НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Програмне забезпечення для підвищення ефективності роботи математичного тренажера.

Виконав студент IV курсу, групи ІТ-93
(шифр групи)

Пономарьов Олексій Вікторович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник асистент Вовк Є. А. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації ст. викладач Головченко М. М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент ст. викладач Галушко Д. О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення комп'ютерних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Пономарьову Олексію Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту — Програмне забезпечення для підвищення ефективності роботи математичного тренажера

керівник проєкту — Вовк Євгеній Андрійович, асистент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «31» травня 2023 р. № 2101-с

2. Термін подання студентом проєкту « 17 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки

1) Аналіз вимог до програмного забезпечення: змістовний опис і аналіз предметної області, аналіз відомих технічних рішень та програмних продуктів, розробка функціональних та нефункціональних вимог.
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, конструювання та архітектура програмного, забезпечення.
3) Аналіз якості та тестування програмного забезпечення.
4) Впровадження та супровід програмного забезпечення.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань _____
2) Схема бази даних _____
3) Схема структурна класів програмного забезпечення _____
_____
_____
_____

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» березня 2023 року _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2023	
2	Аналіз існуючих методів розв'язання задачі	24.03.2023	
3	Постановка та формалізація задачі	07.04.2023	
4	Розробка інформаційного забезпечення	21.04.2023	
5	Алгоритмізація задачі	28.04.2023	
6	Обґрунтування вибору використаних технічних засобів	05.05.2023	
7	Розробка програмного забезпечення	12.05.2023	
8	Налагодження програми	19.05.2023	
9	Виконання графічних документів	26.05.2023	
10	Оформлення пояснювальної записки	01.06.2023	
11	Подання ДП на попередній захист	02.06.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	17.06.2023	

Студент _____ Олексій ПОНОМАРЬОВ
(підпис)

Керівник _____ Євгеній ВОВК
(підпис)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 36 таблиць, 18 рисунків та 46 джерел – загалом 75 сторінок.

Дипломний проєкт присвячений підвищенню ефективності роботи математичного тренажера.

Мета: Покращення якісної ефективності математичного тренажеру.

Об'єкт дослідження: програмне забезпечення покращення знань шляхом проведення тестувань з метою підвищення якості знань та навичок з математики.

Предмет дослідження: дослідження способів покращення ефективності математичного тренажера.

У першому розділі проведено аналіз наявних технічних засобів та існуючих рішень, розроблені функціональні та нефункціональні вимоги до застосунку.

У другому було розроблено архітектуру застосунку. Побудовано схему архітектури застосунку, бізнес процеси застосунку, а також опис класів застосування та опис структури файлів.

У третьому розділі було наведено та проведене тестування застосунку, його функціональних вимог та нефункціональних. Описано результати тестування у таблиці.

У четвертому розділі описано спосіб розгортання застосунку та його впровадження.

У додатках наведено: опис програми, схема структурна компонентів, схема структурна послідовності, креслення вигляду екранних форм.

КЛЮЧОВІ СЛОВА: ВЕБ-ЗАСТОСУНОК, МАТЕМАТИЧНИЙ ТРЕНАЖЕР, МАТЕМАТИКА.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 36 tables, 18 figures and 46 sources – in total 75 pages.

The diploma project is dedicated to increasing the efficiency of the mathematical simulator.

Goal: Improving the qualitative effectiveness of the mathematical simulator.

Object of research: software for improving knowledge by conducting tests in order to improve the quality of knowledge and skills in mathematics.

The subject of the study: the study of ways to improve the qualitative effectiveness of the mathematical simulator.

In the first section, an analysis of available technical means and existing solutions was carried out, functional and non-functional requirements for the application were developed.

In the second, the application architecture was developed. A diagram of the application architecture, business processes of the application, as well as a description of the application classes and a description of the file structure have been built.

In the third section, testing of the application, its functional and non-functional requirements was given and conducted. The test results are described in the table.

The fourth chapter describes how to deploy the application and implement it.

The appendices include: a description of the program, a structural diagram of components, a structural diagram of the sequence, and a drawing of the appearance of the screen forms.

KEY WORDS: WEB APPLICATION, MATHEMATICAL SIMULATOR, MATHEMATICS.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІТ-9303.045440.01.91	Технічне завдання	19	
3	А4	КПІ.ІТ-9303.045440.02.81	Пояснювальна записка	71	
4	А4	КПІ.ІТ-9303.045440.03.12	Текст програми	29	
5	А4	КПІ.ІТ-9303.045440.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІТ-9303.045440.05.34	Керівництво користувача	10	
7	А3	КПІ.ІТ-9303.045440.06.99	Схема структурна варіантів використання;	1	
8	А3	КПІ.ІТ-9303.045440.07.99	Схема бази даних	1	
9	А3	КПІ.ІТ-9303.045440.08.99	Схема структурна класів програмного забезпечення	2	

					КПІ.ІТ-9303.045440.00.90		
Змін.	Арк.	№ докум.	Підп.	Дата			
Розроб.		Пономарьов О.В.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів 1
Перевір.		Вовк Є.А.					
						КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-93	
Н.контр.		Головченко М.М.					
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОБОТИ МАТЕМАТИЧНОГО ТРЕНАЖЕРА

Технічне завдання

КПІ.ІТ-9303.045440.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Євгеній ВОВК

Нормоконтроль:
_____Максим ГОЛОВЧЕНКО

Виконавець:
_____Олексій ПОНОМАРЬОВ

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для підвищення ефективності роботи математичного тренажера.

Галузь застосування: Навчання та освіта.

Наведене технічне завдання поширюється на розробку програмного забезпечення підвищення ефективності роботи математичного тренажера, котра використовується для покращення знань та навичок з математики та призначена для шкіл та університетів, користувачами можуть бути наприклад учні та студенти.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки програмного забезпечення для підвищення ефективності роботи математичного тренажера є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для покращення роботи математичного тренажеру. Розробка призначена для учнів та студентів навчальних закладів для покращення знань і навичок з математики.

Метою розробки є покращення ефективності математичного тренажеру.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу

Перегляд функцій зображено на рисунках 4.1-4.11

4.1.1.1 Обирання теми тесту

Натискання на кнопку “Pick Quiz” на рисунку 4.1

Обирання теми з переліку. Рисунок 4.2

4.1.1.2 Проходження тесту

Обирання відповідей на запитання натискаючи на radio buttons. Рисунок 4.3

4.1.1.3 Отримання результату

Натискання на кнопку Show Results на рисунку 4.3 і демонстрація результату на рисунку 4.4.

4.1.1.4 Реєстрація користувача

Натискання кнопку “Sign Up”, заповнення всіх полей, натискання кнопку “Sign Up” (Рисунок 4.5). Після цього перегляд повідомлення про успішну або не успішну реєстрацію з наступною переадресацією на сторінку авторизації(Рисунок 4.6).

4.1.1.5 Авторизація

Вводиться логін та пароль, натискається кнопка “Login”, отримується повідомлення про успішну або не успішну авторизацію. Рисунок 4.7.

4.1.1.6 Перегляд профілю

Натискається кнопка “Show Profile” на рисунку 4.1. Переглядається профіль(Рисунок 4.8).

4.1.1.7 Редагування профілю

Натискається кнопка “Edit Profile” на рисунку 4.1. Редагується профіль профіль(Рисунок 4.9), натискається кнопка ”Save”. Перенаправлення на перегляд профілю(Рисунок 4.10).

4.1.1.8 Вихід з системи

Натискається кнопка “Logout” на рисунку 4.1. Перенаправлення на форму входу(Рисунок 4.6)

4.1.1.9 Видалення профілю

Натискається кнопка “Delete Profile” на рисунку 4.1. Відображення підтвердження видалення поточного профілю (Рисунок 4.11) Натискання yes або no підтверджує або ні факт видалення користувача.

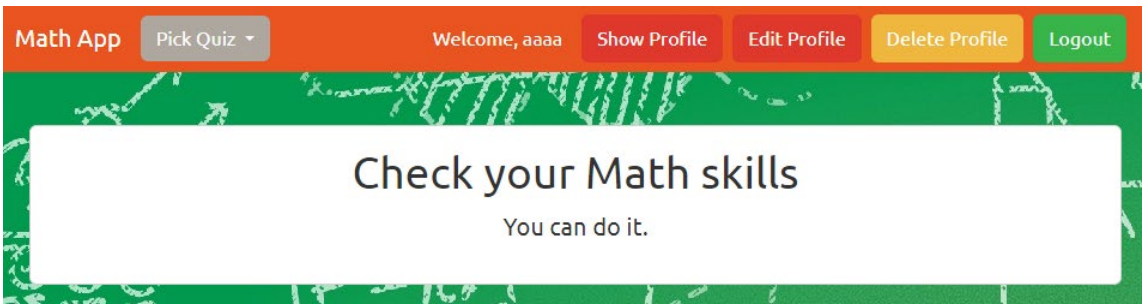


Рисунок 4.1 – Навігаційна панель

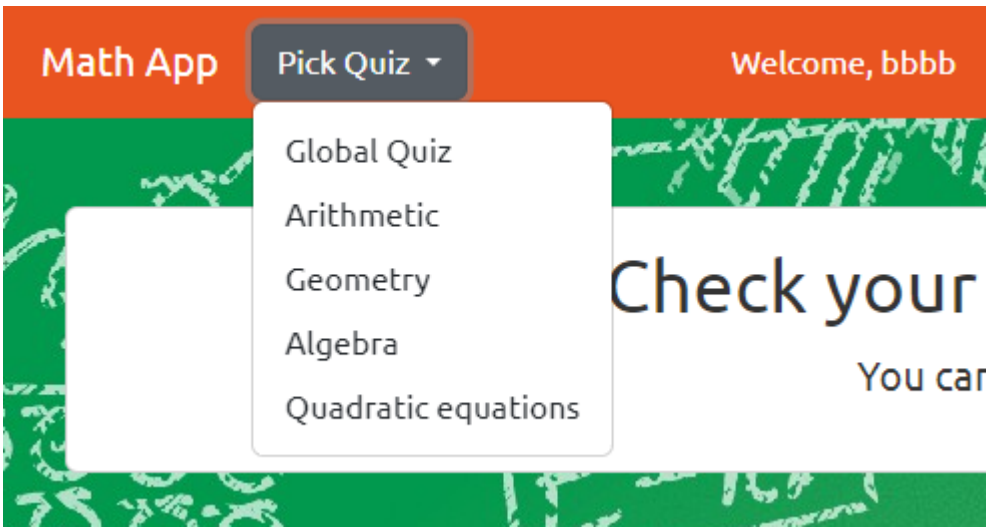


Рисунок 4.2 – Вибір теми тесту

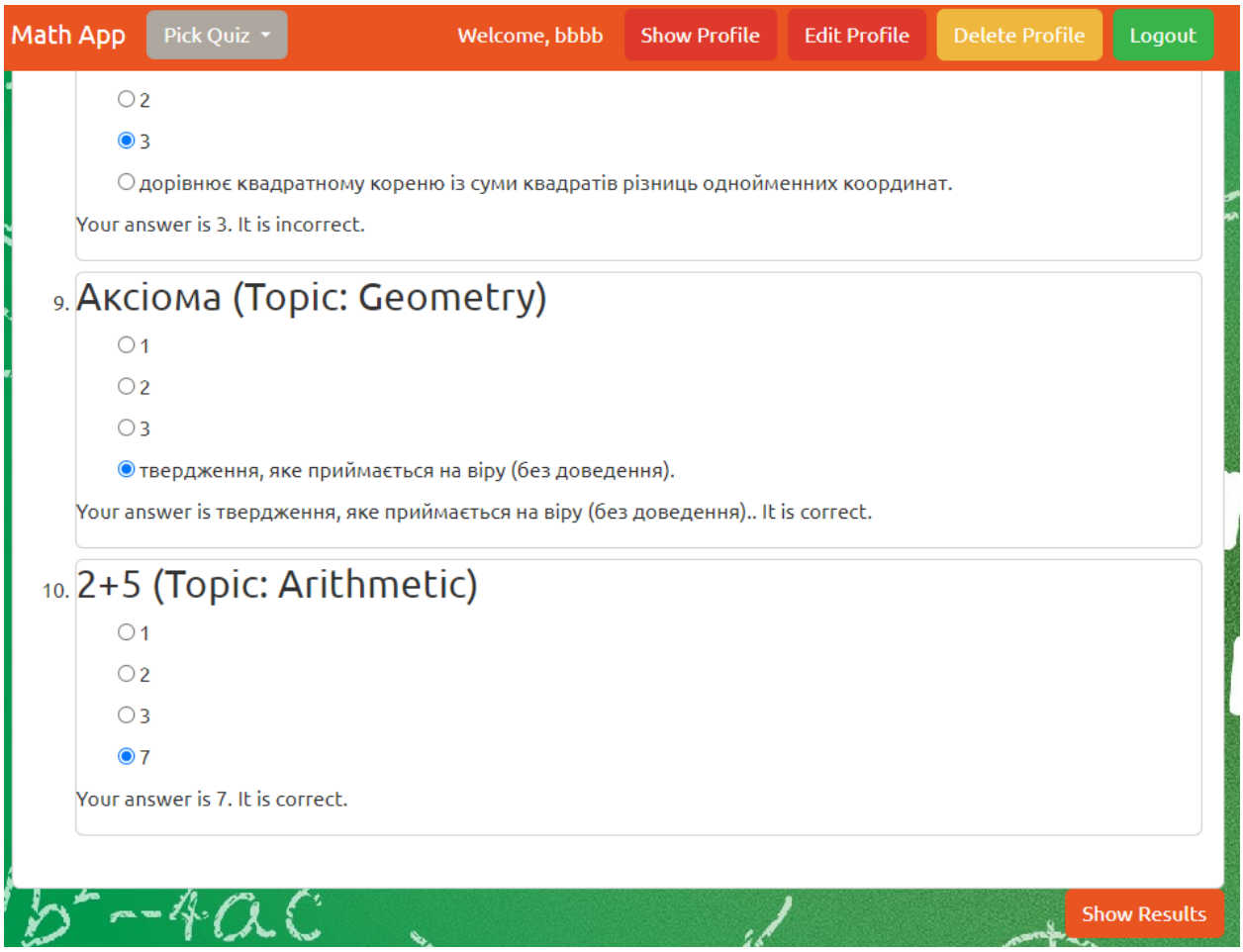


Рисунок 4.3 – Проходження тесту

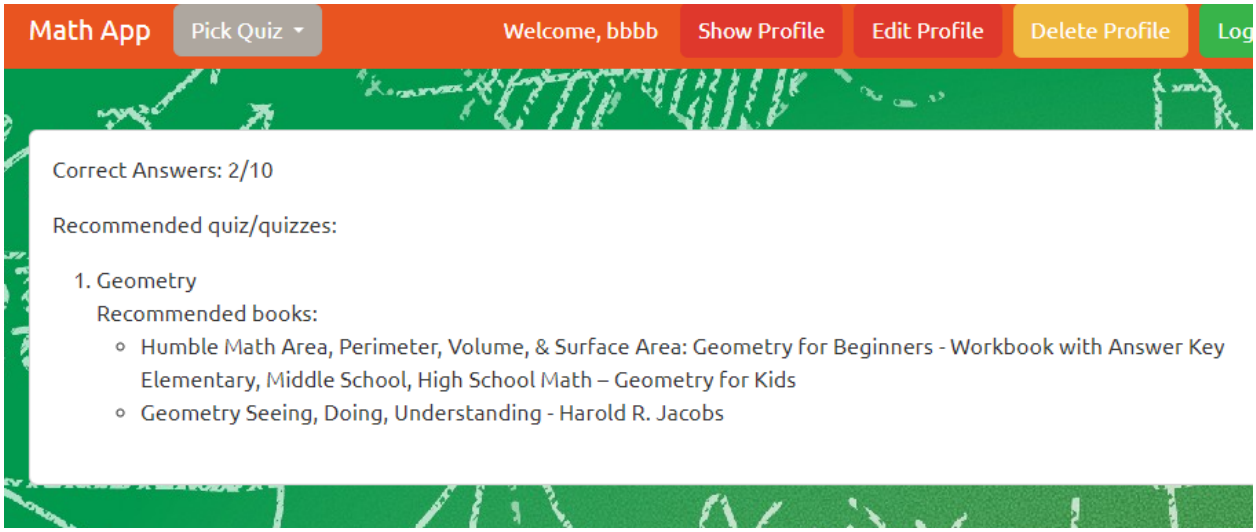


Рисунок 4.4 – Перегляд результатів тесту

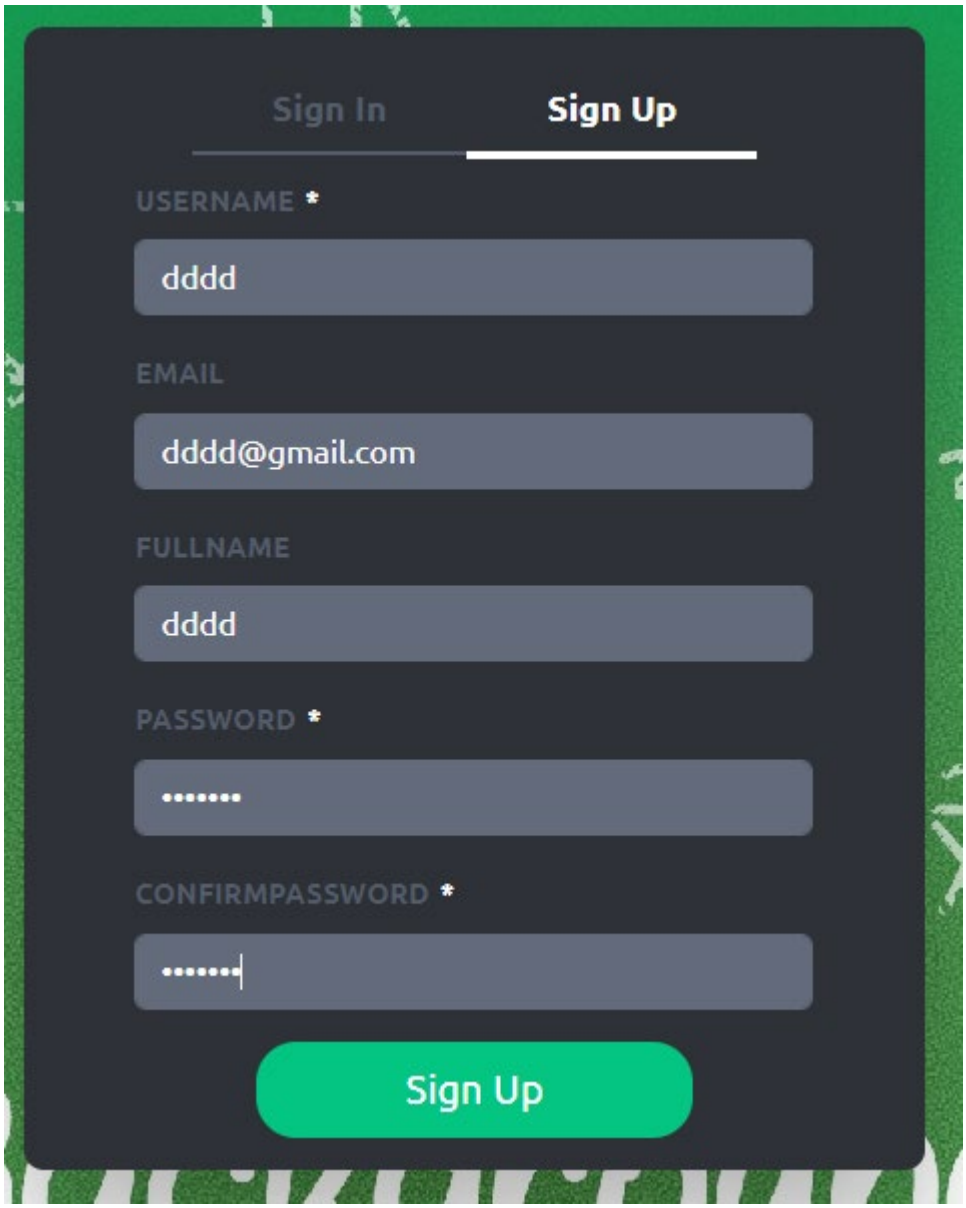


Рисунок 4.5 – Заповнення форми реєстрації

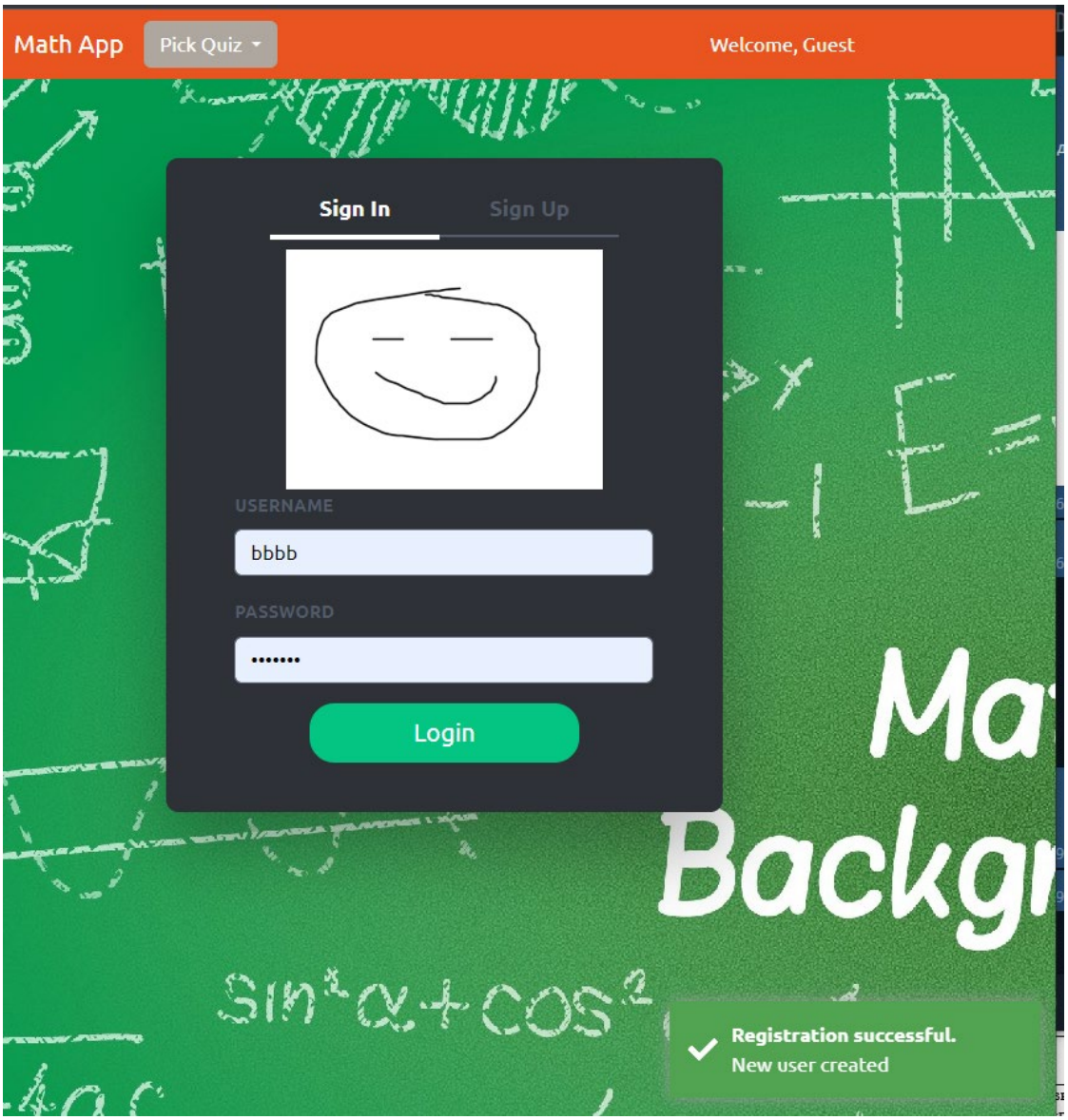


Рисунок 4.6 – Успішна реєстрація

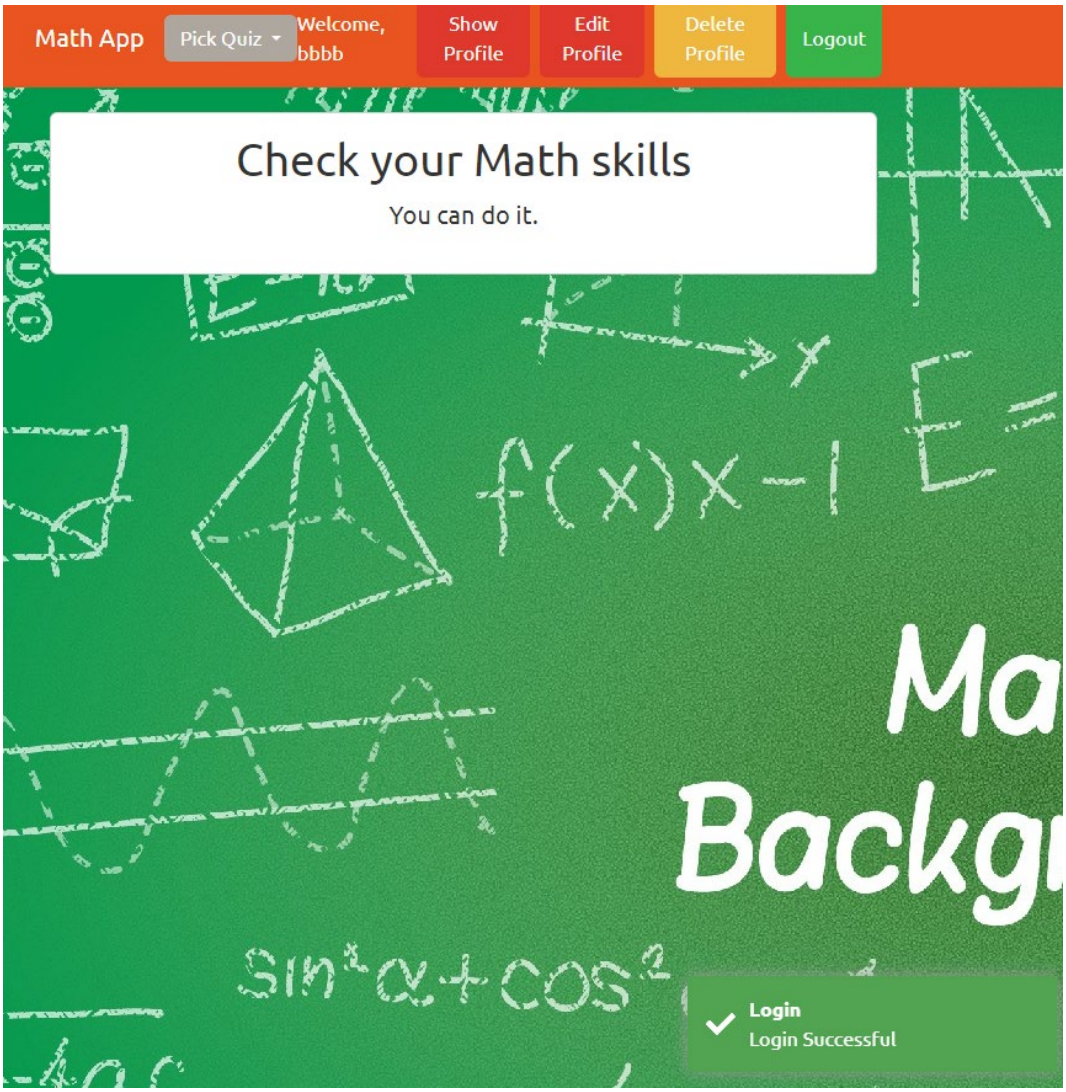


Рисунок 4.7 – Успішна авторизація

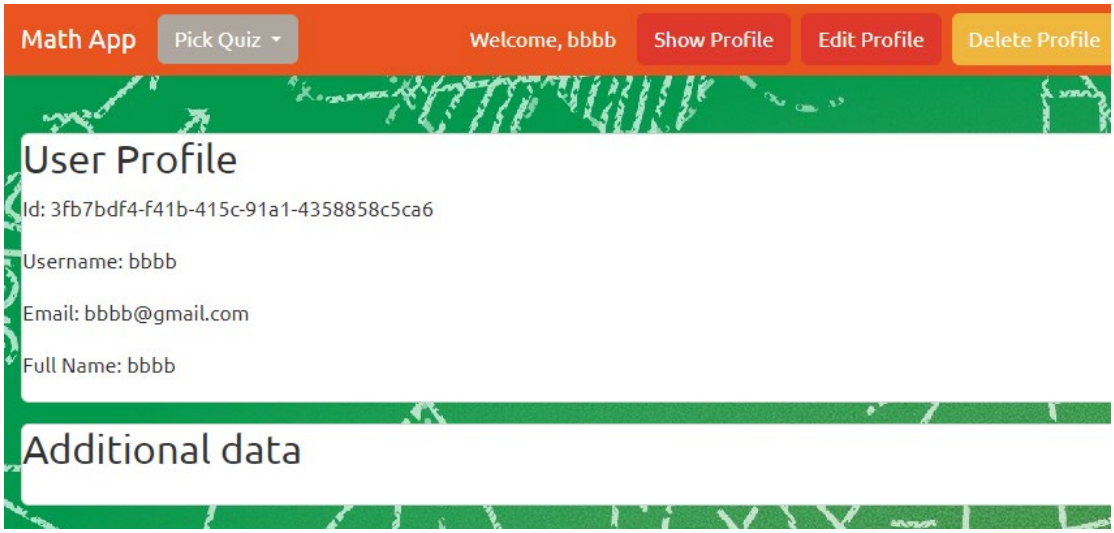


Рисунок 4.8 – Перегляд профілю

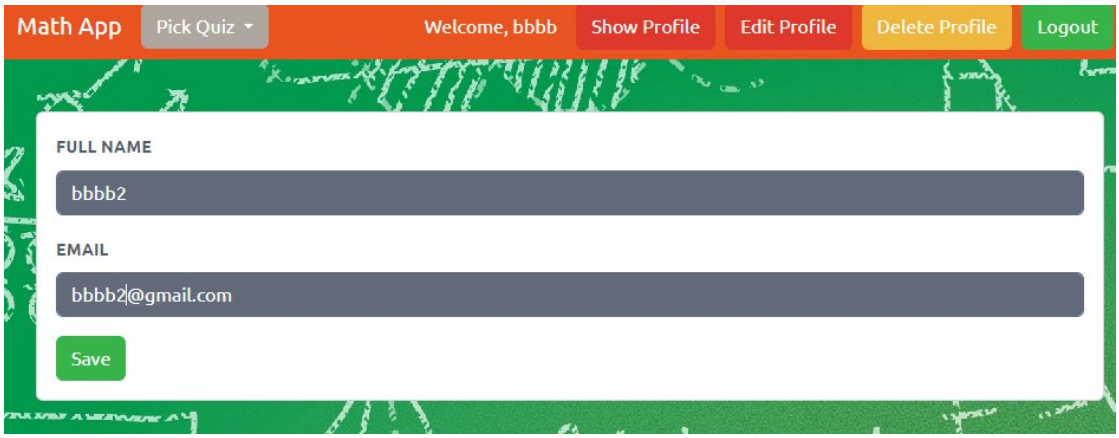


Рисунок 4.9 – Редагування профілю

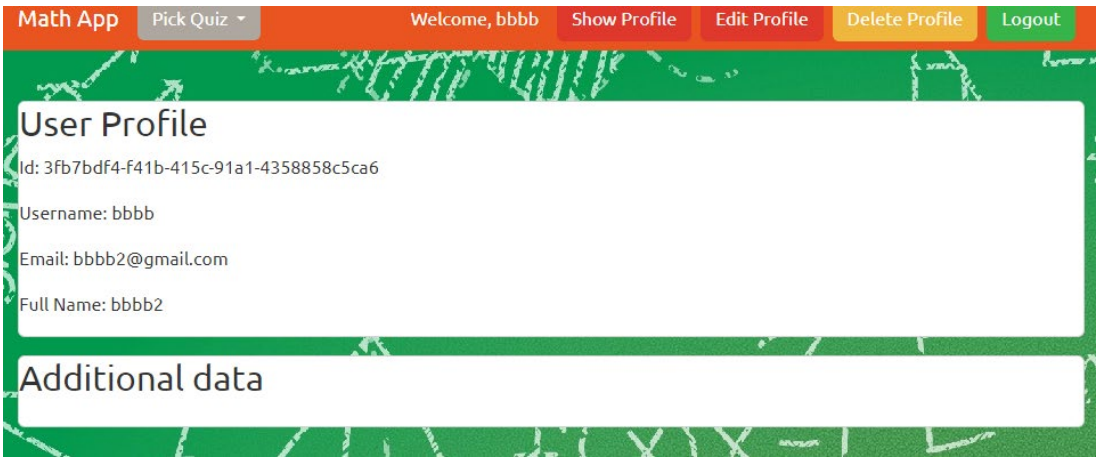


Рисунок 4.10 – Оновлений профіль

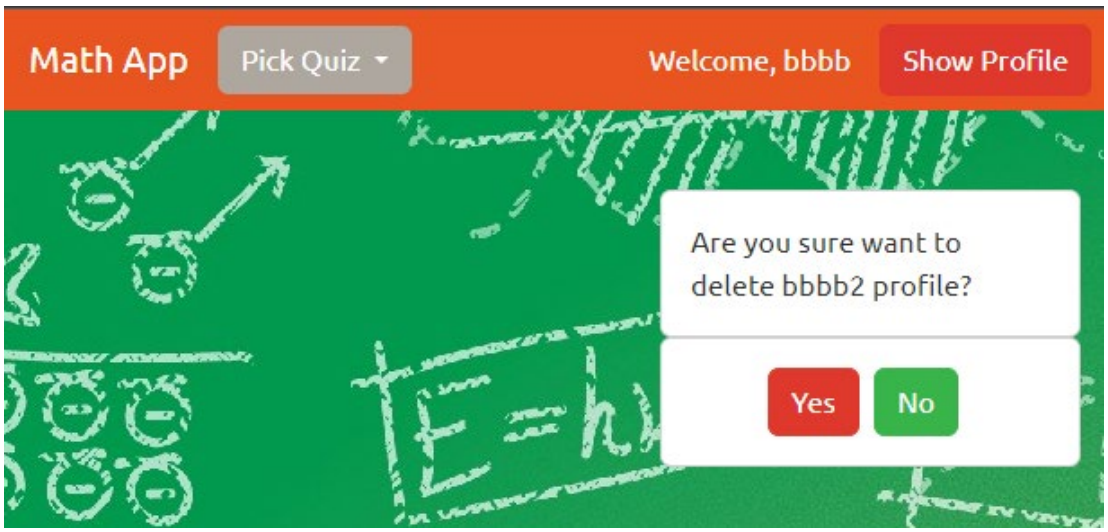


Рисунок 4.11 – Підтвердження видалення користувача

4.1.2 Для користувача:

– Гість може переглянути список доступних тестів
– Гість може зареєструватись

Всі інші функції може виконати тільки зареєстрований користувач.

4.1.3 Для адміністратора системи (якщо він передбачений):

Адміністратор не передбачений

4.1.4 Додаткові вимоги:

Додаткові вимоги не висуваються.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних.

На поточний момент вимагається мінімум 7 символів в паролі. Це повинні бути великі і маленькі латинські літери, цифри та символи.

Передбачена перевірка вводу емейлу. Це повинні бути латинські літери, собачка, доменне ім’я.

4.3 Умови експлуатації

Умови експлуатації згідно ДСанПІН 3.3.2.007-98.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Обслуговуючий персонал не передбачено.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

Мінімальна конфігурація

- тип процесору: Intel Pentium 4 або новіше з підтримкою SSE3.
- RAM: 1 GB для 32-bit або 2 GB для 64-bit.

Рекомендована конфігурація технічних засобів :

Не обмежена

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем

Windows

Windows 10, Windows Server 2016 або більш нові.

Процесор Intel Pentium 4 або більш новий, який підтримує SSE3 інструкції.

Mac

macOS High Sierra 10.13 або більш нові.

Linux

64-bit Ubuntu 18.04+, Debian 10+, openSUSE 15.2+, або Fedora Linux 32+

Процесор Intel Pentium 4 або більш новий, який підтримує SSE3 інструкції.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: json(для зчитування дерева дисциплін).

JSON (JavaScript Object Notation— це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, може бути прочитаним людиною. Формат дає змогу описувати об'єкти та інші структури даних. Цей формат використовується переважно для передавання структурованої інформації через мережу (завдяки процесу, що називають серіалізацією).

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі: json (для збереження дерева дисциплін).

JSON (JavaScript Object Notation— це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, може бути прочитаним людиною. Формат дає змогу описувати об'єкти та інші структури даних. Цей формат використовується переважно для передавання структурованої інформації через мережу (завдяки процесу, що називають серіалізацією).

4.5.3 Вимоги до мови розробки

Розробку виконати на мовах програмування C# та TypeScript.

4.5.4 Вимоги до середовища розробки

Розробку виконати за допомогою середовищ JetBrains Rider та Webstorm.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді текстових файлів .cs, .ts, .css, .html.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;

– технічне завдання;

– текст програми;

– програма та методика тестування;

– керівництво користувача;

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;

– схема бази даних;

– схема структурна класів програмного забезпечення;

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	10.03.2023	
2.	Розробка технічного завдання	24.03.2023	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	07.04.2023	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	21.04.2023	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	28.04.2023	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	05.05.2023	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	12.05.2023	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	19.05.2023	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	26.05.2023	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: Програмне забезпечення для підвищення ефективності роботи математичного тренажера.

КПІ.ІТ-9303.045440.02.81

Київ – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ASP	–	Active Server Pages
API	–	Application programming interface
BLL	–	Business Logic Layer
CORS	–	Cross-Origin Resource Sharing
CSR	–	Client Side Rendering
DAL	–	Data-Access Layer
DTO	–	Data transfer object
EF Core	–	Entity Framework Core
ER diagram	–	Entity–relationship diagram.
GDPR	–	General Data Protection Regulation
IDE	–	Integrated Development Environment
IT	–	Information technology
MVC	–	Model–view–controller
MVP	–	Model–view–presenter
MVVM	–	Model-View-ViewModel
ORM	–	Object-Relational Mapping
SDK	–	Software development kit
SOLID	–	a mnemonic acronym for five design principles
		Single-responsibility principle
		Open–closed principle:
		Liskov substitution principle:
		Interface segregation principle:
		Dependency inversion principle:
SSR	–	Server-Side Rendering
БД	–	База даних.
ОС	–	Операційна система.

ВСТУП

У поточних реаліях пандемії, війни, питання самоосвіти, розширення рівня компетенцій набувають актуальності. Питання самостійного отримання знань з будь-якої сфери передбачає набуття компетенцій та комплексного методичного підходу до їх опанування. Даний процес передбачає декомпозицію задачі вивчення дисципліни на значно менші елементи такі як теми, підтеми і тп. Варто зазначити, що в межах процесу самоосвіти досить важко виконати декомпозицію графу знань на деталізовані елементи, не володіючи ним, а ще складніше визначити які елементи даної декомпозиції були вивчені недосконало або були пропущені, що, в свою чергу, як наслідок приводить до неможливості просуватись далі та розширювати свої знання та вміння. Саме питання визначення недоліків у компетенціях або їх відсутність в межах декомпозиції дерева знань з дисципліни математика є ключовим та обумовлює проблему що буде розглянуто. Для допомоги здобувачеві існує безліч програмних засобів які орієнтовані на перевірку рівня володіння компетенціями з різних тематик. Майже всі з них встановлюють або загальну кількісну чи якісну оцінку володіння деревом знань або конкретним елементом його декомпозиції, проте в межах проведеного огляду предметної області не було знайдено жодного програмного засобу, який містить дерево знань та його декомпозицію, перелік компетенцій елементів декомпозиції та навчальних матеріалів та здатний, на основі результатів виконання контрольних заходів з будь-якої вершини дерева знань, встановити недоліки в опануванні інших елементів що є декомпозицією даного та надати структуровану відповідь відносно рекомендацій та недоліків.

Дані задачі є основними в межах розробки даної роботи.

Алгоритми, які застосовуються в рекомендаційних системах, допомагають користувачам зробити правильний вибір та вибрати найбільш цікавий матеріал для навчання. Рекомендаційні алгоритми побудовані з урахуванням індивідуальних смаків та уподобань кожного користувача.

Такі алгоритми значно спрощують життя як користувачам, так і власникам бізнесу. Наприклад, на основі рекомендацій можна набагато простіше знайти потрібний товар в інтернет-магазині, музичну композицію або фільм, з високою ймовірністю, що вам сподобається.

Рекомендаційні системи є важливим компонентом програмного забезпечення, яке допомагає підвищити ефективність роботи математичних тренажерів шляхом рекомендації наступних курсів на основі помилок користувачів у поточному курсі.

Актуальність теми: у поточних реаліях пандемії, війни, питання самоосвіти, розширення рівня компетенцій набувають актуальності. Питання самостійного отримання знань з будь-якої сфери передбачає набуття компетенцій та комплексного методичного підходу до їх опанування.

Мета роботи: покращення ефективності математичного тренажеру.

Завдання розробки:

Розробка функціоналу для можливості користувачу проходити тести з дисципліни та отримувати рекомендації щодо вибору тесту на основі результатів тесту.

Розробка функціоналу для можливості користувачу отримати рекомендовану літературу для опрацювання проблемних тем.

Реалізація процесу реєстрації нового користувача в додатку.

Розробка функціоналу для автентифікації та авторизації користувача.

Розробка функціоналу для надання користувачеві можливості змінювати свій профіль.

Розробка функціоналу для перегляду користувачем результатів пройдених тестів.

Розробка функціоналу для надання можливості користувачеві видаляти свій профіль з бази даних.

Практичне значення одержаних результатів: розроблений застосунок допоможе перевірити здобувачам освіти поточний рівень своїх знань з

математики, знайти прогалини в знаннях, опрацьовувати помилки та покращувати свої знання і навички, відповідно до результатів.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Математичні тренажери є програмними засобами, які призначені для покращення математичних навичок користувачів. За допомогою таких тренажерів можна вивчати та вдосконалювати навички розв'язування різних математичних завдань, наприклад, арифметичні дії, геометрію, алгебру, статистику та інші[1].

Однією з головних переваг використання математичних тренажерів є те, що користувачі можуть самостійно вибирати навчальний матеріал, рівень складності та тему завдань. Крім того, такі тренажери дозволяють користувачам виконувати завдання в зручний для них час та місце, що дозволяє ефективно використовувати свій час для навчання.

Використання математичних тренажерів є особливо корисним для дітей та молоді, які знаходяться на початкових етапах вивчення математики, а також для тих, хто прагне покращити свої знання та навички в цій галузі. Такі тренажери можуть бути використані як додатковий засіб для підготовки до іспитів та інших оціночних процедур.

Математичні тренажери можуть бути розроблені для різних платформ, таких як комп'ютери, мобільні телефони та планшети, що дозволяє їх використання в будь-якому місці з доступом до Інтернету. Крім того, такі тренажери можуть бути інтерактивними та мультимедійними, що дозволяє користувачам навчатися у цікавій та зручній формі[2].

Математичні тренажери вирішують питання покращення навичок за математики. Вони можуть це робити як послідовне вивчення тем, можуть надавати запитання з конкретної теми. Цим ці тренажери дозволяють перевірити свої знання, та отримати практичний досвід вирішення задач або відповідей на запитання.

1.2 Змістовний опис і аналіз предметної області

Застосунки для перевірки знань осіб що навчаються – математичні тренажери. Вони можуть бути використані в освітніх закладах, наукових дослідженнях або для підготовки до іспитів. Програмний забезпечення може містити різноманітні модулі для розв'язування математичних задач з різних галузей математики, таких як алгебра, геометрія, аналітична геометрія, тригонометрія та інші.

Математичні тренажери є популярними засобами для вивчення та вдосконалення математичних навичок у дітей та дорослих. Існує безліч алгоритмічних та технічних рішень, що можуть використовуватися для розробки таких тренажерів.

Для вирішення задачі покращення здобувачами компетенцій, а саме їх якісної характеристики, в межах дисципліни / тематики існує спеціалізоване ПЗ, задачею якого є підвищення рівня засвоєння матеріалу шляхом ряду методів таких як інтерактивне навчання, візуалізація, адаптація до індивідуальних потреб. Дана задача вирішується за рахунок набору алгоритмічних рішень, серед яких : використання системи випадкових чисел для генерації завдань, використанні адаптивного навчання[3].

Використання системи випадкових чисел для генерації завдань. У цьому випадку, тренажер буде генерувати випадкові числа та математичні вирази, які користувач повинен буде вирішити. Цей алгоритм досить простий у реалізації та дозволяє створювати безліч різних завдань з різними складнощами.

Інший алгоритм полягає у використанні адаптивного навчання. У цьому випадку, тренажер буде адаптуватися до рівня знань користувача та надавати завдання відповідно до його здібностей. Цей алгоритм вимагає більш складної реалізації та аналізу даних, але може бути набагато ефективнішим у досягненні поставленої мети - поліпшення математичних навичок користувача за рахунок того що він адаптується до кожної відповіді користувача[4].

Деякі тренажери рекомендують пройти наступний курс, деякі рекомендують курси зі схожих тем.

Буде запропоноване рішення яке аналізую помилки користувача, яке буде повідомляти про прогалини в знаннях та рекомендувати літературу для закриття цих прогалин з ціллю покращення знань та навичок з математики.

1.3 Аналіз існуючих технологій та успішних ІТ-проєктів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації програмного забезпечення для підвищення ефективності роботи математичного тренажера. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

В межах проведеного огляду предметної області та готових рішень, було виявлено, що користувацький інтерфейс повинен бути зручним у відповідності до UX критеріїв [5], а також виконувати ключову задачу – надавати користувачеві доступ до функціоналу застосунку. Під даний опис підпадають рішення з графічним віконним та веб інтерфейсами.

Рішення з графічним віконним інтерфейсом потребують відповідних умов для використання застосунку. Це повинна бути операційна система з відповідними технічними характеристиками.

Веб інтерфейс дає можливість отримати доступ до застосунку з будь-якого пристрою в якому є браузер. А це є майже будь-який комп’ютер, в тому числі мобільний телефон або планшети. Це дає додатку можливість охопити найбільшу аудиторію користувачів.

Крім того, веб інтерфейс дозволяє користувачам взаємодіяти з програмою, візуалізацію результатів та використання баз даних для зберігання інформації про користувачів та їх досягненнях. Для забезпечення швидкої та ефективної роботи програми можуть використовуватись різні технології, такі

як оптимізація коду, кешування даних, паралелізація обчислень та використання спеціалізованих фреймворків та бібліотек.

Крім того, можуть використовуватись інтерактивні методи для покращення взаємодії з користувачем, такі як використання анімації, звукових ефектів, різноманітних візуальних елементів тощо. Важливо також забезпечити безпеку та захист від зловмисних дій, такі як введення некоректних даних або спроби злому системи. Для цього можуть використовуватись різні методи захисту, такі як перевірка введених даних, шифрування, валідація користувачів та використання.

Тому було прийняте рішення працювати саме з веб інтерфейсом.

В межах створення програмного продукту необхідно обрати архітектурне рішення, що дозволить виконати задачі додатку.

Розглянемо Monolith та Microservices architecture.

Monolith (монолітна) архітектура - це традиційна модель програмного забезпечення, яка є єдиним модулем, що працює автономно і незалежно від інших додатків. Вона полягає у розробці додатка як єдиного, нерозділеного блоку коду, який включає в себе функціонал, базу даних та інші компоненти. Всі функції додатка обслуговуються в межах цього блоку. І хоча це може бути простою архітектурою, монолітні додатки можуть стати важкими для розгортання та масштабування з ростом розміру та складності проекту[6].

Microservices (мікросервісна) архітектура є методом організації архітектури, заснований на ряді незалежно розгортаються служб додатків. Вона полягає в розробці додатка як набору окремих мікросервісів, кожен з яких відповідає за певну функцію. Кожен мікросервіс може бути розгорнутий та масштабований незалежно від інших, що робить процес розгортання та розвитку більш гнучким та ефективним. Однак, для створення та підтримки мікросервісів необхідна додаткова робота з конфігурацією, моніторингом та керуванням[6].

Отже, відмінність між монолітною та мікросервісною архітектурою полягає в тому, що монолітна архітектура зберігає всі функції додатка в

одному блоку коду, тоді як мікросервісна архітектура складається з набору невеликих сервісів, кожен з яких відповідає за свою функцію[6].

Була обрана монолітна архітектура, оскільки реалізувати проект вийде тільки з нею в обмежений час, в поточній ситуації та з поточними можливостями, знаннями та навичками.

Розберемо архітектурні патерни.

MVC, MVP та MVVM - це патерни архітектури програмного забезпечення, які використовуються для розробки програм з графічним інтерфейсом.

MVC (Model-View-Controller- це патерн, в якому програма розділена на три компоненти: Модель (Model), Представлення (View) та Контролер (Controller). Модель відповідає за дані та бізнес-логіку, Представлення - за відображення даних користувачу та Контролер - за управління потоком даних між Моделлю та Представленням)[7].

MVP (Model-View-Presenter) - це патерн, який розділяє програму на три компоненти: Модель (Model), Представлення (View) та Презентер (Presenter). У MVP Модель та Представлення не взаємодіють напряму, а через Презентер, який відповідає за обробку даних та взаємодію з Моделлю та Представленням[8].

MVVM (Model-View-ViewModel) - це патерн, в якому програма розділена на три компоненти: Модель (Model), Представлення (View) та ViewModel. ViewModel відповідає за відображення даних з Моделі на Представленні. ViewModel не взаємодіє напряму з Представленням, а використовує механізм прив'язки даних (Data Binding) [9].

Основна відмінність між цими патернами полягає в способі взаємодії компонентів та управління потоком даних в програмі. MVC використовує Контролер для управління потоком даних, MVP використовує Презентер, а MVVM - механізм прив'язки даних. Крім того, MVP та MVVM більш схильні до тестування, оскільки є можливість тестувати окремі компоненти без

взаємодії з іншими. MVVM в Http забезпечити неможливо, так само як і MVP. Тому було обрано архітектурний патерн MVC.

Відповідно для забезпечення працездатності шару роботи з користувачем можна використати два ключових підходи SSR та CSR

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

Технічні рішення можуть варіюватися в залежності від конкретної реалізації тренажеру. Наприклад, для веб-додатків може бути використано Angular або React для реалізації користувацького інтерфейсу та взаємодії з сервером.

В межах обраної архітектури та CSR підходу наявні додаткові технічні рішення-фреймворки Angular, React, Vue та інші[10].

Angular та React - це два з найбільш популярних фреймворків для розробки веб-додатків. Обидва фреймворки мають свої переваги та недоліки, але зазвичай вибір залежить від конкретних потреб проекту та особистих вподобань розробника.

Основна відмінність між ними полягає в підході до розробки. Angular-це фреймворк, який надає можливості із коробки, такі як шаблони, роутинг, форми, валідація та багато іншого[11]. React же - це бібліотека, яка пропонує більш гнучкий та модульний підхід до розробки, дозволяючи розробникам використовувати інші бібліотеки та рішення згідно їх потреб[12].

Інші відмінності між ними:

Angular використовує TypeScript як основну мову, що надає більшу статичну перевірку та допомагає під час розробки, тоді як React дозволяє використовувати як JavaScript, так і TypeScript.

Angular має вбудований механізм залежностей та ін'єкцію залежностей, тоді як в React ці функції реалізовуються через сторонні бібліотеки.

Angular має більш розвинену систему тестування, тоді як React потребує використання сторонніх бібліотек для тестування.

Висновок з цього такий, що обидва фреймворки мають свої переваги та недоліки, та вибір між ними залежить від конкретних потреб проекту та особистих вподобань розробника.

Таким чином було обрано Angular, оскільки він є атомарним та незалежним від зовнішніх залежностей фреймворком, і одразу дає нам всі необхідні інструменти для розробки у відповідності до поставлених задач.

Для реалізації серверної частини застосунку необхідно вибрати технологію. Виходячи з порівняльної характеристики

Щодо бекенду також необхідно обрати мову програмування та IDE. Будемо обирати між Java та C#. Також розглянемо загальну ситуацію щодо мов програмувань на поточний момент.

Java та C# є двома з найбільш поширених мов програмування в світі. Ці мови мають багато спільного, тому їх часто порівнюють між собою. Ось деякі загальні риси, які можна порівняти у Java та C#:

Схожий синтаксис: C# був розроблений після Java, тому вони мають досить схожий синтаксис. Це означає, що більшість програмістів, які вміють писати на одній з цих мов, можуть легко перейти на іншу.

Переносимість коду: якщо використовувати тільки стандартні бібліотеки, код, написаний на Java або C#, можна легко перенести з однієї платформи на іншу.

Висока продуктивність: обидві мови мають високу продуктивність, що дозволяє розробникам створювати швидкі та ефективні програми.

Однак, є і деякі ключові відмінності між Java та C#, які можна врахувати:

Платформи: Java запускається на віртуальній машині Java (JVM), тоді як C# запускається на платформі .NET. Це може мати вплив на продуктивність та переносимість коду.

Використання пам'яті: Java має більш активну систему збору сміття, що може займати більше часу на виконання програми. C# використовує більш ефективну систему збору сміття, що може дозволити створювати швидкіші програми.

Розширення: Java має більш широкий спектр розширень, що дозволяє розробникам розширювати мову та використовувати її для більш різноманітних проектів.

Спільнота розробників: обидві мови мають досить велику спільноту розробників.

За даними сайту dou.ua, найпопулярнішою мовою серед українських розробників залишається JavaScript (18,8%). На другому місці C#(рисунок 1.1), у неї другий рік поспіль позитивна динаміка. Можна припустити, що завдяки активному зростанню геймдев-індустрії. Далі йде Java, частка якої з 2017 року стабільно зменшується.

Загалом популярність більшості мов цього року зростала. Негативна динаміка, окрім Java і Python, лише у Ruby, C, Scala, Clojure і C++. Остання, до речі, продовжує стрімко втрачати популярність. Не в останню чергу через зниження частки серед новачків[13].

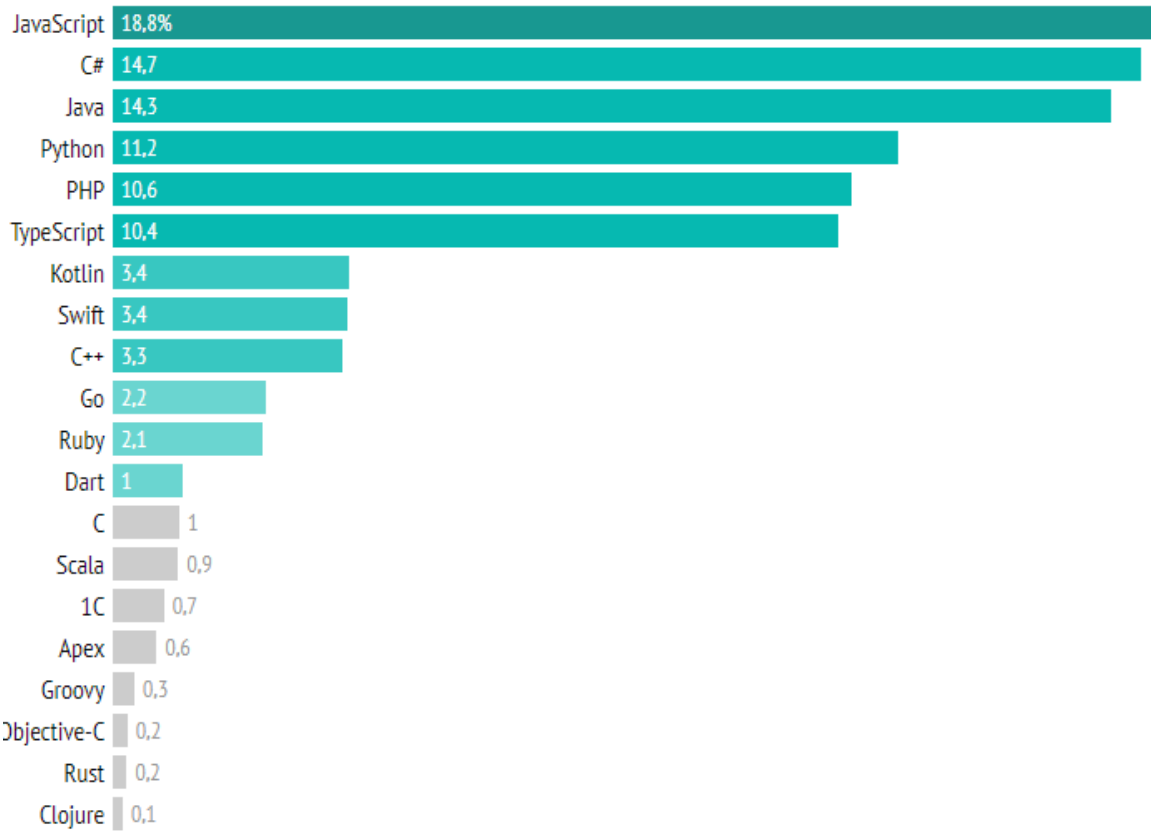


Рисунок 1.1 – Рейтинг мов програмування 2022[13].

Серверна частина передбачає використання технології .NET та мови C#.

Для реалізації написання коду, компіляції, запуску коду існує перелік середовищ розробки. Розглянемо три з них. Це Rider, Visual Studio та Visual Studio Code.

Rider, Visual Studio та Visual Studio Code - це IDE, які підтримують розробку різноманітних мов програмування та платформ. Кожен з цих редакторів має свої переваги та недоліки, які можуть вплинути на вибір IDE для конкретного проекту.

Rider - це IDE, розроблене компанією JetBrains, яке підтримує розробку на різних мовах програмування, включаючи C#, Java та Kotlin. Його основні переваги - це швидкість та продуктивність, підтримка багатьох фреймворків, таких як .NET та ASP.NET, і вбудований дебагер[14].

Visual Studio - це редактор, розроблений компанією Microsoft для розробки на мові програмування C# та інших мовах програмування, які підтримують платформу .NET. Visual Studio є потужним та повністю функціональним редактором, заснованим на компонентах Windows, зі значними можливостями розробки, такими як дебагер, тестування, підтримка контролю версій та багато іншого[15].

Visual Studio Code - це безкоштовний, легкий та потужний редактор коду, який підтримує розробку на різних мовах програмування, включаючи C#, Java, Python та багато інших. Він підтримує плагіни та має зручні можливості для роботи з Git, що робить його ідеальним вибором для розробки маленьких та середніх проектів[16].

Вибір IDE залежить від специфіки проекту та особистих вподобань. Rider та Visual Studio мають більше можливостей для розробки на платформі .NET, тоді як Visual Studio Code є легким та потужним редактором, який можна використовувати для розробки на багатьох мовах програмування.

Було обрано Rider як найбільш функціональну та швидку IDE.

Для зберігання даних користувачів та їх прогресу у вивченні математики. Було прийнято рішення обрати Microsoft SQL Server у зв’язку з тим що вони мають спільного розробника, і як наслідок, ця СКБД тісно пов’язана з мовою програмування C#.

Доступ до даних можна реалізувати за допомогою Entity Framework Core та Dapper. Це дві різні ORM (Object-Relational Mapping) бібліотеки для .NET, які допомагають працювати з базами даних у зручний спосіб.

Entity Framework Core(EF Core) - це легка, розширювана кросплатформна версія популярної технології доступу до даних Entity Framework із відкритим кодом. Це високорівнева бібліотека, яка надає можливості автоматичного створення моделей баз даних на основі класів C# та використовує LINQ для запитів до баз даних. EF Core підтримує багато різних баз даних, включаючи SQL Server, MySQL та PostgreSQL[17].

Dapper – це простий об’єктний маппер. Це бібліотека, вона більш низького рівня, яка пропонує простий та швидкий спосіб взаємодії з базами даних, де розробник вручну пише запити до бази даних. Dapper дозволяє розробнику виконувати свої запити та вручну мапити дані з бази даних в об'єкти C#. Dapper не підтримує багато різних баз даних, але працює з більшістю популярних баз даних, таких як SQL Server, MySQL та PostgreSQL[18].

EF Core підходить для проектів, які потребують високорівневої роботи з базами даних та автоматичного створення моделей на основі класів C#, тоді як Dapper підходить для проектів, які потребують більшого контролю над запитами до баз даних та швидкісного виконання запитів.

Dapper швидше, але вимагає від розробника самостійного написання SQL команд. EF Core повільніше, але має більше можливостей і є більш простим в користуванні.

Було обрано EF Core як більш простий та зручний інструмент для розробки.

Згідно SOLID принципу Dependency Inversion шари не повинні знати один про інший. Тому доведеться користувацьке представлення даних поєднувати з тим що є в базі даних. Як наслідок, доведеться приводити одні сутності і класи до інших сутностей і класів. Для вирішення поточної задачі в рамках обраної архітектури, платформи і технології існує ряд допоміжних бібліотек. Серед яких можна виділити AutoMapper, EmitMapper, Mapster.

AutoMapper – це бібліотека, створена для вирішення оманливо складної проблеми – позбавлення від коду, який ставив у відповідність один об’єкт з іншим. Цей тип коду досить нудний для написання, саме це і реалізує AutoMaper. Він дозволяє зберегти багато часу на написання рутинних завдань, таких як копіювання властивостей з одного об'єкту в інший. AutoMapper автоматично відображає властивості з одного об'єкту на інший, використовуючи конвенції іменування[19].

EmitMapper – це потужний настроюваний інструмент для відображення об’єктів один з одним. Він дозволяє мапити дані між об'єктами різних типів. EmitMapper генерує код на льоту, що дозволяє йому працювати швидко та ефективно[20].

Mapster – це перетворювач, який дозволяє відображати дані між об'єктами різних типів. Mapster має простий та інтуїтивно зрозумілий інтерфейс та дозволяє використовувати лямбда-вирази для визначення відображення. Mapster розроблено таким чином, щоб ефективно використовувати швидкість і пам’ять. Ви можете отримати 4-кратне підвищення продуктивності, використовуючи лише 1/3 пам’яті. І ви можете збільшити продуктивність до 12 разів[21].

Було вирішено використовувати AutoMapper у зв’зку з наявністю якісної документації та великого ком’юніті користувачів даного перетворювача.

Існує задача визначення якості коду. В рамках цієї задачі для її вирішення існує інструментарій який здійснює динамічний аналіз коду та статичний аналіз коду.

Динамічний аналізатор коду і статичний аналізатор коду – це дві різні техніки, які використовуються для аналізу програмного коду з метою виявлення помилок, недоліків та потенційних проблем. Ось порівняння понять динамічного і статичного аналізаторів коду:

Динамічний аналізатор коду.

Плюси.

Здатний виявляти помилки та проблеми, які виникають під час виконання програми.

Надає інформацію про поведінку програми в реальному часі, дозволяючи виявляти проблеми, які можуть залежати від конкретних умов виконання.

Дозволяє виконувати тестування коду шляхом введення тестових даних і перевірки результатів.

Мінуси.

Вимагає виконання програми або її частини, що може бути витратним за часом та ресурсами.

Не здатний знайти помилки, які не виникають під час конкретного виконання програми.

Обмежений обсягом коду, який можна перевірити під час виконання.

Статичний аналізатор коду:

Плюси.

Здатний виявити широкий спектр помилок та проблем, включаючи синтаксичні помилки, потенційні помилки виконання, проблеми з безпекою та інші.

Може аналізувати весь код без виконання програми, що дозволяє знайти помилки, навіть якщо вони не виникають під час виконання.

Забезпечує раннє виявлення проблем, дозволяючи їх виправити на ранніх етапах розробки.

Мінуси.

Може виявляти помилки лише на основі аналізу коду, тому не здатний знайти проблеми, які залежать від конкретних умов виконання.

Може генерувати багато помилкових сигналів або пропустити деякі проблеми, залежно від якості аналізатора та його конфігурації.

Вимагає використання додаткових інструментів або плагінів для інтеграції з редактором коду або процесом розробки.

Було обрано статичний аналізатор коду, оскільки динамічний аналізатору коду оскільки він не може проаналізувати весь обсяг коду.

Для реалізації статичного аналізу розглянемо наступні фреймворки: SonarQube, StyleCop, Roslyn Analyzers.

SonarQube є потужним статичним аналізатором коду, який підтримує мову C# та надає широкий спектр функцій для аналізу якості коду, виявлення вразливостей, метрик продуктивності, покриття тестами та багато іншого. Він має розширену підтримку іншомовних плагінів, які дозволяють використовувати його для проектів у різних мовах програмування[22].

ReSharper є популярним інструментом для статичного аналізу коду в середовищі розробки Visual Studio. Він надає багато функцій, таких як автоматичне виявлення помилок, пропозиції з вдосконалення коду, переформатування та оптимізація. ReSharper підтримує широкий спектр мов програмування, включаючи C#[23].

StyleCop є статичним аналізатором коду, який спрямований на дотримання стандартів оформлення коду. Він перевіряє код на відповідність правилам форматування, іменування, розміщення директив та багато іншого. StyleCop допомагає забезпечити єдність стилю коду в проекті[24].

Roslyn Analyzers використовує вбудований аналізатор компілятора Roslyn, що надає можливість створювати власні правила аналізу коду для виявлення помилок та неправильного використання API. Це дозволяє вам налаштовувати аналізатор під ваші потреби та вимоги проекту[25].

Було обрано SonarQube, завдяки своїй розширеній функціональності.

1.3.3 Аналіз відомих програмних продуктів

Розглянемо декілька математичних тренажерів:

Khan Academy. Цей сайт пропонує безкоштовний доступ до понад 10 000 відеоуроків та практичних завдань з різних тем математики. Плюсами є широкий вибір тем, різнорівневі завдання та можливість відстежувати свій прогрес. Мінусом може бути те, що деякі теми можуть бути недостатньо детально роз'яснені[26].

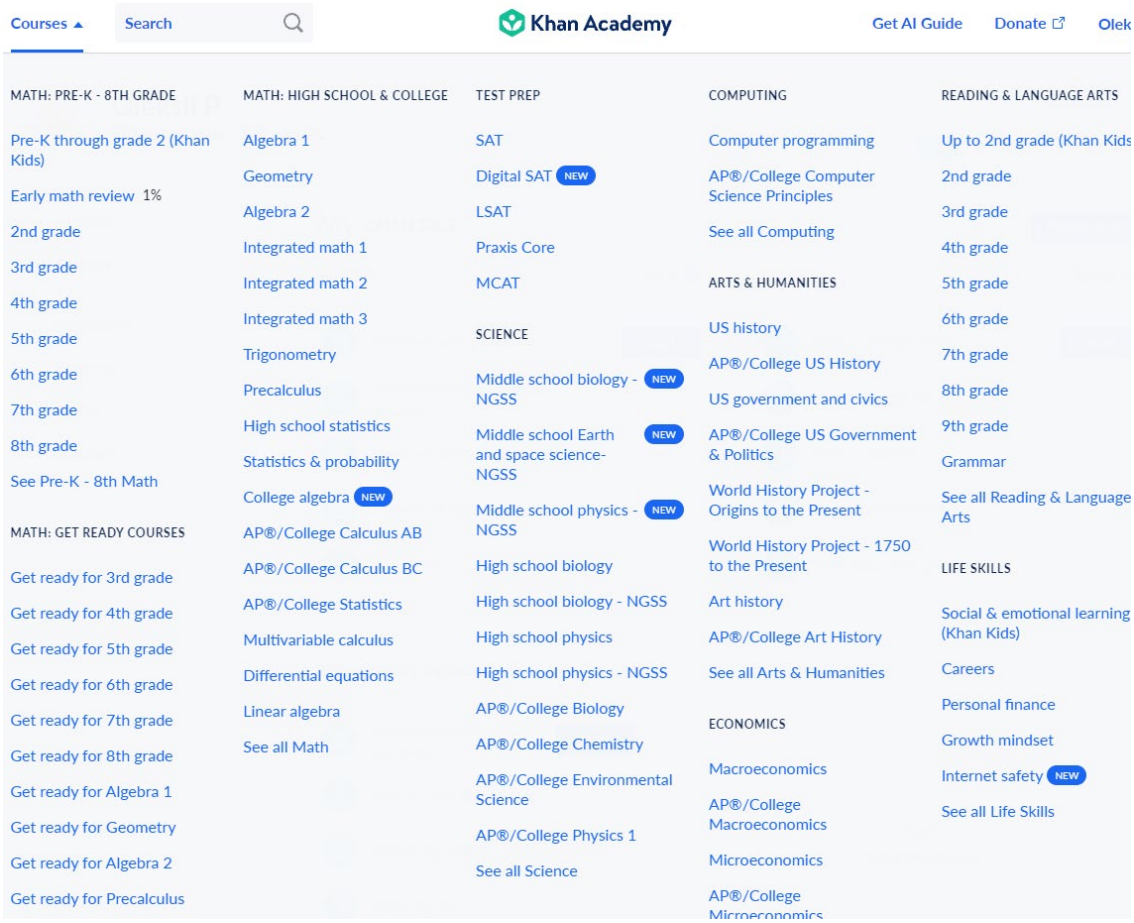


Рисунок 1.2 – Khan Academy

Coursera — це глобальна платформа онлайн-навчання, яка пропонує будь-кому будь-де доступ до онлайн-курсів і ступенів від університетів і компаній світового класу[27].

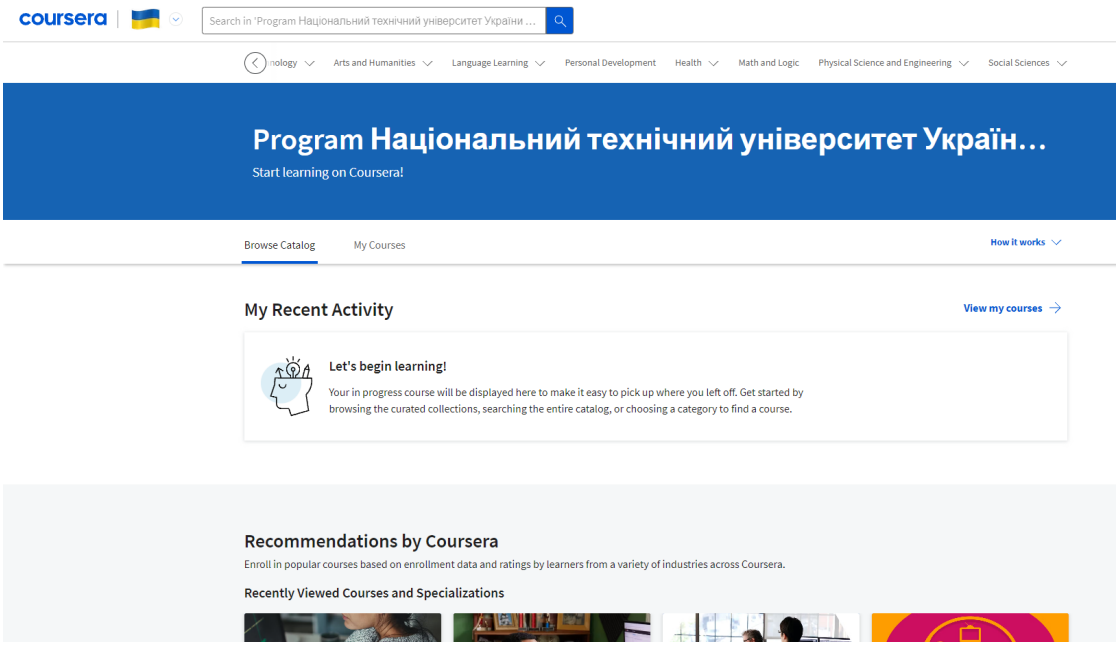


Рисунок 1.3 – Coursera

Coursera співпрацює з понад 275 провідними університетами та компаніями, щоб надати людям гнучке, доступне онлайн-навчання, відповідне роботі. та організації по всьому світу. Ми пропонуємо низку можливостей для навчання — від практичних проектів і курсів до сертифікатів, готових до роботи, і програм отримання ступеню. В тому числі, на цьому сайті є курси з математики.

Brilliant. Цей сайт пропонує курси з різних тем математики для різних рівнів знань. Плюсами є різноманітність тем та рівнів, відеоуроки та інтерактивні завдання. Мінусом може бути висока вартість преміум-версії та деяка складність завдань для початківців[28].

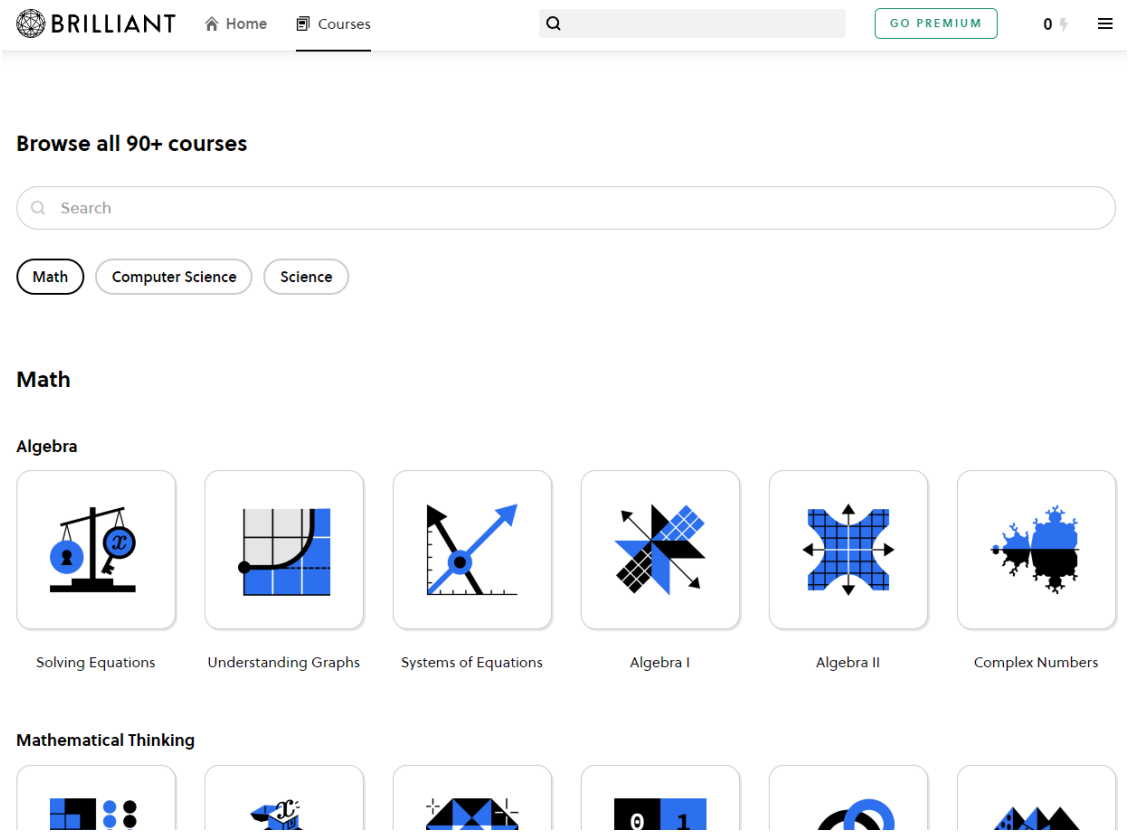


Рисунок 1.4 – Brilliant

Таблиця 1.1 – Порівняння дипломного проєкту з аналогами.

Функціонал	MathApp	Khan Academy	Coursera	Brilliant
Обрати тему/тест	+	+	+	+
Пройти тест	+	+	+	+
Широкий вибір тем	-	+	+	+
Побачити результат	+	+	+	+
Рекомендувати користувачу наступний тест для проходження	Новий тест пропонується на основі помилок в попередньому.	Після першого юніта тобі пропонують другий юніт	Рекомендує популярні курси і курси які є схожими на ваші інтереси(схожі на ті курси які ви вже проходили)	Після першого юніта пропонуєтся другий юніт

1.4 Аналіз вимог до програмного забезпечення

Головною функцією програмного забезпечення є надання можливості здобувачам освіти перевірити та покращити свої знання з математики, більше функцій можна побачити на діаграмі варіантів використання. Вона знаходиться в графічному матеріалі. Креслення 1 - структурна схема варіантів використання.

В системі повинні бути реалізовані наступні функції:

– можливість проходження тесту з дисципліни, і, відповідно, покращення навичок за рахунок рекомендацій наступного тесту на основі результатів пройденого тесту;

– можливість отримати список рекомендованої літератури для відпрацювання проблемних тем;

– процес реєстрації нового користувача в застосунку;

– процес автентифікації та авторизації користувача;

– можливість користувача змінювати свій профіль;

– можливість користувача переглядати результат пройденого тесту;

– можливість користувача видаляти свій профіль з бази даних;

У даному застосунку передбачені варіанти використання, які описані в таблицях 1.2 - 1.12

Таблиця 1.2 – Варіант використання UC01

Use case name	Реєстрація користувача
Use case ID	UC-01
Goals	Реєстрація нового користувача в системі
Actors	Гість (незареєстрований користувач)
Trigger	Користувач бажає зареєструватися
Pre-conditions	-

Продовження таблиці 1.2

Flow of Events	Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: username, fullname, пошта користувача, пароль в системі, та його повтор для підтвердження. Після заповнення даних користувача натискає кнопку реєстрації. Після цього з'являється повідомлення про успішну реєстрацію, і користувач перенаправляється на сторінку входу.
Extension	В випадку введення не коректних даних, кнопка реєстрації стає неактивною. Якщо якесь конкретне поле введено некоректно, то воно підсвічується червоним надписом.
Post-Condition	Створення сторінки користувача, перехід на сторінку входу.

Таблиця 1.3 – Варіант використання UC02

Use case name	Перегляд існуючих тем тестів
Use case ID	UC-02
Goals	Успішний перегляд доступних тем для тестів.
Actors	Гість (незареєстрований користувач)
Trigger	Користувач бажає зареєструватися
Pre-conditions	-
Flow of Events	Користувач переходить домашню сторінку застосунку. Нажимає на кнопку вибору теми. Після цього з'являється с писок тем, і користувач хз переглядає.
Extension	В випадку натискання помилкової кнопки користувач не бачить список тем.
Post-Condition	Користувач успішно переглянув список тем для тестів.

Таблиця 1.4 – Варіант використання UC03

Use case name	Авторизація
Use case ID	UC-03
Goals	Авторизація користувача в системі
Actors	Гість (незареєстрований користувач)
Trigger	Користувач бажає авторизуватися

Продовження таблиці 1.4

Pre-conditions	-
Flow of Events	Користувач переходить на сторінку авторизації. В поля для авторизації вводяться відповідні дані: username та password, Після заповнення даних користувача натискає кнопку авторизації. Після цього з'являється повідомлення про успішну авторизацію, і користувач перенаправляється на домашню сторінку.
Extension	В випадку введення не коректних даних, буде виведено повідомлення про помилку.
Post-Condition	Успішна авторизація. Користувач потрапляє на домашню сторінку.

Таблиця 1.5 – Варіант використання UC04

Use case name	Користувач може вийти з системи.
Use case ID	UC-04
Goals	Успішний вихід користувача з системи.
Actors	Зареєстрований користувач.
Trigger	Користувач бажає вийти з системи.
Pre-conditions	Користувач авторизований.
Flow of Events	Користувач переходить на сторінку виходу з системи. Після цього з'являється повідомлення про успішний вихід з системи і користувач перенаправляється на сторінку входу.
Extension	В випадку якщо користувач натиснув іншу кнопку, користувач залишається авторизованим.
Post-Condition	Вихід з системи, перехід на сторінку входу.

Таблиця 1.6 – Варіант використання UC05

Use case name	Переглянути профіль користувача
Use case ID	UC-05
Goals	Перегляд профілю користувача

Продовження таблиці 1.6

Actors	Зареєстрований користувач
Trigger	Користувач бажає переглянути свій профіль
Pre-conditions	Користувач авторизований
Flow of Events	Користувач переходить на сторінку перегляду свого профілю. Користувач переглядає свій профіль.
Extension	В випадку натискання некоректної кнопки користувач не бачить свій профіль.
Post-Condition	Користувач успішно переглянув свій профіль і залишився на сторінці перегляду свого профіля.

Таблиця 1.7 – Варіант використання UC06

Use case name	Редагування профіля користувача
Use case ID	UC-06
Goals	Змінити профіль користувача.
Actors	Зареєстрований користувач
Trigger	Користувач бажає змінити свій профіль
Pre-conditions	Користувач авторизований
Flow of Events	Користувач переходить на сторінку зміни профіля. В поля для редагування вводяться відповідні дані: пошта користувача, fullname. Після заповнення даних користувача натискає кнопку збереження даних. Після цього користувач перенаправляється на сторінку перегляду профілю.
Extension	В випадку не натискання кнопки збереження даних дані не будуть збережені.
Post-Condition	Після зміни профілю користувач перенаправляється на сторінку перегляду профілю.

Таблиця 1.8 – Варіант використання UC07

Use case name	Видалення профіля користувача
Use case ID	UC-07
Goals	Видалення користувача з системи
Actors	Зареєстрований користувач
Trigger	Користувач бажає видалити свій профіль

Продовження таблиці 1.8

Pre-conditions	Користувач авторизований
Flow of Events	Користувач переходить на сторінку видалення свого профілю. Користувач підтверджує видалення свого профілю. Користувач перенаправляється на сторінку входу.
Extension	В випадку не підтвердження видалення профілю профіль користувача не буде видалено.
Post-Condition	Користувача перенаправлено на сторінку входу.

Таблиця 1.9 – Варіант використання UC08

Use case name	Обирання тесту для проходження
Use case ID	UC-08
Goals	Обрати тест для проходження
Actors	Зареєстрований користувач
Trigger	Користувач бажає обрати тест для проходження
Pre-conditions	Користувач авторизований
Flow of Events	Користувач переходить на домашню сторінку. Користувач нажимає на кнопку вибору теми тесту і обирає потрібну тему зі списку. Після цього завантажуються запитання з обраної теми.
Extension	У випадку не натискання користувача на кнопку вибору теми, користувач не вибере тему.
Post-Condition	Користувач отримав перелік запитань з обраної теми.

Таблиця 1.10 – Варіант використання UC09

Use case name	Проходження тесту
Use case ID	UC-09
Goals	Користувач успішно відповів на запитання.
Actors	Зареєстрований користувач
Trigger	Користувач бажає пройти тест
Pre-conditions	Користувач авторизований, користувач обрав тему тесту.
Flow of Events	Користувач відповідає на запитання обираючи варіанти відповідей з переліку.
Extension	У випадку не натискання на кнопки варіантів відповідей користувач не відповість на відповідні запитання.
Post-Condition	Користувач відповів на запитання тесту.

Таблиця 1.11 – Варіант використання UC10

Use case name	Отримання результатів тесту
Use case ID	UC-10
Goals	Отримання результатів тесту
Actors	Зареєстрований користувач
Trigger	Користувач бажає отримати результат тесту
Pre-conditions	Користувач відповів на запитання
Flow of Events	Користувач нажимає на кнопку показу результатів і бачить результати тесту.
Extension	У випадку не натискання на кнопку показу результатів користувач не побачить результатів тесту.
Post-Condition	Користувач бачить результати тесту

Таблиця 1.12 – Варіант використання UC11

Use case name	Отримання рекомендації
Use case ID	UC-1
Goals	Реєстрація нового користувача в системі
Actors	Зареєстрований користувач
Trigger	Користувач бажає побачити рекомендацію
Pre-conditions	Користувач бачить результати тесту
Flow of Events	Користувач бачить теми в яких є прогалини в знаннях і літературу для покращення навичок з відповідних тем.
Extension	У випадку не проходження тесту користувач не отримав рекомендацію.
Post-Condition	Користувач отримав рекомендацію

1.4.1 Розроблення функціональних вимог

Модель вимог, створена на основі детального аналізу варіантів використання продукту, описана у таблиці 1.14.

Таблиця 1.14 – Модель вимог

Вимога	Код	Пріоритет	Ризик
1. Реєстрація користувача	REQ01	Високий	Середній
1.1 Валідація полів	REQ01.1	Високий	Низький
2. Авторизація користувача	REQ03	Високий	Середній
3. Перегляд тем тестів	REQ02	Низький	Низький
4. Вихід з системи	REQ04	Середній	Низький

Продовження таблиці 1.14

5. Переглянути профіль користувача	REQ05	Низький	Низький
6. Редагування профілю користувача	REQ06	Низький	Низький
7. Видалення профілю користувача	REQ07	Середній	Середній
8. Обирання тесту для проходження	REQ08	Середній	Середній
9. Проходження тесту	REQ09	Високий	Середній
9.1 Завантаження запитань	REQ09.1	Високий	Високий
10. Отримання результату	REQ10	Високий	Середній
10.1 Отримання балів	REQ10	Низький	Низький
10.2 Отримання рекомендації	REQ11	Високий	Високий

Функціональні вимоги додатку описанов таблицях 1.13 – 1.23:

Таблиця 1.13 – Варіант використання REQ01

Номер	REQ01
Назва	Реєстрація користувача
Опис	Гість реєструється в системі, стає користувачем

Таблиця 1.14 – Варіант використання REQ02

Номер	REQ02
Назва	Перегляд існуючих тем тестів
Опис	Гість може переглянути перелік існуючих тем тестів

Таблиця 1.15 – Варіант використання REQ03

Номер	REQ03
Назва	Login
Опис	Користувач може авторизуватись.

Таблиця 1.16 – Варіант використання REQ04

Номер	REQ04
Назва	Logout
Опис	Користувач може вийти зі свого профілю

Таблиця 1.17 – Варіант використання REQ05

Номер	REQ05

Продовження таблиці 1.17

Назва	Переглянути профіль користувача
Опис	Користувач може переглянути свій профіль

Таблиця 1.18 – Варіант використання REQ06

Номер	REQ06
Назва	Редагування профіля користувача
Опис	Користувач може редагувати свій профіль

Таблиця 1.19 – Варіант використання REQ07

Номер	REQ07
Назва	Видалення профіля користувача
Опис	Користувач може видалити свій профіль

Таблиця 1.20 – Варіант використання REQ08

Номер	REQ08
Назва	Обирання тесту для проходження
Опис	Користувач обирає тест.

Таблиця 1.21 – Варіант використання REQ09

Номер	REQ09
Назва	Проходження тесту
Опис	Користувач проходить тест.

Таблиця 1.22 – Варіант використання REQ10

Номер	REQ10
Назва	Отримання результатів тесту
Опис	Користувач бачить результати пройденого тесту

Таблиця 1.23 – Варіант використання REQ11

Номер	REQ11
Назва	Отримання результатів тесту
Опис	Користувач отримує рекомендацію щодо наступного тесту для проходження

Матрицю залежності наведено на рисунку 1.3.

	REQ01	REQ02	REQ03	REQ04	REQ05	REQ06	REQ07	REQ08	REQ09	REQ10	REQ11
UC01	■										
UC02		■									
UC03			■								
UC04				■							
UC05					■						
UC06						■					
UC07							■				
UC08								■			
UC09									■		
UC10										■	
UC11											■

Рисунок 1.3 – Матриця залежності варіантів використання та вимог застосунку.

1.4.2 Розроблення нефункціональних вимог

Нефункціональні вимоги.

Інтерфейс користувача повинен бути інтуїтивно зрозумілим та зручним для використання.

Платформа повинна бути надійною та безпечною, щоб уникнути витоку особистої інформації користувачів..

Швидкість роботи платформи повинна бути достатньою(завантаження запитань тесту повинно займати до 1 секунди) для зручного використання користувачами.

1.5 Постановка задачі

Мета створення даної роботи – створення програмного забезпечення для підвищення ефективності математичного тренажера.

Для того щоб досягнути даної мети, потрібно виконати ряд наступних поставлених задач.

Розробка функціоналу для дозволу користувачеві проходити тести з дисципліни та отримувати рекомендації на основі результатів тесту для поліпшення навичок.

Розробка функціоналу для отримання користувачем списку рекомендованої літератури для відпрацювання проблемних тем.

Реалізація процесу реєстрації нового користувача в додатку.

Розробка функціоналу для автентифікації та авторизації користувача.

Розробка функціоналу для надання користувачеві можливості змінювати свій профіль.

Розробка функціоналу для перегляду користувачем результатів пройдених тестів.

Розробка функціоналу для надання можливості користувачеві видаляти свій профіль з бази даних.

Висновки до розділу

В межах даного розділу було розглянуто та проведено аналіз предметної області результатом якого було виявлено поточну ситуацію щодо даної предметної області.

Було проаналізовано відомі алгоритмічні та технічні рішення. Внаслідок чого було вирішено яким чином буде реалізовано програмне забезпечення поточного проєкту.

Було проаналізовано допоміжні програмні засобі та засоби розробки. Як наслідок, було обрано технології та інструменти які будуть використані в поточній розробці.

Було проведено аналіз та дослідження відомих програмних продуктів. Як наслідок є розуміння чим програмне забезпечення поточного проєкту буде відрізнятись від відомих існуючих рішень.

Було розроблено функціональні вимоги, як наслідок отримано інформацію щодо функцій які повинні бути реалізовані в проєкті.

Було розроблено нефункціональні вимоги. Як результат було отримано інформацію щодо якісних характеристик додатку що розроблюється.

І, на кінець було поставлено чіткі задачі щодо роботи розроблюваного застосунку.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

У додатку використано багатошарову архітектуру. Це дозволяє кожен модуль ізолювати, щоб їх можна було розгортати та підтримувати окремо. Також такий підхід збільшує відмовостійкість системи.

Для опису бізнес процесу програмного забезпечення використовується BPMN модель (рисунок 2.1).

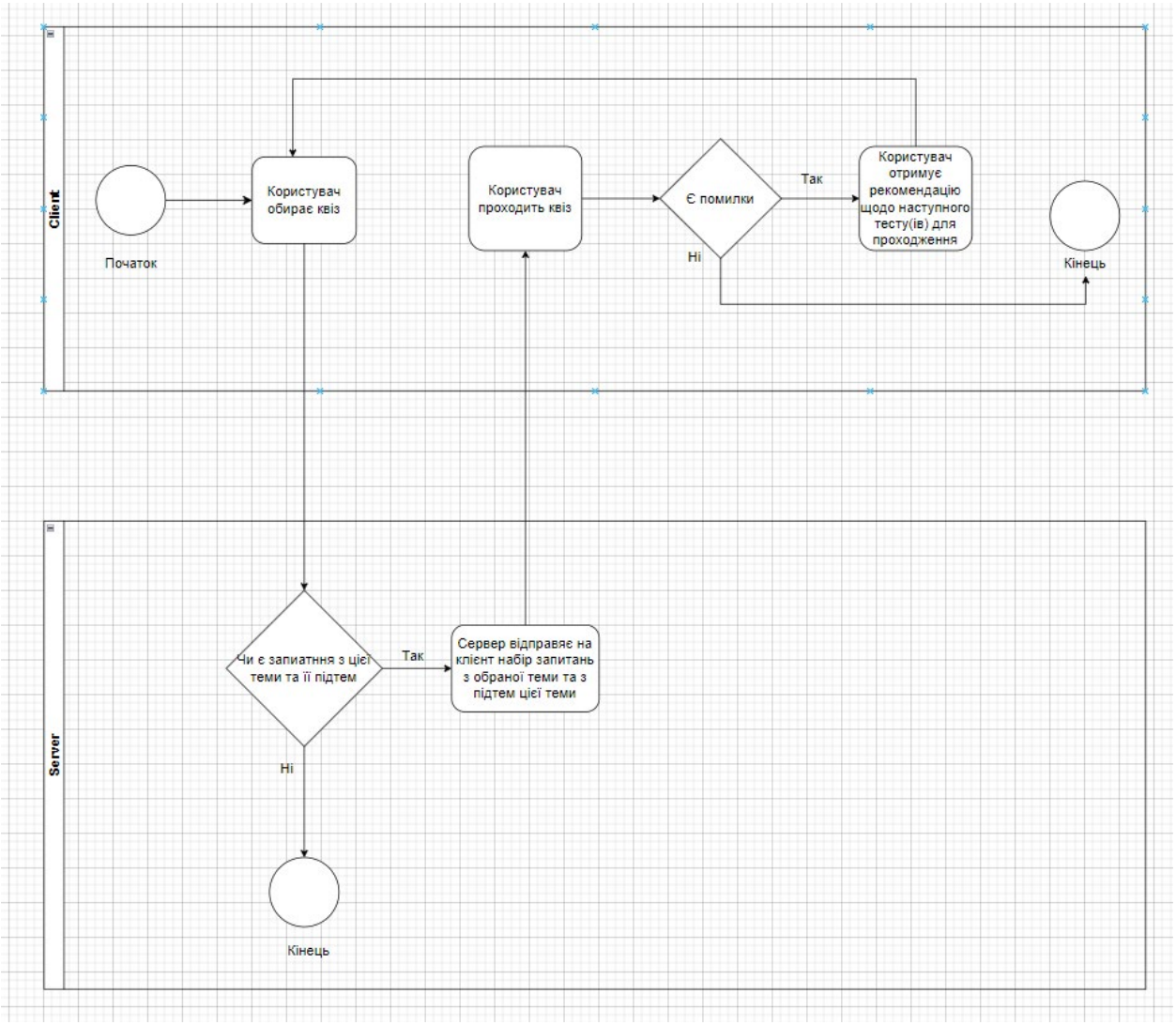


Рисунок 2.1 – Проходження тесту

2.2 Архітектура програмного забезпечення

Як вже було проаналізовано, буде використано монолітну архітектуру. Буде використано архітектурний патерн MVC(Model-View-Controller). Схема архітектури застосунку зображена на рисунку 2.2.

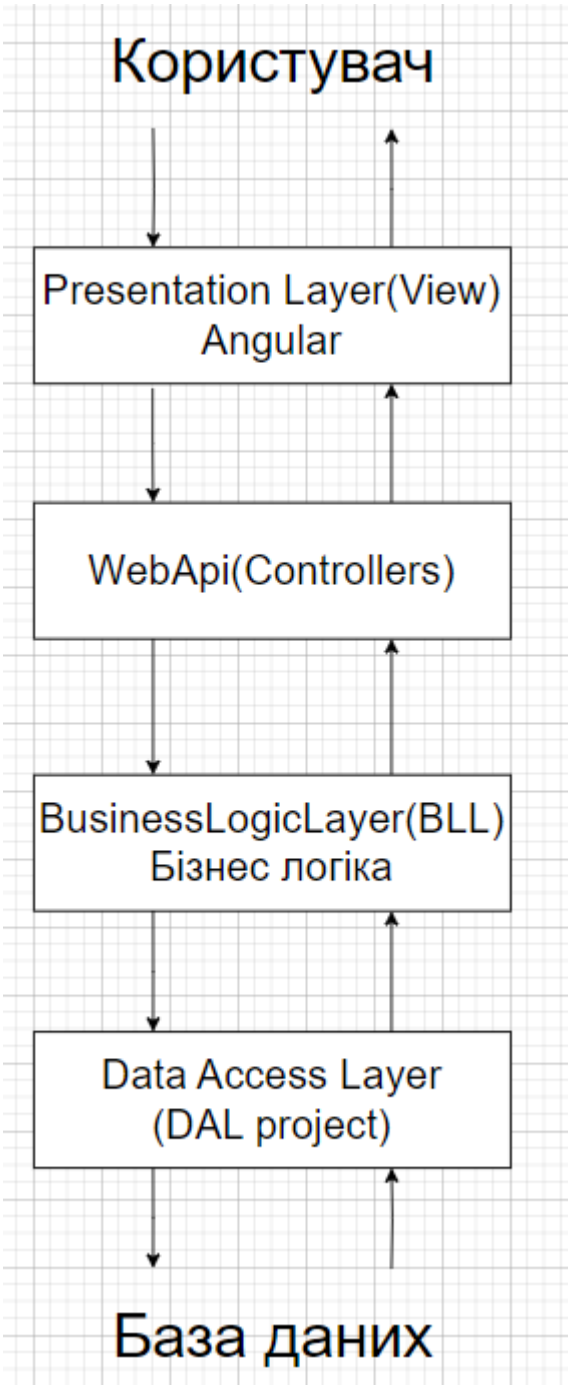


Рисунок 2.2 – Схема архітектури застосунку

Розглянемо архітектуру шляхом дослідження діаграми класів. Діаграма класів. Вона знаходиться в графічному матеріалі. Креслення 3 - Схема структурна класів програмного
забезпечення.

Користувач робить запит.

Запит потрапляє в ендпоінт[29] контролера.

Контролер використовує сервіси BLL.

BLL використовує Unit of Work[30] в якому зберігаються всі репозиторії DAL.

В якості бази даних було використано Microsoft SQL Server.

Для взаємодії з базою даних було використано EntityFramework Core в шарі DAL.

Було використано підхід CodeFirst.

Зв’язки між шарами реалізовані через інтерфейси, відповідно до SOLID.

Після цього, шляхом CORS View на ангулярі отримає з бекенду дані та надає їх користувачу.

2.3 Конструювання програмного забезпечення

Для реалізації основної задачі застосунку було прийнято рішення створити дерево дисциплін/тем/підтем математики.

Дерево складається з вузлів. Вузол можна побачити на рисунку 2.3

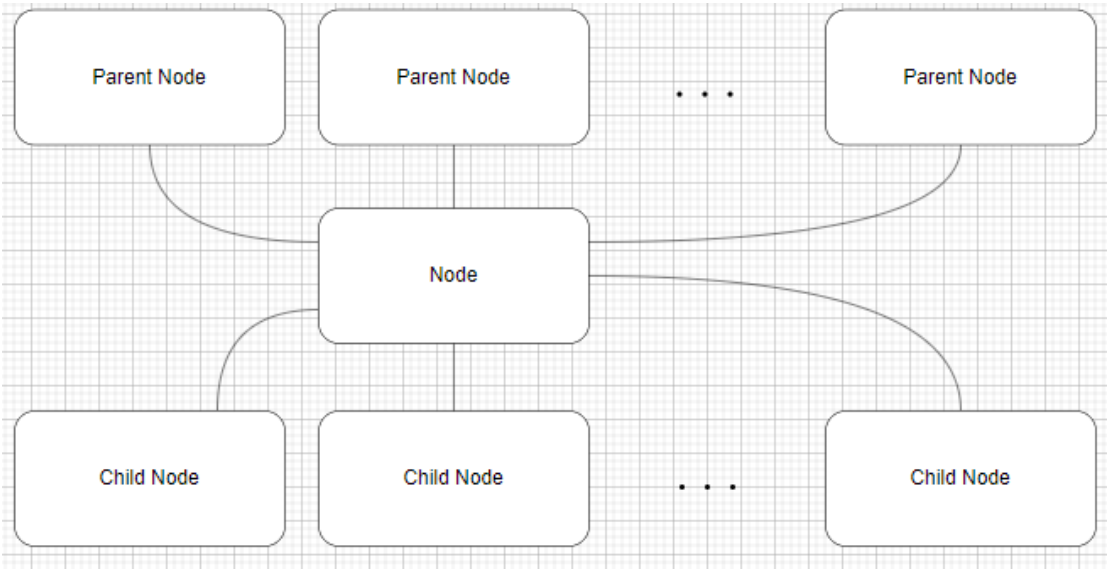


Рисунок 2.3 – Вузол дерева

У вузла є посилання на parent вузли(яких може бути 0 або більше). І, відповідно, є список дочірніх вузлів(яких може бути 0 або більше).

Це дозволяє надати користувачу запитання з підтем тієї теми що він обрав. І, відповідно, завдяки цьому, користувач може отримати запитання з підтем підтем. Це було зроблено рекурсивно, відповідно, користувач може отримати запитання з базових тематик, і, якщо в нього складнощі з базовою тематикою – він отримає рекомендовану літературу для того щоб закрити прогалини в знаннях і навичках з цієї теми.

Приблизний вигляд дерева показано на рисунку 2.4.

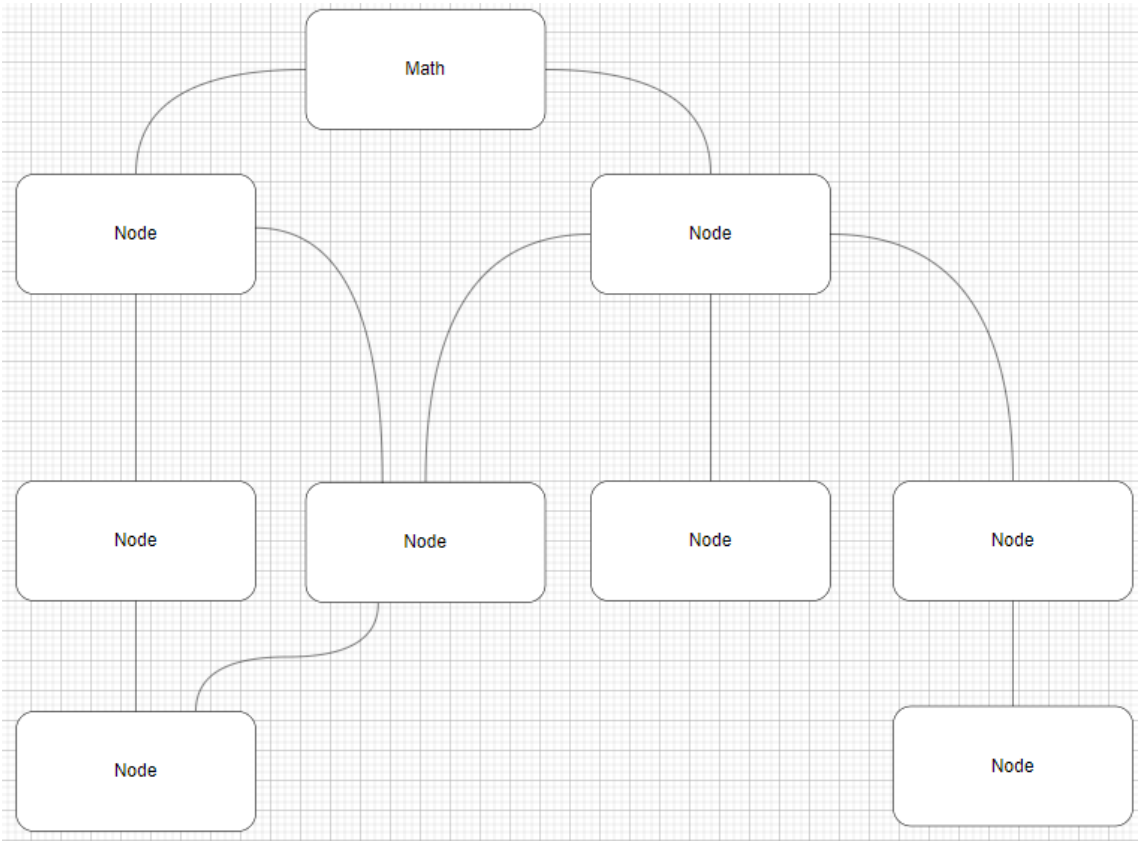


Рисунок 2.4 – Дерево дисциплін

Розглянемо схему бази даних застосунку. Вона знаходиться в графічному матеріалі. Креслення 2 - Схема бази даних.

Опис таблиць бази даних наведено в таблицях 2.1-2.6

Таблиця 2.1 – Опис таблиці AspNetUsers/ApplicationUsers

Таблиця	Назва поля	Тип даних	Опис
AspNetUsers	Id	nvarchar(450)	Ідентифікаційний номер користувача
	email	nvarchar(256)	Електронна пошта користувача
	passwordHash	nvarchar(max)	Хеш пароля користувача

	NormalizedUserName	nvarchar(256)	Описує ім’я користувача нормалізоване щоб користувач не міг реєструватися з вже існуючим в БД ім’ям якщо воно відрізняється тільки регістром тексту.

AspNetUsers – таблиця яка зберігає дані про зареєстрованих користувачів.

Таблиця бази даних бере велику кількість атрибутів із системної оскільки в коді було використано Identity. Тому в таблиці 2.1 наведено тільки декілька атрибутів.

Таблиця 2.2 – Опис таблиці Quizzes

Таблиця	Назва поля	тип даних	Опис
Quizzes	Id	int	Ідентифікаційний номер
	QuizDate	Datetime2(7)	Дата проходження квізу
	ApplicationUserId	nvarchar(450)	Id юзера.

Таблиця 2.3 – Опис таблиці Topics

Таблиця	Назва поля	тип даних	Опис
Topics	Id	int	Ідентифікаційний номер
	Text	nvarchar(max)	Назва теми

Таблиця 2.4 – Опис таблиці QuizTopic

Таблиця	Назва поля	тип даних	Опис
QuizTopic	QuizzezId	int	Ідентифікаційний номер квіза
	TopicsId	int	Ідентифікаційний номер теми

Таблиця 2.5 – Опис таблиці Questions

Таблиця	Назва поля	тип даних	Опис
Questions	Id	int	Ідентифікаційний номер
	Text	nvarchar(max)	Текст запитання
	TopicId	int	Id теми

Таблиця 2.6 – Опис таблиці Answers

Таблиця	Назва поля	тип даних	Опис
Answers	Id	int	Ідентифікаційний номер
	Text	nvarchar(max)	Текст відповіді
	IsCorrect	bit	Чи правильна відповідь
	QuestionId	int	Id запитання

Таблиця 2.7 – Опис таблиці TopicsForQuizzes

Таблиця	Назва поля	тип даних	Опис

Продовження таблиці 2.7

TopicsForQuizzes	Id	int	Ідентифікаційний номер
	QuizId	int	Ідентифікаційний номер квіза
	TopicId	int	Ідентифікаційний номер теми

Таблиця 2.8 – Опис таблиці Books

Таблиця	Назва поля	тип даних	Опис
Books	Id	int	Ідентифікаційний номер
	Text	nvarchar(max)	Інформація про книжку
	TopicId	int	Тема

Quiz має атрибут UserId, тут зв’язок 1:M. Тобто в одного користувача може бути декілька пройдених квізів.

Сам квіз зберігає в собі дату проходження.

Таблиця Topics зберігає в собі тему для тесту. І є основою для створення дерева дисциплін, оскільки ця таблиця посилається сама на себе.

Таблиця Books зберігає в собі рекомендовану літературу по конкретній темі.

QuizTopic – це розв’язочна таблиця для відношення багато до багатьох.

Квіз може мати декілька топіків які потрібно порекомендувати користувачу.

Зв’язок Question-Topic один до багатьох. Однакова тема може бути в кількох запитань.

І, для кожного запитання є декілька варіантів відповідей. Зв’язок - один до багатьох.

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці.

Програмне забезпечення, яке біло використано для розробки.

IDE JetBrains Rider, Webstorm. Rider – це IDE для розробки .NET частини. Webstorm – для розробки на Angular.

SQL Server Management Studio – СКБД.

Postman — це API-платформа, на якій розробники можуть розробляти, створювати, тестувати та повторювати свої API.

Було використано наступні пакети та фреймворки.

.NET Core 6.0 – фреймворк для розробки на C#.

Angular 15.1.0 – JavaScript framework від Google

AutoMapper.Extensions.Microsoft.DependencyInjection – це розширення для використання AutoMapper.

Microsoft.EntityFrameworkCore – це ORM від Microsoft.

Microsoft.AspNetCore.Authentication.JwtBearer - Компонент проміжного шару для перевірки справжності носія, що додається в конвеєр HTTP.

Swashbuckle.AspNetCore Swagger — це набір інструментів для розробки і тестування API.

Microsoft.AspNetCore.Identity.EntityFrameworkCore – пакет для реалізації автентифікації та авторизації.

Microsoft.Extensions.Configuration.UserSecrets – пакет для досягнення можливості не зберігати в репозиторії ConnectionString, JwtBearer та іншу подібну sensitive інформацію.

Аналіз системних вимог.

Web застосунок передбачено використовувати в браузері Google Chrome.

Особливих важких обчислень не передбачено. Тож в даному випадку вимогами до застосунку будуть вимоги браузера Google Chrome. А саме:

Windows

Windows 10, Windows Server 2016 або більш нові.

Процесор Intel Pentium 4 або більш новий, який підтримує SSE3 інструкції.

RAM: 1 GB для 32-bit або 2 GB для 64-bit.

Mac

macOS High Sierra 10.13 або більш нові.

Linux

64-bit Ubuntu 18.04+, Debian 10+, openSUSE 15.2+, або Fedora Linux 32+

Процесор Intel Pentium 4 або більш новий, який підтримує SSE3 інструкції[31.

2.4 Аналіз безпеки даних

Загальний регламент про захист даних (GDPR) є набором правил і положень, що регулюють збір, обробку та збереження персональних даних громадян Європейського Союзу. Цей регламент має на меті захист приватності та конфіденційності особистої інформації громадян.[32]

Основні принципи GDPR включають:

Згода та прозорість: Особи повинні надавати свою згоду на збір та обробку своїх персональних даних, і цей процес повинен бути прозорим і зрозумілим.

Обмеження цілей: персональні дані можуть збиратись лише для конкретних, чітко визначених та законних цілей.

Мінімізація даних: збір персональних даних повинен бути обмежений лише необхідним обсягом для досягнення визначених цілей.

Точність та обмеження збереження: Зібрані персональні дані повинні бути точними, а також зберігатись лише протягом необхідного періоду.

Безпека та захист: застосування відповідних технічних та організаційних заходів для захисту персональних даних від несанкціонованого доступу, втрати або пошкодження.

Права суб'єктів даних: громадяни мають право контролювати свої персональні дані, включаючи право на доступ до них, виправлення, видалення та перенесення.

Відповідальність: відповідність правил GDPR є обов'язковою, і організації, які збирають та обробляють персональні дані, несуть відповідальність за дотримання цих правил.

GDPR надає громадянам більший контроль над їхньою приватністю та особистими даними, а також стимулює організації до впровадження ефективних заходів безпеки та захисту даних. Це сприяє підвищенню довіри між користувачами та організаціями щодо обробки персональних даних.

Організації, що працюють з персональними даними громадян Європейського Союзу, повинні відповідати вимогам GDPR. Це означає, що вони повинні мати встановлені процедури та політики щодо обробки даних, забезпечувати безпеку і конфіденційність персональних даних, а також надавати громадянам права контролю та доступу до своїх даних.

У випадку порушення вимог GDPR, організації можуть бути піддані санкціям, включаючи адміністративні штрафи та відшкодування збитків. Це стимулює компанії до дотримання норм та правил GDPR, забезпечуючи більшу захищеність та конфіденційність персональних даних громадян.

В цілому, GDPR сприяє забезпеченню захисту приватності громадян та встановлює високі стандарти для обробки персональних даних. Впровадження цих правил допомагає зберегти довіру між користувачами та організаціями, а також сприяє створенню безпечного та етичного цифрового середовища.

Окрім того, GDPR накладає додаткові обов'язки на організації щодо повідомлення про порушення безпеки даних. В разі виявлення порушення, організація зобов'язана повідомити про це компетентний державний орган і, в

деяких випадках, самих постраждалих осіб. Це сприяє підвищенню прозорості та відповідальності у випадку порушення безпеки даних.

Одним з ключових аспектів GDPR є згода користувачів на обробку їх персональних даних. Організації повинні отримати згоду від користувачів перед збиранням та використанням їх даних. Згода повинна бути дана зрозумілою та свідомою формою, а користувачі мають право відкликати свою згоду в будь-який час.

Нарешті, GDPR також передбачає права користувачів щодо їх персональних даних. Ці права включають право на доступ до своїх даних, право на виправлення неправильних даних, право на видалення даних (право на забування), право на обмеження обробки даних, право на переносимість даних та право на заперечення проти обробки даних.

Узагальнюючи, GDPR є важливим законодавством, яке регулює обробку персональних даних у Європейському Союзі. Воно створює стандарти та вимоги для захисту приватності та безпеки даних, сприяє підвищенню довіри користувачів та організацій і встановлює права та обов'язки, пов'язані з обробкою персональних даних.

Розглянемо JWT (JSON Web Token) і OAuth (Open Authorization). Це два різні стандарти, які використовуються для забезпечення безпеки і автентифікації веб-додатків і API.

JWT є стандартом для створення та передачі токенів безпеки у форматі JSON. Він забезпечує захищений спосіб обміну даними між сторонами, використовуючи цифровий підпис. Токени JWT містять закодовану інформацію про користувача, таку як ідентифікатори, ролі або додаткові дані. Вони можуть бути використані для автентифікації користувачів і передачі даних між системами.

OAuth - це протокол авторизації, який дозволяє користувачам давати доступ до своїх ресурсів третім сторонам без розкриття свого пароля. OAuth використовує токени доступу для ідентифікації користувачів і надання дозволу на доступ до обмеженого набору ресурсів. Він дозволяє користувачам

авторизувати сторонні додатки, використовуючи існуючі облікові записи з попередньою згодою.

Основна відмінність між JWT і OAuth полягає в їхньому призначенні та використанні. JWT використовується для забезпечення безпеки даних і автентифікації між сторонами, тоді як OAuth - для авторизації користувачів та надання доступу до ресурсів третім сторонам. OAuth може використовувати токени JWT для передачі даних авторизації.

Користуючись JWT, є можливість передавати закодовані дані між сторонами, що дозволяє легко передавати інформацію про користувача і підтверджувати її автентичність. OAuth забезпечує рішення для управління авторизацією та дозволами доступу, дозволяючи користувачам контролювати,

Відповідно, в даному застосунку було використано JWT.

В даному застосунку було використано секрети, що дозволяє не надавати публічного доступу(наприклад, через Github) до sensitive(чутливих) даних, таких як connection string, jwt bearer та інші дані, які потрібні розробнику, але їх публікація у вільний доступ не є хорошою ідеєю.

Для того щоб дані користувача були тільки його власністю, в програмі повинна бути передбачена можливість повного видалення профілю користувача. Така можливість повинна бути в користувача особисто.

Висновки до розділу

В першому розділі було проаналізовано предметну область.

На основі цього аналізу було виділено основні бізнес процеси застосунку.

В межах другого розділу було розглянуто, проведено елемент життєвого циклу моделювання та конструювання програмного забезпечення

Було змодельовано та проаналізовано програмне забезпечення. Як результат, було отримано BPMN діаграму.

Було розроблено архітектуру застосунку. Як результат це дає розуміння щодо створення застосунку.

Було створено базу даних застосунку. Це важливий крок, адже він дає нам розуміння щодо зберігання даних в застосунку.

Було сконструйовано програмне забезпечення. Як результат, було отримано повний код застосунку. На основі коду було створено діаграму класів застосунку.

Було проаналізовано безпеку даних. Було розглянуто GDPR. A GDPR є регуляторним документом, який встановлює правила та умови для збору, обробки та збереження персональних даних громадян Європейського Союзу. Цей регламент спрямований на захист приватності та забезпечення конфіденційності особистої інформації громадян.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Було обрано SonarQube для аналізу якості програмного забезпечення за метриками.

Важливими метриками в даній ситуації є підтримуваність коду, складність коду, покриття тестами, зв’язність коду[33].

Підтримуваність коду (code maintainability): Ця метрика вимірює, наскільки легко розробникам вносити зміни в існуючий код. Вона оцінюється на основі факторів, таких як зрозумілість коду, наявність коментарів, структура програми і використання кращих практик програмування. Висока підтримуваність коду сприяє швидкому внесенню змін і запобігає появі нових помилок[34].

Складність коду (code complexity): ця метрика вимірює складність програмного коду, зазвичай на основі метрик, таких як цикломатична складність, розмір функцій і модулів, глибина вкладеності умовних конструкцій тощо. Висока складність коду може призводити до збільшення кількості помилок і ускладнювати розуміння та підтримку коду[35].

Покриття тестами (test coverage): ця метрика вимірює, наскільки велика частина програмного коду охоплена тестами. Вона вказує, яка кількість коду перевірена під час виконання тестів. Високе покриття тестами забезпечує більшу впевненість у працездатності програми і допомагає виявляти помилки раніше[36].

Зв'язність коду: ця метрика вимірює ступінь залежності між компонентами програмного коду. Вона оцінює, наскільки добре кожна частина коду взаємодіє з іншими частинами. Висока зв'язність сприяє більшій чіткості і простоті коду[37].

Було отримано такий звіт від SonarQube:

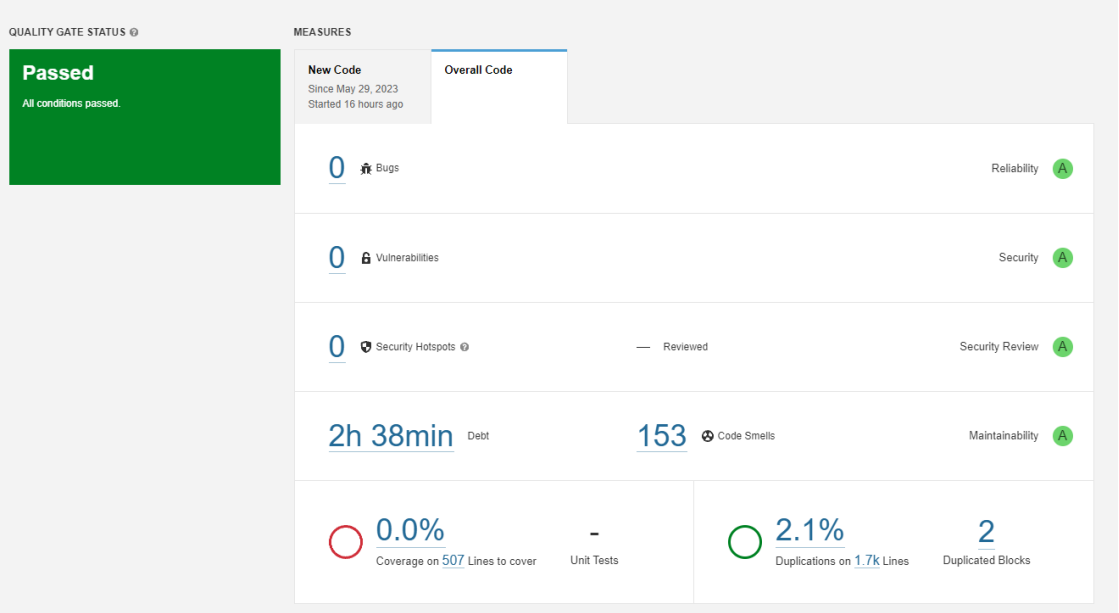


Рисунок 3.1 – Звіт SonarQube щодо якості коду додатку.

Я повинний зазначити що було знайдено велику кількість запахів коду, цю складність можна легко вирішити за наявності часу.

Окремо було протестовано покриття коду юніт-тестами.

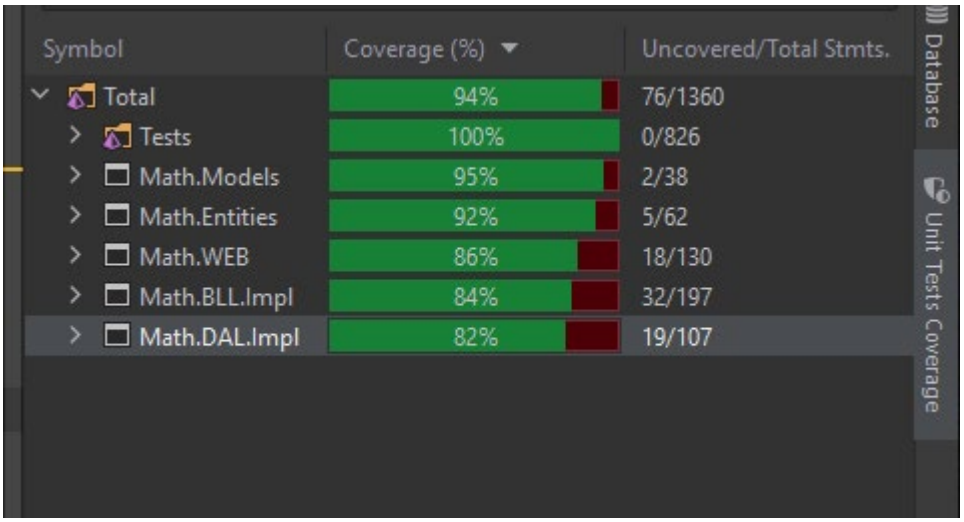


Рисунок 3.2 – Покриття юніт-тестами.

Було досягнуто високого рівня покриття коду юніт-тестами.

Нефункціональними вимогами були:

– Інтерфейс користувача повинен бути інтуїтивно зрозумілим та зручним для використання;

– Платформа повинна бути надійною та безпечною, щоб уникнути витоку особистої інформації користувачів;

– Швидкість роботи платформи повинна бути достатньою(завантаження запитань повинно займати до 1 секунди часу) для зручного використання користувачами;

Проаналізуємо, чи вдалося їх виконати:

– Інтерфейс користувача інтуїтивно зрозумілий та зручний для використання;

– Користувач має можливість видалити свій профіль, відповідно, він володіє можливістю знищити свої дані;

– Швидкість роботи платформи достатня. Завантаження запитань повинно займати до 1 секунди часу для зручного використання користувачами;

3.2 Опис процесів тестування

Процес тестування програмного забезпечення (ПЗ) включає проведення різних видів тестів з метою виявлення помилок, оцінки якості та перевірки відповідності вимогам. Основні типи тестування ПЗ включають наступні:

Юніт-тестування - це тестування найменших функціональних одиниць програмного коду, таких як функції, методи або класи. Мета цього тестування - переконатися, що кожна одиниця працює коректно окремо від інших частин системи[38].

Інтеграційне тестування - це тип тестування який перевіряє взаємодію між різними компонентами або модулями системи. Мета - виявити помилки, що виникають при з'єднанні компонентів та перевірити правильність передачі даних між ними[39].

Системне тестування - це тестування вже побудованої системи як єдиного цілого. Мета - перевірити, чи відповідає система вимогам та функціональним вимогам, виявити помилки та оцінити якість системи[40].

UI тестування (також відоме як тестування користувацького інтерфейсу) - це тип тестування, спрямований на перевірку коректності та придатності інтерфейсу користувача програмного забезпечення. У цьому типі тестування перевіряються елементи, такі як кнопки, поля введення, меню, форми тощо, а також інтерактивні дії користувача та їх вплив на інтерфейс[41].

Ручне тестування - це процес тестування, який виконується вручну без використання автоматизованих засобів чи скриптів. У ручному тестуванні тестер виконує дії, що відповідають реальному користувачу програмного забезпечення, для перевірки правильності функціонування програми та виявлення помилок[42].

В даній ситуації нам необхідні юніт-тести для тестування конкретних методів наших класів. Безумовно, необхідні інтеграційні тести для тестування взаємозв’язків між нашими класами або компонентами. Також, необхідне наскрізне тестування для перевірки роботи системи в цілому.

Безумовно не можна обійтись і без ручного тестування. Ці тести не піддаються автоматизації, їх задає сам тестувальник.

Розглянемо декілька прикладів ручного тестування. Приклади таких тестів наведено в таблицях 3.1-3.4.

Таблиця 3.1 – Тест 1

Тест	Реєстрація користувача.
Модуль	Реєстрація користувача.
Номер тесту	1
Початковий стан системи	Користувач знаходиться на сторінці реєстрації.
Вхідні данні	Електронна пошта, пароль, підтвердження паролю .
Опис проведення тесту	У відповідні поля вводяться: коректна електронна пошта, яка до цього не була зареєстрована в системі, пароль від 4 символів, який містить хоча б з одну англійську літеру, підтвердження паролю, яке співпадає з раніше введеним паролем. Після цього натискається кнопка підтвердження реєстрації.
Очікуваний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації.

Продовження таблиці 3.1

Фактичний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації.

Таблиця 3.2 – Тест 2

Тест	Автентифікація та авторизація користувача.
Модуль	Автентифікація та авторизація користувача.
Номер тесту	2
Початковий стан системи	Користувач знаходиться на сторінці автентифікації.
Вхідні данні	Логін, пароль.
Опис проведення тесту	Користувач вводить логін і пароль до свого профілю.
Очікуваний результат	Автентифікація і авторизація проходить успішно. Користувач успішно входить в систему.
Фактичний результат	Автентифікація і авторизація проходить успішно. Користувач успішно входить в систему.

Таблиця 3.3 – Тест 3

Тест	Автентифікація та авторизація користувача.
Модуль	Автентифікація та авторизація користувача.
Номер тесту	3
Початковий стан системи	Користувач знаходиться на сторінці автентифікації.
Вхідні данні	Логін, хибний пароль.
Опис проведення тесту	Користувач вводить логін і хибний пароль до свого профілю.
Очікуваний результат	Автентифікація і авторизація проходить не успішно. Користувач повертається на сторінку входу в систему.
Фактичний результат	Автентифікація і авторизація проходить не успішно. Користувач повертається на сторінку входу в систему.

Таблиця 3.4 – Тест 4

Тест	Проходження тесту з дисципліни
Модуль	Проходження тесту
Номер тесту	4
Початковий стан системи	Користувач знаходиться на домашній сторінці

Продовження таблиці 3.4

Вхідні данні	Обрана тема.
Опис проведення тесту	Користувач обирає тему зі списку. Отримує набір запитань, відповідає на них. Отримує оцінку та рекомендацію.
Очікуваний результат	Користувач отримав оцінку за проходження тесту та рекомендацію щодо наступного тесту для проходження.
Фактичний результат	Користувач отримав оцінку за проходження тесту та рекомендацію щодо наступного тесту для проходження.

Таблиця 3.5 – Тест 5

Тест	Генерація запитань з вкладеними темами і підтемами
Модуль	Генерація запитань
Номер тесту	5
Початковий стан системи	Користувач знаходиться на домашній сторінці
Вхідні данні	Користувача авторизовано
Опис проведення тесту	Користувач обирає тему зі списку. Отримує набір запитань, з обраної теми та тем і підтем обраної теми
Очікуваний результат	Користувач отримав запитання, з обраної теми та тем і підтем обраної теми.
Фактичний результат	Користувач отримав запитання, з обраної теми та тем і підтем обраної теми.

3.3 Опис контрольного прикладу

Розглянемо повний опис контрольного прикладами з усіма можливими розгалуженнями.

Користувач пропонує два варіанти вхідних даних.

1. username dmytro, email dmytro@gmail.com, fullname Dmytro, password ddd

2. username dmytro, email dmytro@gmail.com, fullname Dmytro, password dddd.

Користувач хоче пройти тест з арифметики.

Опис контрольного прикладу.

Користувач реєструється в системі з першим набором вхідних даних і отримує помилку. (Рисунок 3.3).

Користувач реєструється в системі з другим набором вхідних даних успішно. (Рисунок 3.4).

Користувач проходить автентифікацію не успішно. (Рисунок 3.5).

Користувач проходить автентифікацію успішно. (Рисунок 3.6).

Користувач обирає тест з арифметики(Рисунок 3.7).

Користувач проходить тест(Рисунок 3.8).

Користувач отримує результат, в якому бачить прогалини в знання і рекомендовану літературу(Рисунок 3.9).

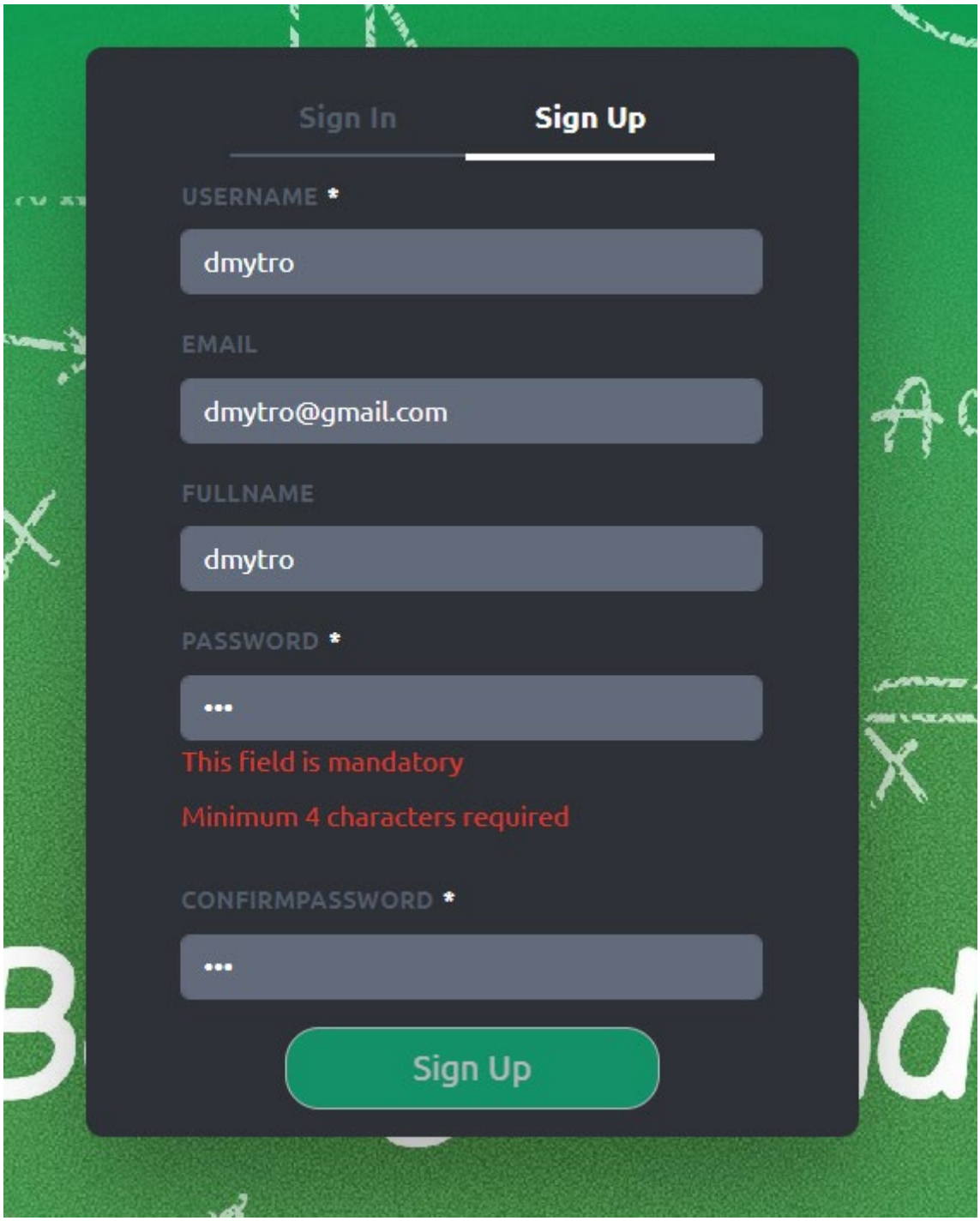


Рисунок 3.3 - Користувач реєструється в системі з першим набором вхідних даних і отримує помилку.

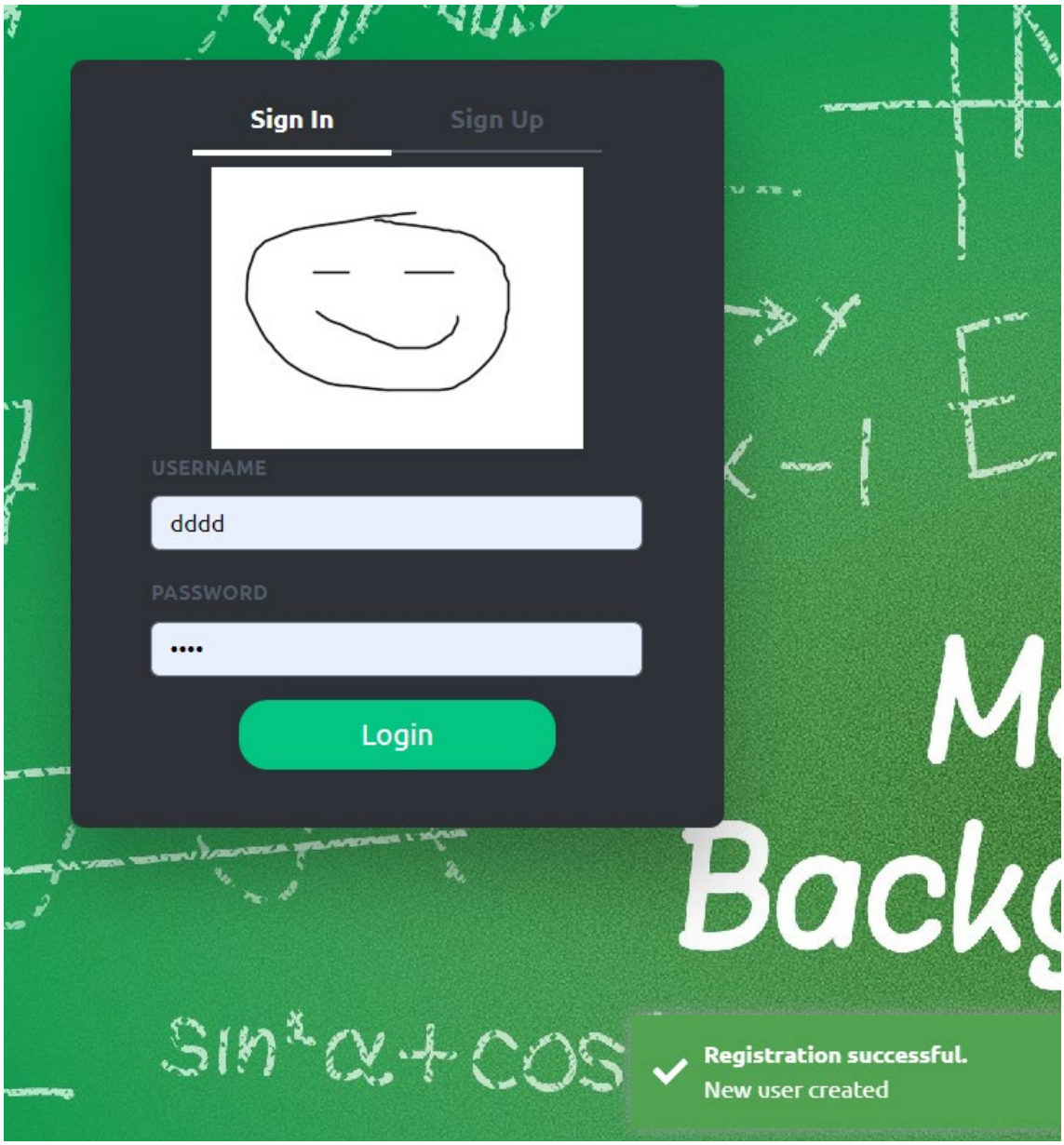


Рисунок 3.4 - Користувач реєструється в системі з другим набором вхідних даних успішно.

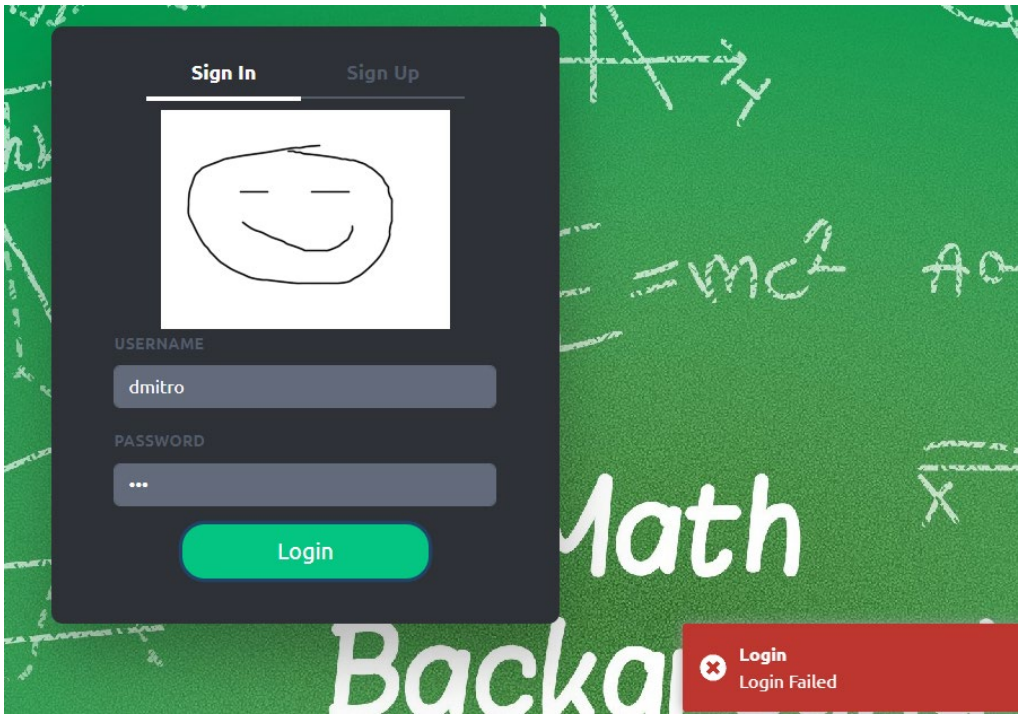


Рисунок 3.5 Користувач проходить автентифікацію не успішно.

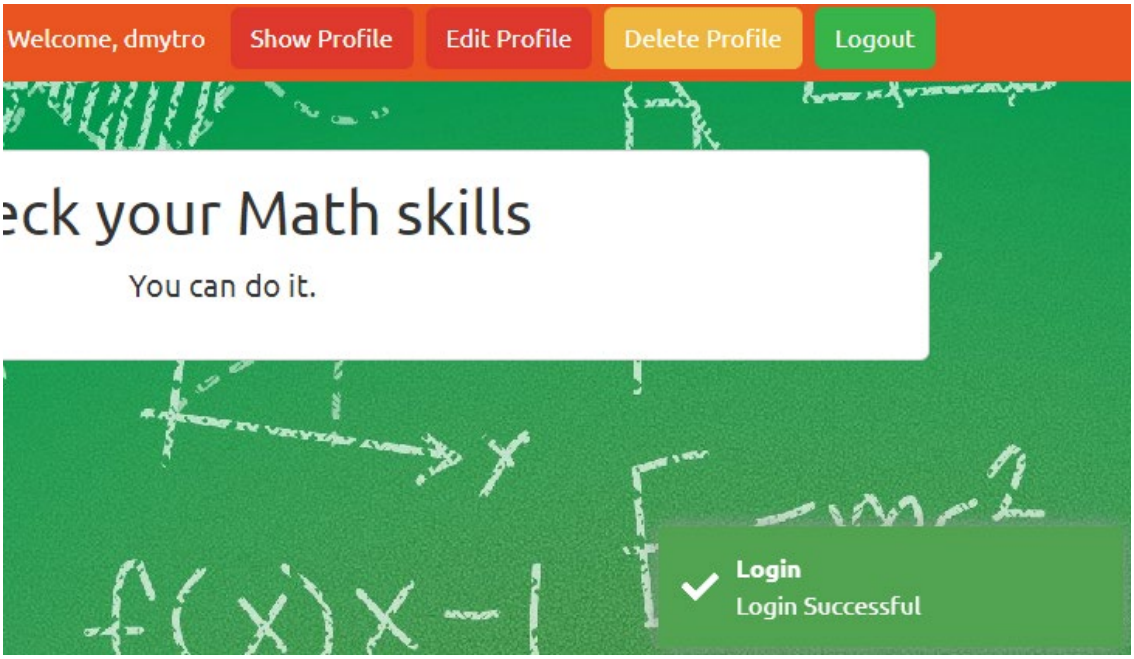


Рисунок 3.6 - Користувач проходить автентифікацію успішно.

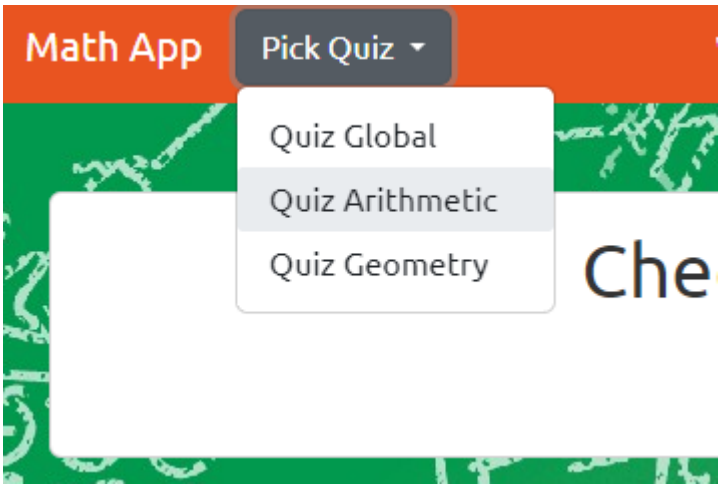


Рисунок 3.7 – Користувач обирає тест з арифметики

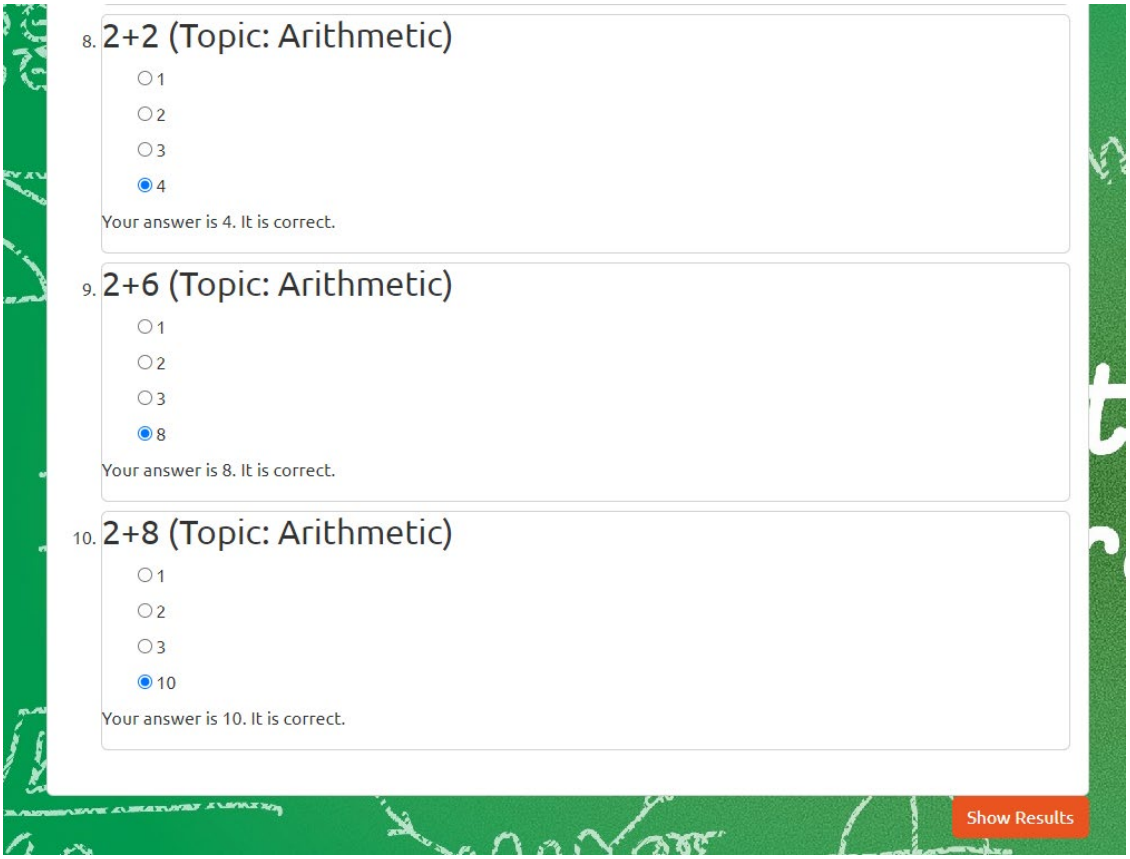


Рисунок 3.8 - Користувач проходить тест

Рисунок 3.9 - Користувач отримує результат.

Висновки до розділу

В межах третього розділу було розглянуто, проведено елемент життєвого циклу аналіз якості та тестування програмного забезпечення.

Було проведено аналіз якості програмного забезпечення. Як результат, було виділено і розглянуто наступні метрики такі як: підтримуваність коду, складність коду, покриття тестами, зв’язність коду.

Було перевірена якість програмного забезпечення статичним аналізатором коду SonarQube. І, як результат була отримана оцінка від цього аналізатора коду.

В рамках даного розділу було проаналізоване виконання нефункціональних вимог. Як результат – вони були успішно виконані.

Було розглянуто такі типи тестування як юніт-тестування, інтеграційне тестування, системне тестування, UI тестування, ручне тестування.

Було наведено перелік тестів ручного тестування. Як результат, було отриману набір необхідних тестів для якісного тестування даного програмного забезпечення. Важливо протестувати як тести з очікуваними успішними так і не успішними діями користувача

Було розглянуто повний опис контрольного прикладами з усіма можливими розгалуженнями. І, як результат було продемонстровано повний

шлях користувача даного програмного забезпечення. Відповідно, було протестовано як успішні варіанти дій користувача, так і не успішні дії.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Проаналізуємо засоби розгортання застосунку.

Існує кілька варіантів розгортання програмного забезпечення, включаючи сервер, хмарні платформи та контейнеризацію. Було проведено порівняння варіантів розгортання та було обрано варіант для поточного застосунку.

1. Самостійний сервер:

Варіант: розгорнути застосунок на фізичному сервері або віртуальній машині, налаштувати серверне програмне забезпечення (наприклад, IIS для .NET, Nginx або Apache для Angular).

Перевагами є повний контроль над інфраструктурою, можливість налаштовувати сервер під власні потреби, висока продуктивність та доступність.

Недоліками є те що він вимагає знань про налаштування сервера та інфраструктури, витрати на обладнання та утримання, обмежені можливості масштабування.

2. Хмарні платформи:

Варіант: використовувати хмарні платформи, такі як Amazon Web Services (AWS), Microsoft Azure або Google Cloud Platform (GCP), для розгортання вашого застосунку.

Перевагами є можливість легко масштабувати ресурси, широкий вибір послуг інфраструктури, гнучкість та мобільність, безпека та відмінна доступність.

Недоліками є вартість (оплата за використання ресурсів), можуть виникати обмеження або залежності від хмарних провайдерів.

3. Контейнеризація:

Варіант: використовувати контейнеризацію (наприклад, Docker) для упакування та розгортання вашого застосунку разом з усіма залежностями та конфігурацією.

Перевагами є переносимість, швидкість розгортання, ізольованість та масштабованість, легке управління ресурсами та конфігурацією.

Недоліками є те що даний варіант вимагає навчання та знань про контейнеризацію, можуть виникати проблеми з мережевими налаштуваннями.

При виборі одного з цих варіантів важливо враховувати потреби, технічні знання, бюджет та майбутні перспективи масштабування. З урахуванням того, що серверна частина написана на C# та клієнтська частина на Angular, контейнеризація може бути привабливим варіантом. Є можливість сконтейнеризувати додаток за допомогою Docker і розгорнути його на будь-якій інфраструктурі, будь то самостійний сервер або хмарна платформа. Контейнеризація дозволить легко керувати ресурсами, швидко розгортати та масштабувати застосунок, зберігаючи його ізольованість та переносимість.

Heroku є хмарною платформою розгортання та управління додатками. Основні особливості Heroku включають:

Простота використання: Heroku надає простий інтерфейс та командний рядок для розгортання та керування додатками. Є можливість легко налаштувати та розгорнути застосунок без необхідності в глибоких знаннях інфраструктури.

Масштабованість. Heroku дозволяє легко масштабувати додатки від невеликих проектів до великих під навантаженням. Є можливість змінювати кількість ресурсів, таких як сервери та бази даних, в залежності від потреби.

Підтримка багатьох мов програмування. Heroku підтримує широкий спектр мов програмування, включаючи C#, Ruby, Python, Node.js, Java, PHP, Go і багато інших. Це дає велику гнучкість у виборі мови для проекту.

Інтеграція з Git. Heroku використовує Git для розгортання додатків. Є можливість легко взаємодіяти з репозиторієм Git, здійснювати релізи нових версій додатку та автоматично розгортати зміни після коміту в Git.

Існують також альтернативи Heroku, які варто розглянути. Ось декілька схожих альтернатив Heroku:

1. AWS Elastic Beanstalk - це сервіс управління розгортанням додатків в Amazon Web Services (AWS). Він дозволяє легко розгортати, масштабувати та керувати додатками на інфраструктурі AWS. Elastic Beanstalk підтримує багато мов програмування та надає широкі можливості налаштування[43].

2. Google App Engine - це хмарна платформа розгортання та керування додатками в Google Cloud Platform. Google App Engine дозволяє розгортати додатки на основі контейнерів або за допомогою серверлес-архітектури. Вона підтримує кілька мов програмування та надає високий рівень автоматизації[44].

3. Microsoft Azure App Service - це сервіс розгортання додатків в Microsoft Azure. Azure App Service підтримує різні мови програмування та платформи, включаючи .NET, Java, Node.js, Python та інші. Він надає можливості автоматичного масштабування та інтеграції з іншими сервісами Azure[45].

4. DigitalOcean App Platform - це хмарна платформа розгортання, яка спрощує процес розгортання та керування додатками. DigitalOcean App Platform підтримує різні мови програмування та надає зручний інтерфейс для налаштування і масштабування додатків[46].

Клієнтську і серверну частини програмного забезпечення було вирішено розгорнути на платформі Heroku. Для розгортання було використано сервіс GitHub Actions, який надає можливості для постійної інтеграції і розгортання.

Розгортання починається коли новий код застосунку доставляється у репозиторій у гілку main/master.

4.2 Підтримка програмного забезпечення

Підтримка програмного забезпечення включає в себе процеси, спрямовані на забезпечення стабільної та безперебійної роботи програмного продукту. Одним з важливих аспектів підтримки є оновлення програмного забезпечення, отримання нової версії і впровадження її в робоче середовище.

Оновлення програмного забезпечення було реалізовано за допомогою таких кроків як отримання нової версії, аналіз змін, тестування, план оновлення і впровадження.

Отримання нової версії. Спочатку потрібно отримати нову версію програмного забезпечення. Для цього використовується репозиторій на Github.

Аналіз змін. Перед оновленням важливо проаналізувати зміни, внесені в нову версію програмного забезпечення. Це допоможе визначити, які зміни вплинуть на вашу систему, функціональність та інтеграцію.

Тестування. Перед впровадженням нової версії необхідно провести тестування. Це необхідно для того щоб виявити можливі проблеми або несумісності з існуючими компонентами або залежностями. Слід виконувати функціональне тестування, регресійне тестування та інші види тестування, щоб переконатися, що програмне забезпечення працює належним чином.

План оновлення. Перед впровадженням нової версії слід визначити план оновлення. Це повинно включати резервне копіювання даних, встановлення нової версії на тестове середовище для фінального тестування та планування часу, коли оновлення буде впроваджено.

Впровадження. Після успішного тестування і підготовки середовища впроваджуйте нову версію програмного забезпечення. Відповідно, нову версію програмного забезпечення встановлюємо на наш сервер. При впровадженні важливо слідкувати за помилками або проблемами та швидко реагувати на них.

Перевірка та моніторинг. Після впровадження нової версії перевіряємо її роботу з метою виявити проблеми. Крім того, необхідний постійний нагляд за роботою програмного забезпечення, щоб вчасно виявляти та вирішувати проблеми, які можуть виникнути після оновлення.

Цей процес оновлення допоможе безпечно та ефективно отримати нову версію програмного забезпечення. Важливо планувати і виконувати оновлення з урахуванням особливостей застосунку та докладати достатньо зусиль для тестування та перевірки нової версії перед впровадженням.

Висновки до розділу

В межах четвертого розділу було розглянуто, проведено елемент життєвого циклу впровадження та супровід програмного забезпечення.

Було розглянуто декілька варіантів розгортання програмного забезпечення таких сервер, хмарні платформи та контейнеризацію. Було розглянуто як їх переваги, так і їх недоліки. Як результат було обрано хмарну платформу для розгортання поточного застосунку.

Було розглянуто хмарну платформу Heroku та її альтернативи такі як AWS Elastic Beanstalk, Google App Engine, Microsoft Azure App Service, DigitalOcean App Platform.

Клієнтську і серверну частини програмного забезпечення було вирішено розгорнути на платформі Heroku. Для розгортання було використано сервіс GitHub Actions, який надає можливості для постійної інтеграції і розгортання.

Наступним етапом було розглянути підтримку програмного забезпечення. Вона включає в себе процеси, спрямовані на забезпечення стабільної та безперебійної роботи програмного продукту. Одним з важливих аспектів підтримки є оновлення програмного забезпечення, отримання нової версії і впровадження її в робоче середовище.

Оновлення програмного забезпечення було реалізувано за допомогою таких кроків як отримання нової версії, аналіз змін, тестування, план оновлення і впровадження. Після цього обов’язково буде перевірена робота

програмного забезпечення та буде виконуватись моніторинг роботи застосунку. Цей процес оновлення допомагає безпечно та ефективно отримати нову версію програмного забезпечення.

ВИСНОВКИ

В межах виконання дипломної роботи було виконано повний життєвий цикл розробки програмного забезпечення. А саме в межах етапів. На першому етапі було проаналізовано вимоги до програмного забезпечення та як наслідок було проаналізовано альтернативні технічні рішення, було розроблено функціональні і нефункціональні вимоги. Було поставлено задачі. На другому етапі було змодельовано та сконструйовано програмне забезпечення. Була побудована архітектура застосунку, використана база даних, проаналізована безпека. На третьому етапі було проведено аналіз якості та протестовано програмне забезпечення. На четвертому етапі було розглянуто питання розгортання та підтримки програмного забезпечення.

Практична значимість одержаних результатів.

В результаті виконання дипломного проєкту було розроблено програмне забезпечення для покращення ефективності математичного тренажеру. Воно було сконструйовано за монолітною архітектурою, з використання шаблону MVC. При розробці програмного забезпечення було використано .NET 6 для серверної частини застосунку та Angular 15 для клієнтської частини застосунку.

В якості середовища розробки обрано Rider та WebStorm від JetBrains

У якості БД використано Microsoft SQL Server.

Користувачі програми мають можливість перевірити свої знання і навички з математики. Вони мають змогу знайти прогалини в своїх знаннях і навичках та усунути ці прогалини.

Дане програмне забезпечення можна використовувати в школах та університетах. Також його можуть використовувати здобувачі освіти самостійно для покращення своїх знань та навичок.

Це програмне забезпечення може бути адаптовано і для інших наукових дисциплін.

Функціонал програмного забезпечення може бути розширено, з додаванням нових можливостей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Використання інформаційних технологій у навчанні математики для формування компетентності особистості [Електронний ресурс] // naurok.com.ua – Режим доступу до ресурсу: https://naurok.com.ua/vikoristannya-informaciynih-tehnologiy-u-navchanni-matematiki-dlya-formuvannya-kompetentnosti-osobistostipre-149469.html (дата звернення: 16.06.2023).

2. Інноваційні технології на уроках математики [Електронний ресурс] // naurok.com.ua – Режим доступу до ресурсу: https://naurok.com.ua/stattya-innovaciyni-tehnologi-na-urokah-matematiki-226498.html (дата звернення: 16.06.2023).

3. Огляд інтерактивних методів [Електронний ресурс] // multycourse.com.u – Режим доступу до ресурсу: http://multycourse.com.ua/ua/page/19/69 (дата звернення: 16.06.2023).

4. МАТЕРІАЛИ ДРУГОЇ МІЖНАРОДНОЇ КОНФЕРЕНЦІЇ З АДАПТИВНИХ ТЕХНОЛОГІЙ УПРАВЛІННЯ НАВЧАННЯМ ATL-2016 [Електронний ресурс] // academia.edu – Режим доступу до ресурсу: https://ww.academia.edu/28701468/%D0%9C%D0%90%D0%A2%D0%95%D0%A0%D0%86%D0%90%D0%9B%D0%98_%D0%94%D0%A0%D0%A3%D0%93%D0%9E%D0%87_%D0%9C%D0%86%D0%96%D0%9D%D0%90%D0%A0%D0%9E%D0%94%D0%9D%D0%9E%D0%87_%D0%9A%D0%9E%D0%9D%D0%A4%D0%95%D0%A0%D0%95%D0%9D%D0%A6%D0%86%D0%87_%D0%97_%D0%90%D0%94%D0%90%D0%9F%D0%A2%D0%98%D0%92%D0%9D%D0%98%D0%A5_%D0%A2%D0%95%D0%A5%D0%9D%D0%9E%D0%9B%D0%9E%D0%93%D0%86%D0%99_%D0%A3%D0%9F%D0%A0%D0%90%D0%92%D0%9B%D0%86%D0%9D%D0%9D%D0%AF_%D0%9D%D0%90%D0%92%D0%A7%D0%90%D0%9D%D0%9D%D0%AF%D0%9C_ATL_2016 (дата звернення: 16.06.2023).

5. User Experience Design [Електронний ресурс] // semanticstudios.com – Режим доступу до ресурсу: http://semanticstudios.com/user_experience_design/ (дата звернення: 31.05.2023).

6. Microservices vs. monolithic architecture [Електронний ресурс] // atlassian.com – Режим доступу до ресурсу: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith (дата звернення: 13.05.2023).

7. MVC [Електронний ресурс] // developer.mozilla.org – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Glossary/MVC (дата звернення: 13.05.2023).

8. Model View Presenter [Електронний ресурс] // medium.com – Режим доступу до ресурсу: https://medium.com/geekculture/model-view-presenter-78725944e5bb (дата звернення: 13.05.2023).

9. Model View ViewModel [Електронний ресурс] // medium.com – Режим доступу до ресурсу: https://medium.com/geekculture/model-view-viewmodel-d20bf341fbd6 (дата звернення: 13.05.2023).

10. 7+1 JavaScript Frameworks Compared [Електронний ресурс] // plainenglish.i – Режим доступу до ресурсу: https://javascript.plainenglish.io/7-1-javascript-frameworks-compared-a58f8f16239a (дата звернення: 31.05.2023).

11. What is Angular? [Електронний ресурс] // angular.io – Режим доступу до ресурсу: https://angular.io/guide/what-is-angular (дата звернення: 13.05.2023).

12. React The library for web and native user interfaces [Електронний ресурс] // react.dev – Режим доступу до ресурсу: https://react.dev/ (дата звернення: 13.05.2023).

13. Рейтинг мов програмування 2022. C# обійшов Java, TypeScript зрівнявся з PHP, а Dart — найбільш комфортна мова [Електронний ресурс] // dou.ua. – 14. – Режим доступу до ресурсу: https://dou.ua/lenta/articles/language-rating-2022/ (дата звернення: 01.05.2023).

14. Fast & powerful cross-platform .NET IDE [Електронний ресурс] // www.jetbrains.com – Режим доступу до ресурсу: https://www.jetbrains.com/rider/features/ (дата звернення: 13.05.2023).

15. Visual Studio 2022 IDE - Programming Tool for Software Developers [Електронний ресурс] // microsoft.com – Режим доступу до ресурсу: https://visualstudio.microsoft.com/vs/ (дата звернення: 13.05.2023).

16. Code editing. Redefined. [Електронний ресурс] // visualstudio.com – Режим доступу до ресурсу: https://code.visualstudio.com/ (дата звернення: 15.05.2023).

17. Entity Framework Core [Електронний ресурс] // microsoft.com – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/ef/core/ (дата звернення: 15.05.2023).

18. Dapper [Електронний ресурс] // github.com – Режим доступу до ресурсу: https://github.com/DapperLib/Dapper (дата звернення: 13.05.2023).

19. AutoMapper [Електронний ресурс] // automapper.org – Режим доступу до ресурсу: https://automapper.org/ (дата звернення: 13.05.2023).

20. EmitMapper.Core [Електронний ресурс] // nuget.org – Режим доступу до ресурсу: https://www.nuget.org/packages/EmitMapper.Core/ (дата звернення: 13.05.2023).

21. MapsterMapper [Електронний ресурс] // github.com – Режим доступу до ресурсу: https://github.com/MapsterMapper/Mapster (дата звернення: 13.05.2023).

22. Code Quality Tool & Secure Analysis with SonarQube [Електронний ресурс] // sonarsource.com – Режим доступу до ресурсу: https://www.sonarsource.com/products/sonarqube/ (дата звернення: 16.06.2023).

23. The Visual Studio Extension for .NET Developers [Електронний ресурс] // jetbrains.com – Режим доступу до ресурсу: https://www.jetbrains.com/resharper/. (дата звернення: 16.06.2023).

24. StyleCop [Електронний ресурс] // visualstudio.com – Режим доступу до ресурсу:

https://marketplace.visualstudio.com/items?itemName=ChrisDahlberg.StyleCop. (дата звернення: 16.06.2023).

25. Roslyn Analyzers [Електронний ресурс] // github.com – Режим доступу до ресурсу: https://github.com/dotnet/roslyn-analyzers. (дата звернення: 16.06.2023).

26. Khan Academy [Електронний ресурс] // Khan Academy – Режим доступу до ресурсу: https://www.khanacademy.org/math (дата звернення: 01.05.2023).

27. Coursera’s Mission, Vision, and Commitment to Our Community | Coursera [Електронний ресурс] // coursera.org – Режим доступу до ресурсу: https://about.coursera.org/. (дата звернення: 26.05.2023).

28. Brilliant [Електронний ресурс] // Brilliant – Режим доступу до ресурсу: https://brilliant.org/courses/math (дата звернення: 01.05.2023).

29. API endpoint [Електронний ресурс] // techtarget.com – Режим доступу до ресурсу: https://www.techtarget.com/searchapparchitecture/definition/API-endpoint (дата звернення: 31.05.2023).

30. Unit of Work in Repository Pattern [Електронний ресурс] // medium.com – Режим доступу до ресурсу: https://medium.com/@rahulrulz680/unit-of-work-in-repository-pattern-b48a862d06a2 (дата звернення: 31.05.2023).

31. Chrome browser system requirements [Електронний ресурс] // google.com – Режим доступу до ресурсу: https://support.google.com/chrome/a/answer/7100626?hl=en (дата звернення: 17.05.2023).

32. General Data Protection RegulationGDPR [Електронний ресурс] – Режим доступу до ресурсу: https://gdpr-info.eu/ (дата звернення: 31.05.2023).

33. Вимоги до якості програмного забезпечення [Електронний ресурс] // elib.lntu.edu.ua – Режим доступу до ресурсу: https://elib.lntu.edu.ua/sites/default/files/elib_upload/%D0%A2%D0%B5%D1%8

1%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F/page5.html (дата звернення: 31.05.2023).

34. DevOps tech: Code maintainability [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/architecture/devops/devops-tech-code-maintainability (дата звернення: 31.05.2023).

35. A Simple Understanding of Code Complexity [Електронний ресурс] // codegrip.tech – Режим доступу до ресурсу: https://www.codegrip.tech/productivity/a-simple-understanding-of-code-complexity/ (дата звернення: 31.05.2023).

36. Test Coverage in Software Testing [Електронний ресурс] // guru99.com – Режим доступу до ресурсу: https://www.guru99.com/test-coverage-in-software-testing.html (дата звернення: 31.05.2023).

37. How do you design and test your objects for low coupling and high cohesion? [Електронний ресурс] // linkedin.com – Режим доступу до ресурсу: https://www.linkedin.com/advice/0/how-do-you-design-test-your-objects-low-coupling (дата звернення: 31.05.2023).

38. Модульне тестування [Електронний ресурс] // qalight.ua – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/modulne-testuvannya/ (дата звернення: 31.05.2023).

39. Інтеграційне тестування [Електронний ресурс] // qalight.ua – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/integratsijne-testuvannya/ (дата звернення: 31.05.2023).

40. Системне тестування [Електронний ресурс] // qalight.ua – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/sistemne-testuvannya/ (дата звернення: 31.05.2023).

41. Тестування веб-проєктів: основні етапи та поради [Електронний ресурс] // qalight.ua – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/testuvannya-veb-proektiv-osnovni-etapi-ta-poradi/ (дата звернення: 31.05.2023).

42. Ручне та автоматизоване тестування [Електронний ресурс] // qalight.ua – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/ruchne-ta-avtomatizovane-testuvannya/ (дата звернення: 31.05.2023).

43. AWS Elastic Beanstalk [Електронний ресурс] // amazon.com – Режим доступу до ресурсу: https://aws.amazon.com/elasticbeanstalk/ (дата звернення: 31.05.2023).

44. App Engine [Електронний ресурс] // google.com – Режим доступу до ресурсу: https://cloud.google.com/appengine (дата звернення: 31.05.2023).

45. App Service [Електронний ресурс] // microsoft.com – Режим доступу до ресурсу: https://azure.microsoft.com/en-us/products/app-service (дата звернення: 31.05.2023).

46. DigitalOcean App Platform [Електронний ресурс] // digitalocean.com – Режим доступу до ресурсу: https://www.digitalocean.com/products/app-platform (дата звернення: 31.05.2023).

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

UNICHECK

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1015479792

Дата перевірки:
07.06.2023 11:54:03 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
07.06.2023 11:54:42 EEST

ID користувача:
76913

Назва документа: **ІТ-93_Пономарьов_ПЗ**

Кількість сторінок: **64** Кількість слів: **9725** Кількість символів: **73597** Розмір файлу: **2.83 MB** ID файлу: **1015137435**

9.72%
Схожість

Найбільша схожість: **2.93%** з джерелом з Бібліотеки (ID файлу: **1014967607**)

4.41% Джерела з Інтернету	99	Сторінка 66
8.93% Джерела з Бібліотеки	273	Сторінка 67

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

Програмне забезпечення для підвищення ефективності роботи математичного тренажера.

Текст програми

КПІ.ІТ-9303.045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Євгеній ВОВК

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Олексій ПОНОМАРЬОВ

Київ – 2023

Файл auth.guard.ts

```
import {Injectable} from '@angular/core';
import {ActivatedRouteSnapshot, CanActivate, Router, RouterStateSnapshot}
from '@angular/router';

import {ToastrService} from "ngx-toastr";

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {

  constructor(private toastr: ToastrService, private router: Router) {
  }

  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): boolean {
    if (localStorage.getItem('token') != null) return true;

    this.router.navigate(['/user/login']);
    return false;
  }
}
```

Файл shared.module.ts

```
import {NgModule} from '@angular/core';
import {CommonModule} from '@angular/common';
import {BsDropdownModule} from "ngx-bootstrap/dropdown";
import {ToastrModule} from "ngx-toastr";


@NgModule({
  declarations: [],
  imports: [
    CommonModule,
    BsDropdownModule.forRoot(),
    ToastrModule.forRoot({
      positionClass: 'toast-bottom-right'
    })
  ],

  exports: [
    BsDropdownModule,
    ToastrModule
  ]
})
export class SharedModule {
}
```

Файл question.service.ts

```
import {Injectable, OnInit} from '@angular/core';
import {HttpClient} from '@angular/common/http';
import {Observable} from 'rxjs';
import {QuestionModel} from "../_models/q/question";
import {environment} from "../../environments/environment";
import {SharedService} from "./shared.service";

@Injectable({
  providedIn: 'root'
})
export class QuestionService implements OnInit{
```

```
  // private readonly baseUrl = 'api/questions';
  private readonly baseUrl = environment.apiUrl + '/Question';

  pickedTopic: string = '';

  constructor(private http: HttpClient, private sharedService: SharedService)
{
  }

  ngOnInit(): void {
      this.sharedService.pickedTopic$.subscribe((newData: string) => {
      this.pickedTopic = newData;
    });
  }

  getQuestions(): Observable<QuestionModel[]> {
    return this.http.get<QuestionModel[]>(this.baseUrl + '/Random');
  }

  getQuestionsByTopic(topic: string): Observable<QuestionModel[]> {
    if (topic == 'Global Quiz') topic='random';
    let route: string = this.baseUrl + '/' + topic;
    console.log(route)
    return this.http.get<QuestionModel[]>(route);
  }

  getQuestionById(id: number): Observable<QuestionModel> {
    return this.http.get<QuestionModel>(`${this.baseUrl}/${id}`);
  }


  addQuestion(question: QuestionModel): Observable<QuestionModel> {
    return this.http.post<QuestionModel>(this.baseUrl, question);
  }

  updateQuestion(question: QuestionModel): Observable<QuestionModel> {
    return this.http.put<QuestionModel>(`${this.baseUrl}/${question.id}`,
question);
  }

  deleteQuestion(id: number): Observable<any> {
    return this.http.delete(`${this.baseUrl}/${id}`);
  }


}
```

Файл shared.service.ts

```
import {Injectable} from '@angular/core';
import {BehaviorSubject, Observable} from "rxjs";

@Injectable({
  providedIn: 'root'
})
export class SharedService {
  fullName$: BehaviorSubject<string> = new BehaviorSubject<string>('');
  isLoggedIn$: BehaviorSubject<any> = new BehaviorSubject<any>('');

  pickedTopic$: BehaviorSubject<string> = new BehaviorSubject<string>('')

  constructor() {
  }
```

```
  setFullName(fullName: string) {
    this.fullName$.next(fullName);
  }

  setIsLoggedIn(isLoggedIn: string) {
    this.isLoggedIn$.next(isLoggedIn);
  }

  updateData(topic: string): void {
    this.pickedTopic$.next(topic);
  }

}
```

Файл topic.service.ts

```
import {Injectable} from '@angular/core';
import {environment} from "../../environments/environment";
import {HttpClient} from "@angular/common/http";
import {SharedService} from "./shared.service";
import {Observable} from "rxjs";
import {TopicModel} from "../_models/q/topic";

@Injectable({
  providedIn: 'root'
})
export class TopicService {
  private readonly baseUrl = environment.apiUrl + '/Topic';

  pickedTopic: string = '';

  constructor(private http: HttpClient, private sharedService: SharedService)
{
  }

  getTopics(): Observable<TopicModel[]> {
    console.log(this.baseUrl);
    let result = this.http.get<TopicModel[]>(this.baseUrl);
    return result;
  }

}
```

Файл user.service.ts

```
import {Injectable, OnInit} from '@angular/core';
import {FormBuilder, FormGroup, Validators} from "@angular/forms";
import {HttpClient} from "@angular/common/http";
import {environment} from "../../environments/environment";
import {BehaviorSubject, Observable, tap} from "rxjs";

@Injectable({
  providedIn: 'root'
})
export class UserService implements OnInit {

  private isLoggedInSubject: BehaviorSubject<boolean> = new
BehaviorSubject<boolean>(false);
  public isLoggedIn$: Observable<boolean> =
this.isLoggedInSubject.asObservable();

  private fullNameSubject: BehaviorSubject<string> = new
BehaviorSubject<string>('');
```

```
  public fullName$: Observable<string> = this.fullNameSubject.asObservable();

  private userIdSubject: BehaviorSubject<string> = new
BehaviorSubject<string>('');
  public userId$: Observable<string> = this.userIdSubject.asObservable();


  constructor(private fb: FormBuilder, private http: HttpClient) {
    const isLoggedIn = !!localStorage.getItem('token');
    this.isLoggedInSubject.next(isLoggedIn);
  }

  ngOnInit(): void {
  }

  readonly BaseURI = environment.apiUrl;


  formModel = this.fb.group({
    UserName: ['', Validators.required],
    Email: ['', Validators.email],
    FullName: [''],
    Passwords: this.fb.group({
      Password: ['', [Validators.required, Validators.minLength(4)]],
      ConfirmPassword: ['', Validators.required]
    }, {validator: this.comparePasswords})

  });

  comparePasswords(fb: FormGroup) {
    const confirmPswrdCtrl = fb.get('ConfirmPassword');
    // passwordMismatch
    // confirmPswrdCtrl.errors={passwordMismatch:true}
    if (confirmPswrdCtrl?.errors == null || 'passwordMismatch' in
confirmPswrdCtrl.errors) {
      if (fb.get('Password')?.value !== confirmPswrdCtrl?.value) {
        confirmPswrdCtrl?.setErrors({passwordMismatch: true});
      } else {
        confirmPswrdCtrl?.setErrors(null);
      }
    }
  }


  register() {
    let body = {
      UserName: this.formModel.value.UserName,
      Email: this.formModel.value.Email,
      FullName: this.formModel.value.FullName,
      Password: this.formModel.value.Passwords.Password
    };
    return this.http.post(this.BaseURI + '/ApplicationUser/Register', body);
  }


  login(formData: any) {
    return this.http.post(this.BaseURI + '/ApplicationUser/Login',
formData).pipe(
      tap((res: any) => {
        // Save the token to local storage
        localStorage.setItem('token', res.token);

        this.getUserProfile().subscribe(
          () => {
```

```
            // User profile retrieved successfully, userName$ will be updated
automatically
          },
          (error: any) => {
            console.log(error);
          }
        );
      })
    );
  }

  getUserProfile() {
    // Include the authorization token in the request headers
    const headers = { 'Authorization': 'Bearer ' +
localStorage.getItem('token') };
    const options = { headers: headers };

    return this.http.get(this.BaseURI + '/UserProfile', options).pipe(
      tap((res: any) => {
        this.fullNameSubject.next(res.userName);
      })
    );
  }

  setLoggedIn(isLoggedIn: boolean): void {
    this.isLoggedInSubject.next(isLoggedIn);
  }

  deleteProfile(userId: string) {
    return
this.http.delete(`${this.BaseURI}/ApplicationUser/Delete/${userId}`);
  }

  doNothing(userId: string) {
    return new Observable<object>;
  }

  setFullName(fullName: string): void {
    this.fullNameSubject.next(fullName);
  }

  setUserId(userId: string): void {
    this.userIdSubject.next(userId);
  }

  updateUser(user: any) {
    return
this.http.patch(`${this.BaseURI}/ApplicationUser/Update/${user.id}`, user);
  }
}
```

Файл home.component.ts

```
import {Component, OnInit} from '@angular/core';
import {UserService} from "../_services/user.service";
import {Router} from "@angular/router";

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {
  registerMode = false;
```

```
  users: any;

  userDetails: any;

  constructor(private router: Router, private service: UserService) {
  }

  ngOnInit(): void {
    this.service.getUserProfile().subscribe(
      res => {
        this.userDetails = res;
      },
      err => {
        console.log(err);
      },
    )
  }

  registerToggle() {
    this.registerMode = !this.registerMode;
  }

  cancelRegisterMode(event: boolean) {
    this.registerMode = event;
  }

  onLogout() {
    localStorage.removeItem('token');
    this.router.navigate(['/user/login']);
  }
}
```

Файл nav.component.ts

```
import {Component, OnInit, HostListener} from '@angular/core';
import {Router} from '@angular/router';
import {ToastrService} from 'ngx-toastr';
import {UserService} from '../_services/user.service';
import {SharedService} from '../_services/shared.service';
import {TopicService} from "../_services/topic.service";
import {TopicModel} from "../_models/q/topic";
import {QuestionModel} from "../_models/q/question";
import {QuestionService} from "../_services/question.service";

@Component({
  selector: 'app-nav',
  templateUrl: './nav.component.html',
  styleUrls: ['./nav.component.css']
})
export class NavComponent implements OnInit {
  isLoggedIn: any;
  fullName: string = '';

  isBurgerMenuOpen: boolean = false;
  isProfileButtonsVisible: boolean = true;

  pickedTopic: string = '';

  topics: TopicModel[] = [new TopicModel('Global Quiz')];

  constructor(
    private router: Router,
    private toastr: ToastrService,
```

```
    private userService: UserService,
    private sharedService: SharedService,
    private topicService: TopicService,
    private questionService: QuestionService
  ) {
  }

  ngOnInit(): void {
    this.getTopics();

    this.userService.fullName$.subscribe(fullName => {
      this.fullName = fullName;
    });

    this.sharedService.fullName$.subscribe(fullName => {
      this.fullName = fullName;
    });

    this.sharedService.isLoggedIn$.subscribe(isLoggedIn => {
      this.isLoggedIn = isLoggedIn;
    });

    this.userService.isLoggedIn$.subscribe(isLoggedIn => {
      this.isLoggedIn = isLoggedIn;
    });

    this.sharedService.pickedTopic$.subscribe(topic => {
      this.pickedTopic = topic;
    });
  }

  @HostListener('window:resize', ['$event'])
  onWindowResize(event: Event) {
    this.updateButtonVisibility();
  }

  toggleBurgerMenu() {
    this.isBurgerMenuOpen = !this.isBurgerMenuOpen;
    this.updateButtonVisibility();
  }

  private updateButtonVisibility() {
    const windowWidth = window.innerWidth;
    if (windowWidth <= 768) {
      this.isProfileButtonsVisible = !this.isBurgerMenuOpen;
    } else {
      this.isProfileButtonsVisible = true;
    }
  }

  SignOut() {
    localStorage.clear();
    this.router.navigate(['/user/login']);
    this.isLoggedIn = null;
    console.log('logged out');
    this.userService.setLoggedIn(false);
    this.toastr.success('Logout successful', 'Logout');
    this.fullName = 'Guest';
  }

  getValue(topic: string) {
    this.sharedService.updateData(topic);
  }
```

```
  showMe() {
    console.log(this.topics);
  }

  getTopics() {
    this.topicService.getTopics().subscribe(
      (topics: TopicModel[]) => {
        for (const topic of topics) {
          if(topic.text == "Math") continue;
          this.topics.push(topic);
        }
      },
      (error: any) => {
        console.log(error);
      }
    );
  }

}
```

Файл quiz.component.ts

```
import {Component} from '@angular/core';
import {QuestionModel} from "../../_models/q/question";
import {QuestionService} from "../../_services/question.service";
import {AnswerModel} from "../../_models/q/answer";
import {Router} from "@angular/router";
import {SharedService} from "../../_services/shared.service";
import {TopicModel} from "../../_models/q/topic";

@Component({
  selector: 'app-quiz',
  templateUrl: './quiz.component.html',
  styleUrls: ['./quiz.component.css']
})
export class QuizComponent {

  questions: QuestionModel[] | null = null;
  correctAnswersCount: number = 0;
  wrongAnswersCount: number = 0;

  isLoadingQuestions: boolean = true;
  showQuestions: boolean = true;
  dictionary: { [key: string]: number } = {};

  maxKeys: string[] = [];

  pickedTopic: string = '';

  showingResults: boolean = false;

  topicsInThisQuiz: TopicModel [] = [];
  topicNamesInThisQuiz: string [] = [];
  topicsToShowInResultOfThisQuiz: TopicModel [] = [];


  constructor(private questionService: QuestionService,
              private router: Router,
              private sharedService: SharedService) {
  }

  ngOnInit(): void {
    console.log('quiz component on init')
```

```
    this.sharedService.pickedTopic$.subscribe((topic: string) => {
      this.pickedTopic = topic;
    });

    this.dictionary = {};
    this.getQuiz();
  }

  getQuiz() {
    console.log('getQuiz');
    if (this.pickedTopic == '') return;
    this.questionService.getQuestionsByTopic(this.pickedTopic)
      .subscribe(
        (questions: QuestionModel[]) => {
          this.questions = questions;
          this.resetAnswersCount();
          this.isLoadingQuestions = false;
        },
        (error: any) => {
          console.error(error);
        }
      );
  };

  checkAnswer(question: QuestionModel, selectedAnswer: AnswerModel) {
    if (!this.showingResults) {
      // Mark the question as answered
      question.answered = true;
      question.answeredAnswer = selectedAnswer;
      // console.log(question.topicModel?.text);

      if (!this.topicNamesInThisQuiz.includes(question.topicModel?.text!)) {
        // console.log('hi');
        // console.log(question.topicModel?.text);
        this.topicNamesInThisQuiz.push(question.topicModel?.text!);
        this.topicsInThisQuiz.push(question.topicModel!);
      }

    }
  }

  showResults() {
    this.showQuestions = false;
    this.showingResults = true;
    this.resetAnswersCount();

    if (this.questions) {
      for (const question of this.questions) {
        if (!question.answered) continue;
        const selectedAnswer = question.answeredAnswer;
        if (selectedAnswer?.isCorrect) {
          this.correctAnswersCount++;
        } else {
          this.wrongAnswersCount++;
          // Add the question topic to the wrong answer topics list
          const topic = question.topicModel?.text;
          if (topic) {
            if (this.dictionary.hasOwnProperty(topic)) {
              this.dictionary[topic] += 1;
            } else {
              this.dictionary[topic] = 1;
            }
          }
        }
```

```
      }
    }

    let maxCount = 0;
    for (const key in this.dictionary) {
      const value = this.dictionary[key];
      if (value > maxCount) {
        maxCount = value;
      }
    }


    for (const key in this.dictionary) {
      if (this.dictionary[key] == maxCount) {
        this.maxKeys?.push(key);
      }
    }
    this.maxKeys?.sort();

    for (const topic of this.maxKeys) {
      for (const topicElement of this.topicsInThisQuiz) {
        if (topicElement.text == topic) {
          this.topicsToShowInResultOfThisQuiz.push(topicElement);
        }
      }
    }

    console.log(this.topicsToShowInResultOfThisQuiz);
  }

  resetAnswersCount() {
    this.correctAnswersCount = 0;
    this.wrongAnswersCount = 0;
    this.maxKeys = [];
  }

  hasUnansweredQuestions(): boolean {
    if (this.questions) {
      return this.questions.some(question => !question.answered);
    }
    return false;
  }
}
```

Файл login.component.ts

```
import {Component, OnInit} from '@angular/core';
import {NgForm} from "@angular/forms";

import {Router} from "@angular/router";
import {ToastrService} from "ngx-toastr";
import {UserService} from "../../_services/user.service";

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit{
  formModel = {
    UserName: '',
    Password: ''
  }
```

```
  constructor(private userService: UserService, private router: Router,
private toastr: ToastrService) {
  }

  ngOnInit(): void {
    if (localStorage.getItem('token') != null){
      this.router.navigate(['/home']);
    }
  }

  onSubmit(form: NgForm) {
    this.userService.login(form.value).subscribe(
      (res: any) => {
        localStorage.setItem('token', res.token);
        this.router.navigateByUrl('/home');

        console.log('logged in');
        this.toastr.success('Login Successful', 'Login');
        this.userService.setLoggedIn(true);
        this.userService.setFullName(res.fullName);
        this.userService.setUserId(res.userId);
      },
      err => {
          console.log(err);
          this.toastr.error('Login Failed',  'Login');
      }
    );
  }

}
```

Файл registration.component.ts

```
import {Component, OnInit} from '@angular/core';
import {ToastrService} from "ngx-toastr";
import {ValidationErrors} from "@angular/forms";
import {UserService} from "../../../_services/user.service";
import {Router} from "@angular/router";

@Component({
  selector: 'app-registration',
  templateUrl: './registration.component.html',
  styleUrls: ['./registration.component.css']
})
export class RegistrationComponent implements OnInit {

  constructor(public userService: UserService, private toastr: ToastrService,
private router: Router) {
  }

  ngOnInit(): void {
    this.userService.formModel.reset();
  }

  onSubmit() {
    this.userService.register().subscribe(
      (res: any) => {
        if (res.succeeded) {
          this.userService.formModel.reset();
          this.toastr.success('New user created', 'Registration
successful.');
          this.router.navigateByUrl('user/login');
        } else {
```

```
          res.errors.forEach((element: ValidationErrors|any) => {
            switch (element.code) {
              case 'DuplicateUserName' :
                this.toastr.error('username has already been taken',
'Registration failed');
                break;

              default:
                this.toastr.error(element.description,  'Registration
failed');
                break;
            }
          });
        }
      },
      err => {
        console.log(err);
      }
    );
  }
}
```

Файл show-user-profile.component.ts

```
import {Component, OnInit} from '@angular/core';
import {UserService} from "../../_services/user.service";

@Component({
  selector: 'app-show-user-profile',
  templateUrl: './show-user-profile.component.html',
  styleUrls: ['./show-user-profile.component.css']
})
export class ShowUserProfileComponent implements OnInit{

  constructor(private userService: UserService) {
  }

  userProfile: any;

  ngOnInit(): void {
    this.userService.getUserProfile().subscribe(
      (res: any) => {
        this.userProfile = res;
        console.log('got user data');
      },
      (error: any) => {
        console.log(error);
      }
    );
  }
}
```

Файл edit-user.component.ts

```
import {Component, OnInit} from '@angular/core';
import {UserService} from "../../_services/user.service";
import {ToastrService} from "ngx-toastr";
import {Router} from "@angular/router";

@Component({
  selector: 'app-edit-user',
  templateUrl: './edit-user.component.html',
  styleUrls: ['./edit-user.component.css']
})
```

```
export class EditUserComponent implements OnInit {
  user: any;


  constructor(
    private userService: UserService,
    private toastr: ToastrService,
    private router: Router
  ) {
    this.user = {
      fullName: '',
      email: ''
    };
  }

  ngOnInit(): void {
    this.userService.getUserProfile().subscribe(
      (res: any) => {
        this.user.id = res.id;
        this.user.email = res.email;
        this.user.fullName = res.fullName;


        console.log('got user data');
      },
      (error: any) => {
        console.log(error);
      }
    );
  }

  onSubmit() {
    console.log('before');
    console.log(this.user);
    this.userService.updateUser(this.user).subscribe(
      () => {
        console.log('after');
        console.log(this.user);
        this.toastr.success('User updated successfully', 'Edit Profile');
        this.router.navigateByUrl('showUserInfo');
      },
      (error: any) => {
        this.toastr.error('Error updating user', 'Edit Profile');
        console.log(error);
      }
    );
  }


}
```

Файл delete-profile.component.ts

```
import {Component} from '@angular/core';
import {UserService} from "../../_services/user.service";
import {Router} from "@angular/router";
import {ToastrService} from "ngx-toastr";
import {SharedService} from "../../_services/shared.service";

@Component({
  selector: 'app-delete-profile',
  templateUrl: './delete-profile.component.html',
  styleUrls: ['./delete-profile.component.css']
})
```

```
export class DeleteProfileComponent {
  constructor(private userService: UserService, private router: Router,
private toastr: ToastrService, private sharedService: SharedService) {
  }

  userProfile: any;

  ngOnInit(): void {
    this.userService.getUserProfile().subscribe(
      (res: any) => {
        this.userProfile = res;
      },
      (error: any) => {
        console.log(error);
      }
    );
  }

  deleteProfile(userId: string) {
    console.log(userId);
    console.log(this.userProfile.id);

    this.userService.deleteProfile(userId).subscribe(
      () => {

        // Profile deleted successfully, handle any additional logic
      },
      (error: any) => {
        // Handle error if the profile deletion fails
      }
    );

    this.sharedService.setFullName('Guest');
    this.sharedService.setIsLoggedIn('');

    this.toastr.success('Delete Successful', 'Delete');

    localStorage.clear();

    this.router.navigate(['user/login']);
  }

  yes(){
    this.deleteProfile(this.userProfile.id);
  }

  no(){
    this.router.navigate(['home']);
  }
}
```

Файл QuestionController.cs

```
using Math.BLL.Abstract.Services;
using Microsoft.AspNetCore.Mvc;
using Models;

namespace Math.WEB.Controllers;

[Route("api/[controller]")]
[ApiController]
public class QuestionController : ControllerBase
{
    private readonly IConfiguration _configuration;
```

```
    private readonly IQuestionService _questionService;
    private readonly ITopicService _topicService;
    private readonly IAnswerService _answerService;


    public QuestionController(IConfiguration configuration,
        IQuestionService questionService, ITopicService topicService,
IAnswerService answerService)
    {
        _configuration = configuration;
        _questionService = questionService;
        _topicService = topicService;
        _answerService = answerService;
    }

    [HttpGet("{topic}")]
    public async Task<ActionResult<ICollection<QuestionModel>>>
GetQuestionsByWord(string topic)
    {
        try
        {
            ICollection<QuestionModel> questions;

            if (topic == "random")
            {
                questions = await _questionService.Get10RandomQuestions();
            }
            else
            {
                questions = await
_questionService.GetQuestionsByTopic(topic);
            }

            if (questions == null || questions.Count == 0)
            {
                return NotFound(); // Return 404 Not Found if no questions
are found for the word
            }

            return Ok(questions);
        }
        catch (Exception ex)
        {
            return StatusCode(500, ex.Message);
        }
    }

    [HttpGet("{id:int}")]
    public async Task<ActionResult<QuestionModel>> GetById(int id)
    {
        try
        {
            var question = await _questionService.GetByIdAsync(id);
            if (question == null)
            {
                return NotFound();
            }

            return Ok(question);
        }
        catch (Exception ex)
        {
            return StatusCode(500, ex.Message);
        }
```

```
    }

    [HttpPost]
    public async Task<ActionResult<QuestionModel>> Create(QuestionModel
model)
    {
        try
        {
            var createdQuestion = await _questionService.CreateAsync(model);
            return CreatedAtAction(nameof(GetById), new { id =
createdQuestion.Id }, createdQuestion);
        }
        catch (Exception ex)
        {
            return StatusCode(500, ex.Message);
        }
    }

    [HttpPatch("{id}")]
    public async Task<IActionResult> Update(int id, QuestionModel model)
    {
        try
        {
            if (id != model.Id)
            {
                return BadRequest();
            }

            var updated = await _questionService.UpdateAsync(model);
            if (updated)
            {
                return NoContent();
            }

            return NotFound();
        }
        catch (Exception ex)
        {
            return StatusCode(500, ex.Message);
        }
    }

    [HttpDelete("{id}")]
    public async Task<IActionResult> Delete(int id)
    {
        try
        {
            var deleted = await _questionService.DeleteAsync(id);
            if (deleted)
            {
                return NoContent();
            }

            return NotFound();
        }
        catch (Exception ex)
        {
            return StatusCode(500, ex.Message);
        }
    }
}
```

Файл QuestionService.cs

```
using AutoMapper;
using Entities;
using Math.BLL.Abstract.Services;
using Math.DAL.Abstract.Repository.Base;
using Models;

namespace Math.BLL.Services;

public class QuestionService : IQuestionService
{
    private readonly IUnitOfWork _unitOfWork;
    protected readonly IMapper _mapper;
    private readonly ITopicService _topicService;

    public QuestionService(IUnitOfWork unitOfWork, IMapper mapper,
ITopicService topicService)
    {
        _unitOfWork = unitOfWork;
        _mapper = mapper;
        _topicService = topicService;
    }

    public async Task<ICollection<QuestionModel>> Get10RandomQuestions()
    {
        int n = 10;
        var allQuestions = await _unitOfWork.QuestionRepository.GetAllAsync(x
=> true);
        int count = allQuestions.Count;
        if (count < n) n = count;
        var tenQuestions = allQuestions.OrderBy(y => Guid.NewGuid()).Take(n);
        var questionModels =
tenQuestions.Select(_mapper.Map<QuestionModel>).ToList();

        return questionModels;
    }

    public async Task<ICollection<QuestionModel>> GetQuestionsByTopic(string
topic)
    {
        var list = await _topicService.GetTopicIdByTopicText(topic);

        int taken = 10;
        var allQuestions = await _unitOfWork.QuestionRepository.GetAllAsync(x
=> list.Contains(x.Topic.Text));
        int count = allQuestions.Count;
        if (count < taken) taken = count;
        var tenQuestions = allQuestions.OrderBy(y =>
Guid.NewGuid()).Take(taken);
        var questionModels =
tenQuestions.Select(_mapper.Map<QuestionModel>).ToList();

        return questionModels;
    }


    public async Task<QuestionModel> CreateAsync(QuestionModel model)
    {
        Question entity = _mapper.Map<Question>(model);
        var newEntity = await
_unitOfWork.QuestionRepository.AddAsync(entity);
        await _unitOfWork.SaveChangesAsync();

        return _mapper.Map<QuestionModel>(newEntity);
    }
```

```
    public async Task<List<QuestionModel>> GetAllAsync()
    {
        var result = await _unitOfWork.QuestionRepository.GetAllAsync(x =>
true);
        var resultMapped =
result.Select(_mapper.Map<QuestionModel>).ToList();

        return resultMapped;
    }

    public async Task<QuestionModel> GetByIdAsync(int id)
    {
        var result = await _unitOfWork.QuestionRepository.GetByIdAsync(id);
        var resultMapped = _mapper.Map<QuestionModel>(result);

        return resultMapped;
    }

    public async Task<bool> UpdateAsync(QuestionModel model)
    {
        if (model == null)
        {
            return false;
        }

        var entity = _mapper.Map<Question>(model);

        var result = await
_unitOfWork.QuestionRepository.UpdateAsync(entity);
        await _unitOfWork.SaveChangesAsync();

        return result;
    }

    public async Task<bool> DeleteAsync(int id)
    {
        var result = await _unitOfWork.QuestionRepository.DeleteAsync(id);
        await _unitOfWork.SaveChangesAsync();

        return result;
    }
}
```

Файл TopicService.cs

```
using System.Text.Json;
using AutoMapper;
using Entities.TopicEntity;
using Math.BLL.Abstract.Services;
using Math.DAL.Abstract.Repository.Base;
using Models.TopicModel;

namespace Math.BLL.Services;

public class TopicService : ITopicService
{
    private readonly IUnitOfWork _unitOfWork;
    private readonly IMapper _mapper;

    private Dictionary<int, List<int>> tree = new();


    public TopicService(IUnitOfWork unitOfWork, IMapper mapper)
```

```
    {
        _unitOfWork = unitOfWork;
        _mapper = mapper;

        tree = LoadTreeFromJson();

        // CreateTree();
    }

    private void CreateTree()
    {
        InitializeTree();
        SaveTreeToJson();
    }
    private async Task InitializeTree()
    {
        var topics = await GetAllAsync();
        foreach (var topic in topics)
        {
            var childrenIds = new List<int>();
            foreach (var child in topic.ChildTopics)
            {
                childrenIds.Add(child.Id);
            }

            tree.Add(topic.Id, childrenIds);
        }
    }

    private void SaveTreeToJson()
    {
        string jsonString = JsonSerializer.Serialize(tree, new
JsonSerializerOptions
        {
            WriteIndented = true
        });

        File.WriteAllText("../tree.json", jsonString);
    }

    private Dictionary<int, List<int>> LoadTreeFromJson()
    {
        string jsonString = File.ReadAllText("../tree.json");

        var options = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true
        };

        return JsonSerializer.Deserialize<Dictionary<int,
List<int>>>(jsonString, options);
    }

    List<int> GetAllTopicIdsRecursive(int id)
    {
        List<int> result = new List<int>();


        if (tree.ContainsKey(id) && tree[id].Count != 0)
        {
            var children = tree[id];

            foreach (var child in children)
            {
```

```
                result.Add(child);
                result.AddRange(GetAllTopicIdsRecursive(child));
            }
        }

        return result;
    }


    public async Task<List<string>> GetTopicIdByTopicText(string text)
    {
        var result = new List<string>();
        var allTopics = await GetAllAsync();
        var topic = allTopics.FirstOrDefault(t => t.Text == text);

        if (topic != null)
        {
            var ids = GetAllTopicIdsRecursive(topic.Id);
            ids.Add(topic.Id);

             result = GetAllAsync().Result.Where(x =>
ids.Contains(x.Id)).Select(x => x.Text).ToList();


        }
        return result;
    }


    public async Task<TopicModel> CreateAsync(TopicModel model)
    {
        Topic entity = _mapper.Map<Topic>(model);
        Topic newEntity = await _unitOfWork.TopicRepository.AddAsync(entity);
        await _unitOfWork.SaveChangesAsync();

        return _mapper.Map<TopicModel>(newEntity);
    }

    public async Task<List<TopicModel>> GetAllAsync()
    {
        var entities = await _unitOfWork.TopicRepository.GetAllAsync(x =>
true);
        var result = entities.Select(_mapper.Map<TopicModel>).ToList();

        return result;
    }

    public async Task<TopicModel> GetByIdAsync(int id)
    {
        var result = _mapper.Map<TopicModel>(await
_unitOfWork.TopicRepository.GetByIdAsync(id));

        return result;
    }

    public async Task<bool> UpdateAsync(TopicModel model)
    {
        if (model == null)
        {
            return false;
        }

        var entity = _mapper.Map<Topic>(model);
```

```
        var result = await _unitOfWork.TopicRepository.UpdateAsync(entity);
        await _unitOfWork.SaveChangesAsync();

        return result;
    }

    public async Task<bool> DeleteAsync(int id)
    {
        var result = await _unitOfWork.TopicRepository.DeleteAsync(id);
        await _unitOfWork.SaveChangesAsync();

        return result;
    }
}
```

Файл MappersProfile.cs

```
using Entities;
using Entities.TopicEntity;
using Models;
using Models.TopicModel;

namespace Math.BLL.Mappers;

public class MappersProfile : AutoMapper.Profile
{
    public MappersProfile()
    {
        CreateMap<Answer, AnswerModel>()
            .ForMember(dest => dest.QuestionModel, opt => opt.MapFrom(src =>
src.Question))
            .PreserveReferences()
            .ReverseMap();

        CreateMap<Question, QuestionModel>()
            .ForMember(dest => dest.AnswerModels, opt => opt.MapFrom(src =>
src.Answers))
            .ForMember(dest => dest.TopicModel, opt => opt.MapFrom(src =>
src.Topic))
            .PreserveReferences()
            .ReverseMap();

        CreateMap<Topic, TopicModel>()
            .ForMember(dest => dest.QuestionModels, opt => opt.MapFrom(src =>
src.Questions))
            .ForMember(dest => dest.BookModels, opt => opt.MapFrom(src =>
src.Books))
            .ForMember(dest => dest.ChildTopics, opt => opt.MapFrom(src =>
src.ChildTopics))
            .ForMember(dest => dest.ParentTopic, opt => opt.MapFrom(src =>
src.ParentTopic))
            .PreserveReferences()
            .ReverseMap();

        CreateMap<Quiz, QuizModel>()
            .ForMember(dest => dest.Topics, opt => opt.MapFrom(src =>
src.Topics))
            .ForMember(dest => dest.ApplicationUser, opt => opt.MapFrom(src
=> src.ApplicationUser))
            .PreserveReferences()
            .ReverseMap();


        CreateMap<Book, BookModel>()
```

```
            .PreserveReferences()
            .ReverseMap();


        CreateMap<TopicForQuiz, TopicForQuizModel>()
            .PreserveReferences()
            .ReverseMap();
    }
}
```

Файл MathContext.cs

```
using Entities;
using Entities.Auth;
using Entities.TopicEntity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace Math.DAL.Context;

public class MathContext : IdentityDbContext
{
    public MathContext(DbContextOptions<MathContext> options) : base(options)
    {
    }

    protected override void OnModelCreating(ModelBuilder builder)
    {
        base.OnModelCreating(builder);

        builder.ApplyConfiguration(new TopicConfiguration());
    }

    public DbSet<Topic> Topics { get; set; }
    public DbSet<Question> Questions { get; set; }
    public DbSet<Answer> Answers { get; set; }

    public DbSet<ApplicationUser> ApplicationUsers { get; set; }
    public DbSet<Quiz> Quizzes { get; set; }
    public DbSet<TopicForQuiz> TopicsForQuizzes { get; set; }
}
```

Файл QuestionRepository.cs

```
using Entities;
using Math.DAL.Abstract.Repository;
using Math.DAL.Context;
using Math.DAL.Repository.Base;
using Microsoft.EntityFrameworkCore;

namespace Math.DAL.Repository;

public class QuestionRepository : GenericRepository<int, Question>,
IQuestionRepository
{
    private readonly MathContext _dbContext;

    public QuestionRepository(MathContext dbContext) : base(dbContext)
    {
        _dbContext = dbContext;
    }

    public override async Task<List<Question>> GetAllAsync(Func<Question,
```

```
bool> predicate)
    {
        List<Question> items = _dbContext.Questions.Include(x =>
x.Answers).Include(x=>x.Topic).ThenInclude(x=>x.Books).Where(predicate).ToLis
t();
        return items;
    }

    public override async Task<Question> GetByIdAsync(int key)
    {
        var item = await _dbContext.Questions.Where(x => x.Id ==
key).Include(x => x.Answers).Include(x => x.Topic).FirstOrDefaultAsync();
        return item;
    }
}
```

Файл GenericRepository.cs

```
using Math.DAL.Abstract.Repository.Base;
using Math.DAL.Context;
using Microsoft.EntityFrameworkCore;

namespace Math.DAL.Repository.Base;

public abstract class GenericRepository<TKey, TEntity> :
IGenericRepository<TKey, TEntity> where TEntity : class
{
    private readonly MathContext _context;

    private DbSet<TEntity> DbSet => _context.Set<TEntity>();

    protected GenericRepository(MathContext dbContext)
    {
        _context = dbContext;
    }

    public async Task<TEntity> AddAsync(TEntity entity)
    {
        var item = await DbSet.AddAsync(entity);
        await _context.SaveChangesAsync();
        return item.Entity;
    }

    public async Task<bool> DeleteAsync(TKey key)
    {
        TEntity item = await DbSet.FindAsync(key);
        if (item == null)
        {
            return false;
        }

        DbSet.Remove(item);
        await _context.SaveChangesAsync();
        return true;
    }

    public virtual async Task<bool> UpdateAsync(TEntity entity)
    {
        if (entity == null)
        {
            return false;
        }
```

```
        DbSet.Attach(entity);
        _context.Entry(entity).State = EntityState.Modified;
        await _context.SaveChangesAsync();
        return true;
    }

    public virtual async Task<List<TEntity>> GetAllAsync(Func<TEntity, bool>
predicate)
    {
        List<TEntity> items = await
Task.FromResult(DbSet.Where(predicate).ToList());
        return items;
    }

    public virtual async Task<TEntity> GetByIdAsync(TKey key)
    {
        TEntity item = await DbSet.FindAsync(key);
        return item;
    }
}
```

Файл Program.cs

```
using System.Text;
using Entities.Auth;
using Entities.Auth.Login;
using Math.BLL;
using Math.DAL;
using Math.DAL.Context;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;

namespace Math.WEB;

internal class Program
{
    public static void Main(string[] args)
    {
        var builder = WebApplication.CreateBuilder(args);

        builder.Services.AddSingleton<IConfiguration>(builder.Configuration);

        // Add secrets configuration
        builder.Configuration.AddUserSecrets<Program>();

        var connectionString =
builder.Configuration.GetConnectionString("DefaultConnection");
        // Configure DbContext with the retrieved connection string



builder.Services.Configure<ApplicationSettings>(builder.Configuration.GetSect
ion("ApplicationSettings"));

        builder.Services.AddDbContext<MathContext>(
            options => options.UseSqlServer(connectionString));

        builder.Services.AddControllers().AddJsonOptions(options =>
        {
            options.JsonSerializerOptions.ReferenceHandler =
```

```
System.Text.Json.Serialization.ReferenceHandler.IgnoreCycles;
        });

        builder.Services.InstallRepositories();
        builder.Services.InstallMappers();
        builder.Services.InstallServices();



        builder.Services.AddAuthentication();
        builder.Services.AddIdentityCore<ApplicationUser>()
            .AddEntityFrameworkStores<MathContext>()
            .AddSignInManager<SignInManager<ApplicationUser>>()
            .AddDefaultTokenProviders();

        builder.Services.Configure<IdentityOptions>(options =>
        {
            options.Password.RequireDigit = false;
            options.Password.RequireNonAlphanumeric = false;
            options.Password.RequireLowercase = false;
            options.Password.RequireUppercase = false;
            options.Password.RequiredLength = 4;
        });

        var client_url =
builder.Configuration["ApplicationSettings:Client_URL"];

        builder.Services.AddCors(options =>
        {
            options.AddPolicy("AllowSpecificOrigin", builder =>
            {
                builder.WithOrigins(client_url)
                    .AllowAnyHeader()
                    .AllowAnyMethod();
            });
        });

        // jwt authentication

        var key =
Encoding.UTF8.GetBytes(builder.Configuration["ApplicationSettings:JWT_Secret"
]);

        builder.Services.AddAuthentication(x =>
        {
            x.DefaultAuthenticateScheme =
JwtBearerDefaults.AuthenticationScheme;
            x.DefaultChallengeScheme =
JwtBearerDefaults.AuthenticationScheme;
            x.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
        }).AddJwtBearer(x =>
        {
            x.RequireHttpsMetadata = false;
            x.SaveToken = false;
            x.TokenValidationParameters = new TokenValidationParameters
            {
                ValidateIssuerSigningKey = true,
                IssuerSigningKey = new SymmetricSecurityKey(key),
                ValidateIssuer = false,
                ValidateAudience = false,
                ClockSkew = TimeSpan.Zero
            };
        });
```

```
        builder.Services.AddControllers();
        builder.Services.AddEndpointsApiExplorer();
        builder.Services.AddSwaggerGen();


        var app = builder.Build();

        if (app.Environment.IsDevelopment())
        {
            app.UseSwagger();
            app.UseSwaggerUI();
        }

        app.UseCors("AllowSpecificOrigin");

        app.UseHttpsRedirection();

        app.UseAuthentication();
        app.UseAuthorization();

        app.MapControllers();

        app.Run();
    }
}
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

Програмне забезпечення для підвищення ефективності роботи математичного тренажера.

Програма та методика тестування

КПІ.ІТ-9303.045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Євгеній ВОВК

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Олексій ПОНОМАРЬОВ

Київ – 2023

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є програмне забезпечення для підвищення ефективності роботи математичного тренажера. Це програмне забезпечення є веб-застосунком з серверною та клієнтською частинами. Воно розроблене під операційні системи Windows, з використання браузеру Google Chrome.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка збереження даних;

– перевірка сумісності веб-додатку з останніми версіями сучасних браузерів Google Chrome, Mozilla Firefox;

– перевірка сумісності застосунку з різними операційними системами (Windows, MacOS. Linux);

– знаходження проблем, помилок і недоліків з метою їх усунення;

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;

– функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;

– мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально та за допомогою написанням unit-тестів, з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

– динамічне тестування на відповідність функціональним вимогам;

– тестування на мобільних пристроях з різною роздільною здатністю екрану;

– тестування на мобільних пристроях з різною версією операційної системи;

– тестування на виведення повідомлень про помилку, коли це необхідно;

– тестування зміни орієнтації екрану;

– тестування інтерфейсу користувача;

– тестування зручності використання;

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОБОТИ МАТЕМАТИЧНОГО ТРЕНАЖЕРА

Керівництво користувача

КПІ.ІТ-9303. 045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Євгеній ВОВК

Нормоконтроль:
_____Максим ГОЛОВЧЕНКО

Виконавець:
_____Олексій ПОНОМАРЬОВ

Київ – 2023

ЗМІСТ

1 Призначення програми

Розробка призначена для покращення роботи математичного тренажеру. Розробка призначена для учнів та студентів навчальних закладів для покращення знань і навичок з математики.

Метою розробки є покращення ефективності математичного тренажеру.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Програмне забезпечення повинно працювати під управлінням операційних систем.

Windows

Windows 10, Windows Server 2016 або більш нові.

Процесор Intel Pentium 4 або більш новий, який підтримує SSE3 інструкції.

Mac

macOS High Sierra 10.13 або більш нові.

Linux

64-bit Ubuntu 18.04+, Debian 10+, openSUSE 15.2+, або Fedora Linux 32+

Процесор Intel Pentium 4 або більш новий, який підтримує SSE3 інструкції.

2.2 Завантаження застосунку

Для запуску застосунку необхідно запустити браузер та перейти на головну сторінку застосунку.

2.3 Перевірка коректної роботи

Для перевірки коректності роботи застосунку потрібно перевірити що головна сторінка застосунку з формою реєстрації та авторизації успішно завантажилась. Також необхідно перевірити що підвантажились теми запитань. Для цього необхідно відкрити dropdown меню кліком по кнопці “Pick Quiz” в лівому верхньому кутку сторінки.

3 ВИКОНАННЯ ПРОГРАМИ

Реєстрація

Робота з застосунком починається з реєстрації. Для реєстрації користувач переходить на головну сторінку і нажимає кнопку “Sign Up” та заповнює всі поля форми та нажимає синю кнопку “Sign Up”. Даний процес показано на рисунку 3.1.

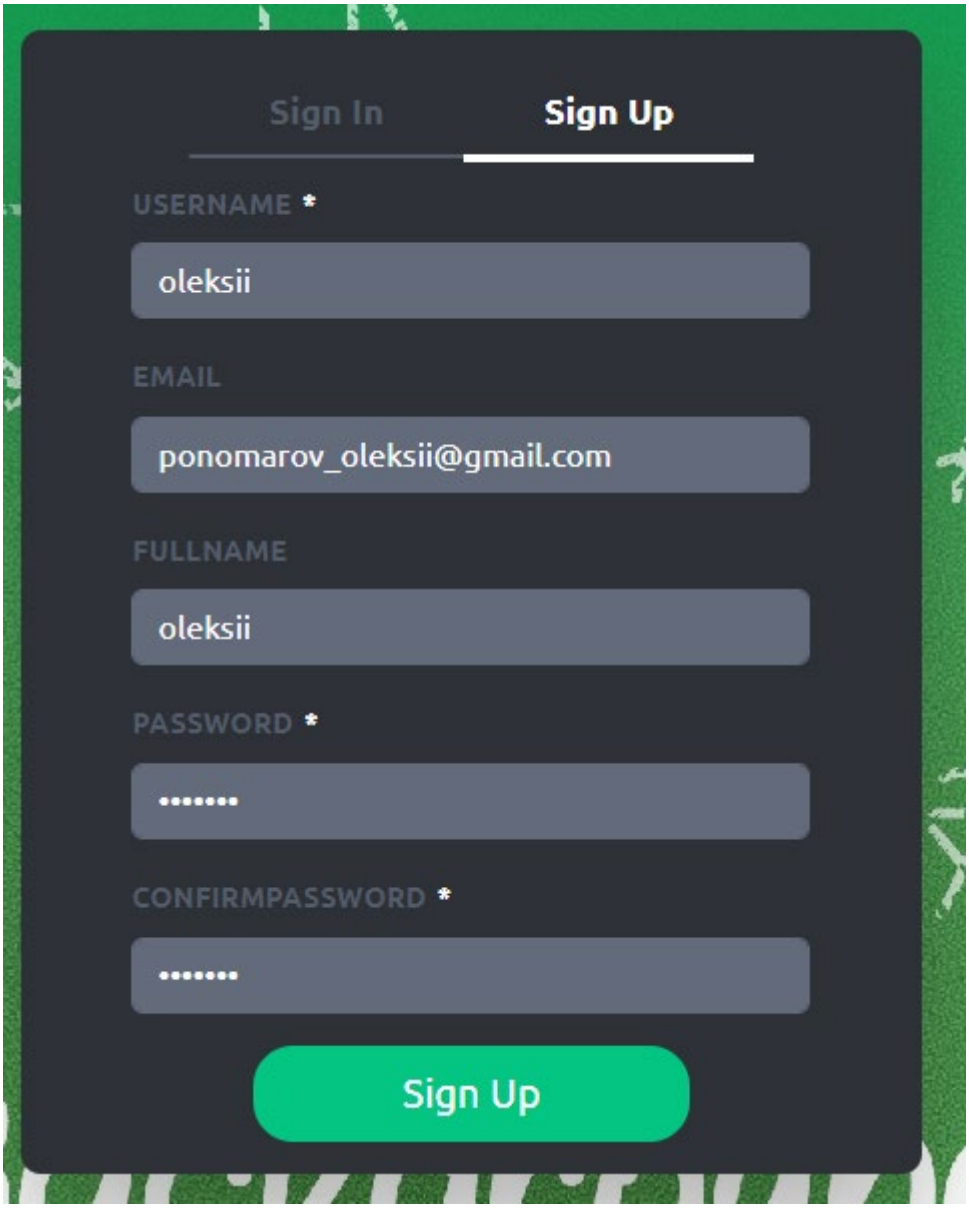


Рисунок 3.1 - Реєстрація

Авторизація

Після успішної реєстрації, користувача перенаправляє на сторінку авторизації. Для цього користувач вводить свій username та password і натискає кнопку “Login”. Це показано на рисунку 3.2.

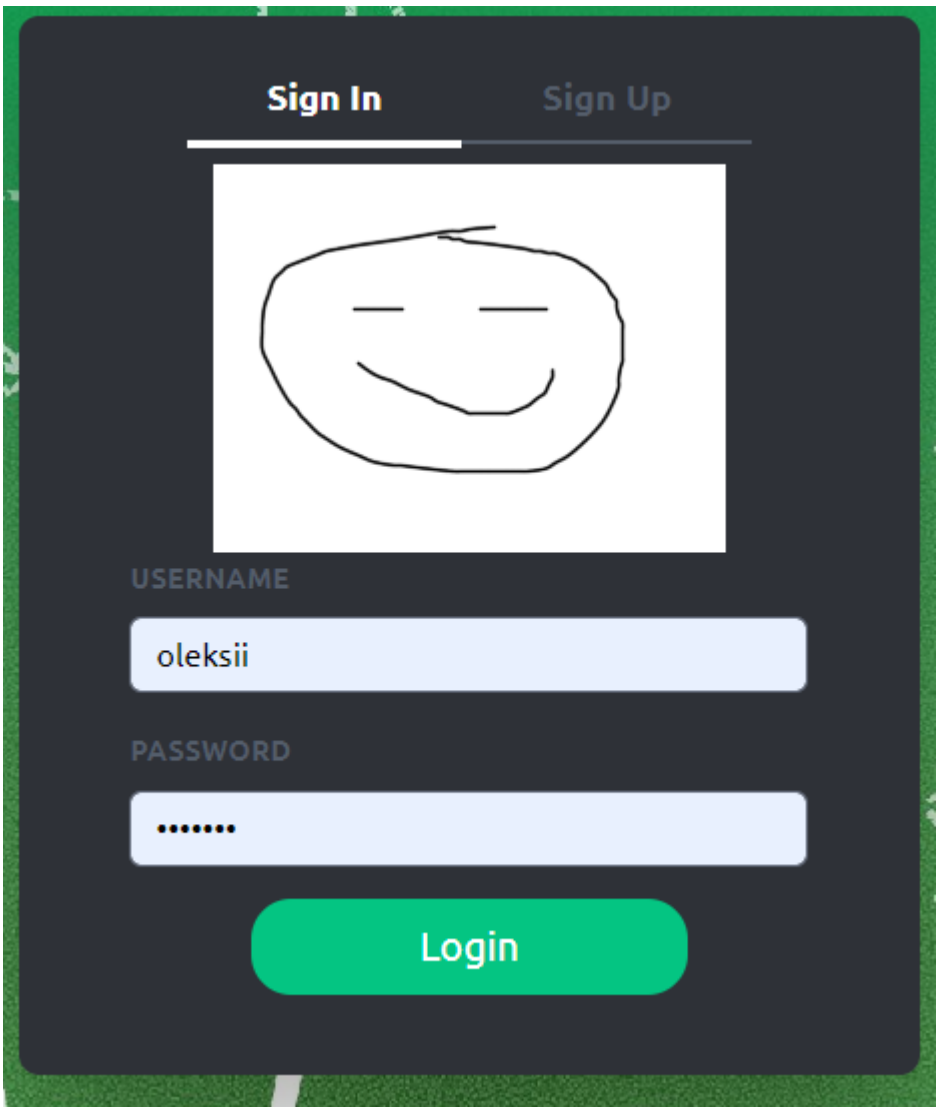


Рисунок 3.2 - Авторизація

Вибір тесту

Для вибору тесту користувач нажимає кнопку “Pick Quiz”, після чого клікає по тесту який бажає пройти. Це показано на рисунку 3.3.

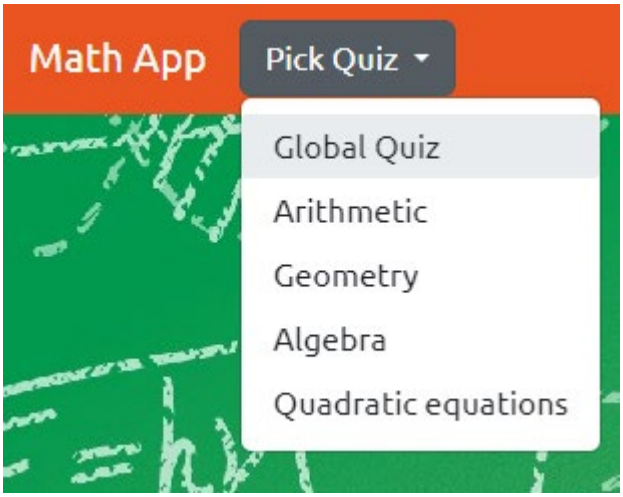


Рисунок 3.3 – Вибір тесту

Проходження тесту

Після вибору тесту завантажаться запитання. Користувачу слід відповісти на всі запитання та натиснути кнопку “Show Results”. Це показано на рисунку 3.4.

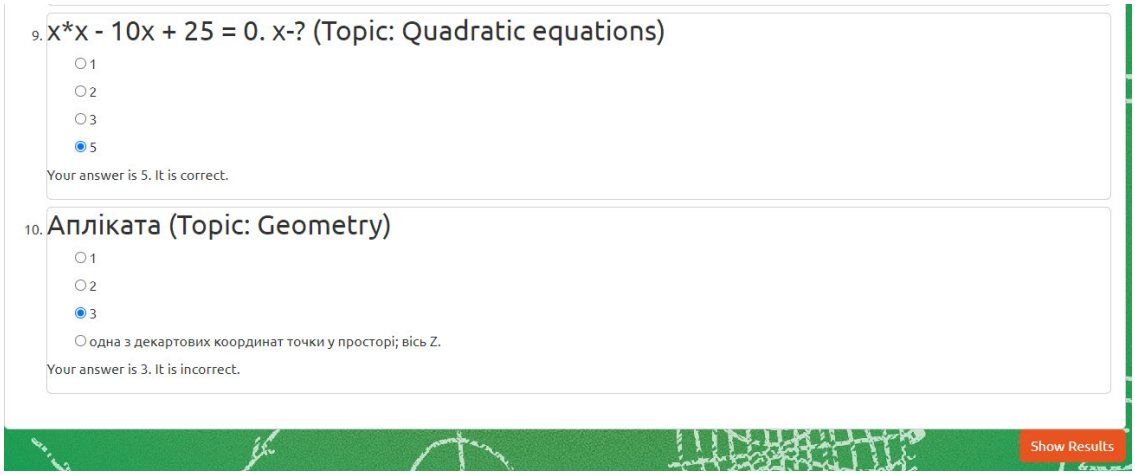


Рисунок 3.4 – Проходження тесту

Отримання рекомендації

Після натискання кнопки “Show Results” користувач побачить кількість правильних варіантів відповідей і отримає рекомендацію щодо відпрацювання допущених помилок. Це показано на рисунку 3.5.

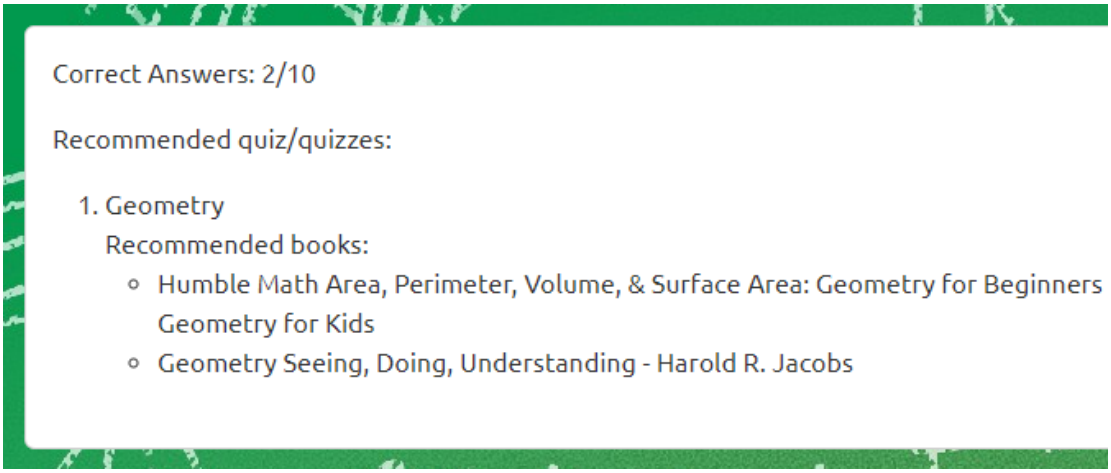


Рисунок 3.5 – Отримання рекомендації

Видалення профілю

Для видалення профілю користувач повинний бути авторизованим в системі. Після цього користувач нажимає жовту кнопку “Delete Profile”, після цього підтверджує свій намір на видалення свого профіля з системи. Це показано на рисунках 3.6 і 3.7.

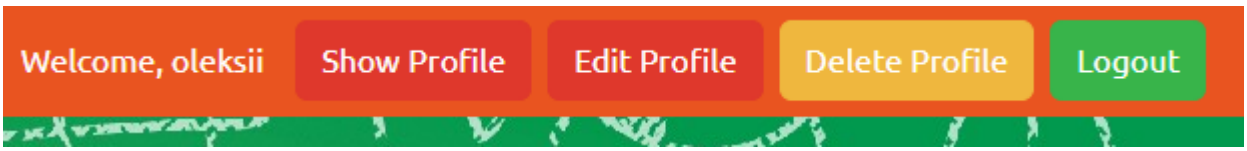


Рисунок 3.6 – Кнопка “Delete Profile”

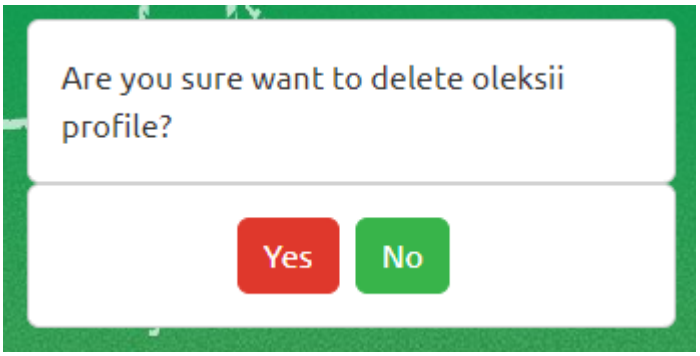


Рисунок 3.7 – Підтвердження видалення профілю

Вихід з системи

Для того щоб вийти з системи користувач нажимає зелену кнопку “Logout” в правому верхньому куті сторінки. Це показано на рисунку 3.8.

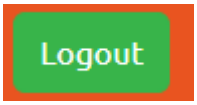


Рисунок 3.8 – Кнопка “Logout”

Перегляд профілю

Користувач може переглянути свій профіль, для цього він нажимає кнопку “Show Profile” в правому верхньому куті сторінки. Це показано на рисунках 3.9 та 3.10.

Рисунок 3.9 – Кнопка “Show Profile”

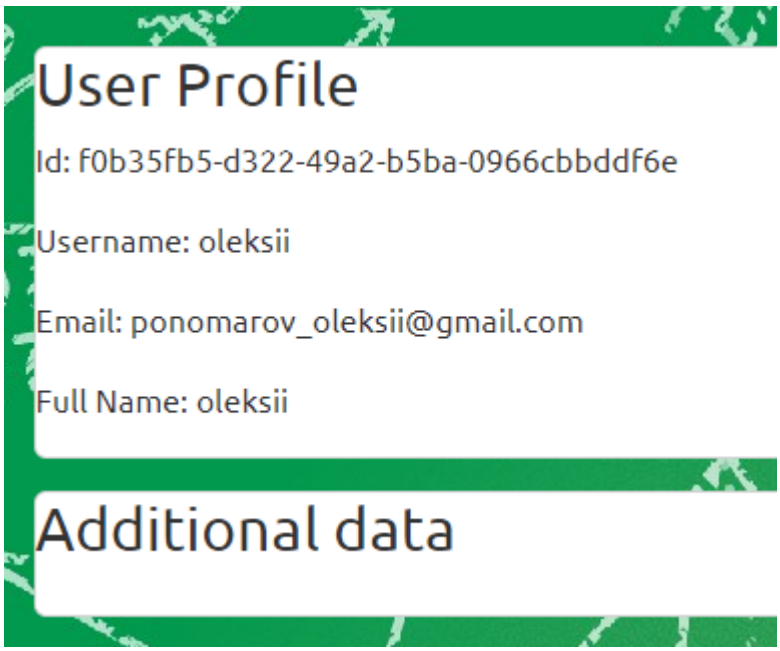


Рисунок 3.10 – Перегляд профілю користувачем

Редагування профілю

Користувач може змінити свій профіль. Для цього він натискає на кнопку “Edit Profile”, вносить нові дані, нажимає кнопку “Save”. Після цього програма покаже користувачу сторінку його профілю з оновленими даними. Це показано на рисунках 3.11 та 3.12.

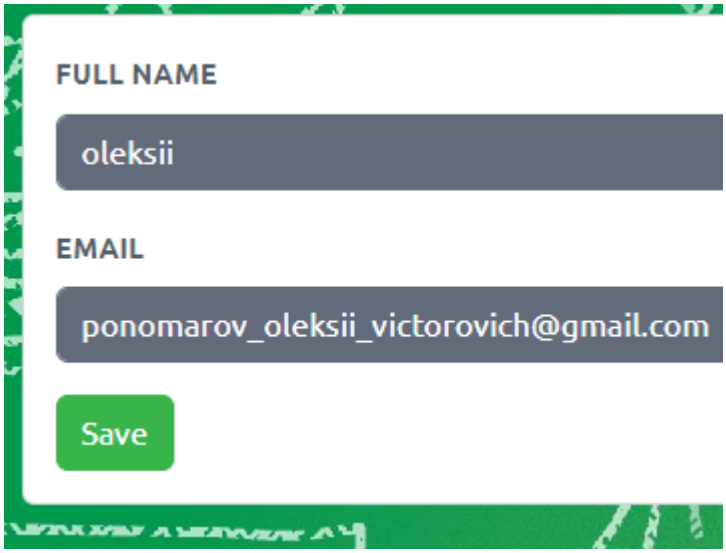


Рисунок 3.10 – Введено нові дані профілю

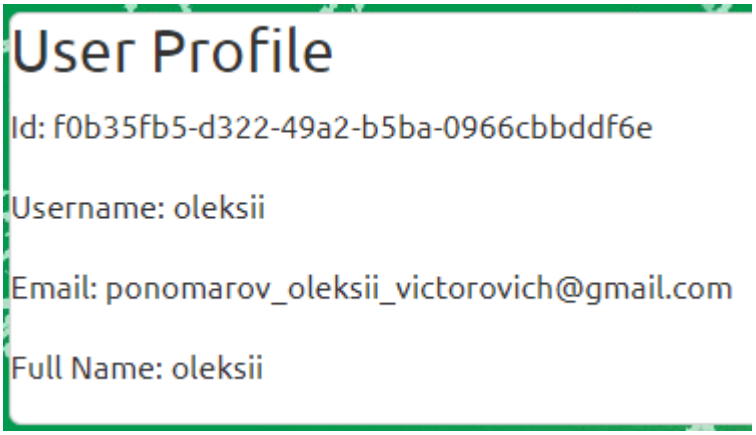


Рисунок 3.11 – Перегляд оновленого профілю

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОБОТИ МАТЕМАТИЧНОГО ТРЕНАЖЕРА

Графічний матеріал

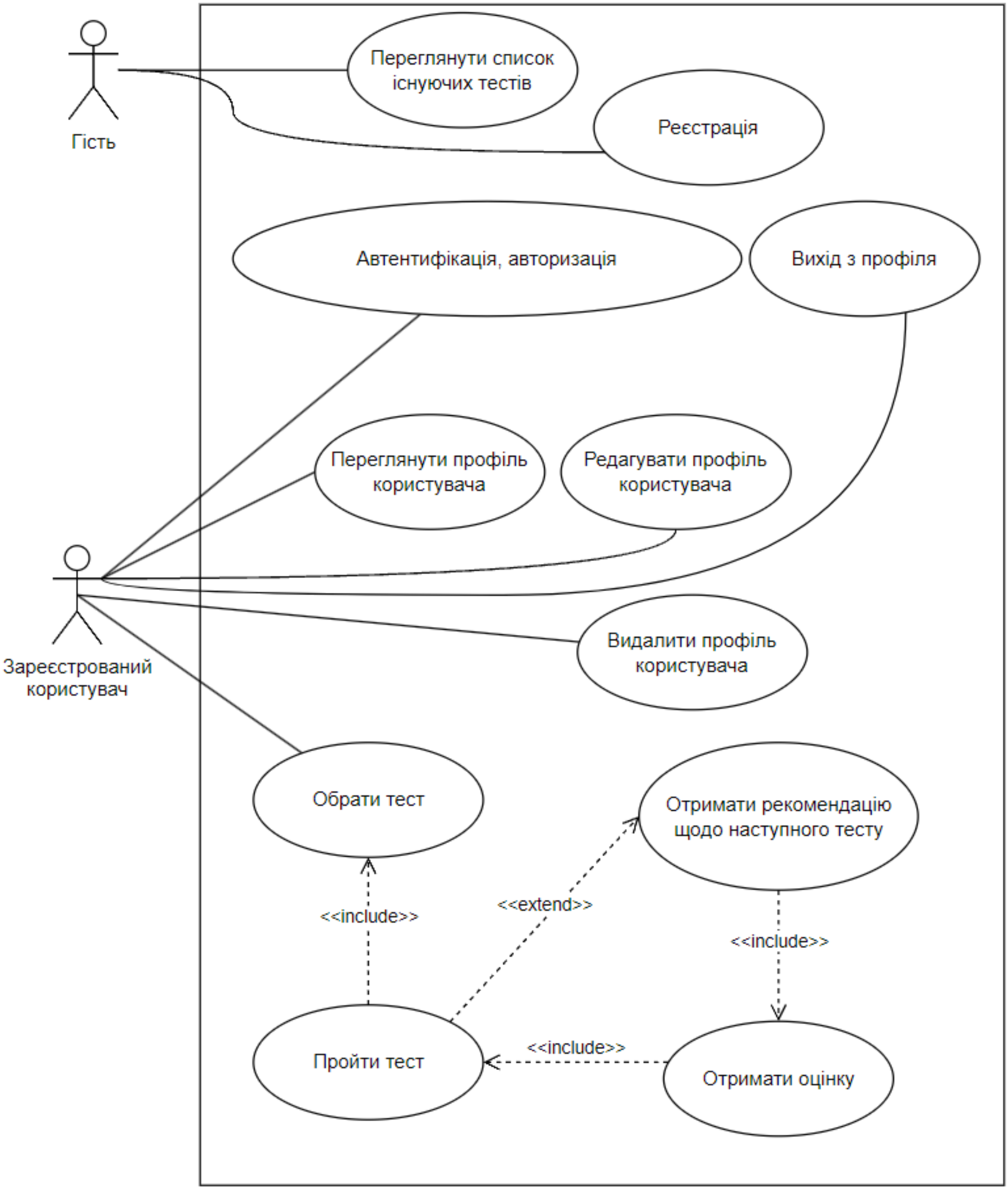
КПІ.ІТ-9303. 045440.06.99

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Євгеній ВОВК

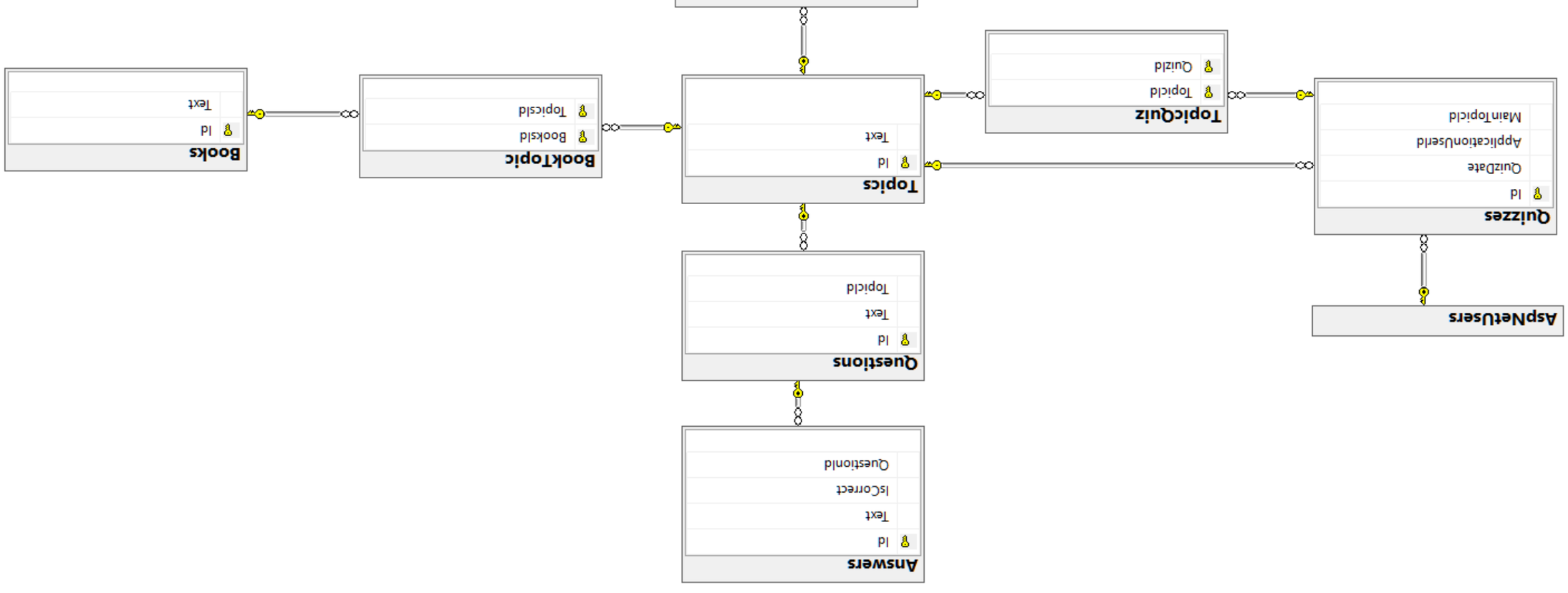
Нормоконтроль:
_____Максим ГОЛОВЧЕНКО

Виконавець:
_____Олексій ПОНОМАРЬОВ

Київ – 2023

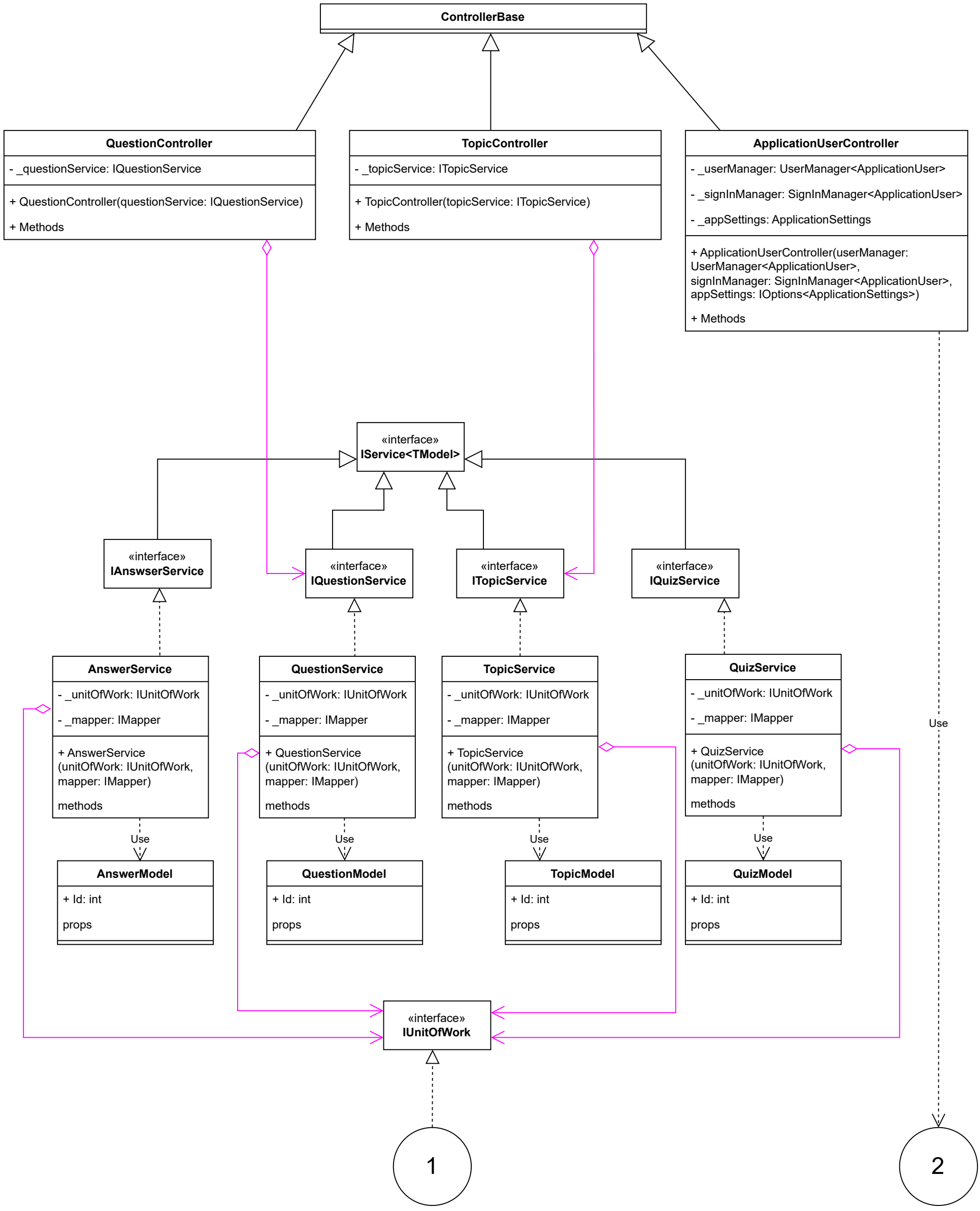


					КПІ.ІТ-9303.045440.06.99.ССВ		
Зм.	Арк.	№ докум.	Підп.	Дата	Схема структурна варіантів використань	Лит.	Арк. / Аркушів: 1
Розроб.		Пономарьов ОВ				Аркуш	Аркушів
Перев.		Вовк Є.А.					
Т. Кон.							
Н. Кон.		Головченко М.М.			Програмне забезпечення для підвищення ефективності роботи математичного тренажера	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-93	
Затв.		Жаріков Е.В.					



Зм.	Арк.	№ документа	Підпис	Дата	КПІ.ІТ-9303.045440.07.99.СБД		
Розробив		Пономарьов О.В.			Схема бази даних	Літера	Арк.	Аркушів
Перевірив		Боєк Є.А.						
Т. кон.						Аркуш	Аркушів
Н. кон.		Голоченко М.М.			Програмне забезпечення для підвищення ефективності роботи математичного прискорювача	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-93	
Затвердив		Жарікова Е.В.					

КПІ.ІТ-9303.045440.08.99.ССК



					КПІ.ІТ-9303.045440.08.99.ССК		
						Лит.	Арк. / Аркушів
Зм.	Арк.	№ докум.	Підп.	Дата	Схема структурна класів програмного забезпечення		1 / 2
Розроб.		Пономарьов ОВ.				Аркуш	Аркушів
Перев.		Вовк Є.А.			Програмне забезпечення для підвищення ефективності роботи математичного тренажера	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-93	
Т. Кон.							
Н. Кон.		Головченко М.М.					
Затв.		Жаріков Е.В.					

1

2

UnitOfWork

- _context: MathContext

+ AnswerRepository: IAnswerRepository

+ QuestionRepository: IQuestionRepository

+ TopicRepository: ITopicRepository

+ QuizRepository: IQuizRepository

+ UnitOfWork(context,: MathContext, questionRepository: IQuestionRepository, answerRepository: IAnswerRepository, topicRepository: ITopicRepository, quizRepository: IQuizRepository)

+ SaveChangesAsync(): Task

«interface»
IGenericRepository<TKey, TEntity>

«interface»
IQuestionRepository<int, Question>

«interface»
IAnswerRepository<int, Answer>

GenericRepository<TKey, TEntity>

- _context: MathContext

- DbSet: DbSet<TEntity>

+ GenericRepository(dbContext): MathContext

+ AddAsync(entity: TEntity) : Task<TEntity>

+ DeleteAsync(key: Tkey): Task<bool>

+ UpdateAsync(entity: TEntity): Task<bool>

+ GetAllAsync(predicate: Func<TEntity, bool>): Task<List<TEntity>>

+ GetByIdAsunc(key: TKey): Task<TEntity>

«interface»
ITopicRepository<int, Topic>

«interface»
IQuizRepository<int, Quiz>

Use

QuestionRepository

- _dbContext: MathContext

+ QuestionRepository (dbContext: MathContext)

methods

AnswerRepository

- _dbContext: MathContext

+ QuestionRepository (dbContext: MathContext)

methods

TopicRepository

- _dbContext: MathContext

+ QuestionRepository (dbContext: MathContext)

methods

QuizRepository

- _dbContext: MathContext

+ QuizRepository (dbContext: MathContext)

methods

IdentityUser

+ Id: Guid

props

Use

Use

Use

Use

Use

Question

+ Id: int

props

Answer

+ Id: int

props

MathContext

props

Topic

+ Id: int

props

Quiz

+ Id: int

props

ApplicationUser

props

Use

Use

Use

Use

					КПІ.ІТ-9303.045440.08.99.ССК	
Зм.	Арк.	№ документа	Підпис	Дата		
Розробив		*Пономарьов ОВ.*			*Схема структурна класів програмного забезпечення*	Літера: — / Арк.: 2 / Аркушів: 2
Перевірив		*Вовк Є.А.*				Аркуш / Аркушів
Т. кон.						
Н. кон.		*Головченко М.М.*			*Програмне забезпечення для підвищення ефективності роботи математичного тренажера*	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-93
Затвердив		*Жаріков Е.В.*				