передає його у ViewBag для відображення користувачу.

using Microsoft.AspNetCore.Mvc;

using System.Text.Json;

using Task\_2.Models;

public class TestController : Controller

{

[HttpGet]

public IActionResult Start()

{

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

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

var questions = DataStorage.LoadQuestions();

TempData["Questions"] = JsonSerializer.Serialize(questions);

TempData["Score"] = 0;

TempData["Current"] = 0;

return RedirectToAction("Question");

}

[HttpGet]

public IActionResult Question()

{

TempData.Keep();

var questions = JsonSerializer.Deserialize<List<TestQuestion>>(TempData.Peek("Questions") as string ?? "")!;

int current = Convert.ToInt32(TempData.Peek("Current"));

if (current >= questions.Count)

return RedirectToAction("Result");

ViewBag.Question = questions[current];

ViewBag.Index = current;

return View();

}

[HttpPost]

public IActionResult Question(string[]? selectedOptions, string? textAnswer)

{

var questions = JsonSerializer.Deserialize<List<TestQuestion>>(TempData["Questions"] as string ?? "")!;

int current = Convert.ToInt32(TempData["Current"]);

int score = Convert.ToInt32(TempData["Score"]);

var q = questions[current];

bool isCorrect = false;

if (q.Type == "radio" && selectedOptions?.Length == 1)

{

isCorrect = q.CorrectIndexes.Contains(int.Parse(selectedOptions[0]));

}

else if (q.Type == "checkbox" && selectedOptions != null)

{

var selected = selectedOptions.Select(int.Parse).OrderBy(i => i);

var correct = q.CorrectIndexes.OrderBy(i => i);

isCorrect = selected.SequenceEqual(correct);

}

else if (q.Type == "text" && textAnswer != null)

{

isCorrect = q.CorrectAnswerText?.Trim().ToLower() == textAnswer.Trim().ToLower();

}

if (isCorrect) score++;

TempData["Score"] = score;

TempData["Current"] = current + 1;

TempData["Questions"] = JsonSerializer.Serialize(questions);

return RedirectToAction("Question");

}

[HttpGet]

public IActionResult Result()

{

int score = Convert.ToInt32(TempData["Score"]);

ViewBag.Score = score;

return View();

}

}

namespace Task\_2.Models;

public class TestQuestion

{

public string Id { get; set; } = Guid.NewGuid().ToString();

public string Question { get; set; } = "";

public string Type { get; set; } = "radio"; // radio | checkbox | text

public List<string> Options { get; set; } = new(); // Для radio, checkbox

public List<int> CorrectIndexes { get; set; } = new(); // Мульти-варіант

public string? CorrectAnswerText { get; set; } // Для текстових

}

@{

var q = ViewBag.Question as Task\_2.Models.TestQuestion;

int index = (int)ViewBag.Index;

ViewData["Title"] = $"Питання {index + 1}";

}

@if (q == null)

{

<div class="alert alert-danger">Питання не знайдено.</div>

}

else

{

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" />

<div class="container text-center mt-5 bg-light bg-opacity-75 p-5 rounded shadow" style="max-width: 800px;">

<h5 class="mb-4">Питання @(@index + 1):</h5>

<h4 class="mb-4">@q.Question</h4>

<**form** **asp-action**="Question" method="post" class="text-start">

@if (q.Type == "radio")

{

for (int i = 0; i < q.Options.Count; i++)

{

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

<input class="form-check-input" type="radio" name="selectedOptions" value="@i" required />

<label class="form-check-label">@q.Options[i]</label>

</div>

}

}

else if (q.Type == "checkbox")

{

for (int i = 0; i < q.Options.Count; i++)

{

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

<input class="form-check-input" type="checkbox" name="selectedOptions" value="@i" />

<label class="form-check-label">@q.Options[i]</label>

</div>

}

}

else if (q.Type == "text")

{

<input type="text" name="textAnswer" class="form-control" placeholder="Ваша відповідь..." required />

}

<div class="mt-4 d-flex justify-content-between">

<a href="/Home/Welcome" class="btn btn-outline-secondary">На головну</a>

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

</div>

</**form**>

</div>

}

@{

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

}

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" />

<div class="container text-center mt-5 bg-light bg-opacity-75 p-5 rounded shadow" style="max-width: 600px;">

<h3 class="text-success mb-4">Результат тесту</h3>

<p class="fs-5">Ви набрали: <strong>@ViewBag.Score</strong> балів з 10</p>

<a href="/Home/Welcome" class="btn btn-secondary mt-4">На головну</a>

</div>

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт, Прямоугольник

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт, Прямоугольник

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунки 10.5-10.7 – Результат роботи класу ExtendedTest

Клас RandomQuestionOrder

**Клас RandomQuestionOrder** відповідає за реалізацію проходження тестування з випадковим порядком запитань у веб-додатку.

Основні функції класу:

**Початок тестування (Start)**

* + Отримує ім’я поточного користувача із сесії.
  + Завантажує всі запитання з файлу через DataStorage.LoadQuestions().
  + Використовує Guid.NewGuid() для перемішування запитань у випадковому порядку.
  + Зберігає список запитань, поточний індекс і кількість балів у TempData.

using Microsoft.AspNetCore.Mvc;

using System.Text.Json;

using Task\_3.Models;

public class TestController : Controller

{

[HttpGet]

public IActionResult Start()

{

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

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

var questions = DataStorage.LoadQuestions()

.OrderBy(q => Guid.NewGuid())

.ToList();

TempData["Questions"] = JsonSerializer.Serialize(questions);

TempData["Score"] = 0;

TempData["Current"] = 0;

return RedirectToAction("Question");

}

[HttpGet]

public IActionResult Question()

{

TempData.Keep();

var questions = JsonSerializer.Deserialize<List<TestQuestion>>(TempData.Peek("Questions") as string ?? "")!;

int current = Convert.ToInt32(TempData.Peek("Current"));

if (current >= questions.Count)

return RedirectToAction("Result");

ViewBag.Question = questions[current];

ViewBag.Index = current;

return View();

}

[HttpPost]

public IActionResult Question(string[]? selectedOptions, string? textAnswer)

{

var questions = JsonSerializer.Deserialize<List<TestQuestion>>(TempData["Questions"] as string ?? "")!;

int current = Convert.ToInt32(TempData["Current"]);

int score = Convert.ToInt32(TempData["Score"]);

var q = questions[current];

bool isCorrect = false;

if (q.Type == "radio" && selectedOptions?.Length == 1)

{

isCorrect = q.CorrectIndexes.Contains(int.Parse(selectedOptions[0]));

}

else if (q.Type == "checkbox" && selectedOptions != null)

{

var selected = selectedOptions.Select(int.Parse).OrderBy(i => i);

var correct = q.CorrectIndexes.OrderBy(i => i);

isCorrect = selected.SequenceEqual(correct);

}

else if (q.Type == "text" && textAnswer != null)

{

isCorrect = q.CorrectAnswerText?.Trim().ToLower() == textAnswer.Trim().ToLower();

}

if (isCorrect) score++;

TempData["Score"] = score;

TempData["Current"] = current + 1;

TempData["Questions"] = JsonSerializer.Serialize(questions);

return RedirectToAction("Question");

}

[HttpGet]

public IActionResult Result()

{

int score = Convert.ToInt32(TempData["Score"]);

ViewBag.Score = score;

return View();

}

}

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт, Прямоугольник

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунки 10.8-10.9 – Результат роботи класу RandomQuestionOrder

Клас 100PointSystem

**Клас 100PointSystem** реалізує тестування користувача з автоматичним підрахунком оцінки за 100-бальною шкалою у веб-додатку.

Основні функції:

1. Завантажує випадковий список питань та зберігає стан у TempData.
2. Обробляє відповіді на питання трьох типів: "radio", "checkbox", "text".
3. За кожну правильну відповідь додає 1 бал до рахунку.
4. Після проходження тесту обчислює підсумкову оцінку як score × 10.
5. Зберігає результат у профілі користувача (якщо ще не збережено).

using Microsoft.AspNetCore.Mvc;

using System.Text.Json;

using Task\_4.Models;

public class TestController : Controller

{

[HttpGet]

public IActionResult Start()

{

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

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

var questions = DataStorage.LoadQuestions()

.OrderBy(q => Guid.NewGuid())

.ToList();

TempData["Questions"] = JsonSerializer.Serialize(questions);

TempData["Score"] = 0;

TempData["Current"] = 0;

return RedirectToAction("Question");

}

[HttpGet]

public IActionResult Question()

{

TempData.Keep();

var questions = JsonSerializer.Deserialize<List<TestQuestion>>(TempData.Peek("Questions") as string ?? "")!;

int current = Convert.ToInt32(TempData.Peek("Current"));

if (current >= questions.Count)

return RedirectToAction("Result");

ViewBag.Question = questions[current];

ViewBag.Index = current;

return View();

}

[HttpPost]

public IActionResult Question(string[]? selectedOptions, string? textAnswer)

{

var questions = JsonSerializer.Deserialize<List<TestQuestion>>(TempData["Questions"] as string ?? "")!;

int current = Convert.ToInt32(TempData["Current"]);

int score = Convert.ToInt32(TempData["Score"]);

var q = questions[current];

bool isCorrect = false;

if (q.Type == "radio" && selectedOptions?.Length == 1)

{

isCorrect = q.CorrectIndexes.Contains(int.Parse(selectedOptions[0]));

}

else if (q.Type == "checkbox" && selectedOptions != null)

{

var selected = selectedOptions.Select(int.Parse).OrderBy(i => i);

var correct = q.CorrectIndexes.OrderBy(i => i);

isCorrect = selected.SequenceEqual(correct);

}

else if (q.Type == "text" && textAnswer != null)

{

isCorrect = q.CorrectAnswerText?.Trim().ToLower() == textAnswer.Trim().ToLower();

}

if (isCorrect) score++;

TempData["Score"] = score;

TempData["Current"] = current + 1;

TempData["Questions"] = JsonSerializer.Serialize(questions);

return RedirectToAction("Question");

}

[HttpGet]

public IActionResult Result()

{

int score = Convert.ToInt32(TempData["Score"]);

int percent = score \* 10;

ViewBag.Score = score;

ViewBag.Percent = percent;

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

var users = DataStorage.LoadUsers();

var user = users.FirstOrDefault(u => u.UserName == userName);

if (user != null && user.TestScore == null)

{

user.TestScore = percent;

DataStorage.SaveUsers(users);

}

return View();

}

}

@{

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

}

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" />

<div class="container text-center mt-5 bg-light bg-opacity-75 p-5 rounded shadow" style="max-width: 600px;">

<h3 class="text-success mb-4">Результат тесту</h3>

<p class="fs-5">Правильних відповідей: <strong>@ViewBag.Score / 10</strong></p>

<p class="fs-5">Підсумкова оцінка: <strong>@ViewBag.Percent балів</strong></p>

<a href="/Home/Welcome" class="btn btn-secondary mt-4">На головну</a>

</div>

Изображение выглядит как текст, снимок экрана, Шрифт, Прямоугольник

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунки 10.10-10.11 – Результат роботи класу 100PointSystem

Клас LimitedAttemps

**Клас LimitedAttempts** реалізує тестування з обмеженою кількістю спроб.

Основні функції:

1. Обмежує кількість спроб проходження тесту до 2-х (через константу MaxTestAttempts).
2. Визначає користувача через сесію та перевіряє його кількість попередніх спроб.
3. Якщо спроби вичерпані — відображає повідомлення та блокує повторне тестування.
4. Зберігає кількість спроб і підсумкову оцінку в профіль користувача.
5. Показує повідомлення про кількість використаних спроб у профілі (Welcome).

using Microsoft.AspNetCore.Mvc;

using System.Text.Json;

using Task\_5.Models;

public class TestController : Controller

{

private const int MaxTestAttempts = 2;

[HttpGet]

public IActionResult Start()

{

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

if (userName == null)

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

var users = DataStorage.LoadUsers();

var user = users.FirstOrDefault(u => u.UserName == userName);

if (user == null) return RedirectToAction("Login", "Home");

if (user.TestAttempts >= MaxTestAttempts)

{

TempData["Error"] = "Ви вичерпали кількість спроб проходження тесту.";

return RedirectToAction("Welcome", "Home");

}

user.TestAttempts++;

DataStorage.SaveUsers(users);

// Загрузка та випадкове сортування

var questions = DataStorage.LoadQuestions()

.OrderBy(q => Guid.NewGuid())

.ToList();

TempData["Questions"] = JsonSerializer.Serialize(questions);

TempData["Score"] = 0;

TempData["Current"] = 0;

return RedirectToAction("Question");

}

[HttpGet]

public IActionResult Question()

{

TempData.Keep();

var questions = JsonSerializer.Deserialize<List<TestQuestion>>(TempData.Peek("Questions") as string ?? "")!;

int current = Convert.ToInt32(TempData.Peek("Current"));

if (current >= questions.Count)

return RedirectToAction("Result");

ViewBag.Question = questions[current];

ViewBag.Index = current;

return View();

}

[HttpPost]

public IActionResult Question(string[]? selectedOptions, string? textAnswer)

{

var questions = JsonSerializer.Deserialize<List<TestQuestion>>(TempData["Questions"] as string ?? "")!;

int current = Convert.ToInt32(TempData["Current"]);

int score = Convert.ToInt32(TempData["Score"]);

var q = questions[current];

bool isCorrect = false;

if (q.Type == "radio" && selectedOptions?.Length == 1)

{

isCorrect = q.CorrectIndexes.Contains(int.Parse(selectedOptions[0]));

}

else if (q.Type == "checkbox" && selectedOptions != null)

{

var selected = selectedOptions.Select(int.Parse).OrderBy(i => i);

var correct = q.CorrectIndexes.OrderBy(i => i);

isCorrect = selected.SequenceEqual(correct);

}

else if (q.Type == "text" && textAnswer != null)

{

isCorrect = q.CorrectAnswerText?.Trim().ToLower() == textAnswer.Trim().ToLower();

}

if (isCorrect) score++;

TempData["Score"] = score;

TempData["Current"] = current + 1;

TempData["Questions"] = JsonSerializer.Serialize(questions);

return RedirectToAction("Question");

}

[HttpGet]

public IActionResult Result()

{

int score = Convert.ToInt32(TempData["Score"]);

int correctAnswers = score / 10;

int percent = (int)Math.Round((double)score, 0);

ViewBag.Score = percent;

ViewBag.CorrectAnswers = correctAnswers;

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

if (userName != null)

{

var users = DataStorage.LoadUsers();

var user = users.FirstOrDefault(u => u.UserName == userName);

if (user != null && user.TestScore == null)

{

user.TestScore = percent;

DataStorage.SaveUsers(users);

}

}

return View();

}

}

@{

ViewData["Title"] = "Ласкаво просимо";

}

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" />

<div class="container text-center mt-5 bg-light p-5 rounded shadow" style="max-width: 600px;">

<h2 class="mb-4">Вітаємо, @ViewBag.UserName!</h2>

<div class="text-start">

<p><strong>Email:</strong> @ViewBag.Email</p>

<p><strong>Дата народження:</strong> @ViewBag.BirthDate</p>

@if (ViewBag.TestScore != null)

{

<p><strong>Оцінка за тест:</strong> @ViewBag.TestScore балів</p>

}

@if (ViewBag.TestAttempts != null)

{

<p><strong>Спроб використано:</strong> @ViewBag.TestAttempts / 2</p>

@if ((int)ViewBag.TestAttempts >= 2)

{

<p class="text-danger">Ви вичерпали дозволену кількість спроб тестування.</p>

}

}

</div>

<div class="mt-4 d-flex flex-column gap-2">

<a href="/Home/Profile" class="btn btn-primary">Редагувати профіль</a>

<a href="/Home/SendMessage" class="btn btn-success">Написати повідомлення</a>

<a href="/Home/Inbox" class="btn btn-danger">Вхідні повідомлення</a>

<a href="/Test/Start" class="btn btn-warning">Пройти тест з аналітичної геометрії</a>

<a href="/Home/Login" class="btn btn-secondary">Разлогінитись</a>

</div>

</div>

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 10.12 – Результат роботи класу LimitedAttemps

Клас ModeratorMode

**Клас ModeratorMode** реалізує функціонал адміністративної панелі веб-додатку для модератора тестів.

Основні функції:

1. **Доступ лише для адміністратора** — перевіряє, чи поточний користувач має роль IsAdmin.
2. **Перегляд користувачів** — виводить список усіх зареєстрованих користувачів із оцінками та кількістю спроб.
3. **Управління питаннями тесту**:
   * Додавання нового питання.
   * Редагування існуючого (текст, тип, варіанти, правильні відповіді).
   * Видалення запитань.
4. **Редагування параметрів тестування** — зміна максимальної кількості дозволених спроб тесту (через AdminStorage).
5. **Збереження всіх змін** — дані оновлюються у JSON-файлах (questions.json, config.json).

using Microsoft.AspNetCore.Mvc;

using Task\_6.Models;

public class AdminController : Controller

{

[HttpGet]

public IActionResult Panel()

{

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

var user = DataStorage.LoadUsers().FirstOrDefault(u => u.UserName == userName);

if (user == null || !user.IsAdmin)

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

ViewBag.Users = DataStorage.LoadUsers();

ViewBag.Questions = DataStorage.LoadQuestions();

ViewBag.MaxAttempts = AdminStorage.MaxTestAttempts;

return View();

}

[HttpGet]

public IActionResult Edit(string id)

{

var questions = DataStorage.LoadQuestions();

var q = questions.FirstOrDefault(q => q.Id == id);

if (q == null) return RedirectToAction("Panel");

return View(q);

}

[HttpPost]

public IActionResult UpdateAttempts(int newLimit)

{

AdminStorage.MaxTestAttempts = newLimit;

return RedirectToAction("Panel");

}

[HttpPost]

public IActionResult DeleteQuestion(string id)

{

var list = DataStorage.LoadQuestions();

list.RemoveAll(q => q.Id == id);

DataStorage.SaveQuestions(list);

return RedirectToAction("Panel");

}

[HttpPost]

public IActionResult EditQuestion(IFormCollection form)

{

var questions = DataStorage.LoadQuestions();

var id = form["Id"];

var q = questions.FirstOrDefault(q => q.Id == id);

if (q == null) return RedirectToAction("Panel");

q.Question = form["Question"].ToString();

q.Type = form["Type"].ToString();

q.Options = form["OptionsString"].ToString().Split('|', StringSplitOptions.RemoveEmptyEntries)

.Select(o => o.Trim()).ToList();

q.CorrectIndexes = form["CorrectIndexesString"].ToString().Split(',')

.Where(x => int.TryParse(x, out \_)).Select(int.Parse).ToList();

q.CorrectAnswerText = form["CorrectAnswerText"];

DataStorage.SaveQuestions(questions);

return RedirectToAction("Panel");

}

[HttpPost]

public IActionResult AddQuestion(IFormCollection form)

{

var questions = DataStorage.LoadQuestions();

var q = new TestQuestion

{

Id = Guid.NewGuid().ToString(),

Question = form["Question"].ToString(),

Type = form["Type"].ToString(),

Options = form["OptionsString"].ToString().Split('|', StringSplitOptions.RemoveEmptyEntries)

.Select(o => o.Trim()).ToList(),

CorrectIndexes = form["CorrectIndexesString"].ToString().Split(',')

.Where(x => int.TryParse(x, out \_)).Select(int.Parse).ToList(),

CorrectAnswerText = form["CorrectAnswerText"]

};

questions.Add(q);

DataStorage.SaveQuestions(questions);

return RedirectToAction("Panel");

}

}

using System.Text.Json;

public static class AdminStorage

{

private static string ConfigPath => Path.Combine(Directory.GetCurrentDirectory(), "App\_Data", "config.json");

public static int MaxTestAttempts

{

get

{

if (!File.Exists(ConfigPath))

return 2;

var json = File.ReadAllText(ConfigPath);

var dict = JsonSerializer.Deserialize<Dictionary<string, int>>(json);

return dict != null && dict.ContainsKey("MaxAttempts") ? dict["MaxAttempts"] : 2;

}

set

{

var dict = new Dictionary<string, int> { { "MaxAttempts", value } };

var json = JsonSerializer.Serialize(dict, new JsonSerializerOptions { WriteIndented = true });

File.WriteAllText(ConfigPath, json);

}

}

}

@model Task\_6.Models.TestQuestion

@{

ViewData["Title"] = "Редагування питання";

}

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" />

<div class="container mt-5" style="max-width: 800px;">

<div class="bg-light bg-opacity-75 p-4 rounded shadow">

<h3>Редагування питання</h3>

<**form** **asp-action**="EditQuestion" method="post">

<input type="hidden" name="Id" value="@Model.Id" />

<div class="mb-2">

<label>Текст:</label>

<input name="Question" class="form-control" value="@Model.Question" />

</div>

<div class="mb-2">

<label>Тип:</label>

<select name="Type" class="form-select">

<**option** **value**="radio" selected="@(Model.Type == "radio")">Один варіант</**option**>

<**option** **value**="checkbox" selected="@(Model.Type == "checkbox")">Кілька варіантів</**option**>

<**option** **value**="text" selected="@(Model.Type == "text")">Текст</**option**>

</select>

</div>

<div class="mb-2">

<label>Варіанти (через |):</label>

<input name="OptionsString" class="form-control" value="@string.Join(" | ", Model.Options)" />

</div>

<div class="mb-2">

<label>Правильні індекси (через ,):</label>

<input name="CorrectIndexesString" class="form-control" value="@string.Join(",", Model.CorrectIndexes)" />

</div>

<div class="mb-2">

<label>Правильна текстова відповідь:</label>

<input name="CorrectAnswerText" class="form-control" value="@Model.CorrectAnswerText" />

</div>

<div class="d-flex justify-content-between mt-3">

<a href="/Admin/Panel" class="btn btn-secondary">Назад</a>

<button class="btn btn-primary">Зберегти</button>

</div>

</**form**>

</div>

</div>

@{

ViewData["Title"] = "Адмін Панель";

}

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" />

<div class="container mt-5" style="max-width: 960px;">

<div class="bg-light bg-opacity-75 p-4 rounded shadow mb-5">

<h3>Користувачі</h3>

<table class="table table-striped table-hover">

<thead>

<tr>

<th>Ім'я</th>

<th>Оцінка</th>

<th>Спроб</th>

</tr>

</thead>

<tbody>

@foreach (var u in ViewBag.Users)

{

<tr>

<td>@u.UserName</td>

<td>@u.TestScore</td>

<td>@u.TestAttempts</td>

</tr>

}

</tbody>

</table>

</div>

<div class="bg-light bg-opacity-75 p-4 rounded shadow mb-5">

<h3>Питання</h3>

<table class="table table-striped table-hover">

<thead>

<tr>

<th>Питання</th>

<th>Тип</th>

<th>Дії</th>

</tr>

</thead>

<tbody>

@foreach (var q in ViewBag.Questions)

{

<tr>

<td>@q.Question</td>

<td>@q.Type</td>

<td class="d-flex gap-2">

<**a** **asp-controller**="Admin" **asp-action**="Edit" **asp-route-id**="@q.Id" class="btn btn-sm btn-warning">Змінити</**a**>

<**form** **asp-action**="DeleteQuestion" method="post" class="d-inline">

<input type="hidden" name="id" value="@q.Id" />

<button class="btn btn-sm btn-danger">Видалити</button>

</**form**>

</td>

</tr>

}

</tbody>

</table>

</div>

<div class="bg-light bg-opacity-75 p-4 rounded shadow mb-5">

<h4>Додати нове питання</h4>

<**form** **asp-action**="AddQuestion" method="post">

<div class="mb-2">

<label>Текст:</label>

<input name="Question" class="form-control" required />

</div>

<div class="mb-2">

<label>Тип:</label>

<select name="Type" class="form-select">

<**option** **value**="radio">Один варіант</**option**>

<**option** **value**="checkbox">Кілька варіантів</**option**>

<**option** **value**="text">Текст</**option**>

</select>

</div>

<div class="mb-2">

<label>Варіанти (через |):</label>

<input name="OptionsString" class="form-control" />

</div>

<div class="mb-2">

<label>Правильні індекси (через ,):</label>

<input name="CorrectIndexesString" class="form-control" />

</div>

<div class="mb-2">

<label>Правильна текстова відповідь:</label>

<input name="CorrectAnswerText" class="form-control" />

</div>

<button class="btn btn-success mt-2">Додати</button>

</**form**>

</div>

<div class="bg-light bg-opacity-75 p-4 rounded shadow">

<h4>Макс. кількість спроб: @ViewBag.MaxAttempts</h4>

<**form** **asp-action**="UpdateAttempts" method="post" class="d-flex align-items-center gap-3">

<input name="newLimit" type="number" class="form-control w-auto" value="@ViewBag.MaxAttempts" />

<button class="btn btn-primary">Оновити</button>

</**form**>

</div>

<div class="text-end mt-4">

<a href="/Home/Welcome" class="btn btn-secondary">Назад</a>

</div>

</div>

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунки 10.13-10.15 – Результат роботи класу TextfileSaving

Висновок

У ході виконання цієї лабораторної роботи я здобув практичні навички розробки повноцінного MVC ASP.NET WEB-застосунку з розширеним функціоналом. Було реалізовано стандартні й додаткові компоненти: форму-заставку (LogoForm), реєстрацію та авторизацію користувачів (AccountRegistration, AccountAuth), профільну форму з можливістю редагування (ProfileForm), систему обміну повідомленнями (MessageForm, ReceivingMessageForm), а також функціонал проходження тестування.

Особливу увагу приділено модулю тестування: реалізовано тестову систему з обробкою варіантів відповідей (radio, checkbox, text), оцінюванням у 100-бальній системі (100PointSystem), обмеженням кількості спроб (LimitedAttempts) та довільним порядком запитань (RandomQuestionOrder). Для адміністрування створено окремий режим модератора (ModeratorMode), який дозволяє керувати питаннями, оцінками та параметрами тесту.

Інформація зберігається у текстових JSON-файлах (users.json, messages.json, config.json), що дало змогу відмовитись від використання бази даних і гнучко управляти даними.

Ця робота допомогла мені краще зрозуміти принципи побудови масштабованих, зручних та безпечних веб-застосунків з використанням архітектури MVC та засобів ASP.NET.