МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії програмного забезпечення, канд. техн. наук, доцент,

_____Є.О. Давиденко

«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

ДОСЛІДЖЕННЯ ШЛЯХІВ ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ ТА ЕФЕКТИВНОСТІ АВТОМАТИЗОВАНИХ МЕТОДІВ ОЦІНКИ ЗНАНЬ

Спеціальність «Інженерія програмного забезпечення»

121 – КРМ – 608м.21920801

Здобувач _____М. С. Кіяшко

підпис

«___» _____ 2024 р.

Керівник канд. техн. наук, доцент _____Є. О. Давиденко

підпис

«___» _____ 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри інженерії програмного

забезпечення, канд.техн.наук, доцент,

_____Є.О. Давиденко

«____»_____2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра

Видано студенту групи 608м факультету комп'ютерних наук

Кіяшко Максиму Сергійовичу

.

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

«Дослідження шляхів підвищення продуктивності та ефективності автоматизованих методів оцінки знань»

Затверджена наказом по ЧНУ від «10» листопада 2023 р. № 234

2. Строк представлення кваліфікаційної роботи «26» лютого 2024 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Вхідні дані до роботи − функціональні та нефункціональні вимоги до програмного забезпечення автоматизації тестування знань. Результат − функціонуюча система автоматизації тестування знань.

4. Перелік питань, що підлягають розробці:

- аналіз принципів та підходів до тестування знань, тобто дослідження предметної області;
- огляд існуючих систем тестування знань;

– розробка вимог до системи на основі зібраної інформації про предметну область та аналоги;
– підвищення ефективності за допомогою інтеграції аналітики та ШІ;
– проєктування автоматизованої системи тестування знань;
– програмна реалізація застосунку для тестування знань.

5. Перелік графічних матеріалів:

Презентація

.

Керівник роботи канд. техн. наук, доцент, Є. О. Давиденко
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

Кіяшко Максим Сергійович

(прізвище, ім'я, по батькові студента)

_____
(підпис)

Дата видачі завдання «____» _____ 2023 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: «Дослідження шляхів підвищення продуктивності та ефективності автоматизованих методів оцінки знань».

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання ККРБ	25.10.2023р.	26.10.2023р.	виконано
2.	Складання календарного плану ККРБ	26.10.2023р.	27.10.2023р.	виконано
3.	Огляд літератури за темою ККРБ	27.10.2021р.	15.11.2023р.	виконано
4.	Аналіз предметної області	15.11.2023р.	25.11.2023р.	виконано
5.	Розробка проєктних рішень	25.11.2023р.	30.11.2023р.	виконано
6.	Моделювання та конструювання ПЗ	30.11.2023р.	15.12.2023р.	виконано
7.	Кодування, тестування та апробація розробленого ПЗ	15.12.2023р.	12.01.2024р.	виконано
8.	Написання спеціальної частини з охорони праці	12.01.2024р.	15.01.2024р.	виконано
9.	Відгук керівника ККРБ	15.01.2024р.	17.01.2024р.	виконано
10.	Оформлення ККРБ та презентації	17.01.2024р.	30.01.2024р.	виконано
11.	Попередній захист	08.02.2024р.	08.02.2024р.	виконано
12.	Завершення оформлення ККРБ та презентації	08.02.2024р.	20.02.2024р.	виконано
13.	Рецензування	22.02.2024р.	26.02.2024р.	виконано
14.	Захист кваліфікаційної роботи	26.02.2024р.	26.02.2024р.	виконано

Розробив студент Кіяшко Максим Сергійович _____
(прізвище, ім'я, по батькові студента) (*підпис*)

«____» _____ 2023 р.

Керівник роботи канд. техн. наук, доцент, Давиденко Є. О. _____
(посада, прізвище, ім'я, по батькові) (*підпис*)

«____» _____ 2023 р.

АНОТАЦІЯ

до кваліфікаційної роботи магістра

«Дослідження шляхів підвищення продуктивності та ефективності автоматизованих методів оцінки знань»

Студент 608м гр.: Кіяшко Максим Сергійович

Керівник: зав. кафедри ІПЗ, канд. техн. наук, доцент Давиденко Є. О.

Дана кваліфікаційна робота присвячена розробці комплексної системи для автоматизації процесу тестування знань в освітніх закладах. В контексті сучасних освітніх викликів, зокрема необхідності дистанційного навчання та індивідуалізації освітнього процесу, автоматизація тестування відіграє ключову роль у підвищенні ефективності та об'єктивності оцінювання знань студентів.

Об’єкт: процеси підготовки, проведення та аналізу результатів тестувань в освітніх установах, зокрема в університетах та коледжах, де існує висока потреба в об’єктивному та ефективному оцінюванні знань студентів.

Предмет: алгоритми штучного інтелекту, зокрема технології машинного навчання та обробки природної мови, які застосовуються для створення та оптимізації тестових завдань, а також методики аналізу даних, що дозволяють оцінювати ефективність тестів та адаптувати навчальний процес.

Метою дослідження є створення інтегрованої системи автоматизованого тестування, яка використовує алгоритми штучного інтелекту для оптимізації процесу підготовки тестових завдань та аналізу їх ефективності. Система повинна надавати викладачам інструменти для гнучкого створення тестів, а також забезпечувати глибокий аналіз відповідей студентів для виявлення слабких місць у навчальному матеріалі та методиках викладання.

У вступі викладається актульність теми, мета та поставлені **завдання** щодо реалізації застосунку.

В першому розділі проводиться аналітичний огляд і обгрунтування теоретичних основ автоматизації оцінки знань. Описано предметну область, проведена класифікація тестових завдань. Також наданий огляд існуючих систем автоматизації тестування, таких як Moodle, Blackboard, Canvas і Google Forms.

Розглянуто роль штучного інтелекту в процесі автоматизації тестування та методики підвищення ефективності тестування за допомогою аналітики. Завершується розділ специфікацією вимог до системи.

У другому розділі представлено проєктування та моделювання системи автоматизації оцінки знань. Включено опис варіантів використання системи, сценаріїв її використання, діаграму станів, діаграму послідовності, концептуальну модель, діаграму розгортання, а також схему бази даних.

Третій розділ зосереджено на архітектурі програмного забезпечення системи. Обговорюється вибір програмних засобів розробки, включаючи клієнт-серверну архітектуру та її роль у системах оцінки знань. Детально описано програмну реалізацію системи, включаючи ASP.NET Web API, Loosely Coupled Monolith, Entity Framework – ORM та міграції бази даних, а також фреймворк Angular. Також включено принципи та методики аналізу ефективності тестів та інтеграцію штучного інтелекту для оптимізації процесу створення тестів.

Четвертий розділ присвячено огляду інтерфейсу та посібнику користувача системи. Представлено макети інтерфейсу, включаючи авторизацію, панель навігації, головну панель, панель створення тесту та статистики, розділ налаштувань, сторінку персональних тестів та результатів, а також сторінку глобального рейтингу. Також описано аналітику та допомогу користувачу засобами штучного інтелекту, включаючи сторінку ефективності тесту та рекомендації ШІ при проєктуванні тесту.У висновках проводиться аналіз роботи та отриманих результатів.

КРМ викладена на 108 сторінок, містить 4 розділи, 38 ілюстрацій, 2 таблиці, 14 джерел в переліку посилань.

Ключові слова: *автоматизація тестування, аналітика результатів, штучний інтелект у освіті, розробка програмного забезпечення.*

ABSTRACT

of the Master's qualification work

"Researching ways to increase the productivity and efficiency of automated knowledge assessment methods"

Student of group 608м: Kiiashko Maksym Serhiiovych

Supervisor: Deputy Dean, Associate Professor of the Department of software engineering Davydenko Ye. O.

This thesis project is dedicated to the development of a comprehensive system for automating the knowledge testing process in educational institutions. In the context of modern educational challenges, in particular the need for distance learning and individualization of the educational process, test automation plays a key role in improving the efficiency and objectivity of student assessment.

Object: the processes of preparing, conducting and analyzing test results in educational institutions, including universities and colleges, where there is a high demand for objective and efficient assessment of students' knowledge.

Subject: artificial intelligence algorithms, including machine learning and natural language processing technologies used to create and optimize test tasks, as well as data analysis techniques that allow to evaluate the effectiveness of tests and adapt the learning process.

The **aim** of the study is to create an integrated automated testing system that uses artificial intelligence algorithms to optimize the process of preparing test tasks and analyzing their effectiveness. The system should provide teachers with tools for flexible test creation, as well as provide in-depth analysis of student responses to identify weaknesses in educational material and teaching methods.

The introduction outlines the relevance of the topic, the objective, and the tasks set for the application's implementation. The first section provides an analytical review and justification of the theoretical foundations for automating knowledge assessment. The subject area is described, and a classification of test tasks is provided. An overview of existing testing automation systems such as Moodle, Blackboard, Canvas, and Google Forms is also given. The role of artificial intelligence in the automation of

testing and methods for increasing testing efficiency through analytics are discussed. The section concludes with a specification of system requirements.

The second section presents the design and modeling of the knowledge assessment automation system. It includes a description of the system's use cases, usage scenarios, state diagram, sequence diagram, conceptual model, deployment diagram, and database schema.

The third section focuses on the software architecture of the system. It discusses the choice of software development tools, including the client-server architecture and its role in knowledge assessment systems. The software implementation of the system is detailed, including ASP.NET Web API, Loosely Coupled Monolith, Entity Framework - ORM, and database migrations, as well as the Angular framework. Also included are principles and techniques for analyzing test effectiveness and integrating artificial intelligence to optimize the test creation process.

The fourth section is dedicated to reviewing the system's interface and user manual. It presents interface layouts, including authorization, navigation panel, main panel, test creation panel and statistics, settings section, personal tests and results page, and the global ranking page. It also describes analytics and AI-assisted user support, including the test efficiency page and AI recommendations in test design. The conclusions analyze the work and results obtained. The thesis is presented on 108 pages, contains 4 sections, 38 illustrations, 2 tables, and 14 sources in the reference list.

Keywords: *testing automation, results analytics, artificial intelligence in education, software development*.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

GUI − графічний інтерфейс користувача (graphical user interface)

NF – Normal form (нормальна форма)

NLP – Natural language processing

БД – база даних

ПЗ – програмне забезпечення

ШІ – штучний інтелект

ВСТУП

У сучасному освітньому просторі, де цифровізація та дистанційне навчання стають новою нормою, актуальність розробки ефективних систем автоматизованого тестування знань студентів набуває особливої ваги. Автоматизовані системи тестування відкривають нові можливості для оцінювання великої кількості студентів одночасно, забезпечуючи швидкість та об'єктивність процесу. Проте, існуючі системи часто зосереджені на кількісних показниках, ігноруючи глибину аналізу та індивідуальний підхід до кожного студента.

Сучасні дослідження в галузі освіти підкреслюють необхідність розвитку адаптивних систем тестування, які можуть враховувати індивідуальні особливості студентів та забезпечувати диференційований підхід до оцінювання. Це створює потребу в інтеграції інтелектуальних алгоритмів, здатних аналізувати відповіді студентів, виявляти закономірності та надавати рекомендації щодо покращення навчальних матеріалів і методик.

Важливість таких інновацій також підкріплюється зростаючими вимогами ринку праці до якості освіти. Роботодавці очікують, що випускники вищих навчальних закладів будуть не тільки володіти фундаментальними знаннями, але й мати здатність критично мислити, аналізувати та застосовувати отримані знання в практичних ситуаціях. Таким чином, розробка системи, яка може глибоко аналізувати ефективність тестів та сприяти розвитку вищезазначених навичок, є вкрай актуальною.

Окрім того, пандемія COVID–19 внесла свої корективи в освітні процеси, змусивши навчальні заклади шукати нові підходи до дистанційного навчання та оцінювання. В цьому контексті, системи автоматизованого тестування, які можуть забезпечити високу якість оцінювання та адаптивність до різних умов навчання, стають не просто зручним інструментом, а необхідністю.

Таким чином, розробка інтегрованої системи, яка поєднує в собі передові технології штучного інтелекту для оптимізації процесу створення та аналізу тестів, є важливим кроком у вдосконаленні освітнього процесу та підвищенні його ефективності.

Об’єктом дослідження є процеси підготовки, проведення та аналізу результатів тестувань в освітніх установах, зокрема в університетах та коледжах, де існує висока потреба в об’єктивному та ефективному оцінюванні знань студентів.

Предметом дослідження є алгоритми штучного інтелекту, зокрема технології машинного навчання та обробки природної мови, які застосовуються для створення та оптимізації тестових завдань, а також методики аналізу даних, що дозволяють оцінювати ефективність тестів та адаптувати навчальний процес.

Метою дослідження є створення інтегрованої системи автоматизованого тестування, яка використовує алгоритми штучного інтелекту для оптимізації процесу підготовки тестових завдань та аналізу їх ефективності. Система повинна надавати викладачам інструменти для гнучкого створення тестів, а також забезпечувати глибокий аналіз відповідей студентів для виявлення слабких місць у навчальному матеріалі та методиках викладання.

Для досягнення поставленої мети необхідне вирішення наступних **завдань**:

1. Аналіз існуючих методів тестування: вивчити і класифікувати різноманітні підходи до тестування, визначити сильні та слабкі сторони сучасних автоматизованих систем тестування, що дозволить закласти основу для розробки вдосконаленої системи.

2. Визначення вимог до системи: опрацювати інформацію щодо специфіки предметної області для формування вимог до функціоналу, інтерфейсу та технічних характеристик майбутньої системи.

3. Моделювання системи: розробити UML-діаграми для візуалізації структури, процесів та взаємодій в системі, що допоможе чітко структурувати процес розробки.

4. Розробка програмного забезпечення: створення як серверної (backend), так і клієнтської (frontend) частин системи, забезпечуючи їх взаємодію та функціональність відповідно до встановлених вимог.

5. Тестування системи: перевірка роботи системи на відповідність вимогам, виявлення та усунення помилок, що гарантуватиме надійність і ефективність системи.

6. Розробка посібника користувача: створення детальної інструкції для користувачів системи, що сприятиме її зручності та інтуїтивному використанню.

7. Інтеграція модуля аналітики: впровадження інструментів для аналізу ефективності тестувань, зокрема, за допомогою алгоритмів обробки даних та штучного інтелекту, що дозволить отримувати зворотній зв'язок та вдосконалювати навчальний процес.

8. Реалізація модуля зі штучним інтелектом: впровадження алгоритмів штучного інтелекту для оптимізації процесу створення тестів, аналізу відповідей та адаптації тестів під індивідуальні особливості користувачів.

Розроблювана система буде включати модуль аналітики, який зможе оцінювати ефективність кожного тесту, використовуючи комплексні метрики. Ці метрики включатимуть не тільки кількісні показники успішності студентів, але й якісні характеристики питань, такі як їх релевантність, валідність, диференціацію та здатність виявляти глибоке розуміння матеріалу.

Для підвищення якості тестів, система буде використовувати алгоритми обробки природної мови для генерації альтернативних формулювань питань, що дозволить викладачам вибирати найбільш точні та ефективні варіанти. Також буде реалізована можливість адаптивного тестування, коли система

автоматично коригує складність питань залежно від рівня підготовки студента, що дозволить досягти більш точної оцінки знань.

Інноваційність даної роботи полягає в інтеграції передових технологій штучного інтелекту в процес тестування, що не тільки підвищить ефективність оцінювання, але й сприятиме розвитку освітніх методик. Використання аналітичних інструментів для оцінки ефективності тестів дозволить викладачам оперативно адаптувати навчальні матеріали та методики, підвищуючи якість освіти.

1 АНАЛІТИЧНИЙ ОГЛЯД ТА ТЕОРЕТИЧНІ ОСНОВИ АВТОМАТИЗАЦІЇ ОЦІНКИ ЗНАНЬ

1.1 Опис предметної області

Сучасний підхід до тестування знань вимагає розуміння його сутності, структури та різноманіття форм. Актуальність теми посилюється швидким розвитком технологій та необхідністю оптимізації навчального процесу. Традиційно, методи оцінювання знань включають:

– усні інтерв'ю: оцінюють рівень підготовки та розуміння матеріалу через безпосередній діалог з викладачем. Якість такого тестування залежить від об'єктивності та комплексності заданих питань;

– письмові контрольні: перевіряють здатність студентів застосовувати теоретичні знання на практиці. Цей метод вважається більш точним, але вимагає значних часових затрат на організацію;

– лабораторні завдання: тестують практичні навички та вміння застосовувати теоретичні засади для вирішення специфічних завдань.

Традиційне тестування часто вимагає значних зусиль для підготовки, проведення та перевірки, що не завжди ефективно з точки зору часу та ресурсів. Автоматизація процесу тестування може мінімізувати ризик помилок, які часто виникають під час монотонної ручної роботи, та сприяти об'єктивності оцінювання.

Автоматизоване тестування відкриває широкі можливості для підвищення ефективності оцінювання знань, зниження навантаження на викладачів та забезпечення об'єктивності результатів. Особливо це стало актуальним у контексті пандемії COVID–19, яка кардинально змінила підходи до організації навчального процесу, зробивши дистанційне навчання нормою [1].

Традиційний процес тестування включає декілька етапів:

1. Підготовка: викладачі розробляють тести для різних цілей – від модульних контролів до державних іспитів.

2. Проведення: студенти отримують завдання у паперовому форматі та інструкції щодо їх виконання.

3. Оцінювання: відповіді перевіряються вручну за допомогою шаблонів чи еталонів, що є трудомістким та схильним до помилок.

4. Аналіз результатів: формується звіт з оцінками, який використовується для виставлення кінцевих оцінок.

Впровадження автоматизованого тестування сприяє:

– ефективності: Значно скорочує час, необхідний для підготовки та оцінювання тестів;

– об'єктивності: Мінімізує вплив людського фактора на результати оцінювання;

– гнучкості: дозволяє швидко адаптувати тести під різні потреби та умови навчання.

Таблиця 1.1 – Порівняння традиційного та автоматизованого тестування

Критерій	Традиційне Тестування	Автоматизоване Тестування
Час Підготовки	Високий	Низький
Ризик Помилок	Високий	Низький
Об'єктивність	Залежить від екзаменатора	Висока
Вартість	Змінна, залежить від масштабу	Знижується з часом
Гнучкість	Обмежена	Висока

Автоматизація процесу тестування знань є стратегічним вибором для підвищення ефективності та якості освітнього процесу. Вона не лише забезпечує економію часу та ресурсів, але й сприяє більшій об'єктивності оцінювання, відкриваючи нові можливості для гнучкого та інтерактивного навчання в умовах постійно змінюваного освітнього середовища.

1.2 Класифікація тестових завдань

Тестові завдання можна умовно поділити на два основні види: стандартні та інтерактивні.

1. Стандартні тести: включають фіксований набір запитань, де кожна правильна відповідь приносить визначену кількість балів. Оцінювання здійснюється на основі сукупності правильних відповідей, ваги кожного питання та їх загальної кількості. Цей тип тестування часто застосовується в освітніх установах для оцінки засвоєння матеріалу студентами.

2. Інтерактивні тести: розпочинаються з питань базового або середнього рівня складності, але адаптуються до відповідей учасника. У разі правильної відповіді наступне питання буде складнішим, у разі помилки – простішим. Цей підхід дозволяє точніше визначити рівень знань користувача. Адаптивні тести зокрема ефективні при визначенні рівня володіння іноземними мовами.

Тестові завдання за типом відповідей можна класифікувати на:

– відкриті завдання: вимагають від учасника самостійно сформулювати відповідь. У контексті автоматизованих тестових систем ці завдання реалізуються через текстові поля для введення відповідей короткої або довшої форми;

– закриті завдання: можуть включати різноманітні підтипи, такі як (рисунок 1.1):

– одиночний вибір: вимагає вибрати одну правильну відповідь з декількох варіантів, часто реалізується за допомогою радіо–кнопок або випадаючих списків;

– множинний вибір: дозволяє вибрати кілька правильних відповідей з наданих варіантів, зазвичай використовуються прапорці (чекбокси);

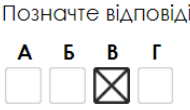

Рисунок 1.1 – Одиночний (множинний) вибір

– встановлення відповідності: завдання, де потрібно зіставити елементи з двох списків (рисунок 1.2);

– послідовність: вимагає розташувати елементи у вірному порядку.

Рисунок 1.2 – Встановлення послідовності

Сучасні технології дозволяють збагатити тестові завдання мультимедійним контентом, що робить процес навчання більш залучаючим та ефективним. Використання зображень, аудіо та відеоматеріалів покращує засвоєння інформації та робить освітній процес більш динамічним і зрозумілим.

Методика підрахунку результатів тестування залежить від його типу та мети. Освітні тести зазвичай фокусуються на кількісній оцінці знань, тоді як психологічні та особистісні тести часто надають кваліативну оцінку, наприклад, визначаючи психологічний тип або схильності індивіда.

Таким чином, автоматизація процесу тестування відкриває нові горизонти для ефективної та об'єктивної оцінки знань, а також забезпечує зручність та адаптивність навчального процесу в умовах швидкого розвитку освітніх технологій.

Останнім часом у сфері автоматизації тестування спостерігається зростаючий інтерес до впровадження технологій штучного інтелекту для створення адаптивних тестів, що здатні налаштовуватися під потреби кожного студента. Крім того, посилюється акцент на мобільних застосунках для тестування, що робить процес більш зручним і доступним (рисунок 1.3).

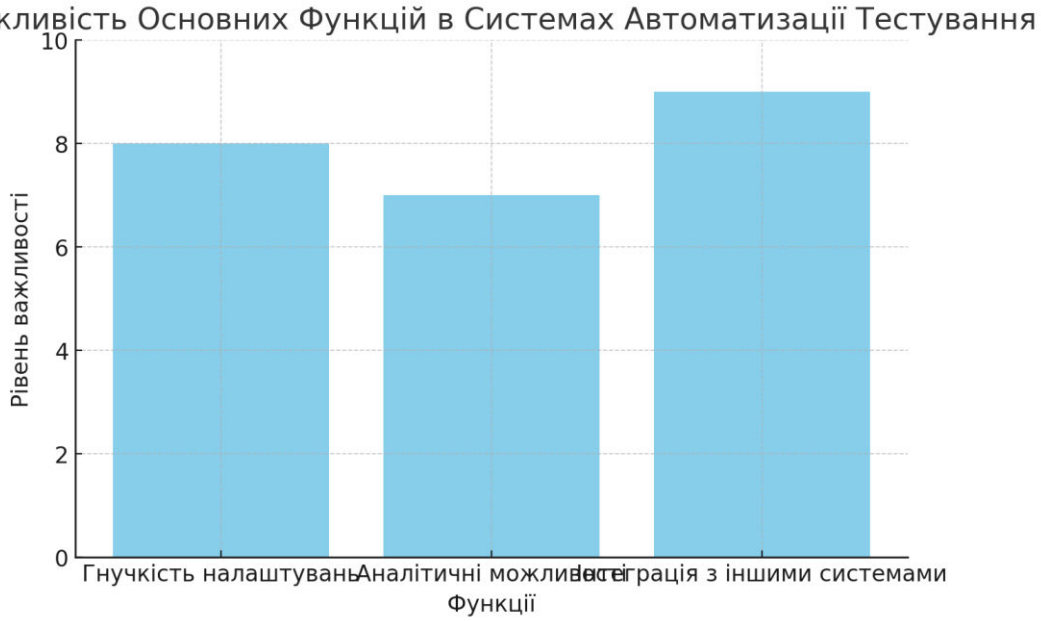

Рисунок 1.3 – Рівень важливості основних функцій

Вище представлено діаграму, яка ілюструє важливість основних функцій в системах автоматизації тестування, згідно з оцінками на шкалі від 1 до 10. На діаграмі відображені такі функції:

1. Гнучкість налаштувань: Оцінено у 8 балів, що підкреслює потребу в системах, здатних адаптуватися до різних освітніх вимог і методик.

2. Аналітичні можливості: Оцінено у 7 балів, це вказує на потребу в розширених аналітичних інструментах для оцінки результатів тестування та відстеження прогресу студентів.

3. Інтеграція з іншими системами: Найвищий показник – 9 балів, що відображає критичну важливість інтеграції систем тестування з іншими освітніми інструментами та платформами.

Ця діаграма демонструє ключові аспекти, на які слід звернути увагу при розробці нового програмного забезпечення для автоматизації тестування, акцентуючи на гнучкості, аналітиці та інтеграції.

Розглядаючи існуючі системи, важливо аналізувати відгуки користувачів та їхні основні потреби. Часто користувачі висловлюють потребу у більш

гнучких налаштуваннях тестів, покращеній аналітиці та інтеграції з іншими освітніми інструментами.

Огляд існуючих систем показує, що для нового програмного забезпечення важливо зосередитися на гнучкості налаштувань, інтегрованій аналітиці та забезпеченні високого рівня інтеграції. Розвиток технологій штучного інтелекту та хмарних обчислень також може надати нові можливості для покращення ефективності та зручності систем автоматизації тестування.

1.3 Огляд існуючих систем автоматизації тестування

Системи автоматизації тестування знань стали невід'ємною частиною освітнього процесу, дозволяючи викладачам та освітнім інституціям ефективно оцінювати знання студентів. Ці системи не лише спрощують процес створення та управління тестами, а й надають можливості для аналізу даних та персоналізованого навчання.

1.3.1 Moodle

Moodle відома своєю універсальністю та гнучкістю, пропонуючи широкий спектр інструментів для налаштування тестів, що дозволяє викладачам створювати тести, які точно відповідають їхнім вимогам до навчального процесу (рисунок 1.4).

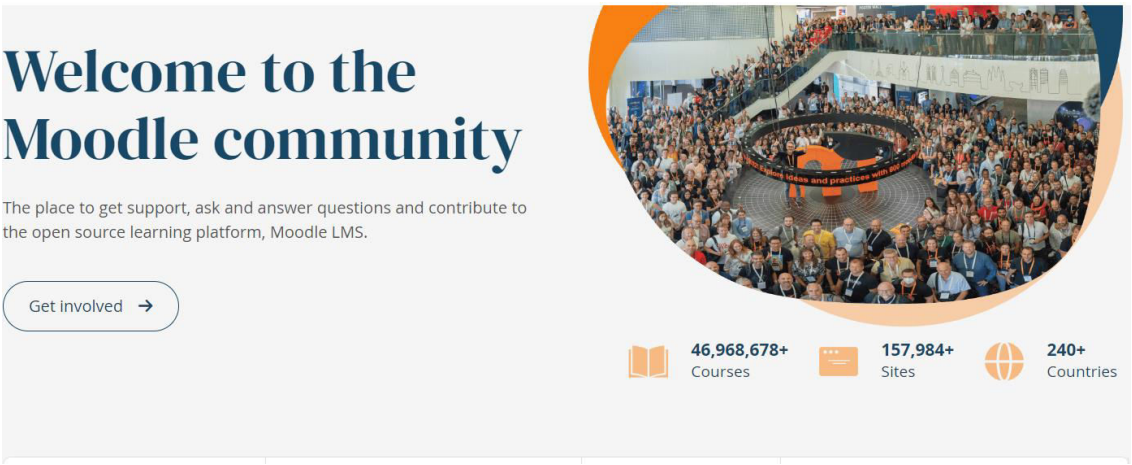

Рисунок 1.4 – Платформа Moodle

Докладний опис основних можливостей Moodle для налаштування тестів [2]:

– часові обмеження: Moodle дозволяє встановлювати часові рамки для тестування, забезпечуючи, що студенти повинні завершити тест протягом визначеного періоду. Це допомагає імітувати умови реального іспиту та забезпечує справедливість тестування;

– випадковий вибір питань: ця функція дозволяє тестам бути динамічними, вибираючи питання випадковим чином з певної бази даних питань. Це робить майже неможливим передбачення питань тесту та зменшує ймовірність списування;

– зворотній зв'язок: Moodle надає можливість налаштувати зворотній зв'язок для відповідей, що дає студентам інформацію про їхній прогрес і допомагає їм зрозуміти, де вони могли допустити помилки;

– інтеграція з іншими модулями курсу: тести в Moodle можуть бути інтегровані з іншими елементами курсу, такими як уроки та ресурси, що дозволяє створювати цілісний навчальний досвід;

– адаптивні тести: Moodle підтримує створення адаптивних тестів, де наступне питання або серія питань може змінюватися в залежності від відповідей студента, забезпечуючи більш персоналізований підхід до оцінювання;

– групування питань: викладачі можуть групувати питання за темами або складністю, що робить процес створення тесту більш організованим та зручним для навігації;

– багаторівнева безпека: Moodle має різні рівні безпеки для тестів, включаючи налаштування паролів для доступу до тесту та обмеження доступу за IP-адресами, щоб забезпечити, що лише авторизовані користувачі можуть брати участь у тестуванні.

Ці можливості роблять Moodle потужним інструментом для створення комплексних та ефективних тестів, які можуть бути точно налаштовані для задоволення різноманітних навчальних потреб і вимог.

1.3.2 Blackboard

Blackboard є однією з найпопулярніших платформ вищої освіти, що надає комплексний набір інструментів для управління курсами та тестами[3]. Ця система є вибором багатьох навчальних закладів завдяки своїй гнучкості та розширеному функціоналу (рисунок 1.5).

Рисунок 1.5 – Blackboard

Ось докладний огляд ключових особливостей Blackboard:

1. Інтегровані інструменти управління курсами: Blackboard надає широкий спектр інструментів для створення та управління курсами, включаючи розробку контенту, розподіл завдань, проведення обговорень та трекінг участі студентів.

2. Глибокі аналітичні засоби: система включає розширені аналітичні інструменти, які дозволяють викладачам та адміністраторам відстежувати

прогрес студентів, аналізувати ефективність навчальних матеріалів та покращувати навчальні методики.

3. Адаптивне навчання: Blackboard підтримує адаптивні навчальні технології, дозволяючи курсам динамічно адаптуватися до потреб та рівня знань кожного студента, забезпечуючи персоналізований навчальний досвід.

4. Інтеграція з іншими системами: платформа легко інтегрується з іншими навчальними та інформаційними системами, що використовуються в університетах та коледжах, включаючи системи управління навчальними ресурсами та студентські інформаційні системи.

5. Багатомовність: Blackboard підтримує багато мов, що робить його доступним для міжнародних навчальних закладів та студентів з різних країн.

6. Мобільність: система має потужні мобільні додатки, які дозволяють студентам та викладачам залишатися зв'язаними з навчальним процесом навіть в русі, надаючи можливість доступу до курсів, матеріалів, форумів та оцінок з будь-якого пристрою.

7. Безпека та Надійність: Blackboard реалізовано з високим рівнем безпеки для захисту особистих даних студентів та інтелектуальної власності викладачів, забезпечуючи стабільну та надійну роботу системи.

1.3.3 Canvas

Платформа Canvas відома своєю простотою використання та гнучкістю, що робить її ідеальним вибором для ефективного управління курсами та інтеграції з різноманітними інструментами (рисунок 1.6).

Рисунок 1.6 – Canvas

Основні особливості Canvas включають [4]:

1. Інтуїтивно зрозумілий інтерфейс: Canvas пропонує чистий, простий інтерфейс, який полегшує навігацію як для викладачів, так і для студентів.

2. Гнучкість налаштувань курсу: викладачі можуть легко створювати, редагувати та управляти курсами, використовуючи широкий спектр інструментів та налаштувань, що включають кастомізацію розкладу курсу, створення модулів навчання, та інтеграцію зовнішніх інструментів.

3. Підтримка різноманітних типів контенту: Canvas дозволяє інтегрувати в курси різні типи медіа-контенту, включаючи відео, зображення, текст та інші вебресурси, що робить матеріал більш інтерактивним та залучаючим.

4. Інструменти співпраці та комунікації: платформа має вбудовані інструменти для дискусій, групових проєктів, чатів та відеоконференцій, що сприяє ефективній взаємодії між учасниками курсу.

5. Широкі можливості оцінювання та зворотного зв'язку: Canvas надає розширені інструменти для створення тестів, опитувань, самооцінок та інших форм оцінювання, а також дозволяє викладачам залишати детальний зворотний зв'язок студентам.

6. Інтеграція з іншими системами: Canvas може інтегруватися з великою кількістю зовнішніх інструментів та сервісів, таких як Google Apps, Microsoft Office, системи відеоконференцій, що розширює можливості навчання.

7. Мобільний додаток: Canvas пропонує мобільні додатки для iOS та Android, які дозволяють користувачам залишатися в курсі оновлень курсу, переглядати матеріали та виконувати завдання на ходу.

Ці характеристики роблять Canvas універсальним рішенням, яке відповідає потребам сучасних навчальних закладів та корпоративного навчання, надаючи інструменти для створення адаптивного та інтерактивного навчального процесу.

1.3.4 Google Forms

Google Forms є популярним сервісом від компанії Google, який забезпечує зручні інструменти для створення опитувань та тестів [5]. Цей веббазований інструмент дозволяє користувачам легко створювати анкети, використовуючи різні типи питань, такі як вибір з множини, короткі відповіді та інші. Однією з особливостей Google Forms є можливість автоматичного збору відповідей та їх аналізу, що робить його корисним інструментом для збору та аналізу даних (рисунок 1.7).

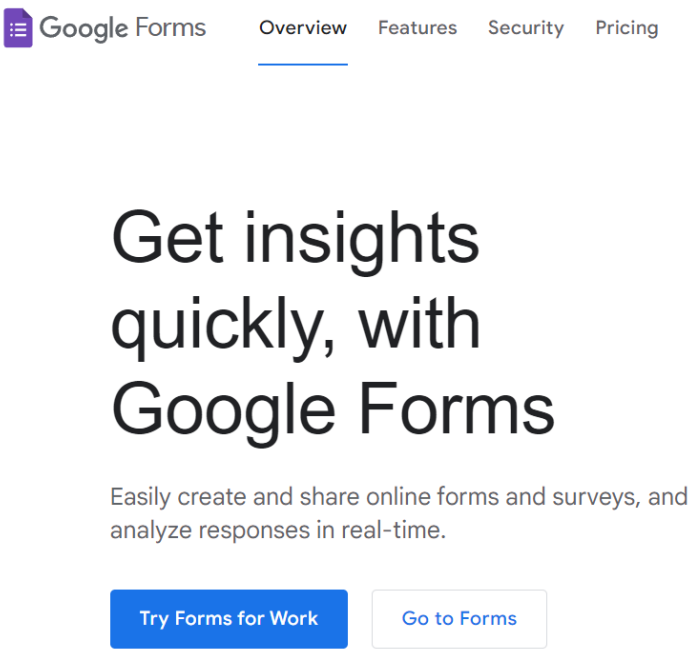

Рисунок 1.7 – Google Forms

Основні характеристики та функції Google Forms включають:

– створення анкет і тестів: Google Forms дозволяє створювати анкети з різноманітними типами питань, забезпечуючи широкі можливості для збору даних;

– налаштування дизайну: користувачі можуть налаштовувати зовнішній вигляд своїх форм, включаючи кольори, шрифти та інші елементи дизайну, для покращення користувацького досвіду;

– автоматичний збір відповідей: відповіді на опитування автоматично збираються та зберігаються, що полегшує аналіз даних;

– інтеграція з іншими інструментами Google: Google Forms легко інтегрується з іншими продуктами Google, такими як Таблиці Google, що дозволяє зручно обробляти зібрані дані.

Переваги Google Forms включають його інтуїтивно зрозумілий інтерфейс, легкість використання та доступність, оскільки сервіс є безкоштовним для користувачів з аккаунтом Google. Крім того, форми легко поширювати, надсилаючи посилання учасникам опитування.

Проте, сервіс має деякі обмеження, зокрема відсутність функції запланування відкриття та закриття анкети на визначений час та обмежені можливості аналізу даних порівняно з більш спеціалізованими платформами для тестування. Також Google Forms може бути менш підходящим для деяких складніших сценаріїв тестування через свою спрощену структуру та набір функцій.

1.4 Штучний інтелект у процесі автоматизації тестування

Застосування штучного інтелекту (ШІ) у системах автоматизації тестування, що становлять значний крок вперед у підвищенні ефективності та точності оцінювання знань [6].

Штучний інтелект дозволяє системам автоматизації тестування адаптуватися до індивідуальних потреб і рівня знань кожного користувача. Завдяки аналізу даних про попередні відповіді, ШІ може змінювати складність тестових завдань в реальному часі, тим самим оптимізуючи навчальний процес і забезпечуючи більш точну оцінку знань.

Однією з інноваційних функцій є використання алгоритмів машинного навчання для аналізу типових помилок і підбору завдань, які допомагають усунути конкретні прогалини у знаннях. Така індивідуалізація навчання сприяє більш ефективному засвоєнню матеріалу.

Інша суттєва перевага полягає у використанні ШІ для генерації тестових завдань. Алгоритми можуть автоматично створювати питання на основі навчального контенту, а також варіювати типи завдань, що робить тестування не тільки більш різноманітним, але й глибшим у плані оцінювання знань.

Додатково, впровадження ШІ може значно поліпшити зворотній зв'язок для користувачів. На основі аналізу відповідей ШІ може надавати деталізовані рекомендації щодо подальшого навчання, вказуючи на теми, які потребують додаткової уваги.

Впровадження штучного інтелекту у системи тестування знань не тільки підвищує точність і об'єктивність оцінювання, але й вносить значний вклад у персоналізацію навчального процесу, роблячи його більш ефективним та зацікавлюючим для користувачів. Це відкриває нові горизонти для розвитку освітніх технологій і створює безпрецедентні можливості для адаптації навчальних систем до індивідуальних особливостей та потреб кожного учня.

1.5 Підвищення ефективності тестування за допомогою аналітики

У сучасному освітньому процесі аналітика даних відіграє ключову роль у підвищенні ефективності тестування знань. Використання розширених аналітичних інструментів та методів дозволяє не тільки оцінювати результати тестування, але й глибше аналізувати процес навчання, виявляти тенденції та визначати напрямки для оптимізації.

Сучасні системи тестування інтегрують різноманітні аналітичні інструменти, що дозволяють:

– відстежувати динаміку змін: аналізувати результати тестів у динаміці, виявляючи як загальні тенденції у групі, так і індивідуальний прогрес кожного учня;

– визначати слабкі місця: виявляти теми та питання, які викликали найбільше труднощів, що дозволяє вчителям скоригувати навчальний план та зосередитися на проблемних аспектах;

– оцінювати ефективність тестів: використовувати статистичні методи для оцінки якості тестових завдань, зокрема, аналізувати рівень їх складності та дискримінативну здатність.

Ефективне представлення аналітичних даних за допомогою графіків, діаграм та інтерактивних звітів дозволяє користувачам системи легко інтерпретувати результати аналізу. Візуалізація сприяє кращому розумінню навчальних тенденцій, підвищує інформативність аналітики та допомагає приймати обґрунтовані рішення.

На основі зібраних та проаналізованих даних система може генерувати звіти та рекомендації для вчителів і студентів. Це включає в себе поради щодо планування подальшого навчання, необхідних корективів у методиці викладання та індивідуальних рекомендацій для студентів щодо підготовки до наступних тестів.

Застосування прогностичної аналітики дозволяє не тільки аналізувати минулі та поточні результати, але й прогнозувати майбутні тенденції. Використання алгоритмів машинного навчання та штучного інтелекту для обробки великих обсягів даних може допомогти передбачити потенційні труднощі студентів та адаптувати навчальний процес заздалегідь.

Для максимальної ефективності, аналітичні системи повинні інтегруватися з іншими навчальними платформами та інструментами, що дозволяє створювати єдиний екосистемний підхід до управління навчальним процесом. Така інтеграція сприяє забезпеченню безперервної зворотнього зв'язку та сталого розвитку освітнього середовища.

Застосування аналітики в системах автоматизації тестування знань не лише підвищує точність та об'єктивність оцінювання, але й сприяє персоналізації та адаптації навчального процесу, відкриваючи нові можливості для підвищення ефективності тестування в освітньому процесі.

1.6 Специфікація вимог

"Test Your Brain" є інноваційним програмним рішенням, створеним для оптимізації процесів тестування. Його розробка мотивована необхідністю удосконалення інструментів оцінювання знань, які використовуються в освітньому процесі, забезпечення їхньої об'єктивності та підвищення ефективності навчання. Актуальність такого ПЗ значно зросла у контексті глобальних викликів, зокрема пандемії та зміни освітнього середовища.

Тестування, як метод оцінки знань, залишається одним з найбільш універсальних і доступних способів верифікації навчальних досягнень.

Автоматизація цього процесу дозволяє мінімізувати часові та ресурсні витрати, підвищити точність результатів і зробити навчальний процес більш адаптивним і залучаючим.

Розроблений вебзастосунок "Test Your Brain" спрямований на автоматизацію створення та проведення тестів, роблячи процес більш ефективним для освітніх установ різних рівнів та індивідуальних користувачів. Він дозволяє:

1. Здійснювати реєстрацію та вхід в систему.
2. Кастомізувати профіль користувача.
3. Конструювати тести з різноманітними типами запитань.
4. Управляти тестами (редагування, видалення).
5. Аналізувати історію проходження та результати тестів.
6. Генерувати детальні звіти щодо ефективності тестів.
7. Планувати час та умови доступу до тестів.

Додатково, "Test Your Brain" інтегрує передові технології для підвищення якості освіти:

1. Аналітика тестів: система надає детальну аналітику щодо ефективності кожного тесту, дозволяючи виявити слабкі місця та оптимізувати навчальний процес.

2. Штучний інтелект: вбудований AI–асистент пропонує рекомендації щодо формулювання питань та підвищує якість тестових завдань, роблячи їх більш зрозумілими та ефективними.

Ролі та Взаємодія Користувачів

Система не обмежує користувачів ролями «викладач» та «учень», надаючи кожному користувачеві можливість як створювати, так і проходити тести. Це робить "Test Your Brain" універсальним інструментом для навчання та самоосвіти.

Технічні Аспекти

"Test Your Brain" реалізовано на мові C# для backend частини та Angular для frontend, з використанням реляційної БД SQLite. Заходи безпеки передбачають зберігання паролів у зашифрованому вигляді, а нормалізація БД забезпечує цілісність даних.

Інтерфейс та Доступність

Інтерфейс "Test Your Brain" розроблено з акцентом на інтуїтивність і доступність, забезпечуючи зручність користування для осіб без спеціалізованих навичок у сфері IT.

Надійність та Умови Експлуатації

Розроблене програмне забезпечення "Test Your Brain" забезпечує високий рівень надійності та безпеки, гарантуючи цілісність та конфіденційність даних користувачів в умовах їх активного використання в освітньому процесі.

Висновки до розділу 1

Перший розділ дипломної роботи занурює у контекст автоматизації оцінки знань, починаючи з опису предметної області і закінчуючи специфікацією вимог до розроблюваної системи. Аналітичний огляд і теоретичні основи, представлені в розділі, надають чітке розуміння важливості і актуальності розробки систем автоматизації тестування для сучасної освітньої сфери.

Вивчення класифікації тестових завдань і огляд існуючих систем, таких як Moodle, Blackboard, Canvas, і Google Forms, демонструють різноманітність підходів і технологій, які можуть бути використані для розробки ефективних рішень у цій області. Особливу увагу привертає аналіз ролі штучного інтелекту та аналітики у процесі автоматизації тестування, що відкриває нові перспективи для підвищення якості та ефективності освітніх процесів.

Специфікація вимог до розроблюваної системи закладає фундамент для подальшої роботи, визначаючи ключові функціональні та нефункціональні

вимоги, що забезпечують чіткі орієнтири для розробки та імплементації проєкту.

Таким чином, перший розділ є важливою підготовчою стадією, яка визначає теоретичну базу та вихідні точки для практичної частини роботи, спрямованої на створення інноваційної системи автоматизації тестування знань.

2 ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Варіанти використання системи

Діаграми варіантів використання є цінним інструментом в інженерії програмного забезпечення, що використовуються для ілюстрації взаємодії між різними акторами (користувачами або системами) та програмною системою. Вони допомагають організувати та наочно демонструвати функціональні вимоги до системи.

У контексті системи з різними рівнями доступу для авторизованих та неавторизованих користувачів такі діаграми можуть бути особливо інформативними. Для неавторизованих користувачів діаграма (рисунок 2.1) може показувати обмежені функціональні можливості, такі як проходження тестів, але з застереженням, що їхні результати не зберігаються. Ця візуальна репрезентація чітко розмежовує можливості, доступні різним типам користувачів.

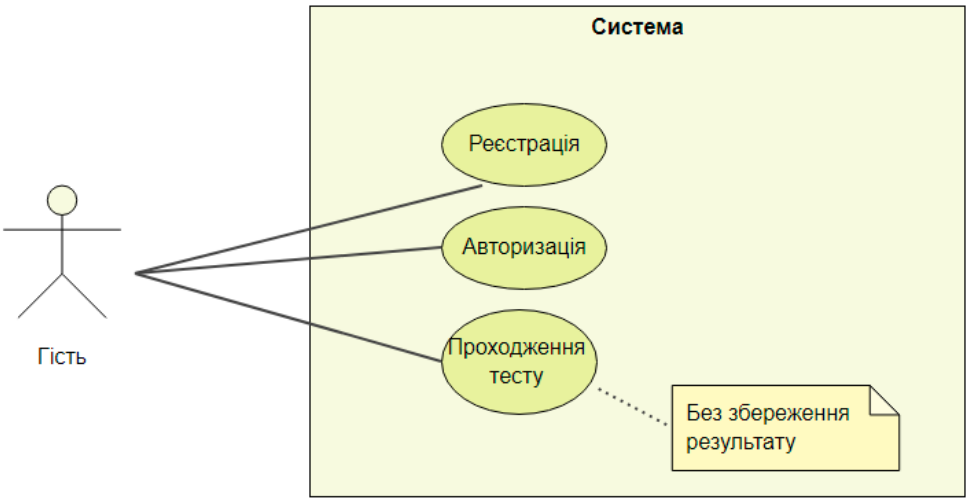

Рисунок 2.1 – Діаграма варіантів використання для неавторизованих користувачів

Для авторизованих користувачів, як зображено у наступній діаграмі (рисунок 2.2), спектр дій значно розширюється. Для спрощення розуміння та забезпечення логічного потоку функції можуть бути категоризовані під ролі,

такі як «Вчитель» та «Учень». Проте цей поділ є умовним, визнаючи, що користувач може змінювати ролі залежно від того, чи вони створюють чи проходять тести. Ця гнучкість підкреслює універсальність системи, де всі перераховані дії доступні будь–якому авторизованому користувачеві.

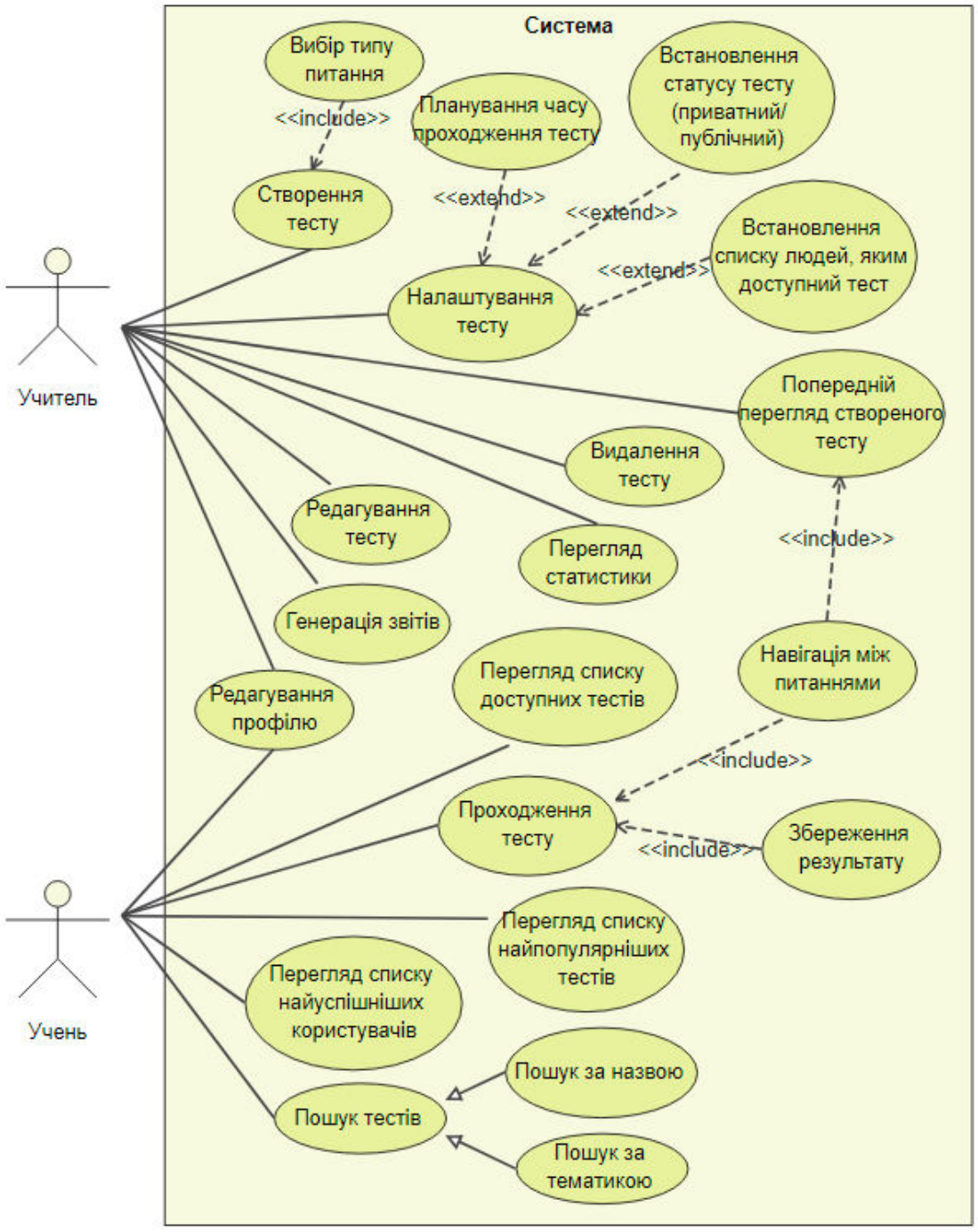

Рисунок 2.2 – Діаграма варіантів використання для авторизованих користувачів

Деякі конкретні функціональні можливості для авторизованих користувачів можуть включати попередній перегляд тесту під час його створення, що відрізняється від проходження тесту тим, що не включає збереження результатів. Ця функція призначена для перевірки та виправлення помилок вчителем, забезпечуючи точність статистики проходження тестів, а також пошук тестів за назвою або предметом.

2.2 Сценарії використання

Сценарії використання допомагають краще зрозуміти функціонування системи, надаючи докладний опис процесів взаємодії користувача з нею.

Приклад використання 1 (спрощена форма): Розробка тесту

Учитель планує створити новий тест. Для цього він переходить на головну панель управління тестами, відому як Dashboard. Там він клацає на кнопку «Створити». Це дія відкриває форму для розробки тесту. У цій формі вчитель вводить назву і тему тесту, може додати зображення. Потім він переходить до створення запитань для тесту, вибирає тип запитань, вносить варіанти відповідей і вказує правильні відповіді. При потребі, питання можуть бути видалені, або скопійовані для створення нових запитань на основі існуючих, що дозволяє заощадити час. Користувач також може визначити, чи є питання обов'язковими або необов'язковими. Після завершення введення всієї інформації, вчитель зберігає створений тест. Новий тест тоді з'являється у списку на сторінці Dashboard.

Сценарій використання 2 (спрощена форма): Аналіз результатів тестів

Розробник тестів бажає оглянути статистику по тестах, які він створив. Він переходить на сторінку «Показники» (Performace) і встановлює часовий період для перегляду статистики. У списку він вибирає конкретні тести, результати яких хоче побачити на діаграмі. Система виводить оновлену діаграму успішності, де відображено, як змінювався середній бал по цим тестам

протягом вказаного часу. Розробник використовує ці дані для покращення тестів та порівняння їх ефективності.

Сценарій використання 3 (спрощена форма): Оновлення профілю

Користувач, зареєстрований лише з базовою інформацією, вирішує оновити свій профіль. Він переходить на сторінку «Обліковий запис» і бачить свої поточні дані. Тут він має можливість виконати декілька дій: завантажити нове фото для свого аватара, змінити пароль, вказати своє ім'я та прізвище, вибрати країну проживання, а також змінити ім'я користувача в системі. Після внесення необхідних змін, система автоматично зберігає оновлену інформацію.

Сценарій використання 4 (спрощена форма): Створення звітів

Користувач переходить на сторінку «Генератор звітів» для отримання звітів про результати тестів. Він обирає зі списку тести, інформацію про які він хоче включити у звіт, і визначає період часу, за який потрібні дані. Після вибору всіх параметрів, система формує звіт, який можна скачати та використовувати для подальшого аналізу.

Сценарій використання 5 (спрощена форма): Процес проходження тесту

Роль: Учень

Основний сценарій:

– Учень відкриває сторінку «Особисті тести» (Personal Tests).

– Вибирає доступний тест зі списку.

– Натискаючи кнопку для початку, учень переходить до змісту тесту.

– Відповідає на запитання, переміщаючись між ними.

– По завершенню, учень підтверджує надсилання своїх відповідей.

– Система розраховує результат та зберігає його в базі даних.

– Учень може ознайомитися з результатами.

– Інформація про дату, час проходження та набрані бали заноситься на сторінку «Результати» (Results) для можливості їх перегляду у майбутньому.

Альтернативні сценарії:

1. Якщо учень має велику кількість доступних тестів, він може використовувати пошук за назвою на цій сторінці.

2. Учень може обирати не лише особисто доступні йому тести, але й тести зі списку найпопулярніших (вкладка «Глобальний рейтинг» у розділі Global Ranking) або знайти цікавий тест через загальний пошук за назвою чи темою.

3. Учень має можливість перервати проходження тесту в будь–який час.

Usecase 6 (повна форма)**: *Налаштування та налагодження тесту***

Таблиця 2.1 – Сценарій «Налаштування та налагодження тест»

Primary Actor	Авторизований користувач
Scope	Система автоматизації тестування знань *Test Your Brain*
Level	Sub–function
Preconditions	Користувач створив хоча б один тест
Stakeholders and interests	1. Розробник тесту (вчитель) – налаштування тесту згідно із поставленими задачами перевірки знань учнів 2. Учень – отримання доступу до тесту у визначений час
Main Success Scenario: 1. Розробник тесту переходить до сторінки Preview у секції Setup. 2. В списку тестів він обирає потрібний і розпочинає його проходження. 3. Користувач відповідає на питання тесту, під час чого перевіряє коректність заповнення тесту. 4. Після відправки форми система обраховує та показує отриманий результат. Проте результат не зберігається в БД. 5. Розробник тесту переходить до сторінки Planning у секції Setup. 6. Система відображає список наявних тестів та налаштування за замовченням. 7. Користувач встановлює значення параметрів для кожного тесту або залишає параметри за замовчуванням. 8. Система зберігає зміни.	

Кінець таблиці 2.1

Result	Тест підготовлено до проходження учнями згідно із побажаннями його розробника.
Extensions	
3a – Розробник тесту виявив помилку у питаннях чи відповідях. 1. Користувач переходить до основної сторінки керування тестами "Dashboard". 2. Серед наявних тестів він обирає потрібний та натискає на іконку «Редагувати». 3. Відкривається форма з даними про тест, користувач вносить зміни. 4. Користувач зберігає зміни. 5. За необхідності користувач повертається до сторінки Preview, щоб впевнитися, що зміст тесту вже повністю правильний.	
7a – Користувач хоче зробити тест загальнодоступним: 1. Користувач встановлює статус тесту як "Public" 2. Система зберігає зміни, тест стає доступним усім користувачам. 7b – Користувач хоче надати доступ до тесту лише визначеним особам: 1. Користувач встановлює статус тесту як "Private". 2. Розробник тесту додає інших користувачів системи у список осіб, що мають доступ до тесту (за адресою електронної пошти або нікнеймом). 3. Система зберігає зміни, тест з’являється у списку на сторінці Personal Tests вказаних користувачів. 7c – Користувач хоче обмежити час, коли тест буде доступний для проходження. 1. Користувач вказує початкову дату та час, коли тест має стати доступним. 2. Користувач вказує кінцеву дату та час, коли тест «закривається» для проходження.	

2.3 Діаграма станів

Для ілюстрації динамічної взаємодії між користувачем та системою була створена діаграма станів. Ця діаграма візуалізує різні шляхи взаємодії з системою від моменту її запуску до завершення роботи, відображаючи ключові стани, у яких може перебувати програма.

Діаграма станів (або Statechart Diagram) демонструє, як програма переходить з одного стану в інший у відповідь на дії користувача або наступ подій [7].

Функціонал, який доступний користувачу, залежить від його статусу в системі. Наприклад, неавторизований користувач може переглядати тільки ті тести, які мають відкритий доступ. Вибравши тест, він може його пройти, а після завершення система надає можливість переглянути результати (як показано на рисунку 2.3).

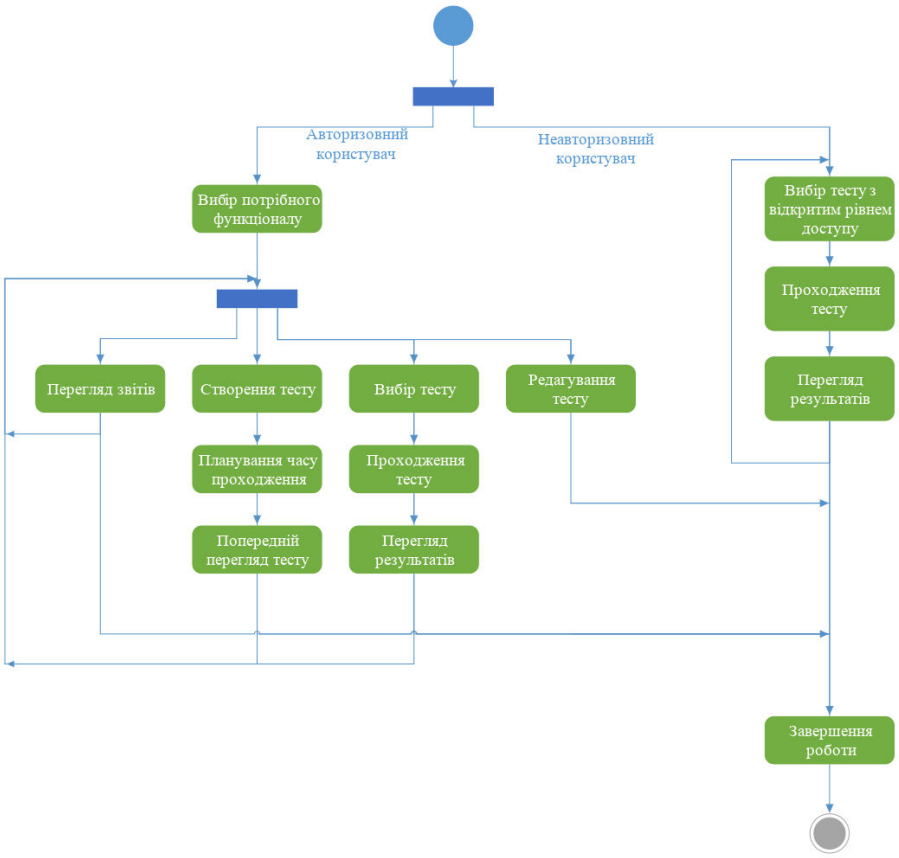

Рисунок 2.3 – Діаграма станів

Коли користувач увійшов у систему, йому відкриваються додаткові можливості взаємодії з додатком. Він може скористатися різними опціями у головному меню, такими як створення нового тесту, вибір тесту для проходження, редагування вже існуючих тестів, або створення звітів на основі вже створених тестів. Завершивши обрану дію, користувач має змогу повернутися до головного меню, де він може обрати іншу діяльність до того, як вийде з системи.

Всі переходи між різними станами системи відбуваються через графічний інтерфейс за допомогою натискання на відповідні кнопки.

2.4 Діаграма послідовності

Взаємодію front–end та back–end частин системи між собою та з користувачем можна зобразити за допомогою діаграм послідовності. На цих діаграмах особлива увага приділяється порядку взаємодій.

Діаграми розроблені для декількох важливих багатокрокових варіантів використання системи, а саме *реєстрації*, *створення тесту* та його *проходження* (рисунок 2.4).

Реєстрація

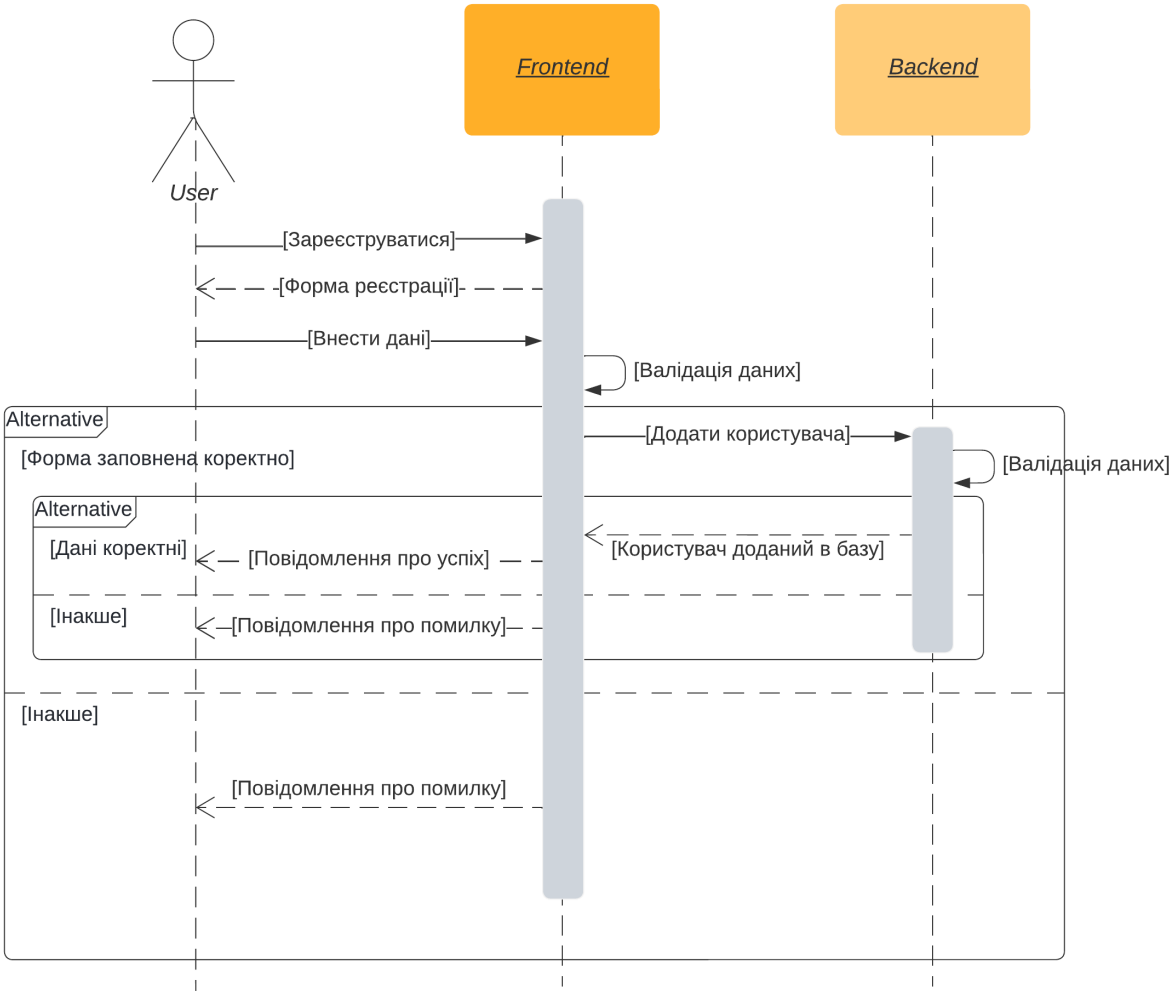

Рисунок 2.4 – Діаграма варіантів використання для реєстрації користувача

Ця діаграма деталізує процес реєстрації в системі, де головним учасником є неавторизований користувач. Вона відображає різні шляхи, якими може розвиватися процес в залежності від точності даних, введених користувачем.

Згідно з діаграмою, процедура валідації даних складається з двох етапів: перевірки на рівні frontend і backend. На першому етапі, frontend виконує первинну перевірку даних, таку як повнота заповнення полів, відповідність типу і формату даних, достатню міцність пароля та інше. Якщо дані виявляються некоректними, користувача інформують про помилку. У випадку успішної первинної валідації, дані передаються на другий етап до backend, де

відбувається перевірка на унікальність імені користувача та зіставлення з іншими даними у базі.

Якщо ім'я користувача виявляється неунікальним, система пропонує вибрати інше ім'я. У випадку, коли електронна адреса вже є в базі, система натякає на те, що користувач можливо вже зареєстрований, і рекомендує йому увійти в систему замість повторної реєстрації.

Створення тесту

Для створення нового тесту користувач спочатку натискає на відповідну кнопку на головній панелі керування. Після цього він вводить інформацію про тест, таку як назву, тематику, і за бажанням додає зображення. Потім користувач приступає до додавання питань до тесту. За кожного нового питання, інтерфейс (frontend) автоматично надає шаблон для його створення. Цей процес продовжується до тих пір, поки не будуть введені всі питання, що відображено на діаграмі за допомогою блоку управління циклом (loop). Користувач також має можливість редагувати або видаляти питання, якщо це необхідно, згідно з відповідними опціональними блоками (opt).

Крім того, на етапі frontend відбувається валідація заповнення форми, така як перевірка наявності усіх обов'язкових полів, що також зображено на діаграмі (рисунок 2.5).

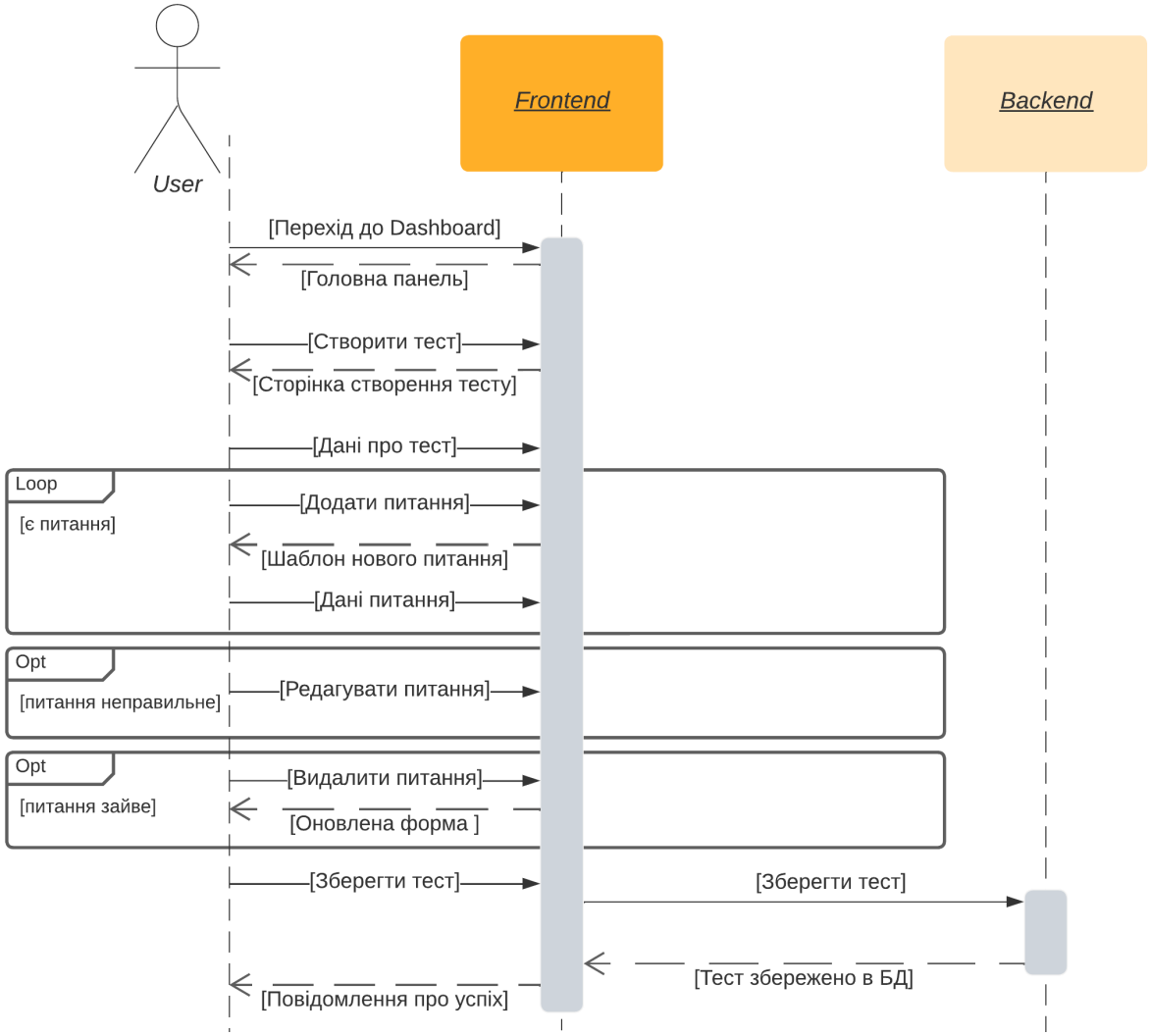

Рисунок 2.5 – Діаграма послідовності процесу створення тесту

По завершенні заповнення форми для створення тесту, інформація надсилається на сервер. Вона обробляється на боці backend, де й відбувається її збереження у базі даних. Успішне збереження тесту супроводжується відправленням повідомлення користувачеві про те, що тест додано успішно.

Ці та наступні діаграми описують діяльність авторизованого користувача, адже саме він має право на створення тестів і проходження їх з можливістю збереження результатів.

Щодо проходження тесту, користувач, який бажає переглянути доступні йому тести, переходить у секцію «Особисті тести» (Personal Tests). Тут відбувається з'єднання із сервером, який надає список тестів, доступних для цього користувача. На сторінці ці тести відображаються у формі списку. Коли

користувач натискає кнопку «Пройти тест», ініціюється запит до backend, у відповідь на який система надсилає питання для обраного тесту, щоб відобразити їх користувачу (як показано на рисунку 2.6).

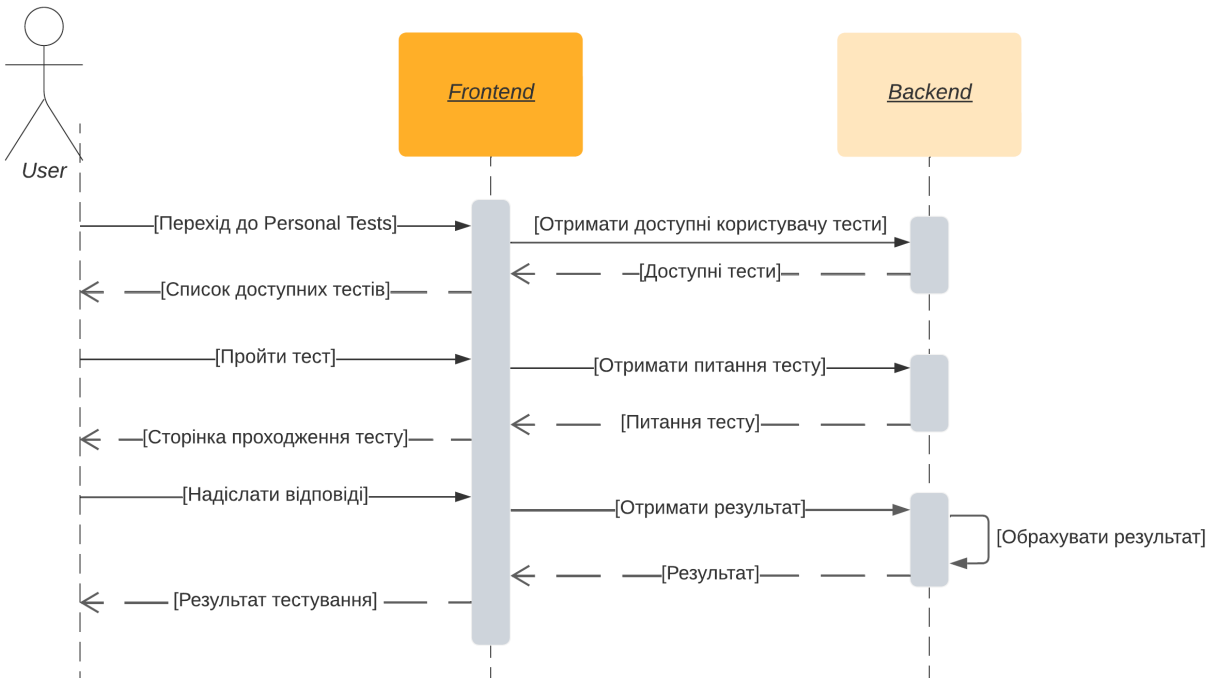

Рисунок 2.6 – Діаграма послідовності для процесу проходження тесту

Користувач виконує тест, вибираючи відповіді, які він вважає правильними, і відправляє заповнену форму тесту. Ці відповіді потім пересилаються на сервер, де вони порівнюються з правильними відповідями, визначеними у ключі до тесту. Після цього відбувається розрахунок кінцевого результату тестування.

Отримані результати тесту потім відсилаються назад до інтерфейсу користувача (frontend), де вони відображаються для ознайомлення. Також ці результати реєструються та зберігаються у базі даних сервера. Відповідно до цього, оновлюються статистика проходження даного тесту, а також рейтинги популярності тестів та успішності користувачів, які беруть участь у тестуванні.

2.5 Концептуальна модель

Під час проєктування бази даних і класів системи важливим кроком є створення концептуальної моделі, що включає сутності та їх взаємозв'язки. Для цього використовується діаграма «сутність-зв'язок» (Entity-Relationship Diagram), яка допомагає уявити структуру системи, відокремлено від її програмної реалізації [8].

ER–діаграми є ключовим інструментом у проєктуванні баз даних, оскільки вони демонструють типи зв'язків між сутностями, такі як 1:N або N:M. Існують два основних типи нотацій для таких діаграм:

– Нотація Пітера Чена, де сутності показані у вигляді прямокутників, а зв'язки – ромбами. Кратність зв'язків вказується числами.

– Нотація «Вороняча лапка», де сутності також представлені прямокутниками, але зв'язки – лініями. Кратність зв'язку зображається графічно.

У даному випадку була обрана нотація Пітера Чена, оскільки вона надає більше деталей щодо природи взаємозв'язків між сутностями. Згідно з діаграмою на рисунку 2.7, учитель має можливість створювати багато тестів, але кожен тест має лише одного автора, тобто декілька учителів не можуть спільно створити один тест. Також учень може проходити безліч тестів, а багато учнів можуть бути частиною загального рейтингу успішності.

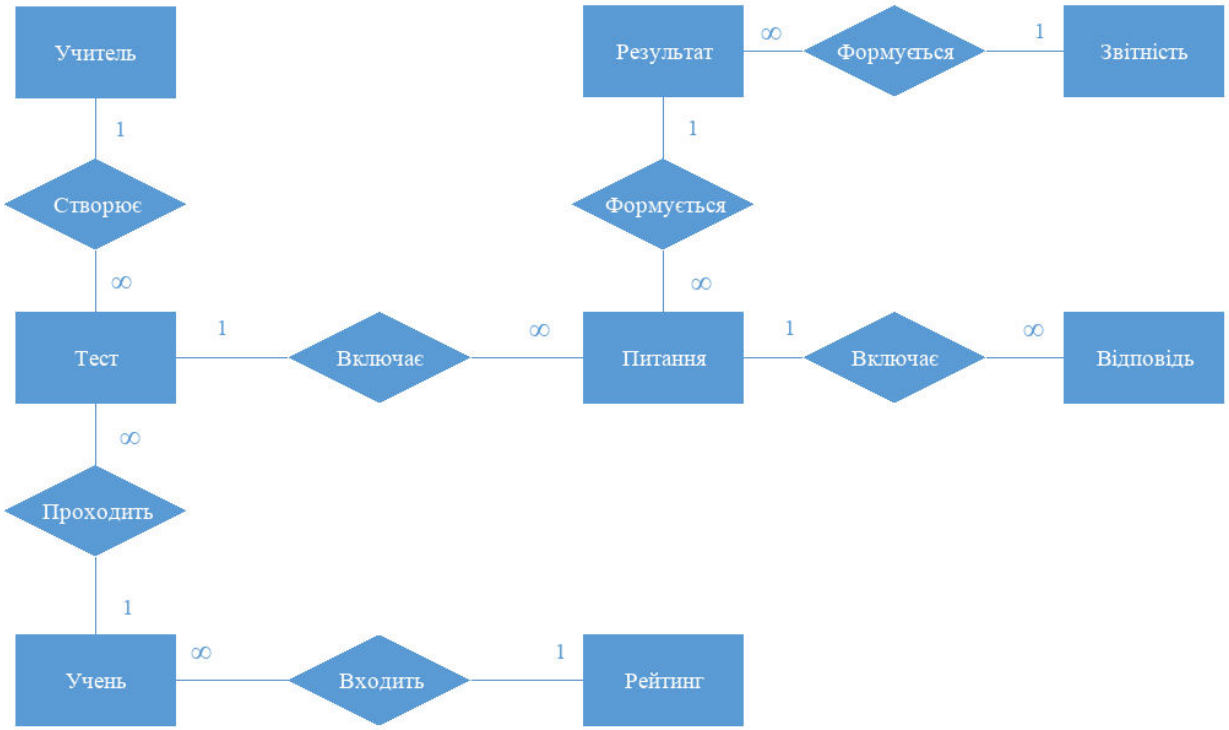

Рисунок 2.7 – Концептуальна модель системи

Тест може включати в себе довільну кількість питань, а питання можуть мати різну кількість відповідей. Кожен результат формується з урахуванням множини питань тесту, а з множини результатів тестів формується звіт про успішність учнів.

2.6 Діаграма розгортання

Для відображення обчислювальних вузлів під час роботи програми було створено діаграму розгортання (рисунок 2.8).

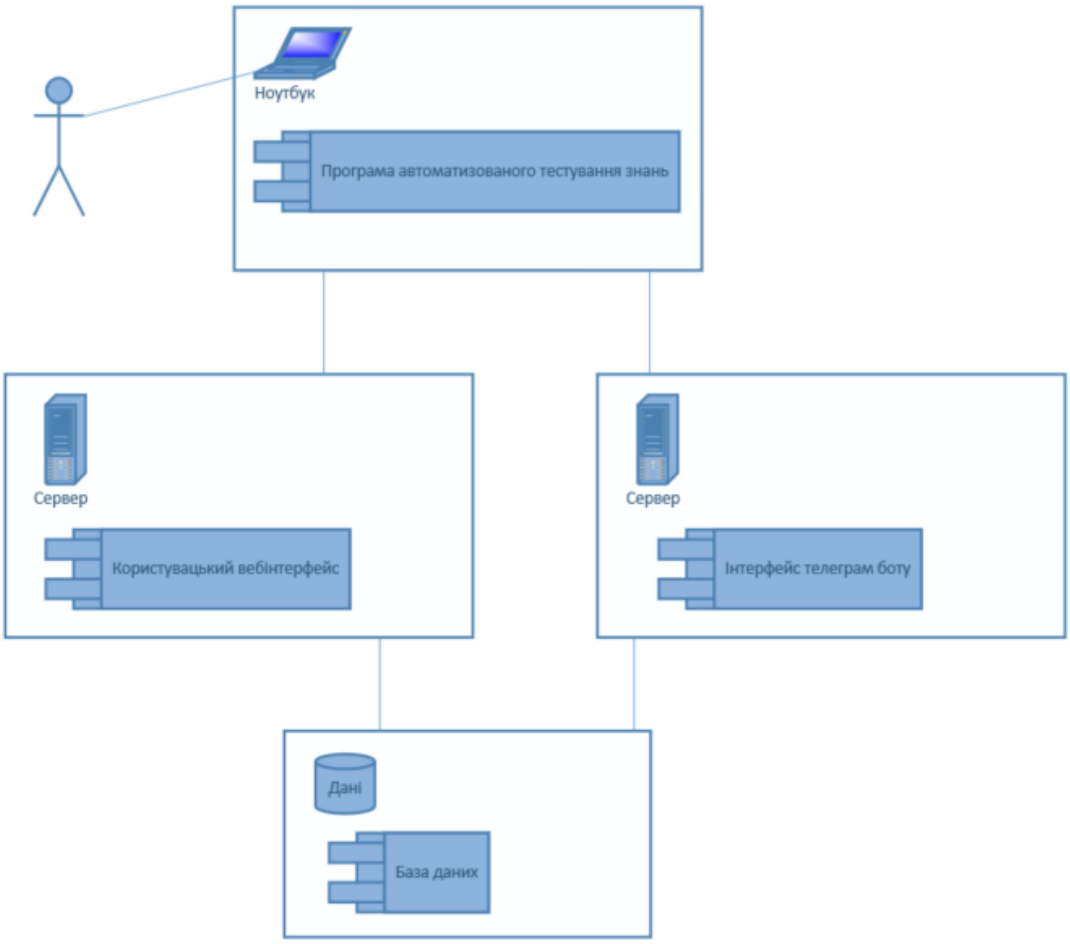

Рисунок 2.8 – Діаграма розгортання

Користувач взаємодіє із системою шляхом використання програми автоматизованого тестування знань. Програма відправляє запити та отримує відповіді від вебсервера. Сервер надає користувацький вебінтерфейс у браузері, або інтерфейс телеграм боту. Сервер відправляє та отримує дані з бази даних.

2.7 Схема бази даних

Схема бази даних відіграє ключову роль у структуруванні та організації даних у рамках реляційної бази даних, охоплюючи аспекти такі як назви таблиць, поля, типи даних та зв'язки між різними сутностями. Ця схема є життєво важливою як для розробників, так і для замовників, особливо у випадках, коли система підлягає розвитку та розширенню новими функціональними можливостями.

Важливість правильного проєктування бази даних полягає в її здатності задовольняти потреби у звітуванні та обробці даних, що в свою чергу стає гарантією надійності системи. Для досягнення цієї мети необхідно слідувати визначеному процесу, який включає кроки, такі як визначення цілей бази даних, упорядкування інформації, розподіл даних по таблицям, перетворення елементів інформації у стовпці, визначення первинних ключів, налаштування зв'язків між таблицями, аналіз створеної бази даних та проведення її нормалізації.

Нормалізація даних відіграє критичну роль у забезпеченні цілісності бази даних, дозволяючи виконувати вимоги до різних «нормальних» форм, починаючи від першої та закінчуючи п'ятою. Цей процес допомагає уникнути повторення даних та залежності від неключових атрибутів, що сприяє більш ефективній організації інформації у базі даних.

Переваги чітко визначеної схеми бази даних включають поліпшення доступності та безпеки даних, полегшення організації та комунікації між учасниками проєкту та забезпечення цілісності даних. Це досягається через упорядковане розміщення даних у окремі сутності, що спрощує доступ до спільних ресурсів та дозволяє розробникам та замовникам краще розуміти структуру та обмеження бази даних.

У розглянутій системі база даних містить шість основних сутностей, які відображають різні аспекти інформаційної системи (рисунок 2.9):

1. Users: зберігає інформацію про користувачів, включаючи їхні ідентифікатори, імена та інші персональні дані.

2. Answers: містить дані про відповіді на питання тестів, включаючи відповідність між відповідями та питаннями.

3. Questions: включає інформацію про питання в тестах, а також вказує, до якого тесту кожне питання належить.

4. Tests: зберігає дані про тести, включно з інформацією про власників тестів, предмети та інші відповідні атрибути.

5. Subscribers: містить інформацію про користувачів, які підписані на певні тести, дозволяючи відстежувати, хто має доступ до конкретних тестів.

6. Results: загальні дані про результати користувачів у тестах, включаючи кількість пройдених тестів та точність відповідей.

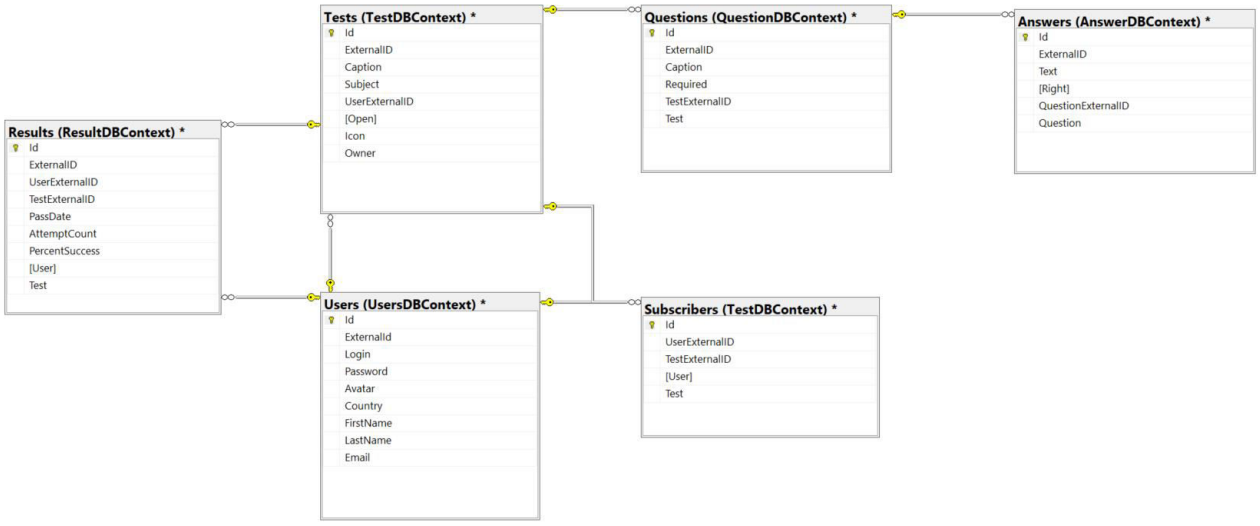

Рисунок 2.9 – Схема БД

В результаті, система зберігає інформацію в легкодоступній та зрозумілій формі, забезпечуючи швидкий доступ до потрібної інформації та слугуючи напрямком для майбутнього розширення системи новими сутностями.

Висновки до розділу 2

У другому розділі розроблено сценарії використання системи. За допомогою концептуальної моделі, діаграм варіантів використання, діаграми станів, послідовності та розгортання змодельовано різні аспекти системи. Створені діаграми UML стануть основою для програмної реалізації продукту. Отже, здійснено структурно–функціональне проєктування системи, що уможливлює подальшу розробку.

3 АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір програмних засобів розробки

Для реалізації backend вибрано ASP.NET Web API з використанням мови програмування C#. ASP.NET Web API є потужним фреймворком для створення HTTP-сервісів, які можуть бути використані різними клієнтськими додатками, включно з браузерами та мобільними додатками. Вибір ASP.NET Web API обумовлений його здатністю легко створювати RESTful API з підтримкою широкого спектру форматів даних для відповідей, включно з JSON і XML, що забезпечує гнучкість у взаємодії з клієнтською частиною [9].

Entity Framework використовується як ORM (Object-Relational Mapping) інструмент, що дозволяє розробникам працювати з базою даних за допомогою об'єктно-орієнтованого підходу, замість безпосереднього використання SQL-запитів. Це спрощує роботу з даними і покращує продуктивність розробки [10].

Для розробки frontend використовується Angular – сучасний фреймворк від Google. Angular надає розробникам потужні інструменти та компоненти для створення інтерактивних односторінкових додатків (SPA). Це дозволяє створити зручний та ефективний користувацький інтерфейс для системи тестування [11].

Обрана архітектура Loosely Coupled Monolith забезпечує гнучкість та масштабованість системи, дозволяючи легко додавати нові функції та компоненти без необхідності вносити значні зміни в існуючу структуру. Такий підхід забезпечує ефективне співвідношення продуктивності, масштабованості та легкості обслуговування системи [12].

У цілому, обраний стек технологій та архітектурних рішень забезпечує надійну основу для створення високопродуктивного, безпечного та масштабованого застосунку для автоматизації тестування знань, що відповідає сучасним вимогам до веброзробки.

3.1.1 Основи клієнт–серверної архітектури

Клієнт-серверна архітектура – це модель розподіленого додатку, де завдання або робочі навантаження розподіляються між постачальниками ресурсів або сервісів, відомими як сервери, та замовниками цих сервісів, відомими як клієнти. Ця модель є фундаментальною для сучасних мережевих застосунків, включно з вебсайтами, додатками для корпоративного використання та, звісно, системами оцінки знань [13].

Ключові компоненти наступні:

1. Сервер: вузол у мережі, який надає ресурси, дані, послуги або програми клієнтам. У контексті систем оцінки знань, сервер зазвичай зберігає тестові дані, керує базою даних користувачів та обробляє логіку тестування.

2. Клієнт: програмне забезпечення або пристрій, який надсилає запити серверу на доступ або використання ресурсів. Клієнти можуть бути веббраузерами, мобільними додатками або настільними програмами, які забезпечують інтерфейс для студентів у системах оцінки знань.

Робочі принципи

– запити та відповіді: клієнти надсилають запити до сервера, який обробляє їх і відправляє відповіді назад. Наприклад, студент (клієнт) може відправити відповіді на тест, які потім обробляються на сервері;

– обробка запитів: сервери виконують обробку даних, зберігання інформації та управління ресурсами, забезпечуючи функціональність, необхідну для тестування;

– масштабування та розподіл навантаження: клієнт–серверна архітектура дозволяє ефективно масштабувати систему, додаючи або оновлюючи сервери залежно від потреб.

Типи серверів у клієнт-серверних архітектурах

1. Вебсервери: Відповідають за обслуговування вебсторінок та вебзастосунків.

2. Бази даних серверів: зберігають та обробляють дані, наприклад, результати тестів, інформацію про користувачів.

3. Додаткові сервери: забезпечують різні специфічні функції, такі як обробка електронної пошти, файлові послуги.

Протоколи комунікації

– HTTP/HTTPS: Найпоширеніші протоколи для комунікації;

– FTP: Використовується для передачі файлів;

– WebSocket: Забезпечує двосторонній комунікаційний канал між клієнтом та сервером.

Безпека в клієнт–серверних архітектурах

– аутентифікація та авторизація: забезпечення, що лише авторизовані користувачі мають доступ до ресурсів;

– шифрування даних: використання HTTPS та інших технологій шифрування для захисту даних під час передачі.

3.1.2 Роль клієнт-серверних архітектур у системах оцінки знань

Централізоване управління даними

– зберігання та обробка даних: сервери у клієнт–серверних архітектурах ефективно зберігають та обробляють великі обсяги даних, включаючи навчальні матеріали, результати тестів, та дані студентів;

– оновлення контенту: централізоване управління дозволяє швидко оновлювати та модифікувати навчальний контент, гарантуючи його актуальність та відповідність сучасним освітнім стандартам.

Масштабованість та надійність

– підтримка великої кількості користувачів: клієнт–серверні системи здатні обслуговувати велику кількість студентів одночасно, забезпечуючи стабільний доступ до тестів та освітніх ресурсів;

– гарантія неперервної роботи: завдяки резервуванню та відмовостійкості, сервери можуть забезпечити постійний доступ до освітніх матеріалів та систем оцінювання, мінімізуючи ризики збоїв та перебоїв у навчанні.

Гнучкість та інтеграція

– інтеграція з різними платформами: клієнт–серверна архітектура дозволяє легко інтегруватися з різними освітніми платформами, ЛМС (системами управління навчанням), та іншими онлайн–сервісами;

– підтримка різноманітних клієнтських пристроїв: студенти та викладачі можуть отримати доступ до системи з будь–якого пристрою, включаючи комп'ютери, планшети, та смартфони, забезпечуючи гнучкість у виборі методів навчання.

Безпека та Конфіденційність

– захист особистих даних: сервери забезпечують високий рівень безпеки для зберігання та обробки персональних даних студентів і викладачів;

– контроль доступу до ресурсів: авторизація та аутентифікація на сервері гарантують, що доступ до навчальних матеріалів та результатів тестування мають лише уповноважені особи.

Оптимізація процесу оцінювання

– швидка обробка запитів: ефективне управління запитами на сервері дозволяє швидко обробляти результати тестів, забезпечуючи оперативне оцінювання;

– аналітика та звітність: сервери можуть обробляти великі набори даних для генерації аналітичних звітів, що допомагає викладачам оцінювати ефективність навчальних програм та методик тестування.

Клієнт–серверні архітектури, таким чином, відіграють вирішальну роль у сучасних системах оцінки знань, забезпечуючи ефективність, масштабованість, безпеку та гнучкість навчальних процесів.

3.2 Програмна реалізація системи

ASP.NET Web API є ключовим інструментом для створення серверних застосунків, особливо в контексті автоматизованих систем оцінки знань. Цей потужний фреймворк від Microsoft дозволяє розробникам створювати гнучкі, масштабовані та ефективні вебсервіси, які можуть легко інтегруватися з різними клієнтськими додатками.

Ключові характеристики та переваги

– масштабованість та висока продуктивність: ASP.NET Web API підтримує асинхронну обробку запитів, що є критично важливим для систем оцінки знань, які обробляють велику кількість одночасних запитів від користувачів;

– підтримка різних форматів даних: фреймворк може обробляти дані у різних форматах, включаючи JSON та XML, забезпечуючи гнучкість у взаємодії з різними клієнтськими додатками.

Роль у розробці системи оцінки знань

– централізоване управління даними: Використання ASP.NET Web API дозволяє централізовано управляти навчальними матеріалами, тестами та оцінками, забезпечуючи єдину точку доступу до даних;

– інтеграція з іншими системами: Фреймворк легко інтегрується з іншими освітніми платформами, системами управління базами даних та аналітичними інструментами, що дозволяє розширювати функціональність та можливості системи.

Безпека та надійність

– безпечне підключення: ASP.NET Web API забезпечує механізми для захисту комунікації між сервером та клієнтами, включаючи шифрування та аутентифікацію;

– стабільність системи: Висока надійність фреймворку гарантує стабільність роботи системи оцінки знань, мінімізуючи ризики збоїв та перебоїв.

Майбутні перспективи та інновації

– інтеграція з хмарними сервісами: Розвиток хмарних технологій відкриває нові можливості для масштабування та оптимізації серверних застосунків на основі ASP.NET Web API;

– використання штучного інтелекту: Інтеграція алгоритмів ШІ може значно розширити функціональність систем оцінки знань, зокрема у сфері аналітики та персоналізації навчального процесу.

Роль ASP.NET Web API у створенні серверних застосунків для систем оцінки знань є визначальною. Цей фреймворк не лише забезпечує високу продуктивність, масштабованість та безпеку, але й відкриває шлях для інтеграції новітніх технологій, що сприяє створенню більш ефективних та адаптивних освітніх інструментів.

3.2.1 Принцип роботи ASP.NET Web API

За допомогою даного фреймворку створюються сервери на основі HTTP, до яких можна звернутися з різних платформ: браузер, операційна система Windows, мобільні пристрої.

Властивості ASP.NET Web API:

- побудований на основі ASP.NET;
- підтримує конвеєр запитів/відповідей ASP.NET;
- платформа для створення RESTful служб;
- відповіді на запити у різних форматах (JSON, XML, BSON);
- зіставляє імена методів з назвами HHTP запитів;
- може бути розгорнутий на власному хості, на IIS, на іншому сервері [8].

На рисунку 3.1 показана взаємодія серверу, написаному на ASP.NET Web API, з клієнтом.

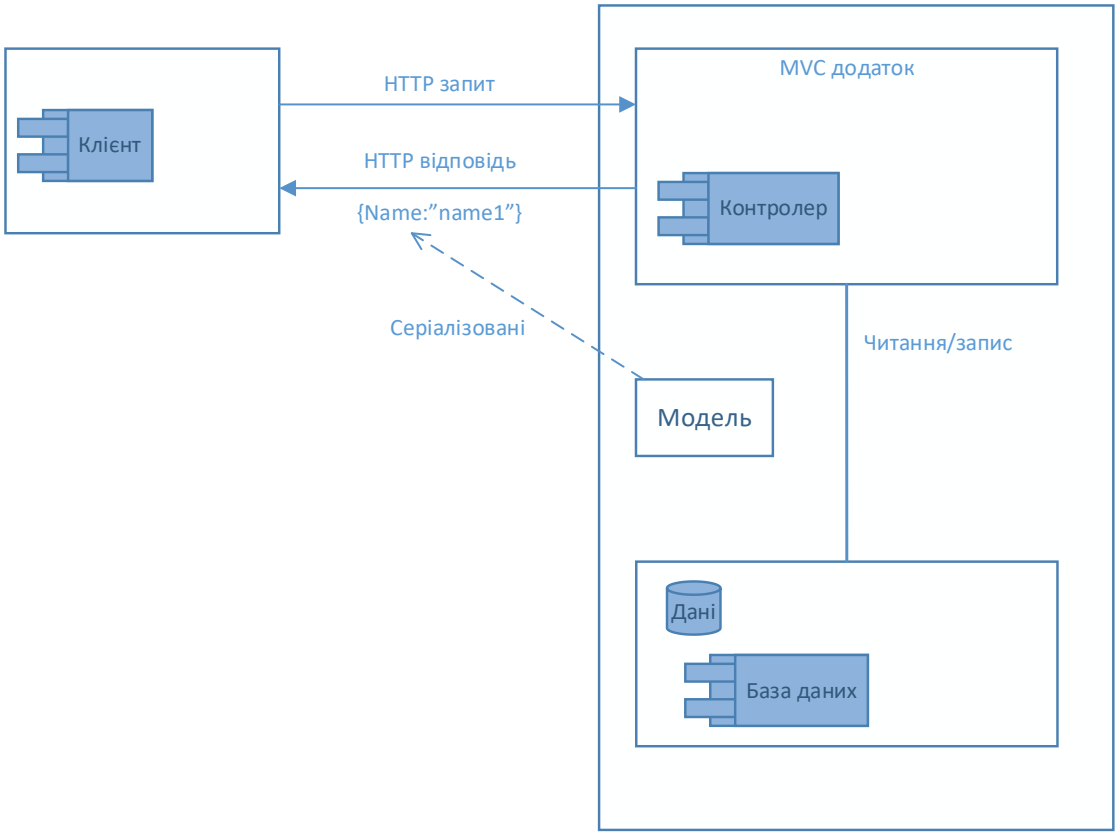

Рисунок 3.1 – Шаблон ASP.NET Web API

Клієнт надсилає запит серверу, контролер серверу виконує операції з читання та запису даних в сховищі даних, виконує маніпуляції з даними, та надає відповідь у вигляді серіалізованих даних.

Основними компонентами системи є:

1. Модель – набір класів, які представляють дані зі сховища даних та керуються контролерами.

2. Контекст бази даних – основний клас, який є похідним від класу Microsoft.EntityFrameworkCore.DbContext та координує функціональність Entity Framework для моделі даних.

Контролер – клас, який оброблює HTTP запити. Методи цього класу оброблюють GET, POST, PUT, DELETE та інші запити, та надають відповідь на той чи інший запит.

3.2.2 Loosely Coupled Monolith

В процесі розробки проєкту була обрана концепція розробки, відома як слабко зв’язаний моноліт, яка передбачає розділення загального функціоналу системи на відокремлені підсистеми. Цей підхід дозволяє забезпечити незалежність компонентів системи один від одного, спрощуючи тим самим процес інтеграції та розширення системи. Кожен модуль в системі розроблено з урахуванням конкретної функціональної області та включає три основні компоненти: договори (Contracts), які містять інтерфейси та об'єкти передачі даних (DTO); реалізацію (Implementation), де знаходиться весь функціонал модуля; та модульні тести (Tests) для перевірки коректності роботи кожного модуля (рисунок 3.2).

Рисунок 3.2 – Структура проєкту

Одним із головних контекстів системи є контекст «Тест». Структура проєкту виглядає наступним чином (рисунок 3.3):

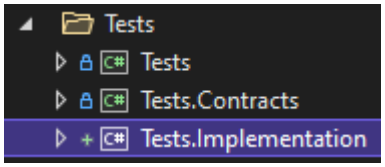

Рисунок 3.3 – Проєкт Tests

Ключовим аспектом такої архітектури є забезпечення взаємодії між різними модулями через систему обміну повідомленнями, що дозволяє

підтримувати їх зв'язок, не порушуючи принципів незалежності та ізоляції. Це також уможливлює локалізацію змін в межах одного модуля без впливу на решту системи, спрощуючи процес розширення та модифікації системи.

Загалом структура системи виглядає наступним чином (рисунок 3.4).

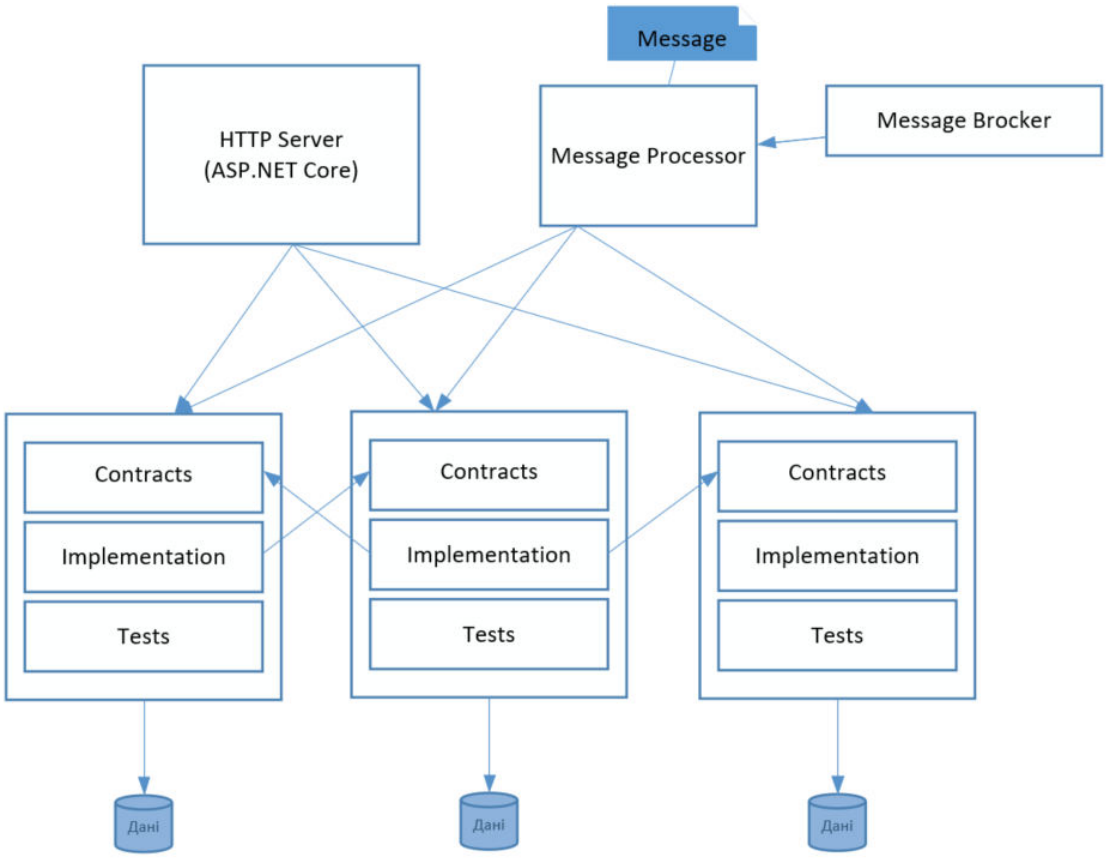

Рисунок 3.4 – Структура системи

Слабко зв’язаний моноліт підтримує ідею одночасного збереження переваг монолітної архітектури, таких як спрощене розгортання та налагодження, та надає гнучкість розподіленої системи за рахунок декомпозиції на модулі. Проте, слід зазначити, що такий підхід має і свої недоліки, зокрема, збільшення складності управління залежностями та потенційні ризики при розгортанні, але загалом для розглянутої системи ця архітектура виявилась оптимальним вибором.

3.2.3 Entity Framework – ORM та міграції бази даних

Для забезпечення гнучкості обробки даних і мінімізації помилок при роботі з ними, у проєкті використовуються технології ORM (Object-Relational Mapping) та система міграції баз даних. ORM дозволяє взаємодіяти з базою даних на вищому рівні абстракції, використовуючи об'єктно-орієнтований підхід, що спрощує роботу розробників. Міграції баз даних, з іншого боку, надають змогу керувати змінами в структурі бази даних, забезпечуючи контроль версій і можливість відстеження історії змін.

Однією з ключових переваг цього підходу є забезпечення цілісності даних, оскільки кожна міграція відображає точний набір операцій, які були виконані для оновлення схеми бази даних. Entity Framework, підтримуваний Microsoft, був обраний як ORM-рішення для даного проєкту через його гнучкість та потужні можливості. Після кожної міграції створюється артефакт, що фіксує зміни, надаючи нову версію бази даних з можливістю збереження попереднього стану (рисунок 3.5).

Після виконання міграції генерується артефакт зі змінами, які були виконанні.

```
ссылка: 1
public partial class InitialCreate : Migration
{
    Ссылок: 0
    protected override void Up(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.EnsureSchema(
            name: "ResultDBContext");

        migrationBuilder.CreateTable(
            name: "Results",
            schema: "ResultDBContext",
            columns: table => new
            {
                Id = table.Column<int>(type: "int", nullable: false)
                    .Annotation("SqlServer:Identity", "1, 1"),
                ExternalID = table.Column<string>(type: "nvarchar(max)", nullable: false),
                UserExternalID = table.Column<string>(type: "nvarchar(max)", nullable: false),
                TestExternalID = table.Column<string>(type: "nvarchar(max)", nullable: false),
                PassDate = table.Column<DateTime>(type: "datetime2", nullable: false),
                AttemptCount = table.Column<int>(type: "int", nullable: false),
                PercentSuccess = table.Column<float>(type: "real", nullable: false)
            },
            constraints: table =>
            {
                table.PrimaryKey("PK_Results", x => x.Id);
            });
```

Рисунок 3.5 – Артефакт міграції

Таким чином, кожна міграція надає нову версію БД, зі збереженням стану минулої версії.

Використання ORM спрощує адаптацію до змін у джерелі даних, дозволяючи внести необхідні зміни лише в ORM-конфігурацію, не змінюючи код, який взаємодіє з даними. Це забезпечує гнучкість системи, сприяє легкому її розширенню та полегшує підтримку коду.

3.2.4 Фреймворк Angular

Фронтенд частина проєкту була розроблена з використанням фреймворку Angular, що є популярним рішенням для створення динамічних і інтерактивних вебзастосунків. Angular надає великий набір інструментів для розробки, що дозволяє ефективно управляти станом додатку, маршрутизацією та взаємодією з сервером. Для програмування використовувалась мова TypeScript, яка розширює можливості JavaScript, додаючи типізацію та інші функції, які сприяють написанню більш безпечного та легко підтримуваного коду (рисунок 3.6).

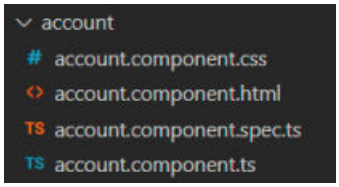

Рисунок 3.6 – Структура компоненту Angular

Розмітка сторінок здійснювалась за допомогою HTML, а стилізація – за допомогою CSS, що є стандартним підходом у веброзробці. Для покращення візуального представлення і користувацького досвіду було застосовано технологію Angular Material, яка надає готові компоненти дизайну, що відповідають принципам матеріального дизайну. Це дозволило створити інтуїтивно зрозумілий і візуально привабливий інтерфейс. Для стилізації також використовувався фреймворк Tailwind CSS, який надає велику кількість утилітарних класів для швидкого створення адаптивних макетів.

Принцип роботи Tailwind наступний (рисунок 3.7): в проєкті створюється вхідний файл, в який імпортуються компоненти фреймворку. Також у цьому файлі можна писати власний код CSS. Після цього Tailwind обробляє цей вхідний файл та під час збірки генерує код CSS у вихідному файлі, з яким пов’язаний файл HTML.

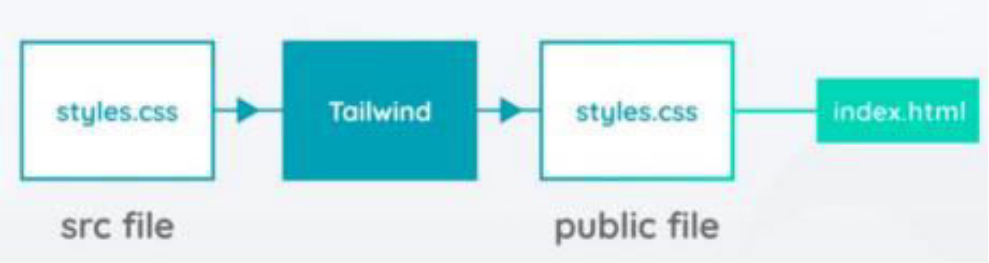

Рисунок 3.7 – Схема роботи Tailwind CSS

Tailwind CSS є утилітарним першим CSS-фреймворком, який дозволяє швидко створювати дизайни безпосередньо в HTML-файлах. Ось деякі ключові переваги Tailwind CSS:

1. Ефективність розробки: завдяки утилітному підходу, Tailwind дозволяє розробникам швидко застосовувати стилі, використовуючи класи прямо в HTML, що значно пришвидшує процес розробки.

2. Гнучкість: Tailwind не нав'язує жорстких дизайн-шаблонів, як деякі інші CSS-фреймворки, дозволяючи розробникам легко створювати унікальні дизайни без необхідності переписування багато CSS-коду.

3. Відгукність «з коробки»: Tailwind містить набір утиліт для створення відгукніх дизайнів, дозволяючи легко застосовувати різні стилі для різних розмірів екранів з використанням префіксів класів.

4. Консистентність дизайну: використання конфігураційних файлів у Tailwind CSS дозволяє встановити дизайн-систему (наприклад, кольори, шрифти, відступи), що забезпечує консистентність дизайну по всьому проєкту.

5. Оптимізація продуктивності: Tailwind автоматично видаляє не використовувані стилі при побудові продакшн-версії проєкту, що призводить

до менших розмірів кінцевих CSS-файлів і покращує продуктивність завантаження сторінки.

6. Спільнота і екосистема: Tailwind має велику і активну спільноту, яка створює безліч плагінів, інструментів і компонентів, що розширюють можливості фреймворку.

Ці переваги роблять Tailwind CSS популярним вибором серед розробників фронтенду, особливо тих, хто цінує швидкість розробки та гнучкість у дизайні.

3.3 Принципи та методики аналізу ефективності тестів

Аналіз ефективності тестів є фундаментальним аспектом у розробці та вдосконаленні систем оцінки знань. Ефективний тест повинен не тільки точно оцінювати знання студентів, але й стимулювати навчальний процес, сприяти засвоєнню матеріалу та виявляти потреби в подальшому навчанні. Нижче наведено ключові принципи та методики, що використовуються для аналізу ефективності тестів:

1. Валідність відноситься до ступеня, в якому тест точно вимірює те, що він призначений вимірювати. Для аналізу валідності можуть використовуватися такі методи:

– змістовна валідність: перевірка, чи тестові питання відображають весь спектр знань, які повинні бути оцінені;

– конструктивна валідність: оцінка, чи тест вимірює теоретичні конструкції або поняття, які він призначений вимірювати;

– критерійна валідність: аналіз кореляції результатів тесту з іншими, вже визнаними мірками оцінки знань або здібностей.

2. Оцінка надійності тестів.

Надійність визначає ступінь, до якого результати тесту консистентні та відтворювані при повторному тестуванні. Для вимірювання надійності використовуються:

– внутрішня узгодженість: міра, що показує, наскільки питання тесту пов'язані між собою і вимірюють один і той же конструкт;

– тест–ретест: повторне проведення тесту з тією ж групою респондентів після певного періоду для оцінки стабільності результатів;

– альтернативні форми: створення двох різних версій тесту, які вимірюють те саме, і порівняння результатів між ними.

3. Аналіз рівня складності питань.

Рівень складності питань визначається через аналіз відсотка правильних відповідей на кожне питання. Питання, на які відповідає велика частина студентів, вважаються менш складними. Аналіз дозволяє виявити питання, які можуть бути надто легкими або занадто складними, та відповідно їх коригувати.

4. Оцінка дискримінаційної здатності.

Дискримінаційна здатність питань вказує на те, наскільки добре питання розрізняє студентів з високим і низьким рівнем знань. Питання з високою дискримінаційною здатністю дозволяють відділити студентів, які добре засвоїли матеріал, від тих, хто має проблеми.

5. Зворотний зв'язок і аналітика.

Збір та аналіз зворотного зв'язку від студентів і викладачів щодо тестів дозволяє виявляти потенційні проблеми та недоліки в тестових матеріалах. Використання аналітичних інструментів допомагає в агрегуванні та інтерпретації даних для прийняття обґрунтованих рішень щодо покращення тестів.

Аналіз ефективності тестів за допомогою зазначених методик та принципів є невід'ємною частиною процесу вдосконалення систем оцінки знань. Валідність, надійність, складність, дискримінаційна здатність та зворотний зв'язок формують основу для розробки ефективних, справедливих та мотивуючих тестів, спрямованих на підвищення якості освіти.

3.3.1 Розробка алгоритму оцінки ефективності тестів

Розробка алгоритму оцінки ефективності тестів є критичною складовою систем автоматизованого оцінювання знань. Цей алгоритм аналізує відповіді користувачів для визначення, наскільки добре тестові питання вимірюють знання та розуміння студентами навчального матеріалу.

Алгоритм оцінки ефективності тестів включає наступні ключові компоненти:

1. Аналіз рівня складності питань: використовується відсоток правильних відповідей на кожне питання для визначення його складності.

Реалізація методу

Збір даних: дані про кожне тестове питання зберігаються в базі даних, включаючи загальну кількість відповідей та кількість правильних відповідей від усіх учасників які пройшли тест.

Розрахунок відсотку правильних відповідей: для кожного питання розраховується відсоток правильних відповідей як відношення кількості правильних відповідей до загальної кількості відповідей (рисунок 3.8).

```
double CalculateDifficultyPercentage(int correctAnswers, int totalAnswers)
{
    if (totalAnswers == 0) return 0; // Уникнення ділення на нуль
    return (double)correctAnswers / totalAnswers * 100;
}
```

Рисунок 3.8 – Метод визначення складності тесту

Класифікація складності: на основі відсотка правильних відповідей класифікуються питання за рівнем складності.

– від 0% до 30%: висока складність;

– від 31% до 60%: середня складність;

– від 61% до 100%: низька складність.

```
string ClassifyDifficulty(double percentage)
{
    if (percentage <= 30) return "Висока складність";
    else if (percentage <= 60) return "Середня складність";
    else return "Низька складність";
}
```

Рисунок 3.9 – Класифікація складності

Аналіз і внесення змін: отримані дані використовуються для аналізу ефективності питань. Питання з високою складністю можуть вимагати перегляду або переформулювання, щоб зробити їх більш зрозумілими для студентів.

2. Дискримінаційна здатність: оцінює, наскільки добре питання розрізняє студентів з різним рівнем знань.

Реалізація методу

Групування учасників: усіх учасників тестування розділено на дві групи за результатами їх загальної успішності: вищу групу (наприклад, 27% учасників з найвищими балами) та нижчу групу (27% учасників з найнижчими балами).

Обчислення відсотку правильних відповідей: для кожного тестового питання обчислюється відсоток учасників з вищої та нижчої груп, які дали на нього правильну відповідь.

Розрахунок дискримінаційного індексу: дискримінаційний індекс кожного питання можна розрахувати як різницю між відсотком правильних відповідей у вищій групі та відсотком правильних відповідей у нижчій групі.

Аналіз результатів: питання з високим дискримінаційним індексом добре розрізняють учасників з різним рівнем знань. Питання з низьким або від'ємним індексом можуть бути недостатньо якісними або неефективними для оцінки знань і, можливо, потребуватимуть перегляду або видалення.

Реалізація обчислення дискримінаційної здатності навдена в додатку А.

3. Аналіз варіативності відповідей: вивчає розподіл відповідей серед студентів для ідентифікації потенційно спірних або неоднозначних питань.

Метод базується на підрахунку кількості кожної з відповідей на певне питання та аналізу цих даних (рисунок 3.10).

```
public List<VariabilityAnalysis> AnalyzeAnswerVariability(List<TestQuestion> questions)
{
    var analyses = new List<VariabilityAnalysis>();
    foreach (var question in questions)
    {
        var analysis = new VariabilityAnalysis { QuestionId = question.QuestionId };
        foreach (var answer in question.Answers)
        {
            if (analysis.AnswerDistribution.ContainsKey(answer.Answer))
                analysis.AnswerDistribution[answer.Answer] += answer.Count;
            else
                analysis.AnswerDistribution.Add(answer.Answer, answer.Count);
        }

        analyses.Add(analysis);
    }

    return analyses;
}
```

Рисунок 3.10 – Метод аналізу варіативності відповідей

Метод приймає список питань і аналізує варіативність відповідей для кожного питання, використовуючи словник для відстеження кількості кожної відповіді.

4. Оцінка часу, витраченого на питання: аналізує середній час, який студенти витрачають на кожне питання, що може вказувати на його складність або недоліки у формулюванні (рисунок 3.11).

```
public List<QuestionTimeAnalysis> AnalyzeTimeSpent(List<TestQuestion> questions)
{
    var analyses = new List<QuestionTimeAnalysis>();
    foreach (var question in questions)
    {
        if (question.Attempts.Count > 0)
        {
            var averageTimeSpent = TimeSpan.FromTicks
                ((long)question.Attempts.Average(a => a.TimeSpent.Ticks));

            analyses.Add(new QuestionTimeAnalysis
            {
                QuestionId = question.QuestionId,
                AverageTimeSpent = averageTimeSpent
            });
        }
    }
    return analyses;
}
```

Рисунок 3.11 – Оцінка часу, витраченого на питання

Метод збирає дані про час, який кожен студент витрачає на кожне тестове питання, а потім обчислює середні значення для кожного питання.

Після розробки алгоритм інтегрується в систему оцінки знань, де він автоматично аналізує відповіді студентів після кожного тесту. Результати аналізу надають викладачам детальний звіт про ефективність кожного тестового питання, дозволяючи їм вносити необхідні корективи для покращення якості тестів.

Розробка та впровадження алгоритму оцінки ефективності тестів значно підвищує якість системи оцінки знань, забезпечуючи більш точне та об'єктивне вимірювання студентських знань. Використання такого алгоритму дозволяє викладачам ідентифікувати та оптимізувати тестові питання, що сприяє підвищенню ефективності навчального процесу.

3.4 Інтеграція штучного інтелекту для оптимізації процесу створення тестів

В даному розділі розглянуто перспективні можливості, які відкриваються перед системами оцінки знань завдяки інтеграції штучного інтелекту (ШІ). Сучасний прогрес у галузі ШІ та машинного навчання надає унікальні інструменти для покращення процесу створення, аналізу та адаптації тестових завдань, що може значно збільшити ефективність та адаптивність оцінювання знань у навчальних системах.

У традиційному процесі створення тестів викладачі стикаються з численними викликами, включаючи об'єктивність питань, їх релевантність до навчальних цілей, та здатність адекватно оцінювати рівень знань студентів. Штучний інтелект може сприяти оптимізації цього процесу, забезпечуючи високу якість тестових матеріалів та їхню адаптацію до індивідуальних потреб студентів.

Інтеграція ШІ у процес створення тестів відкриває широкий спектр можливостей:

– автоматизація генерації питань: ШІ може автоматично генерувати тестові питання на основі навчального контенту, враховуючи попередньо визначені навчальні цілі та ключові концепції;

– адаптивні тести: штучний інтелект дозволяє створювати адаптивні тести, що динамічно змінюють свій склад і рівень складності залежно від відповідей та прогресу студента;

– аналіз та оптимізація питань: застосування алгоритмів машинного навчання для аналізу ефективності тестових питань та їхньої оптимізації з метою покращення якості оцінювання.

Інтеграція ШІ в процес створення тестів також ставить перед розробниками нові виклики, зокрема забезпечення точності та надійності ШІ–генерованих питань, збереження академічної чесності та етичних стандартів, а також управління великими обсягами даних та їх захист.

Розвиток технологій штучного інтелекту та їх інтеграція в освітній процес відкриває шлях до створення нового покоління навчальних систем, здатних забезпечувати високоіндивідуалізоване та ефективне навчання.

3.4.1 Використання ШІ для підтримки формулювання питань тестів

Інтеграція штучного інтелекту (AI) для підтримки формулювання питань тестів є новаторським підходом, що відкриває нові можливості для покращення якості та ефективності оцінювання в освітніх системах. Штучний інтелект може допомогти викладачам та розробникам тестів генерувати питання, які краще відповідають навчальним цілям, а також адаптовані до рівня знань і потреб студентів.

Алгоритм підтримки формулювання питань.

1. Аналіз навчального матеріалу – AI проходить через навчальний контент, використовуючи методи обробки природної мови (NLP) для ідентифікації ключових концепцій, термінів та важливих тем [14].

2. Визначення навчальних цілей – алгоритм аналізує навчальні цілі курсу або модуля для забезпечення відповідності генерованих питань цим цілям.

3. Генерація варіантів питань – на основі аналізу контенту та навчальних цілей, AI генерує кілька варіантів питань, кожне з яких відображає різні аспекти засвоєного матеріалу.

4. Оцінка складності та релевантності – AI оцінює рівень складності кожного питання та його релевантність до навчальних цілей, адаптуючи формулювання для оптимального охоплення теми.

5. Відбір та Оптимізація Питань – на останньому етапі алгоритм відбирає найкращі питання та оптимізує їх формулювання, забезпечуючи чіткість, однозначність і високу дидактичну цінність.

Алгоритм штучного інтелекту для формулювання питань тестів відкриває нові горизонти в освітній оцінці, пропонуючи значні переваги та інноваційні підходи для покращення навчального процесу:

– ефективність та часова економія – розроблений алгоритм значно економить час викладачів та розробників тестів, автоматизуючи процес генерації питань. Це дозволяє освітнім працівникам зосередитися на інших аспектах навчання та взаємодії зі студентами, підвищуючи загальну ефективність навчального процесу;

– підвищення якості навчального матеріалу – штучний інтелект забезпечує високу якість питань тестів шляхом їх ретельного формулювання та адаптації до конкретних навчальних цілей. Використання алгоритмів NLP для аналізу контенту гарантує, що кожне питання є релевантним та цінним для оцінювання розуміння студентами матеріалу.

– адаптація до потреб студентів – однією з ключових інновацій алгоритму є його здатність адаптувати тести до різних рівнів знань студентів. Це досягається за рахунок аналізу складності та дискримінаційної здатності питань, що забезпечує індивідуалізоване та справедливе оцінювання.

– забезпечення об'єктивності – алгоритм мінімізує суб'єктивний вплив у процесі створення тестів, забезпечуючи об'єктивність оцінювання. Автоматизований підхід до формулювання питань допомагає уникнути помилок та неточностей, що можуть виникати при ручному складанні тестів.

– інноваційний підхід до навчання – застосування цього алгоритму є інноваційним кроком у розвитку освітніх технологій, відкриваючи нові можливості для розробки адаптивних та інтерактивних навчальних матеріалів. Це сприяє підвищенню залученості студентів та ефективності навчання.

Основні компоненти функціоналу.

1. Аналітична система: використання алгоритмів обробки природної мови (NLP) та машинного навчання для аналізу навчального контенту та визначення ключових концептів і тем.

2. Генератор підказок: система пропонує викладачам різноманітні формулювання питань, враховуючи аналізовані концепції та навчальні цілі.

3. Інтерактивний інтерфейс: викладачі отримують доступ до інтерактивного інтерфейсу, де вони можуть переглядати підказки AI, модифікувати їх за потреби та інтегрувати в тести.

Етапи реалізації:

– аналіз контенту та навчальних цілей – AI проходить через навчальний матеріал, ідентифікуючи ключові теми та концепти, що мають бути включені в тести;

– генерація підказок – на основі аналізу контенту, AI генерує варіанти формулювань питань, які відповідають навчальним цілям та академічним стандартам;

– взаємодія з викладачами – викладачі переглядають та вибирають підказки, що найкраще відповідають змісту та цілям тесту, вносячи необхідні корективи;

– оцінка та оптимізація – використання зворотного зв'язку від викладачів та аналізу ефективності тестів для подальшого вдосконалення алгоритму генерації підказок.

Фрагменти реалізації алгоритму наведено в додатку А.

Розробка функціоналу AI–підказок для викладачів при створенні тестів є значним кроком у напрямку покращення освітньої оцінки. Цей підхід не тільки спрощує процес розробки тестів, але й забезпечує високу якість та адаптивність питань, сприяючи глибшому засвоєнню знань студентами та забезпечуючи більш точне оцінювання їхніх навчальних досягнень.

У підсумку, розробка та інтеграція алгоритму ШІ для формулювання питань тестів представляє значний крок вперед у підвищенні якості та ефективності освітньої оцінки. Цей підхід не лише оптимізує процес створення тестів, але й сприяє створенню більш адаптивних та індивідуалізованих навчальних досвідів для студентів, відкриваючи нові горизонти для розвитку сучасної освіти.

Висновки до розділу 3

Вибір програмних засобів розробки є критично важливим етапом у створенні ефективної системи оцінки знань. Використання клієнт-серверної архітектури дозволяє розділити обробку даних та представлення, що сприяє гнучкості та масштабованості системи. Клієнт-серверні архітектури відіграють ключову роль у системах оцінки знань, оскільки вони забезпечують ефективне взаємодію між користувачами та серверами, а також дозволяють централізовано зберігати та обробляти тестові дані.

Програмна реалізація системи за допомогою ASP.NET Web API, Loosely Coupled Monolith, Entity Framework та Angular демонструє важливість вибору сучасних та гнучких технологій для створення масштабованої та легко підтримуваної системи. Використання Entity Framework як ORM інструменту

спрощує роботу з базами даних та міграції, забезпечуючи ефективність та безпеку управління даними.

Розробка алгоритму оцінки ефективності тестів та інтеграція модуля аналітики є ключовими для забезпечення постійного вдосконалення якості тестування та забезпечення об'єктивності оцінки знань. Інтеграція штучного інтелекту для оптимізації процесу створення тестів відкриває нові можливості для підвищення ефективності та адаптивності систем оцінки знань, дозволяючи автоматизувати формулювання питань та аналізувати ефективність тестів на основі великих даних.

4 ОЛЯД ІНТЕРФЕЙСУ ТА ПОСІБНИК КОРИСТУВАЧА

4.1 Макети інтерфейсу

Після аналізу літератури, визначення об'єкта, предмета та методів дослідження розроблено програмний продукт із зазначеною вище функціональністю. Система отримала назву "Test Your Brain".

Цей продукт є результатом глибокого дослідження сучасних потреб у сфері освіти та тестування знань. "Test Your Brain" не просто спрощує процес оцінювання знань, але й робить його більш залученим і інтерактивним для користувачів. Використовуючи інноваційні технології, система пропонує гнучкі налаштування тестів, автоматизований збір та аналіз статистичних даних, що дозволяє освітнім закладам не тільки оцінювати знання студентів, а й адаптувати навчальний процес під їх потреби. "Test Your Brain" відкриває нові горизонти для розвитку освітнього процесу, роблячи навчання більш ефективним і захоплюючим.

4.1.1 Авторизація

Для повного доступу до функціоналу системи користувачам необхідно увійти до системи (рисунок 4.1(a)). Ті, хто ще не мають облікового запису, можуть легко зареєструватися (рисунок 4.1(b)і).

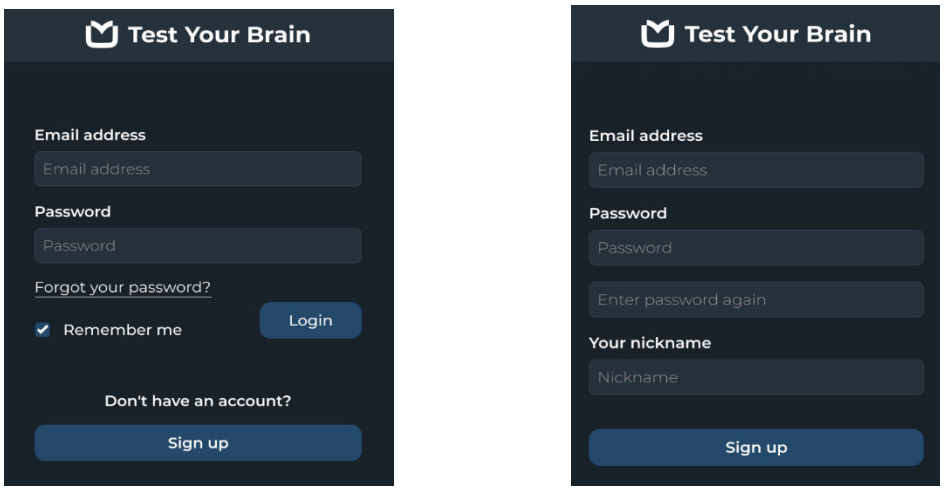

Рисунок 4.1 – Авторизація

Після реєстрації користувачі системи мають можливість додатково вказати інформацію на сторінці облікового запису – ім'я, прізвище, країну. Є також опція додавання аватара, зміни нікнейму та паролю (рисунок 4.i2).

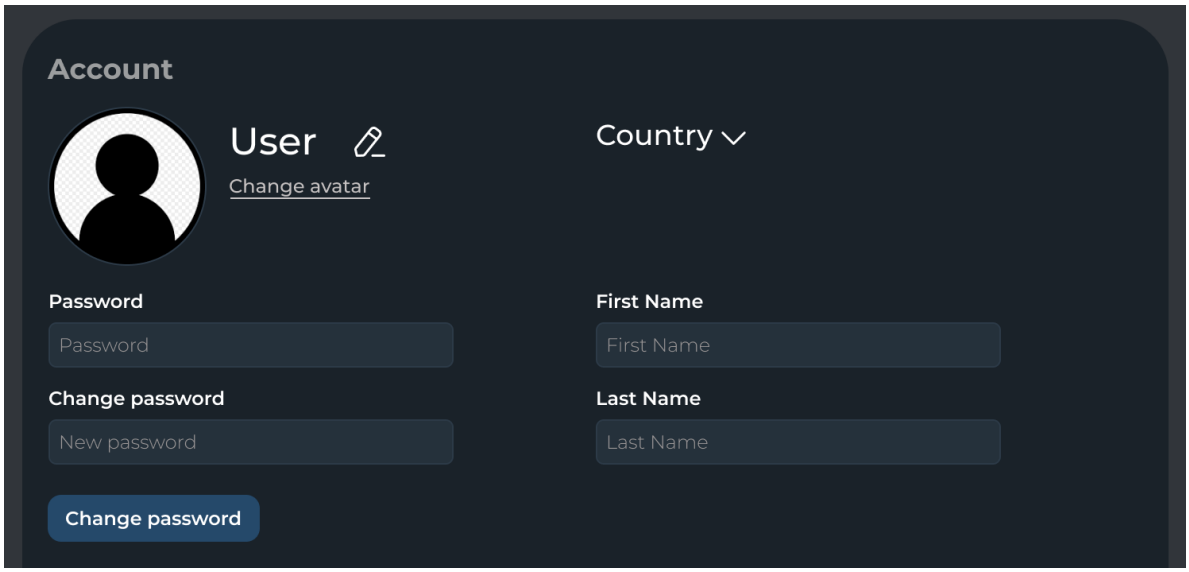

Рисунок 4.2 – Інформація користувача

Ці дані будуть корисними для користувачів, для яких важливо бачити інформацію про тих, хто проходить тестування. Можливість налаштування профілю додає рівень впізнаваності та взаємозв'язку між учасниками, що може сприяти створенню спільноти навколо навчального процесу. Враховуючи, що сучасна освіта все більше залежить від взаємодії та колаборації, наявність добре розробленої системи профілів може значно покращити досвід користувачів із "Test Your Brain".

4.1.2 Панель навігації

Після входу до системи користувач отримує доступ до повного функціоналу системи через панель навігації (рисунок 4.3).

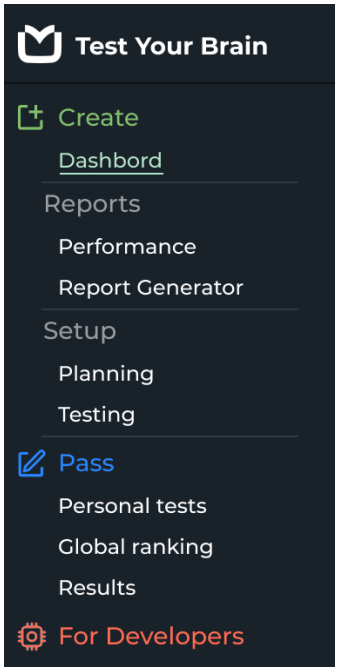

Рисунок 4.3 – Панель навігації

Панель поділена на три розділи:

1. Розділ створення тестів. Закладка «Панель керування» дозволяє створювати тест, редагувати його та видаляти, а також переглядати список створених тестів. Відділ «Звіти» надає дані про результати користувачів, які пройшли тест. Відділ «Налаштування» керує статусом тесту, планує час тестування, дозволяє попередньо переглянути тест для перевірки його зручності.

2. Розділ проходження тестів. Вкладка «Персональні тести» показує інформацію про тести, до яких користувач отримав доступ. Вкладка «Результати» містить інформацію про результати пройдених тестів, а вкладка «Глобальний рейтинг» відображає інформацію про найуспішніших користувачів за кількістю та якістю пройдених тестів.

3. Розділ для розробників містить API, який дозволяє розробникам налаштовувати тести.

Ця структура навігаційної панелі ретельно розроблена таким чином, щоб забезпечити інтуїтивно зрозуміле та ефективне керування процесом навчання, від створення та планування тестів до аналізу досягнень учнів. Це забезпечує

користувачам гладкий і продуктивний досвід використання "Test Your Brain", сприяючи їхньому академічному зростанню і мотивації.

4.1.3 Головна панель

Панель призначена для перегляду та управління даними створених тестів (рисунок 4.4).

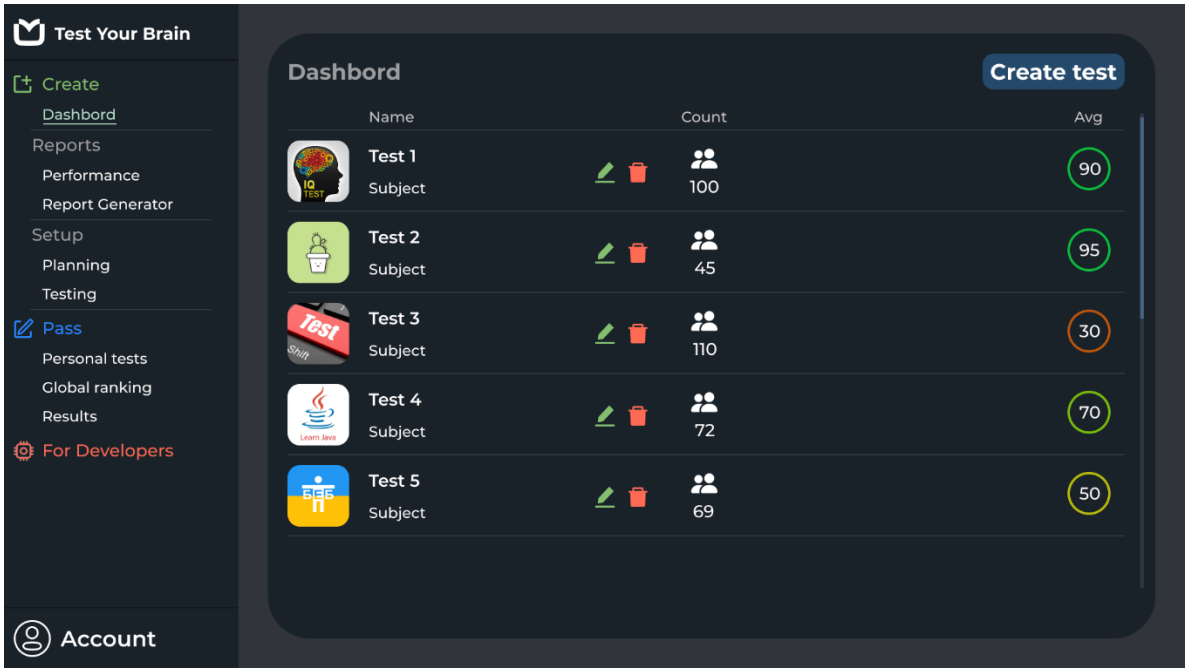

Рисунок 4.4 – Головна панель

На панелі керування можна управляти тестами: їх створювати, редагувати або видаляти. Ця сторінка надає табличну інформацію про кількість людей, які пройшли тест, та середній бал тих, хто пройшов тест.

Панель керування "Test Your Brain" є центральним інструментом для викладачів і тренерів, який забезпечує організований перегляд важливих метрик ефективності тестування. Це може допомогти у виявленні тенденцій та областей для покращення, а також у відстеженні прогресу здобувачів освіти в часі, що є ключовим для розвитку адаптивних навчальних стратегій.

4.1.4 Панель створення тесту

Створення та редагування тестів виконується на відповідній сторінці (рисунок 4.5).

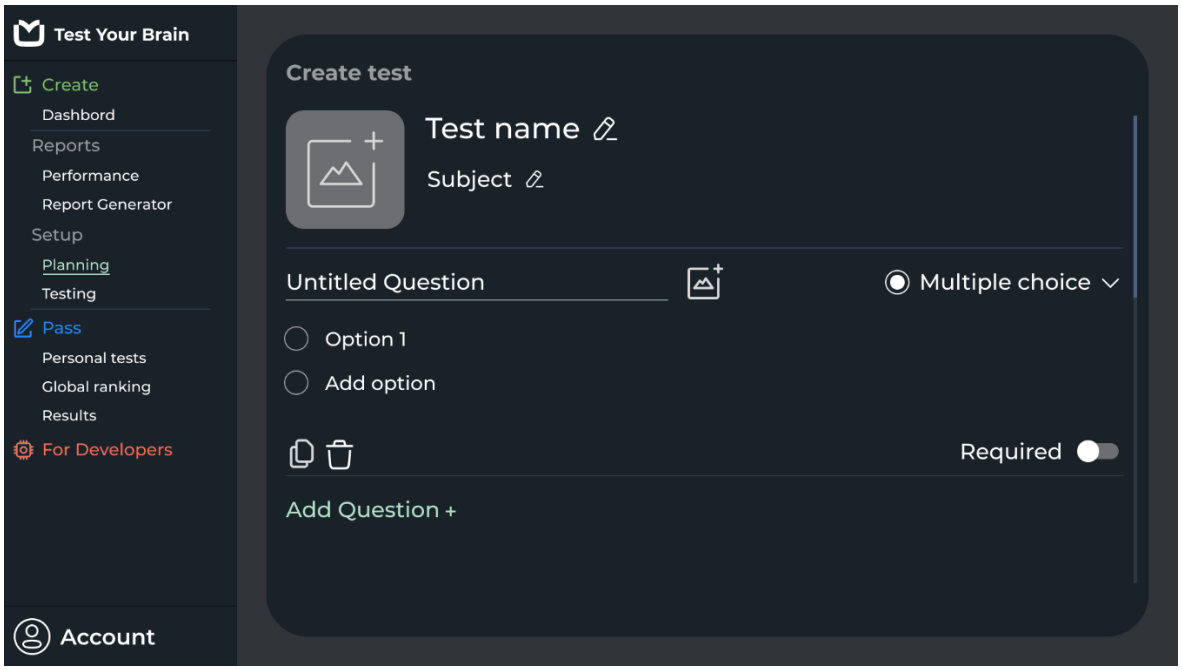

Рисунок 4.5 – Панель створення тесту

Тест має такі властивості: назву, предмет, іконку та список запитань. Кожне питання має свої властивості: воно може бути обов'язковим або необов'язковим, мати одну або декілька правильних відповідей. Ця сторінка надає функціональність для створення та редагування питань тесту та інформації.

Ці властивості підкреслюють гнучкість "Test Your Brain" у підході до створення тестів, що дозволяє викладачам або розробникам курсів налаштувати оцінювання відповідно до конкретних вимог навчальної програми або цілей оцінювання. Платформа також підтримує створення тестів з різноманітними типами запитань, що робить процес тестування більш цікавим та забезпечує краще вимірювання різних аспектів знань та навичок учнів.

4.1.4 Панель статистики

Панель дозволяє переглядати статистику результатів тих, хто пройшов тест, у графічному форматі (рисунок 4.6).

Рисунок 4.6 – Панель статистики

Аналіз ефективності створених тестів проводиться через розділ звітів. Можливе фільтрування за часом, перегляд інформації про конкретний тест та тести в цілому, перегляд кількості користувачів, які пройшли тест, розбитих за днями, перегляд середнього балу за тести та їх порівняння між собою.

Розробник тесту може перейти на сторінку Performance, встановити часовий діапазон для перегляду статистики, вибрати тести, інформацію про які він бажає побачити на графіку. Система забезпечує оновлений графік успішності за обраний період, де видно динаміку зміни середнього балу по тестах. Така інформація може бути використана розробником для покращення змісту тестів та порівняння результатів різних тестів.

4.1.5 Розділ налаштувань

Розділ налаштувань надає функціональність для редагування статусу тесту та його попереднього перегляду. Вкладка «Планування» виглядає так (рисунок 4.7).

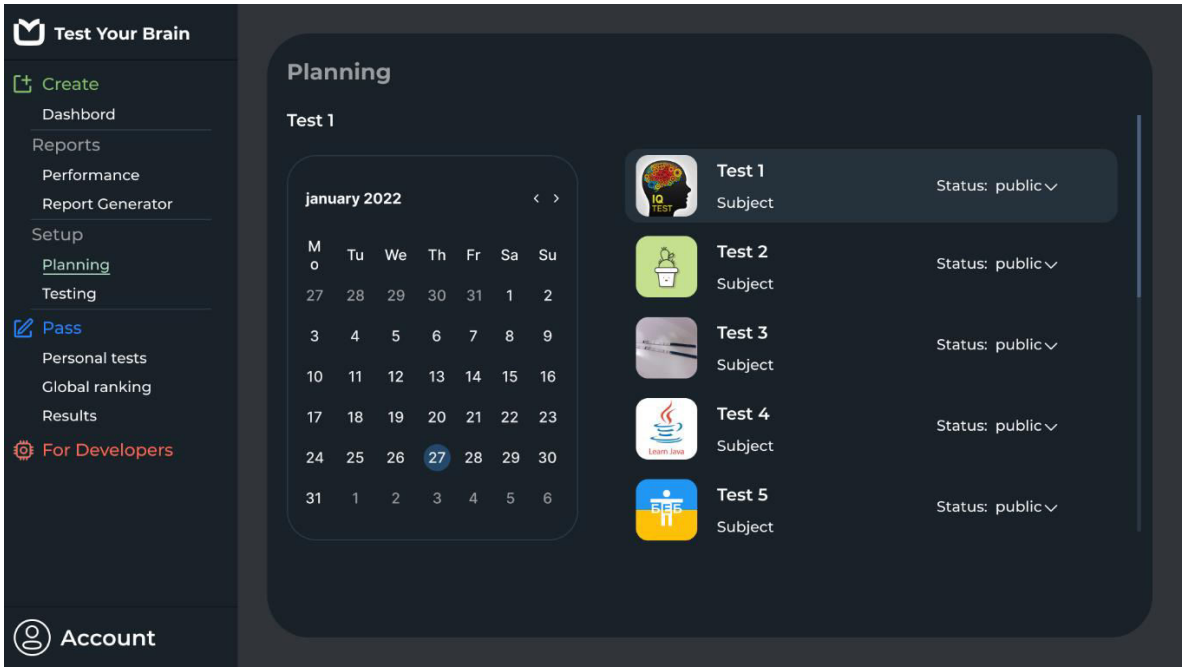

Рисунок 4.7 – Розділ налаштувань

На цій сторінці представлений список тестів із їх статусом: публічний або приватний. Публічний тест видимий для всіх користувачів, приватний - лише для тих, кому надано доступ. Кожному тесту призначено часовий проміжок, коли тест доступний для проходження.

Вкладка «Тестування» призначена для перевірки якості тесту шляхом його проходження (рисунок 4.8).

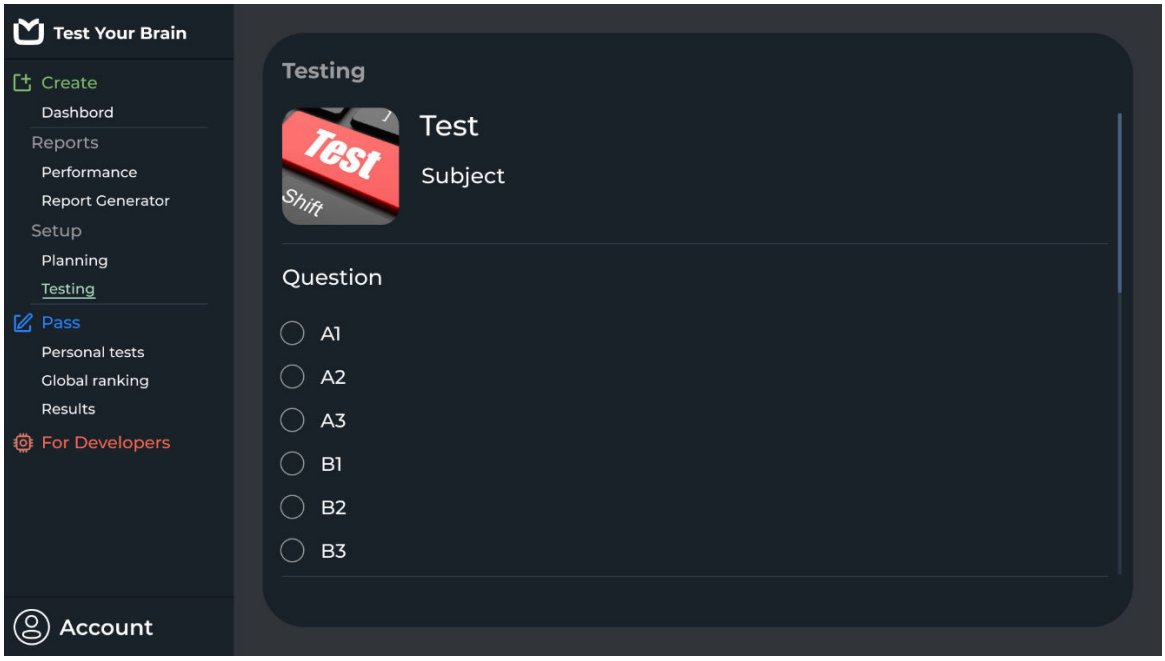

Рисунок 4.8 – Пробне тестування

Сторінка тесту ідентична сторінці проходження, але без збереження результатів тесту. Користувач відповідає на питання, обираючи варіанти, які він вважає правильними, після підтвердження відповідей формується результат тесту.

Після переходу на сторінку "Preview" у секції "Setup", розробник може вибрати потрібний тест і почати його проходження. Це дозволяє перевірити коректність заповнення тесту. Після відправки форми система показує результат, але не зберігає його в базі даних. Далі розробник переходить на сторінку "Planning", де система відображає список наявних тестів та їх налаштування за замовченням. Розробник може встановити значення параметрів для кожного тесту або залишити параметри за замовчуванням. Система зберігає зміни, і тест готовий до проходження учнями згідно з побажаннями розробника.

4.1.6 Сторінка персональних тестів

На цій сторінці відображено список тестів, які доступні для проходження користувачем (рисунок 4.9).

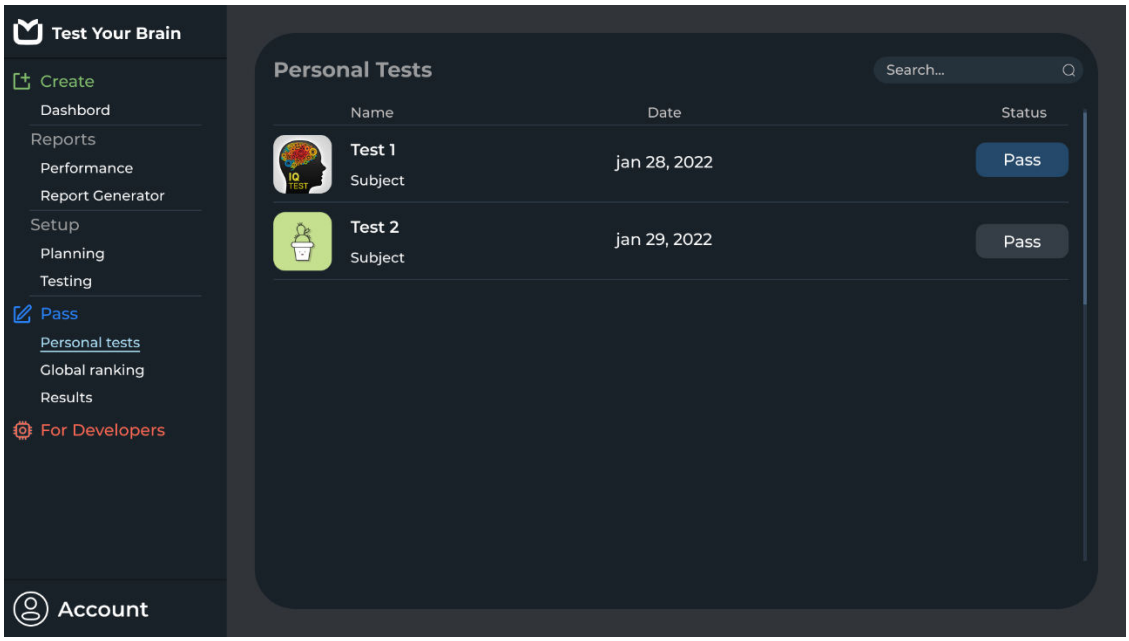

Рисунок 4.9 – Сторінка персональних тестів

Користувач бачить список тестів, які він матиме можливість пройти згодом, а також тести, які вже доступні для проходження.

Основний сценарій використання включає наступні кроки:

1. Користувач переходить на сторінку особистих тестів (Personal Tests).
2. Вибирає зі списку тест, який доступний для проходження.
3. Натискає на кнопку для початку тесту і переходить до його змісту.
4. Відповідає на питання тесту, навігуючи між ними.
5. Завершує тест і підтверджує відправку відповідей.

Система потім обчислює результат і зберігає його в базі даних. Користувач може переглянути отриманий результат, а інформація про дату і час проходження тесту та набрані бали заносяться на сторінку результатів (Results), де користувач може їх переглянути пізніше.

4.1.7 Сторінка результатів

Сторінка результатів надає користувачу інформацію про тести, які вони пройшли (рисунок 4.10).

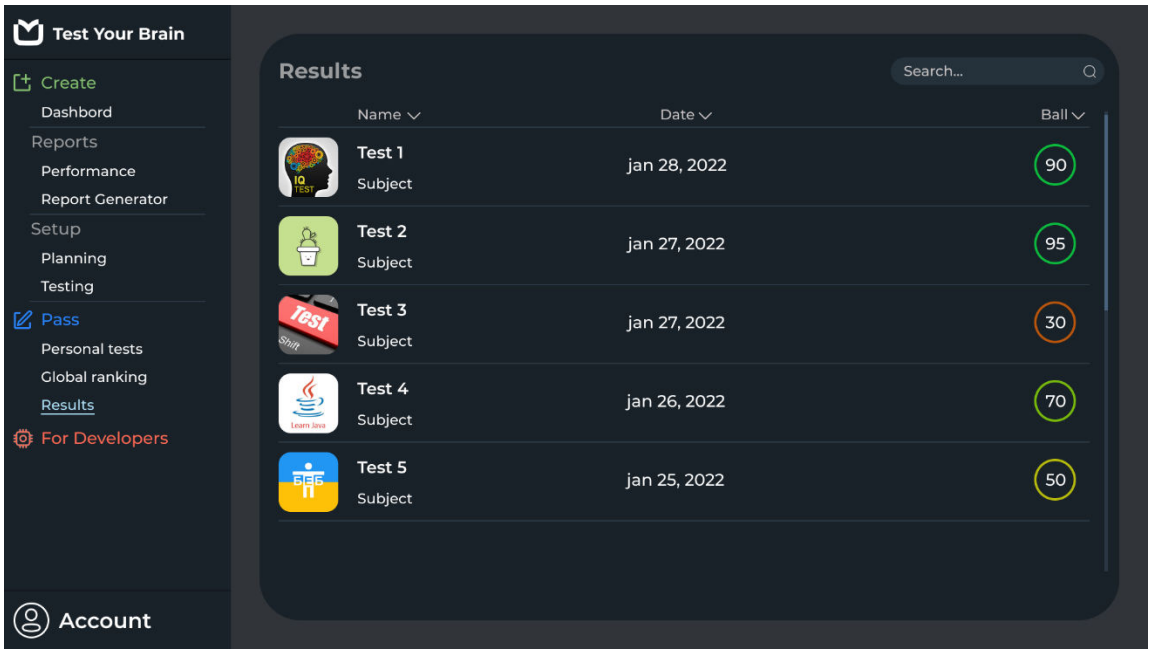

Рисунок 4.10 – Сторінка результатів

На цій сторінці міститься інформація про дату проходження та відсоток правильних відповідей для кожного тесту. Існує можливість сортування даних за назвою, датою та балом.

4.1.8 Сторінка глобального рейтингу

На цій сторінці відображається інформація про найуспішніших користувачів системи та список найпопулярніших тестів.

Вкладка «Топ Тестів» показує інформацію про тести, які були пройдені найбільшу кількість разів, та середній відсоток правильних відповідей від користувачів, які пройшли тест (рисунок 4.11).

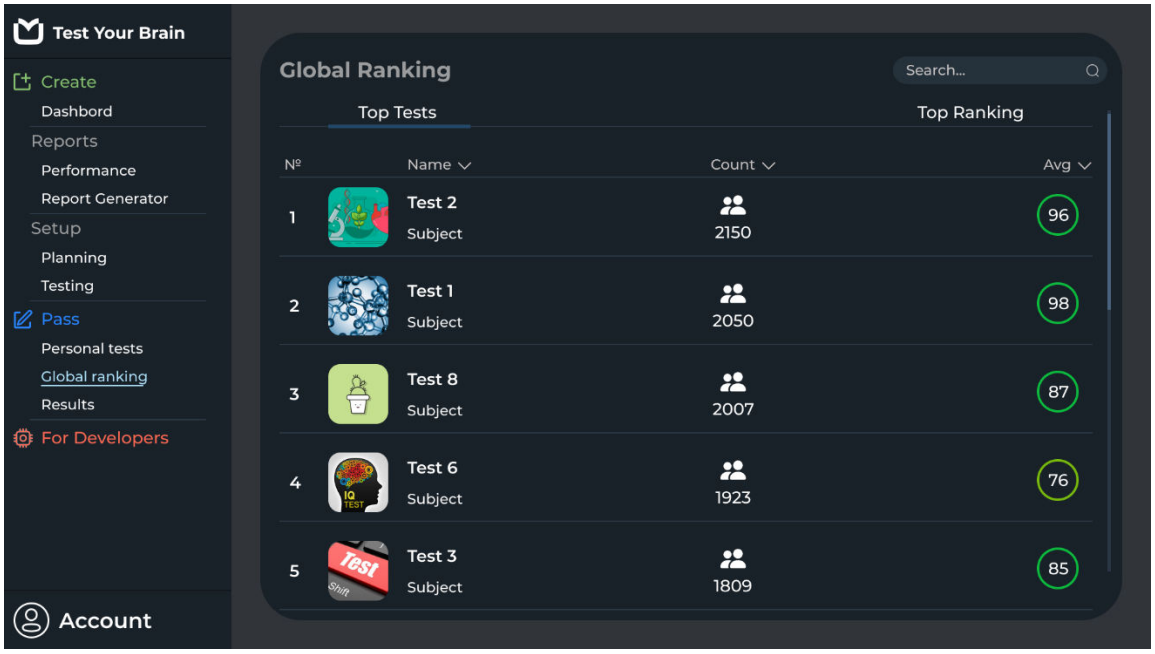

Рисунок 4.11 – Вкладка «Топ тестів»

Вкладка «Топ Рейтингу» відображає інформацію про користувачів, які пройшли найбільшу кількість тестів та тих, хто пройшов тести з максимальним відсотком правильних відповідей (рисунок 4.12).

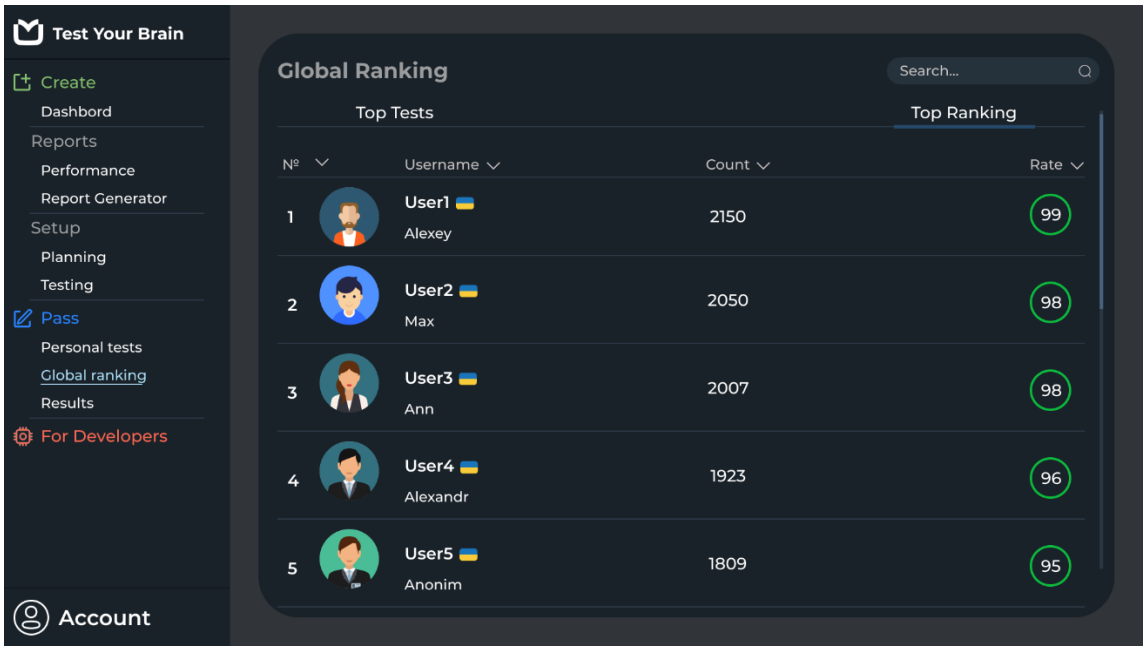

Рисунок 4.12 – Вкладка «Топ рейтингу»

Дані можуть бути відсортовані за кількістю пройдених тестів або за середнім балом. Таким чином, можна переглянути власне місце у глобальному

рейтингу користувачів, а також побачити, наскільки популярними є тести, створені користувачем, серед інших.

4.2 Аналітика та допомога користувачу засобами ШІ

4.2.1 Сторінка ефективності тесту

Сторінка ефективності тестів є ключовим елементом системи автоматизації тестування, що дозволяє авторам тестів аналізувати результати та вносити необхідні корективи для підвищення якості тестових матеріалів. Ця сторінка відображає загальний рейтинг кожного тесту, базуючись на агрегованих даних про успішність учасників, а також надає детальну статистику відповідей на кожне тестове питання (рисунок 4.13).

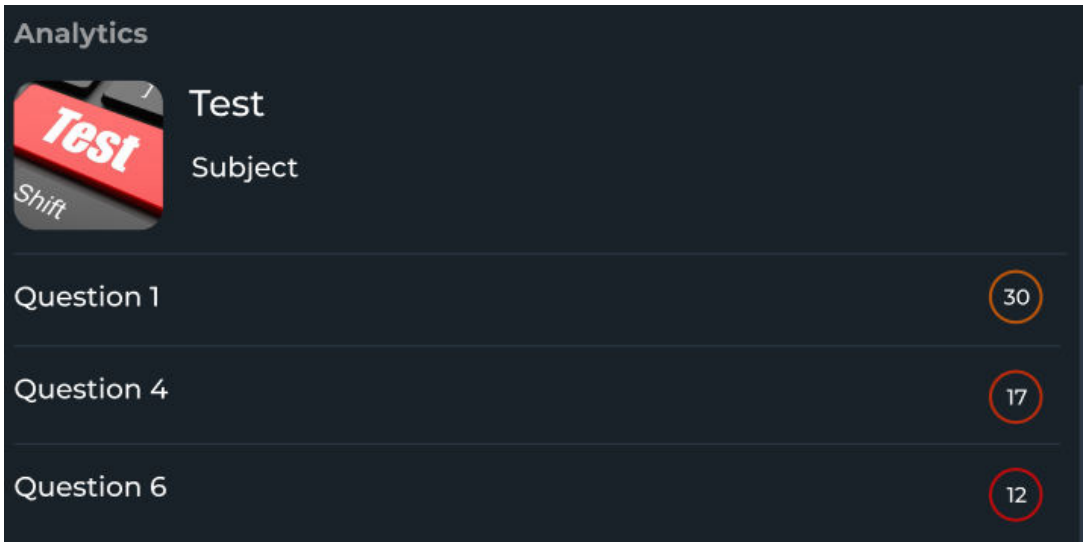

Рисунок 4.13 – Сторінка ефективності тесту

Основні функції сторінки включають:

– відображення загального рейтингу тесту: рейтинг визначається за середніми показниками правильних відповідей та може бути використаний для оцінки загальної складності тесту та його придатності для певної аудиторії;

– аналіз питань, що потребують перегляду: система виділяє питання, на які найчастіше давалися неправильні відповіді або питання, що викликали труднощі у великої кількості учасників, пропонуючи автору тесту переглянути формулювання цих питань або варіанти відповідей;

– рекомендації для покращення: на основі аналітики результатів, система може надавати конкретні рекомендації щодо покращення тесту, наприклад, зміни у розподілі складності питань, додавання пояснень до правильних відповідей або модифікації існуючих питань для кращого охоплення теми тесту.

Такий інструмент є незамінним для викладачів та розробників освітнього контенту, оскільки він не тільки допомагає підвищити якість навчальних матеріалів, але й сприяє адаптації тестів під конкретні потреби та рівень знань учнів.

4.2.2 Рекомендації ШІ при проєктувані тесту

Сторінка створення тесту є центральним елементом системи автоматизації тестування знань, де користувачі мають можливість конструювати власні тести. Головна увага на цій сторінці приділяється забезпеченню інтуїтивно зрозумілого і ефективного інтерфейсу, який сприяє легкості створення тестових завдань (рисунок 4.14).

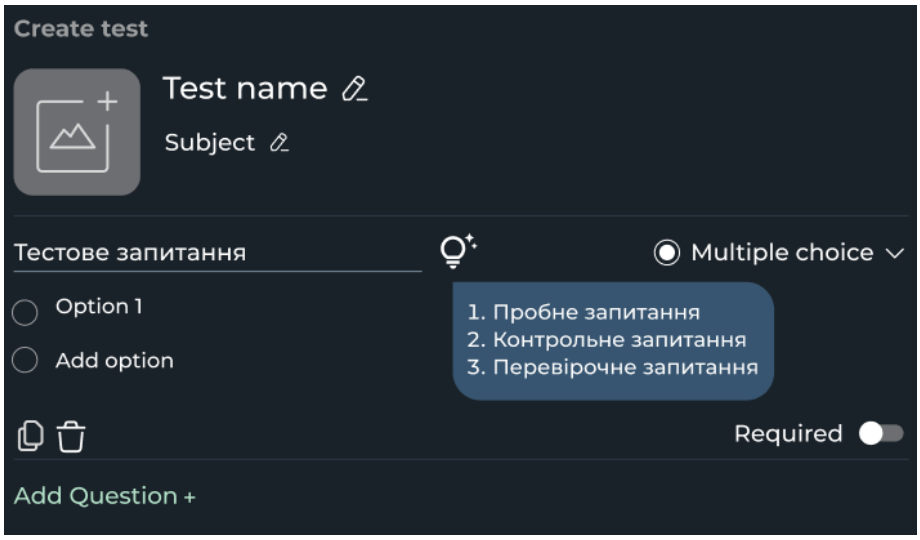

Рисунок 4.14 – Рекомендації ШІ

Рекомендації ШІ при проєктуванні тесту:

1. Аналіз попередніх тестів – ШІ аналізує результати попередніх тестів, виявляючи питання з високим рівнем помилок або нерозуміння, щоб рекомендувати їх перегляд або уточнення.

2. Рекомендації щодо формулювання питань – система надає поради щодо формулювання питань, щоб уникнути двозначностей та підвищити ясність завдань.

3. Баланс складності – на основі аналізу користувацьких відповідей, ШІ рекомендує оптимальний баланс складних, середніх та легких питань для створення збалансованого тесту.

4. Візуальні матеріали – ШІ може запропонувати додавання візуальних матеріалів, таких як діаграми або зображення, до питань для покращення зрозумілості та залученості.

5. Адаптивні Тести – рекомендації щодо створення адаптивних тестів, які налаштовуються під рівень знань користувача на основі його попередніх відповідей.

6. Підтримка різноманітності форматів – поради щодо включення різноманітних типів питань (один вірний варіант, кілька вірних, встановлення послідовності тощо) для збільшення ефективності навчального процесу.

7. Зворотний зв'язок – рекомендації щодо надання конструктивного зворотного зв'язку для неправильних відповідей, щоб сприяти навчанню та розумінню матеріалу.

Ця сторінка використовує передові можливості штучного інтелекту для того, щоб зробити процес створення тестів не тільки більш ефективним, але й інтуїтивно зрозумілим та адаптованим до потреб користувачів, що в кінцевому підсумку підвищує якість навчального процесу.

Висновки до розділу 4

У даному розділі увага приділена інтерфейсу та розробці посібника користувача для системи автоматизації тестування знань. Розглянуто ключові елементи інтерфейсу, включаючи процес авторизації, навігаційну панель, головну панель, панель створення тесту, панель статистики, розділ налаштувань, сторінку персональних тестів, сторінку результатів та сторінку

глобального рейтингу. Особлива увага приділена інтеграції модулів аналітики та допомоги користувачу з використанням технологій штучного інтелекту, що включають сторінку ефективності тесту та рекомендації ШІ при проєктуванні тестів.

ВИСНОВКИ

У результаті розробки інтегрованої системи автоматизованого тестування знань, що використовує алгоритми штучного інтелекту для оптимізації процесів підготовки та аналізу тестових завдань, було досягнуто значного прогресу в напрямку підвищення ефективності освітнього процесу. Автоматизація тестування знань є важливою складовою сучасного освітнього простору, що сприяє зменшенню трудомісткості процесу оцінювання, пришвидшенню обробки результатів та підвищенню їх точності.

В ході роботи були детально досліджені й аналізовані існуючі системи автоматизації тестування, виявлено їх переваги та недоліки, що дозволило визначити напрямки подальшого вдосконалення розроблюваної системи. Особлива увага була приділена розробці вимог до нової системи, моделюванню її архітектури та компонентів з використанням UML-діаграм, що забезпечило чітку структуру та зрозумілість процесів взаємодії в системі.

В ході написання кваліфікаційної роботи магістра були виконані наступні завдання:

– аналіз існуючих методів тестування;

– визначення вимог до системи;

– моделювання системи;

– розробка програмного забезпечення;

– тестування системи;

– розробка посібника користувача;

– інтеграція модуля аналітики;

– реалізація модуля зі штучним інтелектом.

Реалізація системи з клієнт-серверною архітектурою, інтеграція штучного інтелекту та модулів аналітики відкрили нові можливості для адаптивного тестування, забезпечення індивідуального підходу до оцінювання знань

студентів та оптимізації навчального матеріалу на основі аналізу результатів тестувань.

Висновки до кожного з розділів роботи підтверджують важливість та актуальність виконаних завдань, а також ефективність розробленої системи.

Успішне завершення проєктування та моделювання системи стало міцним фундаментом для її подальшої програмної реалізації, що включає розробку back-end та front-end частин, а також користувацького інтерфейсу, що забезпечує зручність та доступність використання системи для широкого кола користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Coronavirus. *World Health Organization (WHO)*. URL: https://www.who.int/health-topics/coronavirus (date of access: 15.02.2024).

2. MoodleDocs. *MoodleDocs*. URL: https://docs.moodle.org/403/en/Main_page (date of access: 15.02.2024).

3. Educational technology services | blackboard | north america. *Educational Technology Services | Blackboard | North America*. URL: https://www.blackboard.com/ (date of access: 15.02.2024).

4. Canvas by instructure | world's #1 teaching and learning software. *Instructure*. URL: https://www.instructure.com/canvas (date of access: 15.02.2024).

5. Google forms: online form creator | google workspace. *Google*. URL: https://www.google.com/forms/about/#features (date of access: 15.02.2024).

6. John M. Artificial intelligence. 2018. 311 p.

7. UML - statechart diagrams. *Online Tutorials, Courses, and eBooks Library | Tutorialspoint*.
URL: https://www.tutorialspoint.com/uml/uml_statechart_diagram.htm#:~:text=Statechart%20diagram%20is%20one%20of,to%20model%20the%20reactive%20systems. (date of access: 15.02.2024).

8. Bagui S., Earp R. Database design using entity-relationship diagrams. Auerbach Publications, 2011. URL: https://doi.org/10.1201/9781439861776 (date of access: 15.02.2024).

9. ASP.NET web apis | rest apis with .NET and C#. *Microsoft*. URL: https://dotnet.microsoft.com/en-us/apps/aspnet/apis (date of access: 15.02.2024).

10. Kimmel P. ADO.NET entity framework. Pearson Education, Limited, 2020. 400 p.

11. Manning L. Angular. Independently Published, 2018.

12. Loosely coupled monolith. *CodeOpinion*. URL: https://codeopinion.com/loosely-coupled-monolith/ (date of access: 15.02.2024).

13. Клієнт-серверна архітектура. *Головна сторінка*. URL: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (дата звернення: 15.02.2024).

14. Nirenburg S., McShane M. J. Natural language processing / ed. by S. E. F. Chipman. Oxford University Press, 2016. URL: https://doi.org/10.1093/oxfordhb/9780199842193.013.13 (date of access: 15.02.2024).

ДОДАТОК А

Оцінка дискримінаційної здатності

```
using System;
using System.Collections.Generic;
using System.Linq;

public class TestQuestion
{
    public int QuestionId { get; set; }
    public List<TestAttempt> Attempts { get; set; } = new List<TestAttempt>();
}

public class TestAttempt
{
    public int UserId { get; set; }
    public bool IsCorrect { get; set; }
}

public class QuestionAnalysis
{
    public int QuestionId { get; set; }
    public double DiscriminationIndex { get; set; }
}

public class TestAnalysis
{
    public List<QuestionAnalysis> AnalyzeQuestions(List<TestQuestion> questions)
    {
        var analyses = new List<QuestionAnalysis>();
```

```
        foreach (var question in questions)
        {
            var topGroup = question.Attempts
                .OrderByDescending(a => a.IsCorrect)
                .Take((int)(question.Attempts.Count * 0.27));

            var bottomGroup = question.Attempts
                .OrderBy(a => a.IsCorrect)
                .Take((int)(question.Attempts.Count * 0.27));

            var topCorrect = topGroup.Count(a => a.IsCorrect);
            var bottomCorrect = bottomGroup.Count(a => a.IsCorrect);

            var discriminationIndex = (double)(topCorrect -
bottomCorrect) / (topGroup.Count() + bottomGroup.Count());

            analyses.Add(new QuestionAnalysis
            {
                QuestionId = question.QuestionId,
                DiscriminationIndex = discriminationIndex
            });
        }

        return analyses;
    }
}
```

Аналіз варіативності відповідей

```
using System;
using System.Collections.Generic;
using System.Linq;

public class TestQuestion
{
```

```
    public int QuestionId { get; set; }
    public List<TestAnswer> Answers { get; set; } = new List<TestAnswer>();
}

public class TestAnswer
{
    public string Answer { get; set; }
    public int Count { get; set; }
}

public class VariabilityAnalysis
{
    public int QuestionId { get; set; }
    public Dictionary<string, int> AnswerDistribution { get; set; } = new Dictionary<string, int>();
}

public class TestVariabilityAnalysis
{
    public List<VariabilityAnalysis> AnalyzeAnswerVariability(List<TestQuestion> questions)
    {
        var analyses = new List<VariabilityAnalysis>();

        foreach (var question in questions)
        {
            var analysis = new VariabilityAnalysis { QuestionId = question.QuestionId };

            foreach (var answer in question.Answers)
            {
```

```
                if
(analysis.AnswerDistribution.ContainsKey(answer.Answer))
                {
                    analysis.AnswerDistribution[answer.Answer]         +=
answer.Count;
                }
                else
                {
                    analysis.AnswerDistribution.Add(answer.Answer,
answer.Count);
                }
            }

            analyses.Add(analysis);
        }

        return analyses;
    }
}
```

Оцінка часу, витраченого на кожне питання

```
using System;
using System.Collections.Generic;
using System.Linq;

public class TestQuestion
{
    public int QuestionId { get; set; }
    public   List<QuestionAttempt>   Attempts   {   get;   set;   }   =   new
List<QuestionAttempt>();
}

public class QuestionAttempt
{
```

```
    public int StudentId { get; set; }
    public TimeSpan TimeSpent { get; set; }
}

public class QuestionTimeAnalysis
{
    public int QuestionId { get; set; }
    public TimeSpan AverageTimeSpent { get; set; }
}

public class TestTimeAnalysis
{
    public List<QuestionTimeAnalysis> AnalyzeTimeSpent(List<TestQuestion>
questions)
    {
        var analyses = new List<QuestionTimeAnalysis>();

        foreach (var question in questions)
        {
            if (question.Attempts.Count > 0)
            {
                var                    averageTimeSpent                    =
TimeSpan.FromTicks((long)question.Attempts.Average(a                    =>
a.TimeSpent.Ticks));

                analyses.Add(new QuestionTimeAnalysis
                {
                    QuestionId = question.QuestionId,
                    AverageTimeSpent = averageTimeSpent
                });
            }
        }
```

```
        return analyses;
    }
}
```

ШІ для підтримки формулювання питань тестів

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Microsoft.Azure.CognitiveServices.Language.TextAnalytics;
using Microsoft.Azure.CognitiveServices.Language.TextAnalytics.Models;

public class QuestionGenerator
{
    private ITextAnalyticsClient textAnalyticsClient;

    public QuestionGenerator(string subscriptionKey, string endpoint)
    {
        textAnalyticsClient = new TextAnalyticsClient(new
ApiKeyServiceClientCredentials(subscriptionKey))
        {
            Endpoint = endpoint
        };
    }

    public async Task<List<string>> GenerateQuestionSuggestions(string
content)
    {
        var keyPhrases = await ExtractKeyPhrases(content);
        var questions = new List<string>();

        foreach (var phrase in keyPhrases)
        {
            questions.Add($"What do you know about {phrase}?");
            questions.Add($"Explain the concept of {phrase}.");
```

```
            questions.Add($"Describe how {phrase} is used in context.");
        }

        return questions;
    }

    private async Task<IList<string>> ExtractKeyPhrases(string content)
    {
        var result = await textAnalyticsClient.KeyPhrasesAsync(false, new
MultiLanguageBatchInput(
            new List<MultiLanguageInput>
            {
                new MultiLanguageInput("en", "1", content)
            }));

        return result.Documents[0].KeyPhrases;
    }
}

public class TeacherInterface
{
    public async Task ShowQuestionSuggestions(Question question)
    {
        var  generator  =  new  QuestionGenerator("YourSubscriptionKey",
"YourEndpoint");
        var content = question.Text
        var               suggestions               =               await
generator.GenerateQuestionSuggestions(content);

        foreach (var suggestion in suggestions)
        {
            Console.WriteLine(suggestion);
        }
```

```
    }}
```

ДОДАТОК Б

Проєкт User

Users.Models

```
using System;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace Users.Models
{
    public class User
    {
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; } = default;
        public string ExternalId { get; set; } = default;
        public string Email { get; set; } = default;
        public string Login { get; set; } = default;
        public string Password { get; set; } = default;
        public string FirstName { get; set; } = default;
        public string LastName { get; set; } = default;
        public string Country { get; set; } = default;
        public string Avatar { get; set; } = default;

        public User() { }

        public User(string login, string password)
        {
            if (!IsLoginValid(login))
                throw new ArgumentException($"Parameter {nameof(login)}
passed into {nameof(User)} constructor is null or empty.");

            if (!IsPasswordValid(password))
```

```
                throw new ArgumentException($"Parameter {nameof(password)} passed into {nameof(User)} constructor is null or empty.");

            Login = login;
            Password = password;
            ExternalId = Guid.NewGuid().ToString();
        }

        public User(string login, string password, string externalId) : this(login, password)
        {
            if (!IsExternalIdValid(externalId))
                throw new ArgumentException($"Parameter {nameof(externalId)} passed into {nameof(User)} constructor is null or empty.");

            ExternalId = externalId;
        }

        public User(int id, string login, string password, string externalId) : this(login, password, externalId)
        {
            Id = id;
            FirstName = "";
            LastName = "";
            Country = "";
            Avatar = "";
            Email = "";
        }

        private static bool IsLoginValid(string login) => !string.IsNullOrWhiteSpace(login);
```

```
        private static bool IsPasswordValid(string password) =>
!string.IsNullOrWhiteSpace(password);
        private static bool IsExternalIdValid(string externalId) =>
!string.IsNullOrWhiteSpace(externalId);

    }
}
```

Users.Controllers

```
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;
using Users.Repository;
using Users.Models;
using Microsoft.AspNetCore.Authorization;
using System.Security.Claims;
using System.IdentityModel.Tokens.Jwt;
using Microsoft.IdentityModel.Tokens;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Configuration;
using System.Security.Cryptography;

namespace Users.Controllers
{
    [ApiVersion("1.0")]
    //[Authorize]
    [ApiController]
    [Route("users-management")]
    public class UsersController : ControllerBase
    {
        private readonly IConfiguration configuration;
        private readonly IUsersRepository usersRepository;
```

```
        public UsersController(IConfiguration configuration,
IUsersRepository usersRepository)
        {
            this.configuration = configuration;
            this.usersRepository = usersRepository;
        }

        [HttpGet]
        [ApiConventionMethod(typeof(DefaultApiConventions),
nameof(DefaultApiConventions.Get))]
        [AllowAnonymous]
        [Route("version")]
        [ProducesResponseType(StatusCodes.Status200OK, Type =
typeof(string))]
        public ActionResult<string> Version(ApiVersion apiVersion) =>
Ok($"Active {nameof(UsersController)} API ver is {apiVersion}");

        [HttpGet]
        [ApiConventionMethod(typeof(DefaultApiConventions),
nameof(DefaultApiConventions.Get))]
        [Route("login")]
        [AllowAnonymous]
        [ProducesResponseType(StatusCodes.Status200OK, Type =
typeof(Token))]
        [ProducesResponseType(StatusCodes.Status400BadRequest)]
        [ProducesResponseType(StatusCodes.Status500InternalServerError)]
        public async Task<ActionResult<Token>> Login([FromQuery] string
email, [FromQuery] string password)
        {
            User user;
            ClaimsIdentity identity;
            try
```

```
            {
                user = await usersRepository.GetUserBy(email, password);
                identity = GetIdentity(user);
            }
            catch (Exception ex)
            {
                return        StatusCode(StatusCodes.Status400BadRequest,
ex.Message);
            }

            string issuer;
            string audience;
            DateTime now;
            int lifetime;
            SymmetricSecurityKey symmetricSecurityKey;

            try
            {
                IConfigurationSection           tokenOptions           =
configuration.GetSection("TokenOptions");
                issuer = tokenOptions.GetSection("issuer").Value;
                audience = tokenOptions.GetSection("audience").Value;
                now = DateTime.UtcNow;
                lifetime = int.Parse(tokenOptions["lifetime"]);
                string key = tokenOptions["key"];
                symmetricSecurityKey                =                new
SymmetricSecurityKey(Encoding.ASCII.GetBytes(key));

            }
            catch (Exception)
            {
```

```
                return
StatusCode(StatusCodes.Status500InternalServerError,     "Error     while
creating access token");
            }

            JwtSecurityToken jwt = new JwtSecurityToken(
                    issuer: issuer,
                    audience: audience,
                    notBefore: now,
                    claims: identity.Claims,
                    expires: now.Add(TimeSpan.FromMinutes(lifetime)),
                    signingCredentials:                                  new
SigningCredentials(symmetricSecurityKey, SecurityAlgorithms.HmacSha256));

            string              encodedJwt              =              new
JwtSecurityTokenHandler().WriteToken(jwt);

            Token  token  =  new  Token(identity.Name,  user.ExternalId,
encodedJwt);
            return Ok(token);
        }

        [NonAction]
        private ClaimsIdentity GetIdentity(User user)
        {
            List<Claim> claims = new List<Claim>
            {
                new            Claim(ClaimsIdentity.DefaultNameClaimType,
user.Login),
                new Claim("UserExternalId", user.ExternalId),
                new Claim(ClaimsIdentity.DefaultRoleClaimType, "Default")
            };
```

```
            ClaimsIdentity claimsIdentity = new ClaimsIdentity(
                claims,
                "Token",
                ClaimsIdentity.DefaultNameClaimType,
                ClaimsIdentity.DefaultRoleClaimType
                );

            return claimsIdentity;
        }

        [HttpPost]
        [ApiConventionMethod(typeof(DefaultApiConventions),
nameof(DefaultApiConventions.Post))]
        [Route("users/signup")]
        [AllowAnonymous]
        [ProducesResponseType(StatusCodes.Status200OK,       Type       =
typeof(User))]
        [ProducesResponseType(StatusCodes.Status400BadRequest)]
        [ProducesResponseType(StatusCodes.Status500InternalServerError)]
        public async Task<ActionResult<User>> SignUp(User user)
        {
            user.ExternalId = Guid.NewGuid().ToString();
            user.Password = HashPassword(user.Password);
            try
            {
                await usersRepository.Insert(user);
                await usersRepository.SaveChanges();
            }
            catch (Exception ex)
            {
                return
StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
            }
```

```
            return Ok(user);
        }

        [HttpGet]
        [ApiConventionMethod(typeof(DefaultApiConventions),
nameof(DefaultApiConventions.Get))]
        [Route("users")]
        [AllowAnonymous]
        [ProducesResponseType(StatusCodes.Status200OK,       Type       =
typeof(IEnumerable<User>))]
        [ProducesResponseType(StatusCodes.Status400BadRequest)]
        public async Task<ActionResult<IEnumerable<User>>> GetUsers()
        {
            IEnumerable<User> users;
            try
            {
                users = await usersRepository.GetUsers();
            }
            catch (Exception ex)
            {
                return        StatusCode(StatusCodes.Status400BadRequest,
ex.Message);
            }

            return Ok(users);
        }

        [NonAction]
        private string HashPassword(string password)
        {
            byte[] salt;
            byte[] buffer2;
            if (password == null)
```

```
            {
                throw new ArgumentNullException("password");
            }
            using (Rfc2898DeriveBytes bytes = new
Rfc2898DeriveBytes(password, 0x10, 0x3e8))
            {
                salt = bytes.Salt;
                buffer2 = bytes.GetBytes(0x20);
            }
            byte[] dst = new byte[0x31];
            Buffer.BlockCopy(salt, 0, dst, 1, 0x10);
            Buffer.BlockCopy(buffer2, 0, dst, 0x11, 0x20);
            return Convert.ToBase64String(dst);
        }

        [HttpDelete]
        [ApiConventionMethod(typeof(DefaultApiConventions),
nameof(DefaultApiConventions.Delete))]
        [Route("users/{userid}")]
        [AllowAnonymous]
        [ProducesResponseType(StatusCodes.Status200OK, Type =
typeof(User))]
        [ProducesResponseType(StatusCodes.Status400BadRequest)]
        public async Task<ActionResult<User>> DeleteUser(string userid)
        {
            User user;
            try
            {
                user = await usersRepository.GetUserBy(userid);
            }
            catch (Exception ex)
            {
```

```
                return        StatusCode(StatusCodes.Status400BadRequest,
ex.Message);
            }

            try
            {
                usersRepository.Delete(user);
                await usersRepository.SaveChanges();
            }
            catch (Exception ex)
            {
                return
StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
            }

            return Ok(user);
        }
    }
}
```